Air-writing Recognition, Part 2: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data

Mingyu Chen, Ghassan AlRegib, *Senior Member, IEEE,* and Biing-Hwang Juang, *Fellow, IEEE*

***Abstract*—Air-writing refers to writing of characters or words in the free space by hand or finger movements. We address airwriting recognition problems in two companion papers. Part 2 addresses detecting and recognizing air-writing activities that are embedded in a continuous motion trajectory without delimitation. Detection of intended writing activities amongst superfluous finger movements unrelated to letters or words presents a challenge that needs to be treated separately from the traditional problem of pattern recognition.**

We first present a data set that contains a mixture of writing and non-writing finger motions in each recording. The LEAP from Leap Motion is used for marker-free and glove-free finger tracking. We propose a window-based approach that automatically detects and extracts the air-writing event in a continuous stream of motion data, containing stray finger movements unrelated to writing. Consecutive writing events are converted into a writing segment. The recognition performance is further evaluated based on the detected writing segment. Our main contribution is to build an air-writing system encompassing both detection and recognition stages and to give insights into how the detected writing segments affect the recognition result. With leave-one-out cross validation, the proposed system achieves an overall segment error rate (SER) of 1.15% for word-based recognition and 9.84% for letter-based recognition.

***Keywords*—air-writing, fingerwriting, air-writing recognition, air-writing detection.**

I. Introduction

WRITING with a finger on a touch-based interface is intuitive because it follows the metaphor of pen-based writing. Recent advances of tracking technology make it possible to track hand and finger motions without user-worn devices, and writing motion is no longer restricted on a physical plane. Air-writing provides a viable alternative interface for text input, particularly when conventional input devices, such as a keyboard or a mouse, are not available or suitable. Compared to other non-traditional input methods such as typing with a virtual keyboard or similar schemes, air-writing offers the advantage of "eye-free" execution, requiring minimum attention focus [1].

When we write with a fingertip in the air and use a controller-free tracking system such as LEAP [2] to track the finger motion, the motion data comprises every bit of the finger movement in an uninterrupted stream, writing or drifting, and the intended writing activity is no longer easily or explicitly located. Thus, the detection and extraction of the writing signal from the continuous motion data stream is challenging. With the finger-precision tracking of the Leap device, the user can write in the air easily with his or her fingertip. To make the Leap a viable writing interface, nevertheless, an intelligent system that is capable of handling both detection and recognition of the air-writing mixed with other stray movements must be designed. Although some specific finger movements can used as in-line delimiter signals to provide endpoint information for a writing activity, writing with these explicit delimiters hinders the user experience of air-writing. In this work, we propose a system that automatically detects, segments, and recognizes the writing part from the continuous motion tracking signal.

In this work, we propose an algorithm that solves the problem of detecting air-writing from general finger movements on a motion-based user interface. Finger motion in the air contains a fair amount of stray movements unrelated to and often inseparable from the intended letters or words. To handle the potentially imprecise detection segments, we re-design the recognition kernel of [1]. We also evaluate the air-writing recognition performance based on the detected writing trajectories to investigate the overall system performance, end-to-end from writing activity detection to recognition. An overview of the finger air-writing system is shown in Figure 1. At the detection stage, the continuous motion data are converted to individually detected writing segments. At the recognition stage, the detected segments are processed for the final result, either a recognition decision or a rejection.

The paper is organized as follows. In Section II, we discuss the related work of motion tracking, handwriting recognition, and the segmentation issues. In Section III, we describe the motion tracking system, the Leap, and data recording procedures for air-writing. Section IV is devoted to air-writing detection. In Section V, we evaluate the recognition performance, and Section VI concludes this paper.

II. Related Work

Both the speed and the precision of hand tracking are critical for interactive applications. Glove-based tracking has been proposed to support hand tracking [3]. However, putting on a glove may be cumbersome and uncomfortable for a long session of interaction. Controller-free motion tracking is believed to bring about the most natural user experience. In [4], two consumer-grade webcams are used to achieve bimanual

M. Chen, G. AlRegib, and B-H. Juang are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA.

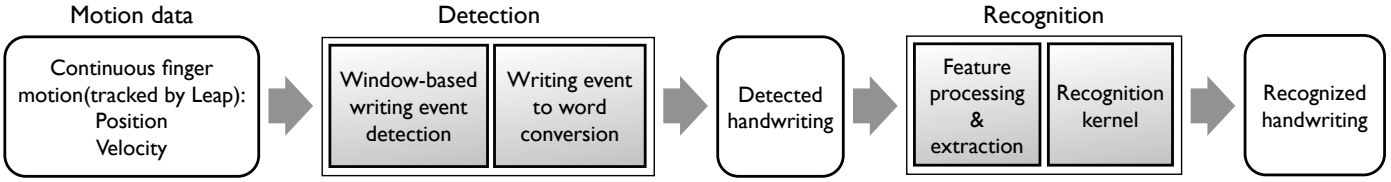


Fig. 1. The system diagram of controller-free air-writing detection and recognition

6-DOF pose estimation at interactive rates for reliable pose detection, such as pinching and pointing. This technology can track the user's hands to finger-precision and is further improved to millimeter-level accuracy with 3D cameras such as Kinect. Leap [2], a small USB peripheral device, is designed to track fingers (or stick-like objects such as a pen or a chopstick) precisely in a desktop environment. The smaller tracking volume and higher resolution of Leap differentiates itself from Kinect, which is designed for body and face tracking in a livingroom-like environment.

Online handwriting recognition addresses the problem from a spatio-temporal point of view, i.e., looking at the writing trajectory instead of the shape [5]. Hidden Markov models (HMM) are especially known for their application in spatio-temporal pattern recognition, including online handwriting recognition [6], [7]. Cursive handwriting contains successive letters that are connected without explicit pen-up moves. Sin and Kim [8] used ligature models to handle the inter-letter patterns for online cursive handwriting recognition. The existing online handwriting data, such as UNIPEN [9], are mostly collected by pen-based devices, which track the 2D trajectory with the pen-up/pen-down information. When people write in the air, different types of tracking devices are needed, e.g., a vision-based hand tracking system [10], inertial sensors attached to a glove [11], [12], or a depth sensor [13].

For pen-based or touch-based writing, the ink information is directly included when writing. The pen-up/pen-down moves naturally delimit the strokes for print writing or segment the word boundaries for cursive writing. For air-writing, the motion is tracked with a continuous stream of sensor data, which means the writing is uni-stroke with no engagement information. In such a case, delimitation can be accomplished with explicit segmentation (push-to-write) or automatic detection (spotting). Explicit segmentation can be achieved with buttons, where the user holds a button to write and releases it to stop [1], [14]. When buttons are not available as in a controller-free system, one alternative is to use a specific posture or gesture to signal the endpoints of a handwriting, e.g., pinch-to-write. There are other approaches for explicit segmentation. Kristensson et al. [15] proposed an input zone for gesture delimitation with Kinect. In [10], a user reaches out to write in the air, and the segmentation is done by thresholding the depth information.

In sum, the approaches that require preambles in any forms to signal engagement are still considered as explicit segmentation. On the other hand, automatic detection of handwriting requires no intentional delimitation and can make the air-writing experience more convenient with controller-free systems. Amma et al. [12] proposed a spotting algorithm for air-writing based on the acceleration and angular speed from inertial sensors attached to a glove and reported a recall of 99% and a low precision of 25% for air-writing spotting. Nonetheless, the recognition performance was evaluated on manually segmented writing instead of the automatically detected segments.

III. Air-writing and Data Recording

Air-writing involves no physical plane to write on and provides no haptic feedback or visualized writing trajectory. We use the box-writing style as presented in [1], i.e., the user writes a uni-stroke word with each letter overlaid in the same virtual box. Compared to the ordinary left-to-right writing, the box-writing style reduces the range of the hand motion and produces less fatigue on the user's part.

The current work differs from [1] on two accounts. First, the writing is rendered with a fingertip instead of a handheld device. Motion acquisition in this work is done by the Leap, which provides motion data of different characteristics from the air-writing data obtained with a hand-held controller. We call this "controller-free air-writing". Second, the push-to-write paradigm is no longer applicable because there is no button for the user to signal the beginning and ending of writing. The motion trajectory will inevitably include both legitimate writing segments and stray parts unrelated to writing, a situation not encountered in [1]. The legitimate writing segments must be detected and separated from those stray ones before effective recognition of letters or words can be reliably accomplished.

In this work, we only consider uppercase letters A to Z with a specified stroke order for each letter as in [1]. The ultimate goal of air-writing recognition should include lowercase letters, allographs, and different stroke orders. In short, these variations all result in different spatio-temporal patterns and need extra recordings to be modeled separately. Hence, we start solving the air-fingerwriting problem in a simplified form without loss of generality.

A. Hand and Finger Tracking with the Leap

For controller-free and glove-free hand tracking, we use the Leap, which advertises a tracking precision of 0.01 mm. With our system setup (Intel core i7 CPU 2.66 GHz, Leap SDK 0.7.4 with USB 2.0 connection), Leap can track at a rate of 120 Hz. We place the Leap roughly 20 cm in front of the monitor so that the tracking volume covers the range of the hand movements with the elbow resting on the desktop. The tracking coordinates of the Leap (see Figure 2) are aligned to

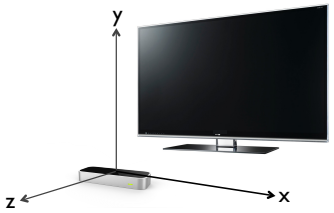


Fig. 2. Leap experiment setup

the monitor with x- and y-axes lying in the horizontal plane parallel to the screen. The positive x-axis points rightward, the positive y-axis points upward, and the positive z-axis points away from the screen. The origin is at the center of the Leap device. The Leap emits infrared light and tracks the fingers (or stick-like objects) with two infrared cameras. We do not place any infrared light sources in the field of view of the Leap to achieve its best tracking performance.

The Leap SDK produces samples of position, velocity, and pointing direction of the "pointables", i.e., stick-like objects, within its view. We use the relatively stable tracking results, i.e., the position and the velocity of the tip of a pointable. Because the Leap can only identify the fingers by the tracking history, we assign the finger closest to the screen as the pointing finger when loss of tracking occurs. Most of the time the tracking of the pointing finger is stable. The position of the pointing finger on the xy-plane is offset and linearly scaled to the pixel position of the cursor on screen, e.g., a finger movement of 32 cm corresponds to a movement of 1920 pixel on screen. To complete the basic functionalities of a 2D user interface, we implement clicking with a gesture of closing the thumb while the index finger points at the target. With our design, the user can use the fingers to do simple point-and-click task similar to using a mouse.

B. Data Recording

First, we create a $1k$-word vocabulary, which includes the most frequent 1000 two-, three-, and four-letter words and four-letter prefixes from the Google Web 1T data set [16]. Among the 1000 words, we carefully select 100 words as the common set, which covers 26 letters and 21 ligature types defined in [1]. The remaining 900 words are shuffled and divided into 18 sets of 50 unique words. The special distribution of vocabulary helps us to evaluate how user dependency and out-of-vocabulary words affect the recognition with limited user data as described in Section V-B.

For data recording, we built a user interface that overrides the mouse cursor control with index-finger motion for pointing and pinch gesture for clicking. This interface updates the tracking results from the Leap at 60 Hz, which shows no discernible delay in both writing and pointing-and-clicking operations. In such a framework, ordinary control motions (of the index finger) are usually: 1) idle or slow swaying motion, 2) reach out to a target for clicking. Non-writing motions are not meant to be any arbitrary motions that are unlikely to show up in this kind of UI. We would like to distinguish between non-writing "control" motions and air-writing motions. Therefore, our design of data recording is to mix non-writing control motions with air-writing.

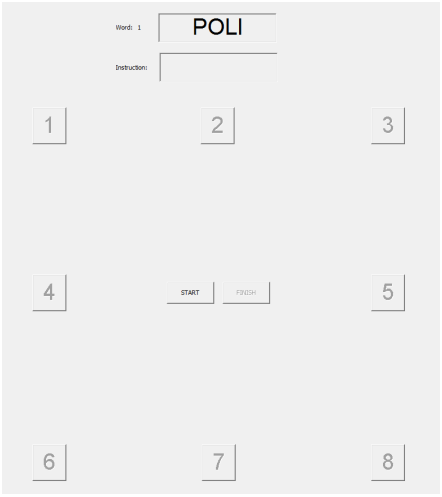


Fig. 3. A screenshot of the recording program

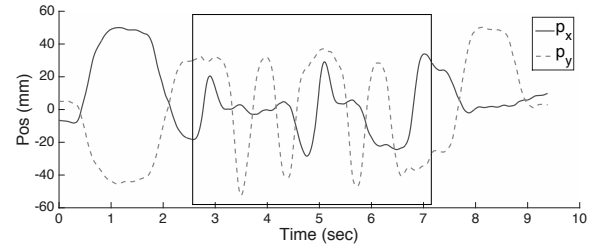
The recording program displays the word to write in a text box and has several buttons: `START` and `FINISH` in the center, and eight buttons, numbered from one to eight, around the center in a 3-by-3 grid. A screenshot of the recording program is shown in Figure 3. The recording procedure is as follows:

1) Click `START` to start recording.
2) Click one numbered button, which is randomly enabled after clicking `START`.
3) Write the prompted word while pressing the `Ctrl` key.
4) Click one numbered button, which is randomly enabled after writing (release `Ctrl` key).
5) Click `FINISH` to stop recording.

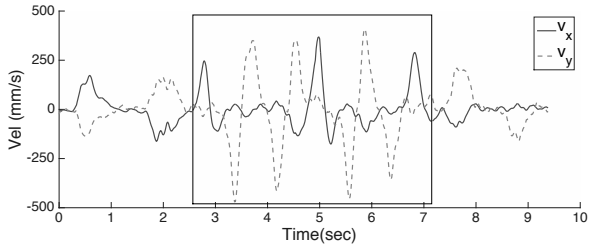
Clicking the randomly enabled buttons in Steps 2 and 4 introduces random cursor movements in each recording. Each recording contains exactly a motion word with random motions before and after the writing, and the subject needs to repeat if the procedure is not followed. In a continuous stream of motion data, the air-writing motion is accompanied by non-writing random motions. A writing segment is the part of the motion data stream when the writing activity occurs. Pressing the `Ctrl` key (with the non-writing hand) labels the ground truth of writing segments. Note that the "ground truth" itself may still contain imprecise segmentation due to the key operation. Thus, stray trajectories remain in the data to allow the study of writing activity detection.

In addition to the recording procedure, we ask each subject to write in a consistent way with the box-writing style and the specific stroke order for each letter. The writing position, scale, and speed are not constrained.

We recruited 18 participants (all right-handed, 13 male and 5 female) to record air-fingerwriting data. Each subject wrote 150 words, which consists of the common set and one unique set. With 18 subjects, we collected a total of 2700 recordings that cover the $1k$-word vocabulary. This dataset is published

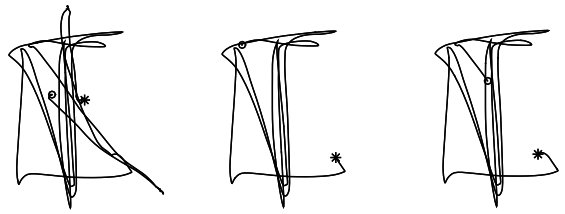


(a) position (p_x, p_y)



(b) velocity (v_x, v_y)

Fig. 4. The recording `TITL` by subject `C1`. The trajectories inside the box (from 2.6 to 7.2 sec) are the "ground-truth" writing segment.



(a) complete recording (b) writing segment (ground truth) (c) writing segment (detection)

Fig. 5. The 2D trajectory of the recording `TITL` by subject `C1`. The circle sign is the start, and the star sign is the end.

online with the complete vocabulary and video demo of air-fingerwriting[1].

As an example, we plot the position and velocity in the xy-plane over time of the recording `TITL` by subject `C1` with the ground-truth label in Figure 4. We also show a 2D trajectory of a complete recording of the same recording in Figure 5a and the ground-truth writing segment in Figure 5b. Both position and velocity are smoothed with a 5-point moving average, and we offset the position in y-axis to zero mean in Figure 4a for illustration purposes. In Figure 4, the signals of the writing part are different from those of the non-writing part with more frequent change with time, which sheds some light on solving the detection problem.

IV. Air-writing Detection

As shown in Figure 1, it is essential to know where the writing segment starts and ends in the stream of motion data before recognition. We propose an approach for air-writing detection that automatically extracts the writing segment (if any) without any explicit delimiters. Ideally, the detection algorithm should spot all writing segments (high recall), produce only a small amount of false alarms (high precision), and impose a minimal delay in the processing pipeline.

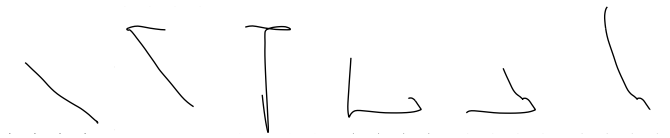
Our first attempt to solve the detection problem is based on a quick distance-based classification of air-writing letters. The segmented writing curves of each character is converted to its own Legendre series representation [17] to form a template. Given a windowed writing curve, we can compute its Legendre series representation and apply distance-based nearest neighbor classification [18]. For the detection task, we slide multiple windows of different lengths through the motion data and compute the corresponding Legendre coefficients. If there is a good match, we then spot a writing character. Classification based on the Legendre coefficients works quite well if the sliding window matches the correct letter segmentation. However, the Legendre coefficients are sensitive to the span of the sliding window even with an offset of five samples (1/12 seconds) in our case. The classification is not robust in most cases where the sliding window does not correctly segment a letter. Therefore, letter boundary detection based on Legendre coefficients does not work for air-fingerwriting. Human visual cognitive capabilities are better. People can extract and recognize a meaningful pattern embedded in a scribble. This inherently scale-invariant and rotation-invariant capability is extremely difficult for a machine to duplicate for cognitive tasks.

A. Window-based Approach and Algorithm

The approach of quick letter classification as discussed fails to generate robust results for air-fingerwriting detection. Thus, it is necessary to reformulate the problem as follows. We take a sliding window approach and detect if there is a writing activity within the observation window. The detector is only responsible for the detection of whether a writing event occurs in the window, which is slided through the continuous motion data. After proper determination of the successive writing events, we combine the detection results of all overlapping windows and pass them to the recognizer.

This reformulated approach is based on the observation that a motion trajectory of writing typically contains signs of contriving effort, in contrast to those relaxing or relenting movements. In particular, a writing event usually involves sharp turns, frequent changes in directions, and complicated shapes rather than a drift or stray motion. The sliding window has to be long enough to capture these writing characteristics to distinguish a writing event. However, a longer window means lower temporal resolution and introduces larger delay in the processing pipeline. In this work, we empirically choose a window length of 60 samples (1 sec) with a step size of 10 samples (167 ms). Before the window-based detection, we have to smooth the motion data from the Leap with a 5-point moving average to remove jitters. In Figure 6, we show the 2D trajectories of several sliding windows. The windows in Figure 6c and 6d contain writing events, and the windows in

[1]The air-fingerwriting data is available at http://www.ece.gatech.edu/6DMG



(a) i_s=021, w_c=0, ll=-65 (b) i_s=111, w_c=16, ll=-1.4 (c) i_s=161, w_c=60, ll=10.98 (d) i_s=381, w_c=50, ll=3.58 (e) i_s=401, w_c=30, ll=0.35 (f) i_s=421, w_c=10, ll=-4.29

Fig. 6. The 2D trajectory of selected sliding windows from `TITL` by subject `C1`, where i_s denotes the first sample index of the window, w_c denotes the count of samples that are labeled as ground-truth writing, and ll denotes the log likelihood of the writing event classification.

Figure 6a and 6f do not. The windows in Figure 6b and 6e are ambiguous with partial writing and non-writing events.

B. Writing Event Detection

It is straightforward to determine a window that contains a tiny motion as a non-writing event. We define an "idling" window as *a*) both b_x and b_y, the edges of the bounding box of p_x and p_y, are smaller than 10 mm; and *b*) the velocity is smaller than 50 mm/s.

After suppression of "idling movements", we extract features from a non-idling window, which can be derived as follows:

$$f_1 = \sum |\Delta\theta_i| \tag{1}$$

$$f_2 = \sum |\Delta\theta_i|^2 \tag{2}$$

$$f_3 = b_x/(b_x + b_y) \tag{3}$$

$$f_4 = ||\text{total travel distance}/\max(b_x, b_y)||_1 \tag{4}$$

$$f_5 = ||\text{total travel distance}/\max(b_x, b_y)||_2 \tag{5}$$

$$f_6 = s_2/s_1 \tag{6}$$

$$f_7 = \#\text{ of zero crossings of } v_x \text{ and } v_y, \tag{7}$$

where $\Delta\theta_i$ is the change in direction of p_x and p_y with a minimum step size of 5 mm; b_x and b_y are the edges of the bounding box of p_x and p_y; s_1 and s_2 are the eigen values (in descending order) of the point clouds of (p_x, p_y).

The angle features f_1 and f_2 capture the properties of sharp and frequent turns of information-bearing writing. If the window contains a writing event, f_3 is likely to be close to 0.5 (because the bounding box is closer to a square), and the normalized travel distances f_4 and f_5 become substantial (because the back-and-forth motion involved in writing). The ratio of eigenvalues (f_6) is an indicator of the shape complexity and tends to be close to one when the window contains a writing event. When computing the number of zero crossings of v_x and v_y (f_7), a threshold of ± 100 mm/s is used to avoid change in direction due to tremor or tracking noise.

Given the writing segment labeled by the subject, the ground-truth label for a window is determined as follows:

a) *writing*: the ground-truth writing segment spans more than $5/6$ of the window

b) *non-writing*: the ground-truth writing segment spans less than $1/6$ of the window

c) *mix*: otherwise

For example, Figure 6a and 6f are labeled as *non-writing*, Figure 6c and 6d are *writing*, and Figure 6b and 6e are *mix*. It is more ambiguous to judge a *mix* window as a writing or a non-writing event.

A Gaussian mixture model (GMM) classifier is used for the binary classification of writing and non-writing events. To train and tune the classifier, only *writing* and *non-writing* windows are considered. We slide the window from the beginning of each recording and obtain around $7k$ *writing* windows and $5k$ *non-writing* ones. We use all of these windows to train and test the classifier because the preliminary results of k-fold cross validation do not show much difference.

We examine the feature distributions for *writing* and *non-writing* and choose a single Gaussian mixture per model. The distributions of some features are heavily skewed, e.g., f_2 and f_6 of *non-writing* windows. To create better models for *writing* and *non-writing*, we modify the features as follows:

$$\hat{f}_i = log(f_i + \epsilon), \quad \text{where } i = 1..6, \quad \epsilon = 10^{-8} \tag{8}$$

$$\hat{f}_7 = log(f_7 + 1). \tag{9}$$

The GMM classifier provides a soft binary decision with likelihood:

$$C = L(\mathbf{f}|G_1) - L(\mathbf{f}|G_0) - d, \tag{10}$$

where $L(\cdot)$ is the log likelihood, $\mathbf{f}$ is the feature vector of a window, G_1 is the GMM of *writing*, G_0 is the GMM of *non-writing*, and d is the threshold to adjust the operating point. A window is classified as *writing* if $C \geq 0$ and *non-writing* if $C < 0$.

We experiment with different feature vectors $\mathbf{f}$ for GMMs and evaluate their receiver operating characteristic (ROC) curves. The ROC curves of individual features show that f_2 and f_3 are less effective in distinguishing *writing* and *non-writing*. Then, we experiment with four sets of feature vectors: 1) $f_{1,4-7}$, 2) $\hat{f}_{1,4-7}$, 3) f_{1-7}, and 4) $\hat{f}_{1-7}$. For GMMs, we also experiment with different types of covariance matrices of the feature vector: diagonal, full, and sparse covariance matrix. A diagonal covariance matrix means that we treat each feature in a feature vector independently. In contrast, a full covariance matrix means full dependency between features. In a sparse covariance matrix, we only correlates features that are intuitively dependent, i.e., the angle-based features (f_1 and f_2), and the distance-based features (f_4 and f_5), and set other cross covariances to zero. In Figure 7, we plot the ROC curves of the best two feature sets of each covariance matrix type. With a false alarm rate of 5.1%, $\hat{f}_{1-7}$ with a full covariance matrix has the highest true positive rate of 96.4% with $d = -1.4$, which is selected as the operating point of the classifier. We also show the corresponding confusion matrix at the operating point in Table I.

A more complicated motion results in a higher score, e.g., $C = 10.98$ for Figure 6c, and a simpler motion results in a lower score, e.g., $C = -4.29$ for Figure 6f. The windows in Figure 6a and 6f are correctly classified as *non-writing*. The windows in Figure 6c and 6d are also correctly classified as *writing*. The *mix* window tends to have a score close to 0. For

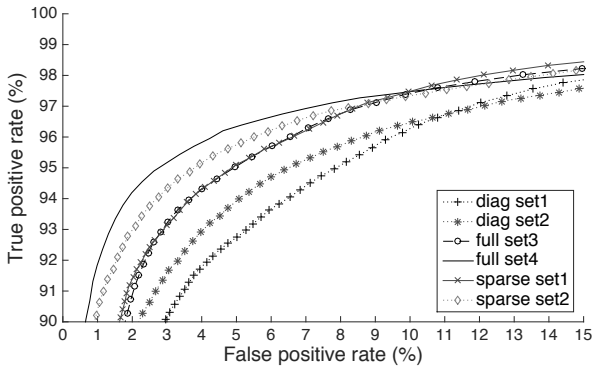


Fig. 7. ROC curves of different feature vectors and covariance matrices, where the feature set 1 to 4 are $f_{1,4-7}$, $\hat{f}_{1,4-7}$, f_{1-7}, $\hat{f}_{1-7}$, respectively, and diag, full, and sparse indicate the type of covariance matrix for the GMM. At a false alarm rate of 5.1%, $\hat{f}_{1-7}$ with a full covariance matrix (set 4) has the highest true positive rate of 96.4% with $d = -1.4$.

TABLE I. THE CONFUSION MATRIX OF WINDOW-BASED WRITING EVENT DETECTION

	classification:	
	writing	*non-writing*
case: *writing*	68324	2562
case: *non-writing*	2608	48136

example, Figure 6b is classified as *non-writing* ($C = -1.38$), and Figure 6e is classified as *writing* ($C = 0.35$). Therefore, some ambiguity is expected around the boundary of a writing segment.

C. From Windows of Writing Events to Writing Segments

After window-based writing event detection, we need to convert windows of writing events to writing segments. With our setting on the window length and step size, every sub-window of 10 samples is covered by six windows. A sub-window is determined as *writing* if two or more of the six windows are *writing*, and we combine consecutive *writing* sub-windows into a writing segment. Figure 5c is an example of the detected writing segment, which contains some distortion around the word boundary.

Compared to typical conversion schemes, e.g., majority vote or sequential testing, our conversion scheme is more greedy in the detection of air-writing because the writing parts missed in the detection stage can never be recovered in the recognition stage. Hence, the recognizer has to handle the imprecise segmentation or false alarms from the detector.

To evaluate the writing detection performance, we categorize the detected writing segments into four types:

a) *discard*: the segment has a length less than or equal to 60 samples
b) *false alarm*: no overlap with the ground-truth segment
c) *imprecise*: less than 80% overlap with the ground-truth segment, or the offset of start/end point is greater than 50 samples
d) *precise*: greater than 80% overlap with the ground-truth segment, and the offset of start/end point is less than 50 samples

Segments of the *discard* category will not be passed on to the latter recognizer. With the detector setting in Section IV-B, we have 2295 *precise*, 478 *imprecise*, 68 *false alarm*, and 164 *discard* detected writing segments out of 2700 recordings. In terms of writing event detection, all the writing activities are detected except the word `II`, i.e., 1 of 2700. The limitation of the proposed detection method is that letter `I` by itself cannot be spotted due to its simple swiping down motion.

Some subjects pause between letters when writing. If the pause is too long, it may result in separate detected writing segments (sub-word) of a word recording, e.g., `HARD` by subject `M3` is detected as three *imprecise* writing segments `H`, `AR`, and `D`. For these sub-word detections, we manually examine the writing segments and assign the correct labels of letters.

As introduced in [1], the letter-based word recognition can handle arbitrary letter sequences, i.e., words and sub-words make no difference. On the other hand, the word-based word recognition can only recognize a complete word that is specified in the vocabulary. Since it is impractical to include all possible sub-words in the vocabulary for word-based recognition, we merge detected segments that are no more than 60 samples apart and include the motion in-between as the ligature between sub-words. The merged detection results have 2225 *precise*, 483 *imprecise*, 30 *false alarm*, and 54 *discard* writing segments. After merging, there are still 18 *imprecise* segments containing partial words, which are excluded in the evaluation of word-based recognition. Note that the merge operation may connect nearby false detections and distorts the boundary of a writing segment, which explains the decrease of *precise*, *false alarm*, and *discard* segments.

V. CONTROLLER-FREE AIR-WRITING RECOGNITION

A. The Recognizer

HMMs are suitable for modeling spatial-temporal signals and commonly used for online handwriting recognition. Writing in the air creates uni-stroke patterns that deviate from conventional writing patterns. In [1], we have shown how to model air-writing with the elements of letter and ligature models. Herein, there are two major differences for air-fingerwriting recognition: *a*) the Leap only tracks the position and velocity; and *b*) explicit segmentation of the writing signal is not available as discussed in the previous section. The HMM-based recognizer of [1] for air-writing has to be modified to handle these differences.

We use the 2D position and velocity on the xy-plane as the feature (observation) vector for the HMMs. Let $P^o = [p_x(i), p_y(i)]^\top$ denote the position, and $V^o = [v_x(i), v_y(i)]^\top$ denotes the velocity, where $i = 1, 2, ...N$, and N is the number of samples in a writing segment. The normalization process is required to make the recognizer scale and speed invariant.

Normalization of P^o and V^o is accomplished as follows:

$$P = \frac{(P^o - \overline{P^o})}{\sigma_y}, \sigma_y \text{ is the standard deviation of } p_y \quad (11)$$

$$V = \frac{V^o}{\max ||V^o(i)||}, i = 1, 2, ...N. \quad (12)$$

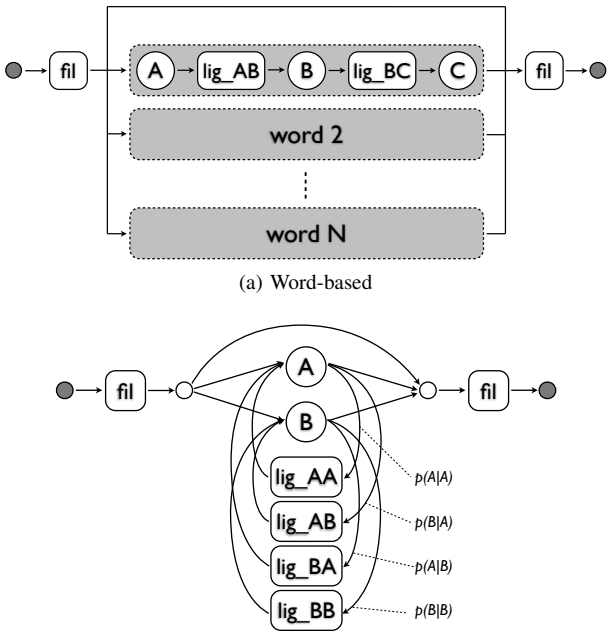
In Equation 11, we make P unit variance in the y-axis because the "height" of letters is a more reliable measurement of the virtual writing box than the "width". For example, letter `I` or `J` are thinner than `A` or `B` but of similar height. The bounding box of P cannot be used as the virtual writing box due to possible distortion introduced by non-writing motions.

The normalization can not be performed properly without roughly knowing the center and scale of the writing motion. Imagine the window of data contains a random motion in a range of 40 cm $\times$ 40 cm followed by air-writing with a bounding box size of 5 cm. In such a case, it is difficult to **offset** and **adjust the scale** of the motion signal to fit our models without knowing where the writing is beforehand. In our approach, the coarse scale of the writing motion can be determined from the detection result.

Another main difference from the push-to-write scheme is the non-writing motion incurred in the detection stage. For automatic speech recognition, the filler (or garbage) model is commonly used to absorb non-speech artifacts and handle out-of-vocabulary words for keyword spotting [19]. In our case, we use the filler model to handle the non-writing motion. The filler HMM is a single state model with self-transition and one Gaussian mixture per state. For the character and ligature models, the chosen HMM topology is identical to [1]. We train the models (with the normalized new features $\hat{P}$ and $\hat{V}$ on previous air-writing data) as the initial values of character and ligature HMMs. The filler model is initialized with zero mean and global variance of all *precise* writing segments.

With the initial character and ligature HMMs, we synthesize the HMM for each word in our vocabulary and append the filler model in the front and back. The leave-one-out cross validation on subjects results in 18 training sets. We perform embedded Baum-Welch re-estimation on all *precise* recordings in each training set. After training, forced alignment is done on the training data. When the end point of a detected segment is fairly close to the ground truth, the filler model is forced to pass through with occupancy of only one sample. When the detected segment is corrupted, the filler absorbs the non-writing motions as expected.

We use the trained HMMs of characters, ligatures, and the filler to build the word-based and letter-based decoding word network as shown in Figure 8. The word networks in Figure 8 are identical to the ones in [1] except the addition of the filler model and the skip arc. The filler models in the front and back are intended to absorb the possible non-writing motions at the begin and end of a detected word segment. The skip arc between the fillers allows the decoding path of no writing at all, which is meant to reject false alarms. We use the Hidden Markov Model Toolkit for fingerwriting modeling and recognition. For details of word-based and letter-based recognition and implementation of the recognizer, see [1].



(a) Word-based

(b) Letter-based (simplified)

Fig. 8. Decoding word networks

B. Recognition of Detected Writing Segments

It is important to understand how air-writing recognition is affected by the detection subsystem. We evaluate the recognition performance with all detected word segments except the *discard* ones. In our recording, each subject writes 100 common words and 50 unique ones. For each testing set of leave-one-out cross validation, we separate the common and unique words into two sets. The 50 unique words do not appear in the vocabulary of the training set. The unique testing set represents the most challenging case, which is equivalent to a new user writing unseen words in the training data.

The $1k$-word vocabulary is used for the word-based decoding network as shown in Figure 8a. We estimate the bigram language model from the $1k$-vocabulary and embed the conditional probabilities in the letter-based decoding network as shown in Figure 8b. The word-based recognition is evaluated with the merged detection results, and we show the average segment error rate (SER) in Table II. The letter-based recognition is evaluated with the detected segments, and we show the average SER and character error rate (CER) in Table III. The SER and CER are calculated as follows:

$$\text{SER} = \frac{E}{N_s} \quad (13)$$

$$\text{CER} = \frac{S + I + D}{N_c}, \quad (14)$$

where E is the total segment errors, N_s is the total number of segments, S, I and D are the counts of substitution, insertion and deletion errors at the character level, and N_c is the total number of characters. For example, if a detected segment `ABC` is recognized as `ABO`, we have one segment error out of one segment and one substitution error out of three characters.

TABLE II. AVERAGE SEGMENT ERROR RATE (SER) OF WORD-BASED RECOGNITION ON THE MERGED DETECTION RESULTS

	segment #	SER (%)
precise (common)	1481	0.34
precise (unique)	744	0.54
imprecise (common)	313	2.56
imprecise (unique)	152	5.26
false alarm	30	20.00
overall	-	1.15
ground truth	-	0.15

TABLE III. AVERAGE SEGMENT ERROR RATE (SER) AND CHARACTER ERROR RATE (CER) OF LETTER-BASED RECOGNITION ON THE DETECTION RESULTS

	segment #	SER (%)	CER (%)
precise (common)	1530	5.16	1.52
precise (unique)	765	6.14	1.83
imprecise (common)	325	23.69	8.21
imprecise (unique)	153	24.84	8.43
false alarm	68	47.06	-
overall	-	9.84	-
ground truth	-	4.59	-

Here, we reserve the definition of "word" as the complete word we prompt the user to write, i.e., the "word" defined in the vocabulary. SER for word-based recognition in Table II has the same meaning as the convention definition of WER, but SER for letter-based recognition in Table III does not. We keep the error metric unified as SER in this work.

We also show the number of segments for each detection case in Tables II and III. The *precise* segments are both around 82.7% of the total segments with and without merging. When computing the overall SER with Equation 13, non-rejected false alarms are counted in E, but we do not include the number of *false alarm* in N_s.

For word-based recognition as shown in Table II, the SER of the unique set is roughly two times larger than that of the common set, and 80% of the *false alarm* segments are rejected (20% SER). For letter-based recognition in Table III, the SER difference between the common and unique sets is relatively small, but the *false alarm* rejection rate drops to 52.94%. The scalability of vocabulary may be an issue for word-based recognition, but should be less a concern for letter-based recognition.

The segmentation quality from the detection result is another factor that affects the recognition performance. For word-based recognition, the weighted average SERs of *precise* and *imprecise* segments are 1.40% and 3.44%, respectively. For letter-based recognition, the weighted average *precise* and *imprecise* SERs are 5.49% and 24.90%. To have a better comparison, we also list the SER of recognition of the ground-truth writing segments, 0.15% for word-based recognition in Table II and 4.59% for letter-based recognition in Table III. The SER of the *precise* segments (with less than 50 samples of end point offset and greater than 80% overlap) is slightly worse than the SER of the ground-truth segments, but the *imprecise* segments result in much higher error rates.

We can see that the quality of writing segmentation directly affects the recognition results, and the overall performance depends on both the detector and the recognizer. Imprecise segmentation can lead to insertion of non-writing motions or deletion of the writing part. It becomes problematic when the detected word boundary is not accurate. For letter-based recognition, the non-writing part of a detected segment may be falsely recognized as a letter while the remaining part is correct. The non-writing part is less of a concern for word-based recognition due to the strong constraint on the vocabulary. For the deletion case, the letter-based recognizer may wrongly decode a partial letter as a filler (non-writing) or other letters. If the deleted part is small, it is possible for the word-based recognizer to recognize correctly based on other letters in the segment.

The overall SER of word-based recognition is 1.15%, and the overall SER of letter-based recognition is 9.84%. If user-assisted correction is allowed, providing the n-best recognition results for the user to select the right one is a good strategy to further reduce the error rates. For example, the overall SER of letter-based recognition is reduced to 5.19% for 2-best recognition and 2.52% for 5-best recognition.

VI. CONCLUSION

In this work, we attempt to detect and recognize controller-free air-writing rendered with the fingertip, which can be used as an alternative for text input on motion-based interfaces, especially when a keyboard is not available.

"Air-fingerwriting" is similar to writing on a touch-based interface, but the major difference is the lack of touch (pen-up and pen-down) information. Thus, we need to detect the writing activity and the corresponding writing segment before performing recognition. Instead of the push-to-write scheme that requires delimiters for explicit writing segmentation, we propose an approach that automatically detects writing events and segments the writing part for recognition. The proposed window-based detector classifies whether a writing event occurs in the sliding window, and the consecutive *writing* windows are processed to form a writing segment. The detected segments are passed to the recognizer for the final result of recognition or rejection. The proposed strategy is found to be effective in extracting writing trajectories from a continuous stream of motion data mixed with stray and spurious movements.

The Leap from Leap Motion is used for glove-free and marker-free finger movement tracking, which provides the 3D position and velocity of the finger motion trajectory. We recruited 18 subjects to record a total of 2700 words of a $1k$-word vocabulary. Each recording consists of two random "cursor" movements before and after the writing part. With the controller-free air-writing data set acquired through the Leap, we first train a window-based GMM detector that works at an operating point of 5.1% false alarm rate and 96.4% true positive rate of *writing* windows. Regardless of the segmentation precision, the detector commits a very low false negative rate: only misses one out of 2700 words.

We further evaluate the recognition performance of the detected segments and compare with the recognition of the ground-truth segments. The writing segmentation quality from the detection stage has a great influence on the recognition performance. The overall SERs of word-based and letter-based

recognition are 1.15% and 9.84%, respectively. In future work, we would like to further improve the recognition performance by exploiting other features, such as angle-based features, applying trigram language models, or re-scoring for n-best results.

REFERENCES

[1] M. Chen, G. AlRegib, and B.-H. Juang, "Air-writing recognition, part 1: Modeling and recognition of characters, words and connecting motions," *IEEE Trans. HumanMach. Syst*, in press.

[2] "Leap motion," http://www.leapmotion.com/, 2015.

[3] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graphics*, vol. 28, no. 3, 2009.

[4] R. Wang, S. Paris, and J. Popović, "6d hands: markerless hand-tracking for computer aided design," in *Proc. of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 549–558.

[5] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, 2000, pp. 63 –84.

[6] J. Makhoul, T. Starner, R. Schwartz, and G. Chou, "On-line cursive handwriting recognition using hidden markov models and statistical grammars," in *Proc. of the workshop on Human Language Technology*, 1994, pp. 432–436.

[7] J. Hu, M. Brown, and W. Turin, "Hmm based online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, 1996, pp. 1039–1045.

[8] B.-K. Sin and J. H. Kim, "Ligature modeling for online cursive script recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, 1997, pp. 623–633.

[9] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "Unipen project of on-line data exchange and recognizer benchmarks," in *Proc. of the 12th intl. conf. on Pattern Recognition*, vol. 2, 1994, pp. 29–33 vol.2.

[10] A. Schick, D. Morlock, C. Amma, T. Schultz, and R. Stiefelhagen, "Vision-based handwriting recognition for unrestricted text input in mid-air," in *Proc. of the 14th ACM intl. conf. on Multimodal interaction*, 2012, pp. 217–220.

[11] C. Amma, D. Gehrig, and T. Schultz, "Airwriting recognition using wearable motion sensors," in *Proc. of the 1st Augmented Human Intl. Conf.*, 2010, pp. 10:1–10:8.

[12] C. Amma, M. Georgi, and T. Schultz, "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors," in *Proc. of the 16th International Symposium on Wearable Computers*, 2012, pp. 52–59.

[13] X. Zhang, Z. Ye, L. Jin, Z. Feng, and S. Xu, "A new writing experience: Finger writing in the air using a kinect sensor," *MultiMedia, IEEE*, vol. 20, no. 4, 2013, pp. 85–93.

[14] M. Chen, G. AlRegib, and B.-H. Juang, "Feature processing and modeling for 6d motion gesture recognition," *IEEE Trans. Multimedia*, vol. 15, no. 3, 2013, pp. 561–571.

[15] P. O. Kristensson, T. Nicholson, and A. Quigley, "Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors," in *Proc. of the ACM intl. conf. on Intelligent User Interfaces*, 2012, pp. 89–92.

[16] T. Brants and A. Franz, "Web 1T 5-gram version 1," Linguistic Data Consortium, Philadelphia, 2006.

[17] O. Golubitsky and S. M. Watt, "Online stroke modeling for handwriting recognition," in *Proc. of conf. of the center for advanced studies on collaborative research: meeting of minds*, 2008, pp. 6:72–6:80.

[18] ——, "Distance-based classification of handwritten symbols," *Int. J. Doc. Anal. Recognit.*, vol. 13, no. 2, 2010, pp. 133–146.

[19] I. Bazzi, "Modelling out-of-vocabulary words for robust speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.