A Study on Fusion Framework for Air-writing Recognition Based on Spatial and Temporal Hand Trajectory Modeling

Submitted to
Graduate School of Information Science and Technology
Osaka University

April 2019

Buntueng Yana

Publication list

Journal Paper

1. B. Yana and T. Onoye, “Air-writing Recognition Based on Fusion Network for Learning Spatial and Temporal Features,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E101-A, no. 11, pp. 1737 – 1744, Nov. 2018.

Conference Papers with Referee

1. B. Yana and T. Onoye, “Air-writing Recognition Based on Fusion of CNN and BLSTM Network,” in *Proc. Int. Workshop on Smart Info-Media System in Asia*, pp. 21 – 26, Sept. 2017.
2. B. Yana and T. Onoye, “Fusion Networks for Air-writing Recognition,” in *Proc. The 24th Int. Conf. on Multimedia Modeling*, pp. 142 – 152, Feb. 2018.
3. B. Yana and T. Onoye, “Real-time Air-writing Recognition in Motion Stream,” in *Proc. SPIE 11049, Int. Workshop on Advanced Image Technology 2019*, vol. 11049, no. 110490G, pp. 1 – 6, Mar. 2019.

Other Publications

1. B. Yana and P. Malarat, “Development of Dust Monitoring at University of Phayao,” in *Proc. Electrical, Electronics, Computer, Telecommunications and Information Technology - 5th Conf. on Application Research and Development*, May 2013.
2. B. Yana and S. Premrudeepreechacharn, “Membrane Switch Resistance Tester,” in *Proc. TRF-Master Research Congress III*, Apr. 2009.
3. B. Yana and S. Premrudeepreechacharn, “Implementation of Resistance Tester for Membrane Switch,” in *Proc. IEEE Conf. on Industrial Electronics and Applications*, pp. 3502 – 3505, May 2009.
4. K. Pengwon, S. Chaitep, P. Watnawanyoo, B. Yana, and T. Komolmis, “A Design of Controller Area Network System for Flying Robot Automatic Cruise Control Application,” in *Proc. The 4th Aerospace Conf. of Thailand*, Mar. 2008.

Abstract

Air-writing refers to writing alphabet or numeric gestures by hand or finger movement in free space. It has attracted attention since it can offer verbal communication. Even the air-writing recognition has been studied more than three decades, creating a robust system is still challenging. The primary objective of this work is studying a fusion framework for air-writing recognition from a vision-based sensor. We employ a fusion scheme by modeling the air-writing with the temporal feature augmented with the spatial feature. We address the air-writing recognition in two categories: motion character recognition and motion word recognition. The underlying assumption of motion character is that the gesture is correctly spotted; therefore, a segmentation process is not necessary for the air-writing recognition. In contrast, the motion word gesture was captured from a user in a motion stream. It does not have a sign to indicate the writing and non-writing part. Moreover, there are ligatures between the characters in motion word.

For learning the motion character, we model the air-writing by using spatial features augmented with an image-like feature. The proposed structure comprises three main parts: a CNN part, an RNN part, and a Fusion part. The CNN part consists of three convolution layers and two subsampling layers. The convolution layers in the CNN part are employed to extract information from the image-like feature. In the RNN part, there are two types of structures that were considered. The first structure is a Bidirectional Long-Short Term Memory (BLSTM), and the other one is a simplified Bidirectional Recurrent Neural Network (simplified BRNN). To obtain useful information from the temporal features, the BLSTM was deployed in the RNN part. The output of the CNN and the RNN parts were combined before feeding into the Fusion part. In the first experiment, the performance of the proposed structure was compared with three baseline references: the CNN, the BLSTM, and Yang's work. The result confirms the fusion scheme outperforms all of the references. In the second experiment, the effects of the recurrent units were examined by varying the number of BLSTM units in the RNN part. The optimum number of the BLSTM units are 15 and 25 for the numeric gesture and alphabet gesture, respectively. From the experimental results, we confined that the execution time of the fusion structure is high due to the complexity of the BLSTM unit. In the third experiment, the simplified BRNN was considered. When comparing the results with the previous experiment, the execution time of simplified BRNN unit reduces in half while the accuracy drops insignificantly. In the last experiment, we demonstrated that using hand position feature (RNN part) and image-like feature (CNN part) are adequate for the fusion network.

For leaning the motion word, a deep recurrent neural network was studied. In the output layer of the proposed structure, the Connectionist Temporal Classification (CTC) loss was considered. The main advantage of using the CTC loss is removing a predefined alignment to create the training set. The features that we studied are the hand position feature and the path signature feature. In the preprocessing stage, we employ a sliding window technique to segment a long sequence of motion gesture into small pieces. Then, each piece of motion was used to generate the hand position feature and the path signature feature. When using the sliding window technique, the most critical parameter is the size of the sliding window. The output of the fusion structure attempts to predict characters in a word; therefore, the size of the sliding window should be set to capture the data no more than one character at a time. For examining the performance of the proposed structure, two public datasets were studied, i.e. a palm-writing dataset and a finger-writing dataset. Each dataset was analyzed to obtain a writing duration per character, which could be used to set the maximum size of the sliding window. The shortest of writing duration per one character in the palm-writing dataset and finger writing dataset are 0.88 seconds and 1.38 seconds, respectively. From the experiments, the appropriate window size of the palm-writing and the finger-writing dataset are determined as 0.5 seconds and 0.25 seconds, respectively. The best recognition accuracy on the palm-writing dataset and the finger-writing dataset are 86.90% and 75.81%, respectively. We also confirmed that the required prediction time per word on the palm-writing dataset and the finger-writing dataset are 3.91 milliseconds and 6.37 milliseconds, respectively. These results confirm the proposed algorithm can be executed in a real-time.

Acknowledgments

The research journey is a challenge and valuable experience I have. This thesis is the milestone that marks all my efforts and achievements along the research journey. The friendships I have made during three years at Osaka University are equally important as well.

First of all, I would like to express my deep and sincere gratitude to my research supervisor, Professor Takao Onoye of the Osaka University, for providing me a precious opportunity and excellent environment to study as a doctoral student in his laboratory. His patient guidance, enthusiastic encouragement, constructive comments, a professional guide, and useful critiques let me successfully achieve this research work.

I am sincerely express my appreciation to Professor Masanori Hashimoto of Osaka University, Associate Professor Ittetsu Taniguchi of Osaka University, and Visiting Professor Fumio Kishino of Kwansei Gakuin University for agreeing to be my dissertation committee. They were more than generous with their expertise and precious time for reviewing my thesis.

I would like to take this opportunity to say warm thanks to all member of Information Systems Synthesis Laboratory of Osaka University, especially Mr. Liu Jin who take care of me when I first visited Japan.

I would like to dedicate this thesis to all of my teachers who mentored me throughout my career. They give me the knowledge and encourage me to study abroad.

Last but not least, I wish to thank my parents for their support and encouragement throughout my study. The love and guidance from them are with me in whatever I pursue. Most importantly, I wish to thank my lovely wife, Ratsada, and my adorable son, Wan, who provide unending inspiration.

Contents

List of tables

List of figures

Chapter 1

Introduction

In recent years, there is an interest in improving the human-computer interaction (HCI) in all aspects. For instance, in the game sector, many devices have been introduced in the market to acquire input data from users. The goal of HCI research is developing an effective way of communication between human and computer device. The communication method should be as natural as the way of human interaction with each other [1]. There are many ways to communicate with computational devices; e.g., the traditional inputs such as mouse and keyboard, touch screen, speech, and gesture. Each technology has pros and cons to meet the specific requirement from a user. For example, the keyboard is most effective for text input, and it needs the user to learn how to type. While the speech is more intuitively, it may interference by noise from many sources. The gesture is viable to be applied in sophisticated computing environments such as Virtual Environment [2], Augmented Reality, smart surveillance, sign language translation, medical systems [3, 4], and robot control application [5]. This research emphasizes to study a dynamic gesture as a text input.

The rest of this chapter is organized as follows. Section 1.1 presents the motivation for coordinating this research. Even though the air-writing recognition has been studied around three decades, there is still room for improvement. In Section 1.2, the objectives and contributions of this work are indicated. Finally, the main content of each chapter has been summarized and exposed in Section 1.3.

1.1 Motivation

In the early stages of the computer era, keyboard and mouse are the first attempt to acquire input directly from a user. With this kind of user interface, the user was required to adapt his/her behavior to fit the machine unnaturally. When the display technology had been developed, and a computer has more processing power, a touchscreen interface was introduced. This technology enables a user to communicate with devices more naturally and intuitively because a user can control a device by writing on the screen. Another type of user input is the human voice, which helps to improve the user experience in controlling devices in the IoT (Internet of Things) era. Another type of input is a gesture-based user interface [6], which plays important roles in the game industry. Although the human gesture is more natural than the other types of user interface, it had been rarely used because of the reliability and the precision of tracking sensors. Besides, the meaningful gesture is relatively small compared with the entire movement of a user. Generally, a system that handles the recognition process needs high processing power to classify and filter out the false positive gesture.

At present, there are a few commercial devices available for tracking the motion gesture, e.g. Microsoft kinect sensor [7, 8] and Leap motion sensor [9, 10]. The Microsoft kinect sensor is mainly designed for tracking the entire body of a user while the Leap motion sensor focuses on tracking a hand gesture. The Leap motion sensor is a small USB device, which is designed to track the finger joints and hand movement with a precision of 0.01 millimeters. We can connect it with the personal computer as illustrated in Fig. 1.1. As declared by the manufacturer, the Leap motion sensor has a maximum frame rate of 120 frames per second.

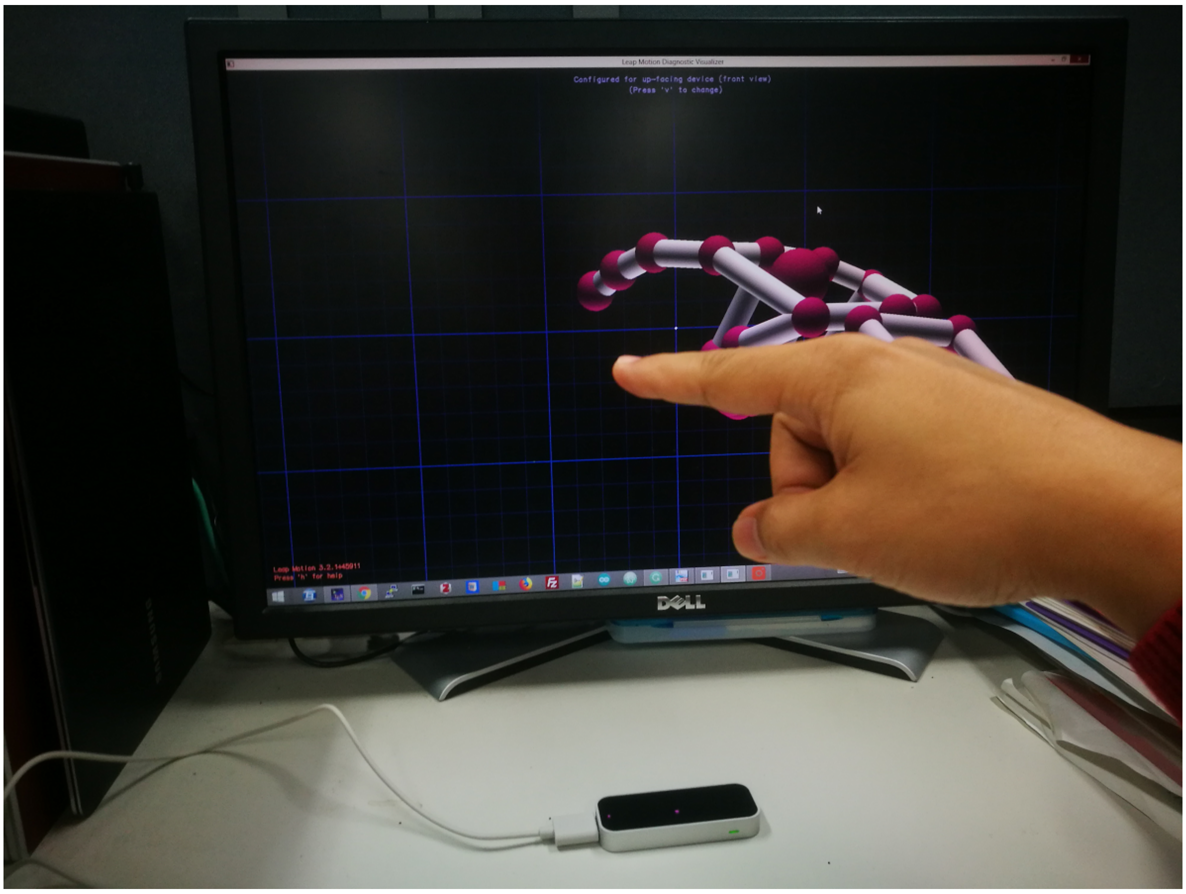


Figure 1.1: Leap motion sensor.

The data that the Leap motion sensor provides is a raw three-dimensional coordinate of the fingers in the field of view of sensors. It is also possible to acquire the confidence rate of the observed data. The value of confidence rate lays between 0 and 1, where a low value indicates the finger position would be incorrect. In some work, the Leap motion sensor has mounted on a virtual reality headset for controlling an avatar [11, 12].

During the last decade, many research actively studies human gestures for a user interface. This type of interface changes the way that we interact with some digital devices. Even though we can use many parts of the body to make a gesture, the hand gesture is the most commonly deployed in many applications since it is natural and more convenient than other types of gestures. Hand gestures can be characterized as a static gesture or a dynamic gesture owing to the motion of a user [3]. A static hand gesture refers to the relation of fingers and the hand without moving [13–15]. Many algorithms consider the static gesture with 2D models such as the image contour and the silhouette. A static gesture has been widely studied due to the low computational complexity. The dynamic gesture is another type of hand gesture, which refers to information embedded in both shape and the motion of hand reference [16, 17]. Consideration of the movement of a dynamic gesture provides more communication capability than a static gesture. In other words, we can send more information by performing the dynamic gesture.

Simple dynamic gestures, such as swipe, circle, open-palm, and close-palm, are often deployed as a command in applications. For instance, a smart television allows a user to change channels by swipe right to left, and set up a volume by moving a hand up or down [18]. Although this type of gesture is easy to manage, the communicating capability is limited by the available gestures. Although the simple gestures work quite well in practice, it is difficult to extend the set of commands. Writing a character or word in the air is a choice to overcome this problem. Referring to the result of Chen's recent study [19], the average speed of air-writing is 5.4 words per minute. It may not be fast enough for the general text input interface. However, the usability results support that the air-writing is suitable for a short-text input, and therefore it can be applied to the motion-based user interface.

Recently, air-writing, which refers to writing an alphabet or numeric gesture in free space, has attracted attention since it can offer verbal communication by constructing a word. Many techniques have been proposed to recognize the air-writing, however creating a robust system for practical application is still challenging because the air-writing is different from traditional writing in many aspects. For example, drawing the characters in the air does not have any visual or haptic feedback [19], and hence the gesture involves inconsistency. Moreover, the shape and size of a gesture may vary among trials even though the same writer writes it.

Even though the air-writing has been studied for three decades, there are some problems as follows: recognition accuracy, ambiguous recognition results, and real-time execution. Most of recent studies focus on feature state transition by modeling the air-writing as a sequence of temporal features that were extracted from the time domain. However, modeling the air-writing with state transition based methods may be vulnerable when a certain gesture consists of a combination of other similar gestures. The confusing issue can be avoided by considering a fusion scheme [20, 21]. By modeling the air-writing with a combination of temporal feature and spatial feature, a fusion structure can learn more information.

1.2 Objectives and contributions

As described in the previous section, the development of air-writing recognition is important for bridging the human-computer barrier. In addition, the recognition algorithm should recognize the air-writing with high accuracy in real-time execution. This thesis proposes fusion frameworks that satisfy those requirements.

First, we propose a fusion framework for motion character recognition. Instead of modeling the air-writing by either spatial feature or temporal feature in the same manner as other works, we employ a fusion scheme [20, 21] to learn both spatial and temporal information concurrently. The proposed framework employs the Convolutional Neural Network (CNN) and Bidirectional Recurrent Neural Network (BRNN) to extract useful information from an image-like feature and temporal features, respectively. For learning both types of features, the learning part is designed based on a deep learning structure. By stacking multiple layers of a fully connected neural network, the proposed framework outperforms all the previous studies. The performance of the proposed framework is investigated on two public datasets, namely a numeric gesture and an alphabet gesture. The average accuracies of the proposed framework achieve 99.83% and 99.25% on the numeric gesture and the alphabet gesture, respectively. When comparing the results with the CNN, the accuracies of the proposed framework improve 1.83% on the numeric gesture, and 1.42% on the alphabet gesture. When compare the results with a Bidirectional Long Short-Term Memory neural network, the average accuracies of the proposed framework improve by 0.53% and 0.22% on the numeric gesture and the alphabet gesture, respectively. By comparing the results with Yang's work, the proposed framework achieves higher accuracy by 0.84% on the numeric gesture, and 0.68% on the alphabet gesture. As for the prediction time, the proposed framework can predict the writing character in 5.57 milliseconds. Among many features that we studied, the image-like feature and the hand position feature are adequate for training the proposed framework.

We also propose a new scheme for a motion word recognition. While studying the motion character recognition, we realize that the Recurrent Neural Network (RNN) has a high impact on learning the air-writing. In the motion word recognition, we consider the multilayers of Long Short-Term Memory network as a central part of the learning structure. Instead of using an image-like feature for learning the spatial information as previous works, we employ the path signature feature which can be calculated at low cost. In the end, we reach the fusion structure which can recognize a motion word consuming low processing power. The performance of the proposed framework is examined with two public datasets, namely a palm-writing dataset and a finger-writing dataset. All features that we exploit are limited in two-dimensional space. The results are classified into three classes: precise prediction, imprecise prediction, and false prediction. From the experiments, the precise predictions of the proposed framework achieve 86.90% and 75.81% on the palm-writing dataset and finger-writing dataset, respectively. When considering the prediction time per word, the average time is 3.91 milliseconds on the palm-writing dataset. In the case of finger-writing, the proposed framework can recognize a word within 6.37 milliseconds.

1.3 Overview of the thesis

The rest of this thesis is structured as follows.

In Chapter 2, the background knowledge and related works for the air-writing recognition are reviewed. The air-writing is introduced at the beginning of the chapter. By reviewing much research, we classify the air-writing recognition into two classes: the motion character and the motion word. At the end of this chapter, the public dataset for investigating the performance of the proposed framework is reviewed and analyzed.

Chapter 3 details the proposed technique on the motion character. For learning the motion character, the proposed technique utilizes two types of neural networks, the CNN and RNN, to extract air-writing information. The CNN has been selected to extract information an image-like feature while RNN takes information out of the temporal features. All features are combined at fusion layer before being fed into a deep neural network. We conduct the experiments on the character dataset, which comprises two types of gestures: the alphabet gesture and the numeric gesture. We have investigated the performance of the fusion structure using a CNN and Bidirectional Long Short-Term Memory as base references. Moreover, we demonstrate the proposed network outperforms Yang's work [20] in both alphabet gesture and numeric gesture.

Chapter 4 describes the proposed technique for air-writing recognition on the motion word approach. Learning the motion word is more complicated than the motion character because the sample comprises gesture and non-gesture motion. Moreover, a sample does not have a clue for indicating the start and stop sign for the word writing. We adopt a segmentation free approach using the Connectionist Temporal Classification technique to minimize loss of the proposed technique. In the experiments, the performance of the proposed structure is investigated on two datasets: the palm-writing dataset and finger-writing dataset. We first examine the performance of the fusion network using a spatial and temporal feature modeling as based references. Then, we also investigate the effect of sliding window size in both datasets.

Chapter 5 concludes this thesis. The motivation, the objectives, and the main contributions are briefly reviewed to link the main conclusions. Finally, we provide directions for future work to improve the performance of the system.

Chapter 2

Background knowledge

In this chapter, we introduce basic knowledge of the air-writing. Tracking techniques and writing style are introduced. Although there are a variety of sensors for tracking a hand reference, we emphasize a vision-based approach because it is more comfortable than a sensor-based approach. We address the air-writing on two groups: motion character and motion word [22,23]. The motion character is a type of air-writing where each character is processed separately. In contrast, the motion word comprises motion characters and some meaningless motion between consequence characters. Cutting-edge techniques for motion character recognition have been reviewed in Section 2.4. Techniques for motion word recognition were reviewed in Section 2.5. In the Section 2.6, the 6DMG database [24] is presented.

2.1 Air writing

Air-writing is a type of dynamic hand gesture. It refers to writing an alphabet gesture or a numeric gesture by hand or finger movement in free space. Because the air-writing is performed in three-dimensional space, the writing trajectory could be represented by the spatial and temporal features as illustrated in Fig. 2.1.

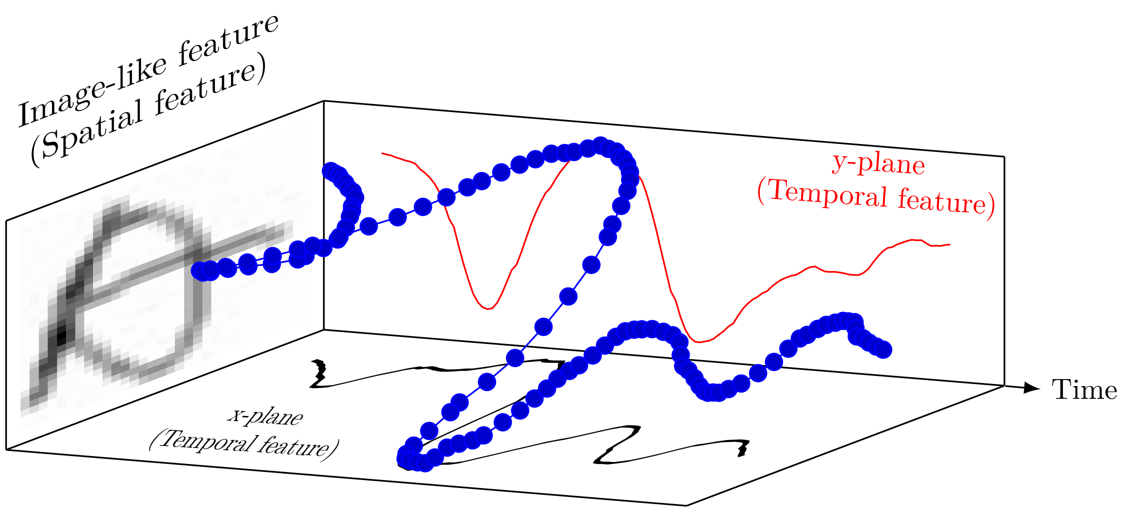
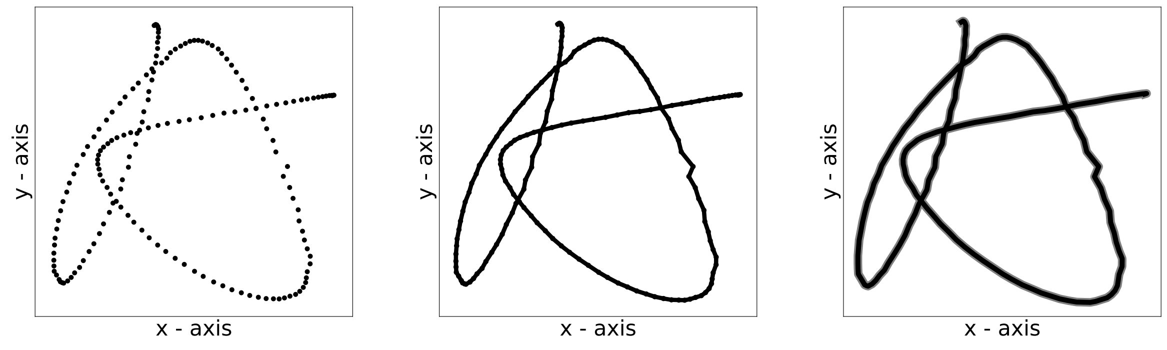


Figure 2.1: Air-writing trajectory in spatio-temporal space.

This figure exemplifies the motion character “A” written in the air. By projecting the writing trajectory in x- and y-plane, we get two one-dimensional signals in the time domain. The other view of this gesture is generated by plotting the entire trajectory points on two-dimensional space. By this way, we get an image-like feature in the spatial space. The process of generating an image-like feature is illustrated in Fig. 2.2. To begin with, the trajectory points are plotted on two-dimensional plane. Then, a gap between two consecutive points is connected with a straight line. Finally, a plot is smoothed using a spatial filter.



(a) plotting hand position on two-dimensional plane. (b) interpolating line between consecutive hand positions. (c) an image-like feature.

Figure 2.2: Image-like feature is generated by plotting entire trajectory points on two-dimensional space.

The air-writing is fundamentally different from the writing on a surface, since it provides no haptic feedback. In conventional handwriting, the sequence of discrete strokes is made by using pen-up and pen-down events. In contrast, the air-writing is rendered in an imaginary plane without feedback and lacks delimited events.

2.2 Hand tracking techniques

There are many techniques for tracking hand movement. For instance, attaching gyroscope, three-axis accelerometer, and three-axis magnetometer sensor on a user’s hand to generate the nine degree-of-freedom data (roll, pitch, and yaw) for hand orientation. Hand tracking techniques can be roughly categorized into two groups based on the motion capture mechanism: sensor-based and vision-based approaches [25–30]. In a sensor-based approach [31–34], a user is required to wear a special device as exemplified in Fig. 2.3, such as a data-glove [33, 35, 36] or tracking sensor [37]. This technique is capable of capturing motion parameters directly from the user’s hand with low latency and computational cost [38]. On the contrary, a vision-based approach is stress-free and more intuitive. Motion parameters are acquired based on the assumption that the gesture is performed in the viewing field of the camera. Even though tracking the gesture by the sensor-based approach is more reliable than the vision-based approach, wearing a cumbersome device affects the nature of user interaction.

Recently, there are commercial sensors which are quite commonly used for tracking the gesture. As an example, the Microsoft kinect [7, 8, 39–41] provides raw three-dimensional coordinates of a moving object. Another one is a Leap motion sensor [9, 11] which provides hand and finger information in three-dimensional space with an accuracy of 0.01 millimeters.

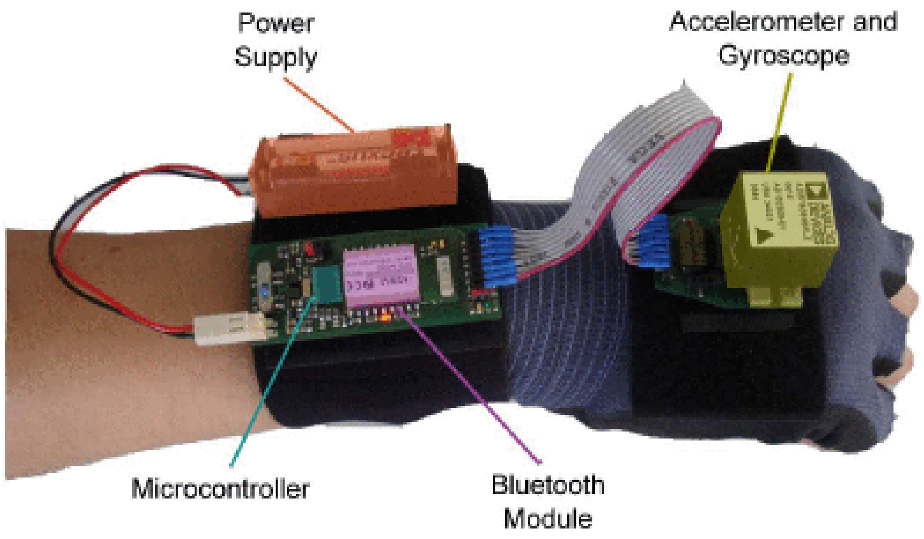


Figure 2.3: Data-glove [33].

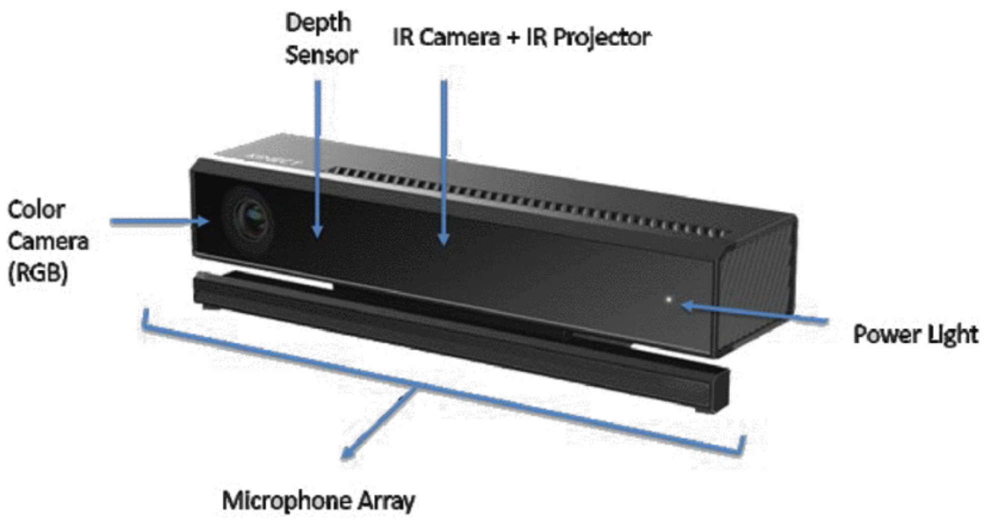


Figure 2.4: Microsoft kinect v2 [42].

These cameras can retrieve motion parameters without the limitation of environmental factors such as illumination changes and partial occlusion. Therefore, the vision-based technique has been potentially deployed recently.

Microsoft kinect was invented to control games on Microsoft Xbox and introduced in 2010 [43]. Hardware of Microsoft kinect composes of visible-light camera, depth sensor, and microphone array. These components support researchers to capture images and voice in multi dimension view. By taking color picture and depth image, Microsoft kinect can be used for tracking, interpreting body movement, and recognizing gestures or voices. Microsoft kinect has been widely used in many applications because it is a low cost RGBD cameras and supported by many software frameworks. From the specification, the Microsoft kinect v2 can capture color image with a resolution of 1920×1080 pixels. The infrared camera in the Microsoft kinect is used to capture infrared and depth image with a resolution of 512×424 pixels. All of these images can be carry out with 30 frames per second.

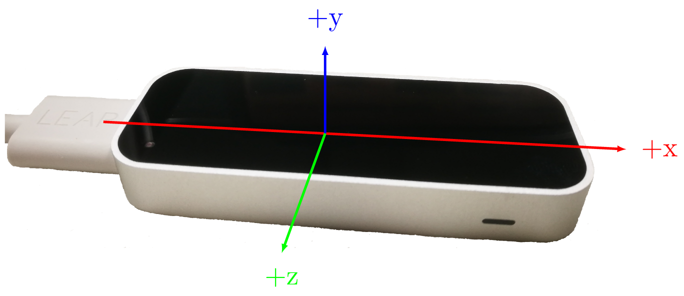
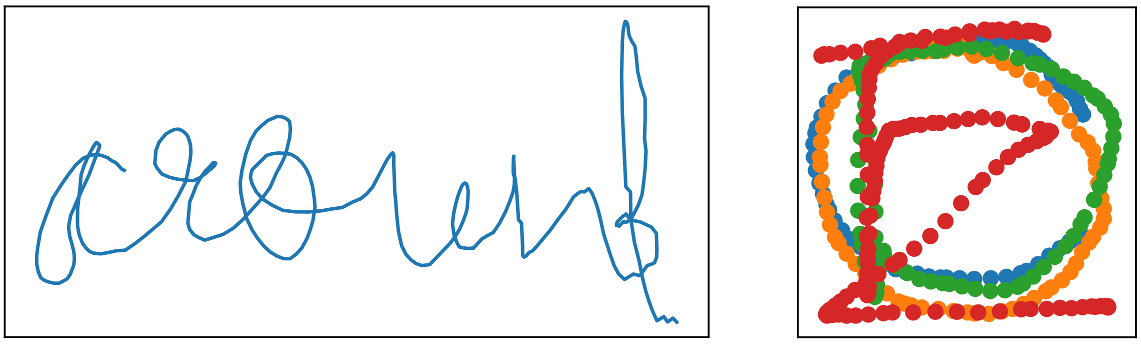


Figure 2.5: Leap motion sensor.

From a hardware perspective, the Leap motion sensor consists of two cameras and three infrared LEDs [44]. These tracking infrared light with a wavelength of 850 nanometers, which is outside the visible light spectrum. The Leap motion sensor provides data on fine-grained locations such as hands and knuckles in three-dimensional space. The tracking direction, i.e. x-, y-, and z-axis, are shown in Fig. 2.5. The Leap motion sensor has an effective range that is approximately 25 to 600 millimeters above the sensor, and it can track hand movement at 200 frames per second.

2.3 Writing style

The air-writing can be commonly classified into two styles of writing depending on the following written ways. The first style is over-writing, and the other style is writing-toward. Writing trajectories in Fig. 2.6 are the example of these styles. These samples were captured by the Leap motion sensor. Fig. 2.6 (a) is a word "around" which is written in the writing-toward style. In contrast, the Fig. 2.6 (b) is an example of the "CODE" word in the over-writing style, when each character in the word is plotted with a different color, and the color of the ligatures between the characters is omitted for clear visualization.



(a) Writing-toward style. (b) Over-writing style.

Figure 2.6: Writing-toward and over-writing style.

In the over-writing style, the characters are written in an imaginary box which has fixed height and width in the field of view of an image sensor. The next character will be drawn over the previous one. By drawing the gesture in the fixed area, the shape and size of a character are easy to control. In the writing-toward style, the characters are written from left to right, which is similar to the traditional writing in many aspects. This writing style is more complicated than the over-writing because the size and shape of the characters may vary. Besides, writing characters in an unconstrained area causes a slant which affects the accuracy of total system. In general, the over-writing style is an excellent choice for many applications because it is easy to manage, even though this writing style is not natural.

2.4 Motion character recognition techniques

The basic assumption on a motion character is that a gesture has been segmented perfectly. The isolated character can be applied with state-of-the-art techniques similar to a motion gesture. However, high variance in length and shape of gesture would be the main challenges in developing a learning algorithm. We first summarize the literature works of motion character recognition as listed in Table 2.1.

There are two fundamental problems affecting the performance of the air-writing recognition algorithm: spatio-temporal variability [45] and segmentation ambiguity [46]. As hand motion is freely rendered in the air, the air-writing shows large variation in scale, speed, and style of writing as illustrated in Fig. 2.7. Even though the same “Y” gesture is performed by the same writer in four times, the gesture may be inconsistent.

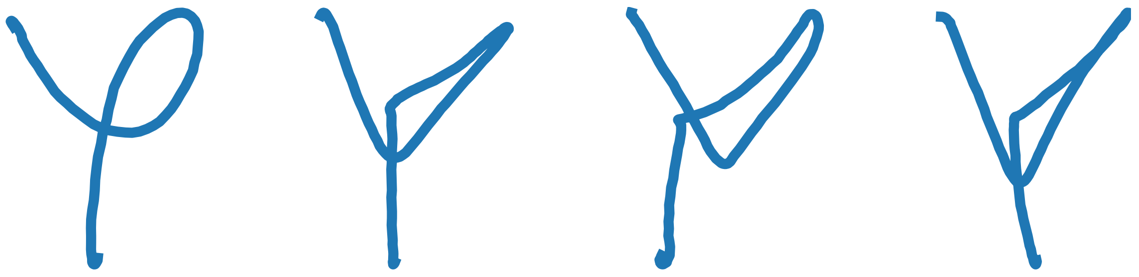


Figure 2.7: Example of a “Y” character written in the air by the same user.

As described in Section 2.1, the air-writing trajectory is represented by spatio-temporal features. Even though the gesture is performed in a three-dimensional space, the air-writing recognition usually limits the hand gesture trajectory in the two-dimensional plane for simplicity and compatibility with other works [47]. From the previous studies, the air-writing recognition is normally modeled by the features in either spatial or temporal space. In spatial space, the writing trajectory is represented by a gray-scale image. It is similar to traditional writing in many aspects. Each image-like character is generated by projecting the hand position into the visual plane. By viewing the air-writing as an image, the state-of-the-art techniques such as CNN [48,49] can be applied to the gesture recognition. In the temporal space, the observed gesture is represented by the sequences of hand position in x-, y-, and z-axis as illustrated in Fig. 2.8. There are also attempts to derive other features from the hand trajectory to raise the recognition accuracy, e.g. the writing velocity, and the angle [36] between the successive points in the trajectory, etc. As shown in Table 2.1, conventional algorithms such as Hidden Markov Model [10,36,50–54], Dynamic Time Warping [55–58], and Conditional Random Field [20,59], are commonly deployed. These techniques model the gesture trajectory as the transition of a reference point in the temporal space.

Table 2.1: Summary of related works in motion character recognition.

Authors	Year	Used features	Recognition method	Dataset
Min et al. [64]	1999	Location, velocity, angular velocity, angle	HMM	private dataset
Bhuyan et al. [65]	2008	Trajectory position, Acceleration, Hand orientation, Velocity	DTW	private dataset
Vikram et al. [66]	2013	Finger position	DTW	private dataset
Zhang et al. [7]	2013	Background, Depth, and Skin Models	Artificial Neural Networks	private dataset
Bhuyan et al. [59]	2014	Motion Chain Code	CRF	private dataset
Ma et al.	2014	Hand orientation	HMM, CRF	private dataset
Murata el al. [41]	2014	Interstroke feature	DTW	private dataset
Hameed and Hernando [53]	2015	Centroid distance , Spatial tangent	HMM	private dataset
Ayachi et al. [67]	2015	Intensity of grid cells, Number of OFF cells in a row, Heuristics patterns	HMM	private dataset
Patil et al. [38]	2016	Angular velocity, Acceleration velocity , Quaternion	DTW	private dataset
Kane and Khanna [68]	2016	Equi-polar signature	DTW, kNN	private dataset
Xu and Xue [54]	2016	angular velocity	HMM	private dataset
Ramasamy et al. [69]	2016	Image-like feature	CNN	private dataset
Islam et al. [70]	2016	Depth Information	DTW	private dataset
Chen et al. [19]	2016	Position, Velocity, Orientation	HMM	6DMG
Poularakis and Katsavounidis [46]	2016	Position Signal	Maximum Cosine Similarity, DTW	2D Graffiti, 6DMG, Kinect
Chang et al. [71]	2016	parametric and non-parametric, curvature	Spatio-Temporal Hough Forest	private dataset
Yang et al. [20]	2016	Hand position and Image-like	Fusion of CNN with CRF	6DMG
Zhang et al. [72]	2017	Writing trajectory	Multi-Layer RNNs	ICDAR-2013

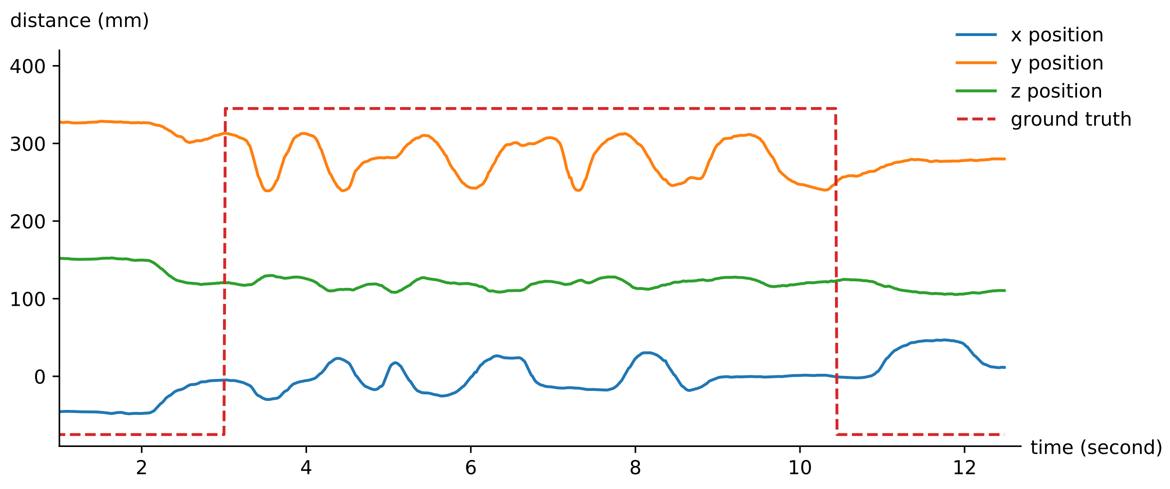


Figure 2.8: Air-writing trajectory represents in temporal space.

The Hidden Markov Model (HMM) [19,60–63] is a mathematical model for the stochastic process which generates random sequences of outcomes according to a certain probability. The HMM models the writing trajectory with a state transition of the moving trajectory. There are three types of HMM topologies: Fully Connected where any state in it can reach from other states, a Left-to-Right model where each state can go back to itself or go to the forward states, and a Left-Right Banded model where each state can go back to itself or the next state only. For modeling the air-writing, the Left-to-Right Banded HMM is commonly considered [63] because it is flexible and suitable for an order-constrained and time-evolving signals. Fig. 2.9 exemplifies an air-writing recognition framework for numeric gesture. In

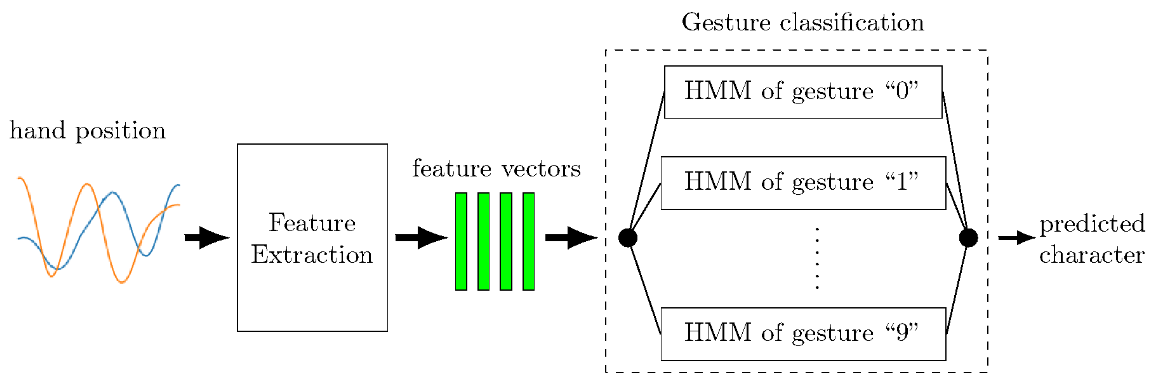


Figure 2.9: Block diagram of a framework for numeric gesture recognition.

the HMM based approach, learning a hand gesture consists of two main processes: feature extraction process and training process. In the feature extraction process, the hand position data is first converted to a sequence of fixed size feature vectors. One of the most commonly used features is an orientation feature, which is computed by the angle between consecutive points as illustrated in Fig. 2.10. Let p_t and p_{t+1} be a hand position at time t and $t+1$, respectively. The angle (θ_t) between p_t and p_{t+1} is firstly computed. Then, the θ_t is quantized into a code word $o_t \in \{d_1, d_2, \ldots, d_8\}$, where a value is finite. After that, a sequence of orientation features is fed to the HMM model in the training process. For instance, let gesture

"2" be represented by a sequence of orientation feature $O_t = \langle d_2, d_1, d_8, d_6, d_1 \rangle$ as exemplified in Fig. 2.11. The HMM model for gesture "2" comprises five state $\mathbb{S} = \{S_1, S_2, S_3, S_4, S_5\}$. At each time instant $t = \{1, 2, \ldots, T\}$, the model is located in one of possible states. An arrow represents a conditional probability depending on either previous state or current input. The

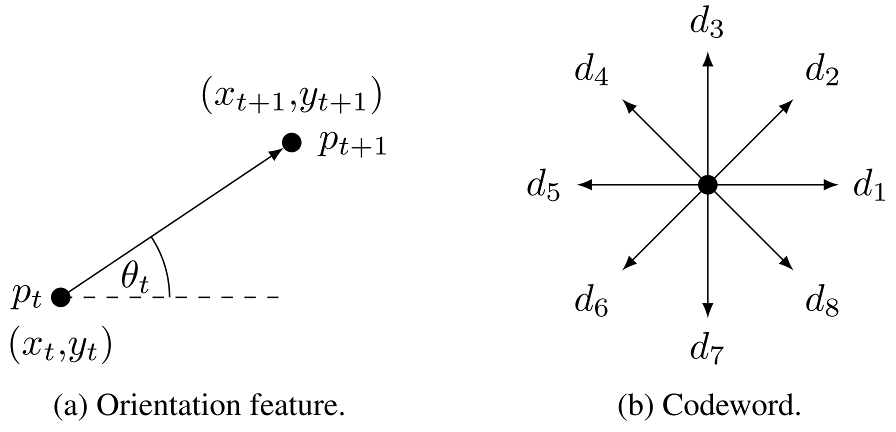


Figure 2.10: Orientation feature and codeword.

underline assumption of the HMM is that all observations are independent. When a training feature is fed to the HMM models, conditional probability distributions among all possible states are computed depending on the current and previous state. The predicted character in the output is selected from a model with the best likelihood.

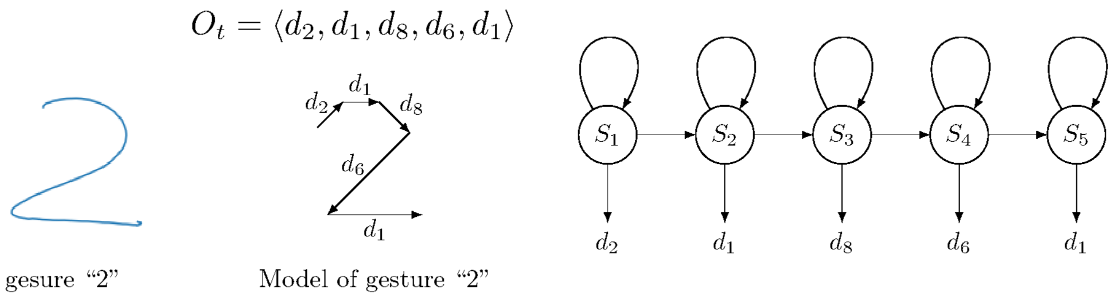
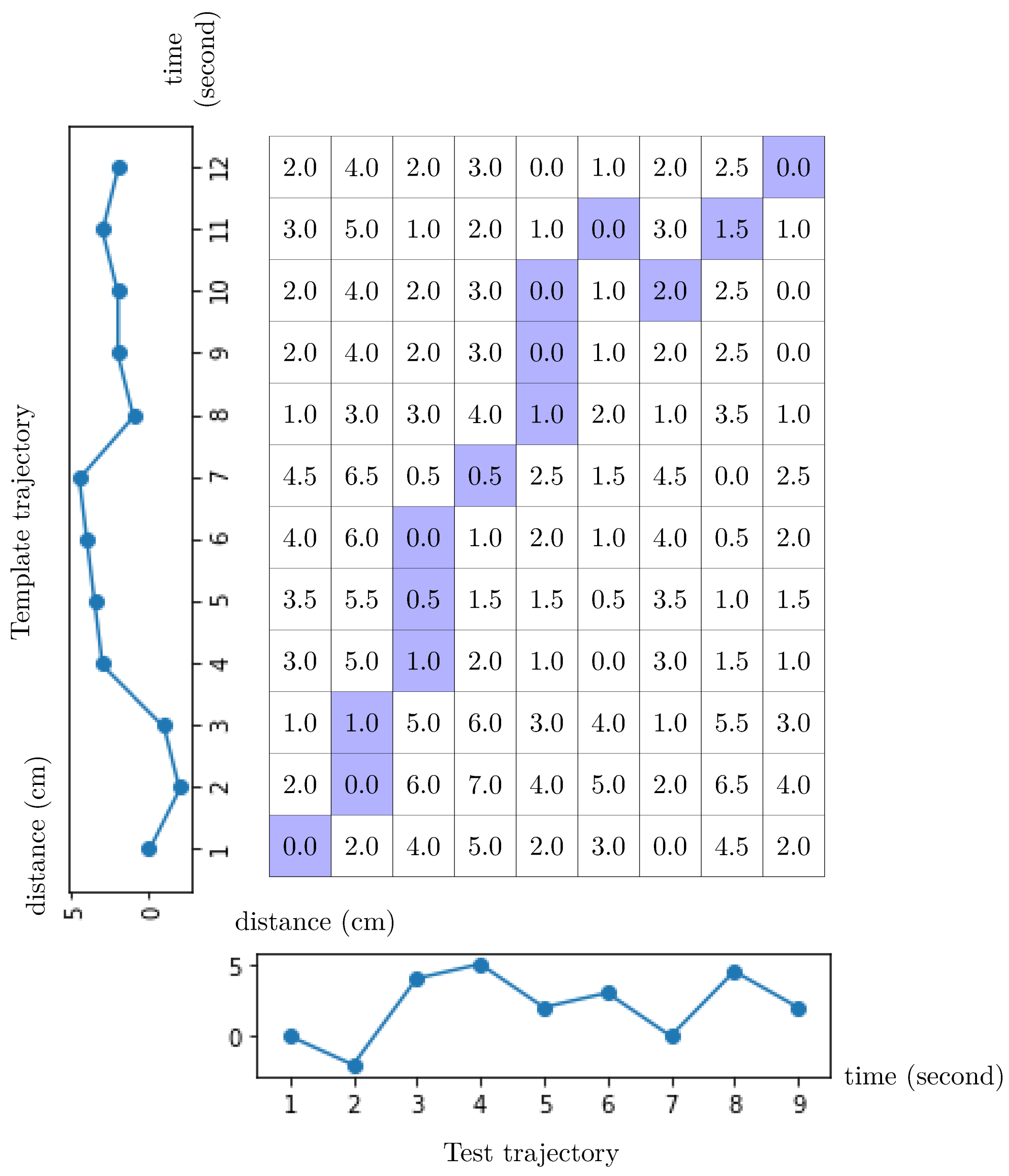


Figure 2.11: Exempler of HMM model of gesture "2".

Dynamic Time Warping (DTW) [55–58] is an algorithm for measuring the similarity between two temporal sequences: template trajectory and test trajectory, which may vary in speed and amplitude. The main advantage of the DTW algorithm is an ability to match two different length sequences as illustrated in Fig. 2.12. The DTW algorithm calculates the distance between each possible pair of input signal and template signal in term of distance values in a matrix. For instance, the template trajectory length in Fig. 2.12 is 12 points, and the test trajectory length is 9 points. When applied these trajectories with the DTW technique, we get a distance matrix size 9×12. Then the similarity of the two signals is computed using the cumulative distance value along a diagonal path in a matrix. The shortest path with a minimum accumulative distance, which is illustrated by the colored block, represents the similarity of two signals. In practice, the searching area should be fixed for controlling the computation cost.



	1	2	3	4	5	6	7	8	9
12	2.0	4.0	2.0	3.0	0.0	1.0	2.0	2.5	0.0
11	3.0	5.0	1.0	2.0	1.0	0.0	3.0	1.5	1.0
10	2.0	4.0	2.0	3.0	0.0	1.0	2.0	2.5	0.0
9	2.0	4.0	2.0	3.0	0.0	1.0	2.0	2.5	0.0
8	1.0	3.0	3.0	4.0	1.0	2.0	1.0	3.5	1.0
7	4.5	6.5	0.5	0.5	2.5	1.5	4.5	0.0	2.5
6	4.0	6.0	0.0	1.0	2.0	1.0	4.0	0.5	2.0
5	3.5	5.5	0.5	1.5	1.5	0.5	3.5	1.0	1.5
4	3.0	5.0	1.0	2.0	1.0	0.0	3.0	1.5	1.0
3	1.0	1.0	5.0	6.0	3.0	4.0	1.0	5.5	3.0
2	2.0	0.0	6.0	7.0	4.0	5.0	2.0	6.5	4.0
1	0.0	2.0	4.0	5.0	2.0	3.0	0.0	4.5	2.0

Figure 2.12: Example of dynamic time warping technique.

The Conditional Random Field (CRF) is a discriminative model, which was first introduced by Lafferty et al. [73]. It has been widely used in natural language processing, hand gesture spotting [74], and gesture recognition [75–77]. Even though the computational graph of the CRF looks like the HMM structure, the CRF allows the dependencies between the state and observations. In other words, the CRF structure is more flexible for learning complex data. A structure of a CRF framework for air-writing recognition is similar to an HMM framework. Instead of assuming that the observation is independent of each other, the conditional probabilities are computed depending on an observation (o_t) and adjacent states as illustrated in Fig. 2.13. When a training feature is fed to the CRF models, conditional probability distributions among all possible states are computed. Even though a CRF model is more powerful than an HMM model, many more parameters are necessary to be estimated. In practice, the

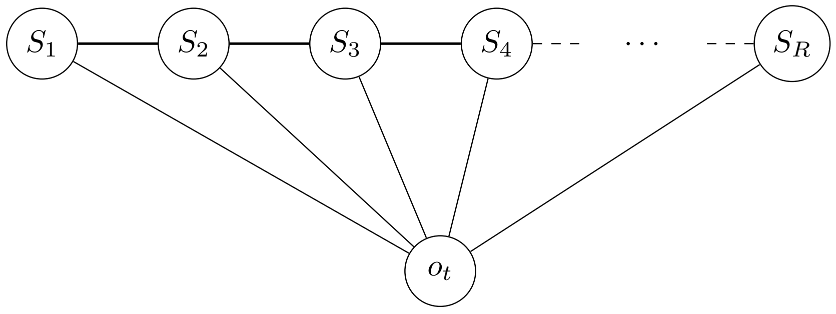


Figure 2.13: A computational graph of a conditional random field technique.

computational constraints limit the use of large state space CRF [78]. Furthermore, the CRF has computational efficiency if dependencies within state sequences are constrained [78].

These previous techniques work well with a simple gesture such as the numeric gesture because each gesture is much different from the others. When applying these techniques to the alphabet gesture, which is more complicated than the numeric one, the accuracy of these techniques may drop. For example, by using only the temporal features, the specific gesture "B" may be predicted as "D," "P," or "B" as depicted in Fig. 2.14. Since each of these candidate gestures consists of similar sequence with the certain gesture [56], if the spotting process did not classify the gesture correctly, the decision process fails. Even if the gesture can be spotted correctly, an occlusion effect may remain when using only temporal features.

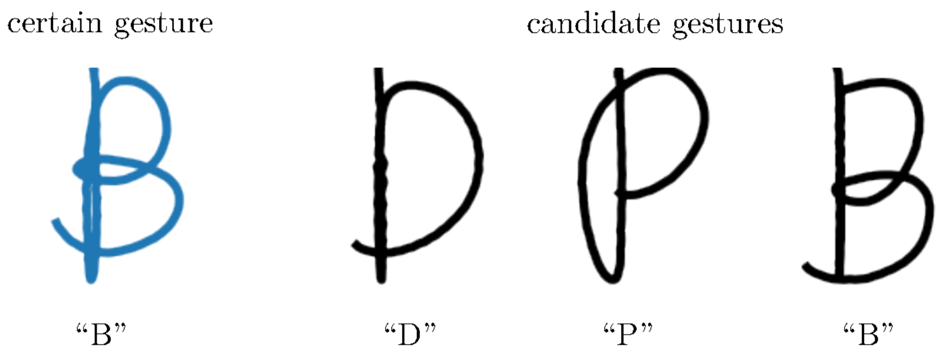


Figure 2.14: Example of occlusion gestures.

Yang et al. [20, 79] proposed a recognition method for alphabet and numeric hand gesture by combining temporal feature state modeling and total trajectory shape modeling as illustrated in Fig. 2.15. They deployed an angular feature and an image-like feature to train the temporal feature state modeling and total trajectory shape modeling, respectively. They split the recognition process into three parts: trajectory shape modeling, temporal feature state modeling, and score fusion of two modeling results. In the trajectory shape modeling, the total trajectory of segmented gesture is spotted on a two-dimensional plane to generate a gray-scale image like feature. The temporal state modeling, the angular features has been derived from the segmented data. Yang's method employed the CNN structure and the CRF structure to learn the trajectory shape and temporal state modeling, respectively. Finally, the fusion score from two output structures has been computed. From their experiments, it has confirmed that the fusion technique outperforms all of the previous works.

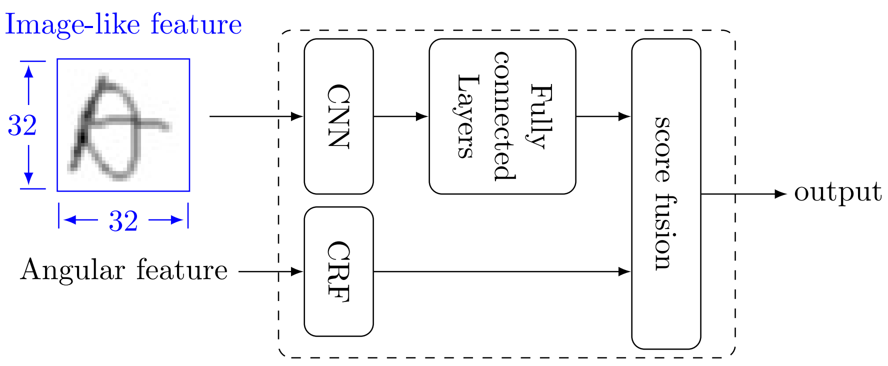


Figure 2.15: Yang's fusion framework.

2.5 Motion word recognition techniques

The motion word is complicated for automatic recognition because it lacks delimited sequence and concrete anchoring. Some works suggest a specific clue to signify the starting and stopping point of writing activity. Although writing with an explicit delimiter obstructs the user experience, the tracking data may be recognized with current recognition techniques.

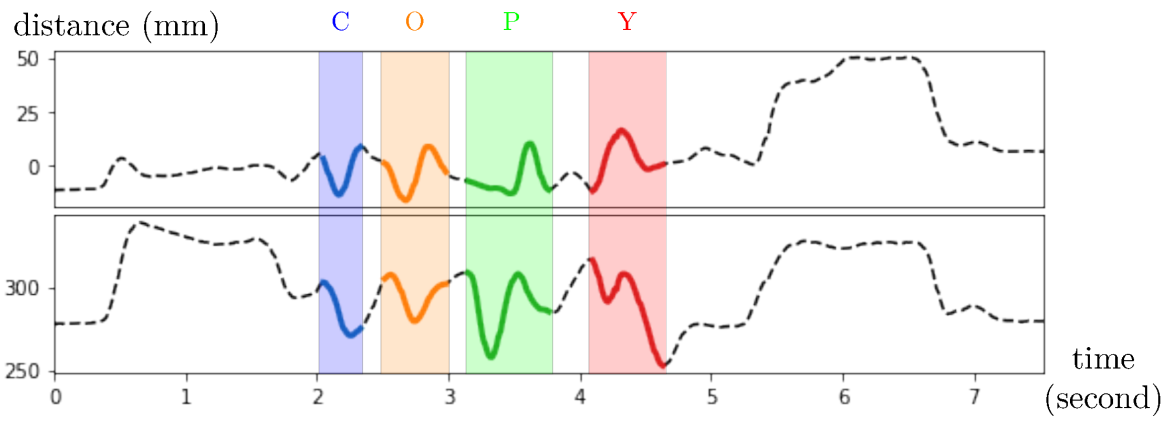


Figure 2.16: Example of a word in finger-writing dataset.

Fig. 2.16 exemplifies a sample from the finger-writing dataset. A record is mixed with writing and non-writing motion. We plot the motion trajectory in x- and y-axis separately for easy visualization in which colored blocks are used for masking a writing motion. Each colored area indicates a character by manual segmentation. From this figure, we can confirm that the meaningful trajectory is less than a half of the record. There are the ligatures between the written characters.

A general framework for the motion word recognition [3, 64, 80, 81] comprises two main processes: the gesture spotting process and gesture recognition process as illustrated in Fig. 2.17. The gesture spotting process attempts locating the meaningful patterns from the motion stream to find the start and the end points of the gesture. In other words, it locates the rightful gesture while ignoring the rest. The gesture spotting process consists of three consequence steps: motion tracking, feature processing, and gesture segmentation. After the gesture was segmented, it may be recognized using the technique which is similar to the motion character recognition. In general, the gesture spotting and gesture recognition are performed with either supervised or unsupervised learning.

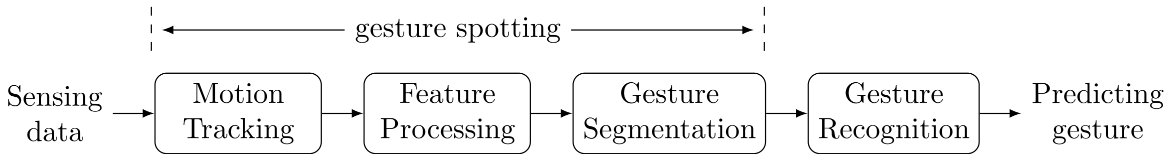


Figure 2.17: Block diagram of a general framework for the air-writing recognition.

Literature works of motion word recognition technique are summarized in Table 2.2. There are small number of researches which report the motion word recognition techniques. The common technique used for recognizing the motion word is the HMM. Recognizing a motion

Table 2.2: Summary of related works in motion word recognition.

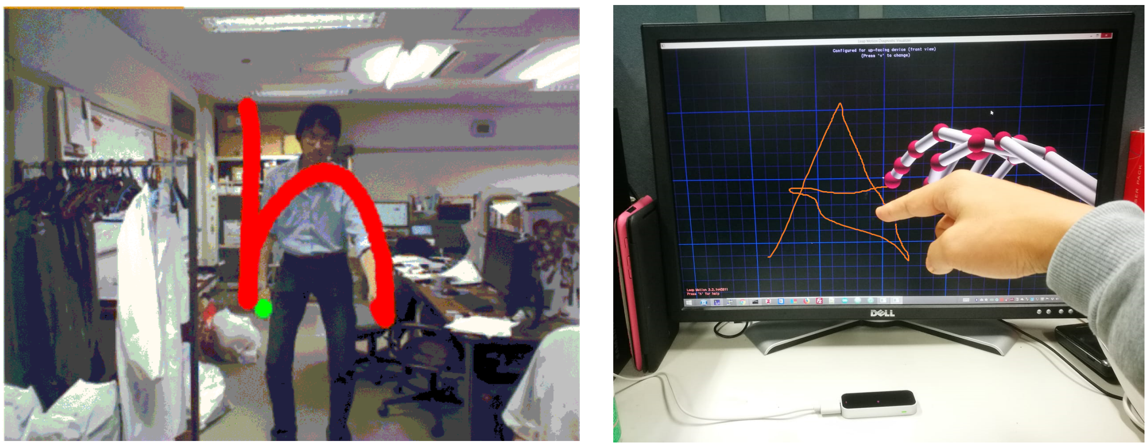
Authors	Year	Used features	Recognition method	Dataset
Amma et al. [36]	2012	Angular velocity, Mean Shift Acceleration	SVM and HMM	private dataset
Chen et al. [52]	2016	Position, Velocity, Orientation	HMM	6DMG
Kumar et al. [28]	2017	Distance, Angular, Data Distribution, Curvature, Slope, Direction	HMM and BLSTM	Leap motion and Kinect dataset
Gang and Wang [82]	2017	Offset, Curvature, Writing Direction	Deep Bidirectional RNN	private dataset

word is more complicated than a motion character approach. If the spotting process can not segment the word writing correctly, the performance of a recognition algorithm is affected. Even though the word gesture is perfectly spotted, the ligature between characters in the word remains the challenge. Therefore, many techniques have been proposed for the segmentation process, and the most efficient method that still plays a crucial role is over-segmentation [52]. The basic concept underlying the over-segmentation is dividing a motion data into a sequence of small motions where it can be used to construct a candidate prediction. Whereas the over-segmentation increases the chances of finding the boundaries of the gesture, it needs a high computational cost of generating many insignificant boundaries.

A segmentation free approach [83] is an alternative method, which completely avoids an explicit spotting process. The spotting and recognition process are executed concurrently, which make this approach flexible and reduce computational cost. In general, the motion stream is converted into slices by sliding window technique. After that, the training features are extracted from each slice and fed to the learning structure. In the training process, the Connectionist Temporal Classification (CTC) loss [85] can be applied to avoid a labeling issue, details of the CTC loss are described in Chapter 4.

2.6 The 6DMG database

The 6DMG [24], stands for 6 degrees of freedom of the motion gesture, comprises three datasets, namely character dataset, palm-writing dataset, and finger-writing dataset. The character dataset and palm-writing dataset were recorded by a WorldViz PPT-X4 and Wiimote modules by using user's hand as a tracking point. In the case of finger writing, samples were captured by the Leap motion sensor as illustrated in Fig. 2.18 (b). The writing trajectory is tracked by the tip of an index finger. Specific details of each dataset are indicated as follows.



(a) Example of palm-writing [84]. (b) Example of finger-writing.

Figure 2.18: Example of palm-writing and finger-writing.

2.6.1 The character dataset

The character dataset comprises three types of gestures: numeric gesture, alphabet gesture, and simple gesture. In this thesis we emphasize on learning alphabet and numeric gestures. The alphabet gesture is a set of the uppercase characters from “A” to “Z” while the numeric is a set of Arabic numbers. The gestures in the database are captured by the hybrid approach between optical tracking system and inertial sensor at 60 Hz. The database was collected from 25 participants. Each type of gestures was written 10 times by each participant, which makes the total number of samples 6,500. The numeric gesture was collected from 6 participants while each character was written 10 times. The simple gesture contains 20 motion gestures, which have been categorized into 5 groups: swipe, forth and back, shape, circle, and twist motions. The minimum and the maximum values of the hand position, in x- and y-directions, were analyzed and summarized in Table. 2.3. According to this table, we can approximate the writing area for each class of gesture. The numeric gesture was written on the virtual plane size of 0.73 meters by 0.75 meters. In the case of the alphabet gesture, the writing space is 1.04 meters by 1.00 meter. The simple gesture was performed on the virtual plane size of 1.61 meters by 1.24 meters.

Table 2.3: Writing area analysis of the character dataset.

Gesture	X Position (m)		Y Position (m)	
	Min.	Max.	Min.	Max.
Numeric gesture	-0.3727	0.3545	-0.2967	0.4493
Alphabet gesture	-0.4803	0.5610	-0.4141	0.5835
Simple gesture	-0.7276	0.8864	-0.4071	0.8287

We analyze the writing time duration on each class of gesture in seconds, and then list it in Table 2.4 and Table 2.5. The shortest writing duration in the numeric gesture is 0.48 seconds, which state in the “1” gesture. While the longest gesture is the “5,” which is performed in 3.67 seconds. The mean of the writing duration on the numeric gesture is 1.82 seconds. In the case of alphabetic gesture, the shortest gesture is “I,” which is performed in 0.45 seconds. And the longest gesture is “E,” where the duration is 6.87 seconds. The average of writing duration on alphabet gesture is 1.83 seconds.

Table 2.4: Writing duration of each class of the numeric gesture in seconds.

gesture	writing duration (seconds)				gesture	writing duration (seconds)			
	Min.	Max.	Mean	STD.		Min.	Max.	Mean	STD.
0	0.98	2.80	1.62	0.48	5	1.72	3.63	2.47	0.43
1	0.48	1.82	0.84	0.33	6	1.05	2.42	1.58	0.35
2	1.05	2.87	1.75	0.44	7	1.28	2.70	1.83	0.39
3	1.27	3.00	2.03	0.46	8	1.43	2.98	2.11	0.43
4	1.32	2.85	2.20	0.41	9	1.30	2.62	1.81	0.33

Table 2.5: Writing duration of each class of the alphabet gesture in seconds.

gesture	writing duration (seconds)				gesture	writing duration (seconds)			
	Min.	Max.	Mean	STD.		Min.	Max.	Mean	STD.
A	1.62	4.47	2.66	0.62	N	1.18	3.17	1.93	0.35
B	1.50	4.40	2.61	0.62	O	0.82	2.28	1.42	0.29
C	0.58	1.97	1.29	0.33	P	1.12	3.38	1.79	0.34
D	1.05	2.90	1.98	0.41	Q	1.07	3.47	1.99	0.44
E	1.72	6.87	3.18	0.81	R	1.48	3.45	2.25	0.41
F	1.37	3.45	2.21	0.46	S	0.92	2.45	1.54	0.30
G	1.38	4.28	2.49	0.58	T	0.92	2.27	1.48	0.27
H	1.13	3.72	2.29	0.50	U	0.70	2.13	1.22	0.24
I	0.45	1.42	0.71	0.17	V	0.65	1.75	1.12	0.19
J	0.62	1.58	1.01	0.20	W	1.08	2.63	1.84	0.32
K	1.23	3.62	2.27	0.44	X	0.97	2.17	1.52	0.27
L	0.57	1.83	1.08	0.25	Y	1.10	2.73	1.76	0.34
M	1.35	3.88	2.44	0.49	Z	0.78	2.47	1.57	0.31

2.6.2 The palm-writing dataset

The palm-writing dataset is composed of 40 words collected from television channels and internet services. The longest word is "DISCOVERY" and the shortest one consists of two characters. The words have been divided into four sets and listed in Table 2.6. Each set has 10 words with four characters per word on average. The samples in palm-writing dataset have been collected from 25 participants. Only 22 people completely perform all 40 words in the vocabulary. Every subject wrote 5 times for each word. The total number of samples in this dataset is 4,665. The palm-writing dataset was written in an over-writing style. The samples in the dataset cover all 26 capital characters.

Table 2.6: Four sets of the palm-writing dataset.

Set 1	Set 2	Set 3	Set 4
ABC	BBC	WEATHER	GAME
CBS	FX	NEWS	VOICE
CNN	HULU	MLB	CALL
DISCOVERY	TNT	NFL	MAIL
DISNEY	MUSIC	TRAVEL	MSG
ESPN	JAZZ	POKER	FB
FOX	ROCK	FOOD	YOU
HBO	DRAMA	KID	GOOGLE
NBC	MOVIE	MAP	SKYPE
TBS	SPORT	TV	QUIZ

We have inferred the size of the writing plane by analyzing the motion trajectory and recorded the results in Table 2.7. The shortest writing duration is a "TV" whose data length is 1.77 seconds. The longest gesture is a "WEATHER," which was performed in 34.23 seconds. The size of the writing area is 1.06 meters by 1.10 meters. When considering the average of writing duration per character, the minimum value is 0.88 seconds.

Table 2.7: Palm-writing dataset analysis.

X Position (m)		Y Position (m)		Gesture duration per character (s)		
Min.	Max.	Min.	Max.	Min.	Mean	Max.
-0.50	0.56	-0.47	0.63	0.88	2.04	4.89

For more information, we have analyzed the statistical information of characters in word dataset and plotted in Fig. 2.19. The most frequent character written in the dataset is "O," which occupies 8.12%. The portions of "J" and "Q" are less than 1 percent.

2.6.3 The finger-writing dataset

Finger-writing refers to the writing motion that is rendered by the tip of an index finger [86–88]. The samples in this dataset were captured by the Leap motion sensor in three-dimensional space. The thumb-up and thumb-close gestures were adopted for labeling the

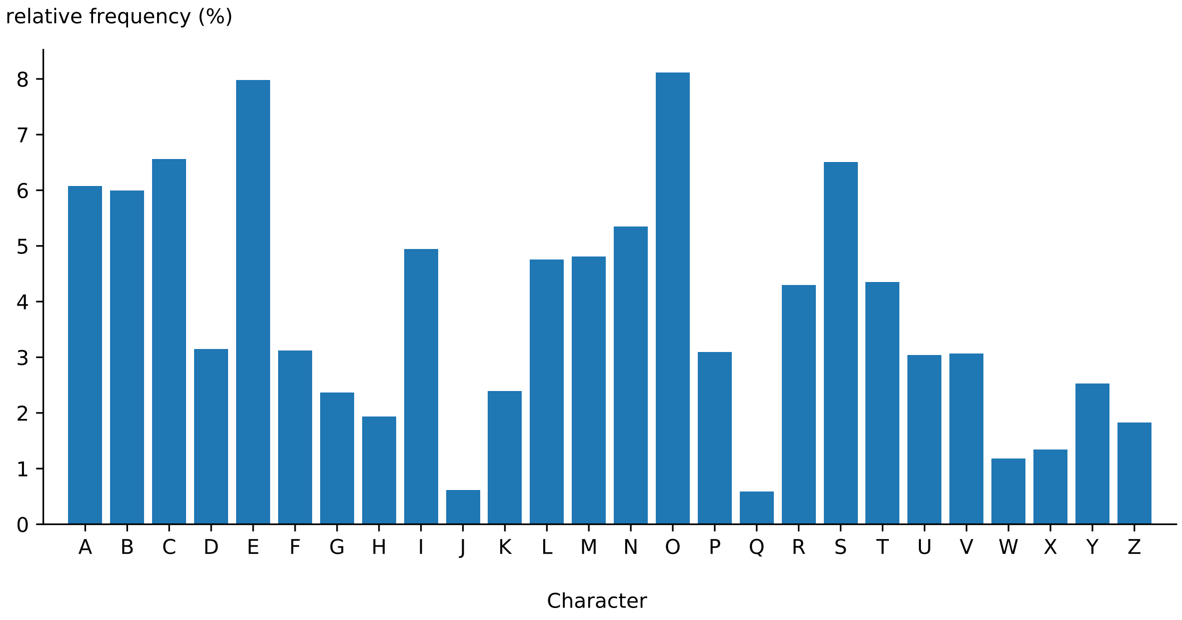
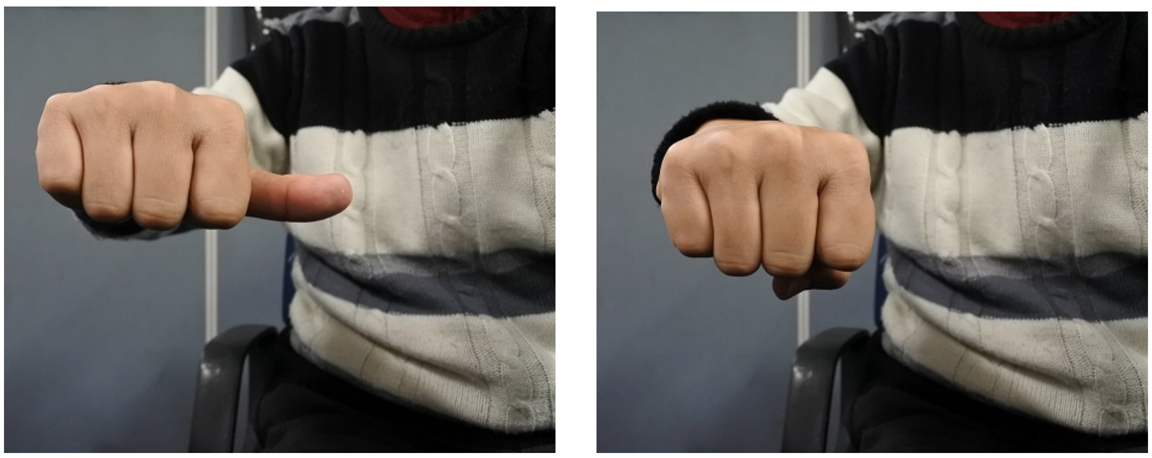


Figure 2.19: Statistics of relative frequency of character in the palm-writing dataset.

ground truth of the writing. The thumbs-up gesture indicates the starting point of the writing while the thumb-close gesture states the end of the motion. The x- and y-axis coordinates were laid in the horizontal plane in front of the user. The finger-writing samples were



(a) Thumb-up gesture. (b) Thumb-close gesture.

Figure 2.20: Thumb-up and thumb-close gestures.

recorded from 18 participants. Each person performs 150 different words: 100 common words and 50 unique words. The unique words are varied for everyone. The total number of samples in this database is 2,700. For all the samples in the dataset, the motion of the character is written with the same stroke order specified in the finger-writing dataset. Each record contains the random motion at the beginning and the end of the writing. Ergonomic information and writing duration of the finger writing have been analyzed, and then listed in Table. 2.8. The shortest word is a “TH,” which is performed in 5.25 seconds. The longest gesture is a “KNOW,” which is recorded in 43.07 seconds. The size of the writing area is

201.5 millimeters by 362.9 millimeters. When considering the average of writing duration per character, the minimum value is 0.57 seconds.

Table 2.8: Finger-writing dataset analysis.

X Position (mm)		Y Position (mm)		Gesture duration per character (s)		
Min.	Max.	Min.	Max.	Min.	Mean	Max.
-94.7	106.8	0.0	362.9	0.57	1.32	3.22

For more information, the statistical information of characters in the finger writing dataset has been analyzed and then illustrated in Fig. 2.21. The most frequent character written the dataset is "E," which occupies 10.44%. The least written character is "Z," which equals 0.70%.

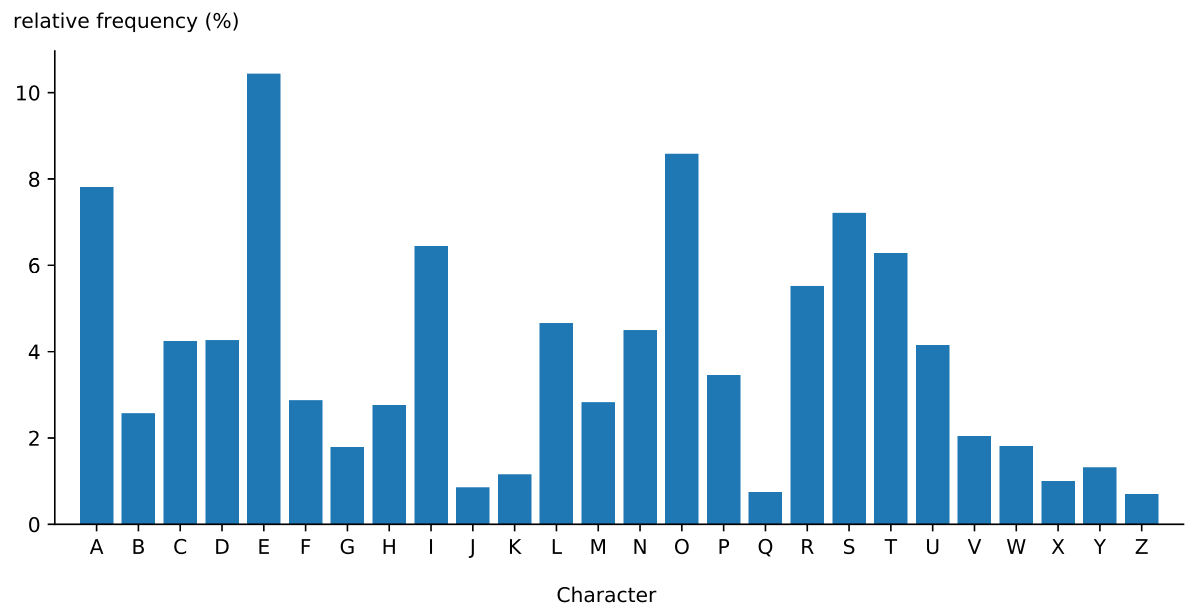


Figure 2.21: Statistics of relative frequency of character in finger-writing dataset.

2.7 Summary

This chapter introduced the basic knowledge of the air-writing. We classified the air-writing into two types: the motion character and the motion word. In the motion character, the gesture was assumed to be correctly segmented; therefore, the spotting process is not necessary. The motion character may be learned with state-of-the-art techniques such as CNN, HMM, CRF, and DTW. In case of the motion word, a sample comprises multiple characters. In general, the recognition framework firstly discriminates between a writing part and non-writing part. Then, recognition algorithms which are similar to the motion word are applied to. This technique needs a high processing cost because it need to execute two processes. We can reduce the processing time by considering the segmentation free technique.

In the last section, we also analyzed samples in the 6DMG database. The 6DMG database comprises three datasets: the character dataset, the palm-writing dataset, and the finger-writing dataset. The character dataset is used for evaluating the performance of the motion character recognition technique in Chapter 3. The palm-writing dataset and finger-writing dataset are used for evaluating the performance of the motion word recognition in Chapter 4. We recorded the writing duration per character of the palm-writing dataset and finger-writing dataset for setting up the maximum size of a sliding window. The minimum values of the writing duration are 0.88 second and 0.57 second for the palm-writing and finger-writing, respectively.

Chapter 3

Motion character recognition

This chapter presents details of the fusion structure for the motion character recognition. The main structure of the proposed network has been developed from Yang's work [20]. Instead of fusing the final score from each modeling similar to Yang's work, we employ the deep neural network for learning the fusion feature to improve the recognition accuracy. When considering the hand movement with the temporal features, some works successfully classify the writing gesture. However, the recognition result may suffer if the gesture is much complicated. The background idea of this work is to gain the advantage from both spatial and temporal views. By employing the spatial information augmented with temporal information, the learning ability of the classifier may be improved even if the hand movement includes high variability and inconsistency. One target of this work is developing the learning structure, which can learn both spatial and temporal features concurrently.

The performance of the proposed network has been evaluated using the state-of-the-art techniques and Yang's work [20] as baseline references. From the results, it is confirmed that learning the isolated written character using hand position feature and image-like feature is adequate. We also investigated the effects of the number of RNN units and the number of convolution maps in the fusion structure.

3.1 The fusion structure

In this section, we propose a method for recognizing the motion character based on fusion networks as illustrated in Fig. 3.1. We assume the gesture has been spotted correctly. The

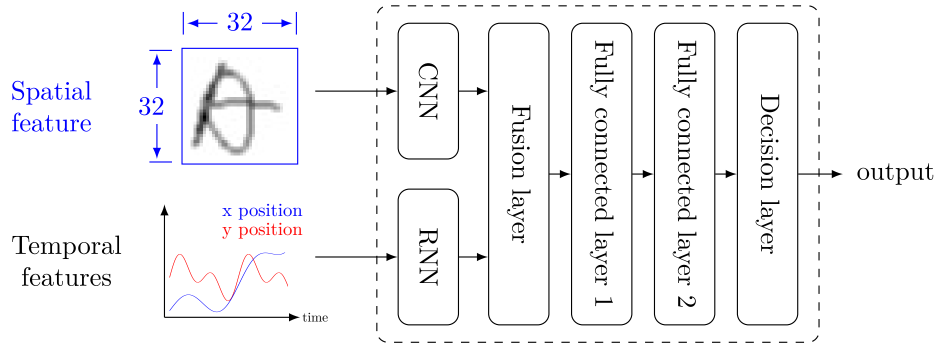


Figure 3.1: Structure of fusion network for character recognition.

entire trajectory has been used to derive spatial feature and temporal features. The spatial feature was created by plotting the hand location on the two-dimension plane. The temporal features that have been studied are hand position, angular, and velocity. By learning both spatial and temporal features, the proposed network can acquire more information than the conventional techniques. For learning a spatial feature, we deploy CNN which has shown a high performance in the handwritten recognition [48]. For learning the temporal features, Recurrent Neural Network (RNN) is considered. The proposed framework mainly comprises three parts, i.e. CNN part, RNN part, and learning part. The CNN part and RNN part are mainly assumed to extract the high dimensional spatial feature and temporal feature, respectively. In the succeeding stage, we fuse both features and utilize the fully connected layers to learn these features. Specific detail of each part is presented as follows.

3.1.1 Convolutional neural network part

The CNN part is made up of neurons that have the learnable weights and biases. It has been designed for extracting the high dimensional features from an image. The CNN comprises two types of layers: convolution layers, and subsampling layers. A convolution layer is the core building block of the CNN that does most of the computation. It transforms the input image to the high-level features such as edge, orientation, blob, or eventually honeycomb or wheel-like patterns. Rather than focusing on one pixel at a time, a convolution layer feeds input data to the predefined filters to generate the feature maps. The important parameters that should be carefully considered are the filter size and the number of filters because these parameters are directly proportional to the computation time and the size of memory. Another type of layer is a subsampling layer, which performs a downsampling operation along the spatial dimensions. The feature maps from a convolution layer are fed into a subsampling layer one map at a time. Fig. 3.2 illustrates an exemplar of a process in the subsampling layer. A feature map is divided in small portions. Then the largest value from each portion is placed in a new matrix while the rest are discarded to generate the new smaller feature map.

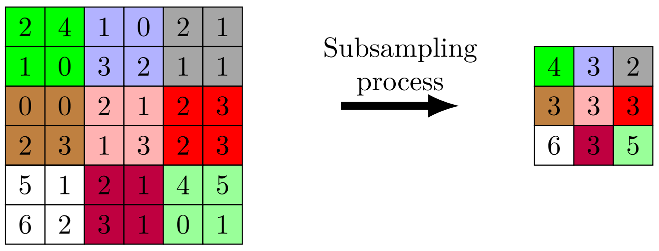


Figure 3.2: Exemplar of a process in subsampling layer.

In this work, the CNN part consists of three convolution layers (C_1,C_2,C_3) and two subsampling layers (S_1,S_2) as depicted in Fig. 3.3. All of the convolution layers utilize the fixed size 5×5 convolution kernel. The kernel size of all subsampling layers is 2×2, which reduces number of trainable parameters by four folds from the previous layer. The spatial feature for training the CNN part is created by plotting the normalized trajectory on a two-dimensional plane. The two consecutive points in the image are connected by a linear approximation. Finally, the path of trajectory image is smoothed by the Gaussian filter. In this work, the variance (σ) of the Gaussian function is set to 0.5 and the kernel size of the image filter is fixed to 3×3 pixels.

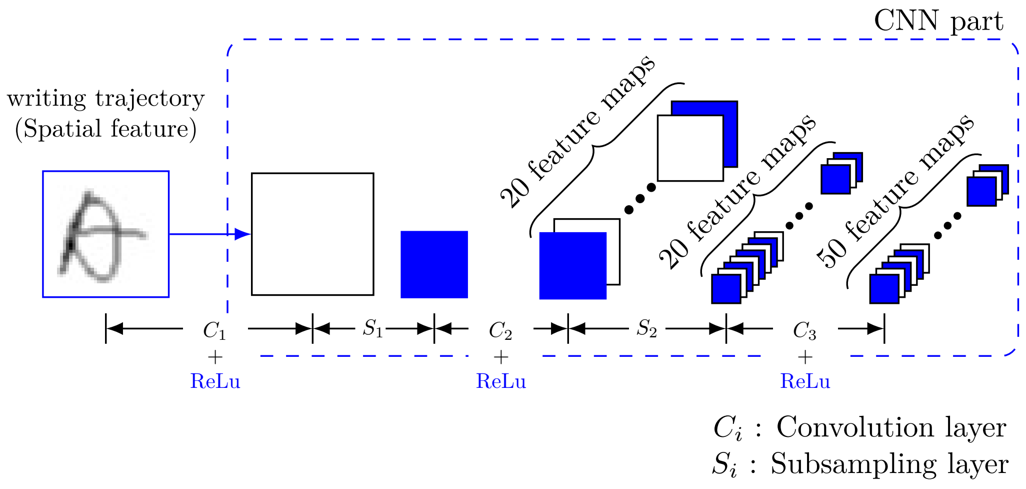


Figure 3.3: Convolution part of CNN.

The first convolution layer has one convolution filter, which produces a single feature map of the same size as an input image. The second convolution layer has 20 filters, so that the dimension and the number of feature maps at C_2 are 16×16 and 20, respectively. In the third convolution layer, the number of convolution filters is 50, and therefore 50 feature maps, each with the size of 8×8, are produced.

3.1.2 Recurrent neural network part

The RNN part comprises the RNN units each with a loop inside allowing previous information to persist. The structure of the RNN unit is represented in Fig. 3.4. In general, the RNN unit uses internal memory to store previous information while processing new input. The RNN network is ideal for learning the time series data because it performs the same computation for every element in the input data. The RNN has been proved successful for learning the dynamic time sequences such as natural language processing [85], speech recognition, and handwriting recognition [89–91].

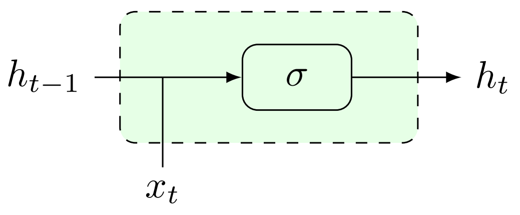
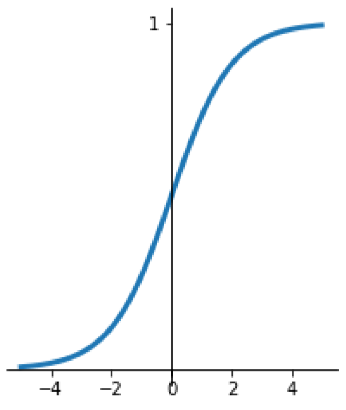


Figure 3.4: Structure of the RNN unit.

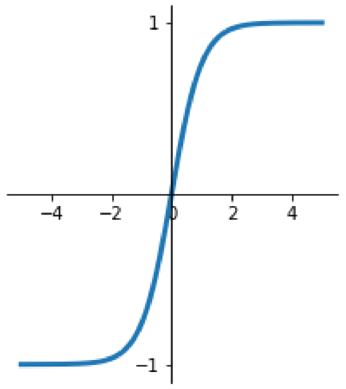
The RNN has been designed for processing variable length sequential data. The state is accumulated until the first output is created. At each time step, the RNN unit updates a hidden state by computing from the current input and the previous state. Let $X = (x_1, x_2, ..., x_T)$ denotes an input sequence where x_t is an input vector at time step t. The RNN updates the recurrent hidden state h_t based on the current input vector x_t and the previous hidden state h_{t-1} as

$$h_t = \begin{cases} 0 & \text{if } t < 0 \\ \sigma(h_{t-1}, x_t) & \text{otherwise,} \end{cases} \tag{3.1}$$

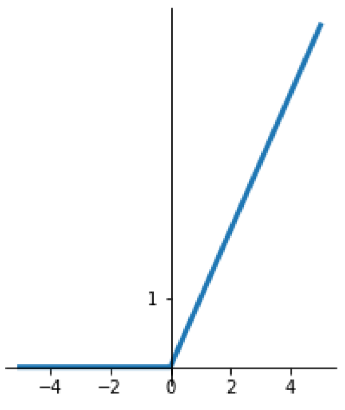
where the σ is a nonlinear function. There are three main types of nonlinear functions of RNN that are usually deployed in many applications. The first one is the sigmoid function, of RNN defined by $f(x) = 1/(1 + e^{-x})$. In general, the sigmoid function is monotonic which squashes the input value into a range between 0 and 1 as illustrated in Fig. 3.5 (a). However, this function saturates the gradients. Another function is the hyperbolic tangent, which is defined by $f(x) = (1 - 1e^{-2x})/(1 + e^{-2x})$, which normalizes a real-valued number to a range between -1 and 1 as shown in Fig. 3.5 (b). Even though the output of the hyperbolic tangent is similar to the sigmoid function, the output is zero-centered. When the input equals to zero the output equals to zero. The main advantage of the hyperbolic tangent is that the negative inputs will be mapped strongly negative, and the zero inputs will be mapped near zero. The third one is the Rectified Linear unit (ReLu). It has become popular in the modern neural network structure. The output function is defined by $f(x) = \max(0, x)$. When the input is less than zero, the output will be zero as illustrated in Fig. 3.5 (c). The output range of ReLu is $[0, \infty)$ which means it can blow up the activation. Many empirical studies have proven that the ReLu function is easier to optimize and converged faster than the others.



(a) Sigmoid function



(b) Hyperbolic tangent function



(c) ReLu function

Figure 3.5: Graphs of nonlinear functions.

A Long-Short-Term Memory (LSTM) is a type of RNN introduced by Hochreiter and Schmidhuber in 1997 [92]. It has been designed to overcome the vanishing gradient problem, which can retain information for long periods of time. This advantage allows important information learned from the early state impacts on the decision at the present state of the sequence. The LSTM unit consists of internal gates to control information flow through the learning parameters. Each unit cell of the LSTM comprises three main gates namely, input gate (i_t), forget gate (f_t), and output gate (o_t). The subscript t in each gate denotes a computation at time step t. The output of each gate is squashed with the sigmoid (σ) activation function.

We implement the RNN part by considering a Bidirectional Long-Short-Term Memory (BLSTM) network [89] as shown in Fig. 3.6. The main structure comprises two layers of the LSTM units. The first layer was designed for learning information from the writing

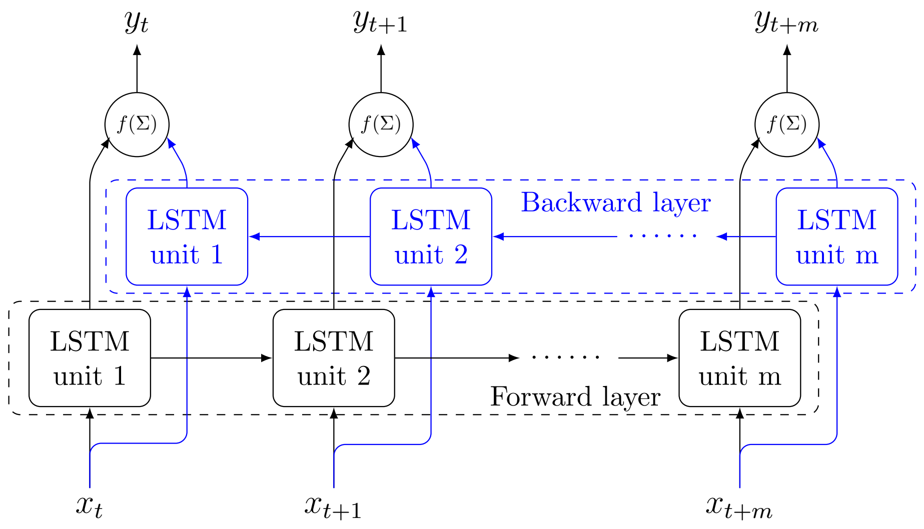


Figure 3.6: Structure of the RNN part.

trajectory in the forward direction. The other layer captures the writing characteristic in the backward direction. By deploying the bidirectional structure, the fusion network can learn the information in both forward and backward directions. Finally, the outputs of both layers were fused to generate the prediction. This type of structure can improve the performance of the gesture classification. The complete structure of the LSTM unit that we have deployed can be described by

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{3.2}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{3.3}$$
$$o_t = \sigma(W_o h_t + U_o h_{t-1} + b_o), \tag{3.4}$$
$$\hat{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{3.5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t, \tag{3.6}$$
$$h_t = o_t \odot \tanh(c_t), \tag{3.7}$$

where W_* is an adaptive weight matrix of the input in the recurrent layer, U_* is an internal weight matrix in the recurrent layer, b_* is a bias vector in each part of the LSTM unit, and σ is a nonlinear activation function. The operator $\odot$ denotes the element-wise product of two matrices. The output of RNN part (y_t) is computed by the hidden state matrix in the forward ($\overrightarrow{h}$) and backward ($\overleftarrow{h}$) layers as

$$y_t = \tanh(W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y). \tag{3.8}$$

3.1.3 Learning part

The learning part of the proposed network comprises three fully connected layers as illustrated in Fig. 3.7, where the last one is a decision layer. The number of neuron in the first and second layers are 200 and 100, respectively. In the decision layer, the number of neurons depends on the number of the gesture classes. For learning the alphabet gesture, the decision layer comprises 26 neurons, while for learning the numeric gesture, the number of neurons in the decision layer is 10. The learning parameters from CNN and RNN parts are flattened and combined to generate input for the learning part.

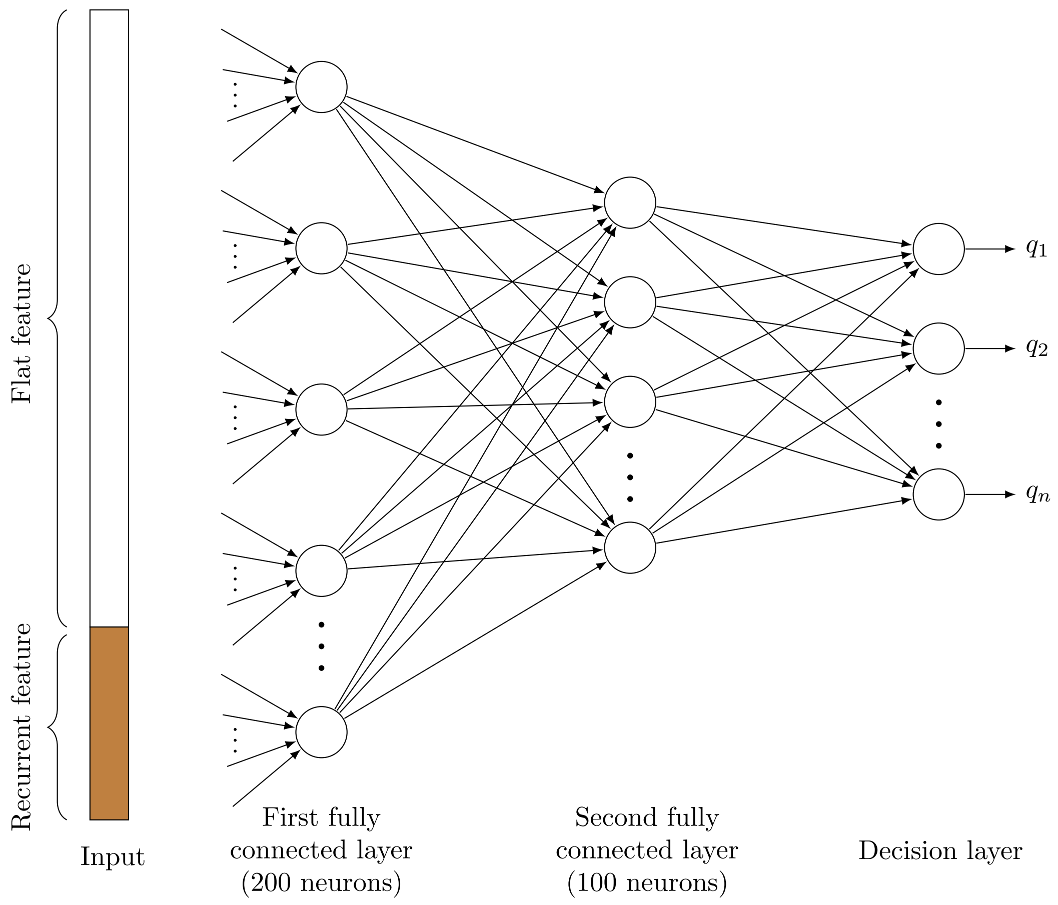

Flat feature
Recurrent feature
Input
First fully connected layer (200 neurons)
Second fully connected layer (100 neurons)
Decision layer
q_1
q_2
q_n

Figure 3.7: Structure of the learning part.

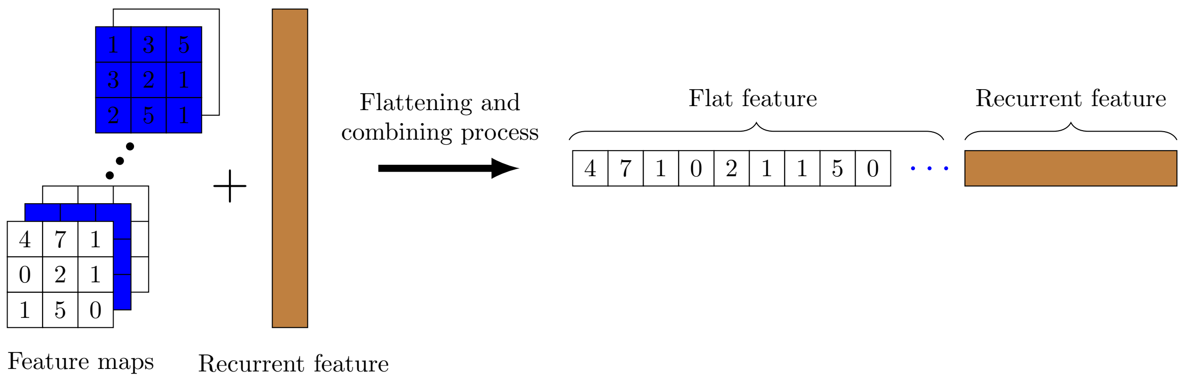


Figure 3.8: Flattening and combining process in learning part.

The process that generates input for the learning part is illustrated in Fig. 3.8. The feature maps from the CNN part are converted into a vector, namely a flat feature. Then, a flat feature and the output of the RNN part are combined. To avoid over-fitting, we also applied the dropout technique [93] for generalizing the learning parameters. The fusion score at the decision layer of the proposed framework is computed by the Softmax function as

$$q_l = \text{softmax}\left(\sum_{j=1}^{M} w_j \sigma\left(\sum_{i=1}^{D} w_i \theta_i + b_i\right) + b_j\right), \tag{3.9}$$

where θ_i is the learnable parameters from the CNN and RNN part. Other variables w_i, w_j, D, M, b_i and b_j are weight matrix of the first layer, the weight matrix of the second layer, numbers of neurons in the first layer, number of neuron in he second layer, bias vector in the input layer, and bias vector in hidden layer, respectively. The prediction class is computed by maximizing the output $\hat{q} = \text{argmax}\{q_l\}$. For learning the alphabet gesture, the output is a set of the upper case letter, i.e. $q_l \in \{A, B, \ldots, Z\}$. For the numeric gesture, the output is a set of digits, i.e. $q_l \in \{0, 1, \ldots, 9\}$. For learning a temporal information, we utilize RNN which receives sequence of x- and y- positions. The input size of RNN part is fixed to the number of recurrent units, while the output size is doubled.

3.2 K-fold cross-validation technique

For evaluating the performance of the proposed framework, we employ the K-fold cross-validation technique which is commonly conducted to verify the learning model. This method is suitable for evaluating the framework where the size of test samples is small. The samples are randomly partitioned into K mutually exclusive subsets. Each subset is approximately divided equal-size as illustrated in Fig. 3.9. One subset is kept for testing while the others are

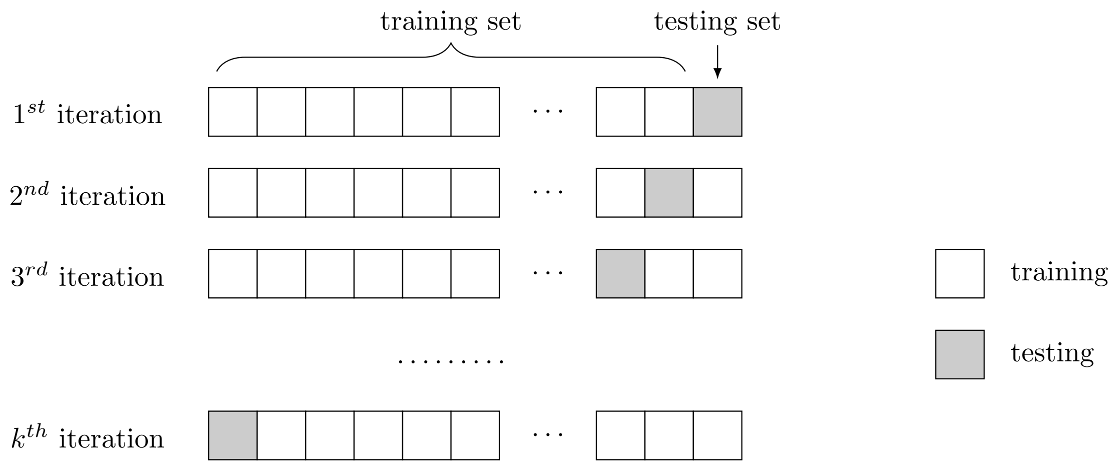


Figure 3.9: K fold cross validation.

used for training the network. This process is iterated throughout the whole K folds and then the results are computed over all the experiments. The average result is computed by

$$E_{CV} = \frac{1}{K} \sum_{k=1}^{K} E_k, \tag{3.10}$$

where E_{CV} is the expected outcome from the cross-validation and E_k is an expected value in the k^{th} fold. The K parameter is carefully selected because a lower value of K is more biased, and hence undesirable. On the other hand, a high value of K is less biased, but it can suffer from large variability. Moreover, when increasing the K value, the computation time of all folds also grows. In general, the recommended value for K in the motion gesture recognition should be more than 5 [94]. For investigating the performance of the proposed framework, the K value was set to 20 and 10 for the motion character and motion word, respectively.

In this research, we considered a similar network structure for learning both alphabet and numeric gestures so that our network can handle with both gestures in the same structure. However, during the evaluation, we have tested alphabet and numeric gestures separately in order to compare the results with previous works. The primary difference between the fusion networks for learning alphabet and numeric gesture is the number of fully connected neurons in the decision layer.

3.3 Experiments and results

This section presents the experiment that we have conducted in the following four subsections. We first studied the tuning parameters of the proposed network by setting many parameters similar to Yang's work. Then the training history of the proposed model has been recorded. By monitoring the training and validating curves, the graphs ensure that the training and tuning parameters have been set correctly. In the learning process, the over-fitting is carefully considered. We added the dropout layer in many layers to avoid the over-fitting effect. The performance of the proposed structure has been investigated on the alphabet and numeric gesture separately. We also examined the effect of learning features and important parameters of the network.

3.3.1 Feature processing

Writing a character in free-space without any visual or haptic feedbacks makes a writing trajectory inconsistency. Even though the same writer draws the same character, the shape and writing duration of the characters are varied. The number of sampling points of the gesture also varies. This phenomenon makes it difficult to execute the classification and recognition. To avoid this problem, a technique that is usually considered is resampling the writing trajectory with equal distance [95]. However, this technique incurs high computing cost. Therefore, instead of using fixed distance resampling technique, we employed a simple rule to remove the redundant points. Specifically, distances of the consecutive points are calculated. If a distance is less than a certain threshold value, the current point would be removed from the sequence. In the experiment, the threshold value for removing the redundant data is set at 5 centimeters. The distance between the consecutive points is calculated as

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}. \tag{3.11}$$

Deploying this technique can reduce the time for training the network. Before feeding the gesture in the fusion networks, we normalize trajectory with a simple process similar to Jaeger's work [96]. The writing trajectory was first splitted into x and y sequences. The normalized trajectory p_{ni} is calculated by

$$p_{ni} = \frac{p_i}{\max(\delta_x, \delta_y)}, \tag{3.12}$$

where the δ_x and δ_y are the differences between the lowest and highest values of the position in x-axis and y-axis, respectively. The δ_x and δ_y are computed as

$$\begin{aligned} \delta_x &= \max_{i=1}^{N}(x_i) - \min_{i=1}^{N}(x_i), \\ \delta_y &= \max_{i=1}^{N}(y_i) - \min_{i=1}^{N}(y_i). \end{aligned} \tag{3.13}$$

This normalization method is simple and the shape of the gesture is kept unchanged.

3.3.2 Model training history

While training the deep neural network, the learning rate is an important verifying quantity to tune the hyperparameters for the deep believe model. We monitor the performance of the proposed method by testing the model on the alphabet dataset. The gestures were partitioned into the training and validation sets. The training data is 90% of the samples, while 10% is used for validation. In the training process, the results were obtained by applying the stochastic gradient descent with a mini-batch size of 32. The momentum of learning parameters are set in the same manner as the demonstration in [97]. The results from training and validation datasets on alphabet gesture for each epoch are plotted in Fig. 3.10.

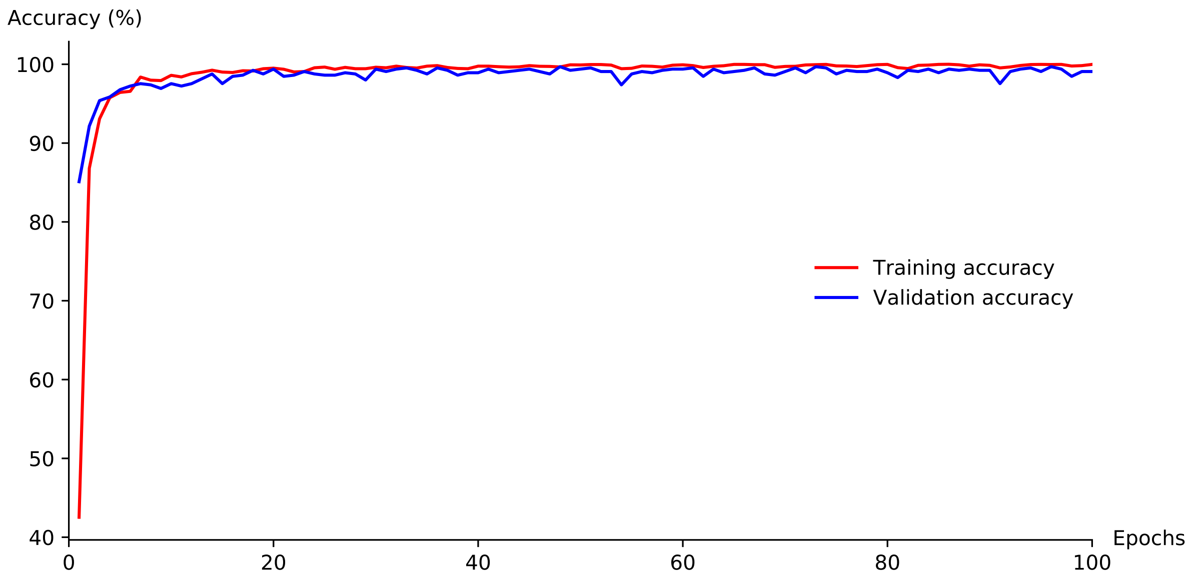


Figure 3.10: Accuracy curves of the alphabet gesture.

From the curves, the validation accuracy tracks the training curve fairly well. Moreover, the validation accuracy converges in less than 20 epochs. These graphs indicate that the learning ability of the proposed network is more than 98%, which is enough for the air-writing recognition. To confirm the performance of the proposed network, we have plotted the training and validation loss in Fig. 3.11.

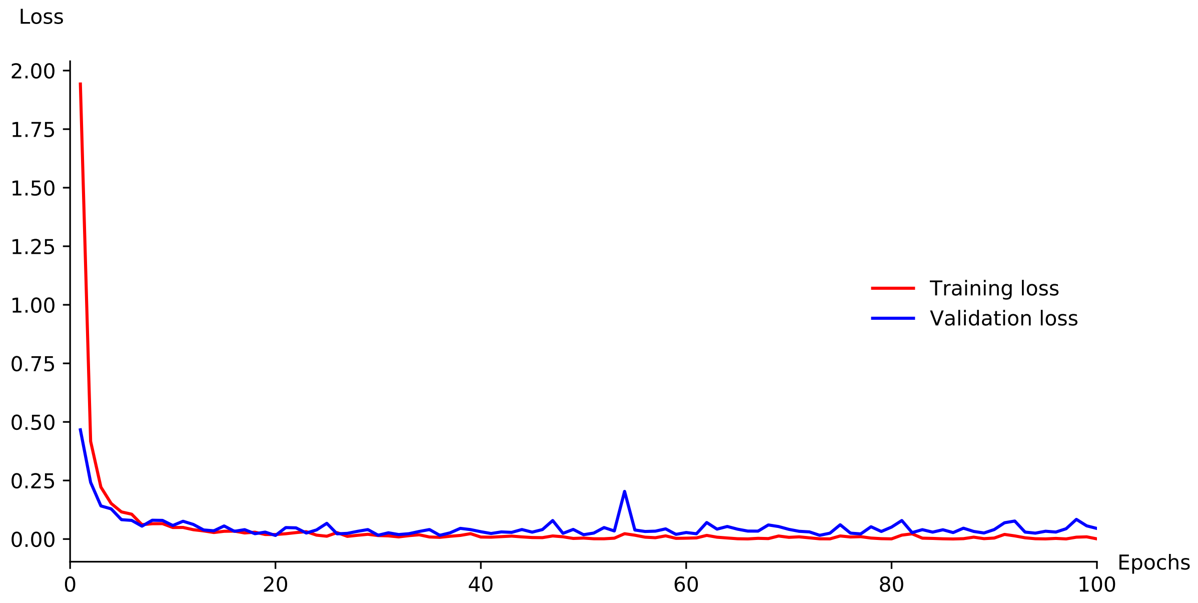


Figure 3.11: Validation loss of the proposed model on the alphabet gesture.

From the plot in Fig. 3.11, the training and validation losses decrease dramatically and converge after 10 epochs. It implies that the proposed model works well with the air-writing gesture and the hyperparameters have been set appropriately. For the next experiments, we decided to set the number of epochs to 50.

3.3.3 Performance comparison

In this section, the performance of the proposed network is first examined by using the CNN, BLSTM networks, and Yang's work [20] as the base references. We set the number of BLSTM units to 40 and applying a 20-fold cross-validation technique to find the average performance of the proposed network. We initialized the cross-validation parameter with a specifical seed, which makes the results reproducible and comparable with the based references. Experimental results on the alphabet and numeric gesture are summarized and listed in Table 3.1. The recognition accuracy of the proposed structure achieves 99.83% and 99.25% on the numeric and alphabet gesture, respectively.

Table 3.1: Accuracy comparison of the fusion network and base references.

Dataset	Base References			Fusion Network
	CNN	BLSTM	Yang's [20]	
Numeric	98.00%	99.3%	98.99%	99.83%
Alphabet	97.83%	99.03%	98.57%	99.25%

In the case of a numeric gesture, the accuracy of the proposed structure improves by 1.83% and 0.53% when compared with the CNN and BLSTM networks, respectively. By investigating results on the alphabet gesture, the fusion network gives the higher recognition rate than the CNN and BLSTM by 1.42% and 0.22%, respectively. From the results in this table, we can confirm that the recognition accuracies of the proposed network are better than those of the based references. When compared with Yang's work [20], the accuracies improve 0.68% and 0.84% on the alphabet and numeric gesture, respectively. For detailed evaluation, the error rate of each class of the numeric and alphabet gesture are plotted in Fig. 3.12 and Fig. 3.13, respectively. From the graph in Fig. 3.12, the performance of the fusion network is

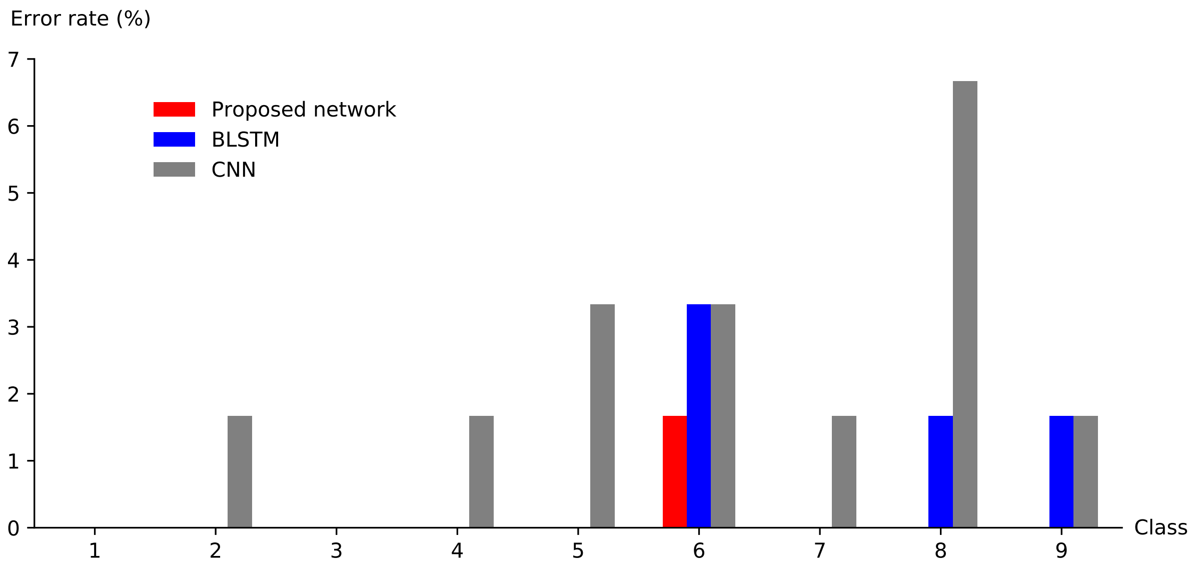


Figure 3.12: Error rate on each class of the numeric gesture.

better than those of both CNN and BLSTM in all the classes. There is only one error located in the "6" gesture. For inspecting the occlusion class of the "6", we have plotted the results in the confusion matrix as shown in Fig. 3.14. Based on the result in the confusion matrix, the actual class of "6" gesture is predicted as "0" for 1.67%.

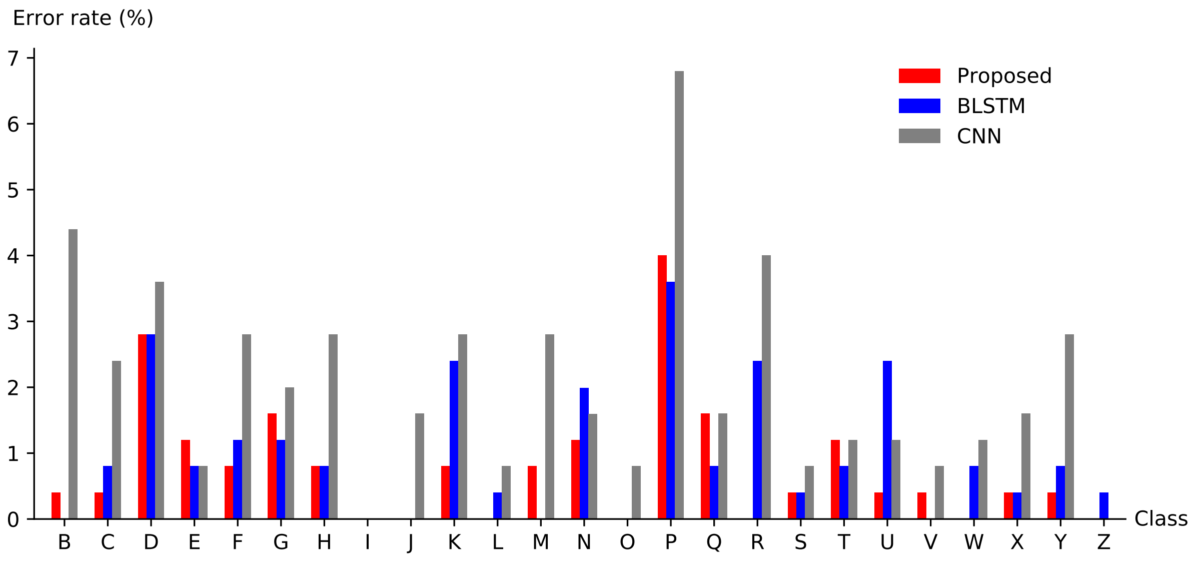


Figure 3.13: Error rate on each class of the alphabet gesture.

For the alphabet dataset, the performance of the fusion network is better than that of the CNN in all classes. While comparing the results from the fusion network with the BLSTM,

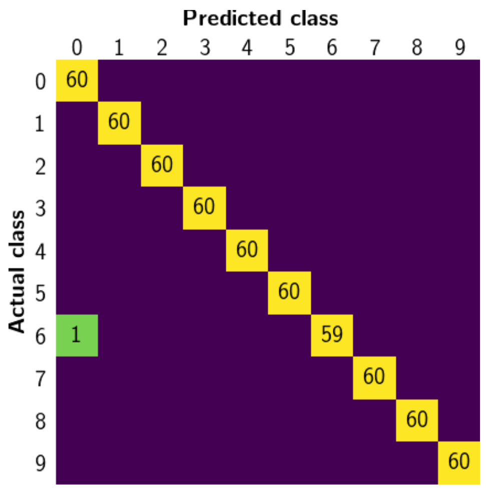


Figure 3.14: The confusion matrix of the numeric gesture.

the errors of some classes in the BLSTM are better than the proposed network. However, the BLSTM may incur a considerable amount of errors in some classes. For precise interpreting, the results of the proposed network are plotted in the confusion matrix as depicted in Fig. 3.15

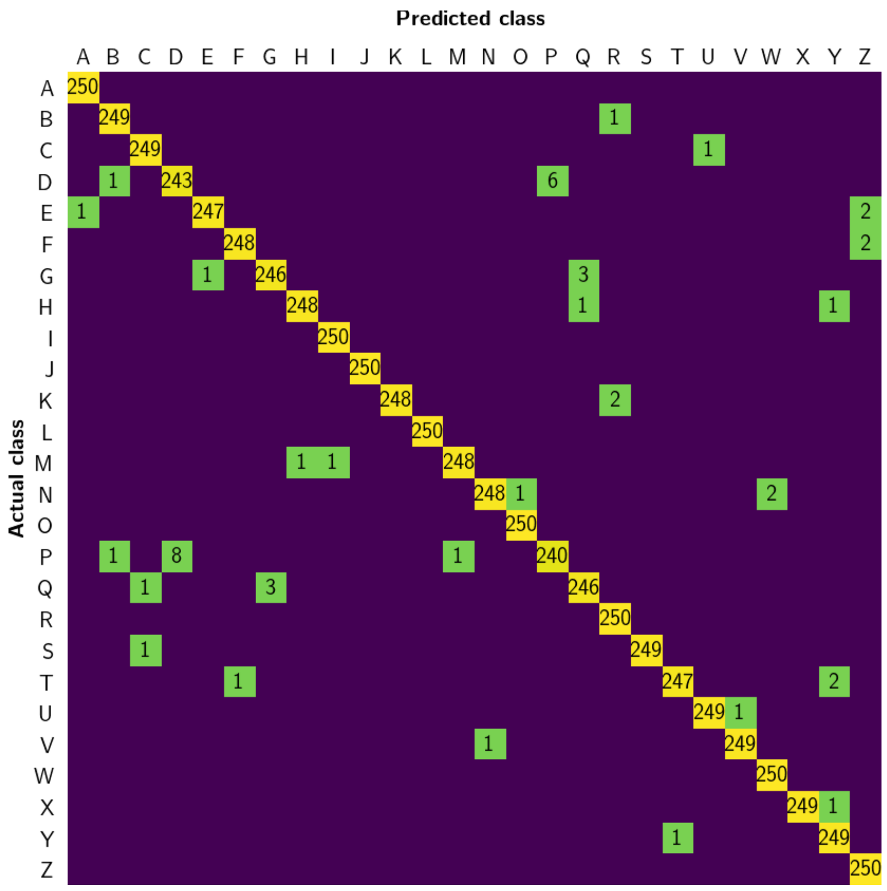


Figure 3.15: Confusion matrix of the alphabet gesture.

From the confusion matrix, there are eight classes of gestures, i.e. “A”, “I”, “J”, “L”, “O”, “R”, “W”, “Z”, which are predicted correctly. The two most misclassified classes are stated at the “P” and “D” gestures. The actual “P” gesture was classified as “D” gesture for 3.2% and the actual “P” gesture was predicted as “D” gesture for 2.4%.

3.3.4 Effect of the training features and RNN units

The main objective of this section is examining the effective learning features in the RNN part. From the reviews, there are three types of features that we are interested in: the hand position, the writing velocity, and the angular feature. We have investigated the effect of RNN units in the fusion network by varying the number of the unit cells from 1 to 40. We derive

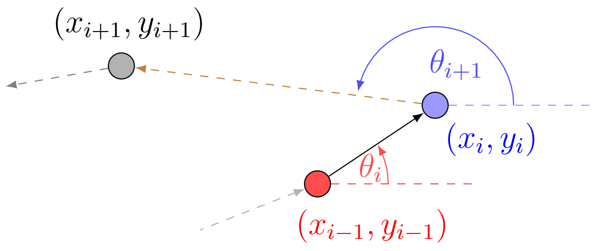


Figure 3.16: Angular feature.

two types of features: the angular and velocity features [51,52] from the hand position, since there features are commonly used for hand gesture recognition. The sequence of angular features are denoted by $A = \{\theta_1, \theta_2, \ldots, \theta_{N-1}\}$ where each element θ_i in this feature is an angle between a current and a previous hand position as depicted in Fig. 3.16. By using the current position as a reference, the angular feature has been measured from the positive x-axis going counterclockwise. The angular feature converts absolute position to the relative direction data. The element of angular feature is computed by

$$\theta_i = \begin{cases} \arctan\left(\frac{\Delta y}{\Delta x}\right), & \Delta x \geq 0 \text{and } \Delta y \geq 0 \\ \pi - \arctan\left(\frac{\Delta y}{\Delta x}\right), & \Delta x < 0 \text{and } \Delta y \geq 0 \\ \pi + \arctan\left(\frac{\Delta y}{\Delta x}\right), & \Delta x \leq 0 \text{and } \Delta y < 0 \\ 2\pi - \arctan\left(\frac{\Delta y}{\Delta x}\right), & \Delta x > 0 \text{and } \Delta y < 0 \end{cases} \tag{3.14}$$

where $\Delta x = x_i - x_{i-1}$ and $\Delta y = y_i - y_{i-1}$ are the distances of consecutive points in x-axis and y-axis, respectively. The range of angular feature is bounded by the interval [0,2π]. Before feeding the angular feature to the fusion networks, every single element of the feature is normalized by 2π. The other feature that we have studied is a velocity feature. This feature has been selected based on the fact that each gesture is made at different speeds. Many learning structures can classify a simple gesture, almost non-varying speed, and complex gestures using the velocity feature. We can view the velocity feature as a distance between the two successive points. Let us denote the velocity feature by $V = \{d_1, d_2, \ldots, d_{N-1}\}$. Each element of the angular feature is derived from the normalized position using the Euclidean distance in Eq. 3.11. The maximum velocity finally normalizes the velocity feature in each gesture, which is similar to the normalized process in the position feature.

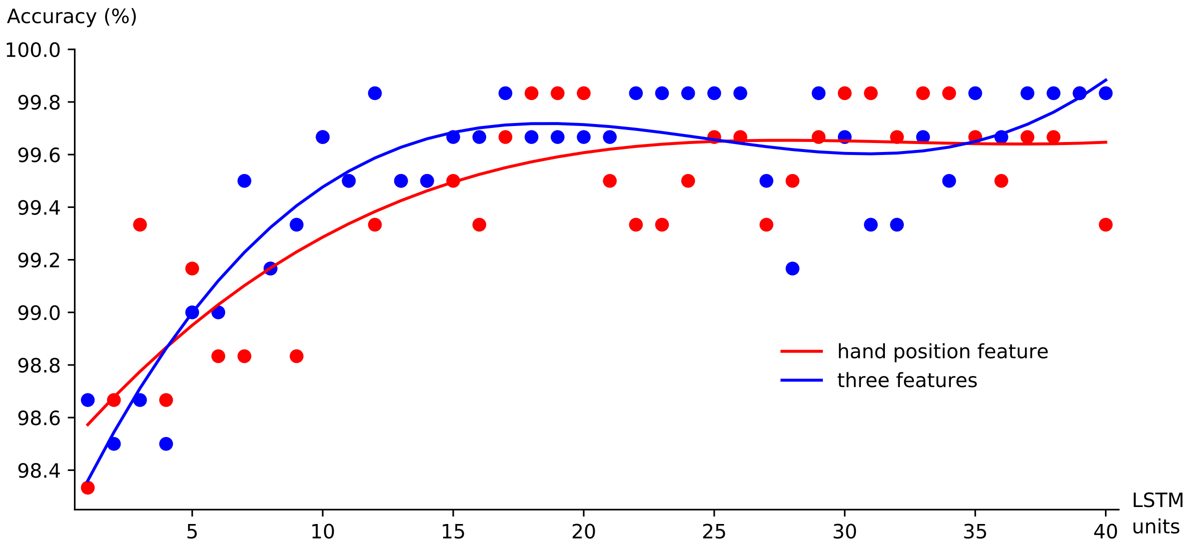


Figure 3.17: Relationship of BLSTM unit and accuracy in numeric gesture.

We start examining the effect of the BLSTM unit in the fusion network by varying the number of BLSTM units from 1 to 40. The training parameters and optimization technique have been set as same as the previous experiment. The proposed network has been trained with the combination of hand position, angular, and velocity features. The results of the numeric and alphabet gestures are plotted in Fig. 3.17 and Fig. 3.18, respectively. To analyze the effect of RNN units, we also fit the relationship of the accuracy and RNN units by using a 3^{rd} order polynomial with the minimum mean squared error. Graphs in Fig. 3.17 illustrate

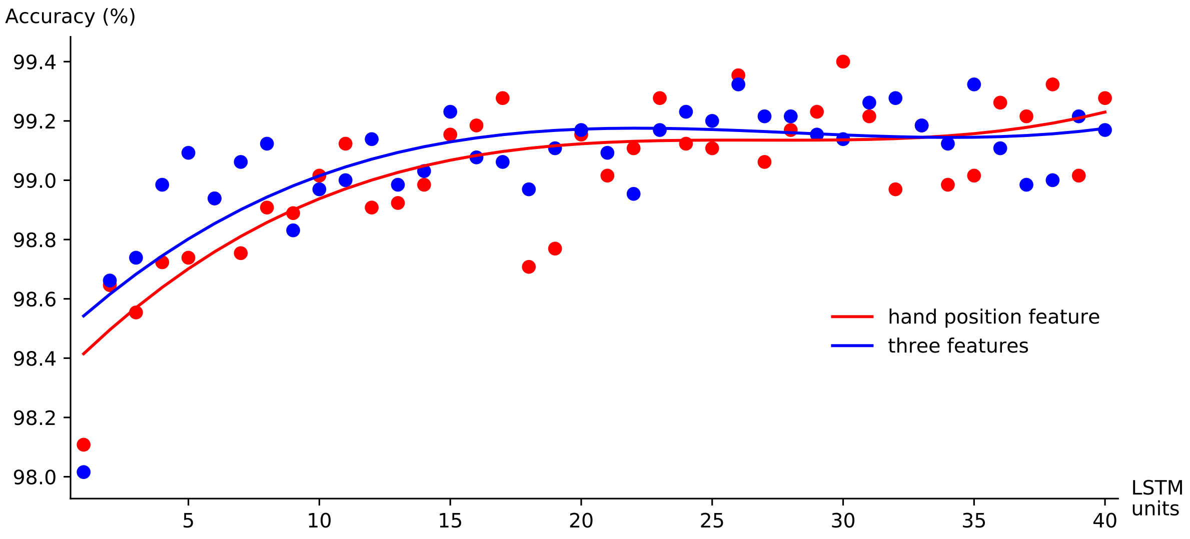


Figure 3.18: Relationship of BLSTM unit and accuracy in alphabet gesture.

the relationship between the number of LSTM unit and the average accuracy. By training the fusion network with the image-like and hand position features, the accuracy improves, as the number of LSTM unit increases, at the beginning and maintains around 99.6%. For more information, the accuracy depends on the number of BLSTM units at the beginning, and then maintains when the number of BLSTM units is more than 25. While training the fusion network using more features, the accuracy of the proposed structure improves as the number of LSTM unit increases. When the number of BLSTM units is more than 15, the accuracy maintains at 99.7%. These results illustrate that the accuracy of learning three features improves by approximately 0.1% when compared with a hand position feature.

The results from the alphabet dataset are similar to the previous experiment. By training the proposed network with three features, the recognition accuracy increases and then maintains when the number of BLSTM units is higher than 25. When comparing the result from multiple features with a hand position feature, the accuracy changes by less than 0.1%.

3.3.5 Performance of simplified bidirectional RNN network

Referring to the experiments in section 3.3.3 and section 3.3.4, training the BLSTM network is relatively slow due to the complexity of the BLSTM structure. There are many parameters that need to be calculated every time step, especially the weight matrix that is involved in the gate mechanism. In this section, we investigate the effect of the recurrent part. Instead of using a high computational cost structure like the LSTM network, we employ a simple recurrent architecture which is described by

$$\begin{aligned} h_t &= \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_n), \\ y_t &= W_{\overrightarrow{hy}}\overrightarrow{h_t} + W_{\overleftarrow{hy}}\overleftarrow{h_t} + b_y. \end{aligned} \tag{3.15}$$

For a fair comparison, we set the training parameters of the simplified Bidirectional Recurrent Neural Network (BRNN) and the BLSTM fusion structure similarly. The number of unit cells in the recurrent part has been investigated. We examined the recognition accuracy of each type of gesture by varying the number of unit cell in the RNN part from 1 to 40 and then plotted the results in the graphs. For determining general trend of the accuracy from both structures we fit the results using the 3^{rd} order polynomial with the mean square error. The computation time and learnable parameters in both structure have been studied and recorded.

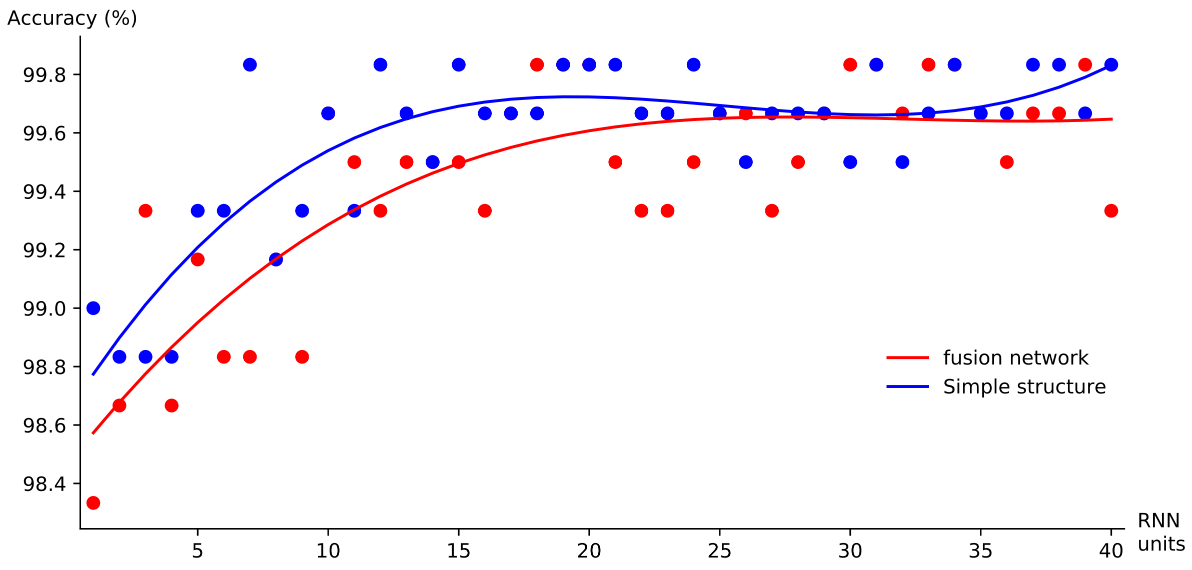


Figure 3.19: Performance comparison between BLSTM and simplified BRNN network in numeric gesture.

Graphs in Fig. 3.19 illustrate the results on a numeric gesture. Both graphs demonstrate the recognition capability of the fusion networks depended on the number of recurrent units, which is similar to the results in the previous section. When the number of RNN units is less than 15, the accuracy of the BLSTM fusion network is lower than the simple BRNN by around 0.2%. From these graphs, the recognition accuracy from the simple structure is better than the BLSTM fusion network. However, the performances of both structures are slightly different. The results on the alphabet gesture have been depicted in Fig. 3.20. The accuracy of the fusion framework increases relative to the number of recurrent units at the beginning. Then it maintains at 99.07% when the number of recurrent units is greater than 25. The results from the simple BRNN are quite similar to the previous experiments. The accuracy increases at the beginning and retains at 99.15% when the number of recurrent units equals to 25. Comparing the results from two fusion structures, the performance of the simple BRNN structure is greater than that of the BLSTM network at the beginning. The accuracy of the simple structure maintains when the number of the recurrent units equals to 15 while another still increases and stops at 25. However, the performance of the simplified networks is comparable with the BLSTM fusion structure.

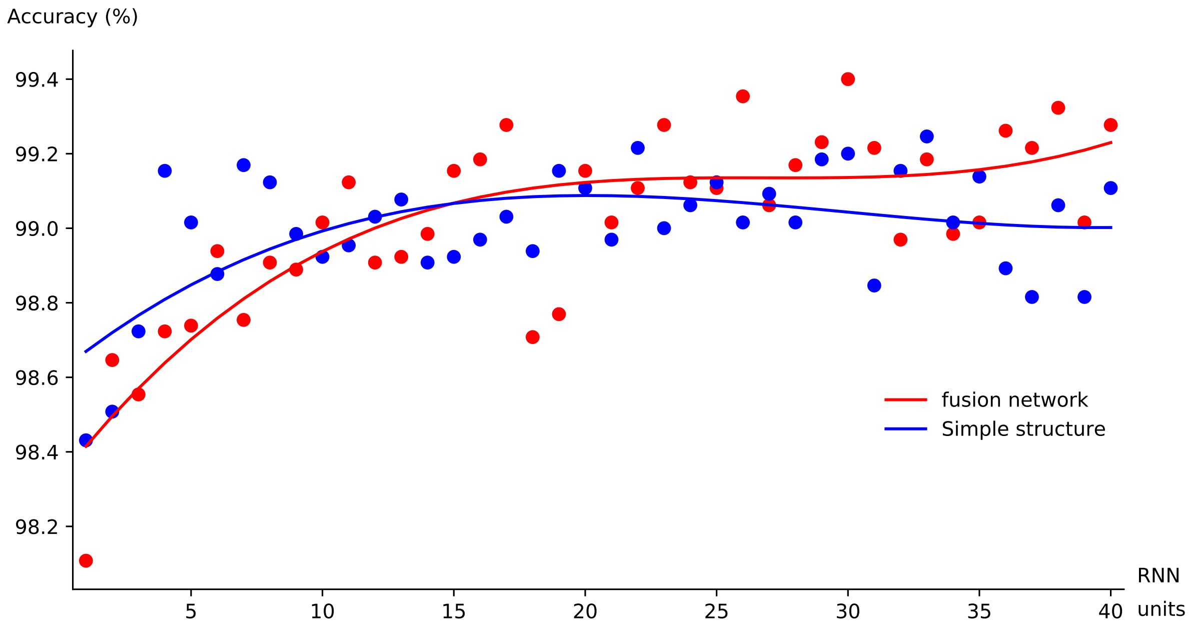


Figure 3.20: Performance comparison between BLSTM and simplified BRNN network in alphabet gesture.

To evaluate the cost of the BLSTM and simplified BRNN network, we run the experiments in an Intel Core i7 (3.4 GHz) Windows machine, and then record the average computation time and adaptive parameters in Table 3.2. These experiments are coded in Python with Tensorflow backend. The average predicting time of the BLSTM network is 5.57 milliseconds and 5.15 milliseconds on a numeric and alphabet gesture, respectively. For the simplified BRNN, the output gesture can be predicted within 2.98 milliseconds and 2.73 milliseconds in the numeric and alphabet gesture, respectively. When comparing the computation time of two networks, the training and testing time of the simplified network is reduced approximately to half. The computation time of both networks gave confidence that the proposed networks can be deployed in real-time applications. The results demonstrate that the adaptive parameters of the simplified BRNN network were reduced by 8.15% and 7.91% in numeric and alphabet gesture, respectively.

Table 3.2: Cost comparison between BLSTM and simplified BRNN.

Gesture	BLSTM			Simplified BRNN		
	Computation Time (ms)		Adaptive parameters	Computation Time (ms)		Adaptive parameters
	Train	Test		Train	Test	
Numeric	73.47	5.57	51,476	32.84	2.98	47,276
Alphabet	57.13	5.15	53,092	25.65	2.73	48,892

In the final part of this section, we examine the effect of the number of convolution maps in the fusion network. From the experiments in this section, the fusion networks reach high performance when the number of recurrent units is more than 25. We examined the fusion structures by choosing the number of BLSTM units of 25 and fixing the convolution kernel size at 5×5 pixels. The number of feature maps in the third convolution layer C_3 is varied from 1 to 50. Then the results are plotted in Fig. 3.21.

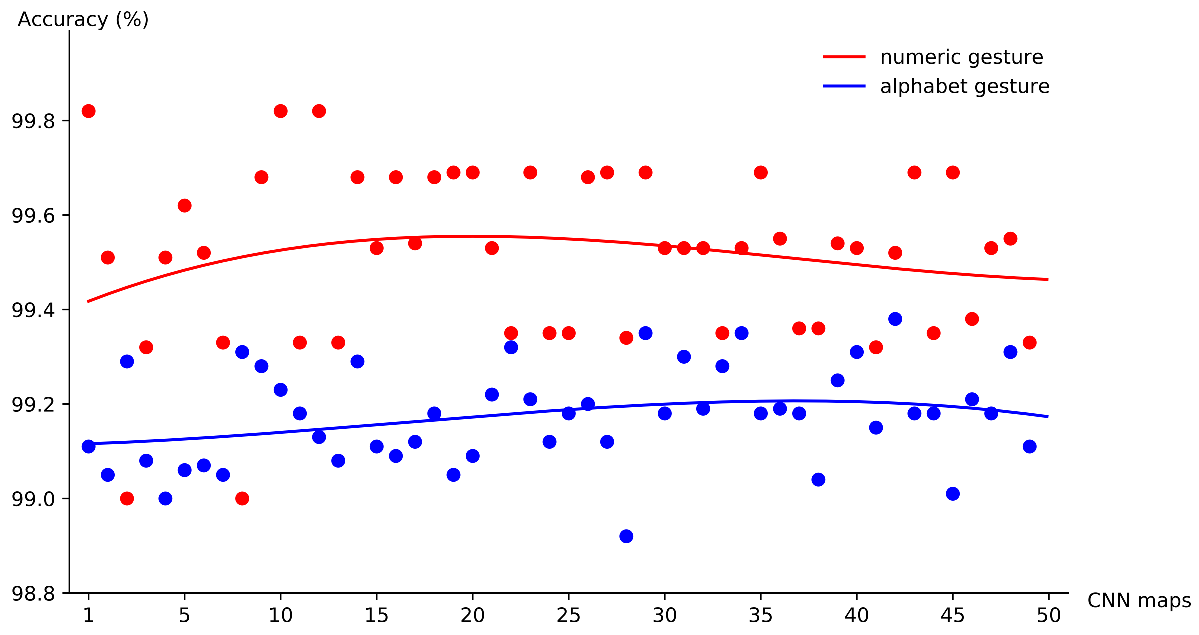


Figure 3.21: Effect of the number of convolution map in C_3 layer.

The graph illustrates the recognition rate of fusion networks. Although the number of recurrent units has been varied, the accuracy maintains around 99.18% and 99.5% for the alphabet and numeric gesture, respectively. Obviously, the average accuracy changes slightly in accordance with the number of convolution maps. We can simplify the fusion network by reducing the number of convolution feature maps in the third CNN layer without disturbing the overall performance of the proposed structures. From the graphs, the most appropriate number of BRNN units is 30. Although this method reduces computation time insignificantly, it can save memory usage for storing the trainable parameters.

3.4 Conclusion

In this chapter, we introduced a fusion framework for air-writing recognition by modeling the hand trajectory using both spatial and temporal features. The proposed structure comprises three main parts, i.e. CNN part, RNN part, and learning part. The CNN part is employed to extract information from the spatial feature. To obtain the information from temporal features, we deploy a BRNN in the RNN part. In the learning part, the extracted features from the CNN part and the RNN part are combined and fed to the fully connected neural networks. We have conducted the experiments on the 6DMG dataset. The results demonstrated the proposed networks achieve higher accuracy than other works. To investigate the performance of the proposed structure, we conduct three experiments in the following.

The first experiment is comparing the accuracy of the proposed structure with the state-of-the-art techniques. There are three techniques: CNN, BLSTM, and Yang's work have been selected as baseline references. The results confirmed that the accuracy of the proposed technique outperforms both CNN and BLSTM networks in the alphabet and numeric gesture. The best accuracy that we can achieve is 99.83% and 99.25% on the numeric and alphabet gesture, respectively. When comparing the results with Yang's work, the accuracy of the proposed network improves by 0.68% and 0.84% in the alphabet and numeric gesture, respectively. We have inspected more details of each class of gesture. The accuracy of the fusion structure is better than those of the CNN and the BLSTM in all classes of the numeric gesture. In the case of alphabet gesture, the recognition error of the fusion structure is lower than the CNN in all classes. By comparing the results of fusion structure with the BLSTM, the accuracy of the fusion structure is lower than the BLSTM technique in some classes. The most misclassified classes are the "P" and the "D" gesture.

In the second experiment, the effects of the recurrent unit in the fusion structure have been investigated by varying the number of BLSTM units from 1 to 40. We also study the three common features: the hand position, the angular, and the velocity features. From the results, the accuracy of the proposed structure improves while the BLSTM unit increases. It maintains when the number of BLSTM units are more than 15 and 25 for learning the numeric and alphabet gestures, respectively. We also demonstrated the fusion network did not learn more information even when adding other training features. In other words, using only hand position is adequate for the proposed structure.

Referring to the experiments in section 3.3.3 and section 3.3.4, training the proposed network needs high processing cost due to the complexity of the network structure in the RNN part. The last experiment, we reduce the computation time by simplifying the structure of the LSTM cell. A simple Bidirectional RNN was introduced to speed up the computation time for training and predicting. We found the fusion framework can learn the air-writing although using a simple structure of BRNN. When decreasing the convolution map in the third convolution layer, the memory usage was reduced. Even though the recurrent unit has been simplified, the accuracy of the fusion structure drops only by 0.33% and 0.07% for the numeric and alphabet gesture, respectively. From the experiments, the number of recurrent units in the fusion network should be set at 25, and the number of convolution maps in the third layer should be 30.

Chapter 4

Motion word recognition

Learning the motion word is more difficult than the motion character because each sample comprises not only gesture parts but also non-gesture parts. Moreover, there are ligatures between characters in the motion word. When considering the public datasets that we used for evaluating the proposed structure, the ground truth is labeled only writing and non-writing parts. In other words, there is no ground truth for the ligature in the dataset. To avoid data labelling issue, we deploy a segmentation free technique by adding a Connectionist Temporal Classification (CTC) loss in a decision layer. Instead of employing an image-like feature to represent the spatial information, we consider the path signature feature [98–102] which incurs lower computational cost than an image-like feature.

This chapter gives improvement of the fusion scheme for motion word recognition. The performance of the proposed structure was examined by two public datasets: the palm-writing dataset and the finger-writing dataset. Although we set the training parameters for the palm-writing dataset similar to the finger-writing dataset, the experiments were conducted separately. We first compared the performance of the fusion scheme with the baseline references: temporal modeling and spatial modeling. After that, the effect of the sliding window size was examined and summarized at the end of the chapter.

4.1 Proposed framework

The primary goal of this chapter is developing a technique to recognize the air-writing on a continuous motion stream. Recognizing the motion word is more complicated than the motion character because it lacks the starting and stopping information. Moreover, there are ligatures between characters in a word.

In this chapter, we considered a deep RNN [83] by stacking two layers of the LSTM units to learn information on the air-writing. Details of the proposed structure are illustrated in Fig. 4.1. The input feature comprises two types of features, i.e. temporal feature and spatial feature. In the case of temporal feature, we deploy only a hand position. For learning the spatial information, we considered a path signature feature which has proved to be sufficient for handwriting recognition. The main advantage of the path signature feature is easy to compute. In other words, employing the path signature feature can save computation time and memory usage. At each time step, a hand position feature and path signature feature are combined and fed to the proposed structure.

In the first layer, the hidden state is computed by using its previous state and the current input. The hidden state in the second layer is computed by using the previous state of the

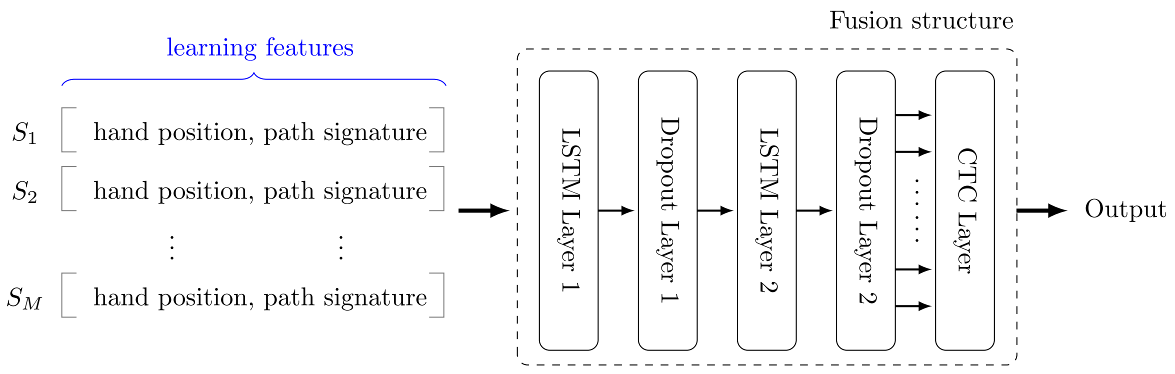

learning features
S_1 hand position, path signature
S_2 hand position, path signature
S_M hand position, path signature
Fusion structure
LSTM Layer 1
Dropout Layer 1
LSTM Layer 2
Dropout Layer 2
CTC Layer
Output

Figure 4.1: Proposed structure for motion word recognition.

second layer and the current state of the first layer. In the output layer, we employed the CTC loss [103] for mapping the air-writing information to a word label. The main advantage of using the CTC loss is removing a predefined alignment for creating the training set. Even though we can stack the RNN more than two layers for improving the learning capability, we balance the computational cost with the performance of the proposed structure.

In practice, the number of recurrent units with CTC loss should be more than ten times of the output label [103]. Referring to the information in section 2.6.2, the longest gesture label is “DISCOVERY” which consists of 9 characters. Thus, we set the number of recurrent units to 100 in all experiments. We avoid the over-fitting by applying a dropout technique, where the dropout value is 20% in all LSTM layers [104, 105]. More technically, the recurrent units are randomly removed 20% on each epoch in a training phase.

4.1.1 Deep recurrent neural network

The main structure for learning the motion word is a multilayer LSTM network. The state equations of each unit cell are computed by Equation (3.2) – (3.7). To extend the learning capability of the LSTM structure, we deploy a deep learning approach by stacking two LSTM layers. The hidden state of the first LSTM layer is fed directly to the input of the second LSTM layer. The output y_t of the proposed structure at each time step is computed by

$$y_t = W_y h_t + b_y, \tag{4.1}$$

where h_t is an internal state of the LSTM units in the second layer. W_y and b_y are a weight matrix and a bias vector in the output layer. For learning the air-writing without an explicit delimiter, we employ the CTC loss building on top of the LSTM layers. A computation graph of the proposed structure is illustrated in Fig 4.2.

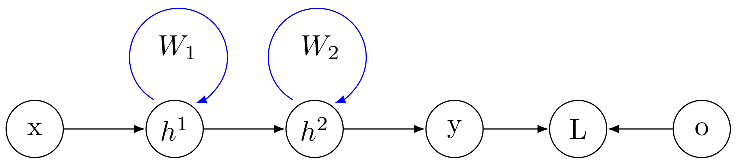


Figure 4.2: Computation graph of the proposed structure for motion word recognition.

There are two layers of the RNNs. In the first layer, the RNN generates the trainable parameters. Then, these parameters are fed to the second layer. By stacking the recurrent layer on top of the other layer, the learning capability of the recurrent structure improves [106, 107]. The hidden states in the first and second layers are computed by

$$h_t^1 = \mathcal{H}(W_{xh^1} x_t + W_{h^1h^1} h_{t-1}^1 + b_h^1), \tag{4.2}$$
$$h_t^2 = \mathcal{H}(W_{h^1h^2} h_t^2 + W_{h^2h^2} h_{t-1}^2 + b_h^2), \tag{4.3}$$

where $\mathcal{H}$ is a nonlinear function in the hidden state. The h_*^1 and h_*^2 are vectors of the hidden states in the first and second layer, respectively.

4.1.2 Connectionist temporal classification loss

The CTC loss was originally proposed by Grave et al. in 2005 [85]. Since then, it has been used successfully in many applications such as speech recognition, handwriting recognition, and phoneme recognition. The CTC introduces a new cost function for training unsegmented sequence directly. The additional blank symbol ($\langle$ *SPACE* $\rangle$) is included between the possible labels. By adding the blank symbol at any location in the target labels, the learning structure becomes more flexible. The learning structure gives a prediction at any moment when the output probability is highly confident. In other words, the blank symbol allows the learning structure to give a strong probability to the correct label at every point in time. The structure of the CTC in this work is illustrated in Fig. 4.3, where the circles represent a matrix of the conditional probability in the output layer. For instance, when the input features are fed to the proposed network, the proposed structure computes the highest probability at each time step. The word prediction is made by searching the path with the maximum probability. The arrows in this figure represent a CTC path, which is used for a prediction.

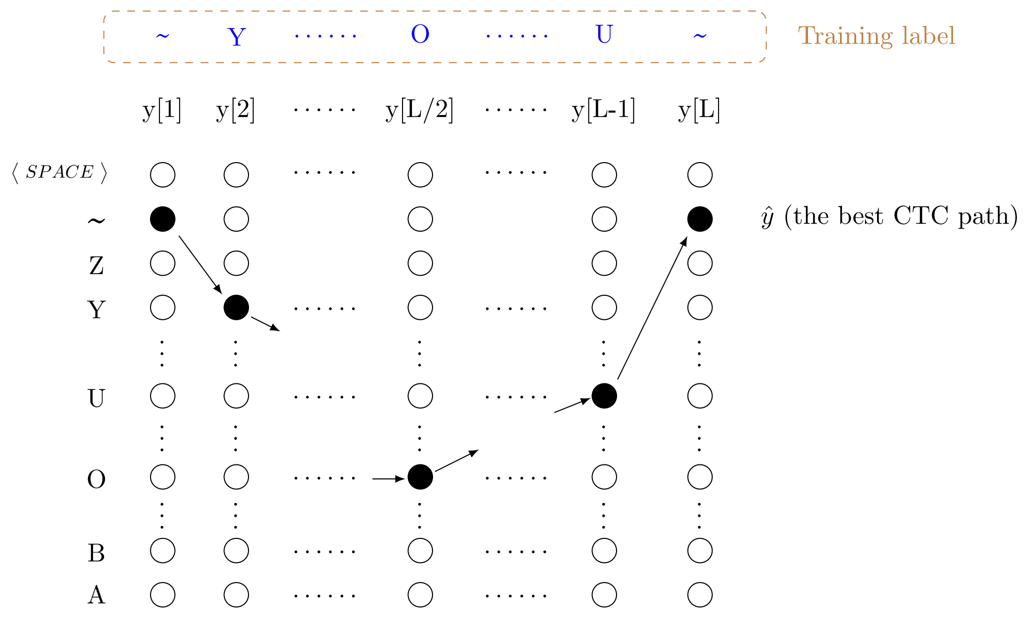


Figure 4.3: CTC structure in this work.

In the learning process, we introduce a unique character (~) for indicating non-writing event at the begin and at the end of the gesture. For instance, the label of the word “YOU” is "∼YOU∼". The unique character (∼) is used for delimiting the predicted words. The (∼) character is removed in the decoding process. Rather than predicting each label at a time, the output sequence was predicted as a set of possible labels. Let $y = \{y[1], y[2], \ldots, y[L]\}$ denotes the possible output labels, where $y[w]$ is a vector with all possible characters in the window w, and L is the number of LSTM units. The connecting line through the nodes at each time step is called a CTC path, which is represented by π. Assuming that the probabilities of CTC path at each time step are conditionally independent of given input x, the conditional probability of the correct output is given by

$$p(\pi|x) = \prod_{t=1}^{T} P(\pi_t|x). \tag{4.4}$$

The CTC loss (L_{CTC}) is defined as the summation of negative log probabilities of correct labels as

$$L_{CTC} = -\ln\left(\sum_{\pi} P(\pi|x)\right). \tag{4.5}$$

The CTC loss uses the Softmax function to separate the distribution of $p(\pi|x)$ at every window step along the input sequence. In this work, the distribution covers all upper case alphabet gesture plus word separation (∼) and extra blank symbol. The output of the CTC is presented by the matrix size $28 \times L$.

In the decoding process, we employed the Greedy algorithm [108, 109] for searching the best path. Then, the symbol with the highest conditional probability at each node was recorded. The probability of the output matrix was searched following the conditional dependency path. In other words, the best symbol was selected one at a time from left to right. The best path ($\hat{y}$) is computed by

$$\hat{y} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{T} \sum_{\pi} P(\pi|x_i; \theta), \tag{4.6}$$

where θ is a set of trainable parameters in the model. In finally, the extra blank symbol and the duplicate characters were removed from the output prediction.

4.2 Sliding window technique

In the preprocessing stage, we considered the sliding window technique to segment a long sequence of motion into small pieces. This technique is commonly used for converting a complicated learning problem into the simple supervised problem, to which the classical techniques can be applied. The sliding window technique is represented in Fig. 4.4, where W_* is the sliding window, and k is a skip size for the next window. The subscript t indicates the preprocessing process run at time step t. The sliding window technique was employed in the motion word recognition because starting and stopping points of the writing are unknown. When applying the sliding window technique, the size of the window should be seriously selected because it is the most critical parameter. If the motion data is split with an appropriate size of the sliding window, the proposed structure can capture both local and global information. In other words, deploying a long window can provide the global information while a short window gives local information. Based on analysis in section 2.6.2 and 2.6.3, writing

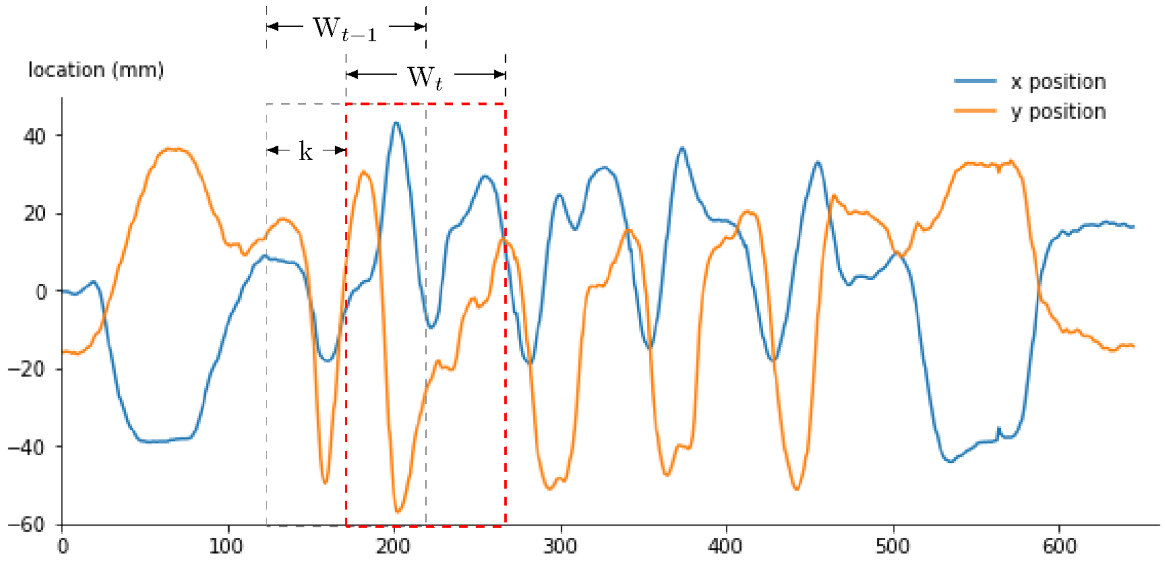


Figure 4.4: Sliding window technique.

duration per character, is set as the maximum size of the sliding window. Specifically, the maximum sizes of the sliding window are 0.88 seconds and 0.57 seconds for the palm-writing and finger-writing datasets, respectively.

The skip size is another parameter that should be carefully considered. It is defined as the number of time steps for skipping to the next window. This parameter helps to improve the speed of computation. In practical, the skip size should be less than 50% of the window size for maintaining the related information between the consecutive windows. In Chen's work [52], it is considered that the appropriate size of the sliding window is 1 second, and the step size is 167 milliseconds.

4.3 Feature extraction

For training the proposed structure, two features were deployed: hand position feature and path signature feature. The hand position feature comprises sequences of the hand position in the x- and y- directions. The hand position feature contains the temporal information, while the path signature embeds the analytic and geometric properties of the writing trajectory [110, 111]. For generating these features, the stream of motion was segmented into small pieces using the sliding window technique. The data of each segmented piece is called a hand position feature. To obtain information from the spatial space, we derived the path signature feature from each segmented data.

Even though the Leap motion sensor records the motion word in a three-dimensional space, we limit the trajectory representation in a two-dimensional coordinate for simplicity and comparability with other works. Therefore, each observed sample is described by a sequence of hand position as

$$P = \{\, \mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_N \,\}, \tag{4.7}$$

where $\mathrm{p}_i = \{p_{xi}, p_{yi}\}$ is the i^{th} instance of the hand position in x- and y-direction as p_x and p_y, respectively. N is the number of sampling points in each gesture.

Path signature feature is a sequence of numbers derived from a continuous path. It can be used to extract information about analytic and geometric properties of a data stream [98, 110]. Although the path signature feature is defined for a continuous path with bounded variation, it is easily calculated from a discrete path based on linear interpolation and Chen's identity [112]. We assume that the length of a discrete path is finite, and the consecutive points always connect with a straight line. The element of the path signature is generally expressed as

$$S(P)^{i_1,i_2,\dots,i_k}_{[0,T]} = \frac{1}{k!}\prod_{j=1}^{k}(P^{i_j}_T - P^{i_j}_0), \tag{4.8}$$

where $P^{i_j}_T$ is the i_j-th value of the path P at time T. $i_k \in \{1,\dots,d\}$ is an order of integral in d-dimensional path. The path signature level m consists of $\left(d^{m+1}/(d-1)\right) - 1$ value. This work, we set the level of integration (m) equal to 4. Because the finger position feature is a two-dimensional signal, the dimension of the path signature equals 31. In this work, the path signature feature within time interval $[0,T] \in \mathbb{R}$ is computed by

$$\begin{aligned} S_m(P)_{[0,T]} = \Big(&1, S(P)^{1}_{[0,T]}, S(P)^{2}_{[0,T]}, S(P)^{1,1}_{[0,T]}, S(P)^{1,2}_{[0,T]}, S(P)^{2,1}_{[0,T]}, S(P)^{2,2}_{[0,T]},\\ &S(P)^{1,1,1}_{[0,T]}, S(P)^{1,1,2}_{[0,T]}, S(P)^{1,2,1}_{[0,T]}, S(P)^{1,2,2}_{[0,T]}, S(P)^{2,1,1}_{[0,T]}, S(P)^{2,1,2}_{[0,T]},\\ &S(P)^{2,2,1}_{[0,T]}, S(P)^{2,2,2}_{[0,T]}, S(P)^{1,1,1,1}_{[0,T]}, S(P)^{1,1,1,2}_{[0,T]}, S(P)^{1,1,2,1}_{[0,T]}, S(P)^{1,1,2,2}_{[0,T]},\\ &S(P)^{1,2,1,1}_{[0,T]}, S(P)^{1,2,1,2}_{[0,T]}, S(P)^{1,2,2,1}_{[0,T]}, S(P)^{1,2,2,2}_{[0,T]}, S(P)^{2,1,1,1}_{[0,T]}, S(P)^{2,1,1,2}_{[0,T]},\\ &S(P)^{2,1,2,1}_{[0,T]}, S(P)^{2,1,2,2}_{[0,T]}, S(P)^{2,2,1,1}_{[0,T]}, S(P)^{2,2,1,2}_{[0,T]}, S(P)^{2,2,2,1}_{[0,T]}, S(P)^{2,2,2,2}_{[0,T]}\Big). \end{aligned} \tag{4.9}$$

Because the first term of the path signature feature equals to one; therefore, we removed it before merging the hand position feature with the path signature feature.

4.4 Experiments and results

In this section, we conduct the experiments on two public datasets [24]: palm-writing dataset and finger-writing dataset. These datasets were collected from a different scenario and captured using different tracking sensors. Therefore, the experiments were conducted separately. In the palm-writing dataset, the user's palm was tracked with optical and inertial sensors. The motion data was recorded by WorldViz PPT-X4 and Wiimote modules at 60 frames per second. Each sample contains motion data, including hand position, orientation, acceleration, and angular speed. As for the finger-writing dataset, 1k-word vocabularies are prepared, which include the most frequently used 1000 two-, three-, and four-letter words from the Google Web 1T dataset [52]. The samples were recorded at 60 frames per second by the Leap motion sensor.

The performance of the fusion framework was monitored using temporal modeling and spatial modeling as baseline references. We first conducted the experiment on the finger-writing dataset. The dataset samples are randomly split into a training and validating with the size of the training set is 90% while the rest is validating set. The CTC loss function was minimized by the mini-batch Gradient Descent algorithm [113]. The mini-batch size was set at 24, and the momentum of the learning parameter was set to 0.9. In the training process, the Adam optimization technique [97] was applied. The optimization parameters β_1 and β_2 used in this technique are 0.7 and 0.99, respectively. We firstly set the size of the sliding window at 1 second, and the skip size is fixed to 167 milliseconds. The proposed structure failed to learn the writing information because the proposed algorithm can not minimize the loss function. In the second attempt, we considered a smaller size of the sliding window. The size of the sliding window was set to 0.5 seconds, and the skip size is fixed at 83 milliseconds. The training history loss was recorded and plotted in Fig. 4.5.

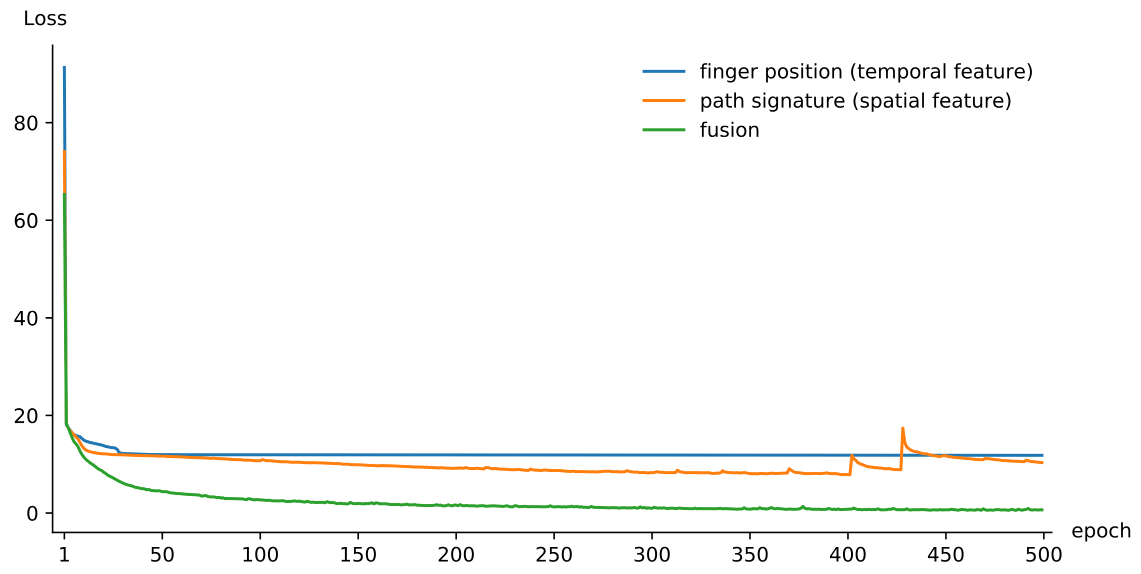


Figure 4.5: The learning rate on finger-writing dataset.

Graphs in Fig. 4.5 illustrate the learning capability of three models: spatial feature model, temporal feature model, and the proposed model. The spatial feature model learns the air-writing from the path signature feature, while the temporal feature model learns the writing information from the hand position in the time domain. After the training process runs 40 epochs, the losses of spatial and temporal models remain around 17, which is too high for learning information. In contrast, the proposed model can minimize the loss below 1. Moreover, the loss of the fusion structure converges within 250 epochs. To confirm the fusion technique can learn more information than the others, we plotted the validation error rate of three models in Fig. 4.6. When comparing the results from all models, the fusion framework achieves the lowest error rate at around 5%, while the validation errors of the others are around 40%. The results from both figures demonstrate that the training parameters of the fusion structure were set appropriately. The tuning and training parameters from this experiment were used for investigating the performance of the proposed structure.

In the next sections, we conduct the experiments on the palm-writing and the finger-writing dataset individually. The results on the palm-writing dataset and finger-writing dataset have been recorded in section 4.4.1 and section 4.4.2, respectively. For estimating the performance of the proposed model, we applied a ten-fold cross-validation technique. The proposed algorithm was coded in Python using the Tensorflow as the backend. To evaluate the computational complexity, the experiments were executed in an Intel Core i7 (3.4 GHz) Windows machine. We categorize the average results into three classes: correct prediction, imprecise precision, and false prediction. In the correct prediction, the prediction word completely matches with the ground truth label. In the case of imprecise prediction, we assume that only

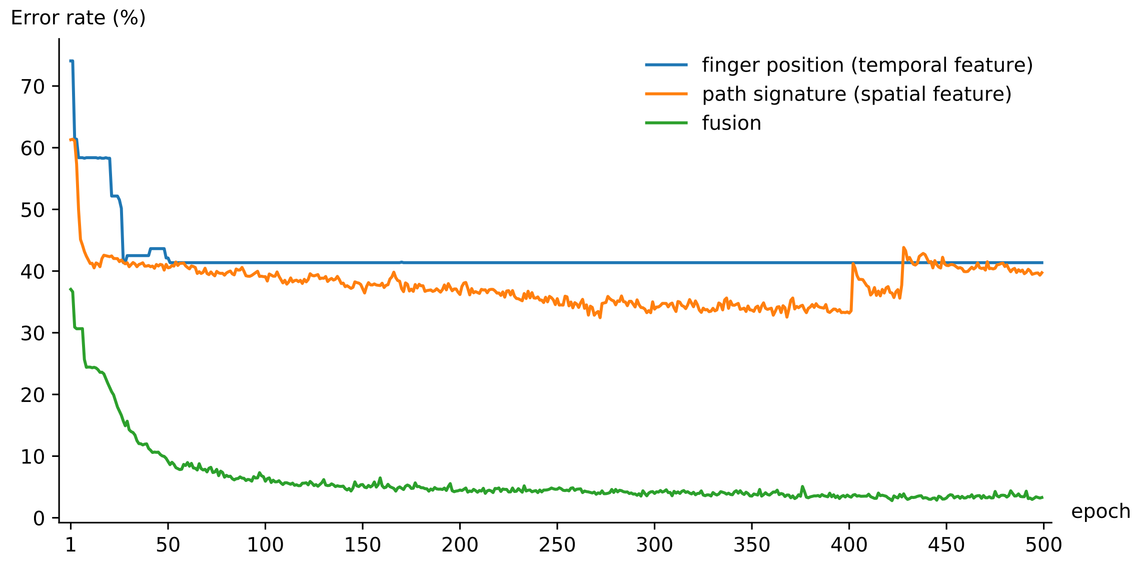


Figure 4.6: The error rates of validation set on finger-writing dataset.

one character is incorrect. Referring to the information in section 2.6.2, the average character per word is four characters. Accordingly, we set the threshold percentage for matching the prediction with the ground truth label more at 75%. If the prediction matches with the ground truth label more than 75%, it will be categorized in the imprecise prediction class. The rest will be categorized as the false prediction class.

4.4.1 Results on the palm-writing dataset

We first investigated the training loss of the proposed structure by varying the size of the sliding window in three steps: 0.25 seconds, 0.5 seconds, and 1 second. The results were recorded and plotted in Fig. 4.7. When the size of the sliding window equals to 0.25 seconds,

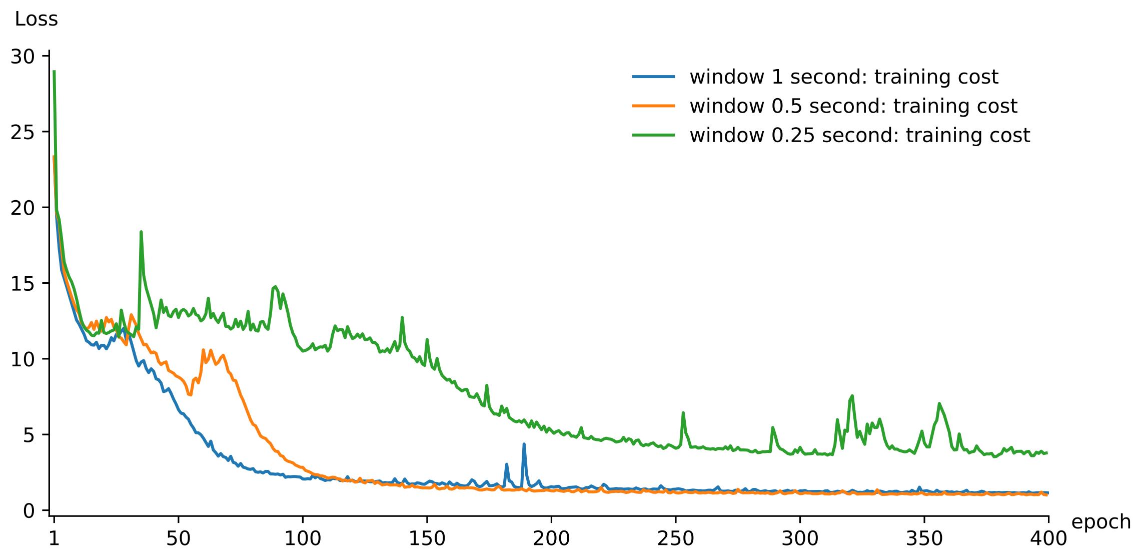


Figure 4.7: Training loss on palm-writing dataset at different window size.

the fusion structure can not learn useful information from the training features. The training loss remains around eight although the training process is executed over 400 epochs. In contrast, the training losses of other sets converge around 1. When the training process was executed more than 200 epoch, we found the window size of 0.5 seconds giving the better result. For more information, we plotted the training and validation error when the size of the sliding window was set at 0.25 seconds in Fig. 4.8.

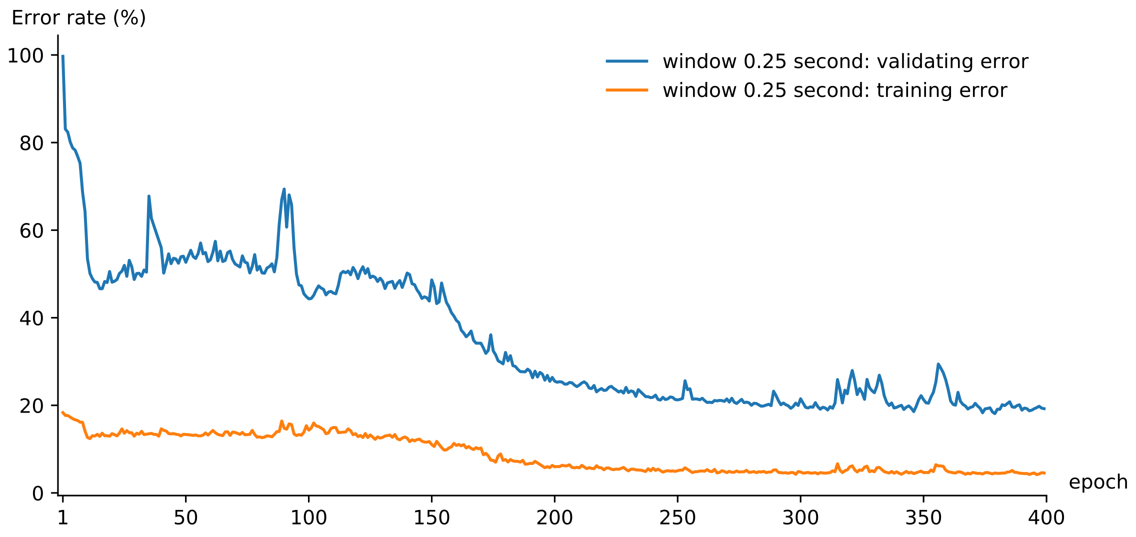


Figure 4.8: Error rate on palm-writing dataset window 0.25 seconds.

The graphs in Fig. 4.8 indicate the fusion structure may not learn more information because the gap between the training and validation error is large. Even though the training process is executed more than 400 epochs, the error rate on a validation set may not reduce. For a good interpretation, the results from ten-fold cross-validation were recorded and listed in Table 4.1. When considering the results in Table 4.1, the average of the correct prediction is 18.50%, and the false prediction is 67.42%. Total testing time in this table is computed over 466 samples. Thus, the testing time per word is 3.05 milliseconds.

Table 4.1: Recognition results on palm-writing dataset: window 0.25 seconds.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold1	26.55	14.35	59.10	33.18	1.48
fold2	4.71	4.07	91.22	33.12	1.42
fold3	10.06	17.34	72.59	33.46	1.37
fold4	37.26	24.84	37.90	33.19	1.41
fold5	7.51	15.88	76.61	33.60	1.35
fold6	6.22	8.80	84.98	33.67	1.44
fold7	9.23	13.95	76.82	33.33	1.50
fold8	39.91	15.24	44.85	33.46	1.46
fold9	33.91	14.38	51.72	33.28	1.53
fold10	9.64	11.99	78.37	33.83	1.43
average	18.50	14.08	67.42	33.41	1.42

In the second experiment, we increased the window size to 0.5 seconds while retaining the skip window at 83 milliseconds. The histories of training and validation errors were recorded and plotted in Fig. 4.9. It can be seen from the plot that the validation error tracks the training

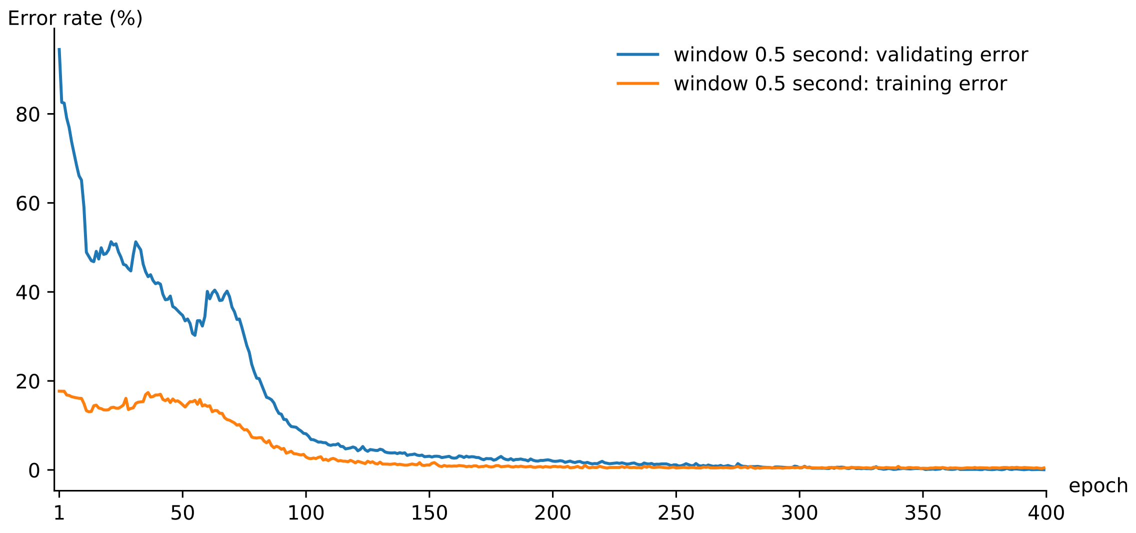


Figure 4.9: Error rate on palm-writing dataset window 0.5 seconds.

error after 200 epochs. In other words, the proposed structure can probably learn the air-writing information. Moreover, the validation error is not lower than the training error which means the model was not overlearned. Detailed result on each fold was recorded and listed in Table 4.2.

Table 4.2: Recognition results on palm-writing dataset : window 0.5 seconds.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold1	73.88	13.92	12.20	36.93	1.78
fold2	94.43	2.14	3.43	36.89	1.80
fold3	83.30	10.06	6.64	38.54	1.81
fold4	81.80	8.14	10.06	37.11	1.71
fold5	77.47	11.37	11.16	37.09	1.90
fold6	95.92	1.50	2.58	37.42	1.81
fold7	89.91	5.15	4.94	39.47	1.79
fold8	93.99	2.15	3.86	36.27	1.88
fold9	95.71	1.72	2.57	36.55	1.86
fold10	82.66	7.28	10.06	40.28	1.84
average	86.90	6.35	6.75	37.66	1.82

The average of correct prediction achieves 86.90%, while the imprecise and false predictions are 6.35% and 6.75%, respectively. When comparing with the previous experiment, the correct prediction improves by 68.4%, and the false prediction reduces by 60.67%. From the results in this table, the training time per epoch increases by 4.25 seconds because increasing the size of the sliding window makes the training feature bigger than the previous experiment. When considering the testing time, the total testing time of 466 samples is 1.82 seconds. Thus, average time per word is 3.91 milliseconds, which is larger than the previous experiment 0.86 milliseconds.

The last experiment in this section, we set the size of the sliding window to 1 second, which equals the value that was suggested by Chen [52]. Then, the error rates were recorded and plotted in Fig 4.10. From the graphs, the validating error is bigger than the training

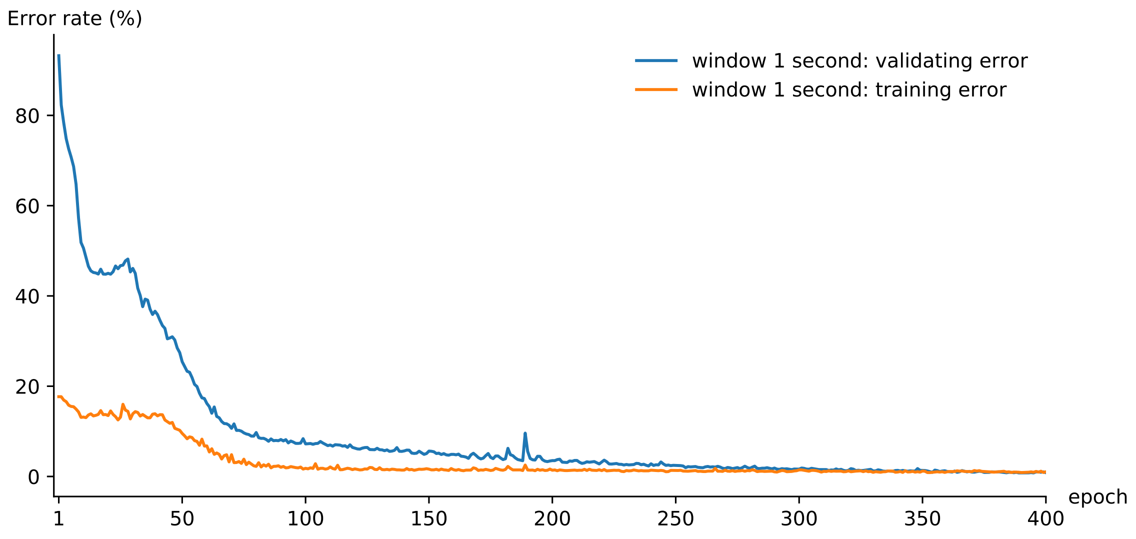


Figure 4.10: Error rate on palm-writing dataset window 1 second.

error which demonstrates the learning structure did not overlearn the information. For more information, we recorded the detailed results in Table 4.3.

Table 4.3: Recognition results on palm-writing dataset : window 1 second.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold1	38.76	20.13	41.11	44.36	1.99
fold2	89.72	4.50	5.78	45.21	2.03
fold3	92.72	2.36	4.92	44.30	1.78
fold4	37.90	24.41	37.69	44.78	1.95
fold5	32.19	17.81	50.00	45.57	1.77
fold6	93.35	1.50	5.15	44.80	2.01
fold7	88.20	2.58	9.22	45.16	1.88
fold8	53.43	22.10	24.46	45.99	1.84
fold9	55.36	18.02	26.62	44.10	1.85
fold10	34.26	23.77	41.97	45.20	1.89
average	61.59	13.72	24.69	44.95	1.90

From the results in Table 4.3, the correct prediction in some fold is small. For instance, the accuracies in fold number one, four, five, and ten are less than 40%. Even we retry to experiment with same fold many times, the results insignificantly change. The average of correct prediction is 61.59%, which is 25.31% smaller than the previous experiment. The imprecise and false predictions increase by 7.37% and 17.94%, respectively. The computation time for training in each epoch is 44.95 seconds, which increases by 7.29 seconds over the previous experiment. When considering the testing time, the average time per word is 4.08 milliseconds.

The experiments in this section demonstrate that the proposed structure adequately learns the air-writing information, where any language model is not necessary. By considering the proposed structure, the output prediction is generated at a single sliding window. When the sliding window captures input information less than one character, the proposed structure achieves high accuracy. Referring to the information in section 2.6.2, the minimum time of writing duration per character is 0.88 seconds. Therefore, the proposed structure recognized the motion gesture at 86.9% when the size of the sliding window equal to 0.5 seconds. When considering the computation time, both training and testing time depend on the size of the sliding window. In other words, increasing the size of the sliding window needs more processing time. When the size of sliding window equals to 0.5 seconds, the testing time per word gesture is 3.91 milliseconds. In any case, the computation of the proposed structure is adequate for a real-time application.

4.4.2 Results on the finger-writing dataset

In this section, we investigate the performance of the proposed structure on the finger-writing dataset. We first examine the learning capability of the proposed structure by varying the size of the sliding window in 3 steps: 0.25 seconds, 0.5 seconds, and 1 second. The skip size was fixed at 83 milliseconds in all experiments. The training and tuning parameters were set as the previous section. The training loss was recorded and plotted in Fig. 4.11. From this graph, the proposed structure partially learns the air-writing information when the size of the sliding window equals to 1 second. The training loss remains over 10, even though the number of training loop was set more than 400 epochs. When considering the training loss from the window size 0.5 seconds and 0.25 seconds, the graphs closely track. The lowest loss is stated when the window size equals 0.25 seconds. For more detail, the results from each window size are recorded and plotted.

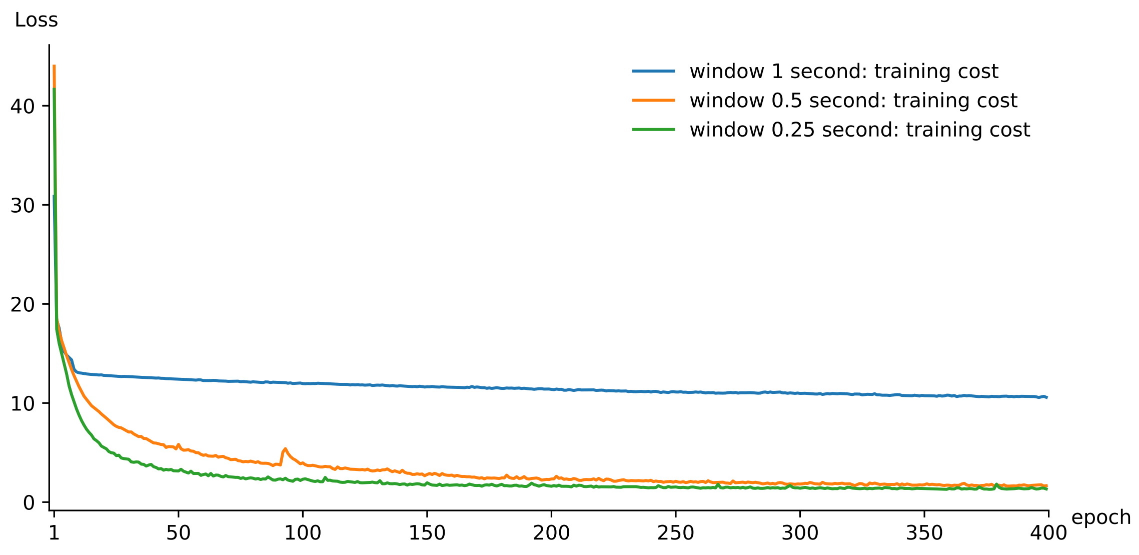


Figure 4.11: Training and validation loss on finger-writing dataset.

Graphs in Fig. 4.12 illustrate the results when the size of the sliding window equal to 0.25 seconds. The validation error dramatically decreases at the beginning and maintains at 3% after 200 epochs. While considering both graphs, the validation error rates track the training error rates well. This results confirm that the fusion structure did not have overlearned. For more information, we recorded the results and listed in Table 4.4. From the results in the

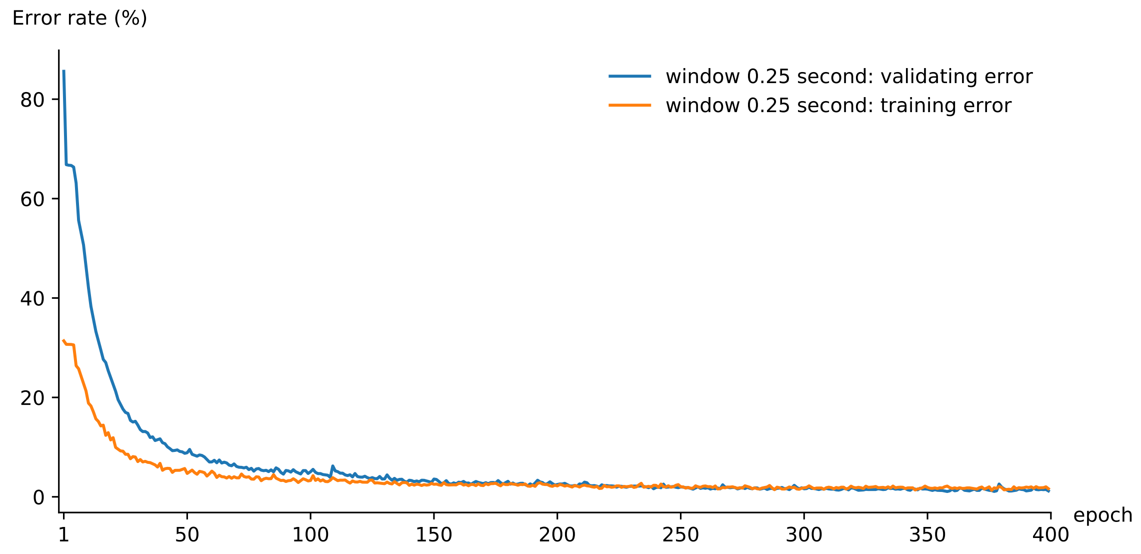


Figure 4.12: Error rate on finger dataset window 0.25 seconds.

table, the correct prediction in each fold ranges between 70.00% and 80.74%. The average of correct prediction over the ten-fold cross-validation is 75.81%. The imprecise and false predictions are 18.37% and 5.82%, respectively. When considering the computation time, the training time per epoch is 12.4 seconds while the prediction time is 6.37 milliseconds per word.

Table 4.4: Recognition results on finger-writing dataset: window 0.25 seconds.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold1	77.04	18.52	4.44	12.37	1.72
fold2	79.26	17.41	3.33	11.95	1.78
fold3	77.78	17.78	4.44	12.10	1.81
fold4	71.85	18.89	9.26	12.44	1.73
fold5	74.44	18.15	7.41	12.48	1.70
fold6	74.07	19.63	6.30	12.54	1.65
fold7	70.00	21.85	8.15	12.83	1.75
fold8	80.74	12.96	6.30	12.62	1.63
fold9	74.44	20.37	5.19	12.35	1.65
fold10	78.52	18.15	3.33	12.31	1.74
average	75.81	18.37	5.82	12.40	1.72

In the second experiment, the size of the sliding window was set at 0.5 seconds while the skip size remained at 83 milliseconds. The training and validation error rates were recorded and plotted in Fig 4.13. From these graphs, the training and validation errors decrease dramatically at the beginning, then stop at 5%. This result indicates that the proposed structure can learn the air-writing within 300 epochs. More detail about the performance of the prediction, the result on each fold was recorded and listed in Table 4.5. From this result, the

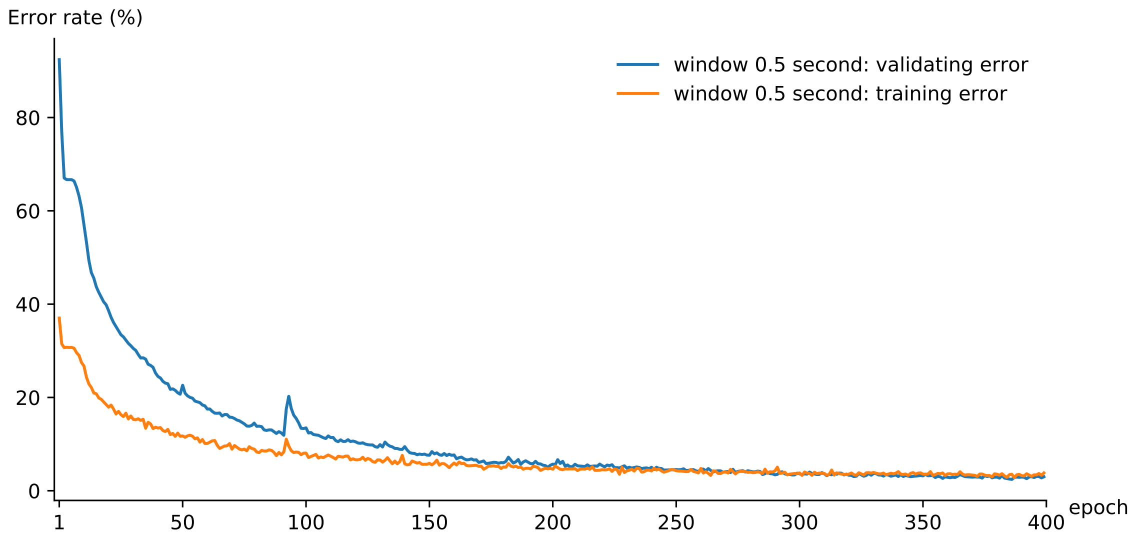


Figure 4.13: Error rate on finger dataset window 0.5 seconds.

correct prediction is 50.96%, which decreases by 24.85% when compared with the previous experiment. In contrast, the false prediction is 17.41%, which increases by 11.59%. In the case of imprecise prediction class, the average accuracy grows by 13.26%. While considering the computation time in the training process, the average time in each epoch is around 12.72 seconds. It grows by 0.3 seconds over the previous experiment. The testing time per word is 6.33 milliseconds.

Table 4.5: Recognition results on finger-writing dataset: window 0.5 seconds.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold 1	53.70	30.00	16.30	12.93	1.72
fold 2	45.18	33.33	21.49	12.95	1.73
fold 3	49.26	37.78	12.96	12.54	1.69
fold 4	42.59	32.59	24.82	12.66	1.71
fold 5	44.44	30.37	25.19	12.65	1.67
fold 6	46.30	35.55	18.15	12.60	1.68
fold 7	48.89	31.11	20.00	12.57	1.74
fold 8	57.99	29.37	12.64	12.70	1.68
fold 9	57.56	32.10	10.34	12.68	1.72
fold 10	63.70	24.07	12.23	12.73	1.73
average	50.96	31.63	17.41	12.70	1.71

The last experiment in this section, we set the size of the sliding window equal to 1 second. The other parameters were set as all previous experiments. The training process was executed for 400 epochs. The history of training and validation error rates were recorded and plotted in Fig. 4.14. From these graphs, the training and validation errors drop dramatically at the beginning. After the 5th epoch, the validation loss gently decreases while the training loss remains constant. For more detail, the results were recorded and listed in Table 4.6.

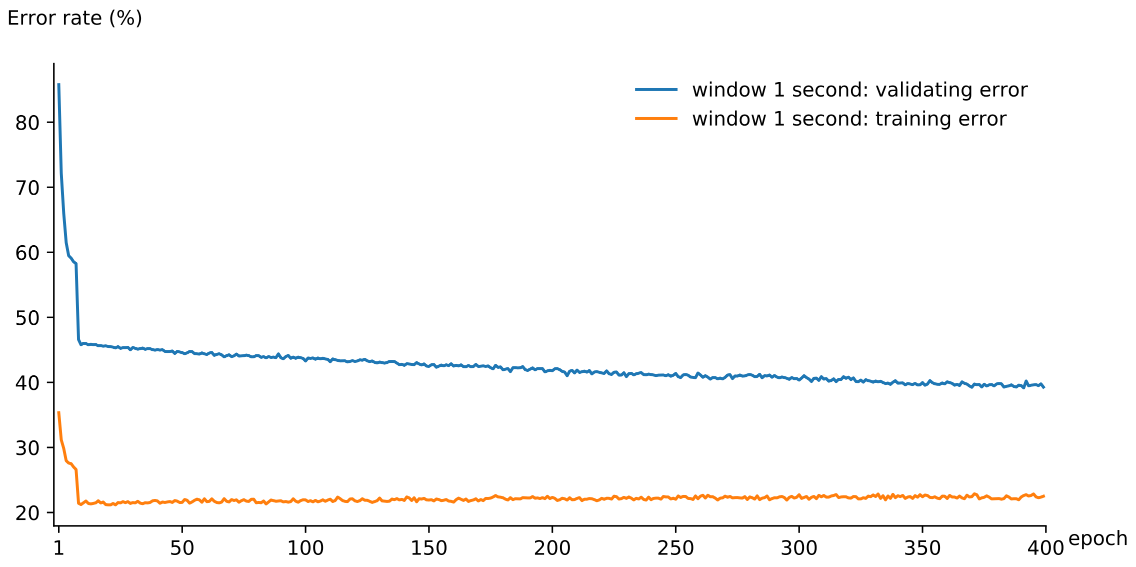


Figure 4.14: Error rate on finger dataset window 1 second.

From the results in Table 4.6, the correct and imprecise predictions are 0% and 0.52%, respectively. In contrast, the false prediction is 99.48%. From this results, we can conclude that the fusion structure completely fails to learn the air-writing when the size of the sliding window equals to 1 second.

Table 4.6: Recognition results on finger-writing dataset: window 1 second.

Fold number	Correct (%)	Imprecise (%)	False (%)	Computation time	
				Training (s)	Testing (s)
fold1	0.00	0.37	99.63	13.64	1.99
fold2	0.00	0.37	99.63	13.57	2.03
fold3	0.00	0.37	99.63	14.45	1.78
fold4	0.00	0.37	99.63	13.62	2.11
fold5	0.00	0.37	99.63	13.56	1.83
fold6	0.00	1.11	98.89	13.57	1.72
fold7	0.00	0.00	100.00	13.68	1.82
fold8	0.00	1.11	98.89	13.62	1.88
fold9	0.00	0.74	99.26	13.68	1.79
fold10	0.00	0.37	99.63	13.57	2.01
average	0.00	0.52	99.48	13.70	1.89

The experiments in this section demonstrate that the fusion structure achieves the best performance at 75.81% when the size of the sliding window equals to 0.25 seconds. By considering the proposed structure, the prediction character is generated one character per a sliding window. Therefore, the size of the sliding window must be smaller than the writing duration per character. This result is confirmed by the information in section 2.6.3, where the minimum time of writing duration per character is 0.57 seconds. When increasing the size of the sliding window, the proposed structure learns less information.

From analyzed information in section 2.6.3, the average of writing duration per character is 1.32 seconds. Therefore, the proposed structure may ultimately fail if the sliding window is larger than 1.32 seconds. When applying the proposed structure with other datasets, the average time of writing duration can be used to control the maximum size of the sliding window. When considering the computation time of the best result, the prediction time per word is 6.37 milliseconds. Accordingly, the proposed structure can be executed in real-time.

4.5 Conclusion

In this chapter, we proposed the fusion structure for a motion word recognition by considering the segmentation free technique. The fusion structure utilizes both spatial and temporal features to recognize the air-writing. There are two consecutive processes: the preprocessing and the recognition processes. In the preprocessing process, the stream of motion data was split into small segments using the sliding window technique. Then, the spatial and temporal features are derived from each segment before feeding to the proposed structure. By using the spatial model and temporal model as baseline references, the results from the first experiment confirmed that the proposed structure could learn more information. The most important parameter in the preprocessing process is the size of the sliding window. If it was set improperly, the fusion structure may ultimately fail to recognize the motion word. Even though we can avoid this problem by using the over-segmentation technique, it needs high processing cost. One goal in this chapter is finding an appropriate window size to recognize the motion word. We investigated the performance of the proposed structure on two public datasets: the palm-writing and the finger-writing datasets. Because the samples in each dataset were recorded with different scenarios, the experiments were conducted on each dataset separately.

In the first experiment, we investigated the performance of the proposed structure on the palm-writing dataset. The sizes of the sliding window were varied in three steps: 0.25 seconds, 0.5 seconds, and 1 second. From the experiments, the proposed structure can recognize unseen words up to 86.90% when the window size equals to 0.5 seconds. When changing the size of the sliding window to 0.25 seconds and 1 second, the correct predictions drop to 68.4% and 25.31%, respectively. From this result, we conclude that the most appropriate window size for recognizing the word dataset is 0.5 seconds. When considering the prediction time per word, the average time is 3.91 milliseconds. This result confirms that the proposed algorithm can be executed in real-time.

In the second experiment, we studied the performance of the fusion structure on the finger-writing dataset. First, the learning capability of the fusion structure was roughly monitored from the training history. The result confirmed that the training and tuning parameters, which are similar to the first experiment, are adequate for learning the finger-writing dataset. For more detail, the fusion structure was examined with ten-fold cross-validation by varying the size of the sliding window. The best recognition accuracy is 75.81% when the window size equals to 0.25 seconds. When the sizes of the sliding window change to 0.5 seconds and 1 second, the correct predictions drop 24.85% and 75.81%, respectively. When the size of the window is larger than 1 second, the fusion structure can not learn any information from the finger-writing dataset. We conclude that the most appropriate window size for recognizing the finger-writing dataset is 0.25 seconds. When considering the prediction time per a word, the average time is 6.37 milliseconds.

Learning the motion word using the sliding window technique, the size of the window is a critical parameter which is carefully considered. In this work, the output prediction of the proposed structure is generated one character per time step. Thus, each sliding window must capture input information less than one character. In other words, the size of sliding window is limited by the writing duration per one character. From the results in section 4.4.1 and section 4.4.2, the best result on each dataset is less than the average time writing duration. We confirmed that the average time of writing duration from section 2.6.2 and 2.6.3 can be used for setting the maximum size of a sliding window. The results from the palm-writing and finger-writing confirm that the proposed structure can be applied for the air-writing recognition. When considering the prediction time per word, the average times are 3.91 milliseconds and 6.97 milliseconds. Thus, the proposed structure can be executed in real-time.

Chapter 5

Conclusion

Human-computer interface (HCI) is a research area focusing on interaction modalities between human and computer. HCI devices have had many changes over time due to technological advancements. At the beginning of the computer era, users must adapt their behavior to fit HCI devices. A user primarily sends data and command to a computer via keyboard and mouse paradigm in the early years. After that, many sensors have been introduced to change the way people interact with devices. For instance, a touch screen and speech recognition technologies let user efficiently provide inputs. Currently, some sensors such as Microsoft kinect and Leap motion sensors further step forward the implementation of natural interfaces in which a human gesture becomes a controller. Even though we can employ a simple gesture to control a computing device, we can extend the communication capability by employing the air-writing recognition technology.

The air-writing is a type of dynamic hand gesture. It refers to writing the alphabet or numeric gestures by hand or finger movement in free space. The air-writing has attracted attention since it can offer the verbal communication, which is suitable for a short-text input interface. By writing a character in the air, the air-writing is mainly different from the surface-based writing in three aspects. First, a user is free for writing cause the air-writing did not have a concrete anchoring and reference position. Second, there is no relationship between the adjacent characters in the spatial domain. When the user performs the air-writing without control the hand or finger movement, the writing may lay in a non-aligned text. Finally, the user can not pause while writing which causes writing mixed between gesture and non-gesture. The captured data is represented in a single stroke without gaps between the consecutive gestures. Although many techniques have been proposed more than three decades, there is a room for improving the performance of air-writing recognition technique.

As summarized in chapter 2, most of the previous studies model air-writing by either the spatial features or temporal features. Even though these models work well with a simple gesture, the recognition accuracy may be vulnerable when applied with the motion character. This work proposed the fusion scheme for air-writing recognition by modeling the air-writing with the temporal features augmented with the spatial features to improve the recognition accuracy. We addressed the design into two categories: motion character recognition and motion word recognition. The underlying assumption of motion character is the gesture was correctly spotted. In other words, the meaningless motion was removed in advance; thus, the segmentation process is not necessary for learning the motion character. In contrast, the motion word is captured from a user in a continuous stream. It does not have a sign to indicate the writing and non-writing part, which is more complicated than the motion character approach.

In chapter 3, the fusion structure for learning the motion character was developed from the Yang's work. Instead of modeling the spatial and temporal feature separately as Yang's work, we employ fully connected neural networks for learning both types of features. For a fair comparison, the training parameters in the CNN part were set similar to Yang's work. We first investigated the performance of the proposed structure using the CNN, BLSTM, and Yang's work as baseline references. The results confirmed that the fusion scheme could improve the recognition accuracy. When comparing the results with Yang's work, the accuracies of the proposed network improve by 0.68% and 0.84% in the alphabet and numeric gesture, respectively. Then, we examined the effect of the recurrent unit in the fusion structure by varying the number of BLSTM unit in the RNN part. The optimum number of the BLSTM units are 15 and 25 for the numeric gesture and alphabet gesture, respectively. By substituting the simplified BRNN layer for the BLSTM layer, the results illustrated that the computation time reduced to half while the accuracies dropped by 0.33% and 0.07% for the numeric and alphabet gestures, respectively. In the last experiment, we investigated the effect of the convolution map in the third layer of the CNN part. From the experiment, the optimum number of convolution map in the third layer is 30. Although this method reduces computation time insignificantly, it saves memory used for storing the trainable parameters. We also demonstrated that the fusion network did not learn more information even adding more training features. In other words, using the hand position feature augmented with an image-like feature is adequate for learning the motion character recognition.

In chapter 4, we improve the fusion scheme for a motion word recognition. We considered a deep RNN with the CTC loss for mapping the air-writing trajectory into a word. The main advantage of this approach is removing a predefined alignment for generating the training set. In the preprocessing stage, we employed the sliding window technique where the size of the sliding window was carefully selected. For evaluating the performance of the proposed structure, we conducted the experiments on two public datasets: palm-writing and finger-writing datasets. Even though these datasets were recorded by a vision-sensor approach, the characteristics of the samples were different. The palm-writing dataset was recorded by WorldViz PPT-X4 and Wiimote modules using a palm as the tracking point. The writing area is 110 time 106 centimeters. In the case of the finger-writing dataset, the samples were recorded by the Leap motion sensor. The size of the writing space is 37 times 21 centimeters, which is 7% of the palm-writing area. Even other works suggest that the size of the sliding window should be 1 second, we studied the effect of the window size of each type of gesture separately. From the experiments, the best recognition accuracies on the palm-writing dataset and the finger-writing dataset are 86.90% and 75.81%, respectively. The appropriate window size for the palm-writing and the finger-writing datasets are 0.5 seconds and 0.25 seconds, respectively. These results illustrate that the proper size of a sliding window must be smaller than the average duration as analyzed in chapter 2. To improve the recognition accuracy of the proposed structure, a language model will be integrated into the CTC layer as mention in the literature [28,52,82]. When considering the prediction time per word on the palm-writing and the finger-writing datasets are 3.91 milliseconds and 6.37 milliseconds, respectively. These results confirm that the proposed algorithm can be executed in a real-time.

From the above results, we concluded that the proposed structure achieves higher accuracy than either spatial model or temporal model. The proposed structure recognizes the motion word by using multi-layers of the RNN with the CTC algorithm. The main advantage of the

CTC algorithm is removing the predefined alignment in the training set. Therefore, a specific signed is not necessary for the segmentation process. Another advantage is speeding up the execution time. When considering the sliding window technique, the size of the sliding window is seriously selected. This issue will be further studied in the future. If the size of the sliding window is not correctly, the learning structure may not learn any information. One direction to overcome this issue is employing the over-segmentation technique. By segmenting the motion data into multiple sizes of window, the leaning structure can acquire more information. Even though this technique can improve the recognition accuracy, increasing the number of learning features needs high processing cost. We also demonstrated that the writing duration was related to the effective size of the sliding window. The minimum size of writing duration per word may be used to bound the maximum size of the sliding window. Another direction to overcome the window selection issue is training the motion word with an unsupervised algorithm similar to Simao's work [114]. Learning the air-writing using unsupervised techniques may increase the execution time.

In the future, we will extend and refine a fusion framework for a real-life application. We will focus a study on the motion word recognition where a dataset contains both alphabet characters and numbers. One direction is adding some constraints such as language models and the lexicon searching to improve the recognition accuracy. By limiting the number of writing words, the accuracy may improve while the execution time remains. Another direction is refining the fusion structure by adding more LSTM layers, which may improve the recognition accuracy. Increasing the number of LSTM layers will increase the execution time. Thus the relation between the number of LSTM layers and the execution time will be studied. Last but not least, the usability study is another point that will be conducted to evaluate the user experience.

Bibliography

[1] N. Sebe, M. S. Lew, and T. S. Huang, “The State-of-the-Art in Human-Computer Interaction,” in *Proc. Computer Vision in Human-Computer Interaction, Lecture Notes in Computer Science*, vol. 3058, pp. 1 – 6, 2004.

[2] F. Zhangjie, J. Xu, Z. Zhu, A. X. Liu, and X. Sun, “Writing in the Air with WiFi Signals for Virtual Reality Devices,” in *IEEE Trans. on Mobile Computing*, pp. 1 – 12, May 2018.

[3] K. M. Sagayam and D. J. Hemanth, “Hand posture and gesture recognition techniques for virtual reality applications: a survey,” in *Virtual reality*, vol. 21, no. 2, pp. 91 – 107, Jun. 2017.

[4] P. Drotar, J. Mekyskaa, I. Rektorovb, L. Masarovab, Z. Smekal, and M. F. Zanuy, “Analysis of in-air movement in handwriting: A novel marker for Parkinson’s disease,” in *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, pp. 405 – 411, Dec. 2014.

[5] H. D. Yang, A. Y. Park, and S. W. Lee, “Gesture Spotting and Recognition for Human-Robot Interaction,” in *IEEE Trans. on Robotics*, vol. 23, no. 2, pp. 256 – 270, Apr. 2007.

[6] S. Cheema, M. Hoffman, and J. J. LaViola, “3D gesture classification with linear acceleration and angular velocity sensing devices for video games,” in *Entertainment Computing*, vol. 4, no. 1, pp. 11 – 24, Feb. 2013.

[7] X. Zhang, Z. Ye, L. Jin, Z. Feng, and S. Xu, “A New Writing Experience: Finger Writing in the Air Using a Kinect Sensor,” in *IEEE MultiMedia*, vol. 20, no. 4, pp. 85 – 93, Nov. 2013.

[8] Y. Lan, J. Li, and Z. Ju, “Data fusion-based real-time hand gesture recognition with Kinect V2,” in *Proc. Int. Conf. on Human System Interactions*, pp. 307 – 310, Jul. 2016.

[9] N. Xu, W. Wang, and X. Xu, “On-line Sample Generation for In-air Written Chinese Character Recognition Based on Leap Motion Controller,” in *Pacific Rim Conf. on Multimedia PCM 2015, Advances in Multimedia Information Processing, Lecture Notes in Computer Science*, vol. 9314, pp. 171 – 180, Nov. 2015.

[10] C. Agarwal, “Segmentation and recognition of text written in 3d using leap motion interface,” in *Proc. 3rd Int. Conf. on Pattern Recognition*, pp. 539 – 543, Nov. 2015.

[11] K. R. Moser and J. E. Swan, “Evaluation of hand and stylus based calibration for optical see-through head-mounted displays using leap motion,” in *Proc. The 2016 IEEE Virtual Reality*, pp. 233 – 234, Mar. 2016.

[12] F. Zhang, S. Chu, R. Pan, N. Ji, and L. Xi, “Double hand-gesture interaction for walk-through in VR environment,” in *Proc. 2017 IEEE/ACIS 16th Int. Conf. on Computer and Information Science*, pp. 539 – 544, May 2017.

[13] S. P. Priyal and P. K. Bora, “A robust static hand gesture recognition system using

geometry based normalizations and Krawtchouk moments," in *Pattern Recognition*, vol. 46, no. 8, pp. 2202 – 2219, Aug. 2013.

[14] H. A. Jalab, "Static Hand Gesture Recognition for Human Computer Interaction," in *Information Technology Journal*, vol. 11, no. 9, pp. 1265 – 1271, 2012.

[15] M. A. Aowal, A. S. Zaman, S. M. M. Rahman, and D. Hatzinakos, "Static hand gesture recognition using discriminative 2D Zernike moments," in *Proc. TENCON 2014 - 2014 IEEE Region 10 Conference*, pp. 1 – 5, Oct. 2014.

[16] G. Plouffe and A. Cretu,"Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping," in *IEEE Trans. on Instrumentation and Measurement*, vol. 65, no. 2, pp. 305 – 316, Feb. 2016.

[17] M. Yeasin and S. Chaudhuri, "Visual understanding of dynamic hand gestures," in *Pattern Recognition*, vol. 33, no. 11, pp. 1805 – 1817, Nov. 2000.

[18] M. M. Luna, T. P. Carvalho, F. A. A. M. N. Soares, H. A. D. Nascimento, and R. M. Costa, "Wrist Player: A Smartwatch Gesture Controller for Smart TVs," in *Proc. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 336 – 341, Jul. 2017.

[19] M. Chen, G. AlRegib, and B. H. Juang, "Air-writing recognition-part I: modeling and recognition of characters, words, and connecting motions," in *IEEE Trans. on Human-Machine Systems*, vol. 46, no. 3, pp. 403 – 413, Jun. 2016.

[20] C. Yang, B. Ku, D. K. Han, and H. Ko, "Alpha-numeric hand gesture recognition based on fusion of spatial feature modelling and temporal feature modelling," in *Electronics Letters*, vol. 52, no. 20, pp. 1679 – 1681, Sept. 2016.

[21] H. Doan, H. Vu, and T. Tran, "Dynamic hand gesture recognition from cyclical hand pattern," in *Proc. 2017 Fifteenth IAPR Int. Conf. on Machine Vision Applications*, pp. 97 – 100. May 2017.

[22] S. Ruffieux, D. Lalanne, E. Mugellini, and O. A. Khaled, "A Survey of Datasets for Human Gesture Recognition," in *Proc. Int. Conf. on Human-Computer Interaction Advanced Interaction Modalities and Techniques, Lecture Notes in Computer Science*, vol. 8511. pp. 337 – 348, 2014.

[23] T. Asano and S. Honda, "Visual interface system by character handwriting gestures in the air," in *Proc. Int. Symp. in Robot and Human Interactive Communication*, pp. 56 – 61, Sept. 2010.

[24] M. Chen, G. AlRegib, and B. Juang, "6DMG: A New 6D Motion Gesture Database," in *Proc. The 2nd ACM Multimedia Systems Conference*, pp. 83 – 88, Feb. 2012.

[25] L. Kane and P. Khanna, "Vision-based mid-air unistroke character input using polar signatures," in *IEEE Trans. on Human-Machine Systems*, vol. 47, pp. 1077 – 1088, Dec. 2017.

[26] P. Kumar, R. Saini, S. K. Behera, D. P. Dogra, and P. P. Roy, "Real-time recognition of sign language gestures and air-writing using leap motion," in *Proc. 15th Int. Conf. on Machine Vision Applications*, pp. 157 – 160, May 2017.

[27] J. K. Sharma, R. Gupta, V. K. Pathak, "Numeral gesture recognition using leap motion sensor," in *Proc. Int. Conf. on Computational Intelligence and Communication Networks*, pp. 411 – 414, Dec. 2015.

[28] P. Kumar, R. Saini, P. P. Roy, and D. P. Dogra, " Study of text segmentation and recognition using leap motion sensor," in *IEEE Sensors Journal*, vol. 17, no. 5, pp. 1293 – 1301, Dec. 2017.

[29] X. Qu, W. Wang, K. Lu, and J. Zhou, "In-air handwritten Chinese character recognition with locality-sensitive sparse representation toward optimized prototype classifier," in *Pattern Recognition*, vol. 78, pp. 267 – 276, June 2018.

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

[31] H. Abualola, H. A. Ghothani, A. N. Eddin, N. Almoosa, K. Poon, "Flexible gesture recognition using wearable inertial sensors," in *Proc. IEEE 59th Int. Midwest Symp. on Circuits and Systems*, pp. 1 – 4, Oct. 2016.

[32] P. Kumar, S. S. Rautaray, A. Agrawal, "Hand data glove: A new generation real-time mouse for Human-Computer Interaction," in *Proc. Int. Conf. on Recent Advances in Information Technology*, pp. 750 – 755, Mar. 2012.

[33] C. Amma, M. Georgi, and T. Schultz, "Airwriting: a wearable handwriting recognition system," in *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 191 – 203, Jan. 2014.

[34] S. Berman and H. Stern, "Sensors for Gesture Recognition Systems," in *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, vol. 42 , no. 3, pp. 277 – 290 , May 2012.

[35] J. H. Kim, N. D. Thang, and T. S. Kim, "3-D hand motion tracking and gesture recognition using a data glove," in *Proc. IEEE Int. Symp. on Industrial Electronics*, pp. 1013 – 1018, Jul. 2009.

[36] 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. 16th Int. Symp. on Wearable Computers*, pp. 52 – 59, Jun. 2012.

[37] N. C. Kiliboz and U. Gudukbay, "A hand gesture recognition technique for human-computer interaction," in *Journal of Visual Communication and Image Representation*, vol. 28, pp. 97 – 104, Apr. 2015.

[38] S. Patil, D. Kim, S. Park, and Y. Chai, "Handwriting Recognition in Free Space Using WIMU-Based Hand Motion Analysis," in *Journal of Sensors*, vol. 2016, pp. 1 – 10, June 2016.

[39] R. Agrawal and N. Gupta, "Real Time Hand Gesture Recognition for Human Computer Interaction" in *Proc. IEEE 6th Int. Conf. on Advanced Computing*, pp. 470 – 475, Feb. 2016.

[40] Z. Feng, S. Xu, X. Zhang, L. Jin, Z. Ye, and W. Yang, "Real-time fingertip tracking and detection using Kinect depth sensor for a new writing-in-the air system," in *Proc. Int. Conf. on Internet Multimedia Computing and Service*, pp. 70 – 74, Sept. 2012.

[41] T. Murata and J. Shin, "Hand Gesture and Character Recognition Based on Kinect Sensor," in *Int. Journal of Distributed Sensor Networks*, vol. 10, no. 7, pp. 1 – 6, Jul. 2014.

[42] P. S. Akhil and A. M. Parimi, "Synchronization of motion from multiple humans and humanoid robot based on kinect V2," in *Proc. 2016 11th Int. Conf. on Industrial and Information Systems (ICIIS)*, pp. 399 – 403, Oct. 2016.

[43] E. Lachat, H. Macher, M.-A. Mittet, T. Landes, and P. Grussenmeyer, "First Experiences with Kinect v2 Sensor for Close Range 3d Modelling," in *Proc. ISPRS - Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 93 – 100, Feb. 2015.

[44] B. Kheli and H. Amiri, "Hand Gesture Recognition Using Leap Motion Controller

for Recognition of Arabic Sign Language," in *3rd Int. Conf. on Automation, Control, Engineering and Computer Science*, pp. 233 – 238, Mar. 2016.

[45] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation", in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1685 – 1699, Sept. 2009.

[46] S. Poularakis and I. Katsavounidis, "Low-Complexity Hand Gesture Recognition System for Continuous Streams of Digits and Letters," in *IEEE Trans. on Cybernetics*, vol. 46, no. 9, pp. 2094 – 2108, Sept. 2016.

[47] T. Chiang and C. P. Fan, "3D Depth Information Based 2D Low-Complexity Hand Posture and Gesture Recognition Design for Human Computer Interactions," in *Proc. Int. Conf. on Computer and Communication Systems*, pp. 233 – 238, Apr. 2018.

[48] Y. LeCun, "Neural networks and gradient-based learning in OCR," in *Proc. The 1997 IEEE Workshop Neural Networks for Signal Processing*, pp. 255 – 255, Sept. 1997.

[49] J. T. Hu, C. X. Fan, and Y. Ming, "Trajectory image based dynamic gesture recognition with convolutional neural networks," in *Proc. 15th Int. Conf. on Control, Automation and Systems*, pp. 1885 – 1889, Oct. 2015.

[50] J. Du, Z. Wang, J. Zhai, and J. Hu, "Deep neural network based hidden Markov model for offline handwritten Chinese text recognition," in *Proc. Int. Conf. on Pattern Recognition*, pp. 3428 – 3433, Dec. 2016.

[51] H. Yoon, J. Soh, Y. J. Bae, and H. S. Yang, "Hand gesture recognition using combined features of location, angle and velocity," in *Pattern Recognition*, vol. 34, no. 7, pp. 1491 – 1501, 2001.

[52] M. Chen, G. AlRegib, and B. H. Juang, " Air-writing recognition-part II: detection and recognition of writing activity in continuous stream of motion data," in *IEEE Trans. on Human-Machine Systems*, vol. 46, no. 3, pp. 436 – 444, Jun. 2016.

[53] M. Z. Hameed and G. G. Hernando, "Novel spatio-temporal features for fingertip writing recognition in egocentric viewpoint," in *Proc. 14th Int. Conf. on Machine Vision Applications*, pp. 484 – 488, May 2015.

[54] S. Xu and Y. Xue, "Air-writing characters modelling and recognition on modified CHMM," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 1510 – 1513, Oct. 2016.

[55] Y. L. Hsu, C. L. Chu, Y. J. Tsai, and J. S. Wang, "An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition," in *IEEE Sensors Journal*, vol. 15, no. 1, pp. 154 – 163, Jan. 2015.

[56] T. H. Tsai, J. W. Hsieh, H. C. Chen, and S. C. Huang, "Reverse time ordered stroke context for air-writing recognition," in *Proc. Int. Conf. on Ubi-media Computing and Workshops*, pp. 1 – 6, Aug. 2017.

[57] S. Athavale and M. Deshmukh, "Dynamic Hand Gesture Recognition for Human Computer interaction: A Comparative Study," in *Int. Journal of Engineering Research and General Science*, vol. 2, no. 2, pp. 38 – 55, Mar 2014.

[58] J. Tang, H. Cheng, Y. Zhao, and H. Guo, "Structured dynamic time warping for continuous hand trajectory gesture recognition," in *Pattern Recognition*, vol. 80, pp. 21 – 31, Aug. 2018.

[59] M. K. Bhuyan, D. A. Kumar, K. F. MacDorman, and Y. Iwahori, "A novel set of features for continuous hand gesture recognition," in *Journal on Multimodal User Interfaces*, vol. 8, no. 4, pp. 333 – 343, Oct. 2014.

[60] M. R. Malgireddy, J. J. Corso, S. Setlur, V. Govindaraju and D. Mandalapu, “A Framework for Hand Gesture Recognition and Spotting Using Sub-gesture Modeling,” in *Proc. Int. Conf. on Pattern Recognition*, pp. 3780 – 3783, Aug. 2010

[61] H. K. Lee and J. H. Kim, “An HMM-Based Threshold Model Approach for Gesture Recognition,”’ in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961 – 973, Oct. 1999.

[62] M. Elmezain, A. A. Hamadi, and B. Michaelis, “ Hand trajectory-based gesture spotting and recognition using HMM,” in *Proc. IEEE Int. Conf. on Image Processing*, pp. 3577 – 3580, Nov. 2009.

[63] M. Elmezain, A. A. Hamadi, J. Appenrodt, and B. Michaelis, “A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory,” in *Proc. 2008 19th Int. Conf. on Pattern Recognition*, pp. 1 – 4, Dec. 2008.

[64] B. W. Min, H. S. Yoon, J. Soh, T. Ohashi, and T. Ejima, “Gesture-based editing system for graphic primitives and alphanumeric character,” in *Engineering applications of artificial intelligence*, vol. 12, no. 4, pp. 429 – 441, Aug. 1999.

[65] M. K. Bhuyan, P. K. Bora, and D. Ghosh, “Trajectory Guided Recognition of Hand Gestures having only Global Motions,” in *Electrical, Computer, Energetic and Communication Engineering*, vol. 2, no. 9, pp. 2012 – 2023, 2008.

[66] S. Vikram, and L. Li, and S. Russell, “Handwriting and Gestures in the Air, Recognizing on the Fly,” in *Proc. Int. Conf. on human-computer interaction 2013*, vol. 13, pp. 1179 – 1184, May 2013.

[67] N. Ayachi, P. Kejriwal, L. Kane, and P. Khanna, “Analysis of the Hand Motion Trajectories for Recognition of Air-Drawn Symbols,” in *Proc. 5th Int. Conf. on Communication Systems and Network Technologies*, pp. 505 – 510, Apr. 2015.

[68] L. Kane and P. Khanna, “A Framework to Plot and Recognize Hand motion Trajectories towards Development of Non-tactile Interfaces,” in *Proc. Int. Conf. on Intelligent Human Computer Interaction*, vol. 84, pp. 6 – 13, 2016.

[69] P. Ramasamy, G. Prabhu, and R. Srinivasan, “An economical air writing system converting finger movements to text using web camera,” in *Proc. Int. Conf. on Recent Trends in Information Technology*, pp. 1 – 6, Apr. 2016.

[70] R. Islam, H. Mahmud, Md. K. Hasan, and H. A. Rubaiyeat, “Alphabet Recognition in Air Writing Using Depth Information,” in *Proc. The 9th Int. Conf. on Advances in Computer -Human Interactions*, pp. 299 – 301, Apr. 2016.

[71] H. J. Chang, G. Garcia-Hernando, D. Tang, and T. Kim, “Spatio-Temporal Hough Forest for efficient detection-localisation-recognition of fingerwriting in egocentric camera,” in *Computer Vision and Image Understanding*, vol. 148, pp. 87 – 96, July 2016.

[72] X. Y. Zhang, F. Yin, Y. M. Zhang, C. L. Liu, and Y. Bengio, “Drawing and Recognizing Chinese Characters with Recurrent Neural Network,” in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 849 – 862, Apr. 2018.

[73] J. Lafferty, A. McCallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” in *Proc. 18th Int. Conf. on Machine Learning 2001*, pp. 282 – 289, Jun. 2001.

[74] M. Elmezain, A. Al-Hamadi, S. Sadek, and B. Michaelis, “Robust methods for hand gesture spotting and recognition using Hidden Markov Models and Conditional Random Fields,” in *Proc. The 10th IEEE Int. Symp. on Signal Processing and Information Technology*, pp. 131 – 136, Dec. 2010.

[75] Y. Yao and C. Li, "A Framework for Real-Time Hand Gesture Recognition in Uncontrolled Environments with Partition Matrix Model Based on Hidden Conditional Random Fields," in *Proc. 2013 IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 1205 – 1210, Oct. 2013.

[76] L. Ma, J. Zhang, and J. Wang, "Modified CRF algorithm for dynamic hand gesture recognition," in *Proc. The 33rd Chinese Control Conference*, pp. 4763 – 4767, Jul. 2014.

[77] S. B. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell, "Hidden Conditional Random Fields for Gesture Recognition," in *Proc. 2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 1521 – 1527, Jun. 2006.

[78] S. Feng, R. Manmatha, and A. McCallum, "Exploring the use of conditional random field models and HMMs for historical handwritten document recognition," in *Proc. Second Int. Conf. on Document Image Analysis for Libraries (DIAL'06)*, pp. 8 – 37. Apr. 2006.

[79] C. Yang, D. K. Han, and H. Ko, "continuous hand gesture recognition based on trajectory shape information," in *Pattern Recognition Letters*, vol. 99, pp. 39 – 47, Nov. 2017.

[80] Y. Zhu and G. Xu, "A Real-Time Approach to the Spotting, Representation, and Recognition of Hand Gesture for Human-Computer Interaction," in *Computer Vision and Image Understanding*, vol. 85, no. 3, pp. 189 – 208, Mar. 2002.

[81] M. Chen, G. AlRegib, and B. Juang, "Feature processing and modeling for 6d motion gesture recognition," in *IEEE Trans. on Multimedia*, vol. 15, no. 3, pp. 561 – 570, Apr. 2013.

[82] J. Gan and W. Wang, "In-air handwritten English word recognition using attention recurrent translator," in *Proc. Neural Computing and Applications*, pp. 1 – 18, Nov. 2017.

[83] Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons, "Learning Spatial-Semantic Context with Fully Convolutional Recurrent Network for Online Handwritten Chinese Text Recognition," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1903 – 1917, Aug. 2018.

[84] N. Akazawa, K. Yawata, D. Takeda, Y. Nakayama, H. Kakuda, and M. Suzuki, "A playing and learning support system using Kinect for Romaji," in *Proc. 2014 IEEE 3rd Global Conf. on Consumer Electronics (GCCE)*, pp. 345–349, Oct. 2014.

[85] A. Graves, and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *2005 IEEE Int. Joint Conf. on Neural Networks*, vol. 4, pp. 2047 – 2052, Aug. 2005.

[86] T. A. C. Bragatto, G. I. S. Ruas, and M. V. Lamar, "Real-time video based finger spelling recognition system using low computational complexity Artificial Neural Networks," in *Proc. Int. Telecommunications Symposium*, pp. 393 – 397, Sept. 2006.

[87] R. M. Prakash, T. Deepa, T. Gunasundari, and N. Kasthuri, "Gesture recognition and finger tip detection for human computer interaction," in *Proc. Int. Conf. on Innovations in Information, Embedded and Communication Systems*, pp. 1 – 4, Mar. 2017.

[88] L. Jin, D. Yang, L. Zhen, and J. Huang, "A novel vision based finger-writing character recognition system," in *Proc. 18th Int. Conf. on Pattern Recognition*, pp. 1104 – 1107, Aug. 2006.

[89] V. Frinken and S. Uchida, "Deep BLSTM neural networks for unconstrained contin-

uous handwritten text recognition," in *Proc. Int. Conf. on Document Analysis and Recognition*, pp. 911 – 915, Aug. 2015.

[90] R. Messina and J. Louradour, "Segmentation-free handwritten Chinese text recognition with LSTM-RNN," in *Proc. Int. Conf. on Document Analysis and Recognition*, pp. 171 – 175, Aug. 2015.

[91] A. Graves, S. Fernandez, M. Liwicki, H. Bunke, and J. Schmidhuber, "Unconstrained online handwriting recognition with recurrent neural networks," in *Proc. 20th Int. Conf. on Neural Information Processing Systems*, pp. 577 – 584, Dec. 2007.

[92] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, vol. 9, no. 8, pp. 1735 – 1780, No. 1997.

[93] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," in *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929 – 1958, Jan. 2014.

[94] S. Xu and Y. Xue, "A Long Term Memory Recognition Framework on Multi-Complexity Motion Gestures," in *Proc. 2017 14th IAPR Int. Conf. on Document Analysis and Recognition*, pp. 201 – 205, Nov. 2017.

[95] A. Holzinger, C. Stocker, B. Peischl, and K. M. Simonic, "On using entropy for enhancing handwriting preprocessing," in *Entropy*, vol. 14, no. 11, pp. 2324 – 2350, Nov. 2012.

[96] S. Jaeger, S. Manke, J. Reichert, and A. Waibel, "Online handwriting recognition: the NPen++ recognizer," in *Int. Journal on Document Analysis and Recognition*, vol. 3, no. 3, pp. 169 – 180, Mar. 2001.

[97] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. 6th Int. Conf. on Learning Representations*, pp. 1 – 15, May. 2015.

[98] W. Yang, L. Jin, H. Ni, and T. Lyons, "Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network," in *Proc. 23rd Int. Conf. on Pattern Recognition*, pp. 4083 – 4088, Dec. 2016.

[99] S. K. Behera, P. Kumar, D. P. Dogra, and P. P. Roy, "Fast signature spotting in continuous air writing," in *Proc. 15th IAPR Int. Conf. on Machine Vision Applications*, pp. 314 – 317, May 2017.

[100] C. Li, X. Zhang and L. Jin, "LPSNet: A Novel Log Path Signature Feature Based Hand Gesture Recognition Framework," in *Proc. 2017 IEEE Int. Conf. on Computer Vision Workshops*, pp. 631 – 639, Oct. 2017.

[101] W. Yang, L. Jin, and M. Liu, "Chinese character-level writer identification using path signature feature, DropStroke and deep CNN," in *Proc. 13th Int. Conf. on Document Analysis and Recognition*, pp. 546 – 550, Aug. 2015.

[102] I. Chevyrev and A. Kormilitzin, "A Primer on the Signature Method in Machine Learning," in *Proc. Machine Learning*, arXiv:1603.03788, Mar. 2016.

[103] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing Acoustic-to-Word CTC Model," in *Proc. Computation and Language*, arXiv:1803.05566, Mar. 2018.

[104] T. Bluche, C. Kermorvant, and J. Louradour, "Where to apply dropout in recurrent neural networks for handwriting recognition?," in *Proc. 13th Int. Conf. on Document Analysis and Recognition*, pp. 681 – 685, Aug. 2015.

[105] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour,"Dropout Improves Recurrent Neural Networks for Handwriting Recognition," in *Proc. 14th Int. Conf. on Frontiers*

in Handwriting Recognition, pp. 285 – 290, Sept. 2014.

[106] A. Dash, A. Sahu, R. Shringi, J. Gamboa, M. Z. Afzal, M. I. Malik, S. Ahmed, and A. Dengel, "AirScript - Creating Documents in Air," in *Proc. 2017 14th IAPR Int. Conf. on Document Analysis and Recognition*, pp. 908 – 913, Nov. 2017.

[107] L. Sun, T. Su, C. Liu, and R. Wang, "Deep LSTM Networks for Online Chinese Handwriting Recognition," in *Proc. 15th Int. Conf. on Frontiers in Handwriting Recognition*, pp. 271 – 276, Oct. 2016.

[108] J. Gu, K. Cho, and V. O.K. Li, "Trainable Greedy Decoding for Neural Machine Translation," in *Proc. Computation and Language and Machine Learning*, arXiv:1702.02429, Feb. 2017.

[109] U. Germann, "Greedy decoding for statistical machine translation in almost linear time," in *Proc. 2003 Conf. of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, vol. 1, pp. 1 – 8, May 2003.

[110] W. Yang, T. Lyons, H. Ni, C. Schmid, L. Jin, and J. Chang, "Leveraging the Path Signature for Skeleton-based Human Action Recognition," in *Proc. Computer Vision and Pattern Recognition*, arXiv:1707.03993, Jul. 2017.

[111] T. Lyons "Rough paths, Signatures and the modelling of functions on streams" in *Proc. Int. Congress of Mathematicians 2014*, arXiv:1405.4537, May 2014.

[112] K. T. Chen. "Integration of paths–A faithful representation of paths by noncommutative formal power series," in *Trans. of the American Mathematical Society*, vol. 89 no. 2, pp. 395 – 407, Nov. 1958.

[113] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson,"Mini-batch gradient descent: Faster convergence under data sparsity," in *Proc. 2017 IEEE 56th Annual Conf. on Decision and Control*, pp. 2880 – 2887, Dec. 2017.

[114] M. A. Simao, P. Neto, and O. Gibaru, "Unsupervised Gesture Segmentation by Motion Detection of a Real-Time Data Stream," in *IEEE Trans. on Industrial Informatics*, vol. 13, no. 2, pp. 473 – 481, Apr. 2017.