Cross-Device Authentication via Motion Co-analysis with a Smartwatch in a Multi-user Multi-device Environment

LEUNG, Ho Man Colman

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Supervised by

Prof. FU, Chi Wing and Prof. HENG, Pheng Ann

The Chinese University of Hong Kong
October 2018

Abstract of thesis entitled:

Cross-Device Authentication via Motion Co-analysis with a Smartwatch in a Multi-user Multi-device Environment

Submitted by LEUNG, Ho Man Colman

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in October 2018

Smart devices contain sensitive information that has to be guarded against unauthorized access through authentication. Existing authentication methods become obsolete as they are designed either for logging-in one device at a time or are ineffective in a multi-user multi-device environment. This thesis explores the possibilities of using an already-authenticated smartwatch as a token for fast access and control of other smart devices by analyzing their motion data. In essence, we look for synchronous rotations between devices to perform authentication. We propose *TwistIn*, a simple gesture which the user twists a device in a to-and-fro manner. This gesture rotates a smartwatch and a handheld device at the same time, which can be detected using our proposed algorithm. To log in a device, one simply needs to pick it up and twist it a few times. Then, by co-analyzing the motion data from the device and the watch, our method can extend the user authentication on the watch to the device. This is a simple and tangible interaction that takes only one to two seconds to perform. It is particularly useful for devices such as smartphones, smart glasses, and small IoT objects. Furthermore, to account for user variation in wrist bending, we decompose wrist and forearm rotations via an optimization

to improve the method accuracy. We implemented TwistIn, collected two thousand gesture samples, and conducted various experiments to evaluate our prototype system and show that it achieved over 95% detection accuracy.

論文題目:

在多用戶多設備環境中通過智能手錶聯合運動分析認證及操作智能物件

作者: 梁可文

學校: 香港中文大學

學系: 計算機科學與工程學系

修讀學位: 哲學碩士

智能物件包含敏的個人資料，必須通過用戶認證以防止未經授權的訪問。現有的用戶驗證方法每次只能登錄一個設備，或在多用戶多設備的環境中無效。 在這篇論文中，我們探討使用一個已登入的智能手錶通過聯合運動分析以認證及控制我們身邊的智能物件。 實質上，我們通過確認設備之間的同步旋轉來執行用戶驗證。 我們提出TwistIn手勢， 用戶只需拿起智能物件並擺動它一到兩秒，我們的系統便能偵測到智能手錶與智能物件的同步旋轉並立即登錄物件。這個方法對智能手機，智能眼鏡和物聯網物件等小型設備特別有效。 此外，我們開發一個嶄新的搖動及旋轉分解數學模型及實時優化算法來對齊及分析智能物件和手錶的運動數據，以提高方法的準確性。我們實行了TwistIn系統，收集了兩千個手勢樣本，並進行了各種實驗來評估系統的性能。 實驗結果顯示TwistIn系統達到超過95%的檢測精度。

Acknowledgments

I would like to thank my supervisor, Prof. Philip Chi-Wing Fu, for his great advice and guidance throughout my study. Other than knowledge, I have learned the value of working hard with patience in doing research, and the skills of presenting an idea clearly and precisely. I will always be grateful for his supervision and support in every aspect.

I would also like to thank my co-supervisor, Prof. Pheng-Ann Heng, for his encouragement in pursuing the MPhil. degree. Without him, I would not even start the journey to research, which is undoubtedly one of the best decisions I have ever made.

I am grateful to the members of the thesis assessment committee, Prof. Jiaya Jia, and Prof. Tien-Tsin Wong, for their constructive comments and invaluable suggestions, which significantly strengthen my thesis. Special thanks to Prof. Goh Wooi Boon from Nanyang Technological University who kindly served as the external examiner for this thesis.

I thank Xin Yang, Lequan Yu, and Hao Chen for their constructive discussion during my study and my friends Man Chung Yue, Sing Fan Chan, Kam Fung Cheung, Yuen Man Pun, and Wai Kuen Lau for walking the research journey with me. I also thank my colleagues, Qi Dou, Xinying Wang, Xiaomeng Li,

Yueming Jin, Huangjing Lin, Xiaowei Hu, Xi Wang, Cheng Chen, Lihao Liu and many others.

Finally, yet most importantly, I would like to thank my parents, for their unconditional love and endless support.

This work is dedicated to my beloved parents.

Contents

Abstract **i**

摘要 **iii**

Acknowledgments **iv**

1 Introduction **1**

1.1 Overview 1

1.2 Our Contributions 2

1.3 Thesis Organization 2

2 Background Study **4**

2.1 Recent Trend of Smart Devices 4

2.2 Current Authentication Approaches 5

2.3 Motivation 8

3 Literature Review **10**

3.1 Smartwatch Interaction 10

3.2 Motion Data Analysis 11

4 Robust Synchronous Rotation Detection **14**

4.1 Introduction 14

4.2 Assumptions 15

4.3 Orientations and Quaternions 15
4.4 Analyze q over Time 17
4.5 Rotations 18
4.6 Basis Transformation 19
4.7 Analyze r over Time 21
4.8 Overall Authentication Procedure 22
4.9 Limitation 23
4.10 Conclusion 23

5 TwistIn Gesture and Rotation Decomposition 25
5.1 Introduction 25
5.2 TwistIn Gesture 26
5.3 Gesture Detection 27
5.3.1 Analyze Δq over time 28
5.3.2 Count the Number of Twists 28
5.4 Rotation Decomposition 30
5.4.1 Reformulate the Optimization 31
5.4.2 Deriving the Optimization Solution 33
5.5 Overall TwistIn Authentication Procedure 35
5.6 Limitation 36
5.7 Conclusion 36

6 Performance Evaluation 37
6.1 Introduction 37
6.2 Evaluating Authentication Accuracy 38
6.2.1 Hardware and Data Collection 38
6.2.2 Evaluation Methodology 40
6.2.3 Offline Analysis Software Implementation 42

6.3 Experiment 1: Determining the Number of Twists for a TwistIn Gesture 46
6.4 Experiment 2: Comparing Performance of Different Methods . 50
6.5 Experiment 3: Challenging Scenarios 55
6.6 Evaluating Usability 60
6.6.1 Correlation between Performance and User Preference . 63
6.6.2 Correlation between Performance and Physical Differences 64
6.6.3 Feedback from participants 64
6.7 Conclusion 67

7 Conclusion **68**

Bibliography **72**

List of Figures

2.1 Examples of smart devices 5

4.1 A 3D rotation represented by two opposite quaternions 16
4.2 Explaining a device's reference frame, orientation, and rotation 18
4.3 An Illustration of the basis transformation between two devices 19

5.1 An illustration of the TwistIn authentication procedure 26
5.2 The composition of two rotations during TwistIn 27
5.3 An illustration of rejecting invalid pairs of motion data by matching the turning points in time 29
5.4 Choosing the correct basis transformation solution 34
5.5 A running example showing the time series processing in the TwistIn Authentication Procedure 35

6.1 A series of images showing how a single twist is performed . . 39
6.2 Using a Gaussian filter to resample the orientation data 39
6.3 A simplified call graph of the program 43
6.4 A simplified UML class diagram of the program 43
6.5 Plotting the device rotations in various challenging scenarios . 57
6.6 Performance of each participant achieved in experiment 2 . . . 61
6.7 Tests for statistical significance between users' physical differences and their corresponding performances 62

List of Tables

6.1 EER achieved with different number of twists in experiment 1 . . . 48
6.2 Effect of prolonged twisting in experiment 1 . . . 48
6.3 Parameters used in experiment 2 . . . 52
6.4 Performance obtained in experiment 2 . . . 53
6.5 Performance obtained in experiment 3 . . . 56
6.6 False positive rate obtained using motion samples from experiment 1 to 3 . . . 56
6.7 Distribution of the participants' physical measurements in experiment 2 . . . 60
6.8 Questions and responses of the interview . . . 65

Chapter 1

Introduction

1.1 Overview

This thesis is about authenticating devices quickly in an environment with multiple users and devices. There are progressively more devices around us due to the advancement in technology. Some of these devices, however, are small with a limited interface. Since current methods are not designed for accessing many of these small devices, a new method that is fast, secure, and intuitive is crucial to a better user experience.

This study aims to explore the possibilities of authenticating without using an interface, or more specifically, using motion data retrieved from the devices. The purpose of this introductory chapter is to provide the overview of this thesis. The contributions are listed in the second section and an organization of this thesis is presented at the end of the chapter.

1.2 Our Contributions

The main contributions of this thesis are listed below:

- The TwistIn gesture as a novel authentication mechanism for fast login to smart devices in a multi-user multi-device environment using a smartwatch as an authentication token is proposed.
- A fast algorithm developed to detect synchronous rotatory motions between the smartwatch and the smart device by solving for the optimal orientation difference between the two devices.
- A rotation decomposition technique formulated to filter out the wrist rotation from the forearm rotation when handling noisy orientation data.

1.3 Thesis Organization

This thesis consists of seven chapters:

Chapter 2 **Background Study**
A study on the recent trend of smart devices is presented. The current authentication approaches are then reviewed. Finally, this thesis is justified with a motivation.

Chapter 3 **Literature Review**
A detailed review of the previous research in Smartwatch Interaction and Motion Data Analysis is presented.

Chapter 4 **Robust Synchronous Rotation Detection**
An authentication framework via detecting motion and synchronous rotation is proposed. Some basic geometry and the use of quaternion to represent 3D rotation is first reviewed. An optimization is introduced to solve the basis transformation between two devices.

Then, the transformation is analyzed over time to complete the authentication process.

Chapter 5 **TwistIn Gesture and Rotation Decomposition**

An improvement to the authentication framework is proposed. The TwistIn gesture is introduced together with two gesture detection techniques. The optimization described in the previous chapter is then modified to incorporate rotation decomposition for filtering out unwanted rotations.

Chapter 6 **Performance Evaluation**

The proposed methods are evaluated in terms of authentication accuracy. Motion data are collected from participants performing the TwistIn gesture and are analyzed offline using a program. The usability of the TwistIn gesture approach is further evaluated through a user study.

Chapter 7 **Conclusion**

The final chapter summarizes this thesis with a discussion of limitation, application, and future work.

□ **End of chapter.**

Chapter 2

Background Study

Summary

The current trend of smart devices and the flaws in existing authentication approaches are reviewed in this chapter, followed by the motivation of this thesis.

2.1 Recent Trend of Smart Devices

Technological advancements have brought us more compact and cost-effective electronics. Nowadays, a large number of devices are augmented with smart functionalities, such as smart alarm, smart speaker, smartphone, smart glasses etc.; see Figure 2.1. It is expected that such a trend will continue to grow with two emerging technologies. The first is "Internet of Things" (IoT), which connects the smart devices around us. The advanced connectivity offered is beneficial for data collection, environment sensing, and ultimately automation. This encourages developers to adopt the IoT platform for the extended functionalities. In fact, recent studies have forecasted that there will be more than 20 billion connected devices by 2020 [15, 7]. The second is "3D Printing",

Figure 2.1: Examples of devices with smart functionalities: Wiimote, Google Glass, LG 360 Cam, mouse, Xbox controller, Raspberry Pi embedded into Lego assembly, sensor tag keychain, alarm clock, 3D electronic print, Samsung Gear 360, ambient orb, radio remote controller, RC toy, and smart speaker (from A to N).

which has been evolved to incorporate electronic circuits during the printing process [51, 42]. Hence, people will soon be able to design and fabricate objects not only for the 3D shapes but also for the electronic functions, right from their home. However, with the combined contributions from these technologies, the exponential growth of smart devices poses a new human-computer interaction problem, where efficient connection and control of these prevalent devices are lacking.

2.2 Current Authentication Approaches

A well-designed authentication system should be intuitive and easy to use, while being able to accept legitimate users and reject imposters [50]. With the increasing number of smart devices in the surrounding, current methods that authenticate devices one at a time becomes burdensome, especially when multiple devices are shared among multiple users.

PIN, Passcode, Swipe Pattern

A simple approach is to build a hard button menu or a touchscreen on a device for users to log into the device via a PIN, a passcode, or a swipe pattern. However, this will inevitably complicate the device and increase the production cost. In addition, this is a knowledge-based authentication, which is not suitable for mobile devices whose passcode could be guessable [57, 2], or easily obtained through a smudge attack [4] or shoulder surfing [48].

Biometrics

Apart from the menu-based approach, one may use physiological biometrics such as fingerprint (e.g., Apple TouchID) and face [6] for authentication. Even though they are easy to use, one drawback is that the biometrics of a person cannot be easily changed. Since biometric authentication systems are vulnerable to spoofing attacks [40], this could lead to serious consequences if such information is disclosed to other parties. Furthermore, the approach may not work in certain circumstances, e.g., fingerprint recognition usually fails for wet/dirty fingers, and strong sunlight could disrupt the cameras for face recognition.

Another similar approach is to use behavioral biometrics, e.g., [14, 26, 63, 64]. This relies on extracting features from sensor data and using classifiers to verify the authentication. Hence, it requires data analysis and training to learn the user's biometrics, and the learned model is also limited to a specific action, meaning that multiple models are needed to recognize variants. Moreover, users are also required by replicating an exact (or similar) action to enable the authentication.

Mobile Application

At present, the norm to interact with smart devices is to make use of mobile applications on a personal smartphone. This, however, requires us to switch our mind from the physical world to a virtual world (app), even though the devices we want to access are just physically located next to us. Besides, the approach is not scalable and not efficient for handling multiple devices. Every time we want to switch from one device to another, we have to search for the respective mobile app and log in again before accessing the device. Such interaction is neither natural nor efficient. Furthermore, it could also have a direct impact on the user's behavior in adopting the login methods and the related security settings, as suggested in some recent user studies [5, 58, 12].

On the other hand, methods that allow authentication to a group of devices are inappropriate for environments with numerous people, e.g., workplace and public venue.

Virtual Assistants

For instance, virtual assistants such as Apple's Siri and Amazon's Alexa may be used to interact with smart devices, but they are ineffective in crowded environments since the noise in such environments could significantly deteriorate the voice recognition performance. Moreover, having multiple people speaking at the same time in different languages is also difficult to process.

Physical Proximity

Another solution is to detect physical proximity and unlock a device when an authentication token is close to the device [59]. Devices equipped with Bluetooth Low Energy [17] such as Apple's iBeacon [3] and Google's Eddystone [1] enable proximity sensing by measuring the received signal strength indicator

and using a path loss model [16]. For example, an Apple's Mac computer can be unlocked by bringing an authenticated Apple Watch near it [10]. While this method does not require any user's attention which provides great user experience, it may cause unintentional logins and unauthorized accesses when the user passes by a device without the intention of using it, or when an attacker carries the device close to the user. Furthermore, the presence of multiple users may also confuse a device, since it cannot deduce who is currently using it.

Computer Vision

Computer Vision-based systems have been deployed in real life scenarios (e.g., Amazon's Go). A large amount of cameras are installed to provide information of an area and computers are used to analyze interaction within the area. It is expensive since the number of cameras and computers needed increases exponentially with the size of the monitored area. In addition, the line of sight of the cameras can be easily blocked inside a crowded room.

2.3 Motivation

Multi-user Multi-device Authentication

To develop an authentication method that is suitable in a multi-user multi-device environment, a key component is to distinguish the user's intention. A device needs to identify the user before it can be authenticated automatically; it is inconvenient if the user has to input the username upon login. Observed that unless the user is accessing the device remotely, he/she has to interact physically. Usually, a small device such as a smartphone has to be picked up before use. If we could identify if that the pick-up action belongs to a user, the smartphone can then be authenticated immediately. Hence, we can use

a smartwatch to capture the motion of the user's hands and then co-analyze with the motion of the smart device.

Smartwatch

Smartwatches are small-size electronics that are commercially brought into the market in the 2010s, e.g., Android Wear and Apple Watch. Unlike digital watches, smartwatches are wearable computers with an operating system, so they are programmable for supporting a wide variety of applications, e.g., information display and health tracking. Indeed, a recent study also showed that people found smartwatches more useful than smartphones for accessing time [36], and for short tasks like checking notifications. The reason behind is that smartwatches are more readily available and users often wear them for a long period of time.

Motivated by these findings, we aim to explore in this thesis *the application of an already-authenticated smartwatch as a token to tangibly gain quick access to some other smart devices in a multi-user multi-devices environment*.

□ **End of chapter.**

Chapter 3

Literature Review

Summary

We review previous research on interactions with a smartwatch and techniques of motion data analysis.

3.1 Smartwatch Interaction

Traditionally, a watch has been used solely for telling the time and date. Throughout the years, additional features have been constantly added, such as a stopwatch and an alarm. Until the 2010s, the ability to run an operating system inside a watch, termed as a "smartwatch", has essentially changed how people interact with these wearable devices.

A smartwatch has distinct characters that contrast with a smartphone, having an edge on specific tasks over other devices, but at the same time posing limitations on interaction design. The major constraint comes from the small size of the watch. This is a problem of balancing between portability and computational capability: too small and the watch will suffer from lowered

processing power, shortened battery life, and limited I/O interfaces; too large and it will no longer be convenient to be strapped on the hand. While electronics are progressively smaller and faster over the years, a small interface is always inevitable for wearable devices. Hence, input methods designed for small devices have been a research problem that has been intensively studied and explored (e.g. [41, 22, 18, 55].

On the other hand, the specific mount location, the constant contact with the skin, and the always-available nature [44] allowed unique interactions to be specially designed for smartwatches. To name a few, one can detect seizure-like activity using accelerometer commonly found in a smartwatch [35]. This activity is detected by looking for rhythmic, repetitive movement of an extremity. One can also perform indoor tracking using activity fingerprint generated from inertial and acoustic sensors of a smartwatch [33]. Pearson et al. [43] explored the possibility of using smartwatches as public displays. This provides relevant information to other people in the proximity, such as weather and the wearer's schedule. Xu et al. [62] recognize gestures and finger-writing using motion sensors installed inside a smartwatch. The acceleration and angular velocity are sufficient to uniquely identify the user's hand and finger gestures.

3.2 Motion Data Analysis

Motion data retrieved from devices are analyzed and utilized in many daily usage scenarios. Various motion gestures have been designed to provide intuitive controls for mobile devices [47, 46, 29]. Moreover, motion data obtained from smartphones and smartwatches had been used for activity classification [30, 54, 19, 52, 38], and for virtual data manipulation [11, 53, 28]. By fusing motion data with data from other sensors, we can produce higher-level information to enhance user interaction. For instance, touch events obtained

from a multi-touch display could be fused with motion data to distinguish users [49, 25, 45]. Hinckley et al. [21] provided touch-enhanced motion gestures on a smartphone to enable more expressive touch interactions. Duet et al. [9] classified interaction on a smartphone by analyzing the phone's touchscreen and the motion data of both smartphone and smartwatch. Wilkinson et al. [60] enriched touch-based interaction on a multi-touch display using wrist-worn inertial measurement units. Bing et al. [8] enhanced the security of traditional password authentication by considering smartwatch motion. Laput et al. [32] fused motion data with readings from various sensors, such as ambient temperature and illumination intensity, to provide general-purpose sensing of the environment.

Previous works analyze motion data from multiple devices for different purposes. For example, Tran et al. [56] identified multi-device gestures using a hidden Markov model, while Kim et al. [27] developed an authoring system for multi-device acceleration-based gestures, where devices can be associated via synchronous patterns. Hinckley [20] detected synchronous bumping gestures against one another by looking for a pair of devices with synchronized opposite accelerations. Lester et al. [34] determined if multiple devices are carried by the same person by analyzing the walking data. Since walking generates a periodic acceleration, gait data can then be analyzed in the frequency domain using a coherence function. Wolf et al. [61] detected object pick-up events by comparing the magnitude of angular velocity among devices. When the magnitudes of two devices match each other, a seamless interaction is reported between the smart devices.

Among the previous works, ShakeUnlock by Findling et al. [13] is the most closely related work to ours. This method unlocks a device by shaking it with

another device that has been authenticated. While their work has similarities with ours by co-analyzing the motions of smartphone and smartwatch, their method simply considers simultaneous shaking of the two devices and matches them by correlating the devices' shaking frequency and magnitude in the acceleration data. Hence, the true matching rate and true non-matching rate reported in [13] are only 0.795 and 0.867, respectively.

□ **End of chapter.**

Chapter 4

Robust Synchronous Rotation Detection

Summary

A review of orientations, rotations, basis transformations, and the representation of rotation using unit quaternions is presented. Then, an authentication procedure based on detecting synchronous rotation is proposed in this chapter.

4.1 Introduction

We focus on a scenario where a user is interacting with a smart device using the hand that is wearing a smartwatch inside a multi-user multi-device environment. To distinguish if that specific user picks up the smart device, we can study and analyze the motion data from both devices together. In this chapter, we first make some assumptions and go through some basic geometry involving orientations, rotations, and basis transformations. Then, we proceed to construct an optimization to robustly solve for the basis transfor-

mations between two sets of rotations and verify the devices are rotating in synchrony. An overview of the whole authentication procedure is summarized in Algorithm 2 (Section 4.8).

4.2 Assumptions

Here, we assume that the smartwatch is already logged in by the user, and will stay in the authenticated state until the user takes off the smartwatch. Moreover, The devices are also synchronized in time, so the motion data can be compared and analyzed across devices.

4.3 Orientations and Quaternions

Assuming each device contains motion sensors including an accelerometer and a gyroscope. They are capable of measuring acceleration and angular velocity in three dimension respectively. The sensors data can then be fused into orientation in unit-quaternion representation using a sensor fusion algorithm such as a Madgwick Filter [37] or a Kalman Filter[39]. Finally, two time series of 3D orientation data represented as quaternions q_A^t and q_B^t for devices A and B at time t can be obtained.

Unit quaternions represent 3D rotations in a straightforward fashion. Indeed, every rotation in axis-angle representation can be easily converted to unit quaternion. For instance, an anticlockwise rotation of angle ϕ around axis $\vec{v} = [v_x, v_y, v_z]$ can be represented by the following quaternion:

$$q = [\cos\frac{\phi}{2}, v_x \sin\frac{\phi}{2}, v_y \sin\frac{\phi}{2}, v_z \sin\frac{\phi}{2}], \tag{4.1}$$

which is in fact a unit vector in 4D space.

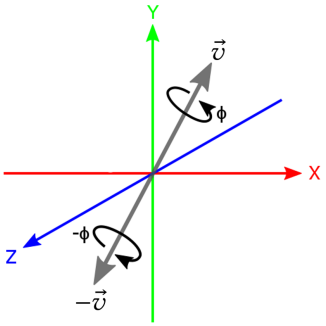


Figure 4.1: A 3D rotation of ϕ degrees around an arbitrary axis $\vec{v}$ is the same as a 3D rotation of $-\phi$ degrees around $-\vec{v}$. Therefore, both quaternions q and $-q$ represents the same 3D rotation.

Note that quaternions are compact representations (i.e., unit vectors in 4D) with which we can smoothly interpolate orientations and rotations in 3D space. In addition, the unit-quaternion notation is simpler for us to enforce the unit magnitude constraint compared to the orthonormal matrix representation for 3D rotations [23].

However, the quaternion formulation represents the same orientation by q or $-q$, whose components have equal magnitude but opposite signs. This is because a 3D rotation of ϕ degrees around an arbitrary axis $\vec{v}$ is the same as a 3D rotation of $-\phi$ degrees around $-\vec{v}$; see Figure 4.1. Hence, before we interpolate between two quaternions, say q_1 and q_2, we should flip the sign of q_2, if the interpolation path (angular changes) between q_1 and $-q_2$ is shorter than that between q_1 and q_2. Therefore, after reading the input orientation data, we first check and flip each quaternion by comparing it with the previous quaternion in the data series. We calculate the angle between two successive quaternions, say q^t and q^{t+1}, in a time series as

$$\theta = \cos^{-1}(\ 2\langle q^t, q^{t+1} \rangle^2 - 1 \ ), \tag{4.2}$$

where $\langle q^t, q^{t+1} \rangle$ stands for the dot product of q^t and q^{t+1}. In other words,

$$\langle q^t, q^{t+1} \rangle = \pm \sqrt{\frac{\cos\theta + 1}{2}} \ . \tag{4.3}$$

It can be seen that the dot product of two quaternions is closely related to the angle θ between them. In fact, if θ is larger than π, their dot product should be negative. Therefore, we calculate the dot product between successive quaternions and negate the next quaternion (q^{t+1}) accordingly; see Algorithm 1.

ALGORITHM 1: Detect and flip successive quaternions to minimize the angular changes between all successive pairs.

Input: q^1, q^2, ..., q^n

Output: q^1, q^2, ..., q^n

for $i = 1, 2, ..., n-1$ **do**
  if $q^t \cdot q^{t+1} < 0$ **then**
    $q^{t+1} = -q^{t+1}$
  end
end

4.4 Analyze q over Time

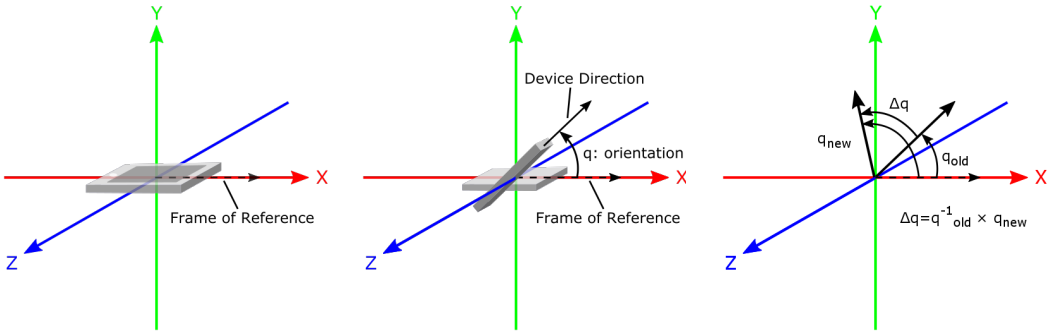
When the user is at rest, the smartwatch should not register any motion other than noise. Similarly to devices that are not in use. Therefore, stationary devices should be ignored in the algorithm. To check if a device is moving or not, we analyze q over time before we perform authentication. We measure the change of device's orientation by performing dot products on consecutive data. The faster the user moves the device, the larger the dot product is. We compare the sum of the dot products against a threshold β_{move} to deduce if the device is sufficiently moved.

$$\beta_{\text{move}} < \sum_{t \in T_{\text{auth}}} 1 - q^t \cdot q^{t+1} \ , \tag{4.4}$$

where T_{auth} is the time window in which we analyze the data and perform authentication.

If the sum of dot product is smaller than the threshold, we can immediately stop and exit the algorithm. Otherwise, we proceed to the next step where we calculate the devices' rotations.

4.5 Rotations



(a) (b) (c)

Figure 4.2: (a) The device's reference frame. (b) The orientation of a device refers to the rotation from the reference frame to the device's current orientation. (c) Given q_{old} and q_{new}, we can compute the device's change in orientation Δq as shown above.

Each orientation data of a device is in fact a 3D rotation from a reference frame; see Figures 4.2a and 4.2b. Typically, the device's reference frame is unknown, since it is hardware- and software-dependent, as well as depending on the initial bearing of the device when the device starts moving. Moreover, the user may have different ways of holding a device at different times, so we cannot assume a fixed orientation difference (i.e., basis transformation) between the reference frames of the two devices. Rather, we have to properly rotate and align the two motions. Hence, to facilitate the comparison between

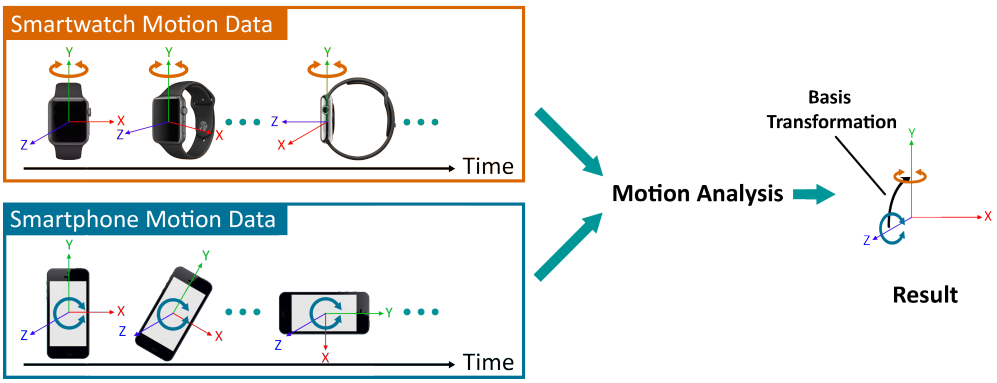


Figure 4.3: To measure the correlation between two motion data acquired in different reference frames (coordinate systems), we first need to find the basis transformation to rotate and align the motions in a common frame.

the quaternion series from the two devices A and B, we compute the change in orientation over time for each of the two devices (see Figures 4.2c):

$$\Delta q_A^t = q_A^{t+\alpha} \times (q_A^t)^{-1} \quad and \quad \Delta q_B^t = q_B^{t+\alpha} \times (q_B^t)^{-1} \ , \tag{4.5}$$

where q^{-1} stands for the inverse of quaternion q, Δq is the change in q over time, and α is empirically set as 0.1 seconds, since computing the difference between two consecutive quaternions (with much shorter time difference) would result in a very tiny rotation, which could be too sensitive to the noise.

4.6 Basis Transformation

If the two devices move together, we should be able to find a basis transformation, which a 3D rotation, to align their orientation data; see Figure 4.3.

Suppose $\Delta q_A^t : q_A^t \rightarrow q_A^{t+\alpha}$ is a linear transformation, which represents a rotation in device A's space from q_A^t to $q_A^{t+\alpha}$ and similarly, $\Delta q_B^t : q_B^t \rightarrow q_B^{t+\alpha}$ for device B's space. Denoting r as the quaternion of the basis transformation, we should have

$$\Delta q_A^t \ = \ r \ \Delta q_B^t \ r^{-1} \quad \forall t \in T_{\text{basis}} \ , \tag{4.6}$$

where T_{basis} is the period in which we estimate the basis transformation r and $T_{\text{basis}} < T_{\text{auth}}$

Next, there are two issues that we have to deal with. First, the basis transformation computed between Δq_A^t and Δq_B^t is not the same as that between Δq_A^t and $\Delta(-q_B)^t$. Fortunately, this issue has been handled after the data acquisition when we detect and flip the quaternion samples; see the previous subsection. Second, as the acquired data does not have infinite precision and is contaminated with noise, we have to find an r that best fits all pairs of orientation data via an optimization that maximizes the alignment of the orientation data:

$$\begin{aligned} \max_{r} \quad & \sum_{t \in T_{\text{basis}}} \langle \Delta q_A^t, r\ \Delta q_B^t\ r^{-1} \rangle \\ \text{subject to} \quad & ||r|| = 1\ . \end{aligned} \tag{4.7}$$

Suppose $\Delta q_A^t = [w_A^t, x_A^t, y_A^t, z_A^t]$ and $\Delta q_B^t = [w_B^t, x_B^t, y_B^t, z_B^t]$. We can solve for r directly by expressing quaternion multiplication as matrix multiplication. We rearrange the objective function to

$$r^T \left(\sum_{i=1}^{n} Q_A^{iT} \bar{Q}_B^i \right) r \tag{4.8}$$

where

$$Q_A^i = \begin{bmatrix} w_A^i & -x_A^i & -y_A^i & -z_A^i \\ x_A^i & w_A^i & -z_A^i & y_A^i \\ y_A^i & z_A^i & w_A^i & -x_A^i \\ z_A^i & -y_A^i & x_A^i & w_A^i \end{bmatrix}$$

$$\bar{Q}_B^i = \begin{bmatrix} w_B^i & -x_B^i & -y_B^i & -z_B^i \\ x_B^i & w_B^i & z_B^i & -y_B^i \\ y_B^i & -z_B^i & w_B^i & x_B^i \\ z_B^i & y_B^i & -x_B^i & w_B^i \end{bmatrix}$$

or

$$r^T \left(\sum_{i=1}^{n} N_i \right) r = r^T N r \tag{4.9}$$

Since N is not symmetrical, we calculate $N' = \frac{N+N^T}{2}$ which is a symmetric matrix. Then, the optimal r that maximizes the optimization will be the eigenvector of N' corresponding to the largest eigenvalue.

4.7 Analyze r over Time

To verify if the motions of the two devices are in synchrony, we analyze r over time by constructing a time series for r (denoted as r^t) over T_{auth}. Note that T_{auth} can be divided into three sub-periods: (i) before the device is picked up, (ii) the user is picking up the device, and (iii) the device is held in the user's hand. Among the sub-periods, we focus on the middle sub-period, since the devices undergo major rotations while the device is being picked up. Therefore, we compute weights (h^t, which is also a time series) associated with r^t by calculating the rotation magnitude between successive orientation samples:

$$h = \sum_{t \in T_{\text{basis}}} \left[(1 - min(1, |\Delta q_A^t \cdot \Delta q_A^{t+1}|)) + (1 - min(1, |\Delta q_B^t \cdot \Delta q_B^{t+1}|)) \right] . \tag{4.10}$$

Then, we compute the weighted mean of r over the period:

$$\bar{r} = \frac{\sum_{t \in T_{\text{auth}}} h^t r^t}{|| \sum_{t \in T_{\text{auth}}} h^t r^t ||} . \tag{4.11}$$

In detail, if the motion of two devices are in sync, the variation of r over T_{auth} should be small and the objective function (see Eq. (4.7)) should also have a small value. Hence, we consider two conditions to see if the motions are in sync: (i) the weighted mean squared prediction error, which describes how $\bar{r}$ fits the data, and (ii) the weighted variance of r^t, which describes the stability of the estimated basis transformation over T_{auth}. Afterwards, we check the results against their corresponding thresholds β_{err} and β_{var}:

$$\sqrt{\frac{\sum h^t(1-(\Delta q_A^t \cdot \bar{r}\Delta q_B^t \bar{r}^{-1})^2)^2}{\sum h^t}} < \beta_{\text{err}} \tag{4.12}$$

$$\sqrt{\frac{\sum h^t(1-(r^t \cdot \bar{r})^2)^2}{\sum h^t}} < \beta_{\text{var}} \ . \tag{4.13}$$

If both conditions are true, we said that the two motions are in sync.

4.8 Overall Authentication Procedure

In this section, the overall authentication procedure is presented, which is built on top of the various components described earlier; see Algorithm 2. To summarize, a series of orientation data from each device is collected and resampled using a Gaussian Filter. The resampled time series are then used to calculate the devices' rotation. After that, a time series of basis transformation is solved via an optimization model (See Eq. (4.7)). After checking if both devices have been moving sufficiently by analyzing q over time, the basis transformation time series is analyzed by calculating the weighted mean squared prediction error and the weighted variance of basis transformation. If both values were smaller than their respective thresholds, the devices will be authenticated.

ALGORITHM 2: Overall Authentication Procedure

Input: Time series of Orientation data from Devices A and B

Output: Authentication State

Resample and smooth orientation data using Gaussian filtering (q_A^t and q_B^t)

Calculate device rotation (Δq_A^t and Δq_B^t)

Calculate basis transformation using optimization (r^t)

if *both devices are in motion (i.e., not stationary)* **then**

  Calculate weighted mean squared prediction error

  Calculate weighted variance of basis transformation

  return true if both values are smaller than their respective thresholds

end

return false

4.9 Limitation

There are two limitations regarding this method. First of all, the human hand consists of a wrist joint. Any motion caused by the wrist will be reflected at the device held by the hand but not the smartwatch. This discrepancy leads to incorrect rejection, which is a type II error. Another problem is the accidental authentication due to the device's incapability of distinguishing user's intention. It will authenticate as long as the same rotation is registered at both devices, such as when the user is making a turn. This is a type I error where the devices are authenticated without the user's consent.

4.10 Conclusion

The mathematical background of motion co-analysis is reviewed in this chapter. We described how the input sensor data is used to verify the authentication. However, there are still a number of shortcomings that are mentioned in section 4.9 that have to be addressed. In the next chapter, we introduce a

twisting gesture which is designed for users to access devices and an improved optimization that filter out unwanted motion through rotation decomposition to tackle these challenges.

□ **End of chapter.**

Chapter 5

TwistIn Gesture and Rotation Decomposition

Summary

An intuitive twisting gesture is introduced and two techniques for gesture detection are proposed. An algorithm is proposed to filtered out unwanted rotation from noisy motion data by decomposing the rotation.

5.1 Introduction

Our aim is to authenticate a device when a user picks it up. In the previous chapter, the devices are authenticated through detecting synchronous rotations between them. However, the devices should not be unlocked if the user has no intention to use them. For example, when the user is on a bus and the bus makes a turn, the same rotation would be registered in both devices; this should not be treated as a successful authentication. Therefore, in this chapter, an easy-to-use twisting gesture that allows users to indicate their intention

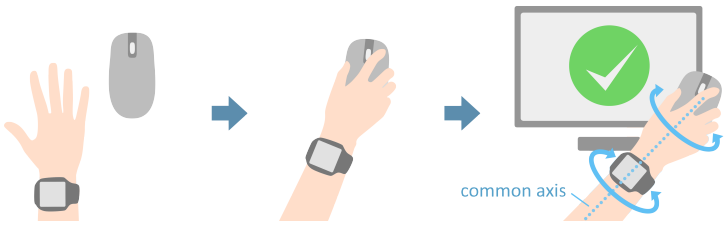


Figure 5.1: Illustrating how *TwistIn* works. In this example, one may log in a computer through a smart mouse. First, one may pick up the mouse (middle) and perform the TwistIn gesture in mid-air for one to two seconds (right). Then, our method can make use of the motion sensor installed in the smart mouse to record the mouse's motion and co-analyzes it with the watch's motion to detect if the two motions are in sync. If so, our method can extend the user authentication from the watch to the computer. Hence, the user can log in to the computer in a split second, and continue to hold and use the mouse as usual.

of using the devices is introduced. Then, the gesture detection algorithm and a new optimization based on rotation decomposition are described. The modified authentication procedure is summarized in Algorithm 3 (Section 5.5.

5.2 TwistIn Gesture

To authenticate only when the user purposely reaches for the device, the user is required to actively perform a gesture. This gesture has to be simple and intuitive to the user while at the same time rotates the devices sufficiently. When the user is wearing a smartwatch and holding a smart device, the most convenient way to rotate both devices at the same time is to twist both devices along the forearm; see figure 5.1 which shows a demonstration of logging into a computer by picking up a mouse and twist, assuming motion sensors are embedded into the mouse. We name this as the "TwistIn" gesture.

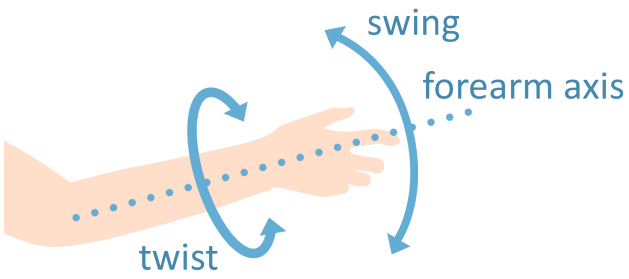


Figure 5.2: Twist about forearm (primary) and swing about wrist (secondary) during the TwistIn gesture.

5.3 Gesture Detection

If the motion of the two devices is in sync, it does not mean that the user performs the TwistIn gesture. In particular, when both devices are kept stationary, their motions are also in sync. In section 4.4, a method to check for device's motion is introduced so that stationary devices are ignored and not considered for authentication. Now that TwistIn is suggested, some patterns can be observed when a TwistIn gesture is performed. Hence, we can detect and see if the device motion is really a TwistIn gesture instead of simply checking if the device is stationary or not.

Note that the user wears a smartwatch and holds a smart device during the TwistIn gesture, and performs TwistIn by revolving the smart device about his/her forearm a few times from left to right and right to left; see Figure 5.2.

Here we replace the method introduced in section 4.4 with the following two methods: (i) analyze Δq over time, and (ii) count the number of twists within T_{auth}. We reuse T_{auth} as the time period in which we detect the TwistIn gesture which is defined in chapter 4.4.

5.3.1 Analyze Δq over time

Instead of analyzing q over time, we analyze Δq instead. This is because when a device is being twisted in a to-and-fro fashion, a great change in rotation direction can be observed within a small time frame. Similar to Section 4.4, we measure the change of device's rotation by performing dot products on consecutive data. The faster the user twists the device, the larger the dot product is. We compare the dot products against a threshold $\beta_{\text{move'}}$ to deduce if the device is sufficiently twisted.

$$\beta_{\text{move'}} < \sum_{t \in T_{\text{auth}}} 1 - \Delta q^t \cdot \Delta q^{t+1} \tag{5.1}$$

where Δq^t is given in Eq. (4.5).

5.3.2 Count the Number of Twists

Although the first method detects if twisting is performed to the devices, we cannot distinguish if the devices are twisted at the same time or not; this results in false positive if the devices are twisted at a similar magnitude, but different phases. In the second method, we count the number of twists and verify if the gesture is detected simultaneously at the devices.

For each device, we analyze the w-component of its orientation data (quaternions) and locate the turning points over the authentication period T_{auth}. In detail, turning points are time moments, where the gradient of the w-component changes sign. In other words, they are local minima or local maxima; see Figure 5.3(a)-(d). Note also that we use the w-component because its changes indicate the angular variations according to the quaternion formulation. In addition, if there are fewer than N turning points within T_{auth}, we regard the device as not performing the TwistIn gesture, e.g., see Figure 5.3(d). On the other hand, since there may be more than N turning points within

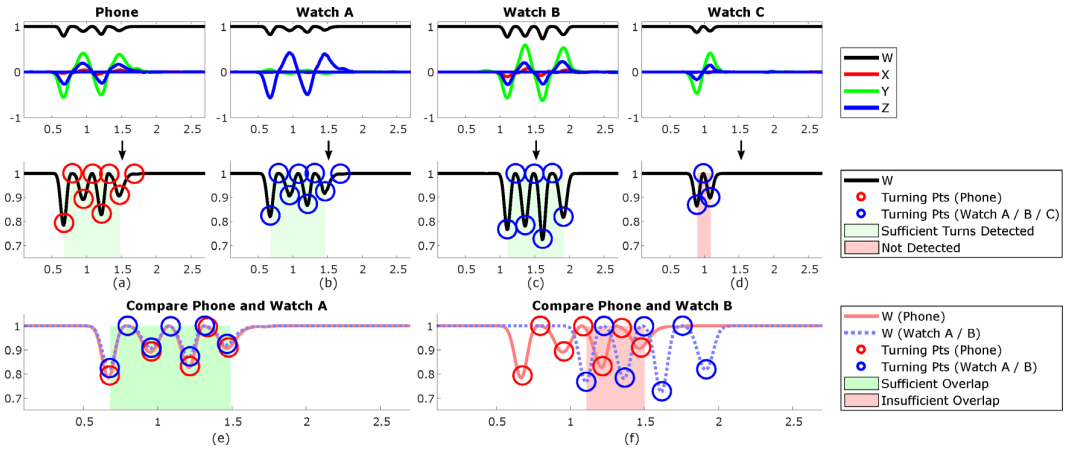


Figure 5.3: An illustration of rejecting invalid pairs of motion data by matching the turning points in time. We adopt two conditions for filtering. First, there should be at least seven turning points within the gesture detection period (T_{auth}); compare (a-c) with (d): watch C does not have sufficient turning points. Second, the turning point periods (T_{turning}) of the two devices should have sufficient overlap; compare (e) and (f): watch B's T_{turning} does not have sufficient overlap with the Phone's T_{turning}. Note also that the horizontal axes in these plots are time.

T_{auth}, for each device, we locate the sequence of N consecutive turning points that has the largest absolute oscillation magnitude in w compared to all other sequences within T_{auth}. We denote $T_{\text{turning}} \subset T_{\text{auth}}$ as the period between the first and last turning points in the sequence.

Next, if both the smart device (e.g., a smartphone) and the smartwatch have sufficient turning points within T_{auth}, we compare their T_{turning} periods and compute how much their T_{turning} periods overlap. If the overlap percentage is lower than β_{overlap} (which is empirically set as 60%), we regard the two devices as not performing the TwistIn gesture together. Note also that this turning point analysis procedure is fast to compute and can help to efficiently filter out motions that do not correspond to the TwistIn gesture.

5.4 Rotation Decomposition

As mentioned in section 4.9, the optimization introduced in section 4.6 Eq. (4.7) is vulnerable for two reasons: (i) the wrist rotation, which would affect the device held on user's hand (see introduction and Figure 5.2); and (ii) the elbow movement, which would affect both devices. Hence, we filter out the unwanted secondary rotations by formulating a rotation decomposition [24] in our optimization model.

The rotation decomposition is formulated as follows. Given a rotation represented as quaternion $q = [w, \vec{q}]$, where w is the scalar part and $\vec{q}$ is the vector part of the quaternion, and given a unit vector $\vec{a}$ as the axis of the primary rotation, we want to decompose q into two parts: (i) the primary rotation (denoted by quaternion p), which rotates around $\vec{a}$, and (ii) the secondary rotation (denoted by quaternion s), which does not rotate around $\vec{a}$.

Hence, $q = sp$, and p can be calculated as:

$$p = \frac{[w, (\vec{a} \cdot \vec{q})\vec{a}]}{\sqrt{w^2 + (\vec{a} \cdot \vec{q})^2}} \ , \tag{5.2}$$

which can be geometrically interpreted as a projection of a rotation q to an axis $\vec{a}$, and then followed by a normalization.

5.4.1 Reformulate the Optimization

In our problem, the primary rotation (twist) axis, which is common to both devices, is known, and they were to be solved through the optimization. Let $\vec{a_A}$ and $\vec{a_B}$ be the primary rotation axes of devices A and B defined in their own reference frames, respectively. In addition, we define the corresponding rotations around $\vec{a_A}$ and $\vec{a_B}$ as quaternions $\tilde{a}_A = [0, \vec{a_A}]$ and $\tilde{a}_B = [0, \vec{a_B}]$, respectively. Note also that r is the quaternion of the basis transformation between $\vec{a_A}$ and $\vec{a_B}$, so we have $\tilde{a}_A = r^{-1}\tilde{a}_B r$.

Suppose $\Delta q_A^t = [w_A^t, \vec{q_A^t}]$ and $\Delta q_B^t = [w_B^t, \vec{q_B^t}]$. We then reformulate the optimization model in Eq. (4.7) as:

$$\begin{aligned} \max_{\vec{a_A}, \vec{a_B}, r} \quad & \sum_{t \in T_{\text{basis}}} \langle p_A^t, r \ p_B^t \ r^{-1} \rangle \\ \text{subject to} \quad & ||\vec{a_A}|| = 1, ||\vec{a_B}|| = 1, \text{ and } ||r|| = 1, \end{aligned} \tag{5.3}$$

where $p_A^t = \frac{[w_A^t, (\vec{a_A} \cdot \vec{q_A^t})\vec{a_A}]}{\sqrt{(w_A^t)^2 + (\vec{a_A} \cdot \vec{q_A^t})^2}}$ and $p_B^t = \frac{[w_B^t, (\vec{a_B} \cdot \vec{q_B^t})\vec{a_B}]}{\sqrt{(w_B^t)^2 + (\vec{a_B} \cdot \vec{q_B^t})^2}}$.

Then, the objective function can be rewritten as

$$\sum_{t \in T_{\text{basis}}} \langle p_A^t \ r, r \ p_B^t \rangle \ . \tag{5.4}$$

Putting $\vec{a_A} = [a_x, a_y, a_z]$ and $\vec{a_B} = [b_x, b_y, b_z]$, we can then multiply out the

objective function in matrix form:

$$\begin{aligned}\langle p_A^t\ r, r\ p_B^t\rangle &= \frac{\begin{bmatrix} w_A^t & -a_x(\vec{a_A}\cdot\vec{q_A^t}) & -a_y(\vec{a_A}\cdot\vec{q_A^t}) & -a_z(\vec{a_A}\cdot\vec{q_A^t}) \\ a_x(\vec{a_A}\cdot\vec{q_A^t}) & w_A^t & a_z(\vec{a_A}\cdot\vec{q_A^t}) & -a_y(\vec{a_A}\cdot\vec{q_A^t}) \\ a_y(\vec{a_A}\cdot\vec{q_A^t}) & -a_z(\vec{a_A}\cdot\vec{q_A^t}) & w_A^t & a_x(\vec{a_A}\cdot\vec{q_A^t}) \\ a_z(\vec{a_A}\cdot\vec{q_A^t}) & a_y(\vec{a_A}\cdot\vec{q_A^t}) & -a_x(\vec{a_A}\cdot\vec{q_A^t}) & w_A^t \end{bmatrix} r}{\sqrt{(w_A^t)^2+(\vec{a_A}\cdot\vec{q_A^t})^2}} \\ &\cdot \frac{\begin{bmatrix} w_B^t & -b_x(\vec{a_B}\cdot\vec{q_B^t}) & -b_y(\vec{a_B}\cdot\vec{q_B^t}) & -b_z(\vec{a_B}\cdot\vec{q_B^t}) \\ b_x(\vec{a_B}\cdot\vec{q_B^t}) & w_B^t & -b_z(\vec{a_B}\cdot\vec{q_B^t}) & b_y(\vec{a_B}\cdot\vec{q_B^t}) \\ b_y(\vec{a_B}\cdot\vec{q_B^t}) & b_z(\vec{a_B}\cdot\vec{q_B^t}) & w_B^t & -b_x(\vec{a_B}\cdot\vec{q_B^t}) \\ b_z(\vec{a_B}\cdot\vec{q_B^t}) & -b_y(\vec{a_B}\cdot\vec{q_B^t}) & b_x(\vec{a_B}\cdot\vec{q_B^t}) & w_B^t \end{bmatrix} r}{\sqrt{(w_B^t)^2+(\vec{a_B}\cdot\vec{q_B^t})^2}} \\ &= \frac{w_A^t I+(\vec{a_A}\cdot\vec{q_A^t})M_A r}{\sqrt{(w_A^t)^2+(\vec{a_A}\cdot\vec{q_A^t})^2}} \cdot \frac{w_B^t I+(\vec{a_B}\cdot\vec{q_B^t})M_B r}{\sqrt{(w_B^t)^2+(\vec{a_B}\cdot\vec{q_B^t})^2}}\ , \end{aligned} \tag{5.5}$$

where we denote $M_A = \begin{bmatrix} 0 & -a_x & -a_y & -a_z \\ a_x & 0 & a_z & -a_y \\ a_y & -a_z & 0 & a_x \\ a_z & a_y & -a_x & 0 \end{bmatrix}$ and $M_B = \begin{bmatrix} 0 & -b_x & -b_y & -b_z \\ b_x & 0 & -b_z & b_y \\ b_y & b_z & 0 & -b_x \\ b_z & -b_y & b_x & 0 \end{bmatrix}$.

Then, we can further multiply out the terms in the numerators and obtain

$$\frac{w_A^t r\cdot w_B^t r+w_A^t r\cdot(\vec{a_B}\cdot\vec{q_B^t})M_B r+w_B^t r\cdot(\vec{a_A}\cdot\vec{q_A^t})M_A r+(\vec{a_A}\cdot\vec{q_A^t})(\vec{a_B}\cdot\vec{q_B^t})[(M_A r)\cdot(M_B r)]}{\sqrt{[(w_A^t)^2+(\vec{a_A}\cdot\vec{q_A^t})^2]\ [(w_B^t)^2+(\vec{a_B}\cdot\vec{q_B^t})^2)]}}\ . \tag{5.6}$$

In the above equation, the matrix-vector multiplication $M_A r$ and $M_B r$ are actually the multiplication of two quaternions, i.e.,

$$M_A r = r\tilde{a}_A \ \text{ and } \ M_B r = \tilde{a}_B r\ , \ \text{ where } \ \tilde{a}_A = [0, \vec{a_A}] \ \text{ and } \ \tilde{a}_B = [0, \vec{a_B}]\ .$$

Hence, we have $(M_A r)\cdot(M_B r) = (r\tilde{a}_A)\cdot(\tilde{a}_B r) = (r\tilde{a}_A r^{-1})\cdot(\tilde{a}_B)$. Note that the term $(r\tilde{a}_A r^{-1})$ is equivalent to applying rotation r to vector $\vec{a_A}$. Since there always exists an r, such that $r\tilde{a}_A r^{-1} = \tilde{a}_B$, the optimized value of $(M_A r)\cdot(M_B r)$

must be 1. In addition, $r \cdot r$ is obviously one, and for rM_Ar and rM_Br, if we multiply them out, we will find that $rM_Ar \ = \ rM_Br \ = \ 0$.

Therefore, we can simplify the optimization model in Eq. (5.3) as

$$\begin{aligned} \max_{\vec{a_A}, \vec{a_B}} \quad & \sum_{t \in T_{\text{basis}}} \frac{w_A^t w_B^t + (\vec{a_A} \cdot \vec{q_A^t})(\vec{a_B} \cdot \vec{q_B^t})}{\sqrt{[(w_A^t)^2 + (\vec{a_A} \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B} \cdot \vec{q_B^t})^2)]}} \\ \text{subject to} \quad & ||\vec{a_A}|| = 1 \ \text{ and } \ ||\vec{a_B}|| = 1 \ . \end{aligned} \tag{5.7}$$

5.4.2 Deriving the Optimization Solution

To solve the optimization problem, we iteratively apply the Lagrange multipliers. Let $\vec{q_A^t} = [x_A^t, y_A^t, z_A^t]$ and $\vec{q_B^t} = [x_B^t, y_B^t, z_B^t]$. We first initialize $\vec{a_A}^0$ and $\vec{a_B}^0$ as the means of the corresponding rotation axes:

$$\vec{a_A}^0 = \frac{1}{n} \sum_{t \in T_{\text{basis}}} \frac{\vec{q_A^t}}{||\vec{q_A^t}||} \quad \text{and} \quad \vec{a_B}^0 = \frac{1}{n} \sum_{t \in T_{\text{basis}}} \frac{\vec{q_B^t}}{||\vec{q_B^t}||} \ , \tag{5.8}$$

where n is the number of quaternion samples within time period T_{basis}. Then, we obtain the following equations using the Lagrange multipliers:

$$\begin{aligned} \vec{a_A}^{k+1} &= \begin{bmatrix} \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_B}^k \cdot \vec{q_B^t}) \ x_A^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \\ \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_B}^k \cdot \vec{q_B^t}) \ y_A^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \\ \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_B}^k \cdot \vec{q_B^t}) \ z_A^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \end{bmatrix} \\ \vec{a_B}^{k+1} &= \begin{bmatrix} \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_A}^k \cdot \vec{q_A^t}) \ x_B^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \\ \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_A}^k \cdot \vec{q_A^t}) \ y_B^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \\ \sum_{t \in T_{\text{basis}}} \frac{(\vec{a_A}^k \cdot \vec{q_A^t}) \ z_B^t}{\sqrt{[(w_A^t)^2 + (\vec{a_A}^k \cdot \vec{q_A^t})^2] \ [(w_B^t)^2 + (\vec{a_B}^k \cdot \vec{q_B^t})^2)]}} \end{bmatrix} \end{aligned} \tag{5.9}$$

followed by normalizations

$$\vec{a_A}^{k+1} = \frac{\vec{a_A}^{k+1}}{||\vec{a_A}^{k+1}||} \quad \text{and} \quad \vec{a_B}^{k+1} = \frac{\vec{a_B}^{k+1}}{||\vec{a_B}^{k+1}||} \ . \tag{5.10}$$

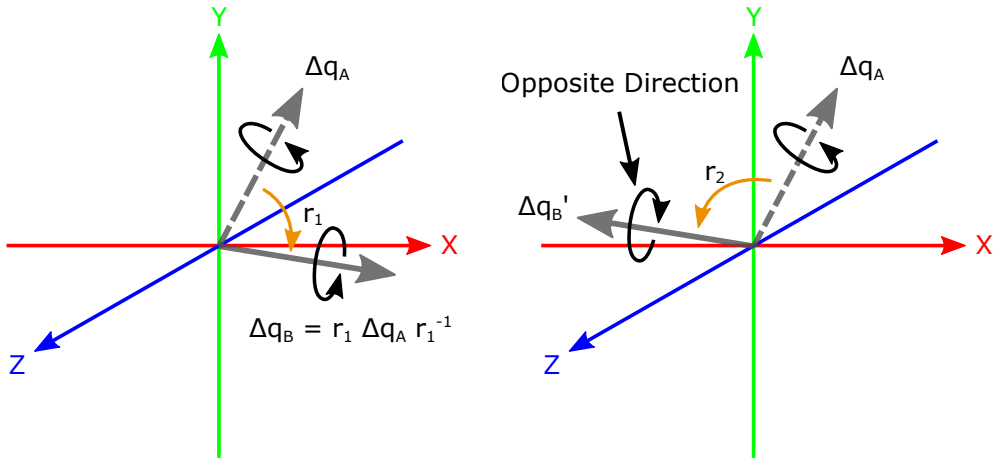


Figure 5.4: Noting that Δq_A is device A's rotation and Δq_B is device B's rotation, and each is expressed as a rotation around an axis. Applying r_1 or r_2 to transform Δq_A can result in Δq_B or $\Delta q'_B$, where $\Delta q'_B$ is $[(\Delta q_B)_w, -(\Delta q_B)_x, -(\Delta q_B)_y, -(\Delta q_B)_z]$.

We observed that $\vec{a_A}^k$ and $\vec{a_B}^k$ both converge in our experiments. In practice, a good approximation of the axes can be found in around three iterations.

After we obtain the optimization solutions $\vec{a_A}$ and $\vec{a_B}$, we have to account for the fact both $\vec{a_A}$ and $-\vec{a_A}$ are valid solutions for the rotation axes of device A, and similarly for $\vec{a_B}$ and $-\vec{a_B}$, as valid rotation axes of device B. Hence, r can be computed in four different ways using $\pm\vec{a_A}$ and $\pm\vec{a_B}$. As a result, we have:

$$r_1 = \frac{[\vec{a_A} \cdot \vec{a_B} + 1, \vec{a_A} \times \vec{a_B}]}{||[\vec{a_A} \cdot \vec{a_B} + 1, \vec{a_A} \times \vec{a_B}]||}, \; r_2 = \frac{[\vec{a_A} \cdot -\vec{a_B} + 1, \vec{a_A} \times -\vec{a_B}]}{||[\vec{a_A} \cdot -\vec{a_B} + 1, \vec{a_A} \times -\vec{a_B}]||} \; . \quad (5.11)$$

Lastly, we apply each of them to transform Δq_A, and check which of the transformed result leads to Δq_B to determine whether r_1 or r_2 is the actual basis transformation of r; see Figure 5.4.

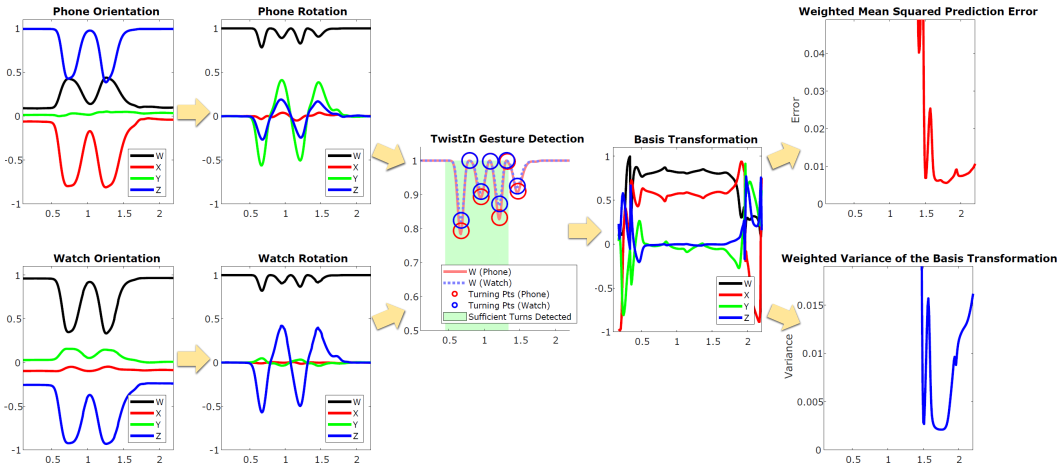


Figure 5.5: A running example showing the time series processing in the TwistIn Authentication Procedure.

5.5 Overall TwistIn Authentication Procedure

In this section, the overall TwistIn authentication procedure is presented, which is modified from the previous procedure; see Algorithm 3. The optimization model now features rotation decomposition and the motion detection is replaced with the gesture detection algorithm. Figure 5.5 presents a complete overview showing how the time series is calculated and processed.

ALGORITHM 3: Overall TwistIn Authentication Procedure

Input: Time series of Orientation data from Devices A and B

Output: Authentication State

Resample and smooth orientation data using Gaussian filtering (q_A^t and q_B^t)

Calculate device rotation (Δq_A^t and Δq_B^t)

*Calculate basis transformation using optimization model with rotation decomposition(r^t)

if **both devices have been sufficiently twisted by analyzing their turning points* **then**

  Calculate weighted mean squared prediction error

  Calculate weighted variance of basis transformation

  return true if both values are smaller than their respective thresholds

end

return false

*Changes are made on the lines marked with an asterisk.

5.6 Limitation

Including rotation decomposition into the optimization is only possible because the TwistIn gesture twists the devices around a single axis. The rotation caused by the wrist is minimal and can be filtered out. However, if the user performs TwistIn gesture while doing something else (e.g., turning around, running, etc.), the magnitude of the unwanted rotation will be comparable to the twist rotation. This affects the performance of the optimization which results in a higher rejection rate of the authenticate attempts. This is reflected in Experiment 3 detailed in the next chapter.

5.7 Conclusion

The shortcomings in the previous chapter are addressed in this chapter. The TwistIn gesture is presented with two gesture detection algorithms. The unwanted secondary rotation caused by the motion of the wrist is filtered out with an improved optimization using rotation decomposition. In the next chapter, the performance of all the previously mentioned methods is evaluated.

□ **End of chapter.**

Chapter 6

Performance Evaluation

Summary

The implementation details of the authentication procedures described in the previous chapters are presented here. They are evaluated in terms of accuracy and usability through a series of experiments. To evaluate the accuracy, motion samples collected from volunteers performing the TwistIn gesture are analyzed offline using a custom software. Then, the usability is evaluated with a user study where volunteers are interviewed.

6.1 Introduction

Since the proposed method is related to an interaction between people and computers (i.e., a Human-Computer Interaction), we have to check if it fits what users need or want, and whether it functions as expected. Hence, the performance of the methods is evaluated from two perspectives. Naturally, this chapter is divided into two parts:

- In the first part, the performance of the authentication techniques presented in the previous chapters are evaluated in terms of accuracy (i.e., the rate of successful authentication and prevention of hacking attempts). This verifies if the proposed method is feasible. A custom software is developed to search for the optimal parameters that maximize the performance for each method. The methods are then compared with each other.

- In the second part, the usability of the TwistIn gesture is evaluated (i.e., if the users like our method, or if they find it useful). This verifies if the proposed method is practical. The quality of a user's experience when interacting with the method is measured by comparing with other existing methods, such as using a PIN or via fingering recognition. A user study is conducted and volunteers are invited to an interview.

6.2 Evaluating Authentication Accuracy

To evaluate the accuracy, volunteers are invited to authenticate using the proposed methods and motion samples are collected from a smartphone and a smartwatch. The details of the hardware and data collection procedures are first described in Section 6.2.1. The evaluation methodology including the metrics and the technique used to search for the optimal parameters are explained in Section 6.2.2. A custom software developed for offline analysis is documented in Section 6.2.3. Finally, three experiments detailed in Section 6.3 to Section 6.5 are conducted to evaluate the performance of the proposed method under different scenarios.

6.2.1 Hardware and Data Collection

We implemented the prototype of TwistIn using Objective-C, and employ this prototype for two purposes in our experiments. First, it provides a user interface to demonstrate our method to the users. Second, it collects motion samples for offline data analysis and evaluation. Concerning the hardware, we used an iPhone 6S (iOS

Figure 6.1: Experiment setup using an iPhone 6S (iOS 10.3.3) and an Apple Watch 1 (watchOS 3.0). The series of images illustrate how a single twist is performed.

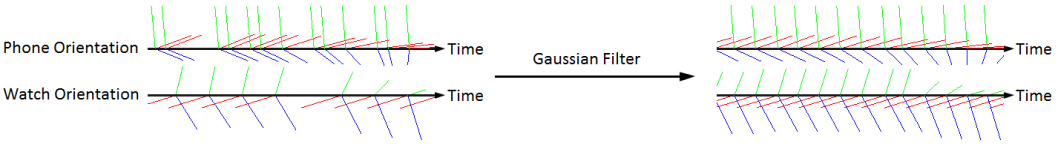


Figure 6.2: Using a Gaussian filter to resample and smooth the orientation data.

10.3.3) and an Apple Watch 1 (watchOS 3.0); see Figure 6.1. Both devices contain an accelerometer and a gyroscope, which measure the acceleration and angular velocity of the device. These data can then be fused into the orientation data using sensor fusion algorithms. The orientation data is represented in quaternion and can be retrieved at 100Hz directly using the Apple's Core Motion Framework. Since we paired the devices via Bluetooth and performed time synchronization prior to the experiment, we could stream the smartwatch's orientation data to the smartphone using Apple's WatchConnectivity Framework.

Note that even though each orientation data is associated with a timestamp, the data is not uniformly sampled over time, so the timestamps of the orientation data from the two devices may not match one another. Hence, we resample the quaternion data from each device and produce two series of uniformly-sampled and synchronized quaternion samples; see Figure 6.2 for an illustrative example. In addition, we should first ensure that the internal clocks of the two devices are synchronized when we start the systems. In detail, the resampling is done by using a Gaussian filter with σ set to be 0.03 seconds over a period of 0.1 seconds, and we perform a normalization on each resampled quaternion to ensure that it is a unit vector. After

the resampling, we obtain two time series of orientation data which can be analyzed and authenticate devices using the procedures described in previous chapters. We also archive the orientation time series from both devices as space-delimited files for further evaluation.

6.2.2 Evaluation Methodology

This section describes the methods and metrics used for evaluating the authentication accuracy.

Evaluation Metrics

We compare the performance using the following evaluation metrics.

- *True Positive Rate (TPR)*
  measures the success rate of accepting a positive case.
- *False Positive Rate (FPR)*
  measures the error rate of accepting a negative case.
- *False Negative Rate (FNR)*
  measures the error rate of rejecting a positive case.
- *True Negative Rate (TNR)*
  measures the success rate of rejecting a negative case.
- *Equal Error Rate (EER)*
  reflects the method's accuracy, which is defined as $EER = \frac{FPR+FNR}{2}$.

These evaluation metrics were calculated using the optimal parameters to perform the authentication procedure on all the motion sample pairs.

Optimal Parameters

The optimal parameters were obtained by exhaustively searching the parameter space using an approach similar to gradient descent, such that the performance was maximized, i.e., minimizing the squared error (MSE = $\sqrt{\text{FNR}^2 + \text{FPR}^2}$). Below, we recall all the parameters used in the methods presented in previous chapters:

T_{auth}	−−	time period taken to evaluate and detect the TwistIn gesture; see Section 4.4, 5.3.1 & 5.3.2;
T_{basis}	−−	time period taken to optimize for the basis transformation; see Section 4.6 & 5.4.1;
β_{move}	−−	threshold for sum of dot products of consecutive orientation q; see Section 4.4;
$\beta_{\text{move'}}$	−−	threshold for sum of dot products of consecutive orientation Δq; see Section 5.3.1;
N	−−	minimum number of turning points in a TwistIn gesture; see Section 5.3.2;
β_{overlap}	−−	minimum overlap percentage between the devices' turning period (T_{turning}); see Section 5.3.2;
β_{err}	−−	threshold for weighted mean squared prediction error; see Section 4.7, and;
β_{var}	−−	threshold for weighted variance of basis transformation; see Section 4.7.

Search Algorithm

There are two types of parameters, evaluation periods and thresholds. The evaluation periods determine how much data is used for calculation; a longer evaluation period gives a more accurate result, but may smooth out details. On the other hand, the thresholds are set to optimize performance, i.e., minimizing the MSE. The following algorithm is used to search for the optimal parameters:

1. The algorithm is initialized using an initial guess of evaluation periods and calculate the time series of Device Rotation, Basis Transformation, Mean Squared Prediction Error, and Variance of Basis Transformation.

2. For every case, the time series of Prediction Error and Variance are compared against an initial guess of thresholds. The number of positive cases that were failed to authenticate (i.e., False Negative) and the number of negative cases that were authenticated accidentally (i.e., False Positive) are recorded and the MSE is calculated.

3. A small constant step size is applied to the thresholds and a set of MSE's is obtained. The search of the parameter space is sped up by focusing only on the neighbors that produce a smaller MSE. This process is repeated until the MSE can no longer be further lowered.

4. Similarly, a small constant step size is applied to the evaluation periods to calculate new sets of time series, which are used to acquire the corresponding optimal thresholds and minimum MSE using the strategy outlined in the previous step.

5. Finally, the optimal parameters are obtained when the MSE cannot be further minimized.

Note that the solution is not unique (i.e. different sets of evaluation periods and thresholds could produce the same performance). In this case, we prefer the solution with the smallest evaluation periods and hence, shorter authentication time.

6.2.3 Offline Analysis Software Implementation

Overview

The search algorithm described is implemented in c++. There are three components in the program: (i) File Manager, (ii) Solver, and (iii) Exporter. The file manager is responsible for reading recorded files to memory, parsing data, and resampling. The solver searches for the optimal parameters and calculates the evaluation metrics. Finally, the exporter saves the output to a file. Figure 6.3 and Figure 6.4 shows the call graph and the UML class diagram of the program respectively.

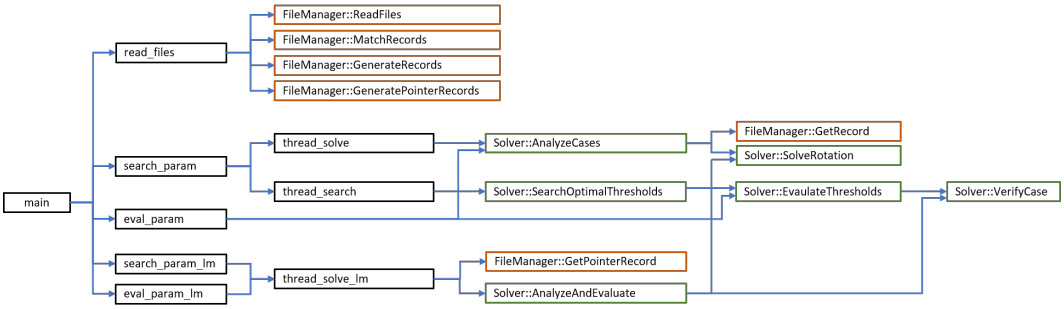


Figure 6.3: A simplified call graph of the program

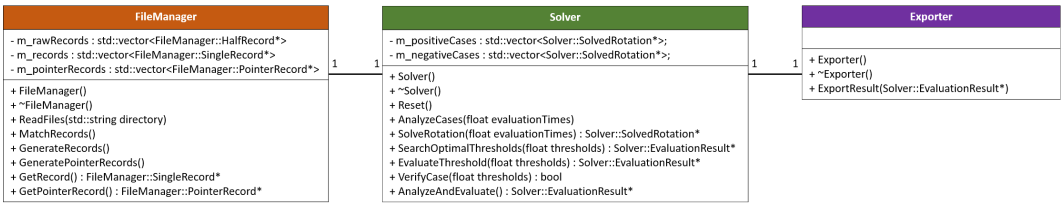


Figure 6.4: A simplified UML class diagram of the program

main

The *main* function reads files, analyzes, evaluate, and exports results according to the user's settings. There are mainly five modes to choose from:

1. *Search Mode*

   This mode searches for the optimal parameters that produce the best performance. To speed up the process, multiple threads are used.

2. *Evaluate Mode*

   This mode simply evaluates the performance using a set of predefined parameters.

3. *Search Mode (Less Memory)*

   This mode behaves similarly to the first mode, but less memory is used.

4. *Evaluate Mode (Less Memory)*

   This mode behaves similarly to the second mode, but less memory is used.

After specifying the mode, it creates threads and objects to analyze and evaluate.

File Manager

After a directory is provided, the File Manager reads all the text file inside the directory into memory. Each text file corresponds to one motion sample of one device, which consists of a time series of the device's orientation and its timestamps. Metadata such as device type and recording date and time is encoded in the filename. The file is parsed with smoothing into a list of float values, which is called a *HalfRecord* since each list refers to a motion sample obtained from one device only. After all the text files are parsed and stored into the container *m_rawRecords*, the *HalfRecords* are combined into *SingleRecords* using one of the methods below:

1. *Match Records*

   As mentioned in Section 6.2.1, one motion sample from a smartwatch and one from a smartphone are collected each time. These samples are simply paired up into a *SingleRecord* and stored in memory. They are considered True Case if both devices are used by the same person, otherwise, they are treated as False Case. Note that if the samples have various lengths, the longer one will be trimmed.

2. *Generate Records*

   In addition to the *SingleRecords* matched in the previous method, more False Cases are generated by combining *HalfRecords* that are not recorded at the same time. They are all stored in memory.

3. *Generate Pointer Records*

   Since the number of False Cases generated using the above method increases exponentially with the number of *HalfRecords*, which could lead to insufficient memory when analyzing a large number of samples. This method instead generate *PointerRecords* with pointers to the *HalfRecords*. They are combined into *SingleRecords* just before they are being analyzed and evaluated. This sacrifices speed for reduced memory consumption.

These *SingleRecords* and *PointerRecords* can then be retrieved using the functions

GetRecord() and *GetPointerRecord()* respectively. These records are read-only, meaning they can be accessed by multiple threads at the same time.

Solver

The Solver uses a set of predefined evaluation time to calculate the required rotations and exhaustively search for thresholds that maximize the performance. In detail, the *AnalyzeCases()* function retrieves all records from the File Manager and pass them to the *SolveRotation()* function, which then calculates the time series of Device Rotation, Basis Transformation, Mean Squared Prediction Error, and Variance of Basis Transformation using two time series of Device Orientation from each record and a given set of evaluation time. Each result is then saved inside a *SolvedRotation* structure. The *SearchOptimalThresholds()* function retrieves a list of *SolvedRotation* and search for the optimal thresholds by passing the list to the *EvaluateThresholds()* function with different sets of thresholds. The set of thresholds with the best performance is regarded as the optimal thresholds. Inside the *EvaluateThresholds()* function, each *SolvedRotation* structure is passed to the *VerifyCase* function with the given thresholds to determine if it is a valid authentication. The performance is calculated by counting the number of cases that are accepted or rejected successfully. In the case where *PointerRecoreds* are used, instead of processing in batches, the records are analyzed and evaluated one at a time in the *AnalyzeAndEvaluate()* function. This greatly reduces memory consumption as we do not store the intermediate data.

Exporter

This class simply exports results to a file. In the *ExportResult* function, the parameters and the evaluation metrics are exported to a file. There are also some debug settings which allow extra information to be exported, such as the performance corresponding to each set of parameters used during the search and the minimum MSE attained for each case.

6.3 Experiment 1: Determining the Number of Twists for a TwistIn Gesture

We performed an experiment to determine the number of twists for quantifying a motion as a TwistIn gesture as described in Chapter 5. Typically, a twist is a back-and-forth rotation about the forearm (see Figure 6.1), such that it starts with either a clockwise or anticlockwise rotation and then rotates back to the initial pose. Hence, having more twists could yield a longer time series with more rotations, thereby enhancing the gesture detection accuracy. However, the user may feel more tired and uncomfortable when performing the gesture. Therefore, we should minimize the number of twists required in a TwistIn gesture, while maintaining the performance at an acceptable level.

Procedure

To compare the effectiveness of using a different number of twists, we asked the participants to perform one to five twists and collected the motion samples using the following procedure:

(i) The TwistIn gesture was introduced to the participant.

(ii) The participant wore the smartwatch on his/her preferred hand tightly and picked up a smartphone.

(iii) The participant seated on a chair, and was given at least five minutes to try and practice the TwistIn gesture. We also reminded the participant to prevent any motion caused by moving the wrist.

(iv) The participant pressed the “Record” button on the smartphone.

(v) After three seconds, the smartphone vibrated, and signaled the start of the data recording.

(vi) The participant performed the TwistIn gesture with one twist.

(vii) The participant waited for around one second before pressing the "Complete" button.

(viii) The participant repeated steps (iv) to (vii) for ten times, and the whole experiment was repeated for four more times but the participant would perform TwistIn with a different number of twists.

Participants

We conducted the experiment with three participants. All are males aged from 26 to 28 (Mean = 27, SD = 1), and each performed the TwistIn gesture simultaneously on the two devices (smartphone and smartwatch) for ten times under five settings, i.e., one twist to five twists. Therefore, we collected 60 motion samples (3 participants $\times$ 2 devices $\times$ 10 times) for each setting. By pairing up the 30 motion samples from the smartwatch and 30 motion samples for the smartphone, where the two devices were actually twisted together, we can have 30 positive cases. On the other hand, we can have 1,740 negative cases formed by pairing up all motion samples and excluding the 30 positive cases, i.e., $\binom{60}{2}$ - 30, where $\binom{60}{2} = 1,770$. Note that since the motion sample pairs in the negative cases may not have the same time period (i.e., one sample could end earlier than the other), we thus look at each motion sample pair, shift their timestamps, and truncate the longer one, so that motion samples in each pair start at the same time and have the same time duration; otherwise, the negative cases will have an extremely low chance to produce the false positives observed in the results.

Discussion

We used the TwistIn algorithm described in Chapter 5 to analyze the motion samples for each setting (i.e., one twist to five twists). The TwistIn gesture is detected by counting the number of twists detailed in Section 5.3.2. The samples are evaluated using the program detailed in Section 6.2.3 and the result is shown in Table 6.1.

Table 6.1: EER achieved when the devices were twisted for different number of twists (#Twists).

#Twists	N	T_{basis}	T_{auth}	β_{overlap}	β_{err}	β_{var}	TPR	FPR	FNR	TNR	EER
1	3	0.30	1.00	0.65	0.035	0.0024	1.0000	0.0276	0.0000	0.9724	0.0138
2	7	0.10	1.20	0.55	0.045	0.0036	1.0000	0.0167	0.0000	0.9833	0.0083
3	11	0.10	1.50	0.55	0.054	0.0059	1.0000	0.0385	0.0000	0.9707	0.0193
4	15	0.10	2.20	0.45	0.047	0.0056	1.0000	0.0230	0.0000	0.9740	0.0115
5	19	0.10	2.30	0.60	0.035	0.0052	1.0000	0.0063	0.0000	0.9937	0.0032

Table 6.2: Effect of prolonged twisting studied by using different N, T_{basis} and T_{auth} on the data where the participants performed five twists.

#Twists	N	T_{basis}	T_{auth}	β_{overlap}	β_{err}	β_{var}	TPR	FPR	FNR	TNR	EER
5	3	0.30	1.00	0.50	0.026	0.0028	0.9667	0.1092	0.0333	0.8908	0.0713
5	7	0.10	1.20	0.60	0.030	0.0031	0.9667	0.0391	0.0333	0.9609	0.0362
5	11	0.10	1.50	0.70	0.040	0.0039	1.0000	0.0420	0.0000	0.9570	0.0210
5	15	0.10	2.20	0.55	0.042	0.0035	0.9667	0.0126	0.0333	0.9874	0.0230
5	19	0.10	2.30	0.60	0.035	0.0052	1.0000	0.0063	0.0000	0.9937	0.0032

From the result, we can see that TwistIn is capable of supporting fast and pretty accurate authentication with low false positive and false negative rates.

Next, we study the effect of prolonged twisting, meaning that the users perform excessive twists (#Twists) even the system demands fewer (N). To this end, we make use of the motion sample data where the users performed five twists, and then apply our method to detect the TwistIn gesture in these data with smaller "minimally required" N; see the 1st and 2nd columns in Table 6.2. Since T_{auth} depends on N and T_{basis} depends on T_{auth}, we reused the T_{auth} and T_{basis} from the corresponding row in Table 6.1; see the 3rd and 4th columns in Table 6.1 and Table 6.2. For parameters β_{overlap}, β_{err} and β_{var} listed in Table 6.2, we exhaustively searched for their values for optimal performance, for each case shown in each row in Table 6.2. Using these parameters, we applied our method to detect TwistIn gestures in the motion samples where the participants performed five twists, and obtained the results shown in the last five columns in Table 6.2. In the results, we observe a high FPR (0.1092) for a short evaluation period and a small N when $T_{\text{auth}} = 1.0s$ and $N = 3$. This shows that a small T_{auth} with a small N is vulnerable to attacks when the devices were twisted for a long period of time. To balance between convenience and security, we decided to use two twists as the minimum requirement in the TwistIn gestures.

6.4 Experiment 2: Comparing Performance of Different Methods

In the second experiment, we evaluated the performance of various methods mentioned in previous chapters. This includes a combination of methods used to detect motion and synchronous rotations.There are three methods that are proposed to detect motion or the TwistIn gesture: (i) Analyzing q over time (Section 4.4), (ii) Analyzing Δq over time (Section 5.3.1), and (iii) Counting the Number of Twists (Section 5.3.2). There are two methods that are proposed to detect synchronous rotations, one with rotation decomposition (RD) mentioned in Section 5.4 and the other without (Section 4.6). The following list shows all the methods that are compared in experiment 2.

1. Analyzing q over time without RD
2. Analyzing Δq over time without RD
3. Counting the Number of Twists without RD
4. Analyzing q over time with RD
5. Analyzing Δq over time with RD
6. Counting the Number of Twists with RD

Since all methods are capable of analyzing the TwistIn gestures, we compared the performance across the methods by evaluating motion samples with users performing the TwistIn gesture.

Procedure

We introduced and demonstrated the TwistIn gesture to the participants. We also reminded them to avoid wrist motions while performing the gesture. Then, they were given an iPhone 6S and an Apple Watch 1, and were asked to wear the watch tightly and to wear it on the same hand that holds the phone. Next, they were free

to hold and wear the devices in any location and orientation based on their own preference. A training session of at least five minutes was given to each of them to try and get familiar with the TwistIn gesture. After that, each participant was asked to perform the TwistIn gesture for ten trials under the following five scenarios:

(i) The watch was worn on the wrist of the preferred hand with the phone being held on the same hand with the screen facing upward. The participant was then signaled to perform the TwistIn gesture.

(ii) Same as scenario (i), but use the non-preferred hand instead.

(iii) The participant use the preferred hand (with the phone's screen facing upward), while standing. Again, the participant performed the TwistIn gesture.

(iv) Two participants were paired up. One acted as a normal user with the watch, while the other acted as an attacker with the phone. Both used the preferred hand with the devices and were positioned next to each other, so the attacker can clearly see the normal user and was free to place his/her hand in any location. Once a signal was given to the normal user, he/she performed the TwistIn gesture, while the attacker tried to mimic the normal user's motion via shoulder surfing.

(v) Scenario (i) was repeated again.

Similar to the previous experiment, the procedure of collecting each motion sample is as follows:

1. The participant pressed the "Record" button on the smartphone.

2. After three seconds, the smartphone vibrated, signaling the start of the recording.

3. The participant performed the TwistIn gesture by twisting the devices twice according to the requirements provided for each scenario.

Table 6.3: Parameters used for methods 1 to 6.

	Evaluation Periods		Motion/Gesture Detection				Authentication	
Method	T_{auth}	T_{basis}	β_{move}	$\beta_{\text{move'}}$	N	β_{overlap}	β_{err}	β_{var}
1	1.30	0.05	0.0005	–	–	–	0.0595	0.0200
2	0.55	0.45	–	0.0010	–	–	0.0740	0.0214
3	1.60	0.80	–	–	7	0.50	0.1025	0.1
4	0.70	0.05	0.0005	–	–	–	0.0650	0.0218
5	0.75	0.05	–	0.0005	–	–	0.0550	0.0244
6	1.20	0.10	–	–	7	0.60	0.0601	0.0188

4. The participant waited briefly for a second and then pressed the "Complete" button.

Participants

We invited 12 participants to Experiment 2, including nine males and three females aged from 19 to 31 (Mean = 24.75, SD = 3.415). Among them, only one participant owned a smartwatch; eight preferred to use the right hand to hold the phone while four preferred left. Note that the preferred hand is not necessarily the dominant hand.

Each participant performed the TwistIn gesture simultaneously on two devices for ten times for the above five different scenarios. We collected a total of 1200 motion samples (12 participants × 2 devices × 10 times × 5 scenarios), with 600 samples from the watch and 600 samples from the phone. There are 480 positive cases (for scenarios 1, 2, 3 and 5) consisted of samples where the two devices were twisted together by the same participant, and there are 120 negative cases for samples from scenario 4.

Table 6.4: Performance obtained for each scenario using method 1 to 6. Note that scenarios 1, 2, 3 and 5 consisted of positive cases while scenario 4 consisted of negative cases only. Therefore, the rates displayed in the table are TPR for scenarios 1, 2, 3 and 5 and TNR for scenario 4.

	Scenario	Method					
		1	2	3	4	5	6
(i)	Sitting (w/ preferred hand)	0.7917	0.7500	0.5917	0.8750	0.8417	0.9833
(ii)	Sitting (w/ non-preferred hand)	0.8333	0.7417	0.3333	0.9500	0.9583	0.9750
(iii)	Standing (w/ preferred hand)	0.7750	0.7667	0.3500	0.9417	0.9417	0.9833
(vi)	Attacking (w/ preferred hand)	0.8333	0.8583	0.9083	0.8750	0.9250	0.9417
(v)	Sitting (w/ preferred hand)	0.8667	0.9333	0.3750	0.9750	0.9750	0.9917
	Overall	0.8200	0.8100	0.5117	0.9233	0.9283	0.9750

Discussion

We evaluated the motion samples from each scenario with each method using the empirical parameters shown in Table 6.3. We then calculated the evaluation metrics (see Section 6.2.2) and tabulated the results in Table 6.4. Since scenarios 1, 2, 3 and 5 only consist of true cases while scenario 4 consists of false cases, only the TPR for scenarios 1, 2, 3 and 5 and TNR for scenario 4 is shown in the table respectively. From the overall performance of each method, applying Twist Decomposition and Counting Number of Twists performed the best, i.e., method 6 performs the best, with both TPR and TNR the highest. While method 3 performs poorly with its low TPR, its TNR is higher than method 1 and 2. This shows that Counting the Number of Twists is superior in defending against attacks attempts. A general improvement in methods 4 to 6 from methods 1 to 3 suggests that applying rotation decomposition to filter out unwanted rotations results in a better performance.

Then, we compared the performance across scenarios. First, we compared the effect of using the preferred hand and the non-preferred hand. A slight improvement in

TPR is observed in methods 1, 4, 5 but a decline in performance for methods 2, 3, and 6 is recorded. The insignificant differences between scenarios 1 and 2 suggested that authenticating using the preferred hand or the non-preferred hand resulted in similar performance, even though some participants mentioned and informed us that it was more difficult and less convenient to twist with the non-preferred hand. Second, we compared the effect of sitting and standing while performing the authentication. Similarly, mixed performance is obtained, indicating there is no significant difference between sitting and standing. Indeed, no participants reported any difference between the scenarios. On the other hand, there was a general improvement reported in scenario 5 when compared to scenario 1. This indicates the possibility of a learning trend, where the participants have learned to perform the TwistIn gesture. In scenario 4, we recorded a high TNR (i.e., a low FPR), especially in method 6. This showed that it was difficult to mimic other's TwistIn gesture. Still, a few attack attempts were successfully carried out by some of the participants. In particular, the attacker had a clear view of the phone and received the starting signal at the same time as the normal user. Moreover, the attacker also had ten trials of attacks. All these conditions were hard to fulfill without being noticed by the normal user. Therefore, we believe that the attack will be much harder in normal circumstances.

6.5 Experiment 3: Challenging Scenarios

In the last experiment, we additionally evaluated the performance of the proposed methods under five different challenging scenarios. Particularly, we considered all combinations of motion samples (across scenarios) that were not twisted together as negative cases.

Participants

We invited five participants who had participated in the second experiment. Each participant performed the TwistIn gesture simultaneously on the two devices for ten times under each of the five scenarios; see below. We collected a total of 500 motion samples (5 participants $\times$ 2 devices $\times$ 10 times $\times$ 5 scenarios), with 250 samples from the smartwatch and 250 samples from the smartphone. Therefore, there are 250 positive cases, which consist of motion sample pairs that were twisted together. Since we have collected 2,000 motion samples from experiments 1 to 3, we could further treat all combinations of the samples that were not twisted together as negative cases. Therefore, we have 1,998,000 negative cases, which is $\binom{2000}{2}$ - 1,000.

Procedure

We consider the following five different challenging scenarios:

(i) The participant holds the smartphone while walking back and forth, and performs the TwistIn gesture.

(ii) The participant holds the smartphone while jogging back and forth, and performs the TwistIn gesture.

(iii) The smartphone is placed on a desk. The participant sits on a chair next to it, and then picks up the smartphone while performing the TwistIn gesture.

(iv) The smartphone is placed in the pocket of the participant's trouser while the participant is standing. The participant takes out the smartphone and

Table 6.5: Performance obtained by applying the parameters in Table 6.3 using method 1 to 6 for the following scenarios.

	Scenario	Method					
		1	2	3	4	5	6
(i)	Walking	0.9000	0.9200	0.4400	0.8600	0.8600	0.6800
(ii)	jogging	0.8400	0.9200	0.7400	0.5600	0.5800	0.5000
(iii)	Picking up from desk	0.8800	0.9000	0.6400	0.9000	0.9600	0.8600
(vi)	Taking out from pocket	0.7400	0.8200	0.7600	0.7400	0.7400	0.7400
(v)	On a bus trip	0.7400	0.6800	0.600	0.8600	0.9200	0.9600
	Overall	0.8200	0.8480	0.6360	0.7840	0.8120	0.7480

Table 6.6: TNR and FPR evaluated with method 1 to 6 using motion samples from experiments 1 to 3. We considered all combinations of motion samples that were not twisted together as negative cases, so we considered 1,998,000 negative cases altogether.

	1	2	3	4	5	6
TNR	0.8773	0.9214	0.9493	0.8940	0.9267	0.9852
FPR	0.1227	0.0786	0.0507	0.1060	0.0733	0.0148

performs the TwistIn gesture at the same time.

(v) The participant holds the smartphone, while sitting inside a shuttle bus, whose route includes uphills, downhills, and turns. The participant performs the TwistIn gesture only when the bus is moving.

Note that we do not provide training sessions to the participants, as they had joined the second experiment and were familiar with the TwistIn gesture. Moreover, the participants used their preferred hand for all the scenarios.

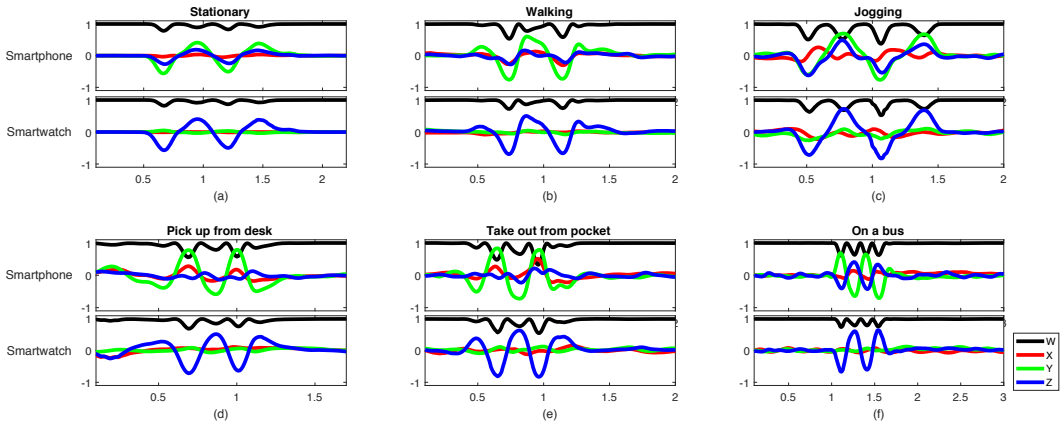


Figure 6.5: Plotting the device rotations in various scenarios. (a) is obtained from scenario 1 in Experiment 2, while (b)-(f) are obtained from scenarios 1 to 5 in Experiment 3, respectively. Note that in all figures, x-axes denote the time and y-axes denote the value of the xyzw-components in the quaternion samples.

Discussion

Similar to the previous experiments, we calculated the TPR, FPR, FNR, and TNR to evaluate our method's performance as in Section 6.2.2. We used the parameters listed in Table 6.3 to evaluate each method under each scenario and the result is tabulated in Table 6.5. The same parameters were also used to analyze the negative cases formed by pairing up all motion samples that were not twisted together, and the result is shown in Table 6.6. In general, the result of Experiment 3 has lower performance than Experiment 2. It is because in Experiment 2, the participant sat on a chair and held the phone on hand just before performing the TwistIn gesture. However, in Experiment 3, one common factor across the five scenarios is that they all include additional motions with extra rotations, when the user performs the TwistIn gesture, e.g., while walking, while picking up the phone from the trouser pocket, etc.

- In scenario 1, the participants performed the TwistIn gesture while moving around and making turns, which involve two different types of rotations. Ob-

viously, both devices rotated severely when the participants moved around and made turns. Also, due to the gravity acted on the smartphone, it was easy to move the wrist at the moment when the leg struck the ground. From Figure 6.5(b) and (c), some additional rotations can be observed in motions for the smartphone but not for the smartwatch.

- In scenarios 3 and 4, the smartphone was picked up from the desk and taken out from the trouser pocket, respectively. These actions will introduce additional rotations caused by the hinge joint.

- In scenario 5, the participants were on a bus, and they were allowed to perform the TwistIn gesture only when the bus was moving. Hence, extra rotations were introduced to the motion data when the bus made turns, went uphill, and went downhill.

While all five scenarios involve additional rotations, we could rank them in terms of the rotation magnitude. The bus produced the least rotations as it did not make a fast turn. Then, picking up the smartphone from the desk normally involves rotations of around 30 degrees. Taking the smartphone out from the trouser pocket and holding it up involves rotations of around 100 degrees. Making turns while walking and jogging could involve rotations as large as 180 degrees. On top of that, jogging suffers more from the extra rotations when the leg strikes the ground than walking.

This somehow matches the results for methods 4 to 6, where the participants performed better in scenario 3 and 5, and performed poorly in scenario 2 and 4. This is because the presence of an external rotation with a magnitude comparable to the twist rotation directly affects the performance of rotation decomposition as mentioned in Section 5.6. Hence, the calculated basis transformation is affected, which results in higher mean squared prediction error. On the other hand, methods 1 to 2 outperforms other methods because the rotation decomposition technique is not used, which The basis transformation is directly calculated using the obtained rota-

tion data. As long as the external rotation is present in both devices, the calculation of the basis transformation will not be affected.

Finally, the results in Table 6.6 shows that method 6 performed the best with an FPR as low as 0.0148. Indeed, counting the number of twists could effectively minimize the chance of unauthorized access, which is reflected by the low FPR for methods 3 and 6. On the contrary, simply checking if the devices are in motion or not yields the worst performance as shown in method 1 and 4, with method 1's FPR being 0.1227.

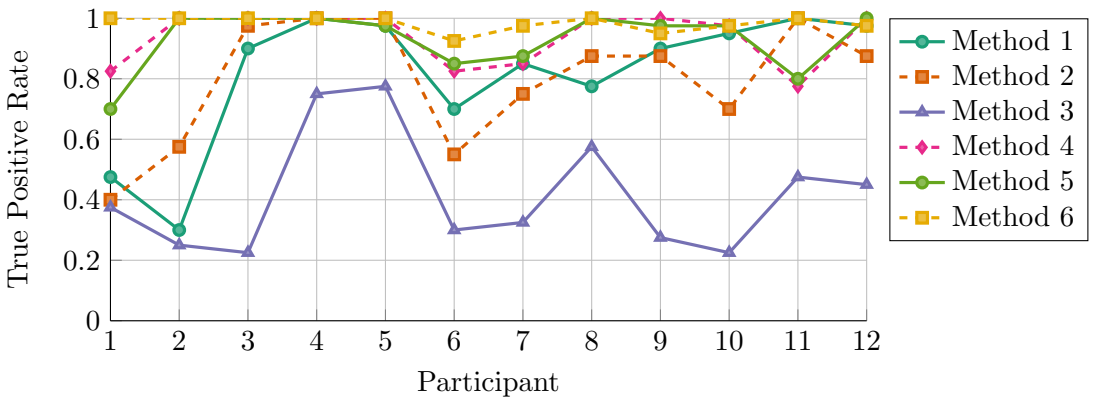
Table 6.7: Distribution of the participants' physical measurements.

Variable (cm)	Mean	S-D	Min.	Max.
Hand Length	18.75	1.422	17	21
Hand Span	20.05	1.735	17	23
Hand Circumference	19.17	1.337	17	22
Wrist Size	15.54	1.157	14	17
Minimum Devices Distance	6.583	2.466	4	11

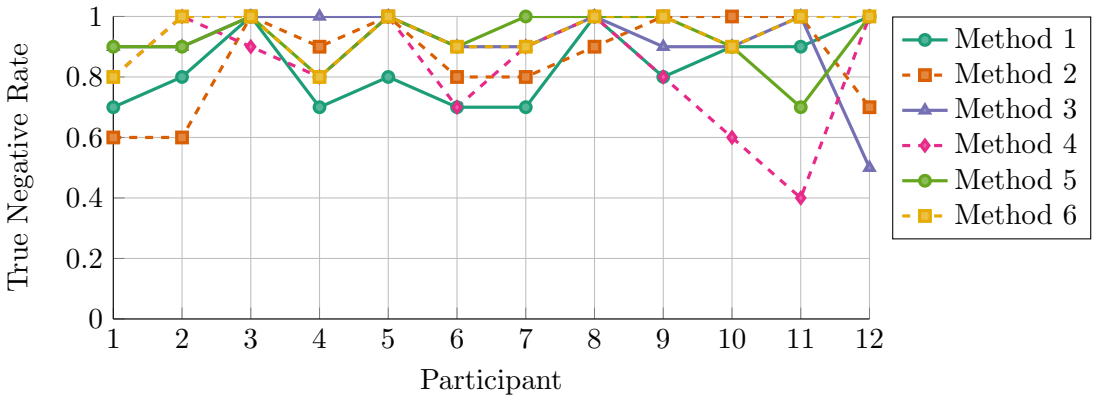
6.6 Evaluating Usability

The usability of the proposed method is evaluated by conducting a user study right after experiment 2 in Section 6.4. The same batch of participants in experiment 2 were asked eight questions concerning their subjective ratings on our method. In detail, each of them was asked to rate on a scale between -5 and +5, with +5 being strongly agree and -5 being strongly disagree. Additionally, we asked for any other feedback and comment towards our method. The eight questions were listed in Table 6.8.

We also measured the participants' hand length, hand span, hand circumference, wrist size, and the minimum distance from the smartphone to the smartwatch to check if the physical differences and the choice of wearing positions affect the performance. Note that the hand length was measured from the top of the middle finger to the base of the palm. The hand span was measured as the maximum distance from the thumb to the pinkie finger while stretching the hand. The hand circumference was measured around the meat of the hand, from where the pointer finger meets the palm to where the little finger meets the palm. The TPR and TNR each participant achieved overall in experiment 2 is plotted in Figure 6.6a and Figure 6.6b respectively.



(a) True positive rate of each participant achieved for methods 1 to 6. This refers to the success rate of authenticating the smartphone in scenarios 1, 2, 3, and 5 combined.



(b) True negative rate of each participant achieved for methods 1 to 6. This refers to the success rate of defending against attacks in scenario 4.

Figure 6.6: Performance of each participant achieved for methods 1 to 6.

(a) Tests for statistical significance using true positive rate.

	Method (True Positive Rate)					
	1	2	3	4	5	6
Linear Regression						
R^2	0.7880	0.3012	0.4748	0.0692	0.2157	0.4397
adjusted R^2	0.6113	-0.2811	0.0372	-0.7064	-0.4379	-0.0272
Hand Length	0.3049	0.7953	0.6257	0.7634	0.5814	0.7247
Hand Span	0.4388	0.9607	0.7601	0.9416	0.9289	0.8721
Hand Circumference	0.0115	0.2669	0.7512	0.9856	0.4451	0.2175
Wrist Size	0.2020	0.7146	0.8723	0.7211	0.5663	0.5729
Minimum Devices Distance	0.2374	0.5775	0.1396	0.6254	0.7010	0.3253
T-Test						
Preferred-hand	0.2870	0.5979	0.8962	0.6257	0.9414	0.0331

(b) Tests for statistical significance using true negative rate

	Method (True Negative Rate)					
	1	2	3	4	5	6
Linear Regression						
R^2	0.5585	0.5738	0.4069	0.5337	0.2226	0.4013
adjusted R^2	0.1906	0.2186	-0.0874	0.1451	-0.4252	-0.0975
Hand Length	0.4850	0.4889	0.1019	0.5168	0.5913	0.7671
Hand Span	0.5897	0.3895	0.1928	0.3712	0.4258	0.5507
Hand Circumference	0.5219	0.1413	0.4068	0.2257	0.7185	0.6013
Wrist Size	0.7587	0.0859	0.2456	0.0617	0.3249	0.7007
Minimum Devices Distance	0.3094	0.8358	0.3323	0.4384	0.7510	0.7956
T-Test						
Preferred-hand	0.5704	0.3246	0.3578	0.4313	0.5178	0.3604

Figure 6.7: The p-value calculated by running linear regressions on the participants' measurements with their performances and performing t-tests to determine if there are significant differences between participants who preferred the right hand and those who preferred the left hand.

6.6.1 Correlation between Performance and User Preference

We analyzed the performance statistically to study if there were any correlation between performance and the user preference. First, we looked for differences between participants who preferred the left hand and participants who preferred the right hand. We performed a t-test using their TPR and TNR; see Table 6.7a and Table 6.7b respectively. The p-values for TNR of methods 1 to 6 were higher than 0.05, meaning that the preference of the preferred hand did not affect the TNR. On the other hand, the p-values for TPR of methods 1 to 5 were also higher than 0.05, except method 6's, which was 0.0331. This suggested that the preference of the preferred hand could affect the performance of TwistIn. A closer look at the results also reveals that the participants who preferred to use their left hand had successful authentications in all of the trials without failing. However, the accuracy of the test was affected by the small sample size. We can perform a larger study involving more participants to better understand the effect of the preferred hand in the future.

Second, we study the effect of the smartwatch's wearing position. Since the smartphone is always held by the hand but the smartwatch can be worn at different locations on the forearm, we used the minimum device distance to represent the smartwatch's position. We performed a linear regression on the Minimum Devices Distance with the participants' individual TPR and TNR; again, see Table 6.7a and Table 6.7b respectively. Since the p-values for all methods were greater than 0.05, we rejected the null hypothesis and concluded that there was no significant correlation between the smartwatch's wearing position and the performance.

Third, we originally planned to see if the wearing orientation of the smartwatch and the holding orientation of the smartphone have any effect on the performance. However, all participants decided to use the same orientation for both devices. Hence, we cannot perform this analysis, which would require a larger scale study.

Lastly, we tried to study the effect of the initial twisting direction to the performance. But we noticed that if the smartphone was held facing upward initially, the initial twisting direction was always toward inwards, i.e., if the smartphone was held by the right hand, the monitor would point to the left, and if the smartphone was held by the left hand, the monitor would point to the right. Hence, it was impossible to twist in the opposite direction due to the ergonomics of the human forearm. Therefore, it is not an option in the TwistIn gesture.

6.6.2 Correlation between Performance and Physical Differences

To find out if the physical differences affect the performance, we measured the hand size and the wrist size of the participants. We then performed linear regression with the participants' individual TPR and TNR, and obtained the results tabulated in Table 6.7a and Table 6.7b respectively. Most of the p-values are greater than 0.05, and we can reject the null hypothesis for those cases. However, the p-value obtained using method 1's TPR and the hand circumference is 0.0115, which is lower than 0.05. This suggests the existence of a correlation between the hand circumference with the performance; since a bigger hand can grab a device more tightly and stable. However, the result could also be affected by the small sample size. Therefore, a more comprehensive user study should be performed with a larger population before we draw any conclusion. Right now, we cannot conclude if the participants' performances had any correlation with their physical differences.

6.6.3 Feedback from participants

We interviewed the participants to rate our method and compare with others. The results are tabulated in Table 6.8, which shows that the participants favored our method over password (Mean = 3.5, SD = 1), PIN (Mean = 2.7, SD = 2.1) and swipe pattern (Mean = 3.1, SD = 1.7). In particular, the participants favored our method mostly because it does not require memory recall. They also mentioned that the three methods are vulnerable against shoulder surfing and smudge attack,

Table 6.8: Questions and responses of the interview on the scale of -5 (being strongly disagree) to +5 (being strongly agree).

Question	Mean	S-D	Min.	Max.
I like to log into a computer by performing the TwistIn gestures on a mouse more than typing your username and password on a keyboard.	3.5	1	2	5
I like to log into a smartphone by performing the TwistIn gestures more than entering a PIN.	2.7	2.1	-1	5
I like to log into a smartphone by performing the TwistIn gestures more than drawing a Swipe Pattern.	3.1	1.7	-1	5
I like to log into a smartphone by performing the TwistIn gestures more than using Fingerprint Authentication.	-2	2.4	-5	2
I like to log into a smartphone by performing the TwistIn gestures more than shaking the smartphone continuously for 2s (i.e., ShakeUnlock).	4.3	0.9	3	5
I like to log into a smartglasses (e.g., Google Glass) by performing the TwistIn gestures more than entering a combination of four gestures at the touchpad.	3.3	2.1	-2	5
I like to access an IoT device by performing the TwistIn gestures and using the smartwatch's interface to control more than searching for the corresponding application on a smartphone and using it to control the device.	0.7	3.1	-4	5
Overall, I enjoy using TwistIn to log in and access devices.	2.8	1.1	0	4

which are less secure compared to ours. ShakeUnlock (Mean = 4.3, SD = 0.9) was not preferred by the participants as more effort is needed to shake continuously for 2s compared to our TwistIn gesture, which can be done in less than 1.2s.

On the contrary, participants preferred fingerprint recognition (Mean = -2, SD = 2.4) because it is faster and more straightforward than our method. However, one participant pointed out that our method allows authentication with multiple users (e.g., Apple's TouchID can only recognize five fingerprints), which could be more useful in some situations. Moreover, some participants mentioned that fingerprint recognition does not work for wet/dirty fingers. In addition, our method requires no external hardware menu and buttons on the device's surface. This suggests our method can complement with fingerprint recognition.

Wearable devices are small and are usually equipped with minimal interfaces. These devices cannot be authenticated with PINs or Swipe Patterns if they do not have a keypad or a touchscreen. For example, Google Glass uses a combination of tap and swipe gestures on the touchpad to authenticate, which is slow and troublesome. On the other hand, our method allows users to simply pick up the smartglasses, perform the TwistIn gesture, and use it. Therefore, our method was highly rated when we compared with Google Glass (Mean = 3.3, SD = 2.1).

Traditionally, people connect to smart devices through smartphone applications. This allows us to use the smartphone interface to remotely access and control the devices. However, if a small device is close to the user, it can be picked up and logged in using the TwistIn gesture. The smartwatch's interface can then be immediately used to access the device. This saves the time for searching for the corresponding app on the smartphone. Mixed responses (Mean = 0.7, SD = 3.1) were received as some participants did not like the idea of having to physically move and pick up the device, while some participants liked to physically engage with the device and use

it because they felt that the interaction is more natural.

6.7 Conclusion

The performance of the presented methods is evaluated in terms of accuracy and usability. The final proposed method achieved state-of-the-art performance. The user study also shows our method is preferred over many other existing methods. While it cannot outperform fingerprint recognition, our method is still superior in a number of circumstances, such as when the hands are dirty where fingerprint recognition would fail. In the next and final chapter, we conclude this thesis with a discussion limitation and future work of this study.

□ **End of chapter.**

Chapter 7

Conclusion

This thesis aims at exploring the application of a smartwatch as an authentication token to tangibly gain quick access to other smart devices in a multi-user multi-devices environment. In particular, a gesture is introduced to facilitate the process. While this method achieves good performance in both accuracy and usability, there are still some limitations that yet has to be overcome. In this final chapter, the limitation and future works are discussed.

Limitation and Future Works

As we implemented and tested TwistIn, we noticed some limitations in our method and experiment design that can be improved. First, while we have collected over 2,000 motion samples, our experiment result is still affected by the small sample size. In addition, we only used two devices in the experiments. However, the difference in size and shape may also affect our method's performance. We plan to perform a more comprehensive study with more participants and more different devices.

Second, as shown in Experiment 3, our method has lower performance when the user performs TwistIn while doing other activities at the same time. We plan to explore the use of deep learning in analyzing the motion data. To train a network capable of performing authentication in a short time frame, we have to extract re-

gions of interest from the collected motion samples. One possible way is to run our proposed algorithm and locate the segment with the minimum MSE in each pair of motion samples for both positive cases and negative cases. These segments are used to train a network for authentication with a mix of convolutional neural networks and recurrent neural networks to generate a feature vector for each motion sample, and then minimize/maximize the distance between the pair of vectors for the positive/negative cases. To verify the accuracy of the authentication network, we can feed successive segments of each pair of motion samples into the network. The network should be able to identify valid authentication attempt in at least one of the segments for a positive case, but none in a negtive case. We believe this approach could achieve a better performance in both experiments 2 and 3.

Third, our current method considers only the motion data in the authentication process. In fact, we can incorporate other metrics. For example, we may include proximity-based detection, so that people far away cannot gain unauthorized access to the smart devices. In addition, we may consider of incorporating biometrics such as the twisting frequency and magnitude into the algorithm. We have to analyze if these biometrics are unique from person to person and is always consistent with the same person at all times.

Fourth, we plan to investigate the hardware and software implementation to make use of TwistIn for IoT devices, e.g., game controllers, so that players can immediately join in a game simply by twisting the game controller. More importantly, the player's profile and preference can then be loaded automatically to provide an instantaneous customized experience. This also matches the real-life situation that most game controllers such as PlayStation 4 Controller and Xbox One Controller do not have a touchscreen. Therefore, the TwistIn also facilitate fast access and login, as well as personalization for the game player. Another example is a smart wallet with RFID support which can be used for touchless transactions. It can be

enabled only a TwistIn authentication is performed, effectively protecting against RFID skimming.

Lastly, our ultimate goal is to completely remove the TwistIn gesture so that the device can be authenticated immediately right after it is picked up by the user. It is an ambitious target which has two challenging problems that have to be tackled. One problem is to classify the user's intention; whether the user intends to access the device or not. Currently, we asked the user to use the TwistIn gesture to specific their intention to use the device. Without the gesture, we need to design another way to distinguish if the user is really picking up the device to use, or it is just some random movements caused by activities such as walking. We will explore the use of machine learning approaches on this matter. The second problem is to create a one-to-one correspondent for the smartwatch and the device in a situation when many users are picking up devices at the same time (i.e., in a multi-user multi-device environment). As shown in our result, a pure synchronous rotation detection results in an unacceptable false positive rate. Inspired by Gierad's work [31], we plan to use a high frequent AHRS sensor to capture features within a user's motion. We believe these features can be utilized to construct the correspondent.

Conclusion

In this thesis, we introduce the TwistIn gesture to tangibly authenticate devices using a smartwatch. We observe that two devices share the same rotation when they are interacted using a single hand, but the difference in reference frame causes divergent results. To find the transformation between the reference frames, we use an optimization to solve for the basis transformation. We also notice that there is a discrepancy in the two motions when one of the devices is strapped on the forearm and the other is being held in hand. To minimize the impact caused by the discrepancy, we reformulated the optimization by incorporating rotation decomposition to filter out unwanted rotation components. Furthermore, by analyzing the turning

points and the variation of basis transformations over time, we designed and developed an algorithm for detecting synchronous TwistIn gesture among multiple smart devices. The algorithm was implemented into a prototyping iOS Application, and we evaluated its performance with 12 participants under 5 different scenarios, and collected 1200 motion samples. Our evaluation with the prototype system reports that the true positive and true negative rates are only 0.9417 and 0.9833 (see Table 6.4), suggesting that this method can be efficiently adopted in small devices for allowing fast access to the devices.

Publication Related to this Work

Leung, H. M. C., Fu, C. W., and Heng, P. A. (2018). TwistIn: Tangible Authentication of Smart Devices via Motion Co-analysis with a Smartwatch. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Volume 2, Issue 2, Article 72. Presented in UbiComp 2018.

□ **End of chapter.**

Bibliography

[1] R. Amadeo. Meet google's "eddystone" - a flexible, open source ibeacon fighter: The ars technica review, 2015. [Online; accessed 17-April-2018].

[2] D. Amitay. Most common iphone passcodes. *Retrieved June*, 15:2011, 2011.

[3] I. Apple. About ibeacon on your iphone, ipad, and ipod touch, 2017. [Online; accessed 17-April-2018].

[4] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge attacks on smartphone touch screens. *Woot*, 10:1–7, 2010.

[5] N. Ben-Asher, N. Kirschnick, H. Sieger, J. Meyer, A. Ben-Oved, and S. Möller. On the need for different security methods on mobile phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 465–473. ACM, 2011.

[6] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. On the use of sift features for face authentication. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 35–35. IEEE, 2006.

[7] J. Camhi. Bi intelligence projects 34 billion devices will be connected by 2020, 2015. [Online; accessed 27-February-2017].

[8] B. Chang, X. Liu, Y. Li, P. Wang, W.-T. Zhu, and Z. Wang. Employing smartwatch for enhanced password authentication. In *International Conference*

on Wireless Algorithms, Systems, and Applications, pages 691–703. Springer, 2017.

[9] X. A. Chen, T. Grossman, D. J. Wigdor, and G. Fitzmaurice. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 159–168. ACM, 2014.

[10] A. Cunningham and L. Hutchinson. macOS 10.12 Sierra: The ars technica review, 2016. [Online; accessed 16-April-2018].

[11] Y. Du, H. Ren, G. Pan, and S. Li. Tilt & touch: Mobile phone for 3d interaction. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 485–486. ACM, 2011.

[12] S. Egelman, S. Jain, R. S. Portnoff, K. Liao, S. Consolvo, and D. Wagner. Are you ready to lock? In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 750–761. ACM, 2014.

[13] R. D. Findling, M. Muaaz, D. Hintze, and R. Mayrhofer. Shakeunlock: Securely unlock mobile devices by shaking them together. In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, pages 165–174. ACM, 2014.

[14] M. Frank, 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.

[15] I. Gartner. Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016, 2017. [Online; accessed 10-May-2017].

[16] A. Ghose, C. Bhaumik, and T. Chakravarty. BlueEye: A system for proximity detection using bluetooth on mobile phones. In *Proceedings of the 2013 ACM*

conference on Pervasive and ubiquitous computing adjunct publication, pages 1135–1142. ACM, 2013.

[17] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[18] J. Gong, L. Li, D. Vogel, and X.-D. Yang. Cito: An actuated smartwatch for extended interactions. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5331–5345. ACM, 2017.

[19] J. J. Guiry, P. van de Ven, and J. Nelson. Multi-sensor fusion for enhanced contextual awareness of everyday activities with ubiquitous devices. *Sensors*, 14(3):5687–5701, 2014.

[20] K. Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 149–158. ACM, 2003.

[21] K. Hinckley and H. Song. Sensor synaesthesia: touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 801–810. ACM, 2011.

[22] J. Hong, S. Heo, P. Isokoski, and G. Lee. Splitboard: A simple split soft keyboard for wristwatch-sized touch screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1233–1236. ACM, 2015.

[23] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[24] C. Hu, M. Q.-H. Meng, M. Mandal, and P. X. Liu. Robot rotation decomposition using quaternions. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pages 1158–1163. IEEE, 2006.

[25] W. Hutama, P. Song, C.-W. Fu, and W. B. Goh. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3315–3318. ACM, 2011.

[26] A. H. Johnston and G. M. Weiss. Smartwatch-based biometric gait recognition. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–6. IEEE, 2015.

[27] J.-W. Kim, H.-J. Kim, and T.-J. Nam. M. gesture: An acceleration-based gesture authoring system on multiple handheld and wearable devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2307–2318. ACM, 2016.

[28] F. Klompmaker, K. Nebe, and J. Eschenlohr. Towards multimodal 3d tabletop interaction using sensor equipped mobile devices. *Mobile Computing, Applications, and Services*, pages 100–114, 2013.

[29] S. Kratz, M. Rohs, and G. Essl. Combining acceleration and gyroscope data for motion gesture recognition using classifiers with dimensionality constraints. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 173–178. ACM, 2013.

[30] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[31] G. Laput, R. Xiao, and C. Harrison. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 321–333. ACM, 2016.

[32] G. Laput, Y. Zhang, and C. Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3986–3999. ACM, 2017.

[33] S. Lee, Y. Kim, D. Ahn, R. Ha, K. Lee, and H. Cha. Non-obstructive room-level locating system in home environments using activity fingerprints from smartwatch. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 939–950. ACM, 2015.

[34] J. Lester, 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*, pages 33–50. Springer, 2004.

[35] J. Lockman, R. S. Fisher, and D. M. Olson. Detection of seizure-like movements using a wrist accelerometer. *Epilepsy & Behavior*, 20(4):638–641, 2011.

[36] K. Lyons. What can a dumb watch teach a smartwatch?: Informing the design of smartwatches. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 3–10. ACM, 2015.

[37] S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 25, 2010.

[38] T. Maekawa, D. Nakai, K. Ohara, and Y. Namioka. Toward practical factory activity recognition: unsupervised understanding of repetitive assembly work in a factory. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1088–1099. ACM, 2016.

[39] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda. An extended kalman filter for quaternion-based orientation estimation using marg sensors. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2003–2011. IEEE, 2001.

[40] I. Natgunanathan, A. Mehmood, Y. Xiang, G. Beliakov, and J. Yearwood. Protection of privacy in biometric data. *IEEE access*, 4:880–892, 2016.

[41] I. Oakley, D. Lee, M. Islam, and A. Esteves. Beats: Tapping gestures for smart watches. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1237–1246. ACM, 2015.

[42] S. Olberding, S. Soto Ortega, K. Hildebrandt, and J. Steimle. Foldio: Digital fabrication of interactive and shape-changing objects with foldable printed electronics. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 223–232. ACM, 2015.

[43] J. Pearson, S. Robinson, and M. Jones. It's about time: Smartwatches as public displays. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1257–1266. ACM, 2015.

[44] S. Pizza, B. Brown, D. McMillan, and A. Lampinen. Smartwatch in vivo. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5456–5469. ACM, 2016.

[45] M. Rofouei, A. Wilson, A. Brush, and S. Tansley. Your phone or mine?: fusing body, touch and device sensing for multi-user device-display interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1915–1918. ACM, 2012.

[46] J. Ruiz and Y. Li. Doubleflip: a motion gesture delimiter for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2717–2720. ACM, 2011.

[47] J. Ruiz, Y. Li, and E. Lank. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 197–206. ACM, 2011.

[48] F. Schaub, R. Deyhle, and M. Weber. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th international conference on mobile and ubiquitous multimedia*, page 13. ACM, 2012.

[49] D. Schmidt, F. Chehimi, E. Rukzio, and H. Gellersen. Phonetouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 13–16. ACM, 2010.

[50] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns: Integrating security and systems engineering.* John Wiley & Sons, 2013.

[51] C. Shemelya, F. Cedillos, E. Aguilera, D. Espalin, D. Muse, R. Wicker, and E. MacDonald. Encapsulated copper wire and copper mesh capacitive sensing for 3-d printing applications. *IEEE Sensors Journal*, 15(2):1280–1286, 2015.

[52] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426, 2016.

[53] P. Song, W. B. Goh, C.-W. Fu, Q. Meng, and P.-A. Heng. Wysiwyf: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1333–1342. ACM, 2011.

[54] X. Su, H. Tong, and P. Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, 2014.

[55] K. Sun, Y. Wang, C. Yu, Y. Yan, H. Wen, and Y. Shi. Float: One-handed and touch-free target selection on smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 692–704. ACM, 2017.

[56] V. H. Tran, K. T. Choo, Y. Lee, R. C. Davis, and A. Misra. Magi: Enabling multi-device gestural applications. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[57] S. Uellenbeck, M. Dürmuth, C. Wolf, and T. Holz. Quantifying the security of graphical passwords: the case of android unlock patterns. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 161–172. ACM, 2013.

[58] D. Van Bruggen, S. Liu, M. Kajzer, A. Striegel, C. R. Crowell, and J. D'Arcy. Modifying smartphone user locking behavior. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 10. ACM, 2013.

[59] A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara. Amigo: Proximity-based authentication of mobile devices. In *International Conference on Ubiquitous Computing*, pages 253–270. Springer, 2007.

[60] G. Wilkinson, A. Kharrufa, J. Hook, B. Pursglove, G. Wood, H. Haeuser, N. Y. Hammerla, S. Hodges, and P. Olivier. Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2832–2844. ACM, 2016.

[61] K. Wolf and J. Willaredt. Pickring: seamless interaction through pick-up detection. In *Proceedings of the 6th Augmented Human International Conference*, pages 13–20. ACM, 2015.

[62] C. Xu, 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*, pages 9–14. ACM, 2015.

[63] J. Yang, Y. Li, and M. Xie. Motionauth: Motion-based authentication for wrist worn smart devices. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 550–555. IEEE, 2015.

[64] H. Zhu, J. Hu, S. Chang, and L. Lu. Shakein: Secure user authentication of smartphones with habitual single-handed shakes. *IEEE Transactions on Mobile Computing*, 16(10):2901–2912, 2017.