3D In-Air-Handwriting based User Login and Identity Input Method

by

Duo Lu

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2021 by the
Graduate Supervisory Committee:

Dijiang Huang, Chair
Baoxin Li
Junshan Zhang
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2021

ABSTRACT

Applications over a gesture-based human-computer interface (HCI) requires a new user login method with gestures because it does not have traditional input devices. For example, a user may be asked to verify the identity to unlock a device in a mobile or wearable platform, or sign in a virtual site over a Virtual Reality (VR) or Augmented Reality (AR) headset, where no physical keyboard or touchscreen is available. This dissertation presents a unified user login framework and an identity input method using 3D In-Air-Handwriting (IAHW), where a user can log in to a virtual site by writing a passcode in the air very fast like a signature. The presented research contains multiple tasks that span motion signal modeling, user authentication, user identification, template protection, and a thorough evaluation in both security and usability. The results of this research show around 0.1% to 3% Equal Error Rate (EER) in user authentication in different conditions as well as 93% accuracy in user identification, on a dataset with over 100 users and two types of gesture input devices. Besides, current research in this area is severely limited by the availability of the gesture input device, datasets, and software tools. This study provides an infrastructure for IAHW research with an open-source library and open datasets of more than 100K IAHW hand movement signals. Additionally, the proposed user identity input method can be extended to a general word input method for both English and Chinese using limited training data. Hence, this dissertation can help the research community in both cybersecurity and HCI to explore IAHW as a new direction, and potentially pave the way to practical adoption of such technologies in the future.

ACKNOWLEDGMENTS

First, I would like to thank Dr. Dijiang Huang for his support of this research project and the supervision of this dissertation. Dr. Huang is my academic advisor and the chair of my PhD dissertation committee. Through my PhD study at Arizona State University (ASU) and the writing of this dissertation, I have received a great deal of help from Dr. Huang, whose expertise is invaluable in pointing out the research directions, creating the research plan, and writing research proposals for funding opportunities. His insightful thinking and feedback helped me to bring this research to a higher level.

Second, I would like to thank my PhD committee, including Dr. Dijiang Huang, Dr. Yezhou Yang, Dr. Junshan Zhang, and Dr. Baoxin Li, who have supported me in my PhD study and provided me great opportunities of participating in research projects outside cybersecurity, applying patents, putting ideas into practice, working with a startup company, and teaching at ASU. These valuable experiences helped me sharpen my mind and allowed me to reach out for different ideas, which eventually advanced my career in academia.

Third, I would like to thank my colleagues at ASU Secure Networking and Computing Research Group, collaborators at ASU Active Perception Group, and schoolmates at ASU. They provided various types of help in this research. I would also like to thank every volunteer who participated in the data collection of this study.

Fourth, I would like to thank my wife, our family, and our friends. I could not have completed this dissertation without their support and care.

Finally, I would like to thank ASU Fulton School of Engineering (FSE) and National Science Foundation (NSF). This project is partially supported by ASU FSE Minor Research Infrastructure grant and partially supported by NSF CCRI grant #1925709 as well as NSF CNS grant #1528099.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER

1 INTRODUCTION 1

2 RELATED WORKS 7

2.1 Gesture-based User Authentication 7

2.2 Deep Neural Network and Deep Hashing 10

2.3 In-Air-Handwriting Recognition 13

3 FMKIT - A CODE LIBRARY AND DATA REPOSITORY FOR IAHW. 15

3.1 Background 15

3.2 IAHW Signal Characterization 16

3.3 IAHW Data Collection Device 20

3.4 IAHW Signal Preprocessing 24

3.5 Formulation of IAHW Signal Variation 27

3.6 The FMKit Code Library 30

3.7 The FMKit Data Repository 32

3.8 Supported Research Tasks 38

3.8.1 User Authentication 39

3.8.2 User Identification 41

3.8.3 Word Recognition 43

3.9 Summary 44

4 FMAUTH - USING IAHW FOR USER AUTHENTICATION 46

4.1 Background 46

4.2 User Authentication System Model 48

4.3 Temporal Feature Analysis 52
4.3.1 Alignment Analysis and DTW Matching 52
4.3.2 Sample Variation Analysis and TTV Matching 57
4.4 Statistical Feature Analysis 66
4.5 Hand Geometry Feature Analysis 70
4.6 Fusion of Multiple Features 72
4.6.1 Score Fusion 72
4.6.2 Feature Fusion 75
4.7 Experimental Evaluation 76
4.8 Analysis of The Hard Cases 83
4.9 Summary 87
5 FMHASH - USING IAHW FOR USER IDENTIFICATION 89
5.1 Background 89
5.2 System Model 91
5.3 FMHashNet 93
5.4 Account Database and User Identification Procedure 98
5.5 Experimental Evaluation 100
5.6 Summary 107
6 FMCODE - A UNIFIED IAHW BASED USER LOGIN FRAMEWORK 108
6.1 Background 108
6.2 System Model 110
6.3 Implementation Detail 113
6.3.1 User Authentication 113
6.3.2 User Identification 116

CHAPTER Page

6.3.3 Template Protection 118

6.3.4 Cost of Computing and Storage 122

6.4 Security Analysis 123

6.4.1 Security Model 123

6.4.2 Analysis of User Authentication 125

6.4.3 Analysis of User Identification 131

6.4.4 Comparative Study 135

6.4.5 Discussions 137

6.5 Usability Analysis 138

6.5.1 User Evaluation 139

6.5.2 Qualitative Evaluation 142

6.6 Summary 145

7 FMWORD - USING IAHW FOR WORD INPUT 146

7.1 Background 146

7.2 System Model 147

7.3 Data Augmentation 150

7.4 Experimental Evaluation 156

7.5 Summary 159

8 CONCLUSIONS 160

REFERENCES 162

LIST OF TABLES

Table Page

2.1 Comparison of Experimental Results and Algorithms of Related Works 11

3.1 Baseline Performance for User Authentication 41

3.2 Baseline Performance for User Identification 43

3.3 Baseline Performance for IAHW Word Recognition 44

4.1 Authentication Performance Results with the Camera Device 78

4.2 Authentication Performance Results with the Glove Device 78

5.1 The FMHashNet Architecture 94

6.1 Empirical Results of User Authentication with the Camera Device 126

6.2 Empirical Results of User Authentication with the Glove Device 127

6.3 Empirical Results for User Identification 132

6.4 Comparison to Existing Works 135

6.5 Comparison to Password and Biometrics 137

6.6 Qualitative Evaluation of Usability. 143

6.7 Qualitative Evaluation of Deployability. 143

6.8 Qualitative Evaluation of Security. 144

7.1 English Word Recognition Results 156

7.2 Chinese Word Recognition Results 157

LIST OF FIGURES

Figure Page

1.1 Example Application Scenarios of IAHW Based User Input 4
3.1 The IAHW Signal Model 17
3.2 The Leap Motion Controller for Finger Motion Capturing 21
3.3 The Custom Data Glove Device for Finger Motion Capturing 22
3.4 An Example of Raw Signals Obtained From the Two Types of Devices . 23
3.5 An Example of Signal Alignment 28
3.6 Example of Signal Alignment and Template Construction 29
3.7 The Interactive Gui-based Software to Browse Our Datasets 31
3.8 Data Collection Procedure for the Leap Motion Controller Device 35
4.1 Authentication Framework Architecture 48
4.2 An Example of a Preprocessed Signal and Trajectory 49
4.3 An Example of Generating an Aligned Signal From a Warping Path . . . 53
4.4 Example of a Warping Path and Signal Alignment Results 54
4.5 Distribution of DTW Distance Scores With Both Types of Devices 55
4.6 Distribution of Warping Path for Different Classes Using Both Devices . 56
4.7 Distribution of Alignment Cost Scores With Both Types of Devices 56
4.8 Joint Distribution of Element-wise Distance and Template Property, with the Camera Device 59
4.9 Joint Distribution of Element-wise Distance and Template Property, with the Camera Device, with the Glove Device. 60
4.10 Relation between Element-wise Distance and Template Property, with the Camera Device. 61
4.11 Relation between Element-wise Distance and Template Property, with the Glove Device. 62

Figure Page

4.12 Distribution of Aggregated Scaled Element-wise Distance in Different Sensor Axes, without Considering Collision 63

4.13 Distribution of Aggregated Scaled Element-wise Distance in Different Sensor Axes, with Considering Collision 64

4.14 Distribution of Aggregated Scaled Element-wise Distance in Time 65

4.15 Distribution of TTV Distance Scores with Both Types of Devices 66

4.16 Distribution of Statistical Difference for Different Classes with Both Types of Devices 68

4.17 Correlation of the Statistical Features of All Axes 69

4.18 Distribution of Statistical Difference Scores With Both Types of Devices 69

4.19 Definition of Hand Geometry Features 70

4.20 Average of Length Differences of Hand Geometry Feature Components 71

4.21 Distribution of the Scores From Various Features 73

4.22 Receiver Operating Characteristic (ROC) 79

4.23 Authentication Performance Change Respect to the Number of Signals for Template Construction 81

4.24 Performance Change of the T-Fusion Method with Respect to the Number of Negative Training Samples 82

4.25 Matching Score Distribution of Each Account Using the F-Fusion Method 85

4.26 Examples of Hard Cases, Blue for the Templates and Orange for the Testing Signals 86

5.1 The Architecture of the FMHash Framework 92

5.2 An Illustration of the FMHashNet Architecture 95

5.3 The Effect of the Regularizer 97

Figure Page

5.4 An Illustration of the Identification Procedure 99

5.5 Identification Results With the Camera Device 103

5.6 Identification Results With the Glove Device 104

5.7 Distribution of the Hamming Distance Between the Hash Code of the Account and the Hash Code of Testing Signals. 105

5.8 Distribution and Correlation of the Zeros and the Ones in the Hash Code 106

5.9 Distribution of the Hamming Distance Between the Hash Code of the Account and the Hash Code of Testing Signals for Each Account. 106

5.10 Examples of Generated Hash Codes for Multiple Accounts 107

6.1 The FMCode Login Framework Architecture and Procedures 111

6.2 An Illustration of Key Binding Using Fuzzy Commitment 121

6.3 Receiver Operating Characteristic (ROC) 127

6.4 Authentication Score Histogram for Different Classes 128

6.5 Authentication Score Among Ten Sessions, with the Camera Device . . . 129

6.6 Authentication Score Among Ten Sessions, with the Glove Device 129

6.7 Variation of True Acceptance Rate (TAR, *i.e.*, 1 - FRR) Among Ten Sessions 130

6.8 Identification Results Considering Spoofing Attacks, with and Without ID Verification 133

6.9 Variation of Identification Accuracy Among Ten Sessions 134

6.10 Results of Subjective User Evaluation on Our Login Framework 140

6.11 Results of Subjective User Evaluation on Our Login Framework Compared With Other Login Methods 141

Figure Page

7.1 The Architecture of the FMWord Framework. 148

7.2 An Example of Data Augmentation with Random Rotation 152

7.3 An Example of Data Augmentation with Random Warping 153

7.4 An Example of Data Augmentation with Random Amplitude Perturbation. 154

7.5 An Example of Data Augmentation with Random Segment Swapping . . 155

7.6 Two Different Writers Write the Same Word "bot" in Different Stroke Sequences 159

Chapter 1

INTRODUCTION

Gesture interfaces are generally considered as the next generation of Human-Computer Interface (HCI), which can fundamentally change the way that a human user interacts with a computer, especially on wearable computers, Virtual Reality (VR) or Augmented Reality (AR) applications, and home entertainment console. Meanwhile, novel computing platforms equipped with gesture interfaces are gaining popularity in consumer electronics, gaming, and education markets. For example, Microsoft sold 35 million kinect gesture sensors with XBox [1] , and Sony sold more than 1 million VR headset with gesture controller [2] . Wearable VR/AR headsets with gesture interface like Facebook Oculus and Microsoft Hololens also attracted attention for the immerse experience in gaming and simulation.

On these platforms, user login is a basic function, either for unlocking the device or log in to a website for accessing personalized services and private data. However, in such an environment, it is usually inconvenient or impractical to use a physical keyboard or virtual keyboard on a touchscreen for typing a password. Most existing VR headsets (*e.g.*, Oculus Rift, HTC Vive, Google Daydream VR) and wearable computers rely on the connected desktop computer or smartphone to finish the login procedure using passwords, which is inconvenient. Some of them present a virtual keyboard in the air for the user to type a password, which is slow because of limited recognition accuracy of the in-the-air key type action and a lack of physical keystroke feedback to the user. Meanwhile, digital identity solutions are becoming more and

[1]http://fortune.com/2017/10/25/microsoft-kinect-xbox-sensor/

[2]https://www.theverge.com/2017/6/5/15719382/playstation-vr-sony-sales-one-million

more commonplace, especially as consumers grow increasingly aware of the weaknesses of authentication methods like passwords. For example, weak passwords can be easily guessed and strong passwords are difficult to remember. A recent study found 32% of security incidents in 2019 involved phishing and 29% involved stolen credentials, both of which would have been much more difficult had secure digital ID systems been in place (Jason Tooley, 2020). Businesses and organizations are already taking steps to close this security gap. Approximately 90% of mid-size organizations and 60% of large companies are expected to begin phasing out passwords by 2022, instead turning to digital ID solutions such as two-factor authentication or biometrics (Gloria Omale, 2019). Biometrics, on the other hand, is linked to the person, which means they cannot be changed and they may raise privacy issues. For example, people are usually reluctant to give their fingerprint to an arbitrary website other than the bank, government offices or other trusted authorities. Additionally, the coronavirus outbreak has presented a new challenge to the extensive deployed fingerprint and facial recognition applications, because many people prefer the contactless authentication interface to avoid virus infection, and they also wear masks to curb the spread of the virus (PYMNTS, 2020).

Compared with password and traditional biometric technologies, behavior-based identity verification technology can be used for login based on their unique behavior when interacting with sensing devices. For examples, people walk (*i.e.*, gait recognition), people use their mouse or keyboard (*i.e.*, biometrics of mouse or keystroke dynamics), people hold and interact with a phone/tablet (*i.e.*, touch gesture biometrics), etc. Among all these methods, this dissertation focuses on **In-Air Handwriting (IAHW)**, due to the following unique properties:

- **Multi-factor**: Generally speaking, existing user identification or identity verification methods include *what you know (such as passwords)*, *who you are (unique* *physiological characteristics such as fingerprints)*, and *what you have (such as a token card)*. An IAHW solution may include all these features. For example, IAHW can contain the meaning of handwritten strings created by the user, the geometric shapes and texture patterns of the hand that can uniquely identify the biological characteristics of the user, and the unique behavioral biological characteristics possessed by the user represented by the handwriting.

- **Limited-contact**: Due to the *in air* nature, *limited-contact* data collection devices can be used, such as cameras, motion capture sensors, personal wearable sensors, which can effectively avoid the spread of microorganism attached to the device due to frequent contacts by different people.

- **Mobile-friendly**: Gesture interfaces have great potentials on wearable computers, Virtual Reality (VR) and Augmented Reality (AR) headsets, where a traditional keyboard or touch screen is not suitable. IAHW is suitable for these applications using a native hand motion capture device on these platforms as an input method.

- **Less-intrusive**: Biometrics is linked to the person, which means they cannot be changed and they may be used to track the users. Hence, people are usually only willing to give their fingerprint to trusted authorities. Instead, IAHW shares most of the features of password, *e.g.*, the content of an IAHW string is changeable and the information in the IAHW is not physically bound to a person.

However, IAHW-based research is still in its infancy, which requires us to answer several unresolved questions. Consider the two scenarios shown in Figure 1.1 as an example, a dialog box asks the user to enter a password, ID, or a short piece of text

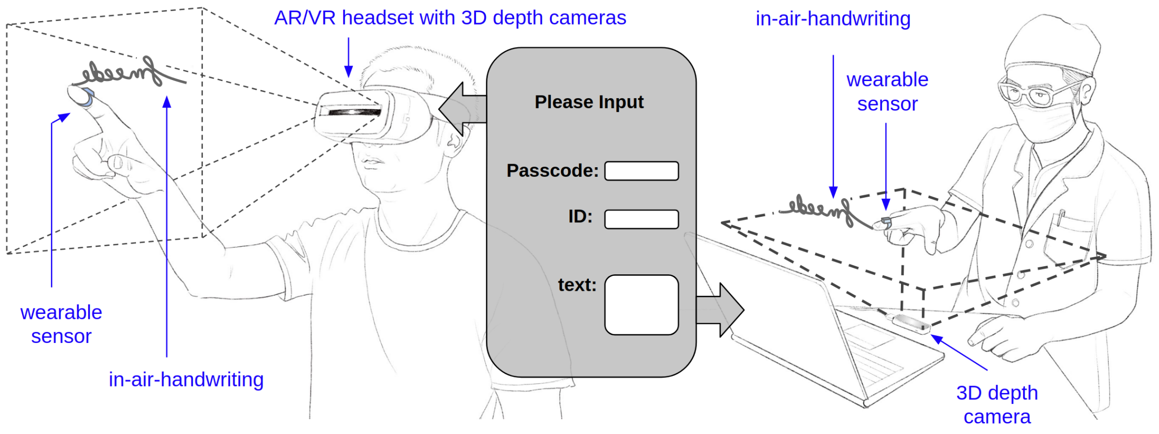


Figure 1.1: Example Application Scenarios of IAHW Based User Input

information (e.g., a sentence) as input information. In the case shown on the left half of the figure, there is no physical input devices for typing, and in the case shown on the right half of the figure, it requires a high degree of cleanness such that touching is not suitable. Instead of typing or touching, for these cases the finger motion can be captured either by a wearable device or a 3D camera, and the information in the motion signal can be extracted to verify the identity of the user or it can be used as generic input text. Our research objective is to study and develop such an IAHW based system over gesture interface, with a focus on the user authentication function and identity input method. To reach this goal, in this dissertation we are going to answer the following seven questions:

Q1) How to capture and represent a piece of IAHW?

Q2) How to determine whether two pieces of IAHW are generated by the same user or not, like verification of a typed alphanumeric password?

Q3) How to efficiently identify the writer using a piece of IAHW among a database of accounts, like an input method of an ID?

Q4) How to protect the identity information on the server from storage leakage, like storing the hash of a password instead of the plaintext password?

Q5) Can a human attacker spoof the handwriting of a legitimate user to pass the login system, like shoulder surfing attack to steal a password?

Q6) How about the usability of such an IAHW based login system, especially compared to password, finger print biometrics, or face recognition?

Q7) Can such an identity input method be extended to a generic word input method?

These questions address various aspects of the desired user input system from the deployment perspective, and their answers can be helpful to engineer an IAHW-based user login system or identity input method through a gesture interface. Hence, this dissertation attempts to work out the answers to these questions and discuss the insights that we discovered in this research project. Specifically, chapter 2 layouts the related works with a literature review. Chapter 3 shows a library and data repository for IAHW analysis, named FMKit, which answers Q1. It is constructed by us and currently made openly available to the academic community. Chapter 4 presents a detailed analysis of the global features of IAHW signals, which can be used to construct user authentication methods, and hence, answers Q2. Chapter 5 provides a system to convert the information in a piece of IAHW movements to a locality-sensitive hash code, which can be used for user identification, and hence, answers Q3. Then, in Chapter 6, a unified user login framework is built based on the previous chapters. Moreover, various practical issues are discussed, including template protection, spoofing attack, and usability, which answers Q4, Q5, and Q6. After that, in chapter 7, an extension of the user identification method is shown to serve as a generic word input method through gesture interfaces, which answers Q7. Finally, chapter 8 summarizes this dissertation and draws the conclusions.

The main contribution of this dissertation is the development of building blocks that satisfy the needs for user login and identity input method through gesture interface in a systematic way. Since currently there is no satisfying solution for gesture interface, our work provides answers on how such a solution could be engineered and what challenges should be addressed, which may further pave the road to the pervasive adoption of such an IAHW-based solution. Additionally, current research in gesture based authentication is severely limited by the availability of gesture input device, dataset and data driven models. This research mainly targets low cost gesture input devices, and we have made our dataset and software packages openly available, which would help other researchers and practitioners in both the field of usable security and the field of human-computer interface. In addition, the in-air-handwriting does not have restrictions on the language, including languages that cannot be typed directly on keyboard, and hence, this research can potentially benefit people around the world.

Chapter 2

RELATED WORKS

In this chapter, we review the related works in three areas – gesture-based authentication, deep hashing using neural networks, and in-air-handwriting recognition.

2.1 Gesture-based User Authentication

Traditional user authentication methods can be based on physiological traits (*e.g.,* biometrics, like fingerprint and iris), knowledge (*e.g.,* password), or possession (*e.g.,* ID card, token). Unfortunately, none of them is perfect because physiological traits can be copied (e.g. fingerprint spoofing by taking fingerprint left on cups or furniture and making silica replica), knowledge can be forgotten (e.g. forgotten password), and possession can be lost or faked (stolen ID card). Currently, password and biometrics are the most widely adopted user authentication methods, but they both have shortcomings. On the password side, remembering different passwords for different sites is a burden for the user, especially when a memorable password is usually not strong, while a strong password is usually difficult to remember (Florencio and Herley, 2007). Most users choose to reuse the same password across many sites (Ives *et al.*, 2004), and they rarely update it or only update it in predictable patterns (Zhang *et al.*, 2010). As a result, a password based system works far from optimal and there are numerous research works that have made attempts to replace password (O'Gorman, 2003; Bonneau *et al.*, 2012). On the biometrics side, although identity verification using fingerprint (Cappelli *et al.*, 2006), iris (Daugman, 2007) and face (Schroff *et al.*, 2015) are convenient, inexpensive and more secure than password in many cases, biometrics are inherently not secrets, and they can not be changed easily for a user.

For example, fingerprints can be leaked by the surfaces we touch and faces can be leaked by photos, which makes it surprisingly easier to spoof than what most people had thought [1] . Once leaked, the biometric information is unable to be revoked easily and it may cause privacy issues because it is directly linked to a person instead of an anonymous account. On VR/AR and wearable computing platforms, whether password and biometrics methods are suitable for user authentication is still an open question.

Gesture-based user authentication can be regarded as a fusion of a secret, a behavior biometrics, and possibly a physiological trait (*e.g.*, the hand geometry). It combines the advantages of both password and biometrics, and it can potentially achieve secure and convenient user login functions through gesture interface in mobile scenarios such as VR/AR and wearable computers. Besides, a gesture-based authentication method is desired to be able to defend spoofing attack to a certain degree. Additionally, a gesture should contain a dynamic information that can be changed or made in a challenge-response way, instead of just present a piece of static information such as fingerprint or face.

Early gesture and body motion based authentication methods use the accelerometer to track simple motions like shaking (Patel *et al.*, 2004; Okumura *et al.*, 2006; Mayrhofer and Gellersen, 2007), walking (Gafurov and Snekkkenes, 2008; Lester *et al.*, 2004), or other predefined gesture patterns (Farella *et al.*, 2006), which proves the feasibility of such methods. Free-form hand gestures such as the in-air-handwriting can be captured by smartphone or other handhold remote control devices (Hong *et al.*, 2015; Liu *et al.*, 2009; Zaharis *et al.*, 2010; Bailador *et al.*, 2011) with an inertial sensor. They have better potential because the complex motion of hand contains more information that is useful to distinguish different users. However, waving a device as

[1]http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid

heavy as a smartphone to mimic writing is unnatural, which may lead to unstable gesture in the long term. Moreover, the gesture behavior may depend on the actual device. Furthermore, it is not easy to directly track the finesse of the movement of fingertip, but choose to use movements of the wrist with a handhold device or a smartwatch (Nassi *et al.*, 2016). Recently, in-air-handwriting is investigated with camera based gesture interface such as Kinect (Tian *et al.*, 2013; Wu *et al.*, 2013; Hayashi *et al.*, 2014) and Leap Motion (Chahar *et al.*, 2015; Aslan *et al.*, 2014; Chan *et al.*, 2015; Piekarczyk and Ogiela, 2015). The major drawback of such methods is strong constraints on the size, speed, and orientation of a user's movement, because these cameras are set up at a fix place. For example, the user is required to move within the field of view of the device, which is usually mounted on the desk. Also the user must not move fast because of the limited capture speed of the camera. These constraints hurt usability and authentication performance, and hence, wearable cameras such as Google Glass (Sajid and Sen-ching, 2015) show advantages. However the vision algorithm for hand motion tracking in real world environments has many difficulties such as separating foreground and background and understanding the shape of a hand in different pose. These technical challenges limit the authentication performance. The critical research issue for gesture based authentication is designing good features and classifiers that can tolerate variations of the captured motions of the same user as well as distinguishing different users with high accuracy. Besides, nearly all existing systems focus on authentication, which assumes the user can provide the specific account to login. In reality, the user is also required to provide an account ID, which creates another problem of user identification, *i.e.*, locate an account ID in a large account database given a piece of gesture input information. For gesture based user identification, the critical research issue is designing efficient way to index gesture signals or features. Existing works (Liu *et al.*, 2009; Bailador *et al.*, 2011; Chahar *et al.*, 2015) have to exhaustively search the account database to find a matching account.

It should be noted that the in-air-handwriting based login system is different from handwriting recognition or hand gesture recognition based systems (**?**Qu *et al.*, 2015; Moazen *et al.*, 2016; Wen *et al.*, 2016; Zhang and Harrison, 2015) because most users write passcode like a signature in an illegible way and technically there is no predefined gesture patterns or strokes. In addition, a handwriting recognition system tries to classify similar finger motion patterns to the same letter or word regardless of the writer. Moreover, the in-air-handwriting based login system is different from signature (Impedovo and Pirlo, 2008) or 2D multi-touch gesture based authentication system (Frank *et al.*, 2013; Gong *et al.*, 2016; Yang *et al.*, 2016; Song *et al.*, 2017) because the useful information is in the 3D trajectory and speed of the fingertip, instead of 2D pressure changes and image features of a pen or digital pen Bashir and Kempf (2009); Renuka *et al.* (2014). Furthermore, it is different from writer identification or signature verification systems designed for non-repudiation. In our system the ID or the passcode can be changed at any time, which are not linked to the users' identity for privacy. Finally, our framework directly targets the VR or wearable computer scenarios with gesture input interface, which is less addressed by existing systems.

A comparison of related works is shown in Table 2.1.

2.2 Deep Neural Network and Deep Hashing

A neural network is a computational model consisting of layers of neurons. Each neuron calculates an linear combination of its input $\mathbf{x}$, and then, a non-linear activation function $f()$ is applied to generate an output h. Mathematically, a neuron can be represented as $h = f(\mathbf{wx} + b)$. One layer of neurons takes the output of its previous

Table 2.1: Comparison of Experimental Results and Algorithms of Related Works

Related Works	# of subjects	Device	Experiment Timespan	EER	Claimed Accuracy	Motion Gesture	Algorithm
Patel *et al.*	NA	Cellphone w/ accelerometer	NA	NA	NA	shake	static threshold
Okumura *et al.*	12 ∼ 22	Cellphone w/ accelerometer	6 weeks	4%	NA	shake	DTW
Mayrhofer and Gellersen	51	Custom device w/ accelerometer	NA	∼2%, ∼10%	NA	shake	frequency coherence
Farella *et al.*	5 ∼ 10	PDA w/ accelerometer	NA	NA	63% ∼ 97%	4 specified gestures	PCA / LLE + kNN
Lester *et al.*	6	Custom device w/ accelerometer	NA	NA	100%	walk	frequency coherence
Gafurov and Snekkkenes	30	Custom device w/ accelerometer	NA	10%	NA	walk	frequency similarity
Liu *et al.*	20 ∼ 25	Nintendo Wii remote	1 ∼ 4 weeks	∼3%, >10%	88% ∼ 99%	free writing	DTW
Zaharis *et al.*	4	Nintendo Wii remote	3 weeks	NA	98.20%	free writing	statistical feature matching
Bailador *et al.*	96	Smartphone	20 days	∼2.5%	NA	free writing	DTW, Bayes, HMM
Lee *et al.*	15	Smartphone	NA	NA	88.40%	tap, flip, etc.	decision tree
Bashir and Kempf	10 ∼ 40	Custom pen	NA	∼1.8%	98.50%	alphabetic or free writing	RDTW
Renuka *et al.*	12	Custom pen	NA	NA	∼ 95%	alphabetic or free writing	NA
Aslan *et al.*	13	Leap Motion	NA	∼10%	NA	2 specified gestures	DTW
Chahar *et al.*	150	Leap Motion	NA	NA	81%	free writing	statistical feature classification
Chan *et al.*	16	Leap Motion	NA	0.80%	>99%	free writing	random forest
Piekarczyk and Ogiela	4 ∼ 5	Leap Motion	NA	NA	88% ∼ 100%	4 specified gestures	DCT + DTW + kNN / LSH
Tian *et al.*	18	Kinect	5 months	∼2%	NA	free writing	DTW
Sajid and Sen-ching	10	Google Glass	NA	NA	97.50%	free writing	PCA + GMM Clustering + DTW
Wu *et al.*	20	Kinect	NA	1.89%	NA	8 gestures	DTW
Hayashi *et al.*	36	Kinect	2 weeks	0.5% ∼ 1.6%	NA	hand waving	SVM

layer as input, and its output is fed to the next layer as input. A deep neural network contains multiple layers, which enables the capability to approximate an arbitrary non-linear function.

A deep convolutional neural network (CNN) is generally composed of convolutional layers and pooling layers. In a CNN, each neuron in the convolutional layer connects to only a few adjacent neurons in the previous layer, and all neurons in the layer share the same set of weights. Similarly, each neuron in the pooling layer connects to only a few neurons locally and output the maximum or average of the inputs. The combination of convolutional layers and pooling layers can learn and detect local features efficiently and automatically by optimizing a loss function on the whole network. Besides, cascading multiple convolutional layers enables the construction of high-level abstract features, which is able to solve complicated pattern recognition problems, most notably the image classification problem (Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2014; Szegedy *et al.*, 2015; He *et al.*, 2016; Chollet, 2017; Huang *et al.*, 2016).

Deep hashing is a deep neural network based method to generate compact binary hash codes from images to achieve fast retrival of semantically similar images from a database (Xia *et al.*, 2014; Lai *et al.*, 2015; Zhang *et al.*, 2015; Zhao *et al.*, 2015; Zhu *et al.*, 2016; Cao *et al.*, 2016; Liu *et al.*, 2016; Lin *et al.*, 2016; Cao *et al.*, 2017; Da *et al.*, 2017; Liu *et al.*, 2017). To search similar images, a query image is converted to a hash code with the same neural network, and images in the database with the exact same or neighboring hash code are returned. Existing works mainly utilize the label of training images through pairwise supervision (Xia *et al.*, 2014; Liu *et al.*, 2016), triplet supervision (Zhang *et al.*, 2015; Lai *et al.*, 2015), or ranking list supervision (Zhao *et al.*, 2015), with various regularization and quantization loss (Liu *et al.*, 2016) to train the network. The goal is to hash images from the same class to the same code and images from different classes to different codes. A similar idea can be applied to identify users with a piece of in-air-handwriting, *i.e.*, we use the hash code of an handwritten ID string to index each account so that at login if a user write the same ID string, the same hash code is generated and the account ID can be retrieved efficiently from a hash table of accounts. Although a CNN can be utilized to generate a hash code from an image, hashing a piece of in-air-handwriting for user identification has different requirements. First, the features of in-air-handwriting motion signal is inherently different from the features of image. Second, identification has different goals from image retrieval. For an identification system, it is desired to maintain the sparsity of the Hamming space to avoid wrong identification instead of maintaining smooth similarity change of the hash code in image retrieval, and the fuzziness in the signal should not propagate to the hash code to increase identification accuracy. Meanwhile, it is desired that a secret key can be generated from the hash code to protect the account information so that long hash code is preferred due to the difficulty for guessing. Third, considering that new accounts are registered from time to time, our network is optimized with smaller number of parameters and faster training speed.

2.3 In-Air-Handwriting Recognition

In-air-handwriting based user authentication and identification is closely related to in-air-handwriting recognition, i.e., recognizing each individual characters, words, and sentence based on a piece of finger motion signal generated by writing in some language in the air. Such finger motion of handwriting can be considered as a special type of hand gesture to convey information in verbal languages. Similar as any gesture input, the recognition of in-air-handwriting builds upon accurate hand motion tracking. Related research works mainly use three types of devices in general. The first type is inertial sensors in the smartphone (Agrawal *et al.*, 2011), smartwatch (Xu *et al.*, 2015), special glove (Amma *et al.*, 2014), or digital pen (Wang *et al.*, 2013; Hsu *et al.*, 2015). The second type is computer vision based sensor systems such as the LeapMotion controller (Qu *et al.*, 2016; Gan and Wang, 2017), Kinect (Zhang *et al.*, 2013; Ye *et al.*, 2013), or active visual tracking (Asano and Honda, 2010; Perrin *et al.*, 2004). The third type is radio based system using RFID (Ding *et al.*, 2017; Wang *et al.*, 2014), WiFi (Sun *et al.*, 2015), or millimeter radar (Lien *et al.*, 2016). Compared to general gesture input recognition, in-air-handwriting relies more heavily on data due to the inherent complexity of handwriting motion and the data-driven models such as hidden markov model (Amma *et al.*, 2014) and neural network (Ren *et al.*, 2017; Ye *et al.*, 2013). Hence, the lack of standard motion sensors and open dataset significantly hinders the advancement of research in this area. To the best of our knowledge, the largest dataset of in-air-handwriting is the IAHCC-UCAS and IAHEW-UCAS (Qu *et al.*, 2016; Gan and Wang, 2017) datasets, which contains the data of 3755 Chinese characters and 2280 English words collected with the Leap Motion controller. Besides the data collection, labeling each character requires a significant amount of effort, which usually exceeds the ability of single research group without special funding for such an infrastructure construction. As a result, currently no such dataset offers annotation in character segmentation and stroke level labeling for analyzing the complexity of a handwriting string and further deriving the strength of a handwriting passcode.

Chapter 3

FMKIT - A CODE LIBRARY AND DATA REPOSITORY FOR IAHW

3.1 Background

In AR/VR applications, 3D in-air hand gestures are natural ways to interact with virtual objects as well as a method to input information (Freeman *et al.*, 2016; May *et al.*, 2017; Piumsomboon *et al.*, 2013; Tsukada and Yasumura, 2001). Meanwhile, gesture input interfaces also have potential in wearable computers, gaming consoles, IoT devices, and facilities with high level of cleanness where direct contact should be avoided. Besides simple gestures like tap, scroll, slide, etc, there are cases when complicated information such as text or identity is needed to be presented to the software as input. Consider the scenario shown in Figure 1.1, a dialog pops up to an AR/VR user asking a passcode, an ID, or a short piece of text information (*e.g.*, a tweet) as input information. One possible method is using IAHW, *i.e.*, instead of typing, the user can write a piece of information in the air, and ask the computer to recognize it. This piece of information may be an ID string, or a passcode, or a string of some text input. However, there are a few challenges. First, recognizing patterns in hand motion generally requires a large amount of data to train machine learning models, but the data collection is generally expensive. Second, there is no standard hand motion capture device and data collection procedures, and existing works choose their own ways, which makes the datasets incompatible and hard to compare. Additionally, most of them are not openly available. Third, given different motion capture devices and multiple IAHW related tasks solved using various algorithms, these mix-and-matches make it difficult to fairly evaluate different systems and compare their performance results (Clark and Lindqvist, 2015).

In this chapter, we present FMKit (Finger Motion analysis Kit), an open-source library and open datasets for IAHW analysis, which serve as the foundation of our IAHW related research projects. Currently, FMKit contains five datasets of IAHW hand motion signals, in total about 110K data signals. Especially, each dataset has two parts collected in identical scenarios with the same IAHW content using two different motion capture devices, so that the results of the two devices can be compared. We also release the details of the devices, data collection protocols, and the source code of the client software online [1] so that researchers can extend our datasets. Meanwhile, we propose three research tasks enabled by the FMKit infrastructure: (a) user authentication (Lu *et al.*, 2018, 2017), (b) user identification (Lu *et al.*, 2019), and (c) IAHW word recognition. These research tasks correspond to the three types of inputs in Figure 1.1 and they consist the main body of this dissertation. Additionally, in this chapter, we provide the performance results for these tasks with our dataset as an overview, which can be considered as baselines for future research. We hope this open infrastructure helps researchers to validate models, benchmark the performance of algorithms, and further facilitate data-driven approaches that can be generalized across sensor types for IAHW analysis. This chapter is based on our published research paper (Lu *et al.*, 2020).

3.2 IAHW Signal Characterization

Handwriting is closely related to the cognitive process of humans (Itaguchi *et al.*, 2015) in different levels, and hence, we model the IAHW as a generative process in four levels: string level, character level, stroke level, and signal level, as shown in Figure 3.1. First, a string contains meaningful symbols, characters, or letters in a certain

[1]https://github.com/duolu/fmkit

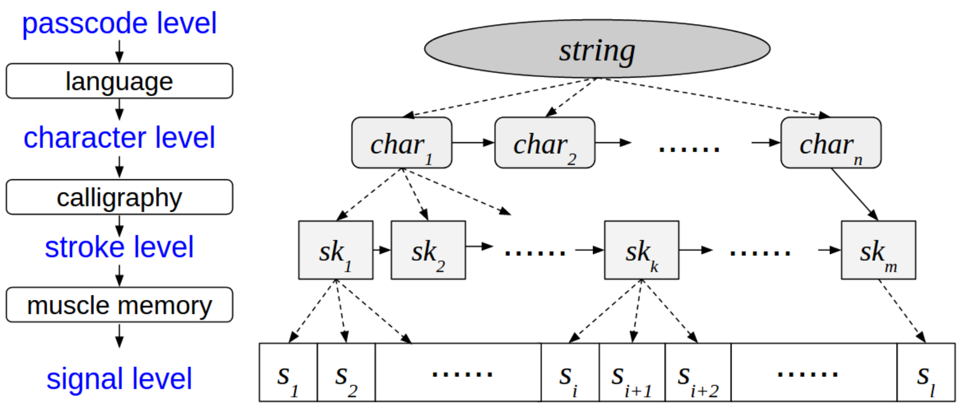


Figure 3.1: The IAHW Signal Model

language. Second, each character or letter of a string in a language is made of strokes defined by the calligraphy of that language. Third, the strokes further determine the hand motion of the writing through the control of muscles on the hand and arm, where a significant portion of the movement is determined by the "muscle memory" of an individual writer. Fourth, the hand movement is eventually represented as a series of samples of physical states of the hand and fingers. To simplify the physical model of a moving hand, we want to only focus on a single point of the hand, typically the tip of the index finger or the center of the palm, depends on the specific application scenarios. Hence, given the hand movement of a person writing a string in the air, we use a certain type of sensor to capture the six degree-of-freedom (6DOF) motion states of that point. Formally, we call the captured data for writing a string in the air as an **IAHW signal**. The specific meaning and format of such a signal depend on the sensor, which is detailed in the next subsection.

Moreover, we can model the string, characters, and strokes can be regarded as hidden states in a stochastic process, where only the signal samples are the observations. In general the IAHW process does not satisfy the Markov property (*i.e.*, signal samples are correlated temporally), and the mapping between signal samples and strokes can be flexible due to the minor variations of the writing speed and amplitude. However, the inherent process in the brain of generating hand motion by writing is acquired and reinforced when a person learns how to write (Klemmer *et al.*, 2006), which indicates that the writing behavior is bound to individuals and persistent in the long term. This can be justified as handwriting signature has been widely used for identity verification for a long time.

Given two or more IAHW signals, we consider four cases (listed as follows), regarding who writes the strings and what the contents of the strings are about.

Case 1 - same writer, same content: For the same person writing the same string in multiple repetitions, we are expected to see signals containing the same sequence of strokes from the same sequence of characters in the same writing style. If this sequence contains a shared secret like a password between the writer and a login server, this shared secret is the foundation of IAHW-based user authentication. Similarly, if this sequence contains some unique identity information specific to the writer like an ID string known by a login server, this unique identity information is the foundation of IAHW-based user identification.

Case 2 - different writers, same content: On the one hand, if different users happen to choose the same string as the shared secret as passwords, or an imposter intentionally writes the same string as the password of a legitimate user, there will be slight differences in the writing style even if they contain the same sequence of strokes, and we believe these different writing style matters for user authentication or user identification. On the other hand, different people can write the same word in the same language as general text input like messages on the phones, and such IAHW signals are derived by the same string determined by the vocabulary of the language. Hence, for IAHW text input methods, different writing style should be tolerated, and only the content should be recognition.

Case 3 - same writer, different contents: As the same person may have multiple accounts with different account IDs or different passwords, in this research, we assume the content difference matters. However, we do not study how to re-identify the same user from different writing content merely based on the writing style. Also, for IAHW text input methods, the recognized input should only depend on the content of the string regardless of the writers.

Case 4 - different writers, different contents: Similar to case 3, if these signals are representing different account IDs or different passwords, they should be distinguished. Although the differences in the writing style may help, we suspect that given large amount of writers, such help would be marginal. Writing is an acquired skill from training and many people write in a very similar way as the training process are all similar, unless a writer specifically write a string in a unique way like signing a signature. However, if these signals are representing different text, such differences in the writing style can make the word recognition task harder, especially if users write them in illegible ways.

As the IAHW content may have significant meanings for different purposes, from this point of view, a signal is nothing but a time series mapped from the space of character sequences to the space of sensor measurements. However, this mapping function contains two types of additional hidden information, the personal writing style and uncontrolled variations in the writing. Hence, even the same person writes the same string twice, the obtained two signals are not exactly the same. Since the extraction of a stable sequence from a handwriting signal is difficult, in this research, we compare the signals in some way as an indirect method to compare the character sequences for our research tasks. Also, we use a learned model to extract a representation of IAHW signals and map the representation to account IDs for user identification. Similarly, other learned models are used to map the IAHW signals to word IDs in a lexicon for word recognition. For all research tasks in this project, we do not employ character level annotation and signal segmentation. Only a class label of a whole signal is used. For user authentication and user identification, this class label is the account ID associated to the signal (detailed in chapter 4, 5, and 6). For word recognition, this class label is the word ID in the lexicon associated to the signal (detailed in chapter 7).

3.3 IAHW Data Collection Device

Our data repository contains multiple datasets, all collected using two types of devices: a 3D camera (the Leap Motion controller, shown in Figure 3.2) and a wearable device (a custom-made data glove with inertial sensors, shown in Figure 3.3).

The Leap Motion controller can capture the 3D position of each joint of the hand at about 110 Hz with an 135 degree Field of View (FOV) and a depth range of 10 cm to 60 cm. The device is essentially a few infrared LEDs and a stereo pair of wide angle infrared camera with a fixed baseline of about 45 mm. Meanwhile, the device is connected to a client computer which runs a driver program to estimate the pose of the hand and stream the results out from a USB port. As mentioned by the manufacturer, this device does not rely on structured light or Time-of-Flight methods to estimate the depth. Instead, 3D poses of the hand skeleton structure are directly estimated from the two synchronized stereo images. However, in the preprocessing step detailed in the next subsection, only one point of the hand skeleton is used. Due to the nature of camera projection, users are restricted to write in the FOV of the camera within the specified depth range. This device is denoted as the "camera device" or the "camera" for short.

The data glove has a Micro-Electro-Mechanical System (MEMS) Inertial Measurement Unit (IMU) on the tip of the index finger, which can measure the 3D acceleration

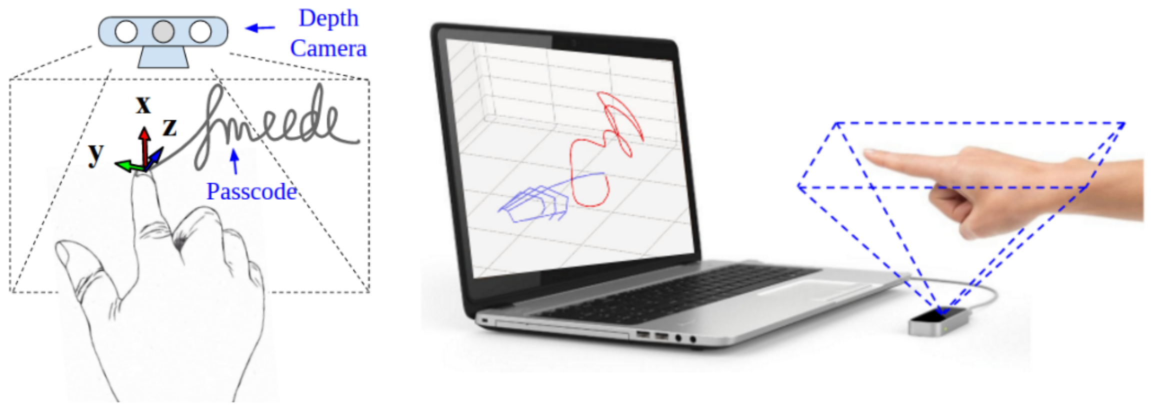


Figure 3.2: The Leap Motion Controller for Finger Motion Capturing

in ±4g and angular speed in ±2000 degree-per-second at 100 Hz. Specifically, we use the Bosch BNO055 IMU in this project. Besides the index finger, there is also an identical IMU on the thumb of the glove device, but data from that sensor is not used in this project. Hence, an IAHW signal obtained from the data glove device always contains information of the movements of the tip of the index finger. Due to the nature of the IMU, users are not restricted to write in a specific region or pointing to a specific orientation. The data glove has a few variants with slight differences in the glove material and microcontrollers. However, the motion sensors are the same and the obtained signals have identical formats. This device is denoted as the "glove device" or the "glove" for short.

Both devices are relatively inexpensive (∼$100) and their capabilities are similar to existing VR/AR platforms with a gesture user interface. Although the prototype system built by us in this project utilizes these two devices, our framework is not restricted to these devices. Also, our framework is not designed to only use devices dedicated for the login or word input purposes. Instead, hand movement tracking is considered a basic function of the gesture user interface.

The basic unit of data in our datasets is called an IAHW signal, which is a piece

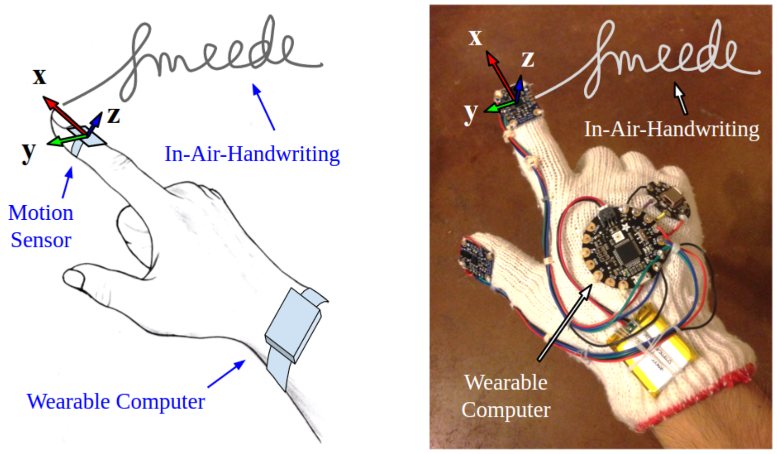


Figure 3.3: The Custom Data Glove Device for Finger Motion Capturing

of hand movement of writing a string in the air recorded using our device. There are four types of signals.

A. The raw signal obtained using the camera device.

B. The raw signal obtained using the glove device.

C. The preprocessed signal from the raw signal obtained using the camera device.

D. The preprocessed signal from the raw signal obtained using the glove device.

These four types of signals share a lot in common, and the formats of all four types of signals are similar. In most cases, there is no need to specify whether a signal is preprocessed or not and we collectively call it a "signal". As a signal is a time series of data samples from a sensor, it is denoted as a vector $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_l)$, and l indicates that there are l samples in this series. Each sample $\mathbf{s}_i$ is another vector $\mathbf{s}_i = (s_{i1}, s_{i2}, ..., s_{id})$, which represents an d individual values obtained by the sensor with d axes. Instead of this generic representation using vector-of-vectors, we can also use an l-by-d matrix S to represent a signal if a sample in different sensor

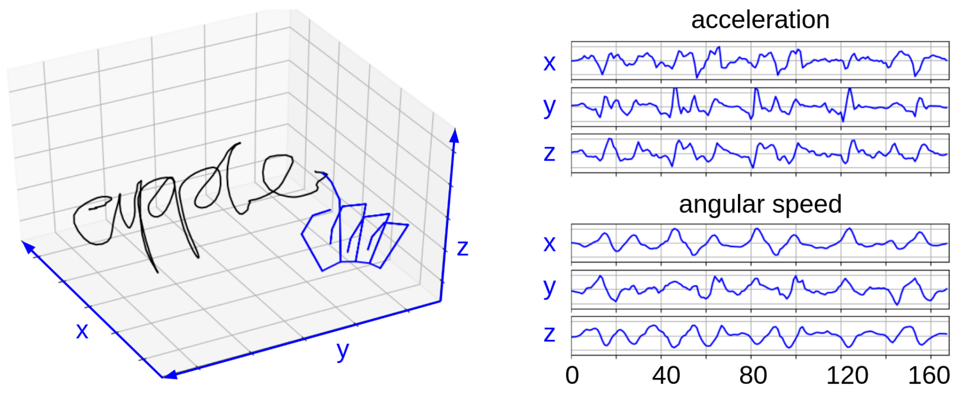


Figure 3.4: An Example of Raw Signals Obtained From the Two Types of Devices

axes are obtained simultaneously (which is the typical case in this dissertation if we only use one kind of device at a time).

For example, if a raw signal is obtained using our data glove device, each sample $\mathbf{s}_i$ may have three axes representing the acceleration of the fingertip of the right hand of a user along the x, y and z-axes, and another three axes representing the angular speed of the same point. The whole signal $\mathbf{s}$ may have 250 samples at 100 Hz. On the other hand, if a raw signal is obtained using our camera device, each sample $\mathbf{s}_i$ may contain a collection (x, y, z) positions for each of the joint of the hand as well as other attributes such as the confidence score of the hand skeleton pose estimation, etc. Since both sensors can generate one sample for different sensor axes simultaneously, either the vector-of-vectors representation or the matrix representation can be used. An example of two raw signals from the two types of devices is shown in Figure 3.4. In this figure, on the left side we show a trajectory of writing "apple" together with the hand skeleton captured by the camera device, and on the right side we show a signal containing the acceleration and the angular speed obtained from the data glove when writing the same word.

3.4 IAHW Signal Preprocessing

In our research, only the tip of the index finger or the center of the palm is interested, because all the joints on the hand usually move in the same way as the wrist moves for writing. Occasionally, for a few users, there are small movements of the index finger relative to the palm during the writing, which is considered insignificant in our data collection and processing procedures. Specifically, for user login using the camera device, we discover that the tracking results of the fingers are not always reliable, and hence, the center of the palm is used. One reason is that users typically write in an illegible way and move the hand very quickly in the air, which may exceed the tracking speed of the sensor. Another reason is that users may occasionally point the index finger to certain directions such that the fingers are partially occluded by the palm from the perspective of the camera. However, for general text input using the camera device, users typically write slowly and in a legible way. Thus, the tip of the index finger is used because its movement is more significant. As for the glove device, all data is obtained from the IMU on the tip of the index finger.

Based on this principle of modeling hand movements using a single point, for a preprocessed signal, it always has 18 sensor axes (*i.e.*, d is always 18) representing the motion states of that point to the second order. The meaning of the sensor axes are as follows.

- 0 - 2: position in x-y-z.
- 3 - 5: velocity in x-y-z.
- 6 - 8: linear acceleration in x-y-z (the gravity vector "g" is removed).
- 9 - 11: orientation (the q_x, q_y, and q_z components of a unit quaternion).
- 12 - 14: angular speed in x-y-z.

- 15 - 17: angular acceleration in x-y-z.

Note that a preprocessed signal is actually device agnostic, and hence, it always has these 18 sensor axes regardless of the hand movement capture devices. The following preprocessing steps are applied on the client side.

Step 1) Down-sampling: The raw sensor data is down-sampled to 50 Hz with linear interpolation on each sensor axis.

Step 2) Posture normalization: The signal is translated and rotated to a reference frame such that the mean of all point positions is the origin, the "average pointing" direction of the hand as the x-axis, and the "average downward direction of the palm" is the on the XOZ plane. For the data glove with inertial sensors, we also remove the influence of the gravity on the acceleration axes. This step is crucial since it removes the uncertainty of the hand posture relative to the sensor coordinate system. However, as the reference of the "standard pose" can only be approximated from a signal, it is not perfect.

Step 3) State estimation: The indirect states are derived from the raw sensor data. Also, we fix any missing signal samples due to the limited capability of the motion capture device [2] . For the camera device, the position is directly observable, and the orientation is estimated by the palm facing direction and hand pointing direction. The orientation is represented by the three imaginary components of a unit quaternion. Higher order states such as the speed and acceleration are derived by differentiation of the position and orientation. For the glove device, the orientation is estimated from the angular speed, while the position and speed are estimated by inertial dead reckoning with a regularizer. The regularizer works like a virtual attractive force to the origin in order to prevent unbounded drift caused by sensor

[2]The Leap Motion controller has a fixed field of view. When the hand moves out of the field of the sensor, samples are not generated. The glove device does not have such an issue.

bias or noise in integration. Although the obtained position does not follow the actual hand movement trajectory (which cannot be accurately obtained solely with this IMU), it is still useful for further processing because of the same regularization algorithm applies to all signals.

Step 4) Low-Pass filtering: remove the high-frequency components above 10 Hz. We believed that any finger movements with a higher frequency than 10 Hz in handwriting cannot be repeatedly generated by a person.

Step 5) Trimming: The samples at the start and the end of the signal are removed where the hand does not move intensively. In practice, the signal can be trimmed in a more aggressive way because we observed that the user behavior can have larger uncertainty at the beginning and the end.

Step 6) Amplitude Normalization (Optional): The data of each sensor axis is normalized individually to zero mean and unit variance, *i.e.*, $s_{ij} \leftarrow (s_{ij} - \mu_j)/\sigma_j$ where $\mu_j = mean(s_{1j}, ..., s_{lj})$, $\sigma_j = std(s_{1j}, ..., s_{lj})$.

Step 7) Temporal Normalization (Optional): The signal length is normalized to 256 samples with linear interpolation

The last two steps of preprocessing are only needed for specific tasks detailed in later chapters. The actual implementation is slightly more complicated and readers are recommended to refer to the code for further details in section 3.6. The preprocessing is performed individually on each raw signal. After preprocessing, signals will have the same format and further processed in each task regardless of the original device.

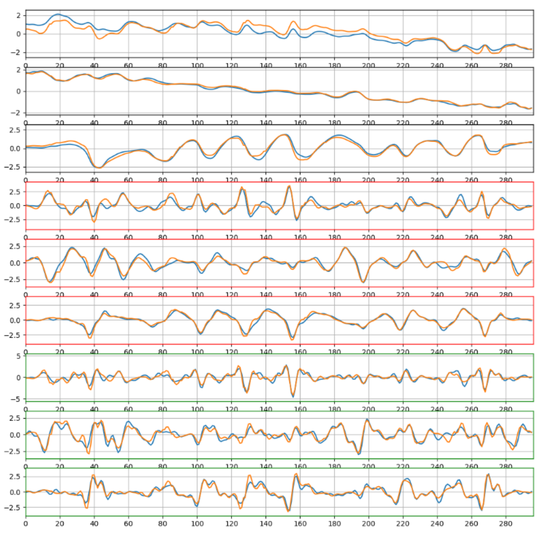
Besides, there is another important preprocessing step not related to the further tasks – **Data Anonymization**. We removed any tags and metadata that is sensitive to user identity. Files storing the data are renamed to strings like "user001_label001_repetition01.csv" and their creation time is reset. As a result, other researchers using this dataset will be unable to deduce the true identity of a subject participating our data collection. It should be noted that many participants use their name as an account ID string, *e.g.*, “Jacob”, and if it is written in a readable way, the existence of a person named “Jacob” can be observed participating our data collection. However, given the fact that a name is not a secret and there are many people with the same name, other researchers using this dataset will not be able to pin-point a person in the real world who writes this string in participation in our dataset easily.

3.5 Formulation of IAHW Signal Variation

Given multiple preprocessed IAHW signals generated by the same person writing the same string, assume one of the signal $\mathbf{s}$ is aligned to an imaginary “standard” signal of that person writing that string in a fashion such that the variation of writing speed is removed, it can be decomposed as

$$\mathbf{s}_i = \mathbf{t}_i + \mathbf{e}_i, \qquad \mathbf{e}_i \sim N(0, \mathbf{\Sigma}_i),$$

where $\mathbf{t}_i$ is a constant vector determined by the content of a piece of IAHW, and $\mathbf{e}_i$ is a vector of Gaussian random variables caused by the sensor noise and unintentional small hand movements. Since $\mathbf{e}_i$ is from different orthogonal sensor axes, we assume these axes are independent, *i.e.,* $\mathbf{\Sigma}_i = \boldsymbol{\sigma}_i^2 I$, where $\boldsymbol{\sigma}_i^2 = (\sigma_{i1}^2, \sigma_{i2}^2, ..., \sigma_{id}^2)$. As mentioned previously, there are uncontrolled variations in the writing behavior of a person such that two signals obtained from writing the same string twice are not identical. Such variations can be further decomposed into two parts - temporal variation (i.e., along the temporal dimension of the signal $\mathbf{s}$) and intensity variation (i.e., along the amplitude dimension of each sample $\mathbf{s}_i$). Here, the intensity variation is represented by $\boldsymbol{\sigma}_i^2$. The temporal variation is more complicated, which can be modeled by the


(a) with alignment

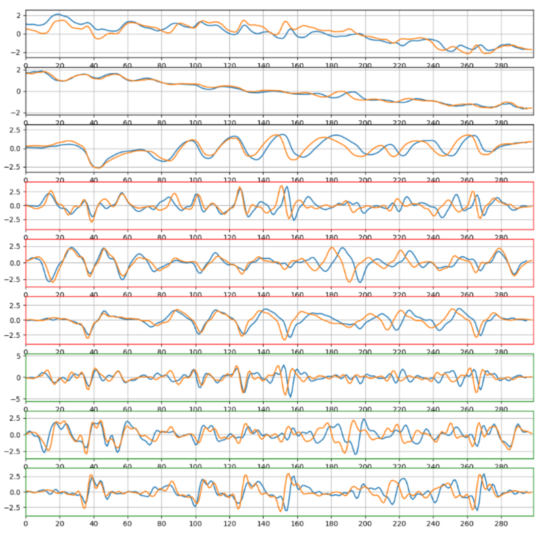

(b) without alignment

Figure 3.5: An Example of Signal Alignment

alignment of the signal $\mathbf{s}$ to the "standard" signal in the format of a "warping path". In reality, there is no way to define such a "standard" signal, but we found that in a collection signals, any one of them can be used as a "standard" signal such that the temporal variation of other signals can be measured relative to that signal. More details are presented in chapter 4. An example of signal alignment is shown in Figure 3.5. In this figure, we show a pair of signals generated by the same person writing the same string "FMKit" twice, with alignment on the left and without alignment on the right. For alignment, both signals are preprocessed and normalized in amplitude. The alignment is computed using Dynamic Time Warping (DTW) (Berndt and Clifford, 1994). Only the first nine sensor axes are shown in this figure.

An approximation of $\mathbf{t}_i$ and $\boldsymbol{\sigma}_i$ can be computed by a set of signals $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, ..., \mathbf{s}^{(K)}\}$ as follows,

$$\hat{\mathbf{t}}_i = \frac{1}{K}\sum_{m=1}^{K} \mathbf{s}_i^{(m)}, \qquad \hat{\boldsymbol{\sigma}}_i^2 = \frac{1}{K}\sum_{m=1}^{K} (\mathbf{s}_i^{(m)} - \hat{\mathbf{t}}_i)^2.$$

Here $\hat{\mathbf{t}}_i$ is the Maximum Likelihood Estimation (MLE) of $\mathbf{t}_i$ and $\hat{\boldsymbol{\sigma}}_i^2$ is the MLE of $\boldsymbol{\sigma}_i^2$. This set of signals $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, ..., \mathbf{s}^{(K)}\}$ are obtained by the same user writing

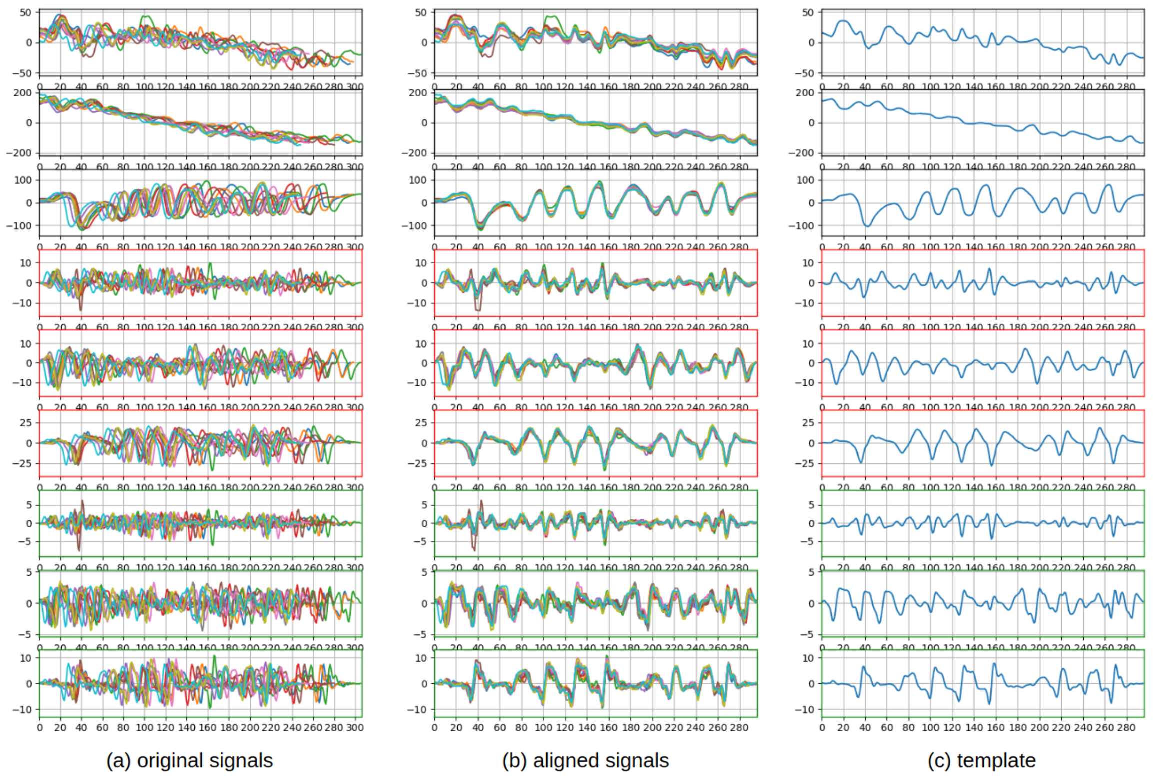


Figure 3.6: Example of Signal Alignment and Template Construction

the same string, typically in the registration step for user authentication or user identification, and typically they are all aligned to the first signal. The number K is chosen as a trade off of usability and security performance. In our project, it is between two and five, and five is typically used. If these signals are used in the user authentication task or the user identification task, $\hat{\mathbf{t}}$ is stored as the *id_template* or the *passcode_template*, depending on whether the set of signal is obtained by writing the ID string or the passcode string. The system can match a signal in the login request and such a template to determine whether this login request should be accepted or rejected. $\hat{\boldsymbol{\sigma}}$ can also stored together with the template to indicate the template uncertainty. Alignment and template construction are not needed for word recognition.

An example of template is shown in Figure 3.6. In this figure, ten signals from the same person writing the same string "FMKit" are shown on the left side; in the middle, the corresponding ten aligned signals are shown; on the right side, the generated template is shown shown. Similar to Figure 3.5, the alignment is computed using DTW and only the first nine sensor axes are shown. However, in this figure, the signals are not normalized in amplitude. Typically, we normalize the signals and alignment them together. However, the alignment can also be done on a single sensor axis and the same warping path can be applied to all sensor axes.

3.6 The FMKit Code Library

We have developed a code library in Python 3 to interpret, manipulate, and visualize a signal as well as a collection of signals in a dataset. This library has been made open source and released in our GitHub repository [3] . This library depends on NumPy and Matplotlib. It contains the following Python modules.

- fmsignal.py: abstraction of the signal and methods to manipulate the signal.
- fmsignal_vis.py: visualization methods of the signal.
- fmsignal_demo.py: demonstration of the usage of the signal data structures and methods, as well as signal visualization.
- fmdataset.py: abstraction of the datasets and methods to use the datasets.
- fmdataset_demo.py: demonstration of the usage of the datasets.
- fmbrowser.py: a simple interactive GUI-based software to browse the signals in our dataset, as shown in Figure 3.7.

[3]https://github.com/duolu/fmkit

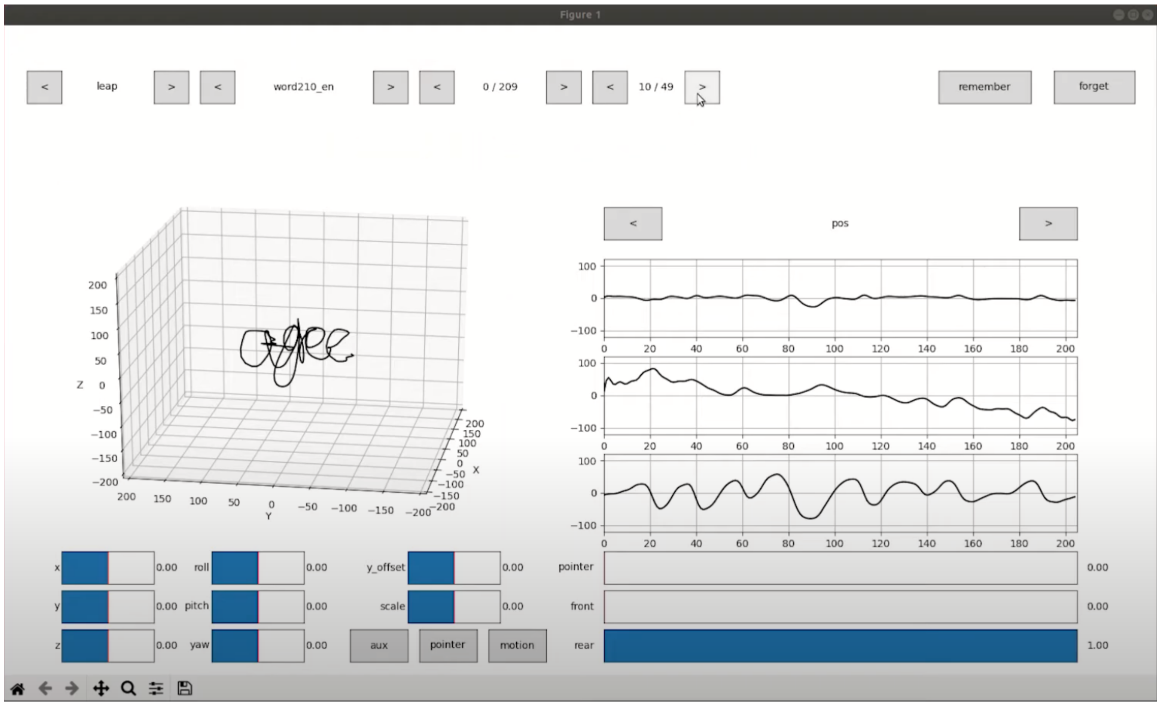


Figure 3.7: The Interactive Gui-based Software to Browse Our Datasets

- pyrotation.py: an implementation of various representations of 3D rotation including unit quaternion.
- pyrotation_demo.py: demonstration of the usage of our implementation of representations of 3D rotation.

Besides, we also implemented a version of DTW through the Python C API and NumPy C API to speed up the signal alignment algorithm. Documentation and a detailed user manual is also provided online [4] . Moreover, the online documentation also provides demo videos and dynamic contents such as animations of replaying a piece of IAHW, which cannot be directly shown in this dissertation.

[4]https://duolu-fmkit.github.io/

3.7 The FMKit Data Repository

We have constructed the following datasets. Details about the data collection procedures are provided at the end of this section. We have obtained an institutional IRB approval from Arizona State University (STUDY00008279 and STUDY00010539)

(1) ID-passcode: We asked each user to create two distinct meaningful strings as an ID string and a passcode. Once the strings are created, we ask the user to write each string in-the-air for 5 repetitions as registration and another 5 repetitions as login attempts, using each device. This simulates the normal sign-up and sign-in procedure. In total, there are 360 (strings) * 10 (repetitions) * 2 (devices) = 7,200 signals.

In this dataset, the content of the string is determined by the user, which may include alphanumeric letters, characters in a language other than English, or meaningful symbols that cannot be typed on a keyboard. We strongly recommend the users avoid using the ID or password they use in their daily lives, but we did not forbid the users to write their name as ID strings because most of the finger trajectory of the IAHW are illegible such that the content is hard to read for a person replaying it on a computer. 58 users wrote words in Chinese for both strings. The remaining 122 users wrote alphanumerical strings or strings with individual symbols similar to letters (*e.g.*, triangles, tilde, etc). We asked each user to create a unique ID string so that all 180 ID strings have unique contents. However, there are no such constraints for the passcodes. Surprisingly, all 180 passcode strings also have unique contents. In fact, there is only one user's ID string that has the same content as someone else's passcode. They are created by two different users, who both choose the string "helloworld" but for different purposes. Also, the way of writing these two "helloworld" strings are different and we can treat them as two different strings if needed. As a general guideline, we discourage simple strings for passcode such as “123456”, and we recommend that the length of an alphanumerical string be at least eight characters. However, it is not a strict restriction, and a few users still created simple and short strings like “USA” or “abc123”. Many users chose a combination of letters and numbers like a traditional password typed on a keyboard, e.g., “stevenC909”, or meaningful phrases, *e.g.*, “PondLakeOcean”, while some users chose a combination without an obvious meaning, e.g., “sqwz1457”. There is also one user that draws three five-pointed stars as a passcode. Because the content of such an IAHW passcode can be very different from a typed password, we use the term “passcode” instead of “password”. For privacy reasons, the contents of the ID and passcode are not published in our datasets.

(2) ID-passcode-collision: We asked 10 skilled imposters to imitate the IAHW of the ID and passcode generated by the users in the “ID-passcode” dataset. In this setting, the imposters know the semantic meaning of the strings written by the legitimate users, but the imposters have not seen the legitimate users writing the ID and passcode in the air, which simulates the case of collision of ID or passcode, or spoofing attack with semantic leakage. The imposters are very familiar to the data collection system. For each string in the “ID-passcode” dataset, we ask each imposter to write it with 5 repetitions using each device. In total, there are 360 (strings) * 10 (imposters) * 5 (repetitions) * 2 (devices) = 36,000 signals.

(3) ID-passcode-spoofing: Similar to the “ID-passcode-collision” dataset, we asked 10 skilled imposters to imitate the IAHW of the ID and passcode generated by the users in the “ID-passcode” dataset. In this setting, the imposters can watch the recorded video of the IAHW and they will be informed with the semantic meaning of the IAHW. The imposters were very familiar to the data collection system and they went through special brief training for this task. There are 180 string in the “ID-passcode" dataset where the user allows us to record a video of the hand movement (exactly 90 ID strings and 90 passcodes corresponding to 90 users). For each of the 180 strings, we ask each imposter to write it with 5 repetitions using each device. In total, there are 360 (strings) * 10 (imposters) * 5 (repetitions) * 2 (devices) = 18,000 signals.

(4) ID-passcode-persistence: We kept collecting the sign-in IAHW of a subset of the users in the "ID-passcode" dataset for a period of about 4 weeks, which simulates login activity in the long term. In that dataset, the user wrote each string 5 repetitions as registration, while in this dataset, besides the registration, the users wrote the strings for the account ID and the account passcode 5 repetitions in a day as a session. There are 10 sessions in total and the data of each session is collected in a different day. 40 users participated in this dataset. In total there are 80 (strings) * 10 (imposters) * 5 (repetitions) * 2 (devices) = 8,000 signals.

(5) word-210: We asked 10 participants to write 210 distinct English words and 210 distinct Chinese words. For each word, each person wrote it with 5 repetitions using both devices. In total, there are 2 (languages) * 210 (strings) * 10 (writers) * 5 (repetitions) * 2 (devices) = 42,000 signals.

Besides, we are currently working on a superset of the word-210 dataset, named **word-10k**. As the name suggests, this dataset contains 10,000 English words and 10,000 Chinese words, where each English word is written by 12 different users with one repetition and each Chinese word is written by 12 different users with one repetition. In total, there will be 480,000 signals. However, the progress is delayed by the pandemic.

The detailed procedure for the Leap Motion controller device is shown in Figure 3.8. The data collection is done on several identical Linux desktop and Linux laptop machines. On each machine, three Linux console terminals are opened, one for the

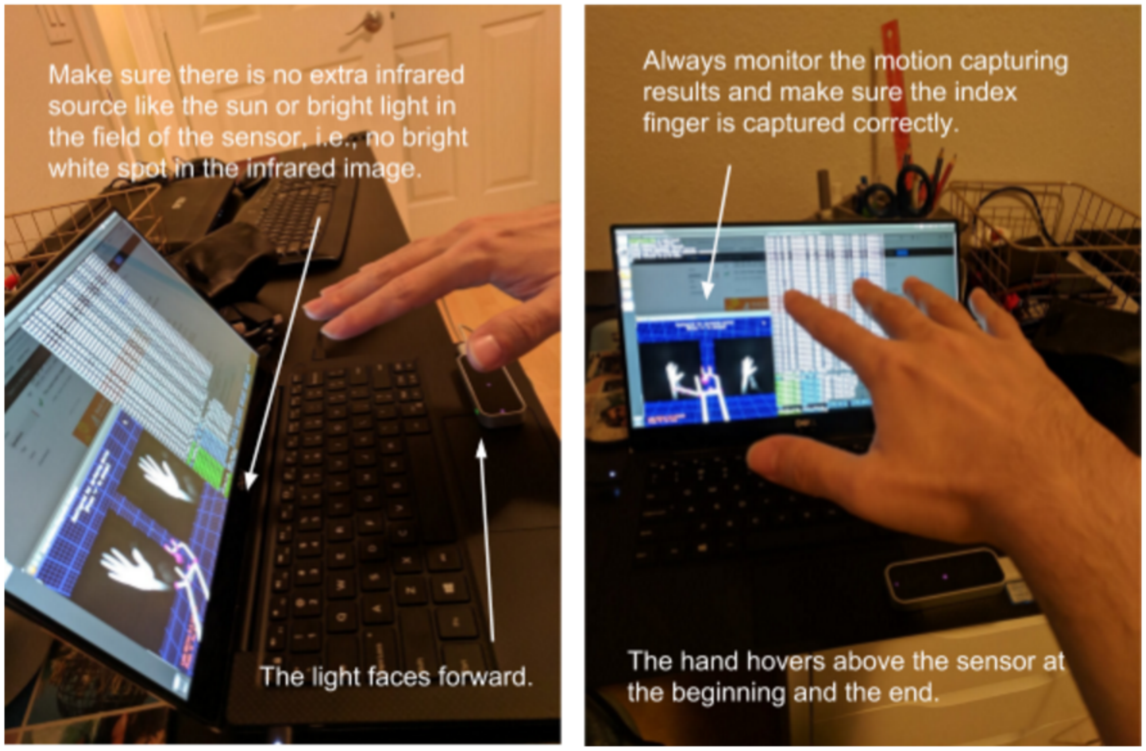

Make sure there is no extra infrared source like the sun or bright light in the field of the sensor, i.e., no bright white spot in the infrared image.
The light faces forward.
Always monitor the motion capturing results and make sure the index finger is captured correctly.
The hand hovers above the sensor at the beginning and the end.

Figure 3.8: Data Collection Procedure for the Leap Motion Controller Device

Leap Motion daemon process (i.e., typing "sudo leapd" in the command line), one for the Leap Motion control panel as well as the visualizer (typing "LeapControl-Panel - -showsettings" and then launch the "diagnose visualizer"), and the last for the actual data collection program (typing "python ./client_leap.py xxx.txt" in the command line or launch the client program with a GUI). At the beginning of the writing of one string, the participant typically hover the hand above the sensor, like that in the following figure. Once the data collection starts, i.e., immediately after the "client_leap.py" script executes or the "start/stop" button on the GUI client is clicked, the hand starts to move. At the end of the writing of one string, the hand restores to the initial position or stop at the last stroke. The data collection program can be stopped by the user, i.e., press the "space" key or clicking the "start/stop" button (typically for the "word-210" dataset), or it can automatically stop if the user's hand stops moving for roughly one second (typically for the datasets related to ID strings and passcode strings). The participant can write from left to right or just write every letter at the same place.

The procedure for the data glove is similar. However, only one Linux console terminal is needed for the actual data collection program. Different from the Leap Motion controller where the user must write within the field of view of the sensor, the data glove does not have such constraints. The user can just start moving after the data collection program starts to collect data. When the user finishes writing, the data collection program can be stopped in the same way as that for the Leap Motion controller.

Each captured handwriting signal is an individually named file. In the "ID-passcode" dataset, for the ID string, it is typically "id_id_xx.csv", where "xx" the sequence number of the repetition. For example, if the ID string is "alice", the file is named as "alice_alice_0.csv" for the first repetition. For the passcode, similarly, it is "id_passcode_xx.csv". For example, if the ID string is "alice" and the passcode is "FMKit", the file name is "alice_FMKit_0.csv". If the user chooses to write in Chinese, pinyin is used in the file name to indicate the content. To protect the privacy, We removed any tags and metadata that is sensitive to user identity. Files storing the data are renamed to something like "user0_user0_0.csv" for an ID string and "user0_passcode0_0.csv" for a passcode. In this way, we can immediately know which user generates which signal for what content from the file name while debugging the code and tuning trained models.

For the "ID-passcode-collision" datset and the "ID-passcode-spoofing" dataset, the procedure of spoofing with semantic leakage is similar to the "sign-in" step in the "ID-passcode" dataset, i.e., only five repetitions for each string. The file name is in the format of “collision_string_xx.csv” or “spoof_string_xx.csv”. For example, if the imposter is designated to be “spoof8”, and the spoofed string is “FMKit”, the file name is “spoof8_FMKit_0.csv” for the first repetition. To facilitate dataset loading and saving, an alternative naming scheme with a slight different difference is also used. It also follows the format “collision_string_xx.csv” or “spoof_string_xx.csv”. All imposters are just designated simply as “collision” or “spoof”. However, “spoof_string_0.csv” to “spoof_string_4.csv” are the five repetitions for the first imposter, “spoof_string_5.csv” to “spoof_string_9.csv” are the five repetitions for the second imposter, and so on. For the same privacy reasons, the “string” part in the file name are renamed to something like “user0” or “passcode0”.

For the long-term persistence study dataset, the procedure and file name convention are the same as sign-up and sign-in. However, the sequence number of repetitions will linearly increase. For example, if the ID string is “alice” and the passcode is “FMKit”, for the five signals in registration, the file name will be “alice_FMKit_0.csv” to “alice_FMKit_4.csv”. For the first login session, the file names will be “alice_FMKit_5.csv” to “alice_FMKit_9.csv”. Similarly, for the second login session, the file names will be increased to “alice_FMKit_10.csv” to “alice_FMKit_14.csv”, and so on. For the same privacy reasons, the files are renamed in the same pattern as that for the “ID-passcode” dataset.

For the “word-210” dataset, the procedure and file name convention are the same as the collision dataset and the spoofing dataset. Instead of using “spoofer”, “user” is used in the file name. For example, if the participant is designated as “user8”, and it writes the English words “agree”, for the first repetition, the file name would be “user8_agree_01.csv”. The same alternative naming scheme is also used, i.e., “user_agree_0.csv” to “user_agree_4.csv” are the five repetitions for the first user, “user_agree_5.csv” to “user_agree_9.csv” are the five repetitions for the second user, and so on. For Chinese words, pinyin is used in the file name. These files do not have privacy issues, and hence, they are not anonymized.

3.8 Supported Research Tasks

A signal in any of the five datasets defined in the previous subsection can be represented as a tuple $(\mathbf{x}, y, z)$, where $\mathbf{x}$ is the signal data, y is a label indicating the meaning of the string, and z is a label indicating the identity of the writer. For example, in the "ID-passcode" dataset, given all the signals collected from writing ID strings, there are 180 different labels for y and 180 different labels for z, where each label y represents an individual ID and each label z is the same as y. Similarly, given all the signals collected from writing passcode, there are 180 different labels for y and 180 different labels for z, where each label y represents an individual passcode and each label z represents the individual ID corresponds to that passcode. The same rule also applies to the "ID-passcode-persistence" dataset. For the "ID-passcode-collision" dataset and the "ID-passcode-spoofing" dataset, this label y also represents the corresponding imitated ID or the passcode in the "ID-passcode" dataset, while the label z of each signal represents the identity of the imposter. Typically, we do not care which imposter generated which signal, while we only care whether the signal is generated by an imposter or the original user who created it in the "ID-passcode" dataset. As for the "word-210" dataset, the label y of each signal indicates its word ID in the lexicon of 210 words, and the label z indicates the identity of the writer. However, since this dataset is designed for word recognition, we generally do not care the writer identity but only distinguish whether two signals are generated by the same writer or not. Based on this definition, we propose the following three types of tasks.

3.8.1 User Authentication

The user authentication task is essentially a binary classification task, which serves as the function of "typing a password" over a gesture interface. In this task, each of the 180 passcode strings in the "ID-passcode" dataset is considered as the passcode of an account. Optionally, we can mix the ID strings and the passcode strings so that there are 360 passcode strings for this user authentication task, which can simulate the situation where one user has two accounts. This can double the dataset size in a certain way for this user authentication task.

Using the "ID-passcode" dataset, The five signals for registration are used to construct templates and train discriminative models for authentication purpose; the five signals for login requests are used to test the performance. Specifically, consider an account has been constructed by a set of registration signals $\{\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^K\}$ in the "ID-passcode" dataset with the same label y. Given another login request signal $(\mathbf{x}, y')$ in the "ID-passcode" dataset, the authentication system should accept the signal if y is the same as y', *i.e.*, it is generated by the same person writing the same string; otherwise, the authentication system should reject it. Note that this binary classifier is specific to an account constructed by those signals with the specific label y corresponding to this account. For each account, there is a unique binary classifer. All these binary classifers construct the authentication system, although they may share the same set of parameters in certain cases (detailed in chapter 4).

For each account, we define those testing signals with $y = y'$ as positive testing signals, and those testing signals with $y \neq y'$ as negative testing signals. The signals in the "ID-passcode-collision" and "ID-passcode-spoofing" datasets are used to evaluate the performance in scenarios with adversaries, *i.e.*, all of them are negative testing signals that should be rejected. in a realistic setting, the collision and spoofing data cannot be obtained by the users in normal registration procedures, and hence, they are in general not available for training models. The signals in the "ID-passcode-persistence" dataset are used to evaluate the authentication system performance in the long term, *i.e.*, all of them are positive testing signals that should be accepted.

Due to the imbalance of the number of positive testing signals (only five) and the number of negative testing signals (potentially thousands or more) for each account, we use the evaluation metrics commonly used in biometrics systems such as False Accept Rate (FAR) and False Reject Rate (FRR) are used, instead of precision and recall. The are defined as follows.

FAR = #{false accepts} / #{negative testing signals}

FRR = #{false rejects} / #{positive testing signals}

Here "#{}" means the number of elements in a set. Note that FAR and FRR can be applied for one specific account or a collection of accounts. In the later case, the number of testing signals for each account must be the same. Otherwise, the formula must be changed to use the average of the FAR and FRR for each account.

Usually, the authentication system has a configurable discriminative threshold that can be changed to trade off FAR and FRR. For example, if the threshold is smaller, there will be more false rejects but less false accepts. When FAR is equal to FRR, this rate is called the Equal Error Rate (EER). The EER is very convenient when comparing different authentication algorithms using the same dataset, since it is just a single number. When the threshold is set to a value such that the FAR is 0.001 or 0.0001, the corresponding FRR is called "FRR@FAR=1e-3" or "FRR@FAR=1e-4". They are also denoted as FAR1K and FAR10K in the biometrics community, which may be confusing for readers who are not familiar with biometrics. These two metrics are more important for practical usage than the EER.

Based on our datasets, we set up three different experiments for each account:

A – All testing signals are from the "ID-passcode" dataset.

B – All positive testing signals are from the "ID-passcode" dataset but all negative testing signals are from the "ID-passcode-collision" dataset.

C – All positive testing signals are from the "ID-passcode" dataset but all negative testing signals are from the "ID-passcode-spoofing" dataset.

Note that currently we do not use the "ID-passcode-persistence" dataset for quantitative evaluation. The number of signals in that dataset is quite limited, which can cause bias or uncertainty. Instead, we mainly use that dataset for qualitative evaluation and analysis.

The performance results are averaged over all accounts and baselines are provided in Table 3.1. They are obtained using a per-account SVM classifier on aligned and temporally normalized signals, detailed in chapter 4.

Table 3.1: Baseline Performance for User Authentication

	Leap Motion			data glove		
	EER	FAR 1K	FAR 10K	EER	FAR 1K	FAR 10K
(A)	0.1%	0.1%	0.8%	0.1%	0.2%	0.4%
(B)	2.4%	8.9%	12.7%	1.2%	1.6%	4.3%
(C)	2.9%	26.2%	33.1%	1.3%	4.9%	22.1%

3.8.2 User Identification

User identification task is essentially a classification task, which serves as the function of "typing an account ID". In this task, each of the 180 strings in the first dataset is considered as an account ID instead of a passcode. Optionally, we can mix the ID strings and the passcode strings to double the dataset for this task similar as that in the user authentication task.

Using the “ID-passcode” dataset, the five signals for registration from each user are collectively used to train a classifier, and the five signals for login requests are used to test the performance. Specifically, consider all accounts have been created and the identification system has been set up. Given an signal with a label of y from those testing signals in the ”ID-passcode” dataset, if the identification system predict the label of the signal as y', y' should be the same as y, i.e., it should recognize the signals generated by the same person writing the same content in the same label. Different from user authentication, this classification is not specific to any account. There is only one system-wide classifier. Also different from a passcode, an ID string is usually not a secret.

Besides predicting one single label, this task can be formulated as retrieving the top-k candidate accounts given a query signal. Together with the user authentication function, a login system can take both the IAHW of an ID and a passcode to first narrow down to one or a few candidate accounts using the ID and then check each candidate account using the passcode to authenticate the user. In this case, the ID is not needed to be returned to the user.

Based on our datasets, we set up two different types of experiments.

- For closed-set identification, the system will always locate at least k accounts, *i.e.*, it returns the best top-k predictions no matter how bad they are.

- For open-set identification, the system may end up with no candidate account and reject the query signal, *i.e.*, it returns best top-k predictions that above a certain threshold that is configured by the identification system administrator.

Note that currently we only set up these experiments using the "ID-passcode" dataset for simplicity. More details are provided in chapter 5. The performance metrics are the top-1 and top-5 identification accuracy and the baselines are provided in Table 3.2. They are obtained using a convolutional neural network encoder trained with pairwise loss, and the identification is done by top-k nearest neighbor search in the latent space. Details are provided in chapter 5.

Table 3.2: Baseline Performance for User Identification

	Leap Motion		data glove	
	top-1 accuracy	top-5 accuracy	top-1 accuracy	top-5 accuracy
open-set	93.2%	94.5%	95.7%	96.7%
closed-set	96.7%	98.9%	97.9%	98.8%

3.8.3 Word Recognition

The IAHW word recognition task is essentially another classification task. However, the label y of each signal is from a word lexicon of the "word-210" dataset instead of a set of accounts in user identification. Meanwhile, in this task, behaviors variations from different people writing the same word need to be tolerated. For example, one person may write the same letter "O" clockwise but another person may write it counterclockwise. Also, variations of stroke sequences need to be considered. For example, when writing the same letter "t", one person may write the horizontal stroke as the last stroke but another person may write it as the first stroke. These problems are challenging especially on our dataset with a limited number of writers.

Using the "word-210" dataset, the signals from nine users are used to tain the recognizer, and the signals of the remaining one user are used to test the performance of the recognizer. This is repeated for ten times by selecting the data from each user as testing data (i.e., 10-fold cross validation). All experiments are closed-set, i.e., the recognizer will always provide the best predictions. Experiments using the data of writing English and the data of writing Chinese are conducted independently.

The performance metrics are the top-1 and top-5 classification accuracy and the baseline performance results are provided in Table 3.3. They are obtained using a similar convolutional neural network as in the previous task, averaged over all accounts with 10-fold cross validation. Details are provided in chapter 7. Current results are on a dataset with a small lexicon and we believe it would be more challenging with a large lexicon.

Table 3.3: Baseline Performance for IAHW Word Recognition

	Leap Motion		data glove	
	top-1 accuracy	top-5 accuracy	top-1 accuracy	top-5 accuracy
English	79.7%	93.2%	78.5%	93.8%
Chinese	87.4%	96.8%	83.4%	94.4%

3.9 Summary

In this chapter, we provide FMKit, an open-source library and data repository for analysis of IAHW hand motion signals, which can potentially benefit AR/VR applications, wearable computer platforms, or game consoles. We also propose three research tasks that can be enabled by FMKit, with preliminary baseline performance results. The construction of FMKit is a continuing effort and it will be expanded in the future. We hope it can help other research work in this area and pave the road to practical usage of IAHW as a general input method.

Chapter 4

FMAUTH - USING IAHW FOR USER AUTHENTICATION

4.1 Background

On Virtual Reality (VR) headsets, wearable computers and game consoles, gesture user interfaces are gaining popularity. For example, users can play VR games (*e.g.*, on Sony PSVR) and interact with virtual objects (*e.g.*, on Microsoft Hololens) using direct hand movements captured with handheld controllers or cameras. Meanwhile, many applications need to authenticate a user to unlock the device or log in to an account to access private data and services. This typically requires the user to type a password. However, on these platforms, presenting a physical keyboard or touchscreen is usually impractical, while a virtual keyboard is inconvenient to use without touch feedback. Instead, a gesture or an IAHW passcode, can potentially be used for authentication by leveraging the native gesture user interface. Similar to a password-based system, the user is asked to ”write a passcode string” instead of ”type a password”, and the captured hand-motion pattern is compared with a template for authentication. The difference is that a password only contains typed alphanumerical symbols and special characters, while the content of an IAHW passcode can be a string in any language, a signature, or a doodle as long as it can be naturally reproduced by the user but difficult to mimic by others. Thus, we denote it as a “passcode” instead of a “password”. Besides the passcode content, the hand movement is also influenced by the writing style, and the hand geometry, which may provide better security and flexibility. For example, when the passcode content is leaked, such an authentication system can still defend imposters to a certain extent using the handwriting style.

Hence, it is more like a signature but different from a password. Meanwhile, it has all desired characteristics of a password such as changeability and revocability. Finally, since the passcode content is not necessarily linked to a person, unlike a biometric trait such as a fingerprint, it can protect the privacy of a user.

There are three major technical challenges due to the inherent characteristics of hand gestures and IAHW. First, limited sensor capability, *i.e.*, accurate tracking of fast hand movement with enough resolution is difficult using affordable consumer electronic motion capture sensor. Second, information fuzziness, *i.e.*, there are small variations even for the same user writing the same passcode twice. It is difficult to tolerate these variations and distinguish legitimate users from imposters at the same time, especially without a deep understanding of the signal variations. Third, limited data availability, *i.e.*, the authentication system can only collect a few samples at registration (typically two to five) for usability reasons. To address these challenges, we propose a multifactor authentication framework in this chapter, and our contributions are as follows:

1) We provide an in-depth analysis on the global features of IAHW signals as well as the hand geometry.

2) We designed and implemented a unified multifactor authentication framework called **FMAuth** using two different types of inexpensive hand motion capture devices (a glove with an inertial sensor and a 3D stereo camera). The framework supports multiple authentication algorithms for different scenarios, and it achieves a performance improvement over the existing baseline with limited training data.

3) We evaluated our framework using the FMKit dataset detailed in chapter 3, and compared the results from the two devices. These results show good potential for the usage of IAHW as a user authentication method.

The remainder of this chapter is organized as follows. Section 4.2 describes the architecture of our framework. Then, section 4.3, 4.4, and 4.5 provide the analysis on temporal features, statistical features, and hand geometry features respectively. After that, section 4.6 explains the multifactor fusion framework, section 4.7 shows the empirical evaluation results and section 4.8 analyze the hard cases. Finally, section 4.9 summaries this chapter. This chapter is based on our published research papers (Lu *et al.*, 2018, 2017).

4.2 User Authentication System Model

Our proposed authentication framework consists of seven components (shown in Figure. 4.1): a hand motion tracking device, a client software for signal preprocessing, an account database, three types of feature extractors, and a decision maker.

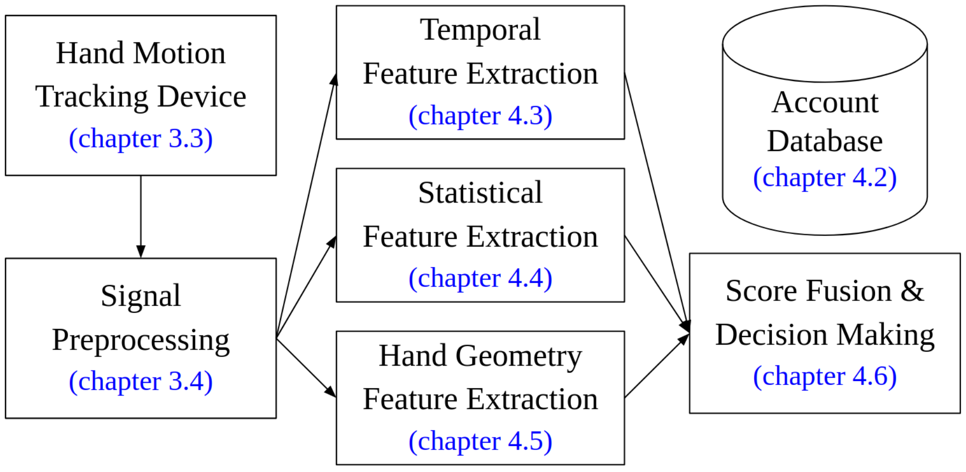

Hand Motion Tracking Device
(chapter 3.3)
Signal Preprocessing
(chapter 3.4)
Temporal Feature Extraction
(chapter 4.3)
Statistical Feature Extraction
(chapter 4.4)
Hand Geometry Feature Extraction
(chapter 4.5)
Account Database
(chapter 4.2)
Score Fusion & Decision Making
(chapter 4.6)

Figure 4.1: Authentication Framework Architecture

The hand motion tracking device and signal preprosessing have been described in chapter 3.3 and chapter 3.4. Both types of devices are used and evaluated in this chapter. Also, all 18 sensor axes of the preprocessed signals are used, and the signals are normalized in amplitude. We denote a preprocessed signal as a $l \times d$ matrix R instead of a vector of vectors. Since the signal is preprocessed, $d = 18$. Given a certain sampling time i and a specific sensor dimension j, the sample value is denoted as R_{ij}. For the convenience of analysis and visualization, a signal is normalized to 256 samples in time using linear interpolation, *i.e.*, $l = 256$. The matching and decision making algorithms for user authentication do not require the signals to be normalized in time.

Here we shown an example of a preprocessed signal obtained from the camera device in Figure 4.2. This signal is obtained by tracking a user writing "123456" in the air, which is probably the worst passcode a user can create. The normalized value of samples for each sensor axis is shown on the left part of this figure in groups, and inside each group, the color red, green, and blue represent the x-axis, y-axis, and z-axis. The trajectory of this signal is shown on the right part of the this figure. Note that this user deliberately writes the passcode from left to right in a legible way for illustration purpose. Majority of the users write every letter or character at the same place very fast in a illegible way like a signature because it is more natural to write in such a way in the air and it is generally harder for imposters to mimic if the writing is illegible. We also segment the trajectory to show different components of the signal for each number as well as the transitioning strokes.

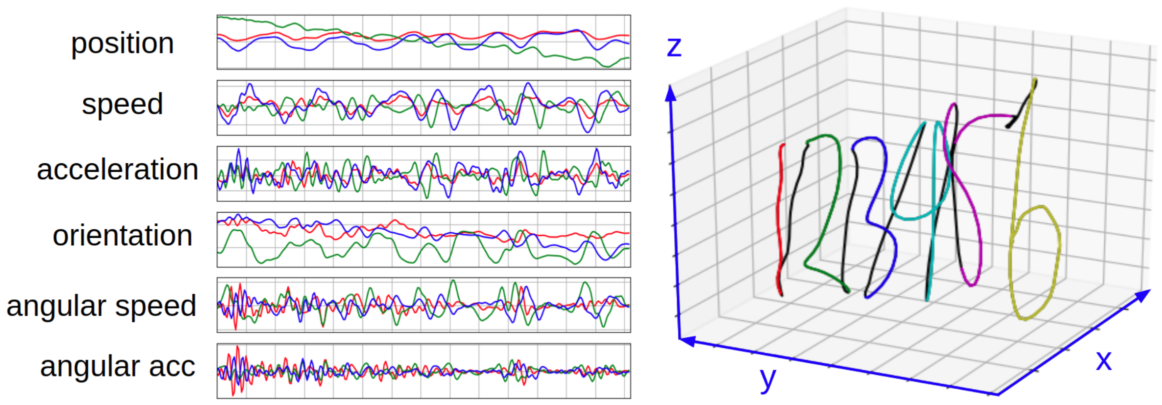


Figure 4.2: An Example of a Preprocessed Signal and Trajectory

At registration, each user is required to create an account by creating a passcode string, and write the passcode string in the air with five repetitions. The system can obtain a set of registration signals in this step. After registration, each account contains a tuple of <ID, templates>. Our framework allows one user to possess multiple accounts with multiple distinct account IDs. The ID can be assigned by the system, and the templates are generated from the registration signals based on each type of features (detailed in the "template generation" step in section 4.3, 4.4, 4.5, and 4.6. Note that this account ID is just an internal ID of an account, not the "IAHW ID string" that a user may use for login purpose. However, it is possible to use an IAHW ID string to retrieve such an internal ID of an account, which is discussed in chapter 5.

At login time, the user will provide an account ID and then writes a passcode for authentication, similar to a password-based system. In this chapter we focus on the authentication problem and we assume the account ID is available, such as remembered by the system, or recognized by the IAHW of an ID string. After finishing writing the passcode, the user keeps the hand still for one second as a termination sign to the system. A login request signal R is generated. Then a matching algorithm processes the signal R and calculates a distance score δ using templates (detailed in later sections). After that, the authentication system makes a decision based on the score δ and a threshold (typically configured by an administrator based on some empirical results using a pilot dataset). If δ is greater than the threshold, output "reject"; otherwise, output "accept".

To facilitate analysis, given a specific account A and a login request signal R, we define an authentication class label c of R in three cases based on the cases defined in chapter 3.2, as follows.

1) $c = \textit{same}$: Both the request signal R and the templates of the account A are generated by the owner of the account A writing the same content. This is legitimate user login. In this case, the request signal R is from the positive testing signals in the "ID-passcode" dataset. To simplify the analysis, we do not use the data from the "ID-passcode-persistence" dataset in this chapter. More analysis for the long term performance is shown in chapter 6.

2) $c = \textit{collision}$: R is generated by a different user other than the owner of the account A but both R and the templates of account A are generated by writing the same content. This can be a passcode collision of two accounts, or an attack from an imposter with the knowledge the passcode content. In this case, the request signal R is from the "ID-passcode-collision" dataset. To simplify the analysis, we do not use the data from the "ID-passcode-spoofing" dataset in this chapter. More analysis for spoofing attacks is shown in chapter 6.

3) $c = \textit{diff}$: R and the templates of account A are generated by writing different content, regardless whether they are generated by the same user or not. This can be a random guessing attack. In this case, the request signal R is from the negative testing signals in the "ID-passcode" dataset.

Generally, in the first case δ should be small and R should be accepted, while in other cases δ should be large and R should be rejected. In practice, a system cannot distinguish case 2 and case 3 and they are only defined separately for analysis purpose. It should be noted that the authentication class label c is defined with the context of a specific account A, and this label can change if the context is changed to a different account. This label is calculated from the content label y and writer label z defined in section 3.8.

4.3 Temporal Feature Analysis

In this section we analyze the signal variation in time (subsection 4.3.1) and sample magnitude (subsection 4.3.2).

4.3.1 Alignment Analysis and DTW Matching

A signal R can be aligned to another signal T by Dynamic Time Warping (DTW) (Berndt and Clifford, 1994) to generate an aligned signal S, as discussed in chapter 3.5. Here T can also be called a template. Specifically, we first run DTW on R and T with all sensor axes and a window constraint of ± 50 samples to obtain a warping path. Then each sample of the aligned signal S_{ij} is calculated by taking the average of those samples of R which are mapped to T_{ij} on the warping path. An illustration of generating an aligned signal from a warping path is shown in Figure 4.3. In this illustration, the signal R with five samples (1, 2, 3, 4, 5) is aligned to a template T with four samples (A, B, C, D), where 1 is mapped to A and B, 2 is mapped to C, 3 is also mapped to C, 4 and 5 are both mapped to D. Since the signal R is aligned to the template T, the aligned signal S will have the same length as T. Hence, for each sample of T, we can compute two indices storing the start and the end of the segment of R that corresponds to this sample of T (*i.e.*, the "start" and the "end" shown in Figure 4.3. With these indices, each sample of S is computed as the average of the corresponding segment of R.

Another example is shown in Figure. 4.4. On the left half of this figure, we show the DTW distance matrix computed by the two signals as an image. The two signals are shown on the edges of this image, where the horizontal one is aligned to the vertical one (*i.e.*, the vertical one is the template). The warping path is shown as a solid line from the bottom left corner of the image to the top right corner of the image.

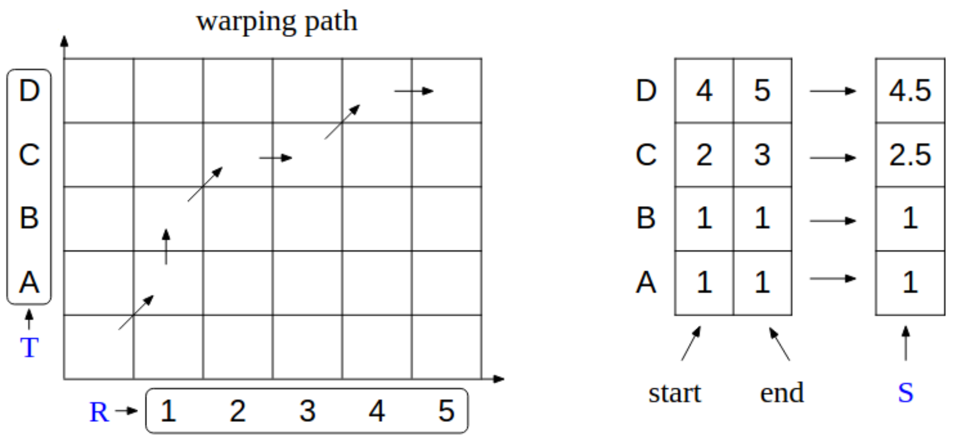


Figure 4.3: An Example of Generating an Aligned Signal From a Warping Path

These two signals are generated by the same person writing the same content twice. If there is no variation in the writing behavior for these two signals, they should be identical (which is impossible for human writers), and the warping path should be a straight diagonal line as the dashed line. Because the writing content and the writing style are the same, there are only small variations in these two signals, and hence, the warping path is close to the diagonal line. The difference between this warping path and the diagonal line can indicate the writing behavior variation in time. Note that these two signals are normalized in time for the convenience of visualization so that the warping path can be directly compared with the diagonal line. The alignment algorithm does not require this normalization step.

This alignment step can be applied on any pair of signals, regardless whether they are generated by writing the same content or not. On the right half of Figure 4.4, we show the alignment results for various cases defined in section 4.2. Here the black signal is the template and the colored signals are login request signals with or without alignment. It can be observed that the signals generated by the same user writing the same content are similar in shape. With this observation, the following DTW matching algorithm is designed as a baseline algorithm.

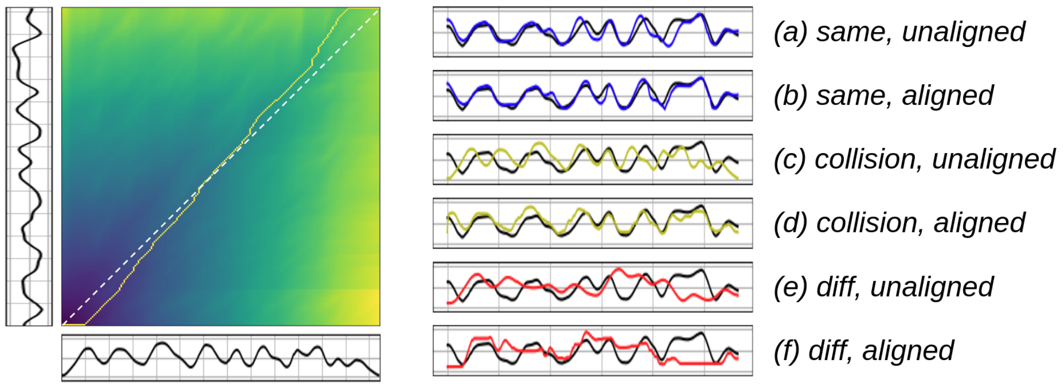


Figure 4.4: Example of a Warping Path and Signal Alignment Results

Template Generation: Given m raw signals at registration for an account, first we preprocess and align them to the first signal to obtain m aligned signals $\{S^{(1)}, S^{(2)}, ..., S^{(m)}\}$. Then we calculate T as follows (essentially element-wise average),

$$T_{ij} = \frac{1}{m} \sum_{k}^{m} S_{ij}^{(k)}.$$

T is stored as the passcode template in the account database.

Matching: Given an account with a template T, and an authentication request signal R, first we preprocess R and align it to T to obtain an aligned signal S. Here S has the same length as T. Then calculate the element-wise distance $D_{ij} = |T_{ij} - S_{ij}|$; finally we calculate the DTW distance score

$$\delta_{DTW} = \frac{1}{ld} \sum_{i}^{l} \sum_{j}^{d} D_{ij}.$$

Here l is the template length, d is the number of sensor axes. δ_{DTW} can be compared with the threshold directly for making a decision, or it can be considered as a feature in the multifactor fusion methods. In fact, this δ_{DTW} has a fairly good discriminative capability to distinguish login request signals from different classes. The distribution of the DTW distance score for all accounts and all login requests in three authentication class labels are shown in Figure 4.5. The overlap of the area of the curves between the class “same” and the other classes can indicate the amount of confusion for this distance score.

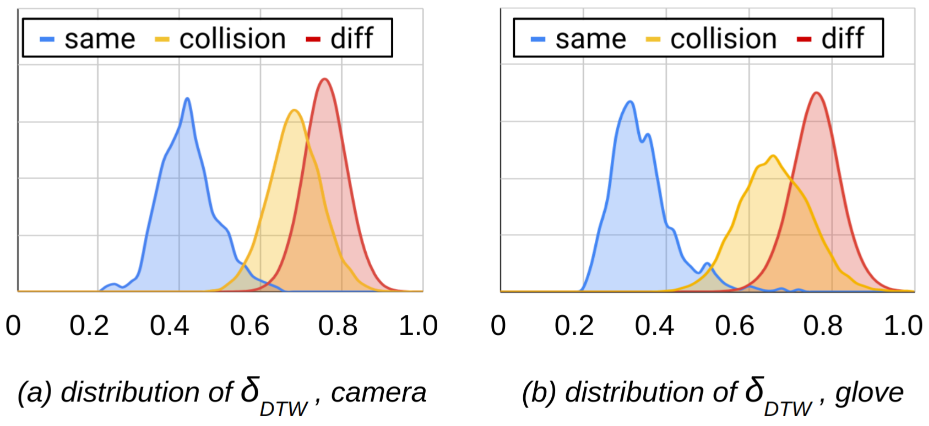


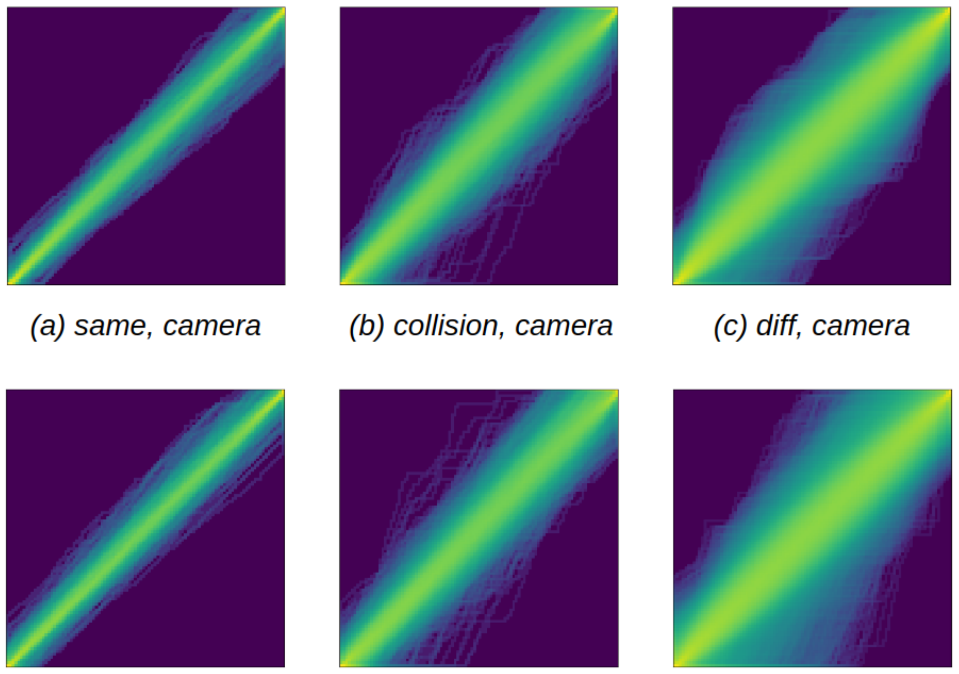
Figure 4.5: Distribution of DTW Distance Scores With Both Types of Devices

Additionally, we show the distribution of the warping paths in Figure 4.6. This is obtained by overlaying warping paths of 1,000 signal and template pairs using both types of devices, where each pair is normalized in time so that warping paths from different accounts can be compared. Hence, we can consider this figure as a “stacking” of all warping paths visualized in Figure 4.4 (left) for 1,000 accounts. This figure indicates signals of different classes are “warped” in different intensity.

We define another distance score called alignment cost (denoted as δ_{AC}) to represent this difference in warping intensity. It is the absolute difference between a warping path and the diagonal line as shown in Fig 4.4 (left), averaged by the template length. Formally, assuming the segment between i'_s and i'_e of R is aligned to the ith sample of T, the alignment cost is computed as follows.

$$\delta_{AC} = \sum_{i}^{l} i - \frac{i'_s + i'_e}{2}$$

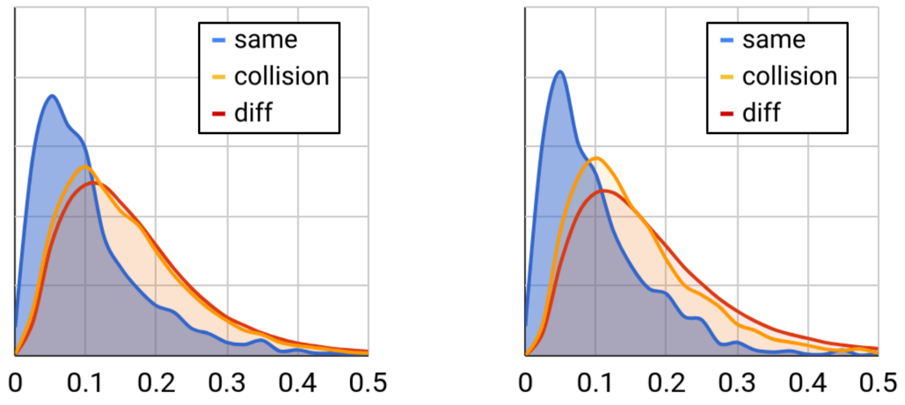
Here i'_s and i'_e are the two indices shown as the “start” and “end” for each sample of the template in Figure 4.3. This alignment cost is a feature with relatively weak


(a) same, camera (b) collision, camera (c) diff, camera

(d) same, glove (e) collision, glove (f) diff, glove

Figure 4.6: Distribution of Warping Path for Different Classes Using Both Devices

discriminative capability, but it can be used in the multifactor fusion methods. The distribution of the alignment cost scores for all accounts and all login requests in three authentication class labels are shown in Figure 4.7.



(a) distribution of δ_{AC}, camera (b) distribution of δ_{AC}, glove

Figure 4.7: Distribution of Alignment Cost Scores With Both Types of Devices

4.3.2 Sample Variation Analysis and TTV Matching

In the DTW distance score calculation, each sample is treated equally, which is reasonable for alignment but not necessarily for matching and authentication decision making. We propose the following Threshold-Then-Vote (TTV) algorithm based on the DTW matching with improvements.

Template Generation: We calculate the template T in the same way as the DTW matching algorithm. Additionally, we calculate the template uncertainty C as follows,

$$C_{ij} = var(S_{ij}^{(1)}, S_{ij}^{(2)}, ..., S_{ij}^{(m)}),$$

where the $var()$ function calculates the variance. Both T and C are stored in the account database.

Matching: Given an account with T, C, and a request signal R, first we preprocess R and align it to T. Then, we calculate element-wise distance D_{ij} in the same way as the DTW matching. After that, we multiply D_{ij} by two scaling factors P_{ij} and Q_{ij} derived from the template value T and variance C (explained later). Finally, we calculate the signal level distance

$$\delta_{TTV} = \frac{1}{l}\sum_{i}^{l} round[\frac{1}{d}\sum_{j}^{d} TTV(D_{ij} * P_{ij} * Q_{ij})].$$

The $round()$ function means rounding to the nearest integer (either 0 or 1 in our case), which is essentially a majority vote. $TTV(x)$ is a step function defined as follows:

$$TTV(x) = \begin{cases} 0, & \text{if } x \le th_1 \\ 0.5, & \text{if } th_1 < x \le th_2 \\ 1, & \text{if } x > th_2 \end{cases} \tag{4.1}$$

The matching procedure is summarized in Algorithm 1.

Algorithm 1: TTV_match()

```
input      : $R, T, C$
parameter: $p, q, th_1, th_2$
output     : dist
$R \leftarrow$ preprocess($R$).
$S \leftarrow$ align($T$, $R$).
$D \leftarrow$ element-wise-abs($T$ - $S$).
$P \leftarrow$ element-wise-inverse($\mathbf{1} + p \times T$)
$Q \leftarrow$ element-wise-inverse($\mathbf{1} + q \times C$)
$D \leftarrow$ element-wise-multiply($D$, $P$).
$D \leftarrow$ element-wise-multiply($D$, $Q$).
for $i$ = 1 to $l$ do
    for $j$ = 1 to $d$ do
        if $D_{ij} < th_1$ then
            $D_{ij} \leftarrow 0$
        else if $D_{ij} > th_2$ then
            $D_{ij} \leftarrow 1$
        else
            $D_{ij} \leftarrow 0.5$
        end
    end
end
dist $\leftarrow \frac{1}{l}\sum_i^l round[\frac{1}{d}\sum_j^d TTV(D_{ij})]$
```

Similar to δ_{DTW}, δ_{TTV} can be used directly for decision making or as a feature in fusing multiple factors.

This matching algorithm maps an element-wise distance to an element-wise decision as “match” (*i.e.,* 0), “non-match” (*i.e.,* 1), or “unsure” (*i.e.,* 0.5). Then the element-wise decisions are aggregated, and the portion of non-match votes is the distance score. Here th_1 and th_2 are tunable parameters determined empirically (in our case, $th_1 = 0.3$, $th_2 = 0.4$). When S and T are obtained from the same user writing the same passcode, because of slight handwriting behavior change or misalignment, such local non-matches exist but their votes are limited. However, when S and T are obtained from different users or from writing different contents, they are different in many places along multiple axes and the non-matches can dominate the votes.

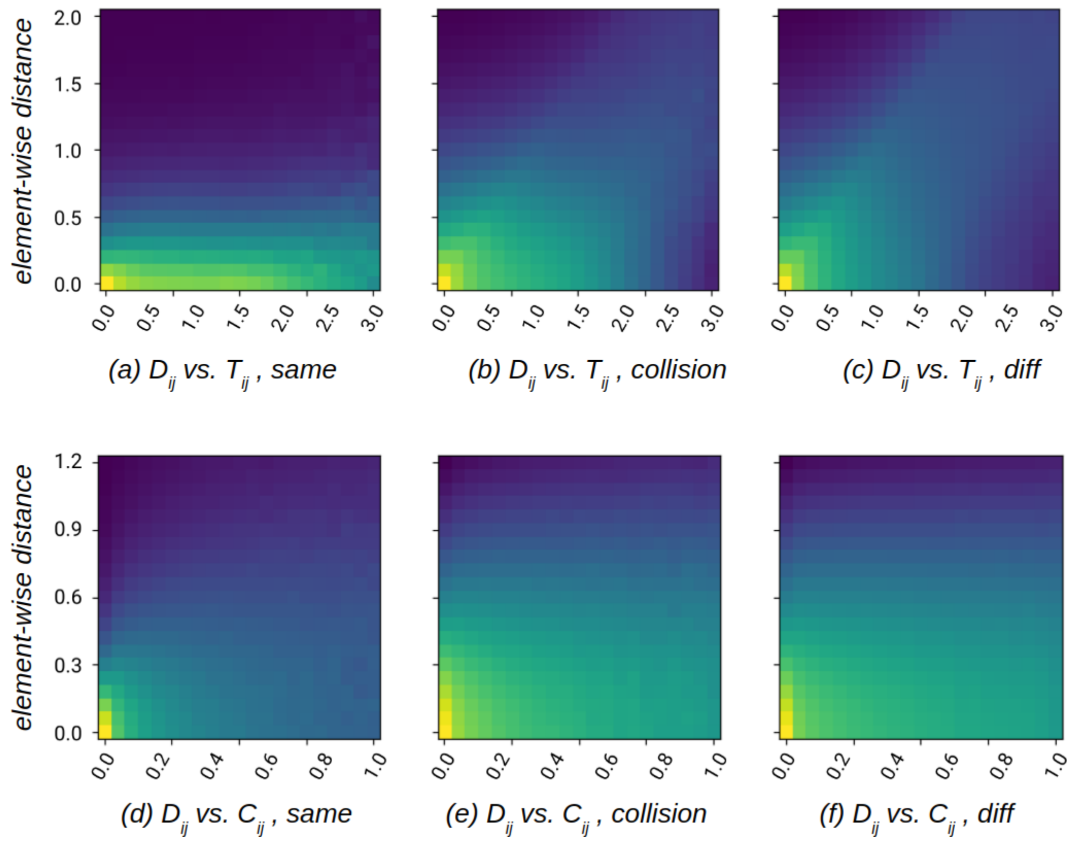


Figure 4.8: Joint Distribution of Element-wise Distance and Template Property, with the Camera Device

The design philosophy of the two two scaling factors P and Q are explained as follows. Rather than varying the thresholds th_1 and th_2, the scaling of element-wise distance are introduced to apply less importance (*i.e.,* more tolerance) to some sam-

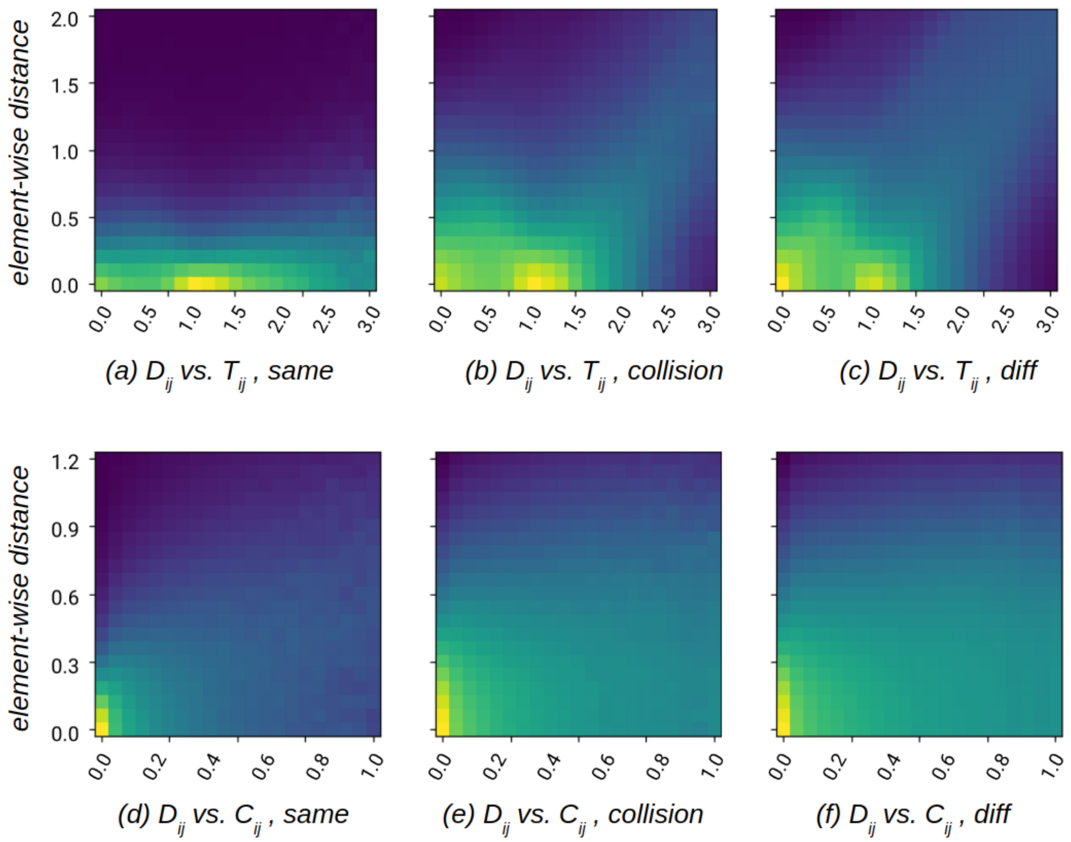
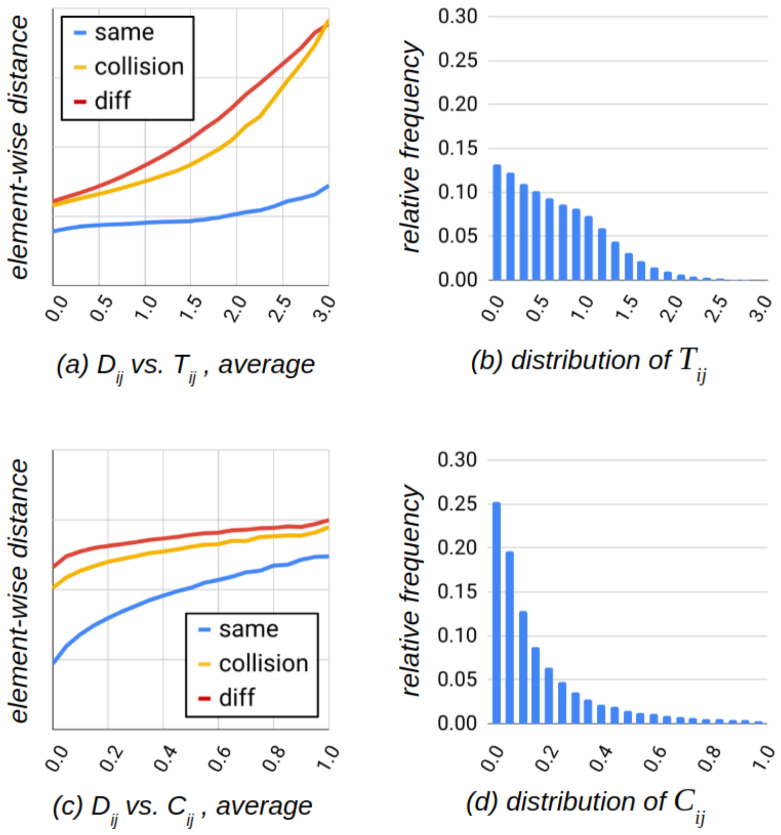


Figure 4.9: Joint Distribution of Element-wise Distance and Template Property, with the Camera Device, with the Glove Device.

ples. The assumption is that the element-wise distance varies differently in different places and along different sensor axes, and D_{ij} is large when T_{ij} and C_{ij} are large. Hence, larger tolerance should be placed at these samples. Here T_{ij} is each template sample value and C_{ij} is the template variance at each sample. To confirm this, we first show the joint distributions of D_{ij} and T_{ij}, as well as the joint distribution of D_{ij} and C_{ij}, in Figure 4.8 and Figure 4.9. These joint distributions are conditioned with the distribution of T_{ij} and C_{ij} themselves. Then, we show the collective trend between D_{ij} and T_{ij}, as well as the trend between D_{ij} and C_{ij} in Figure 4.10 and Figure 4.11. We can approximately fit a line to capture these trends.



(a) D_{ij} vs. T_{ij}, average (b) distribution of T_{ij}

(c) D_{ij} vs. C_{ij}, average (d) distribution of C_{ij}

Figure 4.10: Relation between Element-wise Distance and Template Property, with the Camera Device.

Using these trends, P_{ij} and Q_{ij} are calculated as follows.

$$P_{ij} = 1/(1 + p * T_{ij}), Q_{ij} = 1/(1 + q * C_{ij}).$$

p and q are tunable parameters determined empirically (in our experiments, $p = 0.5$, $q = 1.0$). Basically, p and q are chosen such that the scaled D_{ij} is roughly separable between the same and diff classes using a single threshold, *i.e.*, those lines in Figure 4.10 (left) and Figure 4.11 (left) are roughly flat after D_{ij} is scaled by the two factors. To further accommodate the fuzziness introduced by the writing behavior, two thresholds are used in the TTV algorithm instead of just one threshold.

Now we provide more analysis on the element-wise distance to explain the rationale of the two thresholds. For convenience, $D_{ij} * P_{ij} * Q_{ij}$ is denoted as D'_{ij}.

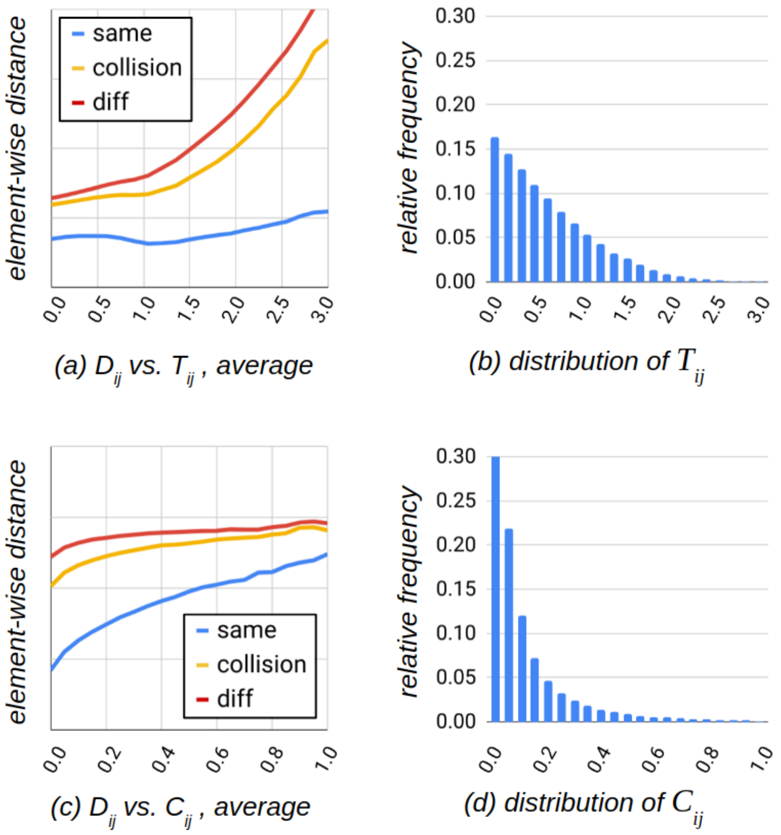
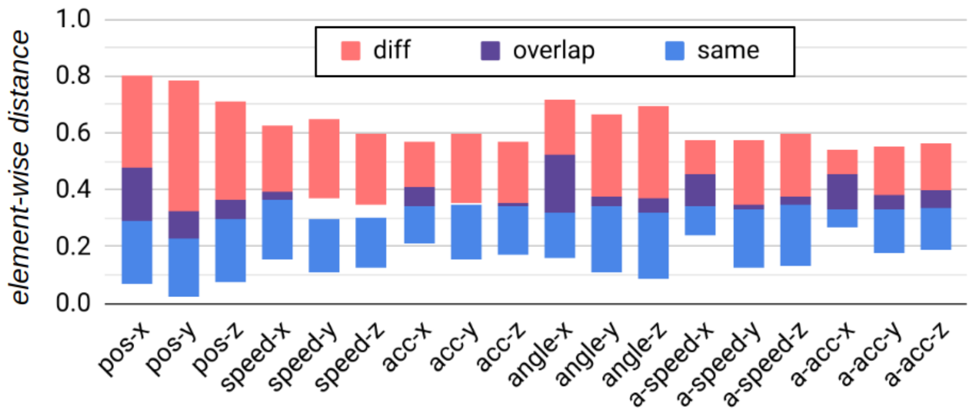


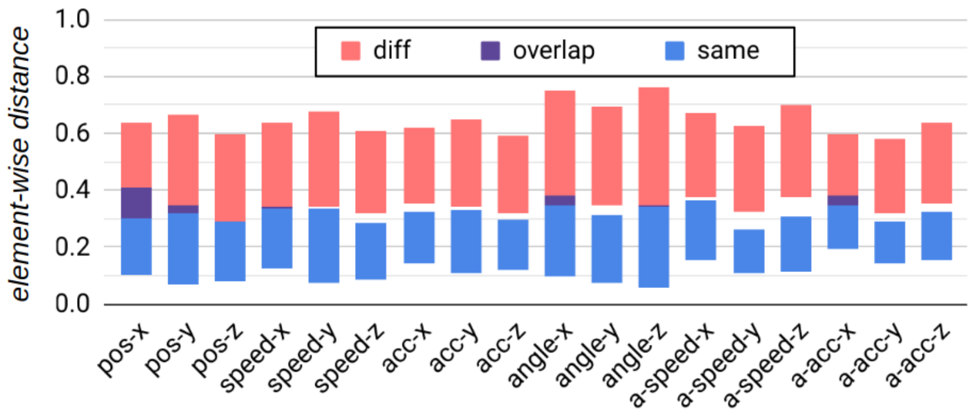
Figure 4.11: Relation between Element-wise Distance and Template Property, with the Glove Device.

First, in Figure 4.12 and Figure 4.13, we show the $\pm 2\sigma$ ranges of D'_{ij} respect to the sensor axes averaged along the time, which are approximations of $p(\frac{1}{l}\sum_i^l D'_{ij}|c)$, *i.e.*, distribution of aggregated D'_{ij} respect to the sensor axes. The 18 sensor axes follow the order defined in chapter 3.4. Because the original orientation representation using unit quaternions cannot be mapped to the x-y-z axes directly, we convert the orientation into three Euler angles roll, pitch, yaw, and show them as angle-x, angle-y, and angle-z in the figures. Also, angular speed and angular acceleration are abbreviated as a-speed and a-acc respectively.

These figures show the variation of the signal in each sensor axis, and larger overlap indicates more difficulty in distinguishing different classes based on that axis. Since the writing behavior in the air is similar to writing on an invisible wall, and



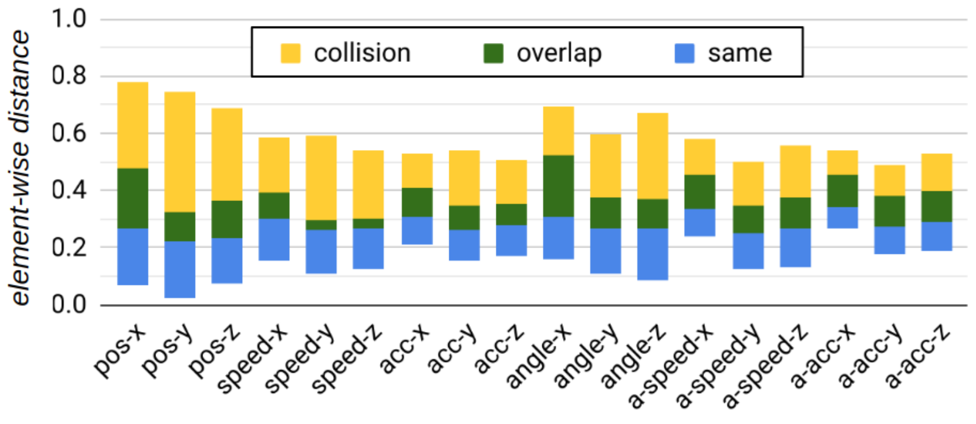
(a) D'_{ij} in different sensor axes, same and diff classes, camera



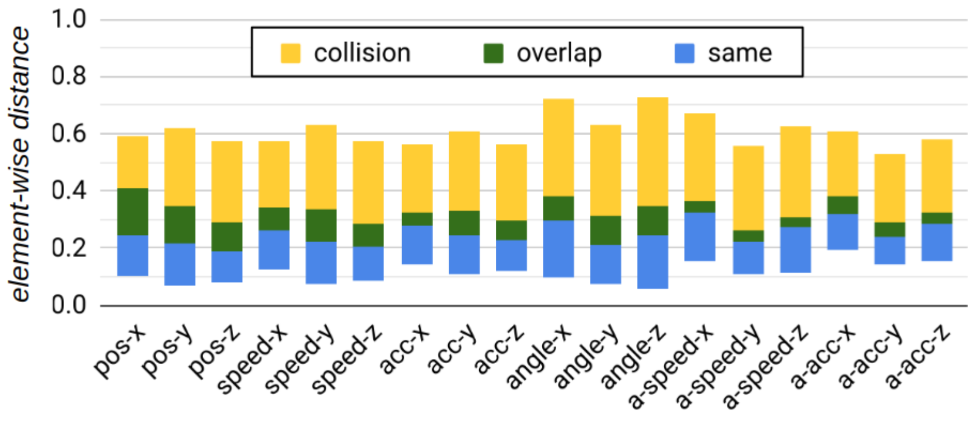
(b) D'_{ij} in different sensor axes, same and diff classes, glove

Figure 4.12: Distribution of Aggregated Scaled Element-wise Distance in Different Sensor Axes, without Considering Collision

in the posture normalization preprocessing step, the x-axis is defined as the averaged pointing direction (*i.e.* against the wall), the uncertainty in that direction is larger. Also, the position axes show larger variation range, which means users are generally better at keeping the same pattern in speed and acceleration than maintaining the same trajectory. We believe that the speed and acceleration of hand movement are directly related to the force generated by the muscle using "muscle memory", while maintaining position and trajectory requires visual feedback and attention from the brain. Since handwriting is an acquired skill, the "muscle memory" is a behavioral



(a) D'_{ij} in different sensor axes, same and collision classes, camera



(b) D'_{ij} in different sensor axes, same and collision classes, glove

Figure 4.13: Distribution of Aggregated Scaled Element-wise Distance in Different Sensor Axes, with Considering Collision

biometric trait and relatively stable.

Second, in Figure 4.14, we show the distribution of D'_{ij} respect to the time averaged over all sensor axe, *i.e.*, an approximation of $p(\frac{1}{d}\sum_j^d D'_{ij}|c)$. For those signal and template generated by the same user writing the same content ($c = same$), the average of D'_{ij} is generally smaller than those generated by different users writing the same content ($c = collision$) or by writing different contents ($c = diff$). This is expected. There is an overlap in a region (shown as the dashed horizontal lines), which determines the two parameters th_1 and th_2. Generally, below th_1, it is desired that

$$p(\frac{1}{d}\sum_j^d D'_{ij}|c = \text{same}) > p(\frac{1}{d}\sum_j^d D'_{ij}|c = \text{collision}),$$

and above th_2, it is desired that

$$p(\frac{1}{d}\sum_j^d D'_{ij}|c = \text{diff}) > p(\frac{1}{d}\sum_j^d D'_{ij}|c = \text{collision}).$$

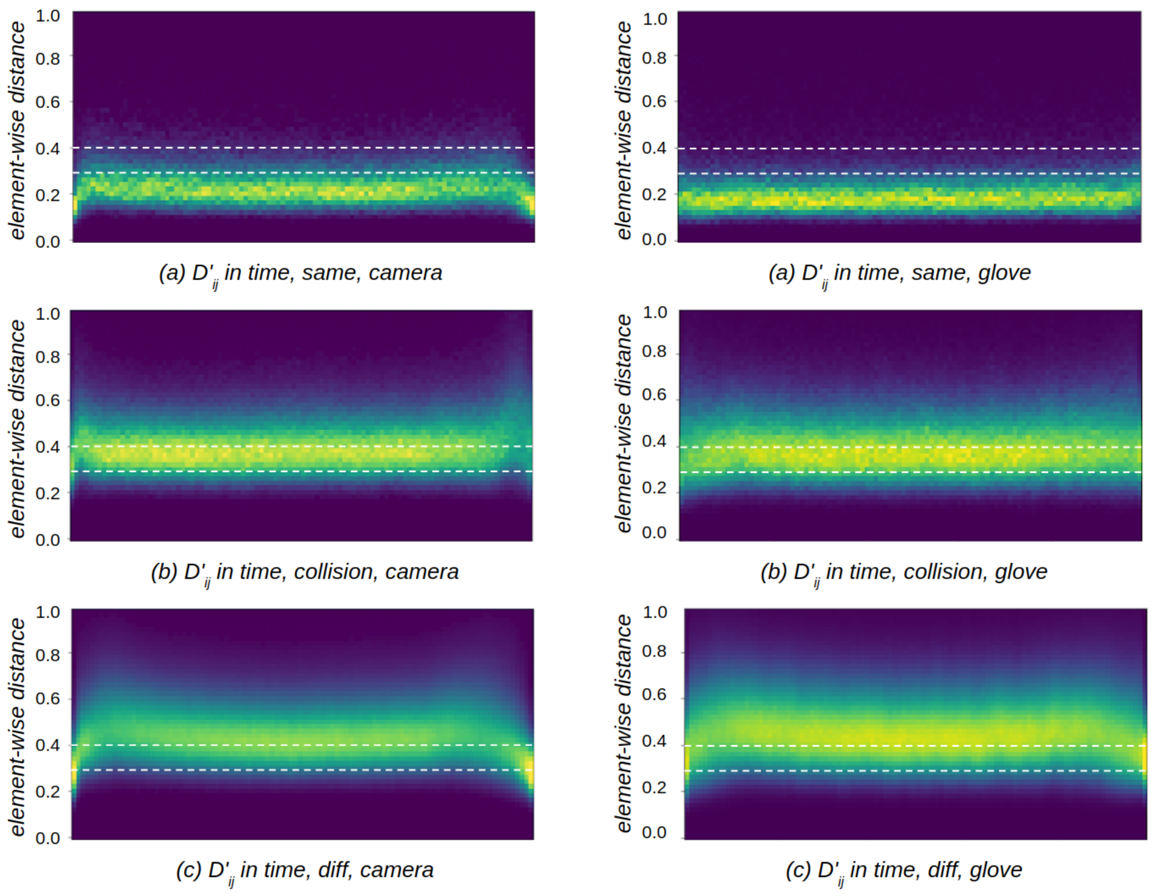
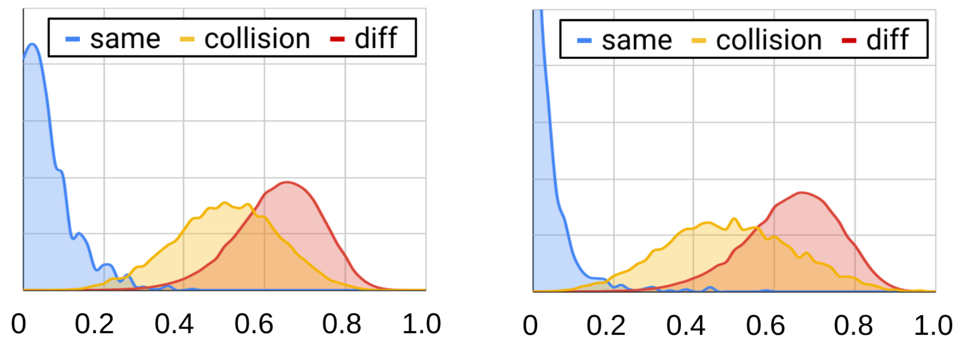


Figure 4.14: Distribution of Aggregated Scaled Element-wise Distance in Time

From these figures, it can be observed that samples in different sensor axes and at different time should be considered with different importance (*i.e.*, using different weight). Also, in Figure 4.14, it can be observed that the distribution of sample differences do not change along the time given a large amount of accounts with diverse passcode contents. However, the local non-matches of signals from different accounts varies in different ways. Hence, a per-account decision maker can perform better than a global decision maker. These observations explains the performance gain of the TTV method and the Feature Fusion method (detailed in section 4.7). Additionally, there are small anomalies on both ends, since users usually raise the hand to start the writing and put down the hand to terminate the writing. Hence, the performance can be improved slightly by trimming the start and the end again after alignment.

Third, we show the distribution of the TTV distance score δ_{TTV} over all accounts in Figure 4.15, *i.e.*, an approximation of the class-conditional probability $p(\delta_{TTV}|c)$. The histograms are normalized as probability and the overlap between the histograms represents the overall discriminative capability of the matching algorithm (smaller overlap is better).



(a) distribution of δ_{TTV}, camera *(b) distribution of δ_{TTV}, glove*

Figure 4.15: Distribution of TTV Distance Scores with Both Types of Devices

4.4 Statistical Feature Analysis

In this section, we focus on those statistical aspects of the signal which are lost in the alignment and normalization steps. Specifically, the following statistical features are studied.

1) The mean of each sensor axis $\mathbf{M} = (\mu_1, ..., \mu_d)$, where $\mu_j = mean(R_{1j}, ..., R_{lj})$.

2) The standard deviation of each sensor axis $\mathbf{\Sigma} = (\sigma_1, ..., \sigma_d)$, where $\sigma_j = std(R_{1j}, ..., R_{lj})$.

3) The correlation $\mathbf{P} = (\alpha_{xy}, \alpha_{yz}, \alpha_{xz}, \beta_{xy}, \beta_{yz}, \beta_{xz}, ...)$ between pairs of adjacent sensor axes, where α_{xy} is the correlation of the three axes of position, β_{xy} is the correlation of the three axes of speed, and so on.

4) Sum of amplitude of each axis $\mathbf{\Lambda} = (\lambda_1, ..., \lambda_d)$, where $\lambda_j = \sum_{i=1}^{l} |R_{ij}|$.

5) Portion of low frequency components of each axis $\mathbf{L} = (\xi_1, ..., \xi_d)$, where $\xi_j = mean(low\_pass(\text{FFT}(R)))$. Here FFT$(R)$ is the Fast Fourier transform coefficients and the $low\_pass()$ function zeros the coefficients higher than 3 Hz.

Collectively, the statistical feature vector of a signal R is the combination of them, defined as $f(R) = (\mathbf{M}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{\Lambda}, \mathbf{L})$, where $f(R)$ is the statistical feature extractor.

Template Generation: Given $\{R^{(1)}, R^{(2)}, ..., R^{(m)}\}$ as the m signals at registration for an account, first run the prepossessing step 1 to 5 for each signal, extract statistical features, and then calculate the element-wise mean $\boldsymbol{\mu}_{SF}$ and standard deviation $\boldsymbol{\sigma}_{SF}$ as templates.

$$\boldsymbol{\mu}_{SF} = mean(f(R^{(1)}), f(R^{(2)}), ..., f(R^{(m)}))$$

$$\boldsymbol{\sigma}_{SF} = std(f(R^{(1)}), f(R^{(2)}), ..., f(R^{(m)}))$$

Matching: Given an account with template $\boldsymbol{\mu}_{SF}$ and $\boldsymbol{\sigma}_{SF}$, and an authentication request signal R, run the same preprocessing steps as those in the template generation, then calculate the following aggregated statistical difference score

$$\delta_{SD} = mean(abs(f(R) - \boldsymbol{\mu}_{SF})/\boldsymbol{\sigma}_{SF})$$

where the subtraction and division are all element-wise, the $mean()$ function averages all elements of its vector input. These statistical differences are based on the Mahalanobis distance.

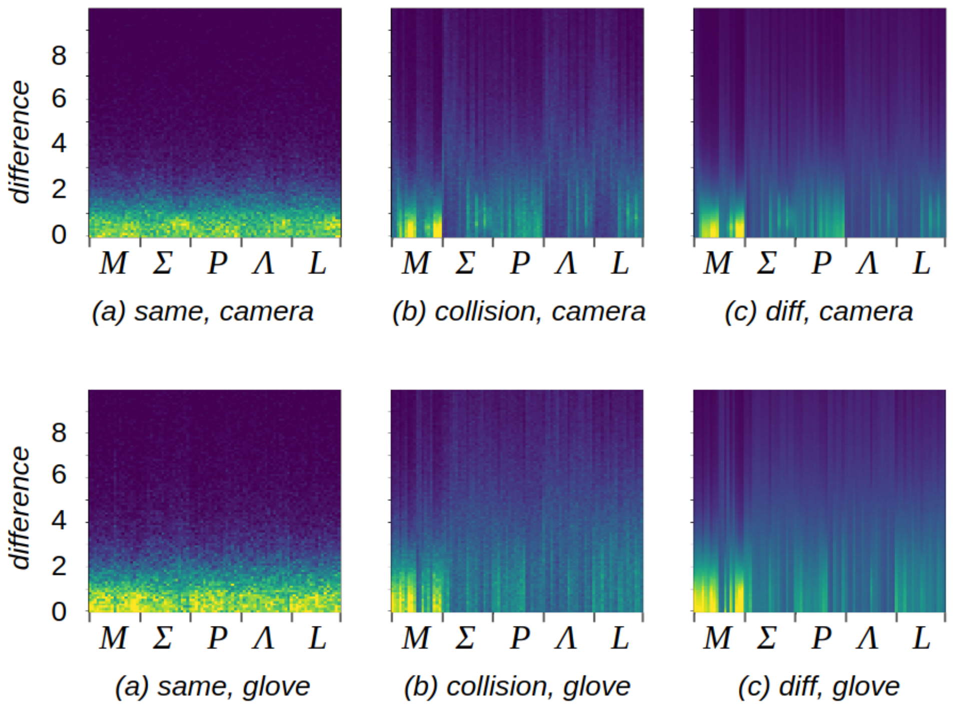


Figure 4.16: Distribution of Statistical Difference for Different Classes with Both Types of Devices

The distribution of each component of the statistical difference vector $abs(f(R) - \boldsymbol{\mu}_{SF})/\boldsymbol{\sigma}_{SF}$ is shown in Figure 4.16.

The correlation of each component of $f(R)$ is shown in Figure 4.16, which indicates positive correlation of signal variance, amplitude, and low frequency components. Generally, if the user writes intensively, all these three features will have a larger magnitude.

We incorporate an additional statistical features, which is the difference of lengths between a signal R and a template T, denoted as δ_{LD}. It is calculated as

$$\delta_{LD} = |l_R - l_T|/l_T$$

Here l_R and l_T are the lengths of the signal R and template T respectively.

In a word, these statistical features are generally weak factors. Combining them together the δ_{SD} may show a certain level of discriminative capability, as shown in

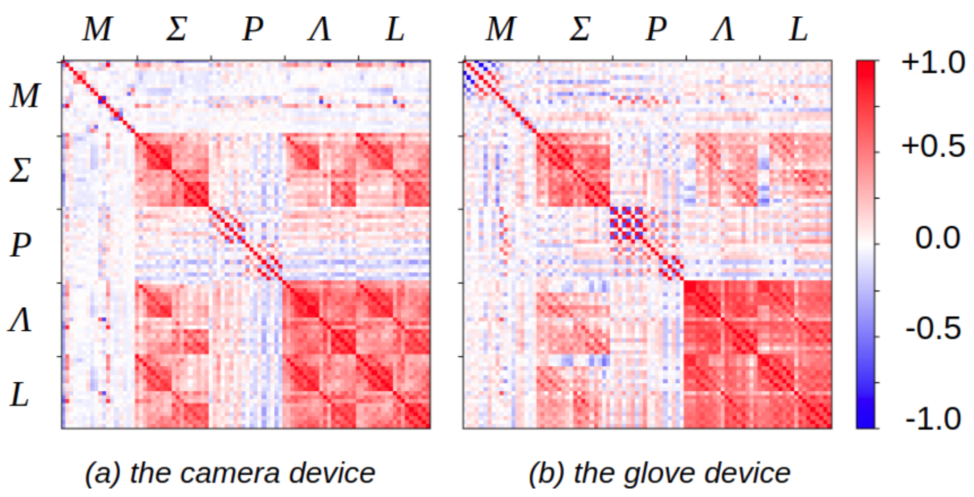


Figure 4.17: Correlation of the Statistical Features of All Axes

the distribution of statistical difference scores in Figure 4.18. However, we argue that removing them in the preprocessing step does not incur much loss of information for the authentication task. This is also confirmed in the comparison of empirical evaluation results in section 4.7. The means of most sensor axes, such as the speed and acceleration, are especially weak features because many of them tend to be around zero anyway.

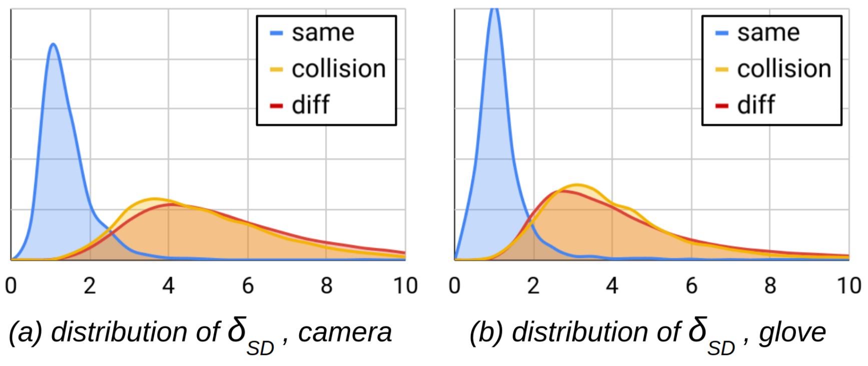


Figure 4.18: Distribution of Statistical Difference Scores With Both Types of Devices

4.5 Hand Geometry Feature Analysis

In our framework, the hand geometry feature **h** is a vector of 22 components including the bone length vector and the hand width vector measured using the onboard software of the camera device. The bone length vector contains the lengths of metacarpals (M), proximal phalanges (P), intermediate phalanges (I), and distal phalanges (D) of the five fingers on a hand, in total 19 components (note that the thumb does not have an intermediate phalanges segment). The hand width vector includes 3 components, which are the distances between the far ends of metacarpals of two adjacent fingers except the thumb. The definition of each component of the hand geometry feature is illustrated in Figure 4.19, and the vector **h** is a concatenation of these components for each finger and the three hand width segments. For the glove device, this feature is not available. The hand geometry feature is utilized as follows.

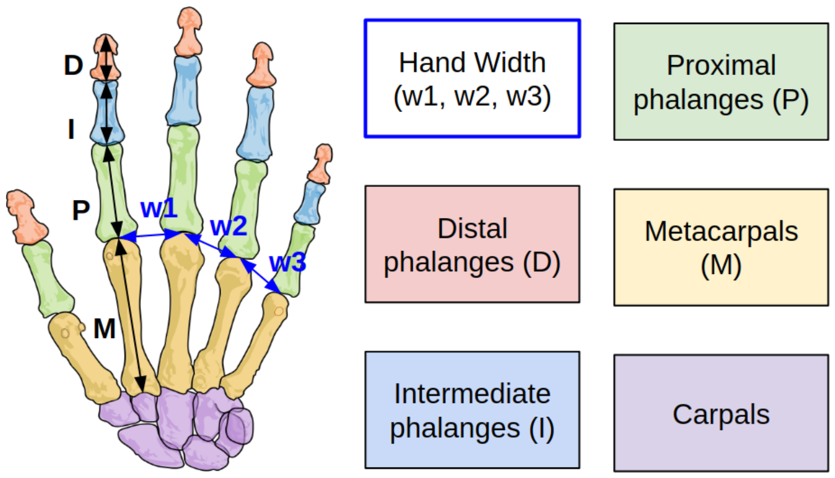


Figure 4.19: Definition of Hand Geometry Features

Template Generation: Given m measures of the user's hand $\{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, ..., \mathbf{h}^{(m)}\}$, construct the template

$$\mathbf{h}_T = mean(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, ..., \mathbf{h}^{(m)}).$$

Matching: Given an account with template $\mathbf{h}_T$, and an authentication request with a measurement $\mathbf{h}_R$, calculate the hand geometry difference (HGD)

$$\delta_{HGD} = mean(abs(\mathbf{h}_R - \mathbf{h}_T)/\mathbf{h}_T)$$

where both the subtraction and division are element-wise.

If the authentication request is generated by the account owner using the same hand, the hand geometry feature extracted from the request should not differ from the stored hand geometry template significantly. We measured the difference of each individual hand geometry component of the same user as well as different users, and the results for each component of $abs(\mathbf{h}_R - \mathbf{h}_T)$ average over all pairs of hands in our dataset is shown in Figure 4.20. It can be seen that a longer bone on a finger causes larger differences among users, and hence, we use the ratio instead of the absolute difference. Also, the hand motion tracking device needs to obtain the hand geometry measurement from the user's hand image in millimeter-level precision, which is challenging. Thus, even the same hand from the same user is measured in multiple times, there are slight differences due to the measurement uncertainty. However, on average this measurement uncertainty is less than the difference between different hands from different users.

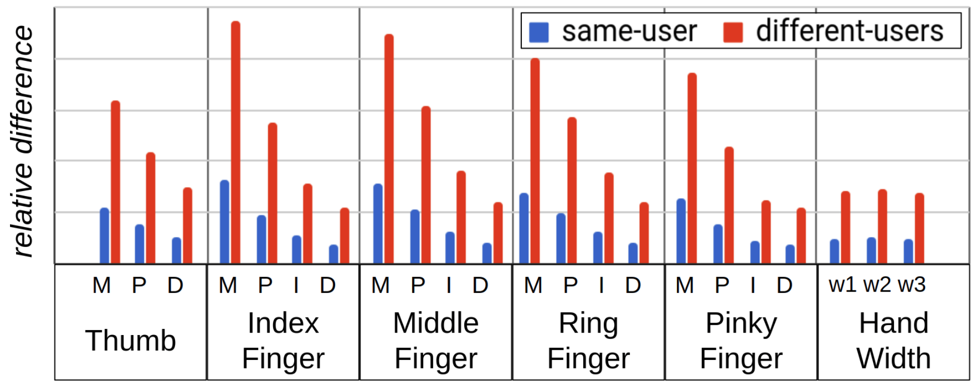


Figure 4.20: Average of Length Differences of Hand Geometry Feature Components

Another important fact is that all these 22 components have strong correlations. Since the five fingers on the same hand grow together, we suspect that the onboard software of the camera device uses an internal model for estimating the hand shape, which scales linearly in hand size. This can be verified by running Principle Component Analysis (PCA) on the hand geometry data. Using our data, the dominant principle component has an eigenvalue more than a thousand times larger than the others. As a result, we average the ratio difference of the feature components to improve the robustness and reduce the dimensionality of this feature.

Obviously, if two hands are different in size and shape, this hand geometry feature will show the difference. However, it is highly possible that two different hands have similar shapes and there is no way to determine whether they are from the same user merely based on the geometry. Meanwhile, the same user may create multiple accounts with the same hand. In a word, hand geometry is a weak factor with very limited discriminative capability. Even if the glove device cannot obtain the hand geometry features, it can still achieve comparable authentication performance as the camera device using hand geometry features, as shown in section 4.7.

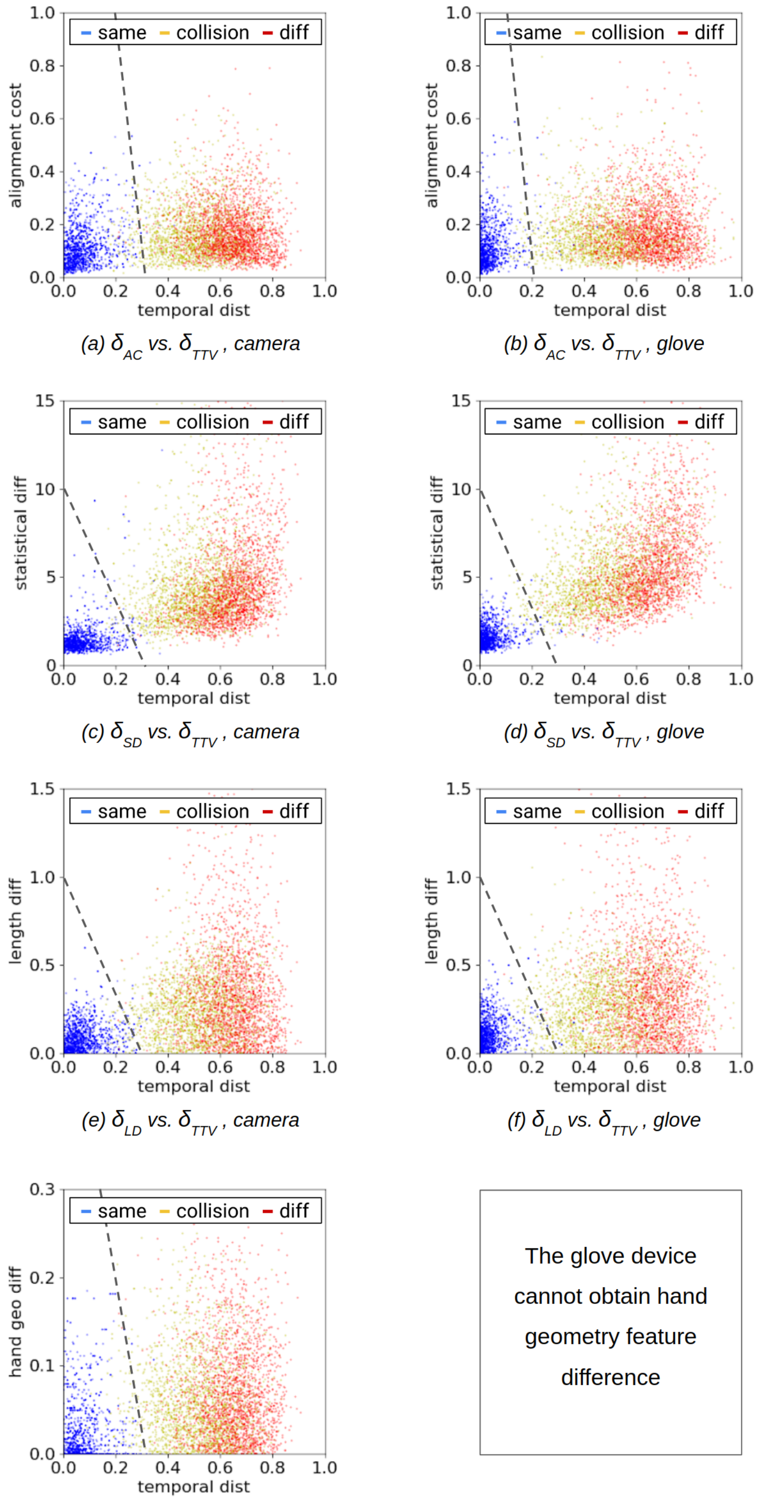
4.6 Fusion of Multiple Features

In this section we propose two methods of fusing multiple factors at score level (section 6.1) and feature level (section 6.2).

4.6.1 Score Fusion

The Score Fusion (S-Fusion) scheme uses a weighted average of the matching scores on each type of feature as follows.

Template Generation: Given the registration signals, generate templates for the temporal, statistical, and hand geometry features.



The glove device cannot obtain hand geometry feature difference

(a) δ_{AC} vs. δ_{TTV}, camera
(b) δ_{AC} vs. δ_{TTV}, glove
(c) δ_{SD} vs. δ_{TTV}, camera
(d) δ_{SD} vs. δ_{TTV}, glove
(e) δ_{LD} vs. δ_{TTV}, camera
(f) δ_{LD} vs. δ_{TTV}, glove
(g) δ_{HGD} vs. δ_{TTV}, camera
(h) δ_{HGD} vs. δ_{TTV}, glove

Figure 4.21: Distribution of the Scores From Various Features

Matching: Run the matching algorithm of each feature and calculate the score:

$$\delta_{other} = w_1\delta_{AC} + w_2\delta_{SD} + w_3\delta_{LD} + w_4\delta_{HGD},$$

$$\delta_{SFusion} = \delta_{TTV} + \delta_{other},$$

where δ_{TTV} is the matching score using the TTV algorithm (*i.e.*, the TTV distance score), δ_{AC} is the alignment cost score, δ_{SD} is the statistical difference score, δ_{HGD} is the hand geometry difference score, w_1 to w_4 are tunable parameters. Here, δ_{TTV} is considered as the primary factor with strong discriminative capability, and the others are supplemental factors.

These weights w_1 to w_4 are determined empirically. In our case, $w_1 = 0.1, w_2 = 0.03, w_3 = 0.3, w_4 = 0.4$, for both types of devices. Since the glove device is unable to measure hand geometry, w_4 is not used for that device. Specifically, we first plot the scores of supplemental factors respect to the scores of the primary factor using 1,000 signals in each class, and then we fit a linear decision boundary manually, as shown in Figure 4.21. The weights w_1 to w_4 are determined by the slope of these lines. For example, in Figure 4.21 (a), the linear decision boundary connects $\delta_{AC} = 3$ and $\delta_{TTV} = 0.3$, and hence, w_1 is set to $\delta_{TTV}/\delta_{AC} = 0.1$. This is designed mainly for simplicity and practicality with a small pilot dataset, which is not necessarily optimal. Also, it is assumed that these supplemental factors are not correlated, which may not be true. More sophisticated score fusion schemes can also be employed in the future. However, generally more sophisticated score fusion schemes use more complex models and require more data to estimate the probability distribution of feature vectors in different classes, which may cause over-fitting on a small pilot dataset.

4.6.2 *Feature Fusion*

If we inspect each individual element-wise distance D_{ij}, it is essentially a type of weak feature. Inspired by the idea of ensemble of weak features using weighted average, we design a Feature Fusion (F-Fusion) scheme as follows.

Template Generation: Same as Score Fusion.

Matching: Align the signal R to T and calculate the score:

$$\delta_{other} = w_1\delta_{AC} + w_2\delta_{SD} + w_3\delta_{LD} + w_4\delta_{HGD},$$

$$\delta_{T-Fusion} = b + \sum_{i=1}^{l}\sum_{j=1}^{d} w_{ij}D_{ij},$$

$$\delta_{F-Fusion} = \delta_{TFusion} + \delta_{other},$$

In this scheme, each feature has its own weight w_{ij}, and since they vary differently, the w_{ij} should be calculated based on the data for each account. b is a bias to keep the score roughly in the range between 0 and 1. DTW matching is a special case of T-Fusion by setting all weights to the same value, and TTV is setting the weights systematically based on a few hyperparameters with additional nonlinear cutoff. Determining the weights w_{ij} is essentially fitting a decision plane in the feature space, and this can be done for each account independently. In our case, these weights w_{ij} as well as w_1 to w_4 are learned from the data at registration using a soft margin Support Vector Machine (SVM) in a per account manner. Specifically, at registration, for each account we use the D_{ij} calculated by the five signals generated by the account owner as positive samples, and we use the D_{ij} calculated by 1,000 signals randomly selected from other accounts as negative samples to train an SVM. One limitation of this method is that at registration data from all accounts must be accessed. This is possible for an online applications with a lot of accounts but infeasible for device unlock scenarios with only one or a few users. However, as our analysis in section 4.7

shows, it is still possible to achieve relatively useful performance with a set of signals not related to other accounts as negative training samples.

4.7 Experimental Evaluation

We adopt the following experimental evaluation protocol.

1) We create each account and the templates using the five registration signals in the "ID-passcode" dataset. We also train the SVM if the feature fusion method is used. We mixed the ID strings and the passcode strings to create 360 individual accounts to slightly increase the dataset size, *i.e.*, each user has two accounts in our experiments, where one account uses the IAHW of the ID string in the "ID-passcode" dataset for passcode purpose, and the other account uses the IAHW of the passcode string "ID-passcode" dataset for passcode purpose. This can improve the resolution of the evaluation metrics, and it also helps us to reveal a few hard cases analyzed in section 4.8. A global threshold is also configured to determine the boundary of acceptance and rejection.

2) For each account and for each of the fives testing signals in "ID-passcode" dataset, we use it as a login request with class label $c = \textit{same}$ to that account. Then, we use it as a login request with class label $c = \textit{diff}$ to all other accounts. The results generated in this step are denoted as the case without collision.

3) For each account and for each of the five testing signals in "ID-passcode", we use it as a login request with class label $c = \textit{save}$ to that account. Then, for each signal generated by impostors in "ID-passcode-collision", we use it as a login request with class label $c = \textit{collision}$ to the target account. The results generated in this step are denoted as the case with collision.

We use the evaluation metrics defined in chapter 3.8.1. Specifically, if a request signal with class label $c = \textit{same}$ is rejected, it is a False Reject (FR); if a request signal with class label $c = \textit{diff}$ or $c = \textit{collision}$ is accepted, it is a False Accept (FA) or a Successful Collision (SC), which is a special case of false accept. The False Reject Rate (FRR) and False Accept Rate (FAR) are defined as follows.

$$FRR = \frac{1}{kn}\sum_{i=1}^{n}\#\{FR_i\},$$

$$FAR = \frac{1}{kn(n-1)}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\#\{FA_{ij}\}, \text{ without collision,}$$

$$FAR = \frac{1}{kmn}\sum_{i=1}^{n}\sum_{j=1}^{m}\#\{SC_{ij}\}, \text{ with collision,}$$

where n is the total number of accounts ($n = 360$); k is the number of testing signals generated by each user or the number of collision signals by each impostor, *i.e.*, both are five in our experiments; m is the number of impostors ($m = 10$). $\#\{FR_i\}$ is the total number of false rejects of the ith account, $\#\{FA_{ij}\}$ is the total number of false accepts for ith account using the signals of jth account as the authentication requests, and $\#\{SC_{ij}\}$ is the total number of successful spoof for the ith account by the jth impostor.

The results are shown in Table 4.1 and 4.2. The Receiver Operating Characteristic (ROC) curves are shown in Figure 4.22. We also use the Equal Error Rate (EER) to evaluate the authentication methods, which is the rate where FRR is equal to FAR. Here FAR1K and FAR10K indicate the FRR when FAR is 10^{-3} and 10^{-4}, respectively. ZeroFAR is the smallest FRR when FAR is zero and ZeroFRR is the smallest FAR when FRR is zero. In the table, the row DTW(2) and TTV(2) shows the results using only the first two signals at registration for template construction. T-Fusion(A) and T-Fusion(E) are explained later in this section.

In this section, we do not evaluate the authentication methods with the data

Table 4.1: Authentication Performance Results with the Camera Device

method	w/o collision (in %)					with collision (in %)		
	EER	FAR 1K	FAR 10K	Zero FAR	Zero FRR	EER	FAR 1K	Zero FAR
DTW(2)	1.24	5.75	17.69	71.40	28.76	5.10	27.6	37.2
TTV(2)	1.00	4.81	12.18	44.57	16.49	3.64	19.9	37.2
DTW	0.81	2.39	7.56	56.36	25.00	3.08	13.3	29.5
TTV	0.70	2.38	7.99	23.64	15.54	2.16	15.2	34.8
S-Fusion	0.50	1.38	4.95	37.75	42.38	**1.83**	**7.4**	**15.4**
T-Fusion	0.22	0.32	0.78	12.33	7.35	2.61	10.9	17.0
F-Fusion	0.26	0.31	0.85	12.00	7.30	2.61	10.7	16.6
T-Fusion(A)	0.21	0.23	**0.57**	**10.62**	**3.23**	2.74	10.6	23.2
T-Fusion(E)	**0.10**	**0.12**	0.78	10.70	4.72	2.40	8.91	18.3

Table 4.2: Authentication Performance Results with the Glove Device

method	w/o collision (in %)					with collision (in %)		
	EER	FAR 1K	FAR 10K	Zero FAR	Zero FRR	EER	FAR 1K	Zero FAR
DTW(2)	1.24	3.80	10.52	49.46	98.18	5.04	24.0	38.9
TTV(2)	1.12	2.97	7.30	48.84	50.97	4.17	23.0	38.9
DTW	0.75	1.33	5.26	43.80	8.56	2.95	12.0	19.6
TTV	0.68	0.85	2.86	35.50	11.23	2.39	13.6	23.6
S-Fusion	0.39	0.93	2.06	36.12	16.21	1.98	15.6	30.9
T-Fusion	**0.16**	**0.16**	**0.39**	2.95	6.57	1.51	4.7	6.2
F-Fusion	**0.16**	**0.16**	**0.39**	3.10	4.25	1.51	4.4	5.6
T-Fusion(A)	**0.16**	**0.16**	0.42	2.79	**0.50**	1.86	6.5	12.2
T-Fusion(E)	**0.16**	**0.16**	**0.39**	**2.02**	2.29	**1.38**	**3.5**	**5.1**

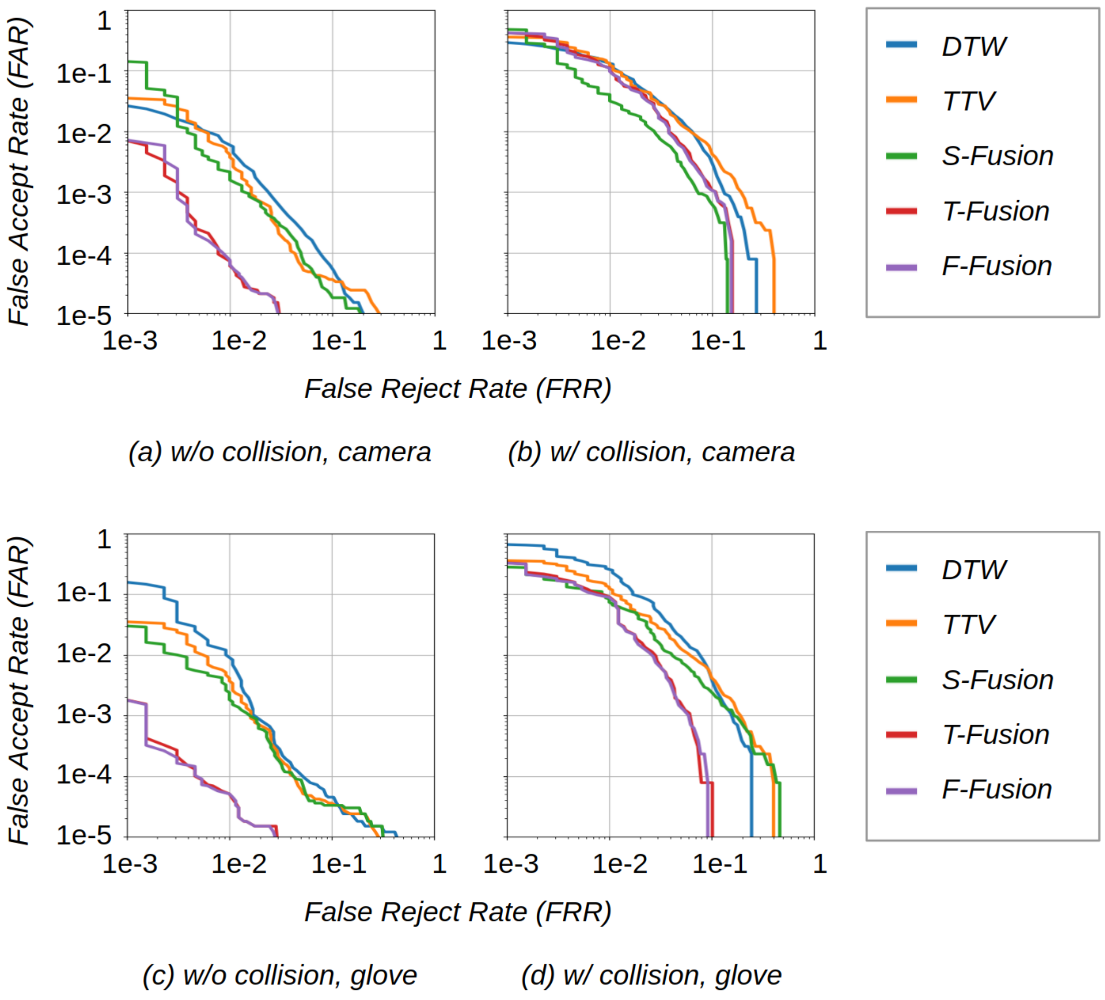


Figure 4.22: Receiver Operating Characteristic (ROC)

“ID-passcode-spoofing” and ”ID-passcode-persistence”, mainly because they are only available for a portion of the accounts. However, in chapter 6, we show the results using these two datasets.

These matching algorithms have a gradual improvement in performance with an increase in the number of parameters. The DTW algorithm is the simplest without any need of parameter tuning. The TTV algorithm requires to determine four parameters (p, q, th_1, th_2), and the S-Fusion algorithm requires another four (w_1 to w_4). From a deployment point of view, they can be set up with a small pilot dataset or using rule-of-the-thumb values provided in this section. They can also work with only one account in the account database (*e.g.*, for device unlock). In T-Fusion and F-Fusion, the numbers of parameters are roughly the same as an additional template signal, and they are learned from the data. The reason of the performance improvements for the T-Fusion method and the F-Fusion method is partially from the added weight parameters that can tolerate variation at different places of the signals, and partially from the added bias parameters which essentially makes the decision threshold adaptable for each account.

The limitation of all evaluated methods is essentially the quality of the template, *i.e.*, it is difficult to use one template generated from limited number of signals at registration to capture the whole picture of the user's handwriting behavior variation. This can be seen by the first two rows of Table 4.1 and 4.2 with the results for DTW(2) and TTV(2). We also show the authentication performance change in terms of EER respect to the number of signals for constructing the template. This results can be used to determine empirically how many repetitions are needed at registration. In general, more signals at registration lead to a better template, *i.e.*, more signals can capture the behavior variation better. However, the benefit is marginal because all registration signals are obtain at roughly the same time and the amount of variation is limited. This can also be confirmed by the results using the "ID-passcode-persistence" data shown in chapter 6.

To mitigate this limitation of the template, we conduct two additional experiments and the results are shown in the last two rows of Table 4.1 and 4.2. For T-Fusion(A), we augment the five positive training signals to 125 signals by performing a slight random rotation, adding a small perturbation, and swapping a random segment given a randomly picked pair of training signals. The idea is to artificially create signal variation so that the learned model may be more robust. For T-Fusion(E), we use

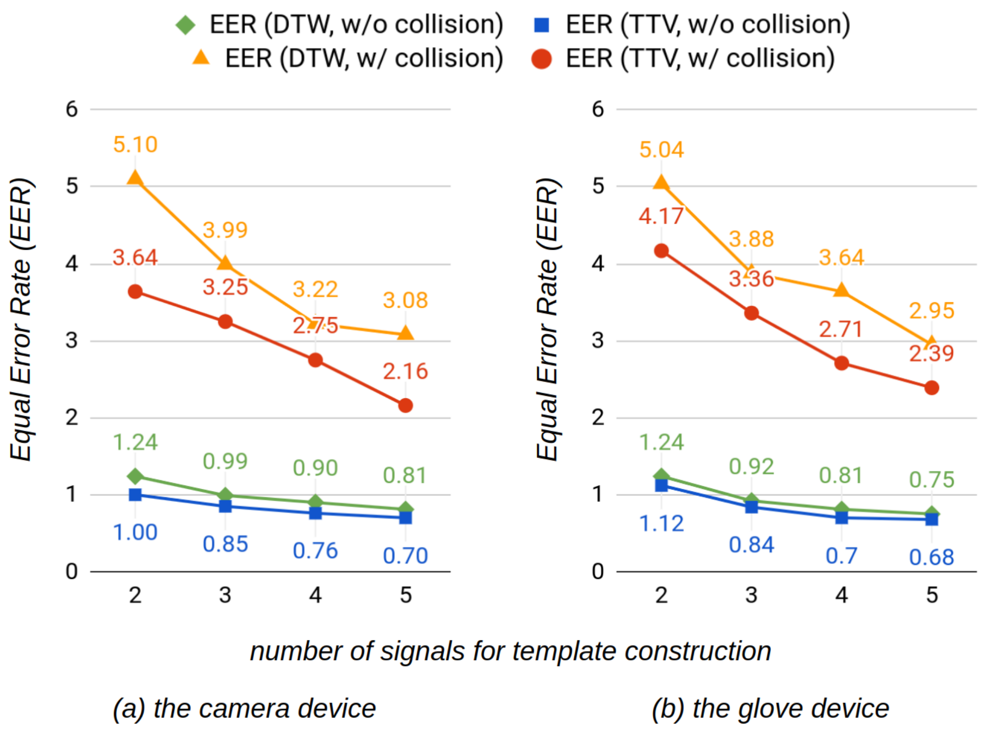


Figure 4.23: Authentication Performance Change Respect to the Number of Signals for Template Construction

each of the five positive training signals as a template, train five individual SVM models, and during the testing we take the minimum of the five scores generated by the five models as the ensemble score. The idea is to use multiple templates instead of one and find the best match. These methods show slight improvements.

Another question is whether the weights learned from other accounts at registration in T-Fusion and S-Fusion can work in general. One may argue that the possible guessing attacks can be enormous and such a trained model can only reject those signals generated by writing passcode strings from other accounts because the contents of these strings are “seen” in the registration process. To answer this question, we conduct another set of experiments where fewer randomly selected negative training samples are used to learn the weights at registration. The results are shown in Figure 4.24. The performance degrades slightly with less negative training sample. However, even if signals from most of other accounts are not observed in training, the matching algorithm can still work. One possible explanation is that the features, *esp.*, D_{ij}, are generally large for negative training samples and they form similar cluster patterns in high dimensional feature space as Figure 4.21. Hence, a linear maximum margin decision plane fitted from training data (*i.e.*, SVM) is relatively robust with the variation of the cluster shape. This also explains why we can obtain a reasonably good performance with only five positive training samples. We believe this is the advantage of template matching based methods since the system only needs to calculate a rough decision boundary relative to the template (which is generally easy), rather than learning a model that must "remember" the templates as in HMM or deep neural network based methods (which is generally hard). In fact, we tried a deep neural network but the performance is inferior. However, since the templates are stored and retrieved in alignment, they are similar to plain text password and they can not be hashed, which needs certain protection mechanism. This is addressed in chapter 6.

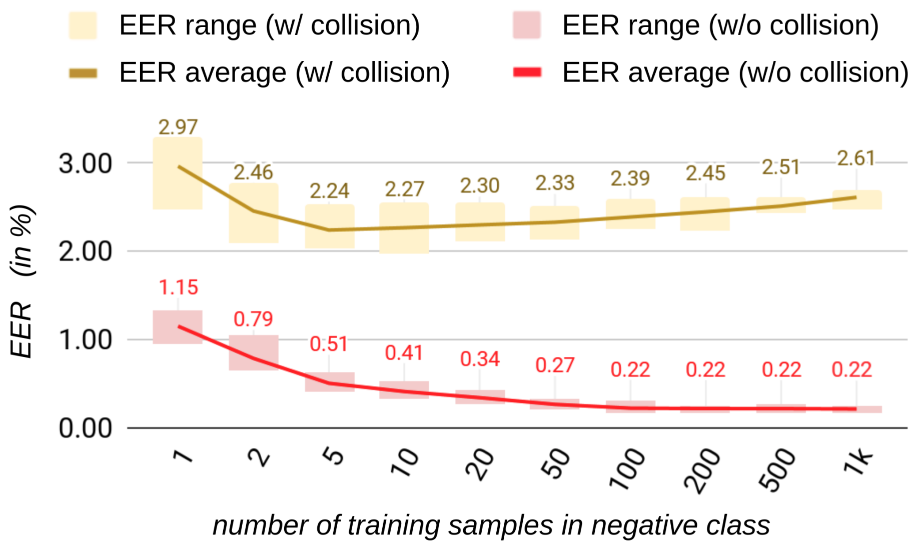


Figure 4.24: Performance Change of the T-Fusion Method with Respect to the Number of Negative Training Samples

It is interesting to see that both the camera device and the glove device share a lot in common in the characteristics of the features and the authentication performances, even though these two types of sensors use fundamentally different motion tracking methods. Our signal model and preprocessing steps removes the sensor dependent information. As a result, we believe our feature analysis and authentication framework can also apply to a broad range of devices. Still, there are small differences between the results of different devices. For example, with the glove device, the signal quality is slightly better, and the user behavior is slightly more consistent. This may be the results of the lack of the field-of-view restriction with the glove device. The hand movement is more stable if a user writes it more naturally with the "muscle memory" rather than attention. However, there might be usability concerns for asking the user to put on a glove. Hence, we will continue our study with a light-weight wearable device like a ring with the same inertial sensor in the future. Another difference is that the tip of the index finger is tracked by the glove, while the center of the palm is tracked by the camera. It seems that the fingertip moves more significantly than the palm center in handwriting in the air.

4.8 Analysis of The Hard Cases

Besides collective performance results, in Figure 4.25, we also show the ranges of matching distance scores δ of the testing signals in different classes using the Feature Fusion method. To accommodate the page width, two thirds of the accounts are randomly selected to generate the figure. The accounts are sorted by the maximum score of the five positive testing signals to show a trend from the "easy cases" on the left to the "hard cases" on the right. On the top half of this figure, we show the ranges of distance scores for signals with $c =$ *same* and signals with $c =$ *diff*. These distance scores are obtained in the experimental evaluation protocol step 2 in section 4.7. Here, "diff-min" and "diff-avg" are the minimum and average of the distance scores for those login request signals with class label $c = \textit{diff}$ for each accounts. The range of these scores are shown by the colored area as "diff-range". Similarly, "same-max" is the maximum of the distance scores for those login request signals with class label $c = \textit{same}$ for each accounts, and the range of these scores are shown by the colored area as "same-range". The intersection between the "diff-range" and "same-range" are shown in darker colors and denoted as "overlap". On the bottom half of this figure, we show the ranges of distance scores for signals with $c = \textit{same}$ and signals with $c = \textit{collision}$ in a similar way. These distance scores are obtained in the experimental evaluation protocol step 3 in section 4.7. It should be noted that there are way more signals with class label $c = \textit{diff}$ than those with class label $c = \textit{collision}$, *i.e.*, 359 * 5 vs. 10 * 5. Hence, the range denoted as "collision-range" has larger uncertainty, and on the figure it "jumps" in a random way for different accounts.

It can be observed that for most of the accounts, there is a gap between the scores computed from login request signals with $c = \textit{same}$ and those with $c = \textit{diff}$ or $c = \textit{collision}$. Clearly, the gap between the class "same" and the class "diff" is larger than the gap between the class *same* and the class *collision*, which shows that the difference of signals due to both the passcode string content and the writing style is more significant than the difference of signals merely due to the writing style.

Still, there are a few testing signals with $c = \textit{diff}$ which have smaller matching scores below a "reasonable threshold" (around 0.4, where FAR is $\approx$ 1e-5 without considering collision). They are the first type of the "hard cases", which has a significant influence on the security strength of the authentication system and further determines the threshold which should be used in practice. These are partly caused by short and simple IAHW passcode strings created carelessly by users, and partly caused by multiple accounts owned by the same user since we mixed the signals for ID strings and signals for passcode strings. On the one hand, even if these passcode

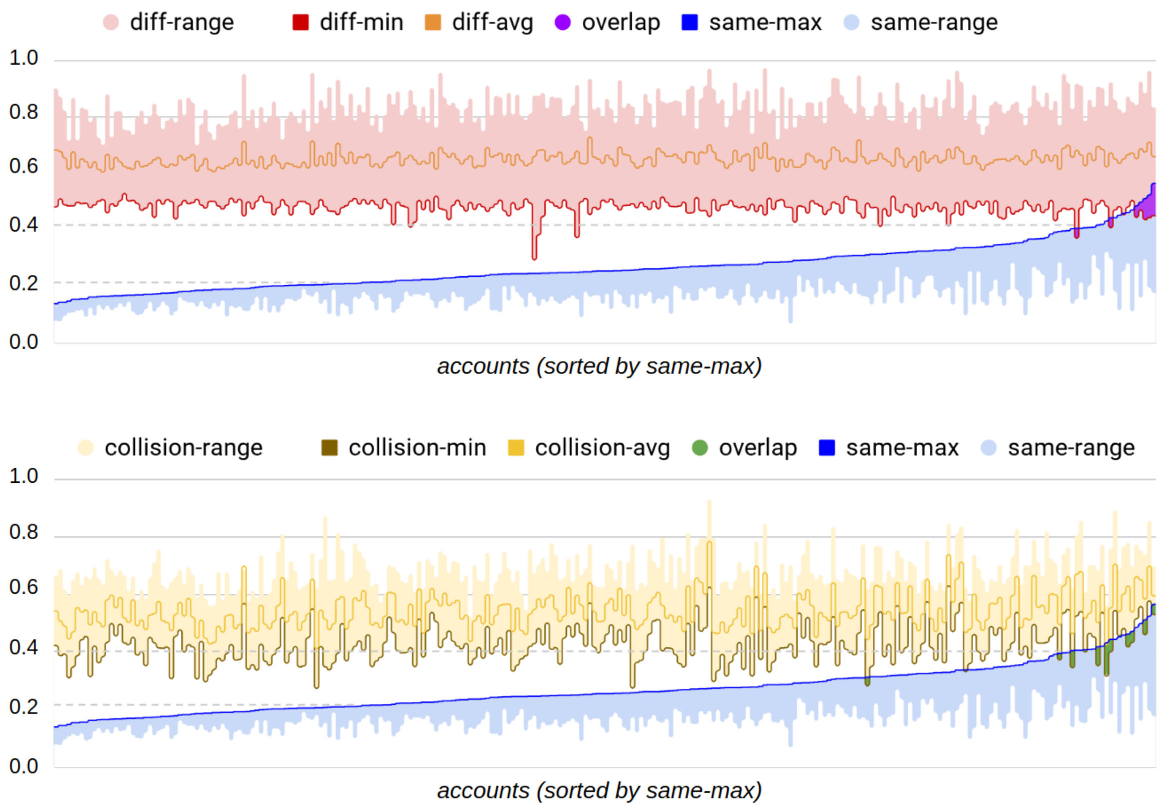


Figure 4.25: Matching Score Distribution of Each Account Using the F-Fusion Method

strings are different in content, they may have similar stroke patterns when writing in the air because of the transitioning strokes. For example, "FMKit" and "EWKje" have very similar strokes in almost identical sequence, although four out of five letters are different. On the other hand, one user may have multiple accounts with very similar or even identical passcode strings. This happens when we intentionally mixed the signals from the ID strings and the signals from passcode strings using the "ID-passcode" dataset to allow each user to have two accounts. For example, the passcode strings for the two accounts of a user is "feng" and "yang", and these signals are very similar because the similarity in content and writing style, as shown in Figure 4.26 (a). This pair of signal and template has a matching distance score of 0.294. Another example is shown in Figure 4.26 (b), where a user creates two accounts using the string "Kaicheng" and "Kirkland". This pair of signal and template has a matching distance score of 0.373. We believe writing style itself is a weak factor, *i.e.*, given a large number of users we can always find users with similar writing styles. However, it is not easy to mimic the writing style of a user if it has a certain level of uniqueness, which is discussed in the spoofing attack analysis in chapter 6.

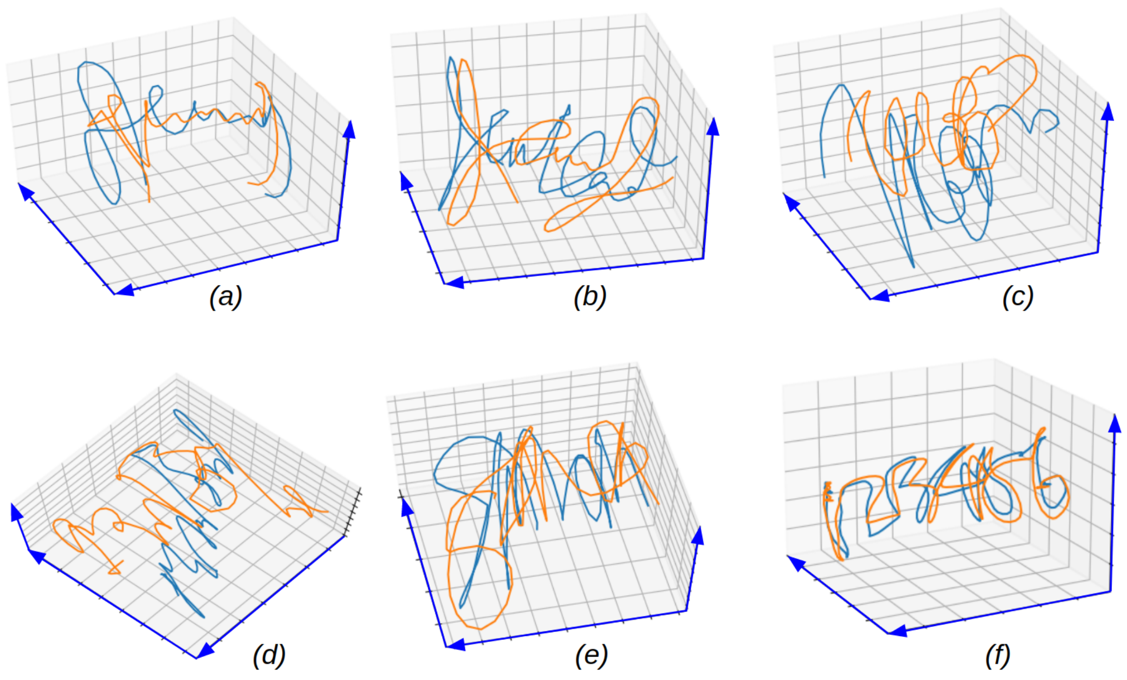


Figure 4.26: Examples of Hard Cases, Blue for the Templates and Orange for the Testing Signals

On the rightmost of the figure, there are testing signals with $c = same$ with large matching distance scores above the reasonable threshold. They are the second type of hard cases which influences the usability. They are usually generated by users that changes the writing style or inconsistent behaviors that cannot accommodated by the system. Two examples of the hard cases are shown here. For Figure 4.26 (c), the signal and the template are generated by the same user writing the same passcode content "Victor", but the user changes the writing behavior, and the resulting δ is

0.524. For Figure 4.26 (d) the signal and the template are generated by the same user writing the same passcode content “harrison67”, and the resulting δ is 0.566. In this example, at the end of the signal there are many meaningless hand movements that cause interference for the posture normalization step in the signal preprosessing procedure as well as the alignment procedure. The posture normalization relies on the hand pointing direction while the two signals have the same pointing direction but different hand moving direction. As a result, the scores of these hard cases are comparable to the typical distance score between a template and a signal with $c = collision$. Such an example for collision is shown in Figure 4.26 (e), where both the user and the imposter wrote “Simon” and the resulting distance score δ is 0.555. On the contrary, in Figure 4.26 (f), we show a template and a signal obtained by the same user writing “123456”, and the distance score δ is only 0.187. We believe that this second type of hard cases is more important because it determines the trade off between the discriminative capability and the tolerance of minor behavior variations. We can set the decision threshold at a conservative level to handle the first type of hard cases, but if this threshold is too conservative such that majority of the second type of hard cases are rejected, it will annoy the user and this system will be less likely to be used in practice.

4.9 Summary

In this chapter we present a user authentication framework with two different types of devices using various features of IAHW passcode including temporal features, statistical features, and hand geometry features. Although the results are promising, it is still in an early stage and requires further investigation, and we believe such an IAHW passcode based authentication method has potential in application, especially in VR headset or home entertainment scenarios with native gesture user interface instead of a keyboard or a touchscreen.

Chapter 5

FMHASH - USING IAHW FOR USER IDENTIFICATION

5.1 Background

Gesture user interfaces are considered as the future way for people to interact with Virtual Reality (VR) or Augmented Reality (AR) applications and other devices such as a smart door bell or smart wearable devices (Lien *et al.*, 2016). Such interfaces can capture and track hand motions in the air, and allow a user to manipulate menus, dialogues, and other virtual objects directly by hand gesture. However, for security related tasks such as sign-up and sign-in, entering the user ID string and password through a virtual keyboard on gesture interfaces become cumbersome due to the lack of keystroke feedback. Existing researches (Okumura *et al.*, 2006; Farella *et al.*, 2006; Liu *et al.*, 2009; Bailador *et al.*, 2011; Chahar *et al.*, 2015; Tian *et al.*, 2013; Lu *et al.*, 2017, 2018) exploit the rich information in native gestures, and esp., IAHW, to authenticate a user. Yet, a usually neglected function is user identification. Authentication is a true or false question, *i.e.*, answering whether the user owns the account which he or she claims to own. On the other hand, identification is a multiple choice question, *i.e.*, answering which account the user wants to login among a database of many accounts. If we make an analogy of the sign-in procedure on a web with a desktop computer, the authentication procedure resembles typing and checking the password, while the identification procedure resembles searching the database given an ID number or ID string. Is it possible to construct a system that is capable of (1) taking a piece of IAHW of an ID string instead of typing, (2) searching a potentially large database of accounts registered using the IAHW, and (3) returning the matched identity or account number with high accuracy and short respond time?

There are challenges for gesture-based user identification due to the unique characteristics of the hand motion. First, hand motion has inherent fuzziness. Even if the same user writes the same string in the air twice, the generated two motion signals are not identical, but with minor variations. Yet, the system should be able to tolerate the fuzziness and identify these two signals as the same user. This is different from typing an ID string of characters twice where even a single bit difference in the typed ID can cause failure in the identification. Second, it is difficult for many native gestures to provide enough information to enable a large account ID space as well as distinctiveness. Third, the traditional method of hash table for indexing a large account database using an ID string or account number does not work with handwriting signal, unless there is a way to generate a fixed size binary hash code from the signal with tolerance of inherent fuzziness (*i.e.*, fuzzy hash).

In this chapter, we propose a framework called **FMHash**, i.e., Finger Motion Hash, to efficiently obtain a user's account ID from the hand motion signal of writing an ID string in the air. FMHash uses a camera-based gesture user input device to capture the hand motion and a deep convolutional neural network (called FMHashNet) to convert the in-air-handwriting signal to a binary hash code (*i.e.*, deep hashing). With the hash code of the signals of all accounts, it further builds a hash table to index the whole account database to enable efficient user identification with hash table search. This is similar to face recognition, where a large database of identities are indexed using faces and an ID can be retrieved by presenting an image of a face. However, FMHash has a few unique advantages compared to face recognition. For example, one face is linked to one person, and a user can neither have multiple faces for multiple accounts nor change or revoke his or her own face. Moreover, the users may be worried about privacy because it is impossible to stay completely anonymous if a website requires face image to register. Yet, with in-air-handwriting of an ID string, the user can have multiple accounts with different ID strings, change or revoke the ID string, and stay anonymous by writing something unrelated to the true identity. In summary, our contributions are as follows:

1) We proposed a deep hashing framework of IAHW for user identification over gesture interface, named FMHash, which can generate fuzzy hash code from IAHW to index user accounts. Our method can accommodate gesture fuzziness by hashing multiple instances of the same handwriting by the same user to the same binary code with high probability of success ($\geq$97% precision and $\geq$92% recall).

2) We designed a neural network model with a regularizer called the *pq-regularizer* and a progressive training procedure. With the *pq-regularizer*, hashcode of IAHW signals of different accounts are separated more than two bits in over 99% of the change. Meanwhile, we can maintain a reasonably fast training speed ($\sim$10 minutes for a full training on our dataset).

3) We provided a detailed analysis on the hash code fuzziness using datasets collected by us with two different types of devices.

The remainder of this chapter is organized as follows. Section 5.2 presents the architecture of the proposed FMHash framework. Section 5.3 provides details of our neural network model. Section 5.4 explains the user identification procedure. Section 5.5 shows the empirical evaluation results. Section 5.6 summarizes this chapter. This chapter is based on our published research paper (Lu *et al.*, 2019).

5.2 System Model

The proposed FMHash framework contains the following five components, as shown in shown in Figure 7.1.

(1) An IAHW hand motion capture device, *i.e.*, the camera device or the glove

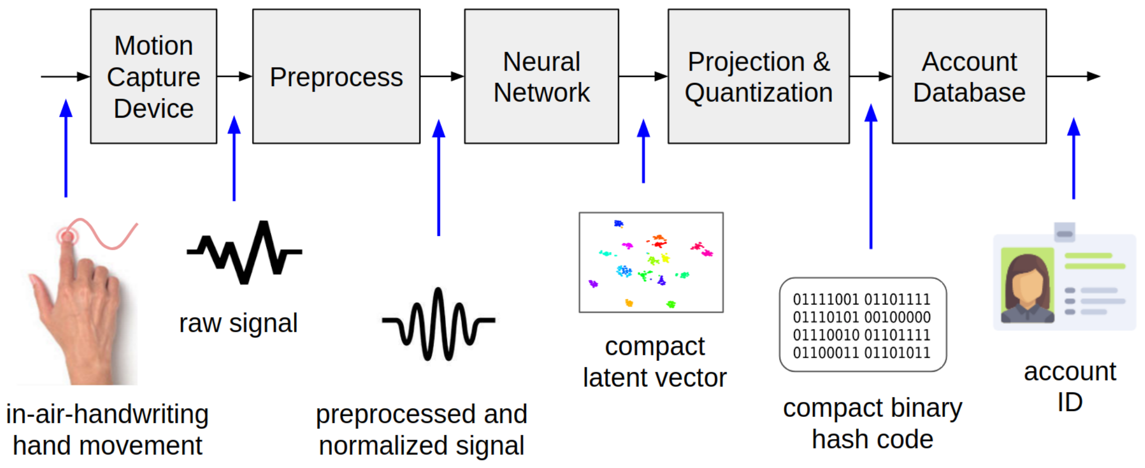


Figure 5.1: The Architecture of the FMHash Framework

device in chapter 3.3.

(2) A preprocessing module that smooths and normalize the captured motion signal, detailed in chapter 3.4. All preprocessing steps are used, including amplitude normalization and temporal normalization. After the preprocessing, each signal has the format of a vector $\mathbf{x}$ of 256 samples and each sample is again a vector containing 18 individual elements. We use the vector representation in this chapter to accommodate the widely used terminologies of neural network.

(3) A deep neural network that takes a preprocessed motion signal $\mathbf{x}$ as input and generates a high dimensional floating point latent vector $\mathbf{h}$. This step is denoted as a function $f(\mathbf{x}) = \mathbf{h}$, detailed in section 5.3.

(4) An additional neural network layer that projects the latent vector $\mathbf{h}$ to low dimensional space and quantize the projected result to B-bit binary fuzzy hash code $\mathbf{b} \in \{-1, +1\}^B$. This step is denoted as another function $g(\mathbf{h}) = \mathbf{b}$. The deep neural network and the additional projection-quantization layer are implemented together, which are collectively called the FMHashNet, also detailed in section 5.3. The value of B can be 16, 32, 48, 64, etc., which is configured by the system administrator. Note

that we use $\{-1, +1\}$ instead of $\{0, 1\}$ for the convenience of explaining the output of the neural network. in practice, it is trivial to convert a sequence of numbers made of $\{-1, +1\}$ to a binary hash code (just replacing any occurrences of -1 to 0 and convert the number sequence to a binary code).

(5) An account database that stores a collection of account tuples <ID, $\mathbf{b}^{ID}$, $\mathbf{h}^{ID}$ >, where ID is the account ID (usually a unique number generated by the system at registration), $\mathbf{b}^{ID}$ and $\mathbf{h}^{ID}$ are the hash code and latent vector corresponding to the account. This account database also contains a hash table that maps $\mathbf{b}^{ID}$ to account ID, so that once a fussy hash code $\mathbf{b}^{ID}$ is obtained from the neural network, the corresponding account ID can be retrieved. The usage of the account database and the user identification procedure are detailed in section 5.4.

Optionally, the ID generated by the hash table lookup step using the account database can be considered as a candidate ID. Multiple candidate IDs can be retrieved given a a fussy hash code $\mathbf{b}^{ID}$ by considering all neighbors within a certain Hamming distance. Once a set of candidate IDs are obtained, there is an additional verification module that runs a procedure similar to the authentication procedure explained in chapter 4 by comparing the IAHW signal of the ID string in the identification request to the templates stored in the account corresponding to those candidate IDs, which can further eliminate wrong identification results. This module is detailed in chapter 6.

5.3 FMHashNet

An overview of the FMHashNet is shown in Table 5.1. There are multiple design goals for this neural network. First, given a pair of IAHW signals $(\mathbf{x}_1, \mathbf{x}_2)$, if they are generated by the same user writing the same ID string, the corresponding hash codes $(\mathbf{b}_1, \mathbf{b}_2)$ should be the same in most cases or differ only in one or two bits

Table 5.1: The FMHashNet Architecture

layer	kernel	output	#para
input: 256 * 18			
conv-pool1	$3\rightarrow1$ conv, $2\rightarrow1$ max pool	128 * 48	1.3k
conv-pool2	$3\rightarrow1$ conv, $2\rightarrow1$ max pool	64 * 96	14k
conv-pool3	$3\rightarrow1$ conv, $2\rightarrow1$ max pool	32 * 128	37k
conv-pool4	$3\rightarrow1$ conv, $2\rightarrow1$ avg pool	16 * 192	74k
conv-pool5	$3\rightarrow1$ conv, $2\rightarrow1$ avg pool	8 * 256	147k
fc (latent)	fully connected	512	1,048k

layer	output	#para
softmax	N	512*N
cross-entropy loss		

layer	output	#para
projection	B	512*B
pairwise loss		

sometimes due to the variation of the writing behavior of the user. However, if they are generated from different ID strings (regardless of the same user or different users), $(\mathbf{b}_1, \mathbf{b}_2)$ should differ by at least three bits. Second, the neural network should learn latent representations $\mathbf{h}$ for each ID string that distinguish the differences of these ID strings so as to facilitate the projection. Third, it should be easy to train and fast to converge as it may need to be updated when new users are added.

To achieve these goals, we design the FMHashNet in the following way, as shown in Table 1 and illustrated in Figure 5.2. First, we apply five convolutional and pooling layers with VGG-like kernel (Simonyan and Zisserman, 2014) and a fully connected layer to map input signal $\mathbf{x}$ to latent vectors $\mathbf{h}$. Both the convolutional layer and the fully connected layer have leaky ReLU activation. Next, the projection layer projects the latent vector $\mathbf{h}$ to a space in the same dimension as the desired hash code, i.e.,

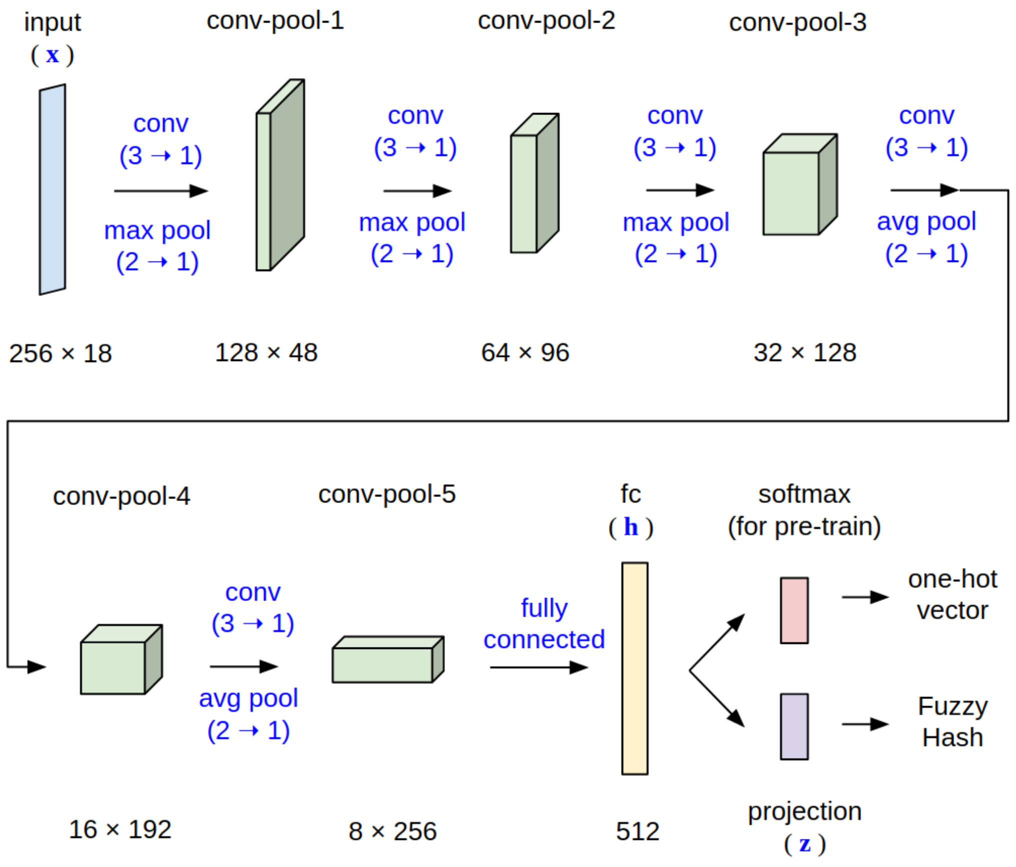


Figure 5.2: An Illustration of the FMHashNet Architecture

$\mathbf{z} = W\mathbf{h} + \mathbf{c}$, where $\mathbf{z}$ is the projected vector whose dimension is B. Here W and c are trainable parameters. After that, the hash code is generated by taking the sign of the projected vector $b_i = sign(z_i), 1 \leq i \leq B$. This is essentially partitioning the latent space by B hyperplanes to obtain at most 2^B regions, where each region is associated with a unique hash code. Additionally, a softmax layer for classification with N categories is added in parallel with the projection layer to facilitate the training the neural network.

We believe that the training process of the FMHashNet is essentially setting up a mapping function from the 256-by-18 dimensional signal space to the B dimensional Hamming space. The mapping function is also equivalent to placing all these signals from the registered accounts into these 2^B regions, such that different accounts should be placed in different regions. This is achieved progressively in two steps. First, we train the network with the softmax layer and cross-entropy loss to allow the convolutional filters to converge. In this step the projection layer is not activated. Note that the softmax classification layer does not need to contain all accounts if the account number is large, or even a independent dataset for pretraining can be utilized.

Second, we train the network using the projection layer with the following pairwise loss L, and minibatches of $2M$ pairs of signals $(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}), 1 \leq i \leq 2M$. Here $2M$ (*i.e.*, the batch size) is a hyperparameter chosen empirically. Larger batch size leads to more computation in the training but more stable convergence behavior. In the minibatch, half pairs of signals are from the same account ($y^{(i)} = 0$), and the other half pairs of signals are from different accounts ($y^{(i)} = 1$). The loss function L is as follows:

$$L = \frac{1}{2M} \sum_{i=1}^{2M} L^{(i)},$$

$$L^{(i)} = (1 - y^{(i)})||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}|| + y^{(i)}\max(m - ||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}||, 0)$$
$$+\alpha(P(\mathbf{z}_1^{(i)}) + P(\mathbf{z}_2^{(i)})) + \beta(Q(\mathbf{z}_1^{(i)}) + Q(\mathbf{z}_2^{(i)})).$$

Here $||.||$ is the Euclidean norm. In this loss function, the first term forces the projected vectors of the same account to the same value, and the second term forces the projected vectors of different accounts to separate at least m in Euclidean distance. The remaining terms $P(\mathbf{z})$ and $Q(\mathbf{z})$ are the so-called *pq-regularizer* which is specially designed to help place all registered accounts into different regions and avoid ambiguity in quantization. These two terms are defined as follows:

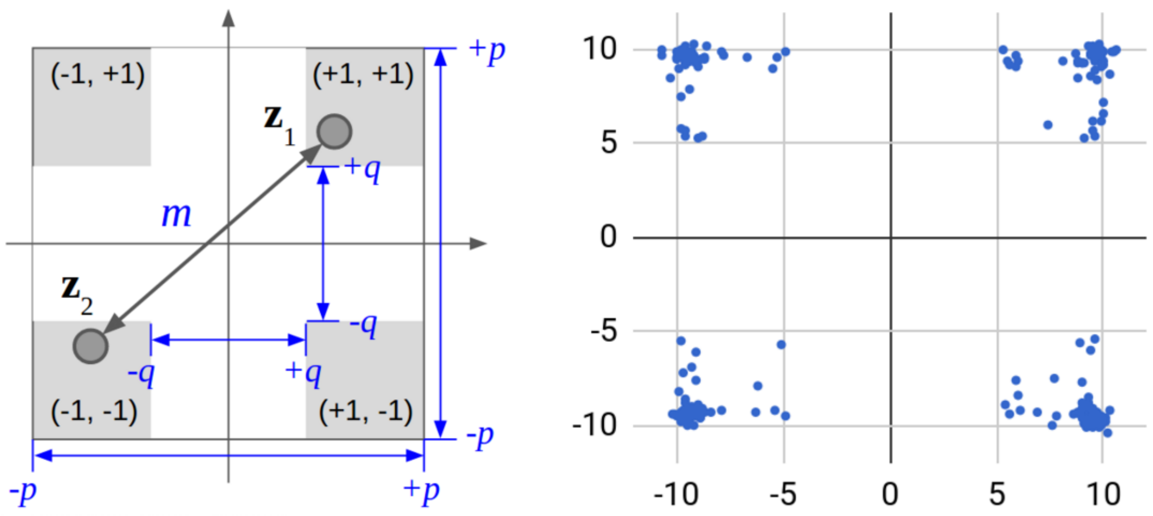


Figure 5.3: The Effect of the Regularizer

$$P(\mathbf{z}^{(i)}) = \sum_{j=1}^{B} \max(|z_j^{(i)}| - p, 0),$$

$$Q(\mathbf{z}^{(i)}) = \sum_{j=1}^{B} \max(q - |z_j^{(i)}|, 0),$$

where p and q are hyperparameters chosen empirically, $|z_j^{(i)}|$ is taking absolute value of the jth component of $\mathbf{z}^{(i)}$. This regularizer forces each element of the projected vector $\mathbf{z}$ to reside in the region $[-p, -q]$ or the region $[+q, +p]$. The element $\mathbf{z}^{(i)}$ is quantized to the bit -1 if it is less than zero or bit +1 if it is greater or equal to zero, so as to generate the final fuzzy hash code bit b_i. The p prevents $\mathbf{z}^{(i)}$ taking unbounded large values, and the q prevents $\mathbf{z}^{(i)}$ taking values close to zero which causes quantization ambiguity.

With a careful choice of m, a pair of $(\mathbf{z}_1^{(i)}, \mathbf{z}_2^{(i)})$ of different accounts can be separated in different places in the latent space, and hence, the binary codes obtained after quantization will differ at least on or more bits. Essentially, the pq-regularizer implement a virtual "force" in the training process that pushes $(\mathbf{z}_1, \mathbf{z}_2)$ from different accounts to different regions, which is illustrated in Figure 5.3 (left). The actual effect is shown in Figure 5.3 (right), which is obtained by plotting the first two dimensions of $\mathbf{z}$ of 200 training signals from 200 different accounts, with p=10 and q=5. One example choice of m as in our experiment is $p\sqrt{B}$, which is the Euclidean distance from the origin to the point $\mathbf{z}^* = (p, p, ..., p)$. This forces the hash code of signals of different accounts differ at least one bit. Our experience shows larger m helps separation, but hurts convergence. The hyperparameter α, β controls the portion of contribution of the regularizer in the total loss and gradients. Our design differs from most related works of deep hashing which try to minimize quantization loss (i.e., forces the projected vector to be close to the vertices of Hamming hypercube). Instead, we map the input to a bounded Euclidean space and push them away from the decision boundary $z_j = 0$, where a relatively large region that can be quantized to the same bit value regardless of the quantization error. The effectiveness of our regularizer relies on the assumption that ID strings are distinctive, which is true in an identification system but not in an image retrival system. Meanwhile, both the FMHashNet activation function and the regularizer are piece-wise linear, which is easier to compute and train compared to the saturation methods such as tanh or sigmoid relaxation commonly used in deep hashing.

5.4 Account Database and User Identification Procedure

The user identification system maintains an account database, where each account contains a tuple of <ID, $\mathbf{b}^{ID}$, $\mathbf{h}^{ID}$ >. At registration time, the system generates a unique account ID for the registered account. The user is asked to create an ID string and write it K times. The obtained K IAHW signals $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(K)}\}$ are utilized to train the FMHashNet. Once the training is finished, we can use the training signals to construct $\mathbf{b}^{ID}$ and $\mathbf{h}^{ID}$ as follows:

$$\mathbf{h}^{ID} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{h}^{(i)} = \frac{1}{K}\sum_{i=1}^{K} f(\mathbf{x}^{(i)}),$$
$$\mathbf{b}^{ID} = g(\mathbf{h}^{ID}) = \mathrm{sign}(W\mathbf{h}^{ID} + \mathbf{c}),$$

where $f()$ is the deep neural network $g()$ is the projection and quantization process, and $sign()$ is element-wise sign function. A hash table is also constructed to index all account tuples using the hash code $\mathbf{b}^{ID}$, *i.e.*, in this hash table the hash code $\mathbf{b}^{ID}$ is the key and the account ID is the value.

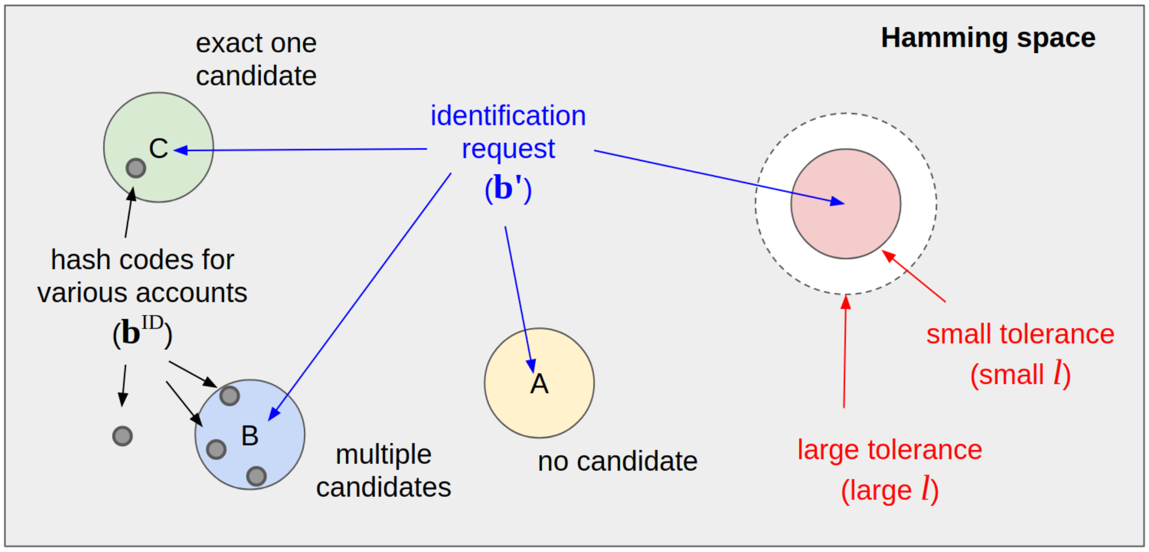


Figure 5.4: An Illustration of the Identification Procedure

At identification time, given a preprocessed IAHW signal $\mathbf{x}'$ as an identification requests, the following steps are proceeded to obtain the account ID. First, we run the forward path of FMHashNet to obtain a latent vector $\mathbf{h}'$ and $\mathbf{b}'$. Second, we search the hash table using $\mathbf{b}'$ with a tolerance of l bits. If l is 0, we just search the hash table using $\mathbf{b}'$. If l is greater than 0, we construct a set of hash codes S containing all possible hash codes with a Hamming distance less or equal than l bits from $\mathbf{b}'$, then we search the hash table multiple times with each element of a collection of hash codes S. The rationale is that the fuzziness in the writing behavior eventually lead to errors that make $\mathbf{b}'$ differ from the hash code of its real account, but this difference should be smaller than l bits. We usually set l to 1 or 2 to limit the total number of searches for prompt response. It should be noted that the set S is introduced to explain the procedure in a simple way. In practice, since l is a system wide configurable parameter, the implementation may use a different way to allow one account to occupy 2^l buckets during the registration instead of generating 2^l search at identification time.

An illustration is shown in Figure 5.4. The colored circles are the identification requests $\mathbf{b}'$, and the size of the circle represents the tolerance level of l bits. Assuming the dimension of the binary hash code is B, after the registration, the hash codes of all existing accounts $\mathbf{b}^{ID}$ will spread on the Hamming space. Hence, one such request with a tolerance of l bits will cover one or multiple buckets in the Hamming space during the hash table searching. There are three cases, denoted as the colored circles named A, B, and C. In case A, there is no corresponding accounts retrieved. In case B, there is exact one candidate account retrieved. In case C, there are multiple candidate accounts retrieved, and in this case, we compare the $\mathbf{h}^{ID}$ of the candidate accounts and the $\mathbf{h}'$ to select the nearest neighbor of $\mathbf{h}'$ in Euclidean distance as the final candidate. Finally, the ID of the candidate account is returned as the identified ID. If no candidate account is found, the system returns “unidentified”.

5.5 Experimental Evaluation

We use the “ID-passcode” dataset introduced in chapter 3.6 to evaluate the FMHash framework. We mixed the ID strings and the passcode strings to create one account for each string (*i.e.*, $N = 360$), similar to that in chapter 4.

The FMHashNet is implemented in TensorFlow (Abadi *et al.*, 2016) on a Nvidia

GTX 1080 Ti GPU. The weight parameters are initialized with the Xavier method (Glorot and Bengio, 2010) and the Adam optimizer (Kingma and Ba, 2014) with a initial learning rate of 0.001 is used. The leaky ReLU negative slope is set to 0.2. For the training protocol, we first enable the softmax layer and train the whole network using cross-entropy loss for 1,000 iterations as a “warm-up”. In this mode, the projection layer is disabled. The network is provided with mini-batches of IAHW signals randomly picked from the registration signals with the corresponding account ID as labels, and the network is trained in a similar way to a classification network.

Then we disable the softmax layer and enable the projection layer. In this mode, we train the whole network using the pairwise loss with pq-regularizer for another 10,000 iterations. During these 10,000 iterations, α is always set to 0.1, and β is initially set 0.0001 for the first 2,000 iterations, and gradually increased 10 times per every 2,000 iterations until 0.1. The regularizer hyperparameter p is set to 10, q is set to 5. The inter-class distance m is set to $p\sqrt{B}$, and the hash code size B is 16, 32, 48 and 64 respectively in multiple experiments. The training pairs are selected online, and minibatch size is set to 200 pairs, where the signals of each of the first 100 pairs are from the same account and the signals of each of the second 100 pairs are from different accounts. For a pair of signals from the same account, we randomly select an account and two training signals of that account. For a pair of signals from different accounts, we calculate the account hash code $\mathbf{b}^{ID}$ for all accounts every 20 iterations, and select a pair from those accounts whose hash codes differs less than three bits. If no such account exists or the number of such accounts is less than the minibatch size, we randomly choose pairs of signals from different accounts.

Another major challenge we encountered is the limited amount of training data (only five signals per account). To overcome this challenge, we augment the training dataset in two steps. First, given K signals $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(K)}\}$ obtained at registration, for each $\mathbf{x}^{(k)}$ in this set, we align all the other signals to $\mathbf{x}^{(k)}$ to create $K-1$ additional signals using Dynamic Time Warping (Berndt and Clifford, 1994), and in total we can obtain K^2 signals (in our case 25 signals). Second, we randomly pick two aligned signals and exchange a random segment to create a new signal, and this step is repeated many times. Finally each account has 125 training signals.

We trained the FMHashNet with different hash code sizes $B = 16, 32, 48, 64$ and tested it with fuzziness tolerances $l = 0, 1, 2$. In a single experiment, we train the FMHashNet from scratch using the registration signals with augmentation and ran the identification procedure with the testing signals for each account. Given a testing signal $\mathbf{x}$ of an account A, if $\mathbf{x}$ is correctly identified as account A, it is a true positive of account A; if it is wrongly identified as some other account B, it is a false negative of account A and false positive of account B, also counted as a mis-identification; if it is not identified as any account, it is counted as a failure of identification. The performance metrics are the average precision of all accounts, the average recall of all accounts, the mis-identification rate (total number of mis-identification divided by N$\times$5), and the failure of identification rate (total number of failure of identification divided by N$\times$5). The results are shown in Figure 5.5 and 5.6.

Due to the stochastic nature of neural network, the results are obtained by averaging the performance of five repetitions of the same experiment with the same parameter settings. For each repetition, the neural network is trained from scratch. The ID verification step is not used in this evaluation. These results show that our FMHash framework performs consistently in the user identification task on our dataset with different hash code sizes. In general, longer hash code size provides better security since it is more difficult to guess without knowing the writing content, but it is also more difficult to train due to the added parameters. Also, a larger fuzziness tolerance l leads to less failure of identification (*i.e.*, improved recall) but

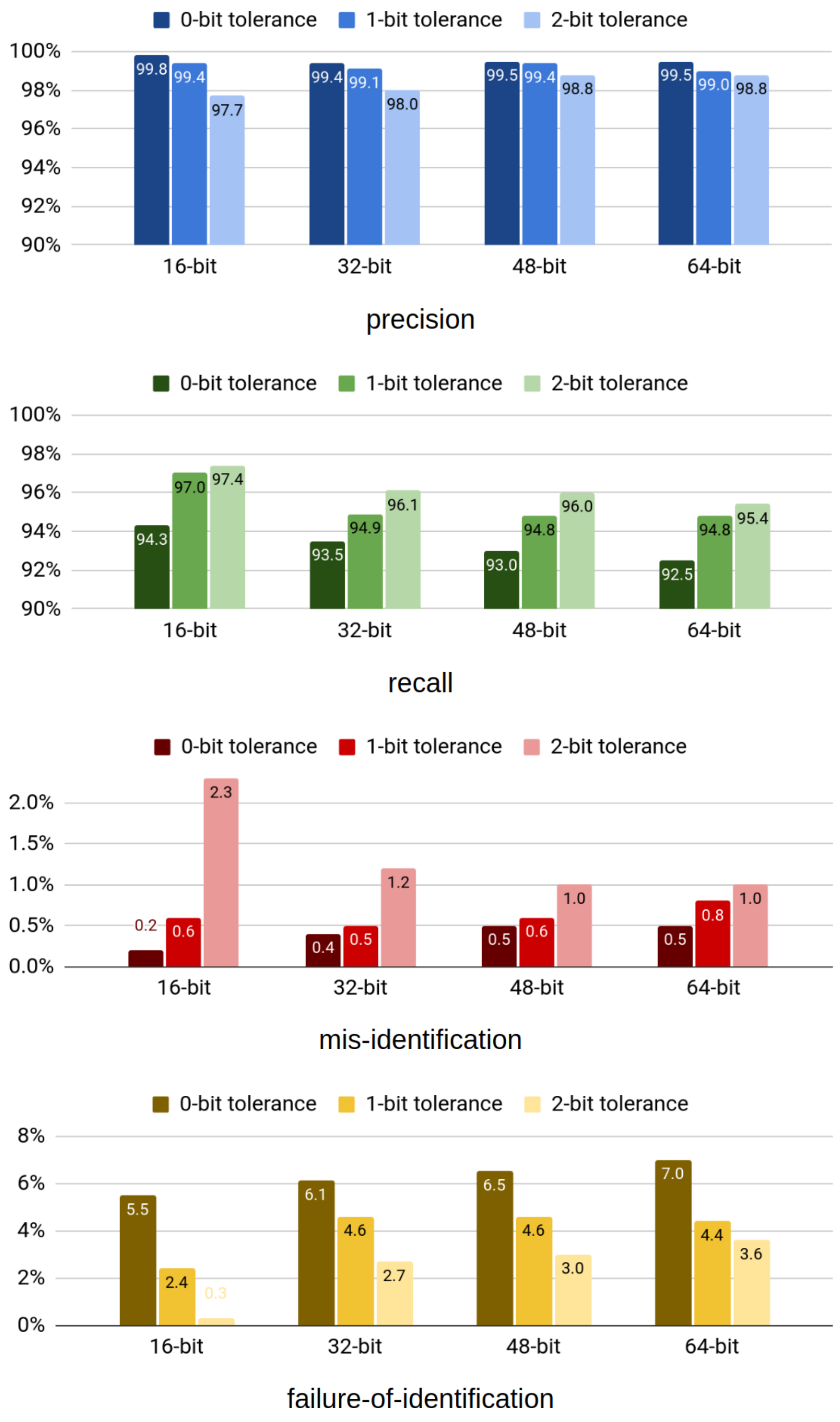


Figure 5.5: Identification Results With the Camera Device

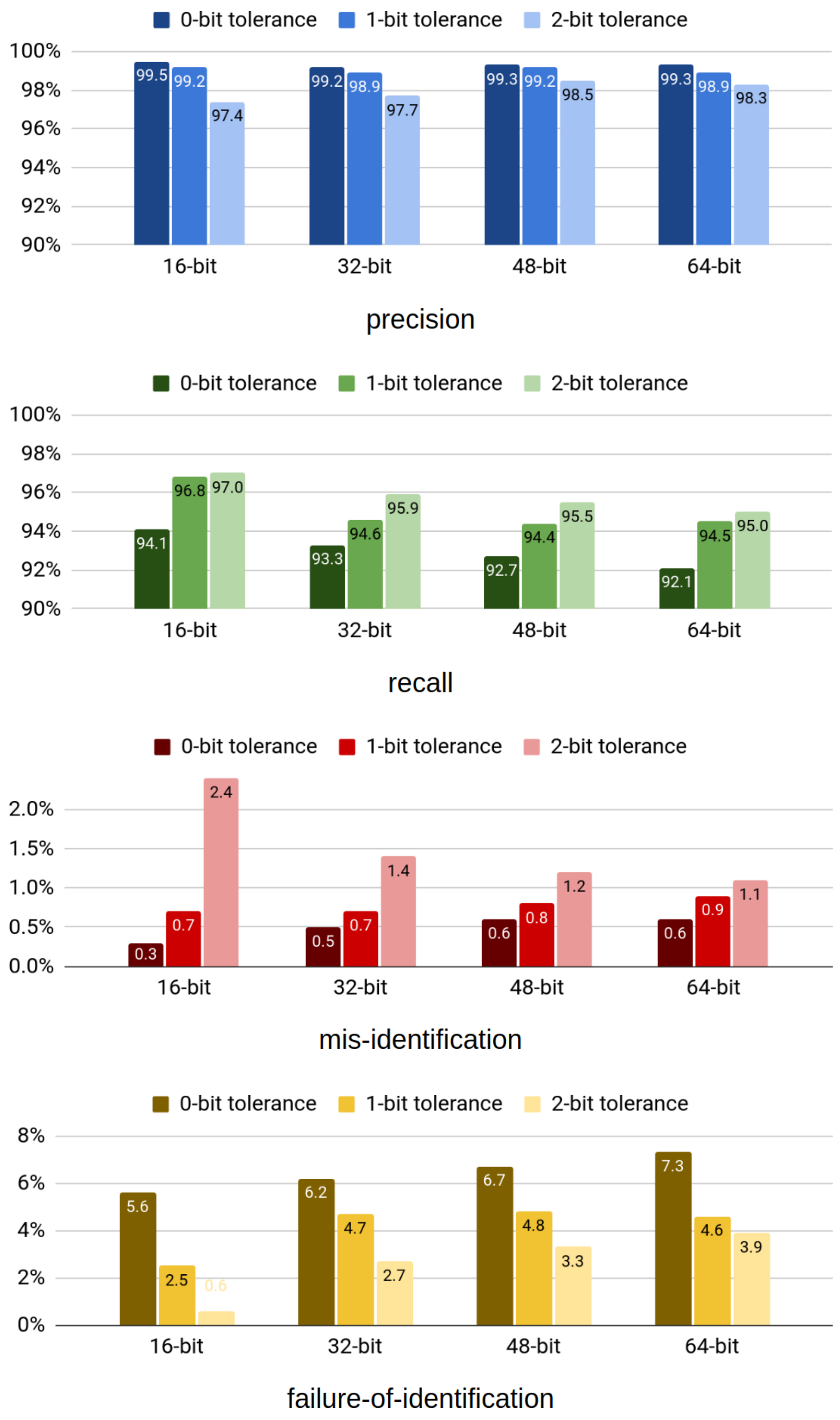


Figure 5.6: Identification Results With the Glove Device

more mis-identifications. For a practical identification system, we recommend to set $l = 0$ without ID verification for simplicity or set $l = 2$ with ID verification for better security.

Next, we evaluate how much fuzziness is in the hash code caused by the inherent variation in the IAHW. As shown in Figure 5.7 (left), 1.8% of the testing signals are hashed more than 2 bits away from the their real accounts. Such fuzziness is mitigated by the separation of the hash codes of different classes, as shown in Figure 5.7 (right), *i.e.*, in 99% of the case the hash code of a signal of one account is at least three bits far away from the hash code of other accounts.

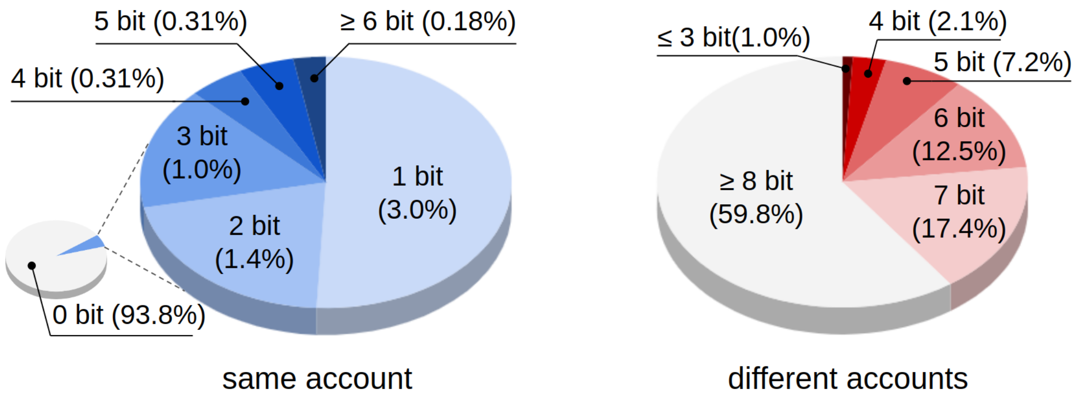


Figure 5.7: Distribution of the Hamming Distance Between the Hash Code of the Account and the Hash Code of Testing Signals.

Then, we study how the hash codes of all accounts are placed in the Hamming space. Here we use 16-bit hash codes for convenience. First, the distribution of zero and one of each bit in the hash code generated in an experiment is shown in Fig. 5.8 (left). There are roughly equal amounts of zeros and ones in each bit, indicating that the hash codes are evenly spread. Second, the correlation of every bit pair is shown in Fig. 5.8 (right). In this figure, the correlation is close to zero for every pair of bit i and j if $i \neq j$, indicating that each bit carries different information in the hash code. Third, the distribution of the distances of hash codes between any two accounts are

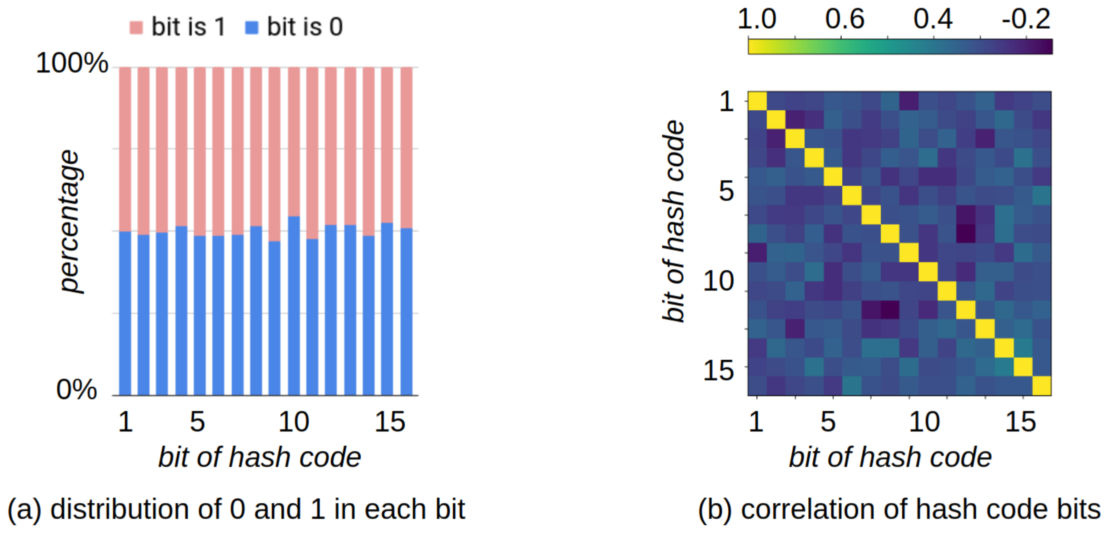


Figure 5.8: Distribution and Correlation of the Zeros and the Ones in the Hash Code

shown in Figure 5.9, where the minimum distance is 3 to 4 bits, the average is 7 to 8 bits, and the maximum is 13 to 14 bits. From this figure, we can see that hash codes of the accounts are sparsely located in the Hamming space and the distance between any two accounts are at least a few bits away. This property of sparsity is the key for an identification system, and it is from our careful design of the regularizer.

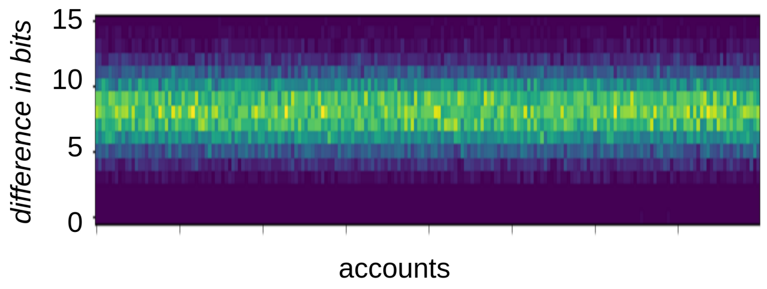
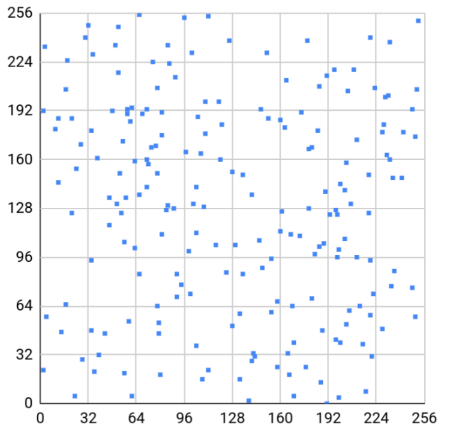


Figure 5.9: Distribution of the Hamming Distance Between the Hash Code of the Account and the Hash Code of Testing Signals for Each Account.

At last we show examples of the generated hash codes in 16 bits for a few accounts in Figure 5.10 (left). It should be noted that due to the stochastic nature of the initialization of the neural network, each trained network will generate a different set of hash codes for all accounts. We also plot 200 hash codes in 16 bits from 200 accounts in Figure 5.10 (right), where the x-axis is the first 8 bits and the y-axis is the second 8 bits. This visualization can provide an overview of the random spreading of these hash codes in the Hamming space.

account	hash code
account A	11101101 11010110
account B	00010010 01001001
account C	11010100 11110100
account D	11010001 01001010
account E	00000110 10010111
account F	11101101 01000110
account G	00111011 11011111
account H	11111101 01001100
account I	11111101 11010110
account J	11111101 11010110
account K	11010100 10110000

(a) examples of accounts with hash codes



(b) illustration of the spread of hash codes

Figure 5.10: Examples of Generated Hash Codes for Multiple Accounts

5.6 Summary

In this chapter, we present a user identification framework named FMHash that can generate a compact binary hash code and efficiently locate an account in a database given a piece of IAHW of an ID string. the empirical results obtained from our prototype evaluation demonstrates the feasibility of the idea of deep hashing of IAHW for user identification. The ability to convert a finger motion signal to fuzzy hash code gives FMHash great potential for sign-in over gesture input interface. However, it has certain limitations such as the requirement of retraining the neural network on creating new accounts or updating an existing ID. In the next chapter, we will continue improving the proposed framework, investigating the long term performance, and designing possible key generation schemes from the fuzzy hash.

Chapter 6

FMCODE - A UNIFIED IAHW BASED USER LOGIN FRAMEWORK

6.1 Background

Recent studies have shown the feasibility of authentication methods using gestures and the IAHW of a password on various gesture input interfaces, including hand-held controllers (Bailador *et al.*, 2011; Liu *et al.*, 2009; Zaharis *et al.*, 2010), cameras (Sajid and Sen-ching, 2015; Chan *et al.*, 2015; Chahar *et al.*, 2015; Tian *et al.*, 2013), and touchscreens (Frank *et al.*, 2013; Gong *et al.*, 2016; Yang *et al.*, 2016). In chapter 4 and chapter 5, we also present a user authentication framework and a user identification frame work using IAHW. However, there are a few fundamental challenges. First, existing gesture input sensors have different modalities (*e.g.,* inertial, visual, *etc.*) and various capabilities (*e.g.,* differences in accuracy, resolution, sampling speed, field of view, *etc.*). There is no unified login framework with both user authentication and user identification functions. Moreover, there is no existing framework that can abstract and accommodate the two different types of sensors introduced in chapter 3.3. As a result, it is hard to objectively evaluate and compare the performance of different gesture-based login schemes. Second, the captured hand movement contains "fuzzy information", *i.e.,* there are minor variations in speed and shape even for the same user writing the same content. Unlike a password that does not tolerate a single bit difference in matching, this fuzziness leads to difficulty in the design of verification algorithms in the hashed or encrypted domain. As a result, a authentication system usually needs to store a template for alignment and comparison at login time. This is equivalent to storing a plain text password and it is not secure. Third, a login system requires not only an identity secret (like a password) but also an account ID. If this account ID is also obtained from a hand movement signal with fuzziness, it is difficult to use such a signal to build index on a large amount of accounts at registration and search an account efficiently upon a login request. As a result, existing login solutions need to compare the signal in the login request with the template of each account exhaustively.

To address these issues, in this chapter, we propose FMCode (*i.e.*, Finger Motion Passcode), a unified user login framework using IAHW of an ID string and a passcode. In our framework, the hand movement can be captured by either a wearable inertial sensor or a 3D camera, as shown in chapter 3.3. The hand motion signal is preprocessed and abstracted as a time series of physical states of a point on the hand independent from sensor modalities, which means the same login framework can be used with different types of sensor devices. Meanwhile, it does not recognize the writing content, but matches the signals in terms of features, which allows the user to write fast in an illegible way, or write in virtually any language that may be unable to be directly typed on a keyboard. In summary, our contributions are as follows.

1) We propose a unified login framework for gesture interface using finger movements in the air, which can perform both effective user authentication and efficient user identification without exhaustive account database search. Our framework supports two different gesture input devices including a wearable inertial sensor or a contactless 3D camera, and it enables fair comparison of performance with different sensors among different writing contents.

2) We present two template protection schemes using deep hashing, cryptographic hashing, and fuzzy commitment (Juels and Wattenberg, 1999), which allows the login server to store encrypted templates for each account.

3) We build a prototype system that implements the proposed framework and conduct extensive experimental evaluations as well as usability evaluation in various scenarios including active spoofing attacks.

The remainder of this chapter is organized as follows. Section 6.2 describes the system model of our framework. Section 6.3 explains the implementation details including user authentication, user identification, and template protection methods. Section 6.4 presents analysis on the security aspects and section 6.5 presents analysis on the usability aspects. Section 6.6 shows further discussions on the practical issues and section 6.7 summarize this chapter. This chapter is based on our published patent (Huang and Lu, 2020).

6.2 System Model

The FMCode framework contains two functional modules, as shown in Figure 6.1: a gesture user interface device that is equipped with motion capture sensors, and a server that stores the account database and runs the algorithms for registration and login. The hand motion capture devices and signal preprocessing steps are detailed in chapter 3.3 and chapter 3.4. Similar to password based login systems, our framework requires an initial registration comparable to the password "sign up" step. In this workflow, the login system will ask the user to create an account and write an ID string and a passcode string multiple times. After that, the user can login by writing the same ID string and password and the login server process the hand motion signals through a login workflow.

At registration, the user is required to create an account by creating an ID string as well as a passcode string, and write them in the air for a few times (in our demo system, we use five times). An ID string is unique and distinctive but it is usually not a secret, while a passcode does not need to be unique but it is a secret. In this step, the obtained hand movement signals are preprocessed on the device side and send to

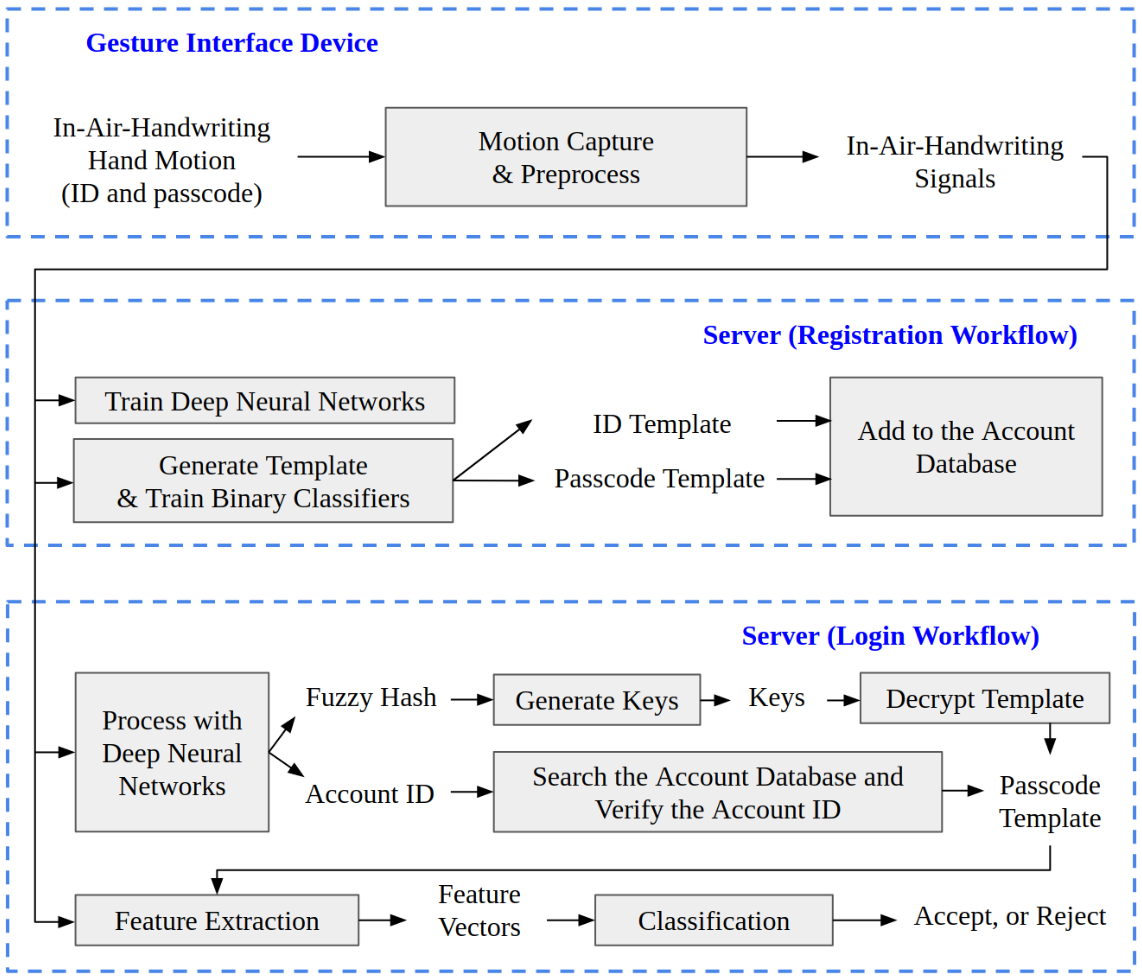

Gesture Interface Device
In-Air-Handwriting Hand Motion (ID and passcode)
Motion Capture & Preprocess
In-Air-Handwriting Signals
Server (Registration Workflow)
Train Deep Neural Networks
Generate Template & Train Binary Classifiers
ID Template
Passcode Template
Add to the Account Database
Server (Login Workflow)
Process with Deep Neural Networks
Fuzzy Hash
Generate Keys
Keys
Decrypt Template
Account ID
Search the Account Database and Verify the Account ID
Passcode Template
Feature Extraction
Feature Vectors
Classification
Accept, or Reject

Figure 6.1: The FMCode Login Framework Architecture and Procedures

the server. The server has an account database where each account in it contains a tuple of (*account number, ID template, passcode template*). The account number is a number or a string allocated by the server to uniquely index each account, and the two templates are computed from the signals obtained at registration. The server may check the uniqueness of the ID string, analyze the "strength" of the passcode, and provide feedback to the user (these additional functions are planned in our future work, not implemented in our current prototype system).

The server has the following three relatively independent functions that work together as a login framework.

1) User Authentication: Given a specific account number and an IAHW signal of passcode as a login request, the server verifies the user identity using a binary classifier with the signal and the passcode template of that account as inputs, and "accept" or "reject" the login request. The account number is obtained using the identification function detailed below. Both the *passcode template* and the binary classifier are computed and trained at registration. In our prototype system, this is implemented with a variant of the T-Fusion method described in chapter 4.

2) User Identification: Given an IAHW signal of the ID string, the server obtains an account number corresponding to this from the account database. In our prototype system, this is implemented with the FMHash method described in chapter 5. It serves as an index of the account database using the IAHW of the ID string, which allows efficient searching at login time.

3) Template Protection: Given the IAHW signals of the ID string and the passcode, the server generates a key, so that the templates can be stored in the account database in an encrypted format, and the user can only be able to decrypt them if both the signals of the ID string and the passcode string are "close" enough to the templates at registration. Note that in this step, the signals are not required to be compared with the templates. In our prototype system, this is implemented with a variant of the FMHash method together with a cryptographic hash function such as SHA-256 or a fuzzy commitment scheme (Juels and Wattenberg, 1999).

The target application scenarios of FMCode are mainly VR/AR headsets and wearable computing platforms that already have a gesture interface but lack a keyboard or a touchscreen. Other application scenarios can also benefit from our framework, *e.g.*, scenarios that prefer touchless gesture interfaces for a level of cleanliness. Our framework can be used for two different purposes. The first one is online user login, where the server is remotely connected to the client via the Internet. For example, the user can sign into an online personal account through the gesture interface on a VR headset. The second one is local user login, where the client and server reside on the same machine. For example, the user can unlock his or her wearable devices through the gesture interface. In addition, our framework can also be used as a supplementary authentication factor in a Multi-Factor Authentication (MFA) system.

FMCode is compatible with existing server infrastructures using password based login because on the server side only software changes are required. These changes including the construction of templates, the implementation of feature extraction algorithms, the classifier for authentication function and the deep neural networks for identification and template protection functions. The requirement of the network between the client and the server for remote user login is the same as existing secure web application through a secure communication channel over SSL/TSL with Public Key Infrastructure (PKI).

6.3 Implementation Detail

We implemented a prototype system for the proposed FMCode framework with the aforementioned two gesture interface devices. This section provides details of the algorithms for the three functions of our framework.

6.3.1 User Authentication

The task of authentication is essentially a binary classification problem. We use a variant of the T-Fusion method in chapter 4 for the user authentication function. Specifically, for registration, we first align all registration signals of passcode strings to the first one and compute a template $\hat{\mathbf{T}} = (\hat{\mathbf{t}}_1, \hat{\mathbf{t}}_2, ..., \hat{\mathbf{t}}_l)$ and the corresponding template variance $\hat{\boldsymbol{\Sigma}} = (\hat{\boldsymbol{\sigma}}_1, \hat{\boldsymbol{\sigma}}_2, ..., \hat{\boldsymbol{\sigma}}_l)$ following the method explained in chapter 3.5. They are stored in the account database as the "passcode template" shown in the server registration workflow of Figure 6.1. The same procedure is also applied to the signals of ID strings to generate the "ID template" shown in the server registration workflow of Figure 6.1.

A slight variation of the T-Fusion method is made in the following feature vector extraction steps. Given an account with a passcode template $\hat{\mathbf{T}}$, a template variance $\hat{\boldsymbol{\Sigma}}$, and a signal $\mathbf{S}$, we run the following steps.

Step 1) Assume $\mathbf{S}$ is preprocessed with amplitude normalization. Align $\mathbf{S}$ to $\hat{\mathbf{T}}$ using Dynamic Time Warping as explained in chapter 3.5.

Step 2) Calculate the element-wise distance $\mathbf{D} = (\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_l)$, where $\mathbf{d}_i = abs(\mathbf{s}_i - \hat{\mathbf{t}}_i)$, where $abs()$ means the element-wise absolute function.

Step 3) Resample the template and the signal with linear interpolation, such that its length is a multiple of a predefined number H. Then, segment $\mathbf{D}$ into H local windows of length $W = l/H$. This is essentially partitioning elements of $\mathbf{D}$ into groups of $(\mathbf{D}_1, \mathbf{D}_2, ..., \mathbf{D}_j, ..., \mathbf{D}_H)$, where $\mathbf{D}_j$ is a group containing the elements in the jth window, *i.e.*, $\mathbf{D}_j = (\mathbf{d}_{j\times W+1}, \mathbf{d}_{j\times W+2}, ..., \mathbf{d}_{j\times W+W})$.

Step 4) For each window $\mathbf{D}_j$, randomly select a sample from it as a feature element, *i.e.*, choose one element $\mathbf{x}_j \in \mathbf{D}_j$, and then use these feature element to form a feature vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_H)$. Here $\mathbf{x}$ is essentially the element-wise distance at a specific local region, which indicates the difference of the stroke segment between the signal and the template at that place. For example, assume $\mathbf{D}$ has 10 samples, segmented to two windows (i.e., $l = 10, H = 2, W = 5$), and we can randomly pick $(\mathbf{d}_2, \mathbf{d}_9)$ or $(\mathbf{d}_3, \mathbf{d}_6)$ as feature vectors.

Step 5) Given a certain window set, step 4 can be repeated multiple times. Additionally, we can consider $\mathbf{d}_i$ as a Gaussian random variable and draw samples from the distribution $\mathbf{d}_i \sim N(abs(\mathbf{s}_i - \hat{\mathbf{t}}_i), \hat{\boldsymbol{\sigma}}_i^2)$ in step 4. As a result, more than one feature vectors can be obtained from a pair of signal and template. This technique allows us to augment the training data from the limited number of registration signals, since asking the user to write more repetitions hurts the usability. Also, it allows us to run the classifier multiple times given a single login request, which can slightly improve the robustness.

With the feature vectors extracted from signals, a soft-margin SVM is trained as our binary classifier at registration. We choose to use SVM mainly because it can be trained efficiently with limited amount of data and high dimension of features, which are suitable for the authentication task. Specifially, given a training set with n data points $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(n)}, y^{(n)})\}$, where $\{\mathbf{x}^{(i)}\}$ are the feature vectors and $\{y^{(i)}\}$ are binary class labels from $\{0, 1\}$, an SVM seeks a hyperplane $f(\mathbf{x}) = \mathbf{wx} + b = 0$ to maximize the separation of the data points of the two classes, which can be done efficiently. Here class label 0 means that the feature vectors are from the registration signals of the same account as the template, while class label 1 means that the feature vectors are from the registration signals of other accounts different from the template. This is the same as the T-Fusion method.

At login, the server extracts feature vectors from the login request signal, and obtains a prediction score $\{f(\mathbf{x}) = \mathbf{wx} + b\}$. Finally, this score is compared with a predefined decision threshold. If $score < threshold$, the authentication request with signal is accepted, otherwise, it is rejected.

The aim of the feature extraction method and classifier ensemble is to achieve a better separation of signals from different classes in the feature space and maintain a stable performance in the long term. If a signal is from the account owner writing the same content as that in registration, it should have a small score because they are from similar hand movements. While signals from the other people or writing different contents should have a larger score due to the hand movement differences, which is caused by the difference in content and writing style. However, as the analysis in chapter 4 shows, the distance in sample level has uncertainties because of the minor variation of writing behavior for the same user writing the same content. Misclassification caused by such uncertainties may happen. Hence, we group local samples into segments which roughly map to the strokes of the IAHW, so as to achieve better tolerance the signal level uncertainties by comparing subset of strokes instead of individual sample. An further improvement can be made to select a special window sequence for each account using a boosting method instead of ensemble method, or using multiple templates as in chapter 4.

6.3.2 User Identification

The task of identification is a multipleclass classification problem, which should be done efficiently without linearly query the account database. Given an identification request with a signal, the server is expected to generate a set of candidate account numbers, verify each candidate, and return the best matched account number candidate, or "unidentified" if all candidates fail the verification. For this function, we use the variant of the FMHash framework explained in chapter 5, with an additional ID verification step.

As in the last step of preprocessing, a signal is normalized both in time and in amplitude to a fixed length of 256 elements in 18 channels. We use a similar neural network architecture as that in chapter 5 with slight modification. For the first two convolutional and pooling layers, we changed them to a depthwise convolutional kernel (Chollet, 2017) of size 3 on each individual channels, and a 2-by-1 max pooling on the output of the convolutional layers. Then the filters are combined to form a 64-by-144 tensor. The remaining three convolutional layers are replaced with separable convolutional layers (Chollet, 2017), and the three pooling layers use average pooling. For all convolutional layers, the input tensor is padded to maintain the same length after the convolution operation. Finally, the output tensor of the these layers is flattened, and the remaining is the same as that in chapter 5. We use 16 bit fuzzy hash code.

A deep convolutional neural network can learn the features in the hand movement signal in multiple abstraction levels (Zeiler and Fergus, 2014). For example, each neuron in the third conv-pool layer has a receptive field of 16 samples in the original signal, which is roughly corresponds to one stroke. The filters in this layer can capture different types of features in the stroke level, and we believe the neuron activation of this layer can represent the presences of different types of basic finger movement when writing a stroke segment, for example, straight motion in different directions, circular motions, and various sharp turning between adjacent strokes. Similarly, we believe the fourth and fifth conv-pool layers can detect the presence of higher level features of stroke combinations, so that the neural network reflects the hierarchy of IAHW signal characterization shown in chapter 3.2. The depthwise and separable convolution operators are introduced to reduce the total number of parameters and help learning discriminative features on each individual sensor axis at an early stage. Eventually, the fully connected layer map the aggregated features to latent vectors of 512 dimension and the last layer project and quantize these latent vectors to the Hamming space. As a major challenge of training the CNN is the limited amount of data at registration, which can lead to overfitting easily. To overcome this hurdle, we augment the five registration signals to 125 signals in the same way in chapter 5. Besides, we add another step to randomly stretch a segment both in time for each augmented signal, and then normalize the signal to 256 samples.

As mentioned previously, there is an optional step for ID verification. For ID verification, the system stores an ID template of the handwriting of the ID string generated at registration for each account, like those templates for authentication purposes. Upon an identification request, the system can compare the signal in the request with the templates of all candidate accounts obtained by the hash table searching and run a matching procedure similar to an authentication system to verify each candidate ID. The motivation is that the hashing step loses information in the IAHW, which may lead to collisions if an imposter writes the same ID string as a legitimate user, and hence, a verification step can reduce misidentification significantly.

6.3.3 Template Protection

Since the login server stores the templates, which are essentially in the same way as plain text password, there might be server storage leakage by insider attackers. Hence, the templates should protected like salted hash of passwords in a password-based login system. However, due to the inherent hand movement variation, even if the same user writes the same content, the obtained signals are not identical, and hence, cryptographic hashing of the signals can not be compared directly to verify the identity of the writer. In this section, we describe two possible approaches to protected the templates. Both of them use the FMHash as a foundation, which is a method to convert a signal to a fixed-size fuzzy binary hash code.

In the first approach, we run the following steps at registration to directly generate a key from a login request.

Step 1): We concatenate the temporally normalized signal of writing the ID string and the signal of writing the passcode in terms of sensor axes to make a new signal. That means, instead of 18 sensor axes, the concatenated signal has 36 sensor axes. Each signal is also normalized to 256 samples to facilitate the concatenation. As the ID string is unique but not secret while a passcode string is secret but not unique, concatenating the signals from these two strings make the new signal both unique and secret.

Step 2): We trained a slightly different FMHash system that can take 256-by-36 dimensional data and generate 128-bit fuzzy hash code. Then, we obtain the fuzzy hash codes for each new signal generated in the previous step using the set of registration signals. These fuzzy hash codes will be identical or only differ slightly. For each bit of them, we keep the majority vote of that bit and form an aggregated fuzzy hash code. This is the hash code for the corresponding account.

Step 3): We apply a cryptographic hash function such as SHA-256 on the aggregated fuzzy hash code XORed with a random salt to obtain a cryptographic key.

Step 4): We use this key to encrypt the passcode template, and store both the encrypted template and the salt (in plain text) into the account database.

At login, we proceed from step 1 to 3. Here in step 2, there will be only one fuzzy hash code for the login request and no aggregation is needed. In step 3, the salt is retrieved from the account database instead of randomly generated. Then we use the generated key to decrypt the template. As shown in the analysis in chapter 5, if the signals in the login request are "close enough" to those at registration, the generated fuzzy hash code will be identical and so as the key. To tolerate "fuzziness", the server can keep multiple hash codes in step 2, and hence, multiple keys as well as multiple copies of encrypted templates in step 3 with an account, such that if the derived key at login can decrypt any of the copies of encrypted templates, the authentication procedure can proceed. This number of multiple hash codes can be set as a system-wide or per-account parameter. Typically, we can tolerate 1 bit or 2 bits as shown in chapter 5.

The second approach is similar to the first approach but it binds a key to the account instead of directly generating one from the signal. At registration, the first two steps of the previous direct key generation approach are proceeded. However, in the third approach, a special random key is generated and bound to the fuzzy hash code through fuzzy commitment (Juels and Wattenberg, 1999). The last step is the same as the direct key generation approach. At login, we proceed with the first step, and in the third step, the key can be decommited by a hash code generated from the signals in the login request, if these signals are "close enough" to those at registration. The amount of "fuzziness" tolerance can be controlled by the Error Correcting Code (ECC) scheme used in the fuzzy commitment.

A brief explanation of key binding using fuzzy commitment in the context of our usage is provided as follows, and also illustrated in Figure 6.2. Consider a user wants to bind a secret key $\mathbf{k}$ to a fuzzy hash $\mathbf{x}$, where both $\mathbf{k}$ and $\mathbf{x}$ are binary strings with the same length of n bits. For example, in Figure 6.2, they are all six bits. This illustration shows the Hamming space in 2D, where the x-axis is the first three bits and the y-axis is the next three bits. Both axes use Grey code instead of ordinary binary code to show the Hamming distance in a way similar to Euclidean distance. Assuming F is a field of n bits. Let C be the set of codewords in F using some ECC scheme. Here, in Figure 6.2, F is the grid structure where each individual one is indexed by a six-bit string composed by the coordinates of that grid. C is the set of shaded grids.

At registration, for each account, given the generated fuzzy hash code of the account as $\mathbf{x}$, the system selects a codeword $\mathbf{k}$ uniformly at random from C, compute $\mathbf{d} = \mathbf{k} - \mathbf{x}$, and then store $\mathbf{d}$ and the cryptographic hash of $\mathbf{k}$ (such as SHA-256) in the account database. Here the subtraction operation is implemented as binary XOR. Because $\mathbf{k}$ is randomly selected, and $\mathbf{d}$ is a "distance" between $\mathbf{k}$ and $\mathbf{x}$, given a certain registered account with a determined $\mathbf{x}$, $\mathbf{d}$ has the same level of randomness as $\mathbf{k}$. This step commits $\mathbf{k}$ with some $\mathbf{x}$, as shown in Figure 6.2 (a). Note that the system does not want to reveal either $\mathbf{k}$ or $\mathbf{x}$, and the templates are encrypted using $\mathbf{k}$ as the secret key.

At login time, given another fuzzy hash code $\mathbf{x}'$, the system compute $\mathbf{y} = \mathbf{d} + \mathbf{x}'$. Here the addition operation is also implemented as binary XOR. Then, it uses the ECC scheme to find the proper codeword $\mathbf{k}'$ for $\mathbf{y}$. In this step, if the fuzzy hash code $\mathbf{x}'$ generated by the login request is "close enough" to the account hash code $\mathbf{x}$ in the registration, the recovered codeword $\mathbf{k}'$ after ECC should be the same as $\mathbf{k}$. Hence, the level of fuzziness tolerance is managed by this ECC scheme. After that, the system compute the cryptographic hash of $\mathbf{k}'$ and compare it with the stored cryptographic hash of $\mathbf{k}$. If they are the same, it means that the fuzzy hash codes $\mathbf{x}'$ and $\mathbf{x}$ are "close enough", *i.e.*, they can be considered as generated by the same user writing the same content. This also means that $\mathbf{k} = \mathbf{y}$. Finally, the templates are decrypted using $\mathbf{k}$ as the secret key.

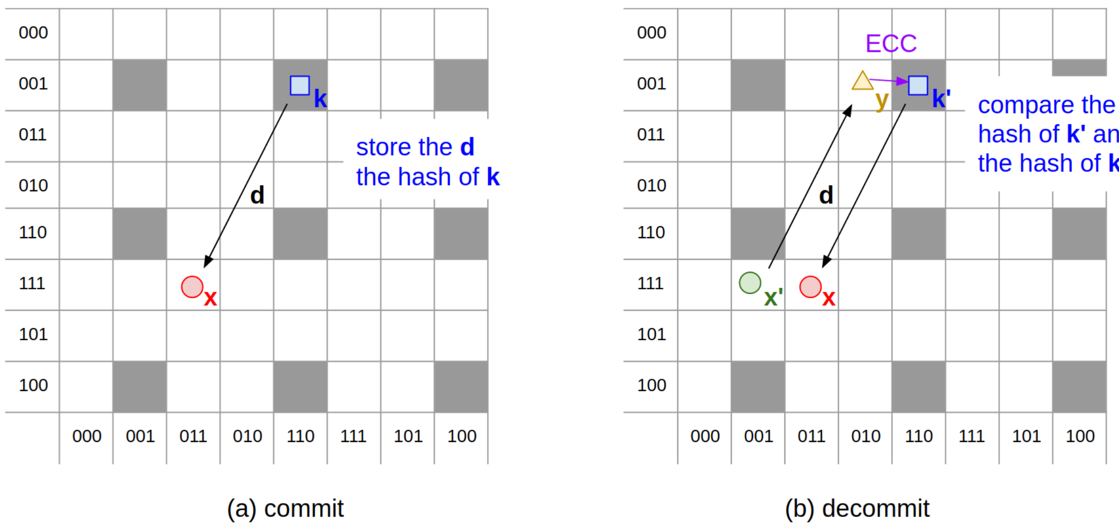


Figure 6.2: An Illustration of Key Binding Using Fuzzy Commitment

For both approaches, the security are guaranteed by the cryptographic tools, *i.e.*, SHA-256 or fuzzy commitment. However, the actual "strength" of the key depends on the fuzzy hash code and the level of tolerance of fuzziness. On the one hand, the fuzzy hash code should be long enough to create a large key space to prevent brute-force attacks. This can be achieved by training multiple deep hash neural network to generate multiple segments of 64-bit hash code and concatenate them to a longer hash code. Due to the stochastic nature of the initialization of the deep hash neural network, these networks are unrelated. However, the fuzziness will also grow with the hash code length. On the other hand, the fuzzy hash code should be spread pseudo-randomly in the Hamming space to prevent systematic guessing attacks. Although the fuzziness tolerance scheme can reduce the "strength" of the key, the amount of tolerance can be controlled as a trade-off between security and usability.

6.3.4 Cost of Computing and Storage

We implemented our prototype system in Python with sklearn library and TensorFlow Abadi *et al.* (2016). Preprocessing a single signal cost about 25 ms for the glove device (excluding any steps running on the device) and 100 ms for the camera device. The filtering step is the bottleneck because the complexity of Fast Fourier Transform (FFT) is O(nlogn) and the complexity of all other preprocessing steps are essentially O(n). For authentication, generate template cost 2.1 ms, and training the SVM cost 30 ms for each account; classification of each signal cost less than 1 ms, which is negligible compared to the time for writing the string. The time consumption measured here does not contain loading the data and model from disk to the main memory, because the disk speed varies significantly due to the page cache in the main memory and our dataset is small enough to be fully fit in the cache, which is quite different when used in real world scenarios. This measurement is conducted with a single threaded program on a desktop computer with Intel Xeon E3-1230 CPU (quad-core 3.2GHz, 8M cache) and 32 GB main memory. For identification, training the deep CNN from the scratch takes about 10 minutes on a Nvida GTX 1080 Ti GPU. Obtaining candidate account IDs using the CNN costs less than 1 ms on the GPU and ID verification takes less than 1 ms on the CPU. The space cost for template storage and the amount of data needed to be transferred between the server and the client is proportional to the signal length. If each sensor axis of a sample is represented in single precision floating point number (four bytes), the average space cost of a signal is about 18 KB with our datasets for both devices, considering each preprocessed signal is a 256-by-18 single precision floating point array. If all parameters are represented in single precision floating point number, storing the SVM classifiers also costs about 18 KB per account on average. Storing the CNN itself requires around 6 MB because of the 1.4 million weights and biases parameters.

6.4 Security Analysis

In this section, we first set up a security model for the FMCode framework, and analyze it frome various security aspects.

6.4.1 Security Model

FMCode has the same security assumptions as existing gesture-based authentication and identification systems as follows: (1) the device on the user side is secure(*i.e.,* no sniffing backdoor); (2) the memory and computing infrastructure of the authentication server is secure (*i.e.,* the server program will not leak the decrypted template to another program); and (3) the communication channel between the user and the server is secure (*i.e,* no man-in-the-middle attacks). These security assumptions are also similar to traditional biometrics for device unlock and password-based online login system. Based on the assumptions, we are mainly interested in the following normal scenario and attacks on the user side, as listed below:

1) Legitimate user login, *i.e.,* the account owner writes the correct ID string and passcode string to login. This type of login request is labeled as the "**true-user**" class. This corresponds to the signals labeled as "same" in chapter 4.2 and "same account" in chapter 5.5.

2) Random guessing attack, *i.e.,* the attacker tries to enter a user's account by guessing an ID string and a passcode string and signs it on the same gesture interface, without any knowledge of the content of the two strings. This type of login request is labeled as the "**guessing**" class. This corresponds to the signals labeled as "diff" in chapter 4.2 and "different account" in chapter 5.5.

3) ID and passcode collision or spoofing attack with semantic leakage, *i.e.,* the attacker tries to enter a user's account by writing the correct ID string and passcode string of the target user in the air through the same gesture interface, with only the knowledge of the content of the passcode. This is similar to password collision or password leakage. This type of login request is labeled as the "**collision**" class. This corresponds to the signals labeled as "collision" in chapter 4.2.

4) Spoofing attack with both semantic leakage and visual leakage, which is similar to attack scenario (2), but the attacker has seen the legitimate user writing the correct ID string or passcode string. Moreover, the attacker may recorded the hand movements with a camera and watch them carefully multiple times. We use the pairs of signals from the "ID-passcode-spoofing" dataset to generate this type of login requests. This type of login request is labeled as the "**spoofing**" class.

For the scenarios of collision and spoofing attack, we assume that the attack source is a human attacker, and the attacker's goal is to sign into the victim's account or be identified as the victim. If the attack is successful, the account owner may suffer from loss of the account or leakage of private information. Though it is generally considered that ID is not a secret, in many cases, if the attacker is wrongly identified as the victim, he or she may launch further attacks *e.g.,* phishing other sensitive personal information of the victim.

Based on this security model, we set up the experiments with the following protocol, which is similar to that in chapter 4.7.

A. Registration: We follow the registration process to create all the accounts using the signals in the "ID-passcode" dataset, construct the templates, and train the models for each account. All five registration pairs of signals of each account in the dataset are used to construct the template and train the model.

B. Legitimate user login experiment: For each account, we use each of the five pairs of testing signals of the same account in the "ID-passcode" dataset as a login request to this account.

C. Random guessing attack experiment: For each account, we use each of the five pairs of testing signals of other accounts in the "ID-passcode" dataset as a random guessing attack request to this account.

D. Collision or semantic spoofing attack experiment: We use each of the five pairs of spoofing signals in the "ID-passcode-collision" dataset as a collision or semantic spoofing attack.

E. Visual spoofing attack experiment: We use each of the five pairs of spoofing signals in the "ID-passcode-spoofing" dataset as a visual spoofing attack.

6.4.2 Analysis of User Authentication

For authentication, we define the false reject, false accept, successful collision in the same way as those in chapter 4.7. We also defined successful spoofing in a similar way as successful collision using the visual spoofing attack experiment results. The

Table 6.1: Empirical Results of User Authentication with the Camera Device

method	EER	FRR@ FAR1e-3	FRR@ FAR1e-4	FRR@ ZeroFAR	FAR@ FRR1%
FMCode (guessing)	0.1	0.1	0.8	1.1	0.0
FMCode (collision)	2.4	8.9	12.7	14.8	7.7
FMCode (spoofing)	2.9	26.2	33.1	33.1	12.1
DTW (guessing)	0.7	3.1	5.4	9.2	0.6
DTW (collision)	4.4	15.5	33.4	33.4	19.6
DTW (spoofing)	4.7	59.3	78.3	78.3	24.5

evaluation metrics, *i.e.*, False Reject Rate (FRR), False Accept Rate (FAR), and Equal Error Rate (EER) are defined in the same way as those in chapter 4.7. The evaluation results are shown in Table 6.4.2 and Table 6.4.2. Here, we use the term FRR@FAR1e-3 and the term FRR@FAR1e-4 denotes the smallest FRR when FAR is 10^{-3} and 10^{-4}, respectively. In chapter 4.7 they are called FAR1K and FAR10K. We use FRR@ZeroFAR to denote the smallest FRR when FAR is zero, and we use FAR@FRR1% to denote the smallest FAR when FRR is 1%. These terminologies are better understandable for general readers instead of readers with a background in biometrics. It should be noted that visual spoofing attacks are not available for every account and metrics related to visual spoofing attacks only concerns those accounts with corresponding data from the "ID-passcode-spoofing" dataset.

Since both FRR and FAR can change with the decision threshold, we vary it to obtain all possible FRR and FAR pairs and show them as the Receiver Operating Characteristic (ROC) curve in Figure 6.3. The score of each login request does not depend on the decision threshold, and collectively they are shown in the histogram for different classes in Figure 6.4. For comparison with different methods, we use the

Table 6.2: Empirical Results of User Authentication with the Glove Device

method	EER	FRR@ FAR1e-3	FRR@ FAR1e-4	FRR@ ZeroFAR	FAR@ FRR1%
FMCode (guessing)	0.1	0.2	0.4	0.4	0.0
FMCode (collision)	1.2	1.6	4.3	4.6	1.9
FMCode (spoofing)	1.3	4.9	22.1	22.1	3.5
DTW (guessing)	0.3	0.6	0.9	3.5	0.0
DTW (collision)	1.7	7.6	20.6	20.6	3.5
DTW (spoofing)	2.3	19.6	52.7	52.7	7.2

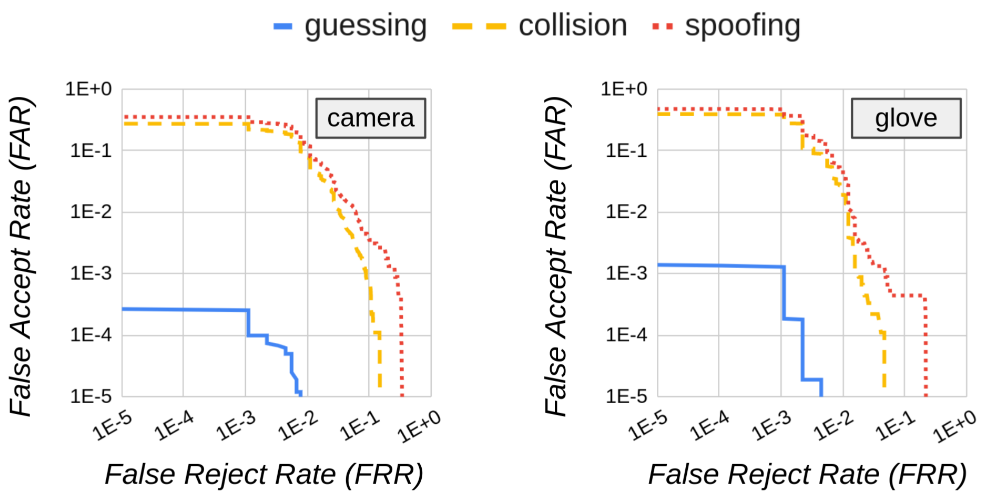


Figure 6.3: Receiver Operating Characteristic (ROC)

EER since it is a single number that captures the general performance. However, for practical usage, typically a realistic decision threshold is set at the smallest value such that the corresponding FAR of the random guessing attacks is at most 10^{-4} and the FRR is between 90% to 99%. The rationale is that using this decision threshold, an attacker without the knowledge of the passcode needs to make at least 10^4 times guessing to break the passcode, which is comparable to a 4-digit PIN for device unlocking. For security considerations, this decision threshold can be set to a lower

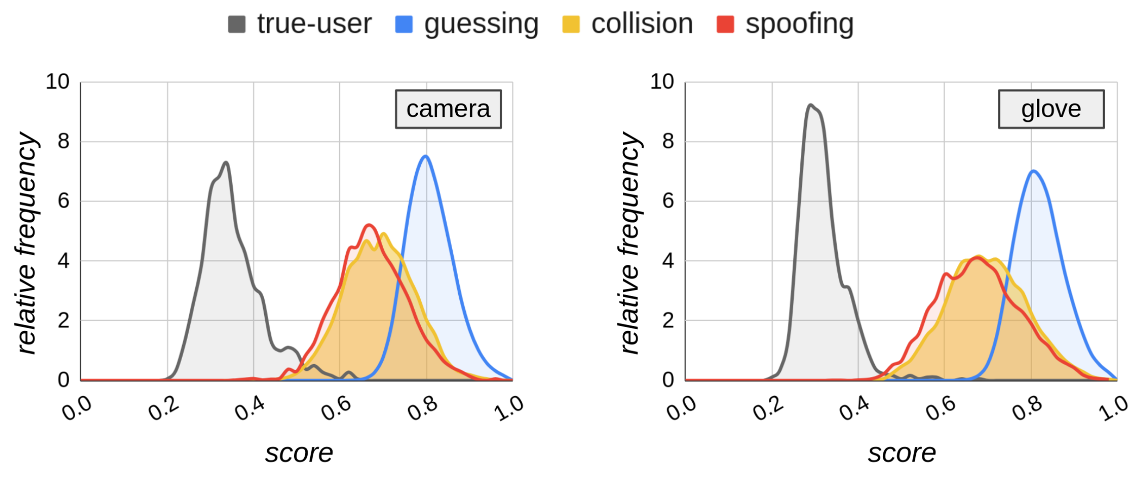


Figure 6.4: Authentication Score Histogram for Different Classes

value at a slightly tolerable usability cost of rejecting legitimate user a little more frequently, and usually this false reject rate is at most 10%.

We also apply the legitimate user login experiment and the random guessing experiment on the data in each session of the "ID-passcode-persistence" dataset. The score variation among the ten sessions is shown in Figure 6.5 (left) and Figure 6.6 (left). In these two figures, each column of corresponds to the results in one session, which is similar to the score histogram but the scores of login requests from all 40 accounts are plot as scattered points from left to right across the width of the column. The red lines represents a decision threshold, which is set to 0.613 for the camera device and 0.601 for the glove device, such that the corresponding single session performance metrics are $FRR \approx 1\%$, $FAR_{guessing} \approx 10^{-5}$, $FAR_{collision} \approx 12\%$, and $FAR_{spoofing} \approx 20\%$.

There are two observations. On the one hand, there is an increase of the scores of the login requests of legitimate users over time if the templates and the classifiers are not updated, although the extent of increase slows down after the fourth session. On the other hand, the scores of the login requests of the random guessing attacks do not have noticeable changes. These two factors make the system harder to distinguish legitimate users from attackers, which eventually leads to more frequent false rejection of legitimate users if the decision threshold is not changed.

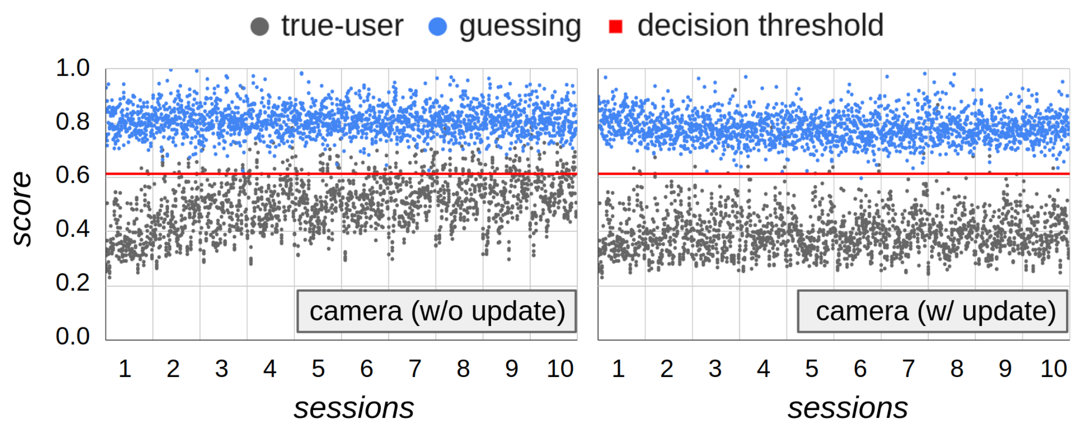


Figure 6.5: Authentication Score Among Ten Sessions, with the Camera Device

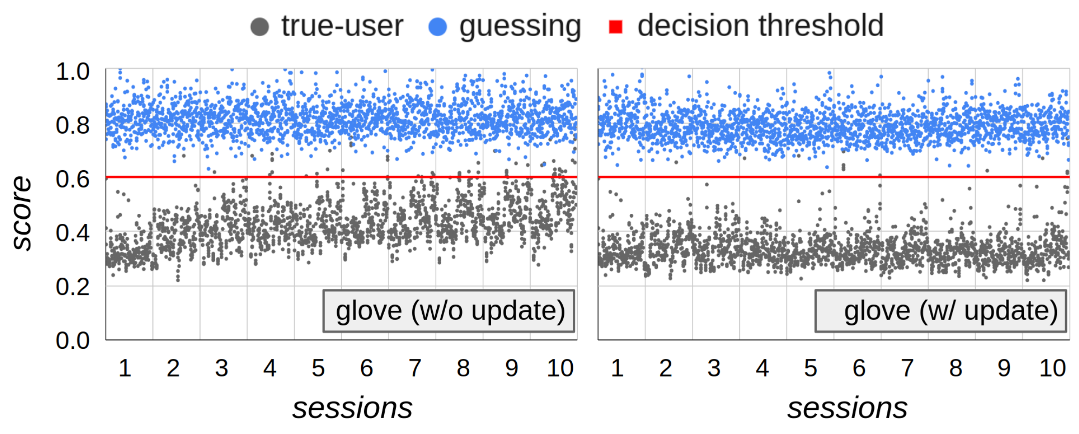


Figure 6.6: Authentication Score Among Ten Sessions, with the Glove Device

There are two reasons. First, there are inherent variations of user behaviors, which leads to differences in signals even for the same user writing the same string. For example, most users write from left to right using the index finger, but this left-to-right movements of hand usually do not have a fixed angle relative to the pointing direction of the index finger when writing the same content every time in the air. However, the xyz-axes are determined by the pointing direction in the signal preprocessing steps, which leads to variations in the signals. This is more significant for the camera device because it directly tracks position and orientation. Second, the authentication algorithm is not perfect. There are limited amount of signals obtained at registration, which make the system difficult to capture or predict the long-term in-air-handwriting behavior variation. Meanwhile, the classification are designed to utilize signal-level differences directly for more effective detection of spoofing attacks from legitimate login requests, which also limits its generalization capability under long-term variation.

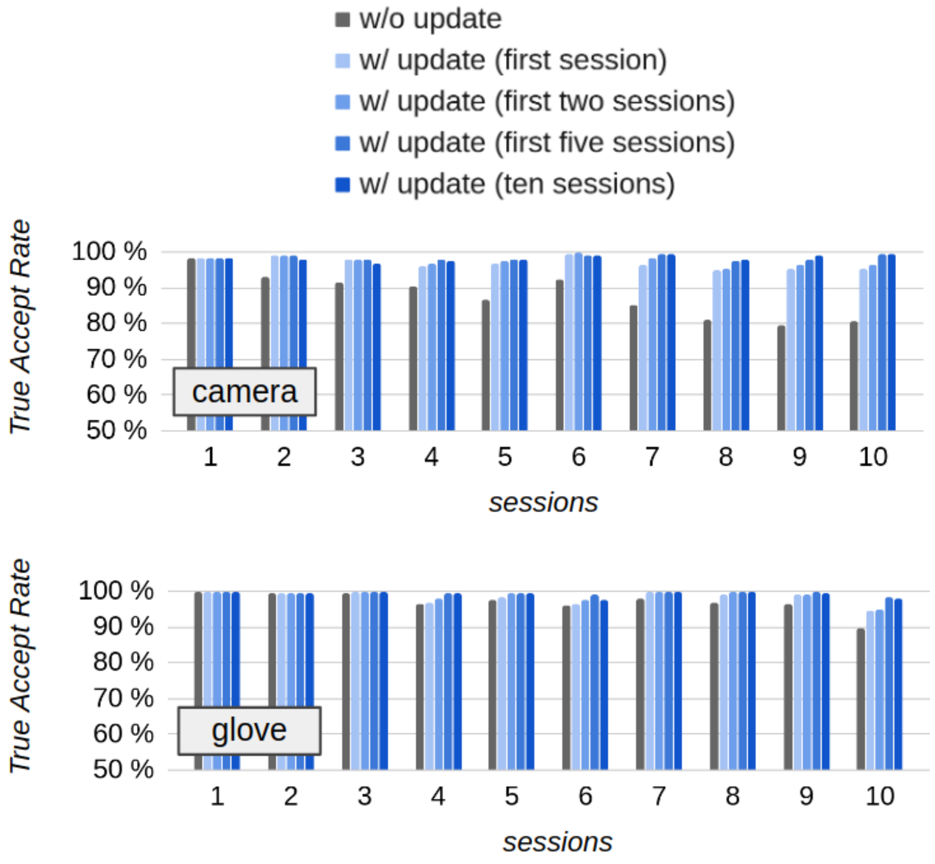


Figure 6.7: Variation of True Acceptance Rate (TAR, *i.e.*, 1 - FRR) Among Ten Sessions

To accommodate these variations, one solution is asking the users to update the templates and retrain the classifier. The updated template is a linear combination of an old template $\hat{\mathbf{t}}$ and the updating signal $\mathbf{s}$, *i.e.,* $\hat{\mathbf{t}} \leftarrow (1-\lambda)\hat{\mathbf{t}} + \lambda\mathbf{s}$, where λ is the update factor set to 0.2 in our experiments. After the updating, the classifier is retrained using the old templates as well as the updating signals as positive data points, and using the templates of other accounts as negative data points. In Figure 6.7, we show the performance change with four different scenarios: (a) only do the updating in the first session while not changing the templates and the classifier after that; (b) similar to (a) but do the updating in both the first session and the second session; (c) similar to (a) but do the updating in the first five sessions; (d) do the updating in all ten sessions. The results with updating in all ten sessions are also shown in Figure 6.5 (right) and Figure 6.6 (right). From these results, we can observe that long term performance can be improved by the updating, even only updating at the first login immediately after registration can boost the true accept rate to above 95% for all remaining sessions.

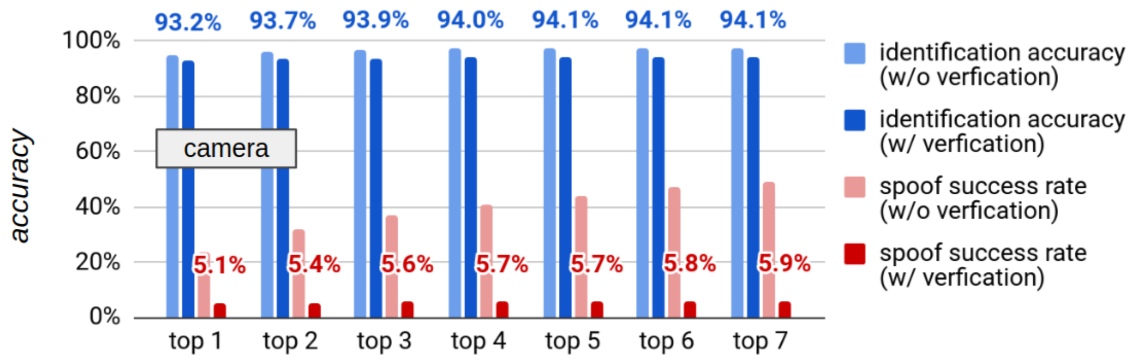
6.4.3 Analysis of User Identification

For identification, we set up two scenarios, *i.e.*, open-set and closed-set, as mentioned in chapter 3.8. For the open-set scenario, we retrieve the top-k nearest neighbor as candidate accounts in the Hamming space within four bit fuzziness tolerance. The ID verification is applied on the candidate account set and it will remove any accounts that the verification score is below a threshold. For the closed-set scenario, we retrieve all candidate accounts within a certain fuzziness tolerance and use the ID verification step to select the top-k candidate accounts using the verification score. The initial fuzziness tolerance is four bits. If there are not enough candidate accounts to select the top-k within this fuzziness tolerance level (which rarely happens), the tolerance is increased one bit at a time until the top-k candidate accounts are selected. If a login request in the legitimate user login experiment retrieves the correct account ID in the candidate account set, it is a correct identification, and otherwise, it is an incorrect identification. We define identification accuracy as the number of correct identification divided by the total number of login requests in this experiment. Similarly, if a login request in the visual spoofing attack experiment retrieves the correct account ID, it is a successful spoof. We also define the spoof success rate as the number of successful spoof divided by the total number of spoofing attacks in this experiment. The results of user identification without considering collision and spoofing is shown in Table 6.4.3.

Table 6.3: Empirical Results for User Identification

	camera		glove	
	top-1 accuracy	top-5 accuracy	top-1 accuracy	top-5 accuracy
open-set	93.2%	94.5%	95.7%	96.7%
closed-set	96.7%	98.9%	97.9%	98.8%

It is commonly believed that ID is not a secret and the identity verification should not be too strict to hurt usability. Thus, we choose the threshold for the identity verification such that roughly 5% of the signals from the legitimate users are rejected (*i.e.*, a threshold at $FRR \approx 5\%$). We vary k from 1 to 7 and show the results in Figure 6.8. In general, increasing the number of candidates helps identification accuracy, at a marginal cost of slightly increased spoof success rate. However, if identity verification is skipped, spoof success rate will have a significant increase, which renders the system vulnerable or even useless. The main cause is that the network learns features for distinguishing difference stroke combinations instead of distinguishing fine differences in the signal segments of the same stroke. Also in practice, there is no spoofing data available at registration time to train the neural network. Essentially, the network also serves as an index for all accounts in the database, which locates the probable accounts given a signal instead of search it exhaustively. With exhaustive search, our prototype system can achieve around 99.5% accuracy with both types of devices. However, it takes more than one second to exhaustive search on a database containing 180 accounts. The time consumption is mainly caused by accessing the stored template as well as aligning the signal, and it will grow linearly with the number of accounts in the database.



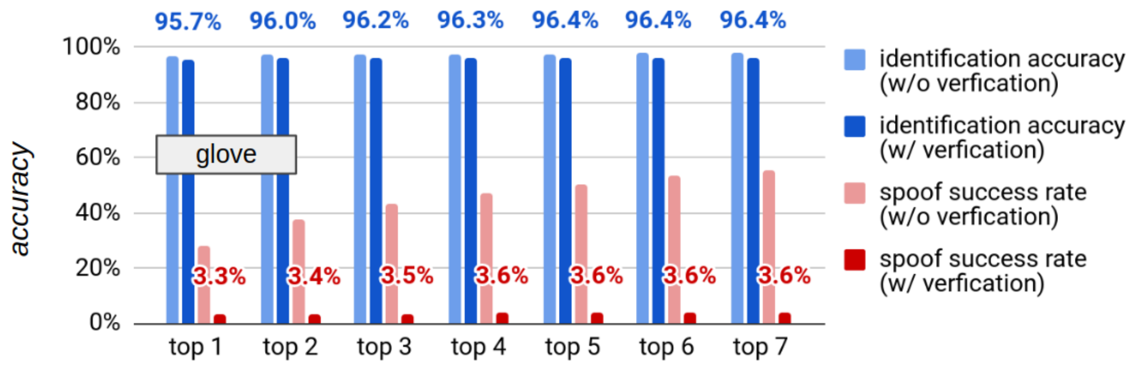


Figure 6.8: Identification Results Considering Spoofing Attacks, with and Without ID Verification

We also apply the legitimate user login experiment and the random guessing experiment on the data in each session of the "ID-passcode-persistence" dataset. The variation of identification accuracy among ten sessions is shown in Figure 6.9. It can be observed that there is slight performance degradation with time, which is mainly caused by the ID verification process for the same reason as that in the authentication results. Similarly, the accuracy can be improved significantly if the templates and the SVM classifiers for the ID verification process are both updated with the new signals at the end of the session. We believe that for some users merely 5 signals at registration cannot fully capture the uncertainty of the writing behavior, even with our data augmentation methods. In practice, typing a password can always be employed as a fallback. On the smartphone, if the user does not unlock it immediately with fingerprint or face, the smartphone will ask the user to type a password. If the password passes, it will update the fingerprint or face template accordingly. Such a strategy can also be utilized in our framework since showing a virtual keyboard to type a password can always be an backup option, though it is inconvenient.

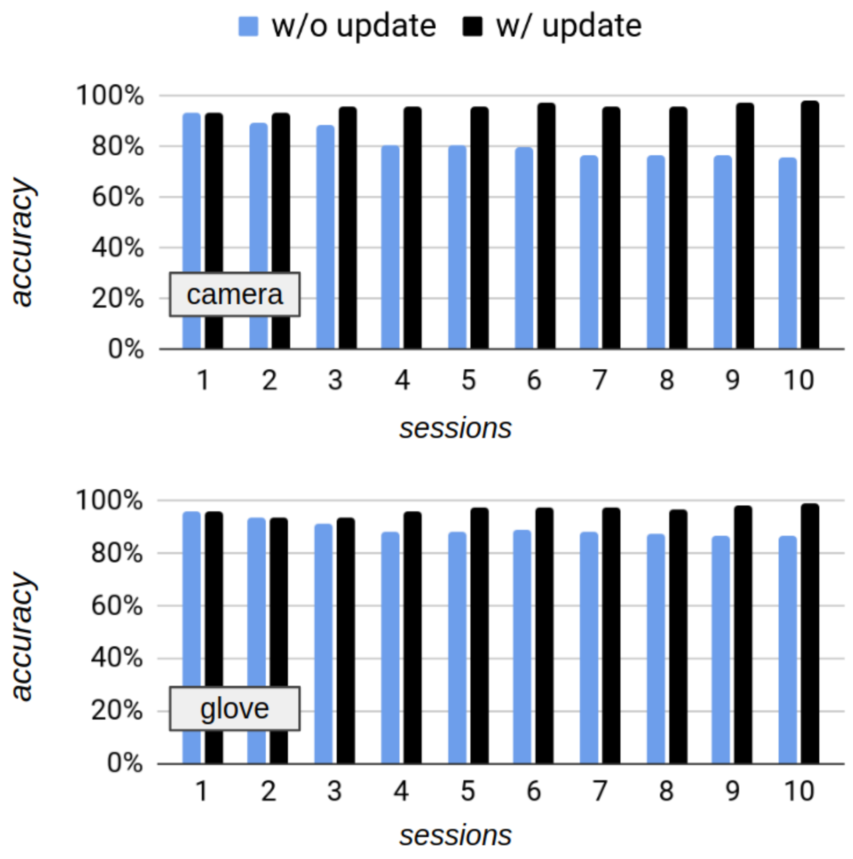


Figure 6.9: Variation of Identification Accuracy Among Ten Sessions

6.4.4 Comparative Study

A comparison to existing works which also use in-air-handwriting is shown in the Table 6.4. The major characteristics that differentiate our framework from them are as follows. First we use a data driven method by designing features and utilizing machine learning models, instead of crafting algorithms calculating a matching distance for authentication. Second, we avoid exhaustively searching and comparing the whole account database for user identification. Third, we evaluate performance of our framework under active spoofing attacks and with a time span of near a month, which is usually omitted by existing works. Fourth, our system has a significant performance improvement on a dataset with reasonable size. A more complete list and comparison of related work in this area is provided in chapter 2.1.

Table 6.4: Comparison to Existing Works

Ref.	dataset size	EER (w/o spoof)	EER (w/ spoof)	Identification Accuracy	Device
FMCode (camera)	180	0.1%	2.9%	93.2% (96.7%)	Leap Motion
FMCode (glove)	180	0.1%	1.3%	95.7% (97.9%)	data glove
Liu *et al.* (2009)	25	∼3%	∼10%	88 ∼ 98.4%	Wii remote
Bailador *et al.* (2011)	96	1.8%∼2.1%	∼5%	N/A	smartphone
Bashir *et al.* (2009)	40	∼1.8%	N/A	98.5%	digital pen
Chan *et al.* (2015)	16	0.8%	N/A	98%	Leap Motion
Tian *et al.* (2013)	18	∼2%	N/A	N/A	Kinect

In Table 6.4.4 we present a comparison of FMCode with password and biometrics based system. The results shown here are obtained from different publications with various datasets, which merely show limited information as an intuition about their performance, instead of the performance with serious and strongly supervised evaluation. First we show the classification accuracy in terms of EER. Here FM-Code is comparable to fingerprint (on FVC2004 (Cappelli *et al.*, 2006) among all the datasets), face, iris, and signature. Comparing to biometrics, a considerable portion of discrimination capability comes from the large content space of the in-air-handwriting. Next we show the equivalent key space in number of bits. For password used in device unlock, the commonly used 4-digit password (default setting on most smartphone) are considered, and for biometrics, the equivalent key space is defined by $log_2(1/FAR)$ (O'Gorman, 2003). For FMCode we calculate key space with the corresponding FAR by setting the decision threshold at a place where the true user has 95% successful login rate (*i.e.,* 5% FRR). The results show that FMCode is also comparable to password based login and authentication system. Due to the limited amount of data, we cannot achieve an FAR resolution lower than 5×10^{-6}, *i.e.,* more than 17.6 bit key space. For the glove device, at the 5% FRR decision threshold, the FAR is already 0 but we can only conclude that the equivalent key space is larger than 17.6 bits. In practice, recommended password key space is between 28 to 128 bits (Crocker and Schiller, 2011) while web users typically choose passwords with 20 to 90 bits key space and on average it is 40 to 41 bits (Florencio and Herley, 2007). However such large key space is underutilized because it is well known that people are not good at creating and memorizing strong password, and the actually entropy is much less than the whole key space (Ma *et al.*, 2010). Moreover, since password must contain letters that can be typed on keyboard, efficient password guessing strategies such as dictionary attack further weaken the calculated password quality in number of bits.

Table 6.5: Comparison to Password and Biometrics

method	EER	Key Space (bit)
FMCode (camera)	0.1% ∼ 2.9%	∼17.6
FMCode (glove)	0.1% ∼ 1.3%	∼17.6
Password (online)	N/A	20∼90 (Florencio *et al.*, 2007)
Password (device)	N/A	13.3 (O'Gorman, 2003)
Fingerprint	0.28%∼2% (Cappelli *et al.*, 2006)	N/A
Face Recognition	2.6% ∼8.6% (Parkhi *et al.*, 2015)	N/A
Iris	0.11% (O'Gorman, 2003)	19.9 (O'Gorman, 2003)
Signature	1% ∼3% (Impedovo *et al.*, 2008)	N/A

6.4.5 Discussions

There are other potential attacks on our authentication system if some of our security assumptions do not hold. For example, the attacker may be able to access the server's storage to steal the templates. Due to the inherent fuzziness in the authentication step, traditional hash is not able to protect the templates like the hashed password. One possible solution that we are working on is to adapt the deep CNN to generate a fuzzy hash and further a set of keys for each account using the signals at registration. The templates can be encrypted by each of the key to form a pool. At login time, the same neural network is applied on the signal in the login request to obtain a fuzzy hash and a key. It is highly possible that this key can decrypt an encrypted templates in the pool, if the signal is generated by the legitimate user performing the correct FMCode. After the template decryption, the authentication procedure can continue. If the templates can be successfully decrypted by the generated key from the signal in the login request, we can not conclude that the user is authenticated. As we have shown in the identification results, the ID is easier to spoof without a verification step, thus an impostor with the knowledge of the ID and the passcode may also be able to derive an acceptable key. However, the attacker that steals the templates does not have such knowledge.

Another possible threat is man-in-the-middle attacks, where the attacker intercepts and records the messages between the server and the client. We believe this is not critical because machine-to-machine authentication and message encryption can be used first to allow the client and the server to communicate securely. Existing technologies such as SSL/TLS and public key infrastructure (PKI) over the Internet can be leveraged to build a secure communication channel before user login, similar to the case of password-based login on most websites over the Internet nowadays.

On the user side, record and replay attacks must also be handled. The authentication request require to present a multi-dimensional signal about the physical states of the hand during the writing. In a preliminary experiment, we found that it is difficult to synthesis such a signal from a video recorded by monolens camera or depth camera placed 1 meter away from the user. The main reason is the drastic spatial resolution drop with distance, which cause limited hand skeleton estimation accuracy. A secondary reason is the posture uncertainty of the user and the motion blur due to insufficient frame rate of the camera. If the camera is close to the user, the user might get alerted, just like if someone within a proximity is watching a user typing his or her password, the user will be alerted and stop typing. More details about this type of attack will be presented in future work.

6.5 Usability Analysis

In this section, we analyze the usability of the FMCode framework.

6.5.1 User Evaluation

We investigated the usability of our framework by taking questionnaire from 100 users with the experience of both the glove device and the camera device. These users participated in our data collection of the "ID-passcode" dataset. Each user is asked to submit a questionnaire.

First, the users evaluate various aspects of our IAHW based login framework by putting down a score from 1 (strongly disagree) to 5 (strongly agree) on the following statements. In these statements, "the user" refers to the user taking this questionnaire. The users are required to provide these scores based on their subjective feeling.

A. Since the passcode string is created by the user, the passcode content is easy to memorize.

B. From the perspective of a stranger, the passcode content created by the user is difficult to guess.

C. The passcode content is difficult to leak visually, *i.e.*, it is difficult to guess the passcode content based on the IAHW hand movements.

D. From the perspective of an imposter, it is difficult to mimic the IAHW hand movements after seen them.

E. It is easy to learn and register with our IAHW-based login framework.

F. It is fast to login through our IAHW-based login framework.

G. It is easy to update or revoke the passcode if the user forgets it or if the user wants to change it.

H. The user prefers to use it as the primary login method through a gesture input interface.

The results are shown in Fig. 6.10. Overall, users feel positive to our framework, although the usability is not exceptionally good in a few aspects.

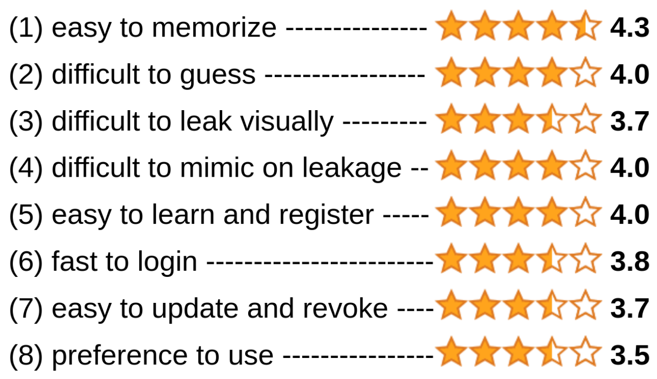


Figure 6.10: Results of Subjective User Evaluation on Our Login Framework

Second, we ask the user to compare our framework with the password based systems and biometrics including fingerprint and face. The comparison is made on three aspects, the easiness of usage, login speed, and security based on the feeling of users. The user has three options: (a) our framework is better, (b) both methods are the same, or it is difficult to decide which one is better or worse, (c) our framework is worse. The results are shown in Fig. 6.11. We can see that the user has a mixed attitude on the usability compared to traditional password, but clearly FMCode can not compete biometrics like fingerprint and face for device unlock. However, the majority of the users feel that FMCode is more secure than traditional password, and more than half of them feel it is more secure than fingerprint and face.

Third, we ask the following questions:

1) Compared to password, our framework considers both handwriting content and handwriting style, *i.e.*, combining a password and a biometric trait. Is this feature important for a login system?

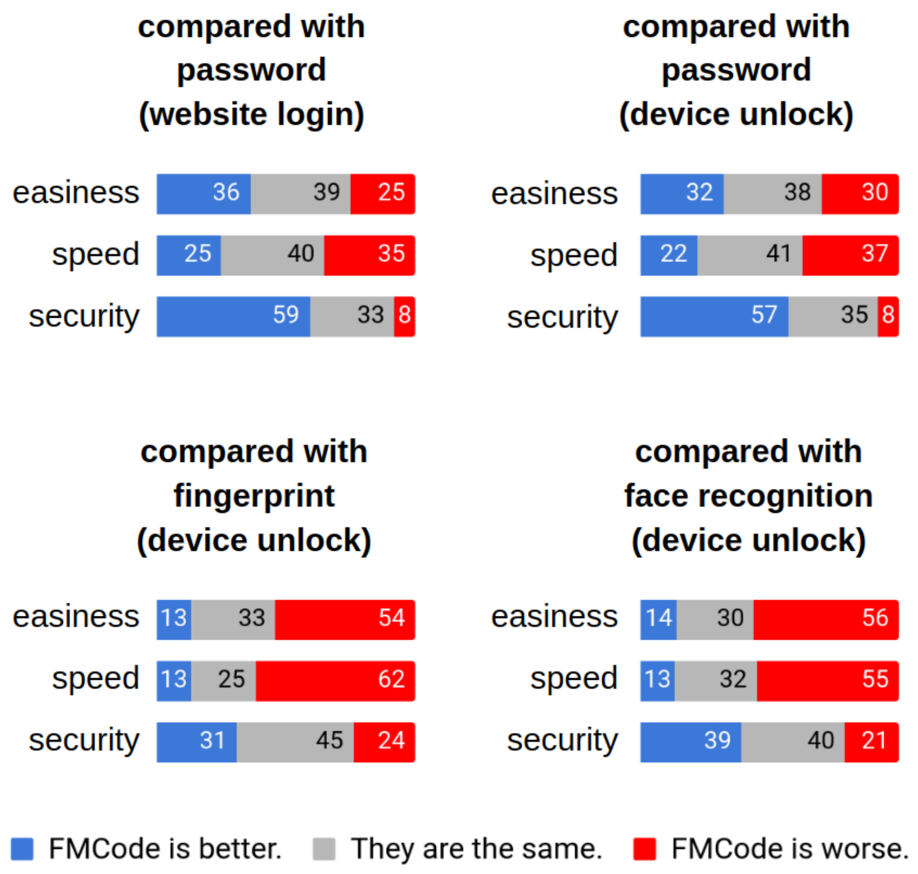


Figure 6.11: Results of Subjective User Evaluation on Our Login Framework Compared With Other Login Methods

2) Compared to biometrics, our framework allows change and revoke the in-air-handwriting passcode, which is unlinked to personal identity. Is this feature important for a login system?

Among the surveyed users, 89% and 82% of them answer "important" for the first and second features respectively. Combined with the previous results, we can conclude that FMCode does not intend to replace existing password-based solution or biometrics. Instead, due to its unique features that passwords and biometrics lack, we believe that FMCode is suitable in scenarios where such features matter and where passwords and biometrics are not applicable, for example, login over gesture interface on VR headset or in operating theater.

At last, we ask the user which type of device is preferred between a wearable device and a contactless device for hand motion tracking. 21% of the users choose the wearable device and the other 79% choose the contactless device.

6.5.2 Qualitative Evaluation

We also evaluated the usability, deployability and security features of FMCode qualitatively using the a well-known set of criteria (Bonneau *et al.*, 2012), and the result is shown in Table 6.5.2, Table 6.5.2, and Table 6.5.2. We added two aspects in deployability, "configurable" and "developer friendly" mentioned in an exiting survey work (Clark and Lindqvist, 2015). Each usability, deployability and security item is evaluated by whether our FMCode method possess the characteristics, and a plus / minus sign means that our method is better / worse than password. We explain a few items here but the readers are recommended to refer to the original study (Bonneau *et al.*, 2012) and survey work (Clark and Lindqvist, 2015) for the definition of these characteristics. In general, compared with password and biometrics, FMCode achieves nearly all their usability and deployability characteristics, and it contains the security advantages from both password and biometrics.

On the usability side, nothing to carry means the user does not to present a physical item such as a card. Though our prototype system uses a glove, the glove is treated as a general gesture input device, just like password does not require the user to bring a keyboard everywhere. Arguably, FMCode has less memory burden because the discrimination capability comes from both the content and the writing style, instead of just the combination of characters in password, so memorable FMCode is not necessarily weak any more. However, there might be potentially more frequent rejection of legitimate user login than password because the internal fluctuation in the stability of finger motion.

Table 6.6: Qualitative Evaluation of Usability.

Usability	FMCode
Memory effortless	Maybe (+)
Scalable for users	Maybe
Nothing to carry	Yes
Physically effortless	No
Easy to learn	Yes
Efficient to use	Yes
Infrequent errors	Maybe (-)
Easy to recovery	Yes

Table 6.7: Qualitative Evaluation of Deployability.

Deployability	FMCode
Accessible	Yes
Scalable	Yes
Server compatible	Maybe (-)
Browser compatible	Maybe (-)
Mature	No (-)
Non-proprietary	Maybe (-)
Configurable	Yes
Developer friendly	Yes

Table 6.8: Qualitative Evaluation of Security.

Security	FMCode
Resilient to physical observation	Maybe (+)
Resilient to targeted impersonation	Maybe (+)
Resilient to throttled guessing	Maybe
Resilient to unthrottled guessing	No
Resilient to theft	Yes (+)
No trusted third party required	Maybe (-)
Requiring explicit consent	Yes
Unlinkable	Yes

On the deployability side, since the server only needs to store the template as a secret, FMCode is similar as those behavior biometrics without using special devices (*e.g.,* keyboard or mouse dynamics), thus it can be compatible with most server with small software modification. FMCode can be supported by browsers on devices with gesture input interface, where the FMCode input of a user can be treated as a very long password and sent to the server over the Internet, similar as the case of logining in a website using password.

On the security side, FMCode can withstand spoof under semantic or visual disclosure and targeted impersonation to a certain extent, which is more like biometrics than password. But unlike passive biometrics, FMCode is changable and more difficult to steal. For example, an attacker can collect a user's fingerprint from a cup after the user has touched it, but FMCode can only be recorded when the user performs it. FMCode suffers from server storage leakage and internal observer if the template is not properly protected, and such proper template protection might be difficult because of fussiness in alignment and matching. Also it shares all other security deficiencies of password and biometrics under attacks such as brute-force guessing/collision, dictionary guessing, phishing and cross-site leakage.

Compared with traditional password, FMCode uses handwriting, which makes it a behavior biometrics, and this provides certain protection under semantic or visual disclosure of the passcode. On the other side, compared with unchangeable biometrics like fingerprint, FMCode keeps most of the advantages of a password such as revocability and privacy preserving. This also allows one user to have more than one FMCode, different from traditional behavior biometrics like gait or voice in active speaker recognition. The most similar authentication techniques would be online signature verification (technically speaking, handwriting verification), or graphical password with stroke dynamics, but most of they assume a 2D pressure sensitive touchscreen instead of writing in the air.

6.6 Summary

In this chapter, we present FMCode, a unified user authentication and identification framework for applications using gesture input interface. FMCode does not completely replace password or biometrics in all use cases, but it can be an alternative way for user login over gesture interface where password and passive biometrics are inconvenient or not applicable. Our prototype system and experimental results show great potential of this method. However, FMCode also has limitations under certain security assumptions, which require further investigation.

Chapter 7

FMWORD - USING IAHW FOR WORD INPUT

7.1 Background

People natually communicate with each other over speech and writing. Hence, speech and handwriting are also natural ways for users to provide input information to a computer, if the computer can understand them. For example, there are many smart home appliances with virtual assistant technology enabled by modern AI technology, such that people can directly ask questions or issue control command over voice. However, there are also many application scenarios requiring a quite and private environment where voice command is not suitable. In such scenarios, typing or writing would be preferred rather than speaking. Meanwhile, many computing platforms are equipped with a gesture input interface and they lack a traditional keyboard, such as VR/AR headset, wearable devices, or game console. On these platforms, writing is more natural and efficient than typing.

Compared to speech and traditional 2D handwriting on a touch screen, IAHW is not widely studied as an input method of general text. This is partially due to the limited popularity of gesture input interfaces that can track hand movements, and partially due to the inherent challenges of recognizing IAHW words. For example, there is no consensus on what kind of motion capture sensors should be used, and hence, there is no standard dataset for IAHW word recognition research and benchmark. Moreover, collecting and labeling such a dataset is usually expensive, and as a result, it is difficult to train data-driven models for IAHW recognition.

In this chapter, we present a framework named FMWord, which utilizes a deep learning method for the IAHW word recognition task. It is an extension of the FMHash framework, which generates compact binary hash codes for the IAHW signals obtained from writing words in a predefined lexicon. These hash codes are used to search candidate words as the recognition results. Additionally, our framework can be trained with limited amount of data. Our contributions are summarized as follows.

1) We designed an IAHW word recognition framework with a deep learning method that achieve reasonably useful performance with limited amount of training data. Our framework support two types of hand movement capturing devices and two different natural languages (English and Chinese).

2) We implemented a set of data augmentation schemes that can improve the performance when large amount of training data is not available.

3) We collected a dataset from 20 users writing 210 English words as well as 210 Chinese words to evaluate our framework. Our FMWord framework can achieve over 93% top-5 recognition accuracy.

The remainder of this chapter is organized as follows. Section 2 explains the details of our FMWord framework and section 3 details the data augmentation methods we used to train our deep neural network models with limited amount of data. Section 4 shows experimental evaluation results with our dataset. Section 5 summarize this chapter.

7.2 System Model

The architecture of FMWord is essentially the same as the FMHash framework. There is only one difference, *i.e.*, the account database is replaced with a lexicon. The IAHW hand movement signals are processed in the same way as that in FMHash to generate a compact binary hash, and a pre-built hash table is used to index the words in the lexicon for word recognition instead of the accounts in the account database

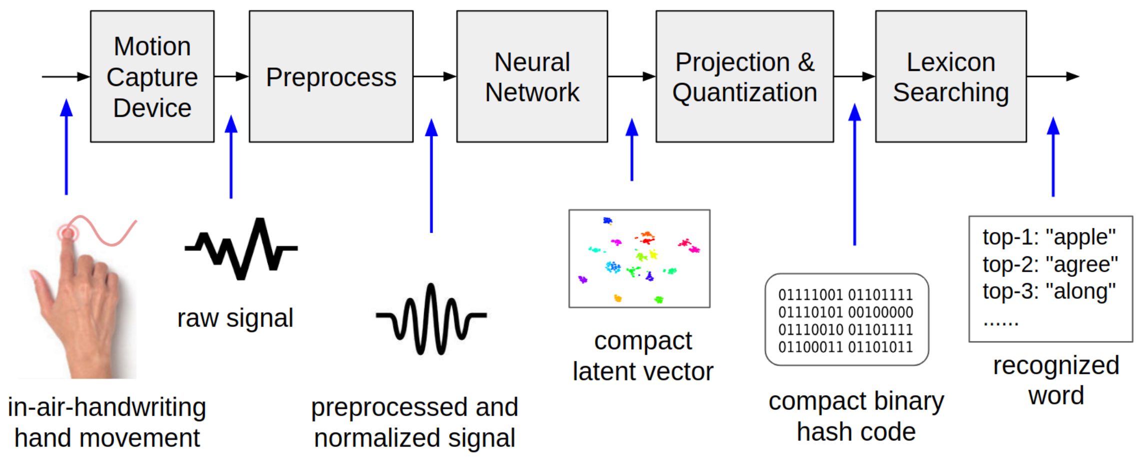


Figure 7.1: The Architecture of the FMWord Framework.

for user identification. Essentially, we convert this word recognition problem into an information retrieval problem, *i.e.*, given a request signal, the system retrieves the word ID in a lexicon by finding those signals that are most similar to the request signal. The similarity is measured through a metric on learned features defined in the loss function in the training process (Hamming distance in our case).

However, different from closed-set user identification, for this word recognition task, our framework should always return a list of candidate words and a probability of each candidate word as a confidence score. To achieve this, we modify the structure of the hash table and we use the following "registration" procedure.

Step 1): We use the same training process to generate the fuzzy hash code for each word in the lexicon. The hash code or the ith word in the lexicon is denoted as $\mathbf{b}^{(i)}$, which is essentially the same as the $\mathbf{b}^{ID}$ in chapter 5. **Step 2)**: We set up a hash table with 2^B buckets, where B is the length of the fuzzy hash code. We initiate each hash table bucket with an empty list.

Step 3): Given a hash table bucket indexed by the hash code $\mathbf{b}^{<k>}$, for each word in the lexicon with hash code $\mathbf{b}^{(i)}$, we compute the Hamming distance $d = \mathbf{b}^{<k>}$ XOR $\mathbf{b}^{(i)}$. Then, if d is greater than a certain threshold d_{th}, we further compute a probability using the following probability density function of a Gaussian distribution:

$$p(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2}}.$$

Here both the threshold d_{th} and the σ in the Gaussian probability density function are configurable parameters. After that, we put the tuple $< i, p(d) >$ into the list corresponding to the hash table bucket $\mathbf{b}^{<k>}$.

Step 4): We repeat step 3 for every hash table bucket. The insight is that each word in the lexicon is hashed to a ball with a Gaussian probability instead of just one code with absolute certainty in the Hamming space. This Gaussian ball is centered at $\mathbf{b}^{(i)}$ with a radius of d_{th} and a variance indicator of σ. If a bucket is within that ball, it contains that word in the list associated to that bucket. Finally, for each hash table bucket with a list of tuples, we normalize the $p(d)$ of the whole list and sort the list according to $p(d)$.

Once this "registration" procedure is done, the system can be used to recognize IAHW words. The recognition procedure is relatively simple. Given an IAHW signal, we use the neural network to compute its fuzzy hash code, and use the hash code to search the hash table to obtain a list of tuples. The top-k tuples are returned as the recognition result, where the first element of each tuple indicates the word in the lexicon, and the second element indicates the confidence score. Moreover, if this word recognition system is used as a word input method, typically the user will choose the word that the user intend to write from the top-k list. If the top-k list does not contain this word, the user typically want to search the next k tuples in the list until the word is found or the user gives up. Hence, we store the whole list for each bucket instead of just the top-k, and we use the threshold d_{th} to control the length of the list.

Furthermore, the user may not follow the procedure of writing one word, selecting the word from a list of candidate, and writing another word. Instead, the user may just keep writing. In this case, a continuous IAHW recognition system may automatically slice the continuous signal of writing a sentence into multiple small segments of different length during the writing, run our word recognition system for each segment, and use an n-gram model or a hidden Markov model trained on natural language texts to recognize the whole input sentence in a way similar to continuous voice recognition. In this case, the $p(d)$ associated with each tuple will be the likelihood in the n-gram model or the observation probability of the hidden Markov model. Also, $p(d)$ does not need to be normalized. This continuous IAHW recognition system is a downstream system of our word recognition framework and it is beyond the scope of this chapter. We refer this function to emphasize the importance of predicting a list of candidates and probability scores for the candidates.

In our implementation, σ is also computed from the training data. Specifically, given the training signals for the ith word in the lexicon, after the neural network is trained, we can obtain a set of fuzzy hash codes for the training signals, denoted as $H = \{\mathbf{b}^{<1>}, \mathbf{b}^{<2>}, ..., \mathbf{b}^{<m>}, ...\}$. We first do a majority vote on each bit of each hash code in H to compute the fuzzy hash code of the ith word, denoted as $\mathbf{b}^{(i)}$. Then, for each fuzzy hash code in H, we compute the Hamming distance $d_m = \mathbf{b}^{<m>}$ XOR $\mathbf{b}^{(i)}$. After that, we compute the average distance $d_{avg} = \frac{1}{K}\sum_{m=1}^{K} d_m$. Finally, we set $\sigma = 1 + d_{avg}$. The rationale is measuring the extent of "spreading" of the training signals in order to get an estimation of the shape of the Gaussian ball.

7.3 Data Augmentation

Our FMWord framework has many similarities with existing works on metric meta learning (Tang and Lian, 2021), metric-based few-shot learning (Cheng *et al.*, 2019), and prototypical learning (Snell *et al.*, 2017). These properties allow efficient training of our framework from limited amount of data (in theory our framework can be trained from only one signal for each word). However, these methods do not guarantee a useful level of recognition accuracy. Hence, to further improve the accuracy and robustness of the neural network, we implemented four data augmentation methods to generate more training data with plausible variations.

First, we applied three random rotations on the signal in the order of yaw-pitch-roll. The yaw rotation and the pitch rotation are generated randomly between zero degree and 30 degree. For the camera device, the roll rotation is generated randomly between zero degree and 15 degree, and for the glove device, it is generated randomly between zero degree and 30 degree . Here we show an example of the signal obtained by writing the string "FMWord" in Figure 7.2 (a). Five augmented signals obtained by randomly rotating this signal are shown in Figure 7.2 (b) and (c) in different perspectives angles.

We design this data augmentation step mainly to address an issue related to the pose normalization in the signal preprocessing. As explained in chapter 3, for both types of devices, we normalize the pose such that the average pointing direction is the x-axis. However, it is not guaranteed that all users will write from left to right in an invisible wall perpendicular to the pointing direction. Even for the same user writing the same content multiple times, there are variations in the direction of pointing and the direction of writing (assuming in a way similar as left-to-right). We observed this by fitting a plane on the trajectory of the signal from the camera device, and we measured that the pointing direction of the index finger and the plane is not always perpendicular. The angle between the pointing direction and its projection on the plane is somewhat randomly distributed between 60 degree to 90 degree. Also, for the camera device, we observed that the roll rotation has less variation using the gravity as a reference. Hence, we empirically choose the ranges of random rotation based on our observation of the data. However, the glove device has an additional cause of the uncertainty. When the user puts on the glove device, it may not align perfectly with the finger, especially, the IMU sensor may be rotated slightly around the finger pointing direction, which generates larger uncertainty in the roll angles relative to the gravity.

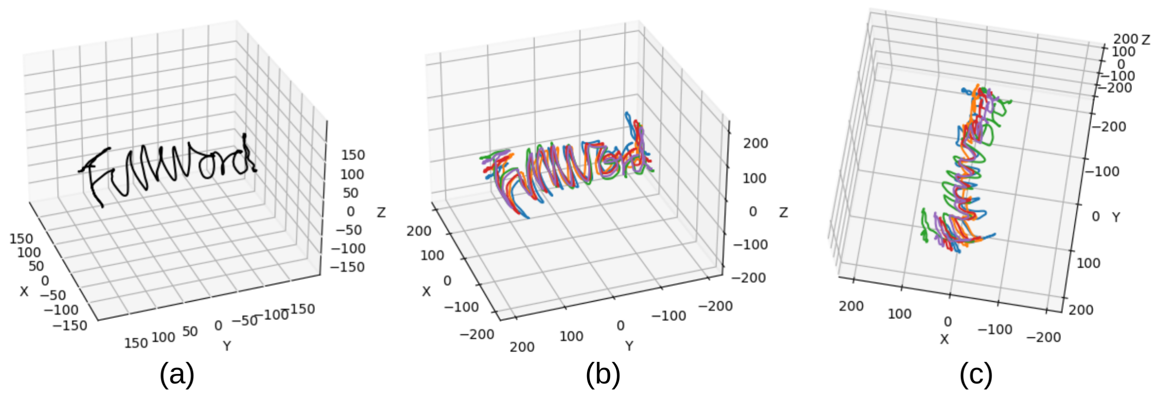


Figure 7.2: An Example of Data Augmentation with Random Rotation

Second, we applied a random warping on a signal. Similar to the signal alignment, we can modify the signal to generate a new signal according to a warping path. The rationale is that different people have different writing speed for different strokes even for writing the same word, and these differences in writing speed can be represented by the warping paths, as explained in our analysis in chapter 4.3.1. A warping path is a 2D structure as shown in Figure 4.4 (left), and it is represented by two arrays (i.e., the “start” array and the “end” array as shown in Figure 4.3). As long as one dimension of the warping path has the same length as the signal. This warping operation can be done even if the warping path is derived from other signals.

In practice, we saved all the warping paths for the “collision” class obtained in chapter 4. Given a signal for data augmentation, we randomly choose a saved warping path, resize it with linear interpolation such that both dimension of the warping path is the same as the length of the signal to be augmented, and then we use the algorithm shown in Figure 4.3 to "warp" the signal. An example is shown in Figure 7.3, where the blue signal is the original signal and orange signal is the signal after warping. Note that this step does not change the length or the trajectory of the signal. It only changes the positions of each sample on the trajectory.

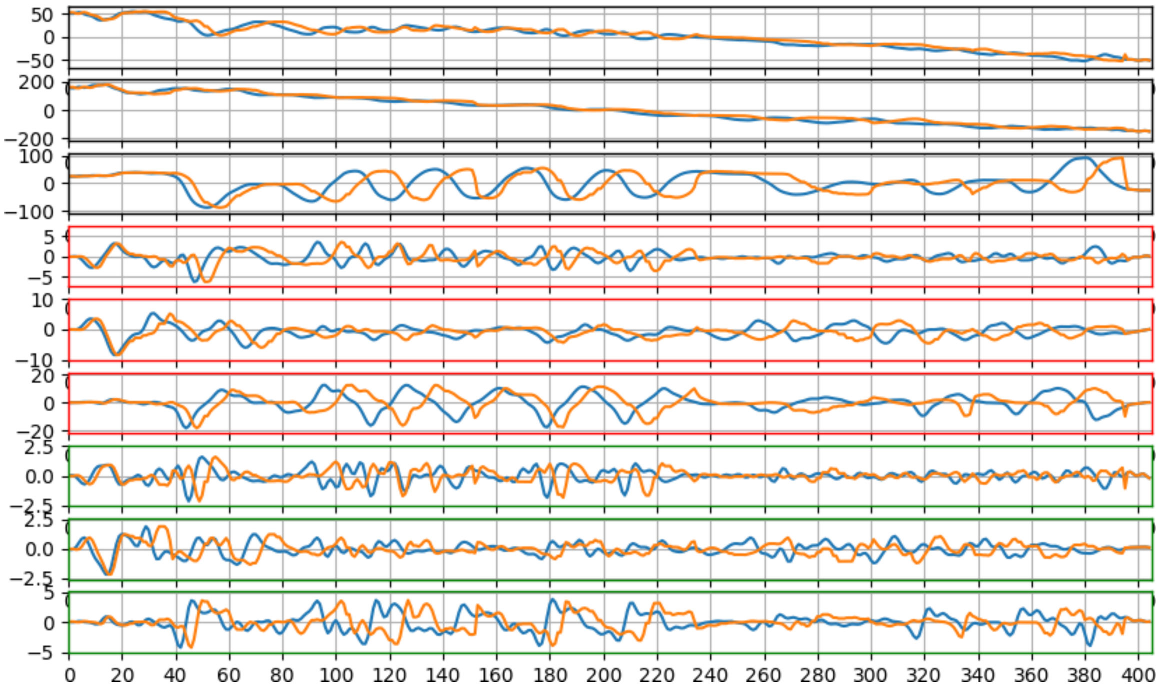


Figure 7.3: An Example of Data Augmentation with Random Warping

Third, we applied random perturbations on the signal amplitude in each sensor axis. Specifically, we create an augmented signal by adding a Gaussian shaped segment to the original signal at a randomly generated place. An example is shown in Figure 7.2, where the subplot (a) is the original signal, subplot (b) is the comparison of the original signal vs. the augmented signal, and subplot (c) is a close look at the perturbed signal segment. In this example, the random perturbation is applied on the segment from the index 100 the the index 150, which roughly corresponds to the letter "M" in the string "FMWord".

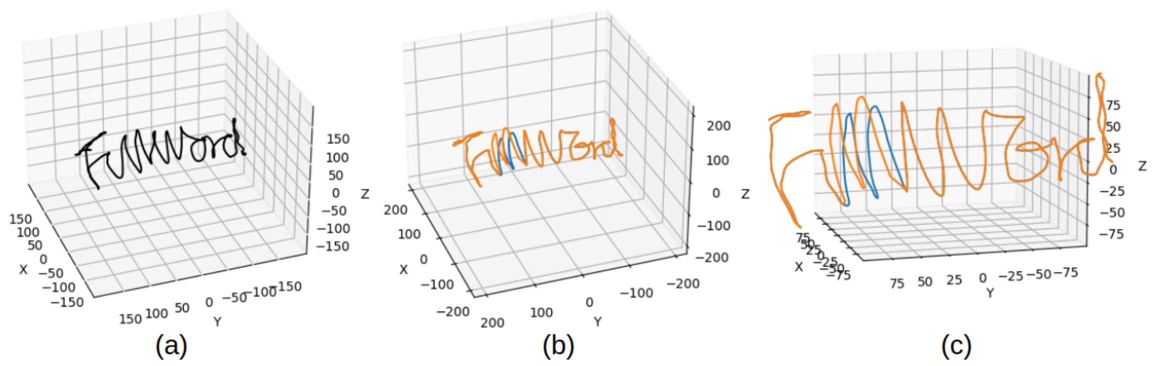
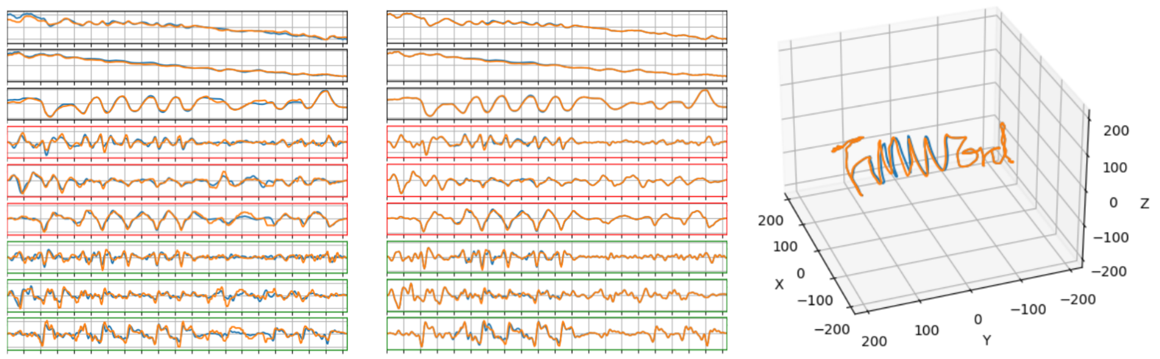


Figure 7.4: An Example of Data Augmentation with Random Amplitude Perturbation

The rationale of this data augmentation method is that the signal can vary in amplitude even for the same person writing the same content, which is analyzed in chapter 4.3.2. We observed that for a set of signals of writing the same content regardless of the writer, they have observable differences in signal amplitude after alignment. Often these differences happens in a few local segment. Hence, we perturb a segment of the signal by adding a Gaussian shaped signal segment, mimicking the effect of a random uncontrolled force generated by the muscle during the writing. The shape of the added segment is empirically determined using the analysis results of the element-wise distance for the "collision" class in chapter 4.3.2. This random perturbations on the signal amplitude can be done multiple times on the same signal.

In fact, we can use a similar method but with much more aggressive perturbation. Besides the random Gaussian perturbation, given a signal A that we want to augment, we can essentially align any signal B to signal A, and then we can generate a new signal $A' = A + \alpha B$. Here αB is the perturbation and α is usually a small number that is randomly generated (it can be either positive or negative). We typically choose B from those signals obtained by writing the same content as signal A but from other users. A signal B that is completely unrelated to A can also be chosen for greater randomness.

Fourth, for signals obtained by writing the same word, we randomly pick a pair of signals and swap a segment. Specifically, given a pair of signal A and signal B, we align B to A, randomly select a segment and overwrite this segment of A with the corresponding segment of B. To avoid sudden change of the sample values, we smooth both ends of the segment of A using a gradually transition from A to B. This process creates a new signal A'. In practice, we use a linear interpolate $A' = (1 - \lambda)A + \lambda B$ between the signal A and the aligned signal B, where λ is gradually increased from 0 to 1 in the transitioning part on both ends of the segment to be swapped. The same procedure can be applied on B with the signal A aligned to B, and a new signal B'. The pair of signals A' and B' are returned as the augmented signals. An example is shown in Figure 7.5, where the subplot (a) shows the original pair of signal A and signal B, subplot (b) shows the original signal A and the signal A' after the segment swapping, and subplot (c) shows the trajectory of the signal A and the signal A' after the segment swapping. The swapped segment starts from index 100 to index 200, which roughly corresponds to the letter "M" and the letter "W" in the string "FMWord".



(a) signal A and signal B (b) signal A and signal A' (c) trajectory of A and A'

Figure 7.5: An Example of Data Augmentation with Random Segment Swapping

7.4 Experimental Evaluation

We use the "word-210" dataset introduced in chapter 3.7 to evaluate our FMWord framework. Due to the limited amount of training data in this dataset, we use ten-fold cross validation. Specifically, for each writer among the ten writers, we split the dataset into two sets, where the testing set contains the signals generated by this writer, and the training set contains the signals generated by other nine writers. After that, the neural network model is trained using the data of nine writers and tested using the data of one writer.

Table 7.1: English Word Recognition Results

camera			glove		
testing writer	top-1 accuracy	top-5 accuracy	testing writer	top-1 accuracy	top-5 accuracy
#1	59.6	75.3	#1	69.8	92.5
#2	71.5	86.5	#2	75.0	94.2
#3	91.9	98.7	#3	98.9	99.9
#4	90.4	99.0	#4	65.9	85.0
#5	59.7	85.8	#5	79.3	96.3
#6	86.3	99.3	#6	64.5	87.2
#7	87.3	98.4	#7	82.8	97.9
#8	85.5	97.4	#8	73.9	91.3
#9	71.6	91.7	#9	90.9	97.9
#10	92.7	99.4	#10	83.7	96.2
average	**79.7**	**93.2**	**average**	**78.5**	**93.8**

Our lexicon contains only 210 words. For each word, there are 5 signals for testing

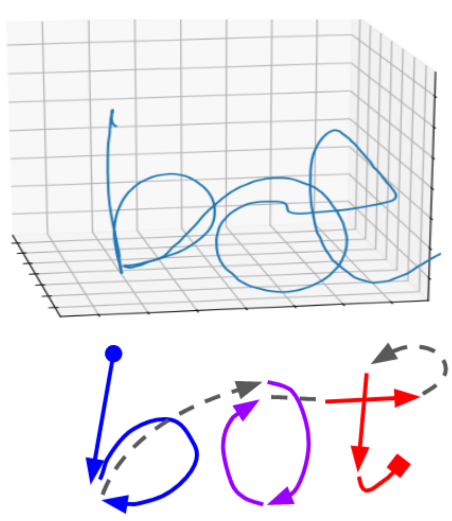
Table 7.2: Chinese Word Recognition Results

camera			glove		
testing writer	top-1 accuracy	top-5 accuracy	test writer	top-1 accuracy	top-5 accuracy
#1	94.7	99.4	#1	89.1	97.6
#2	97.2	99.6	#2	97.6	99.9
#3	87.2	97.4	#3	93.8	98.8
#4	89.1	97.1	#4	84.8	94.8
#5	72.5	89.5	#5	87.7	97.5
#6	86.4	97.6	#6	62.8	83.5
#7	87.6	98.1	#7	93.2	99.1
#8	78.9	95.4	#8	80.2	94.4
#9	87.9	96.8	#9	67.1	86.5
#10	92.5	97.2	#10	77.2	91.5
avg	**87.4**	**96.8**	**avg**	**83.4**	**94.4**

and 45 signals for training (with 5 repetitions by each of the 9 writers). We augment the training data with the following procedure. First, given a word, for each of the 5 signals generated by each of the nine writers, we use the random rotation method to create 50 signals. Second, we applied random warping on each of the 50 signals. Third, for each of the 50 signals and for each sensor axis of the signal, we add random Gaussian perturbations at a random place three times. After this step, we mix the signals from all writers and create a set of 450 signals. Fourth, for each signal A in the set of 450 signals, we randomly select another signal B from a different writer. Then, we randomly swap a segment between A and B to obtain a signal A', and we put A' into another set (the signal B' is not needed). Finally, we add the original 45 signals into the set of 450 signals with segment swapped, and we obtain a set of 495 signals. These signals are used in the training. The testing signals are not augmented.

The evaluation metrics are the top-1 recognition accuracy and top-5 recognition accuracy. The results are shown in Table 7.4 and Table 7.4. All accuracy numbers are in percentage. From these results, we can see our framework can achieve reasonably good top-5 recognition accuracy with training data from only nine writers. However, for a few specific testing writers, the recognition performance metrics are significantly worse than other testing writers.

There are two main reasons for this performance drop. First, compared to other writers, these writers wrote in a relatively less legible way. Second, different writers may write the same content in different stroke sequences or in different styles. For example, in Figure 7.6, we show examples of trajectories of two persons writing the same word "bot". It can be seen that the vertical stroke and the circular stroke in the letter "b" are basically the same for the two writers. Meanwhile, the first person writes the letter "o" clockwise while the second person writes it counter-clockwise. Also, they write "t" in different styles, which can be distinguished by determining whether the horizontal stroke is written first or last. Both English and Chinese have this kind of difference in stroke sequence and writing style with different writers. In our dataset, the writer #1 has unique behaviors, e.g., this writer wrote "o" clockwise while all other writers wrote "o" counterclockwise. Hence, when the writer #1 is selected as the testing writer, because of the writing behavioral uniqueness, many signals generated by this writer is not correctly recognized, and our data augmentation methods cannot help much.



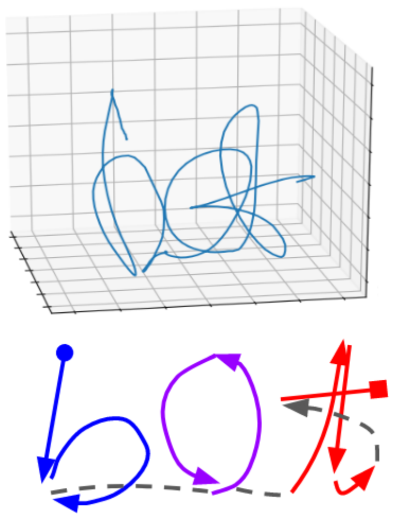


Figure 7.6: Two Different Writers Write the Same Word "bot" in Different Stroke Sequences

7.5 Summary

In this chapter, we present an IAHW word recognition framework using a neural network, which is extended from our user identification framework detailed in chapter 5. To facilitate efficient training of the neural network from limited amount of data, we developed four data augmentation methods for our IAHW signals. Our framework can achieve over 93% top-5 word recognition accuracy for both English and Chinese with a lexicon of 210 words. However, there are still several limitations regarding unique writing behaviors and illegible writing. In the future, we plan to collect a much bigger dataset with 400,000 signals and 10,000 words in the lexicon and develop a extended version of the current FMWord framework. We believe our framework has great potential for applications in VR/AR headsets, wearable devices, and home entertainment console with native gesture input interface but lack a traditional keyboard.

Chapter 8

CONCLUSIONS

This dissertation presents a unified user login framework and identity input methods using 3D In-Air-Handwriting (IAHW), where a user can log in to a virtual site by writing a passcode in the air very fast like a signature. The finger motion is captured by the gesture input interface and the information is extracted from the motion signal to verify the identity of the user. IAHW, as a combination of password and handwriting biometrics, addresses the above challenges through novel signal processing, feature design, and data-driven models such as a deep neural network. As a result, a finger motion passcode is changeable, revocable, unlinked to personal identity, and robust to spoof to a certain degree even under the leakage of the passcode content. The proposed research contains multiple tasks that span motion signal modeling, user authentication, user identification, stroke analysis for assessing the passcode strength, fuzzy-hash based cryptographic key generation from the motion signal, and a thorough evaluation in both security and usability. Our results show around 0.1% best Equal Error Rate (EER) in user authentication as well as around 93% accuracy in user identification, on a dataset collected by us with 180 users using two types of gesture input devices (a custom data glove and the Leap Motion controller). Additionally, we also present a word recognition framework as an extension of our user identification framework, which can achieve over 93% top-5 word recognition accuracy for both English and Chinese with a lexicon of 210 words.

The main contribution of this research is the framework for user login and identity input methods through gesture interfaces. Since currently there is no satisfying login solution for gesture interface, our work provides answers on how such a solution could be engineered and what challenges should be addressed. Besides, current research in this area is severely limited by the availability of the gesture input device, datasets, and software tools. Our research targets inexpensive device (less than $100), and our research outcome is made openly available, which includes our datasets, source code, a demo system, and publications, which can help research communities in both usable security and human-computer interface. Additionally, our framework is inherently language neutral, which can potentially benefit people around the world.

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning", in "Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA", (2016).

Agrawal, S., I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves and F. DeRuyter, "Using mobile phones to write in air", in "Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services", MobiSys '11 (ACM, 2011).

Amma, C., M. Georgi and T. Schultz, "Airwriting: a wearable handwriting recognition system", Personal and ubiquitous computing **18**, 1, 191–203 (2014).

Asano, T. and S. Honda, "Visual interface system by character handwriting gestures in the air", in "RO-MAN, 2010 IEEE", pp. 56–61 (IEEE, 2010).

Aslan, I., A. Uhl, A. Meschtscherjakov and M. Tscheligi, "Mid-air authentication gestures: an exploration of authentication based on palm and finger motions", in "Proceedings of the 16th International Conference on Multimodal Interaction", pp. 311–318 (ACM, 2014).

Bailador, G., C. Sanchez-Avila, J. Guerra-Casanova and A. de Santos Sierra, "Analysis of pattern recognition techniques for in-air signature biometrics", Pattern Recognition **44**, 10, 2468–2478 (2011).

Bashir, M. and J. Kempf, "Person authentication with rdtw based on handwritten pin and signature with a novel biometric smart pen device", in "Computational Intelligence in Biometrics: Theory, Algorithms, and Applications, 2009. CIB 2009. IEEE Workshop on", pp. 63–68 (IEEE, 2009).

Berndt, D. J. and J. Clifford, "Using dynamic time warping to find patterns in time series.", in "KDD workshop", vol. 10 (Seattle, WA, 1994).

Bonneau, J., C. Herley, P. C. Van Oorschot and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes", in "2012 IEEE Symposium on Security and Privacy", (IEEE, 2012).

Cao, Y., M. Long, J. Wang and S. Liu, "Deep visual-semantic quantization for efficient image retrieval", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1328–1337 (2017).

Cao, Y., M. Long, J. Wang, H. Zhu and Q. Wen, "Deep quantization network for efficient image retrieval.", in "AAAI", pp. 3457–3463 (2016).

Cappelli, R., D. Maio, D. Maltoni, J. L. Wayman and A. K. Jain, "Performance evaluation of fingerprint verification systems", IEEE transactions on pattern analysis and machine intelligence **28**, 1, 3–18 (2006).

Chahar, A., S. Yadav, I. Nigam, R. Singh and M. Vatsa, “A leap password based verification system”, in “IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)”, (IEEE, 2015).

Chan, A., T. Halevi and N. Memon, “Leap motion controller for authentication via hand geometry and gestures”, in “International Conference on Human Aspects of Information Security, Privacy, and Trust”, (Springer, 2015).

Cheng, Y., M. Yu, X. Guo and B. Zhou, “Few-shot learning with meta metric learners”, arXiv preprint arXiv:1901.09890 (2019).

Chollet, F., “Xception: Deep learning with depthwise separable convolutions”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1251–1258 (2017).

Clark, G. D. and J. Lindqvist, “Engineering gesture-based authentication systems”, IEEE Pervasive Computing **14**, 1, 18–25 (2015).

Crocker, S. and J. Schiller, “Rfc 4086-randomness requirements for security”, (2011).

Da, C., S. Xu, K. Ding, G. Meng, S. Xiang and C. Pan, “Amvh: Asymmetric multi-valued hashing”, in “The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, (2017).

Daugman, J., “New methods in iris recognition”, IEEE Transactions on Systems, Man, and Cybernetics, Part B **37**, 5, 1167–1175 (2007).

Ding, H., C. Qian, J. Han, G. Wang, W. Xi, K. Zhao and J. Zhao, “Rfipad: Enabling cost-efficient and device-free in-air handwriting using passive tags”, in “Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on”, pp. 447–457 (IEEE, 2017).

Farella, E., S. O’Modhrain, L. Benini and B. Riccó, “Gesture signature for ambient intelligence applications: a feasibility study”, in “International Conference on Pervasive Computing”, pp. 288–304 (Springer, 2006).

Florencio, D. and C. Herley, “A large-scale study of web password habits”, in “Proceedings of the 16th international conference on World Wide Web”, pp. 657–666 (ACM, 2007).

Frank, M., R. Biedert, E. Ma, I. Martinovic and D. Song, “Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication”, IEEE transactions on information forensics and security **8**, 1, 136–148 (2013).

Freeman, E., S. Brewster and V. Lantz, “Do that, there: an interaction technique for addressing in-air gesture systems”, in “CHI”, (2016).

Gafurov, D. and E. Snekkkenes, “Arm swing as a weak biometric for unobtrusive user authentication”, in “Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP’08 International Conference on”, pp. 1080–1087 (IEEE, 2008).

Gan, J. and W. Wang, “In-air handwritten english word recognition using attention recurrent translator”, Neural Computing and Applications pp. 1–18 (2017).

Gloria Omale, “Eliminate centrally managed passwords for better security, fewer breaches, lower support costs and enhanced user experience”, Gartner Embrace a Passworless Aprpoach to Improve Security (2019).

Glorot, X. and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, in “Proceedings of the thirteenth international conference on artificial intelligence and statistics”, (2010).

Gong, N. Z., M. Payer, R. Moazzezi and M. Frank, “Forgery-resistant touch-based authentication on mobile devices”, in “Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security”, pp. 499–510 (ACM, 2016).

Hayashi, E., M. Maas and J. I. Hong, “Wave to me: user identification using body lengths and natural gestures”, in “Proceedings of the 32nd annual ACM conference on Human factors in computing systems”, pp. 3453–3462 (ACM, 2014).

He, K., X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 770–778 (2016).

Hong, F., M. Wei, S. You, Y. Feng and Z. Guo, “Waving authentication: your smartphone authenticate you on motion gesture”, in “Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems”, pp. 263–266 (ACM, 2015).

Hsu, Y.-L., C.-L. Chu, Y.-J. Tsai and J.-S. Wang, “An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition”, IEEE Sensors Journal **15**, 1, 154–163 (2015).

Huang, D. and D. Lu, “Three-dimensional in-the-air finger motion based user login framework for gesture interface”, (2020).

Huang, G., Z. Liu, K. Q. Weinberger and L. van der Maaten, “Densely connected convolutional networks”, arXiv preprint arXiv:1608.06993 (2016).

Impedovo, D. and G. Pirlo, “Automatic signature verification: the state of the art”, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**, 5, 609–635 (2008).

Itaguchi, Y., C. Yamada and K. Fukuzawa, “Writing in the air: Contributions of finger movement to cognitive processing”, PloS one **10**, 6, e0128419 (2015).

Ives, B., K. R. Walsh and H. Schneider, “The domino effect of password reuse”, Communications of the ACM **47**, 4, 75–78 (2004).

Jason Tooley, “Global Augmented and Virtual Reality Market Will Reach USD 814.7 Billion By 2025: Zion Market Research”, Blogs published at biometricupdate.com (2020).

Juels, A. and M. Wattenberg, “A fuzzy commitment scheme”, in “Proceedings of the 6th ACM conference on Computer and communications security”, pp. 28–36 (ACM, 1999).

Kingma, D. and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980 (2014).

Klemmer, S. R., B. Hartmann and L. Takayama, “How bodies matter: five themes for interaction design”, in “Proceedings of the 6th conference on Designing Interactive systems”, pp. 140–149 (ACM, 2006).

Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in “Advances in neural information processing systems”, pp. 1097–1105 (2012).

Lai, H., Y. Pan, Y. Liu and S. Yan, “Simultaneous feature learning and hash coding with deep neural networks”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 3270–3278 (2015).

Lee, S., K. Song and J. Choi, “Access to an automated security system using gesture-based passwords”, in “2012 15th International Conference on Network-Based Information Systems”, pp. 760–765 (IEEE, 2012).

Lester, J., B. Hannaford and G. Borriello, ““are you with me?”–using accelerometers to determine if two devices are carried by the same person”, in “International Conference on Pervasive Computing”, (Springer, 2004).

Lien, J., N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja and I. Poupyrev, “Soli: Ubiquitous gesture sensing with millimeter wave radar”, ACM Transactions on Graphics (TOG) **35**, 4, 142 (2016).

Lin, K., J. Lu, C.-S. Chen and J. Zhou, “Learning compact binary descriptors with unsupervised deep neural networks”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 1183–1192 (2016).

Liu, H., R. Wang, S. Shan and X. Chen, “Deep supervised hashing for fast image retrieval”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 2064–2072 (2016).

Liu, J., L. Zhong, J. Wickramasuriya and V. Vasudevan, “uwave: Accelerometer-based personalized gesture recognition and its applications”, Pervasive and Mobile Computing **5**, 6, 657–675 (2009).

Liu, L., L. Shao, F. Shen and M. Yu, “Discretely coding semantic rank orders for supervised image hashing”, in “The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, (2017).

Lu, D., D. Huang, Y. Deng and A. Alshamrani, “Multifactor user authentication with in-air-handwriting and hand geometry”, in “submitted, 2018 International Conference on Biometrics (ICB)”, (IEEE, 2018).

Lu, D., D. Huang and A. Rai, "Fmhash: Deep hashing of in-air-handwriting for user identification", (2019).

Lu, D., L. Luo, D. Huang and Y. Yang, "Fmkit - an in-air-handwriting analysis library and data repository", in "in proceedings of the Fourth Workshop on Computer Vision for AR/VR in conjunction of CVPR", (2020).

Lu, D., K. Xu and D. Huang, "A data driven in-air-handwriting biometric authentication system", in "International Joint Conf. on Biometrics (IJCB 2017)", (IEEE, 2017).

Ma, W., J. Campbell, D. Tran and D. Kleeman, "Password entropy and password quality", in "Network and System Security (NSS), 2010 4th International Conference on", pp. 583–587 (IEEE, 2010).

May, K. R., T. M. Gable and B. N. Walker, "Designing an in-vehicle air gesture set using elicitation methods", in "Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications", (2017).

Mayrhofer, R. and H. Gellersen, "Shake well before use: Authentication based on accelerometer data", in "International Conference on Pervasive Computing", pp. 144–161 (Springer, 2007).

Moazen, D., S. A. Sajjadi and A. Nahapetian, "Airdraw: Leveraging smart watch motion sensors for mobile human computer interactions", in "Consumer Communications & Networking Conference (CCNC), 13th IEEE Annual", (IEEE, 2016).

Nassi, B., A. Levy, Y. Elovici and E. Shmueli, "Handwritten signature verification using hand-worn devices", arXiv preprint arXiv:1612.06305 (2016).

O'Gorman, L., "Comparing passwords, tokens, and biometrics for user authentication", Proceedings of the IEEE **91**, 12, 2021–2040 (2003).

Okumura, F., A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto and A. Koike, "A study on biometric authentication based on arm sweep action with acceleration sensor", in "2006 International Symposium on Intelligent Signal Processing and Communications", pp. 219–222 (IEEE, 2006).

Parkhi, O. M., A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition.", in "BMVC", vol. 1, p. 6 (2015).

Patel, S. N., J. S. Pierce and G. D. Abowd, "A gesture-based authentication scheme for untrusted public terminals", in "Proceedings of the 17th annual ACM symposium on User interface software and technology", (ACM, 2004).

Perrin, S., A. Cassinelli and M. Ishikawa, "Gesture recognition using laser-based tracking system", in "Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on", pp. 541–546 (IEEE, 2004).

Piekarczyk, M. and M. R. Ogiela, "On using palm and finger movements as a gesture-based biometrics", in "Intelligent Networking and Collaborative Systems (INCOS), 2015 International Conference on", pp. 211–216 (IEEE, 2015).

Piumsomboon, T., A. Clark, M. Billinghurst and A. Cockburn, "User-defined gestures for augmented reality", in "IFIP Conference on Human-Computer Interaction", (Springer, 2013).

PYMNTS, "Deep Dive: COVID-19 Spurs Biometric Innovations For A Contactless Global Society", PYMNTS.com (2020).

Qu, C., D. Zhang and J. Tian, "Online kinect handwritten digit recognition based on dynamic time warping and support vector machine", Journal Of Information &Computational Science **12**, 1 (2015).

Qu, X., W. Wang and K. Lu, "In-air handwritten chinese character recognition using discriminative projection based on locality-sensitive sparse representation", in "Pattern Recognition (ICPR), 2016 23rd International Conference on", pp. 1137–1140 (IEEE, 2016).

Ren, H., W. Wang, K. Lu, J. Zhou and Q. Yuan, "An end-to-end recognizer for in-air handwritten chinese characters based on a new recurrent neural networks", in "Multimedia and Expo (ICME), 2017 IEEE International Conference on", pp. 841–846 (IEEE, 2017).

Renuka, R., V. Suganya and A. Kumar, "Online hand written character recognition using digital pen for static authentication", in "Computer Communication and Informatics (ICCCI), 2014 International Conference on", pp. 1–5 (IEEE, 2014).

Sajid, H. and S. C. Sen-ching, "Vsig: Hand-gestured signature recognition and authentication with wearable camera", in "Information Forensics and Security (WIFS), 2015 IEEE International Workshop on", pp. 1–6 (IEEE, 2015).

Schroff, F., D. Kalenichenko and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 815–823 (2015).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556 (2014).

Snell, J., K. Swersky and R. S. Zemel, "Prototypical networks for few-shot learning", arXiv preprint arXiv:1703.05175 (2017).

Song, Y., Z. Cai and Z.-L. Zhang, "Multi-touch authentication using hand geometry and behavioral information", in "Security and Privacy (SP), 2017 IEEE Symposium on", pp. 357–372 (IEEE, 2017).

Sun, L., S. Sen, D. Koutsonikolas and K.-H. Kim, "Widraw: Enabling hands-free drawing in the air on commodity wifi devices", in "Proceedings of the 21st Annual International Conference on Mobile Computing and Networking", pp. 77–89 (ACM, 2015).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1–9 (2015).

Tang, S. and Z. Lian, “Write like you: Synthesizing your cursive online chinese handwriting via metric-based meta learning”, in “Computer Graphics Forum”, vol. 40, pp. 141–151 (Wiley Online Library, 2021).

Tian, J., C. Qu, W. Xu and S. Wang, “Kinwrite: Handwriting-based authentication using kinect.”, in “NDSS”, (2013).

Tsukada, K. and M. Yasumura, “Ubi-finger: Gesture input device for mobile use”, in “Ubicomp 2001 Informal Companion Proceedings”, (2001).

Wang, J., D. Vasisht and D. Katabi, “Rf-idraw: Virtual touch screen in the air using rf signals”, in “Proceedings of the 2014 ACM Conference on SIGCOMM”, SIGCOMM ’14 (ACM, 2014).

Wang, J.-S., Y.-L. Hsu and C.-L. Chu, “Online handwriting recognition using an accelerometer-based pen device”, in “2nd International Conference on Advances in Computer Science and Engineering”, pp. 229–232 (2013).

Wen, H., J. Ramos Rojas and A. K. Dey, “Serendipity: Finger gesture recognition using an off-the-shelf smartwatch”, in “Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems”, pp. 3847–3851 (ACM, 2016).

Wu, J., J. Konrad and P. Ishwar, “Dynamic time warping for gesture-based user identification and authentication with kinect”, in “2013 IEEE International Conference on Acoustics, Speech and Signal Processing”, (IEEE, 2013).

Xia, R., Y. Pan, H. Lai, C. Liu and S. Yan, “Supervised hashing for image retrieval via image representation learning.”, in “AAAI”, vol. 1, pp. 2156–2162 (2014).

Xu, C., P. H. Pathak and P. Mohapatra, “Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch”, in “Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications”, pp. 9–14 (ACM, 2015).

Yang, Y., G. D. Clark, J. Lindqvist and A. Oulasvirta, “Free-form gesture authentication in the wild”, in “Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems”, pp. 3722–3735 (ACM, 2016).

Ye, Z., X. Zhang, L. Jin, Z. Feng and S. Xu, “Finger-writing-in-the-air system using kinect sensor”, in “Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on”, pp. 1–4 (IEEE, 2013).

Zaharis, A., A. Martini, P. Kikiras and G. Stamoulis, “User authentication method and implementation using a three-axis accelerometer”, in “International Conference on Mobile Lightweight Wireless Systems”, pp. 192–202 (Springer, 2010).

Zeiler, M. D. and R. Fergus, “Visualizing and understanding convolutional networks”, in “European conference on computer vision”, pp. 818–833 (Springer, 2014).

Zhang, R., L. Lin, R. Zhang, W. Zuo and L. Zhang, “Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification”, IEEE Transactions on Image Processing **24**, 12, 4766–4779 (2015).

Zhang, X., Z. Ye, L. Jin, Z. Feng and S. Xu, “A new writing experience: Finger writing in the air using a kinect sensor”, IEEE MultiMedia **20**, 4, 85–93 (2013).

Zhang, Y. and C. Harrison, “Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition”, in “Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology”, (ACM, 2015).

Zhang, Y., F. Monrose and M. K. Reiter, “The security of modern password expiration: An algorithmic framework and empirical analysis”, in “Proceedings of the 17th ACM conference on Computer and communications security”, pp. 176–186 (ACM, 2010).

Zhao, F., Y. Huang, L. Wang and T. Tan, “Deep semantic ranking based hashing for multi-label image retrieval”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 1556–1564 (2015).

Zhu, H., M. Long, J. Wang and Y. Cao, “Deep hashing network for efficient similarity retrieval.”, in “AAAI”, pp. 2415–2421 (2016).