TwistIn: Tangible Authentication of Smart Devices via Motion Co-analysis with a Smartwatch

HO-MAN COLMAN LEUNG, The Chinese University of Hong Kong, Hong Kong
CHI-WING FU, The Chinese University of Hong Kong, Hong Kong
PHENG-ANN HENG, The Chinese University of Hong Kong, Hong Kong

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 paper presents *TwistIn*, a simple gesture that takes a smartwatch as an authentication token for fast access and control of other smart devices. Our mechanism is particularly useful for devices such as smartphones, smart glasses, and small IoT objects. To log in a device, one simply need 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. 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 thousands of gesture samples, and conducted various experiments to evaluate our prototype system and show that it achieved over 95% detection accuracy.

CCS Concepts: • **Security and privacy → Authentication**; • **Human-centered computing → Gestural input**;

Additional Key Words and Phrases: Tangible authentication, motion analysis, cross-device interaction, smart device

ACM Reference Format:
Ho-Man Colman LEUNG, Chi-Wing FU, and Pheng-Ann HENG. 2018. TwistIn: Tangible Authentication of Smart Devices via Motion Co-analysis with a Smartwatch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 72 (June 2018), 24 pages. https://doi.org/10.1145/3214275

1 INTRODUCTION

Technological advancements have brought to us more compact and cost-effective electronics. Nowadays, a large number of devices are augmented with smart functionalities. It is expected that such trend will continue to grow with two emerging technologies. The first is "Internet of Things" (IoT), which connects the smart devices around us. Recent studies have forecasted that there will be more than 20 billion connected devices by 2020 [7, 15]. The second is 3D-printing, which has been evolved to contain electronic circuits [33, 39]. 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.

To authenticate smart devices, current methods either handle the devices one at a time, or are not suitable for environments with multiple users and multiple devices. For instance, a simple approach is to build a hard button menu or a touchscreen on a device for users to log on the device via a PIN or passcode, or via 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

Authors' addresses: Ho-Man Colman LEUNG, The Chinese University of Hong Kong, Department of Computer Science and Engineering, Shatin, N.T., Hong Kong; Chi-Wing FU, The Chinese University of Hong Kong, Department of Computer Science and Engineering, Shatin, N.T., Hong Kong; Pheng-Ann HENG, The Chinese University of Hong Kong, Department of Computer Science and Engineering, Shatin, N.T., Hong Kong.

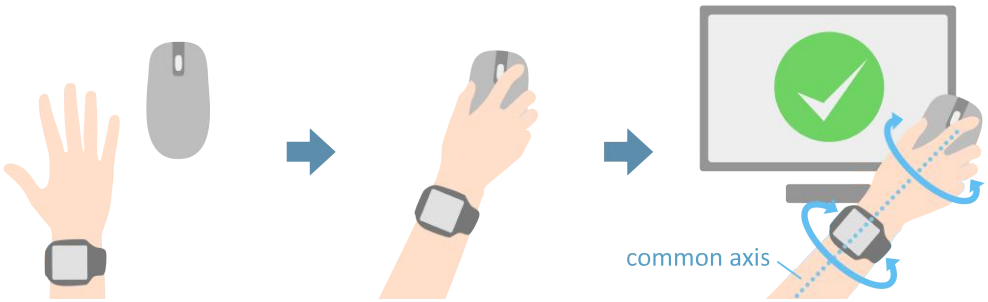

Fig. 1. Illustrating how *TwistIn* works. In this example, one may log in a computer through a smart mouse. First, one may pick up the mouse (middle) and perform the TwistIn gesture in mid-air for one to two seconds (right). Then, our method can make use of the motion sensor installed in the smart mouse 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.

guessable [2, 44], or easily obtained through a smudge attack [4] or shoulder surfing [37]. 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. There could be serious consequences, if such information is disclosed to other parties. Moreover, 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, 24, 49, 50]. This approach 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.

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, 12, 45].

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. 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. Another solution is to detect physical proximity and unlock a device when an authentication token is close to the device [46]. 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 attentions 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.

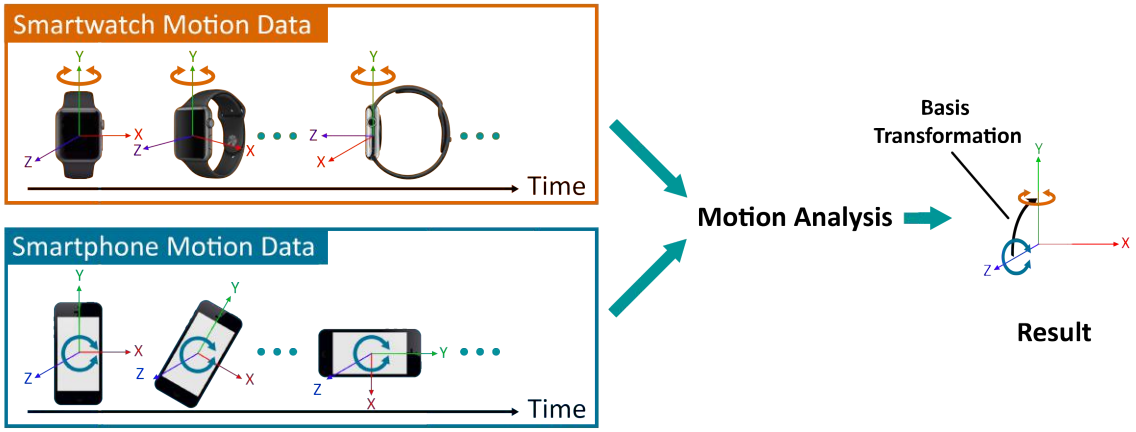

Fig. 2. To measure the correlation between two motion data acquired in different reference frames (coordinate systems), we first need to find the basis transformation to rotate and align the motions in a common frame.

Our approach. 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 [31], 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 about smartwatches, we aim to explore in this work *the application of smartwatches as the authentication tokens to enable us to gain quick access to some other smart devices.*

In this paper, we present *TwistIn*, a tangible authentication mechanism designed for quick access to smart devices by using a smartwatch; see Figure 1. Our method exploits an already-authenticated smartwatch and takes it as a token to authenticate and control other smart devices. To start, the user grabs a smart device and performs a TwistIn gesture while wearing a smartwatch, our method then co-analyzes the motion of the watch and the smart device, and detects if the two motions are in sync; see Figure 2. After a valid authentication, the user can immediately gain access to the smart device and can control it through the smartwatch's interface.

Our approach is natural and efficient, and allows fast smart device authentication and control in a multi-user multi-device environment. It is particularly useful when the user wants to work with multiple devices and switch between devices; by simply grabbing another device and performing a TwistIn gesture, the user can switch to a new device in just one to two seconds, and begin to use it through the interface on the smartwatch. Lastly, our method requires no hardware interface such as buttons and touchscreen on the smart devices, so it is suitable for devices such as smart glasses, drones and 3D-printed objects with electronic circuits. In the end of this paper, we will present experiments and results to demonstrate the applicability and efficiency of our method.

Challenges. Technically, the key problem is to detect whether the watch's motion is in sync with the smart device's motion when the user performs the TwistIn gesture; see Figure 2. This is a motion co-analysis problem that involves several challenges. First, the coordinate frames of the two devices are generally not aligned, since the coordinate frames are hardware- and software-dependence. 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. Second, when the user performs the TwistIn gesture, the watch's motion is mostly composed of the rotations

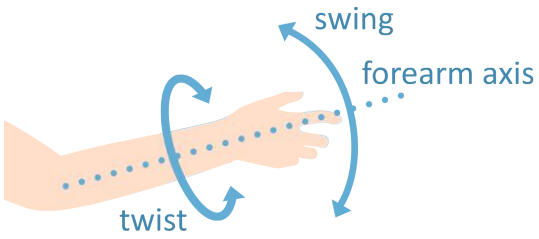

Fig. 3. Twist about forearm (primary) and swing about wrist (secondary) during the TwistIn gesture.

around the forearm axis, while the device's motion may compose of not only the rotations around the forearm (primary) but also the rotations around the wrist (secondary); see Figure 3. The presence of the secondary rotation could affect the detection. Third, we have to account for the gestural variation among different users, since different users may have slightly different speed and wrist rotation when performing the TwistIn gesture. Lastly, the detection computation has to be done in real-time to support interactive applications.

Our Contributions. To meet the above challenges, we first collect the orientation of the smartwatch and the smart device on user's hand over time, and calculate the transformation that rotates and aligns the orientation of one device to another, i.e., basis transformation. If such transformation is nearly a constant over time (while both devices are not stationary), we could deduce that the user is holding the two devices. Moreover, our method should compare and consider not only the timestamps and magnitude of the orientation data, but also the device orientations and the orientation changes, as well as to account for the challenges we described earlier.

In detail, we first formulate an optimization model for the problem, where the goal is to find an optimal orientation difference to best align the reference frames of the two devices during the TwistIn gesture. In the optimization, we also aim to take out the secondary wrist rotations performed on the smart device, since such rotation should be roughly perpendicular to the forearm rotation. Additionally, we consider the twisting frequency, which is related to the periodic rotations in the TwistIn gesture. By solving this optimization, we can effectively filter out the wrist rotations and noise, and maximize the correlation between the two motion data after aligning them with the basis transformation. Lastly, our method authenticates the smart device if a valid and stable transformation is observed between the device and the watch. We are not aware of any previous work that adopts our TwistIn gesture or a similar optimization technique to analyze motion data retrieved from multiple devices.

To summarize, we make the following contributions:

- First, we propose the TwistIn gesture, as a novel authentication mechanism for fast login to smart devices in a multi-user multi-device environment by taking a smartwatch as an authentication token.
- Second, we formulate a rotation decomposition technique to filter out the wrist rotation from the forearm rotation when handling noisy orientation data.
- Lastly, we develop a fast algorithm to detect synchronous rotatory motions between the smart watch and the smart device by solving for the optimal orientation difference between the two devices and by using the rotation decomposition technique.

Paper organization. We first review the related work on motion analysis in Section 2. We then present our method in Section 3. After that, we present the implementation details and experiments to evaluate our method in Section 4. We discuss the limitations and future works in Section 5. Finally, we conclude our work in Section 6.

2 RELATED WORK

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 [27, 35, 36]. Moreover, motion data

obtained from smartphones and smartwatches had been used for activity classification [18, 28, 32, 40, 42], and for virtual data manipulation [11, 26, 41]. 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 [23, 34, 38]. Hinckley et al. [20] 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. [47] 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. [29] 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. [43] identified multi-device gestures using a hidden Markov model, while Kim et al. [25] developed an authoring system for multi-device acceleration-based gestures, where devices can be associated via synchronous patterns. Hinckley [19] detected synchronous bumping gestures against one another by looking for a pair of devices with synchronized opposite accelerations. Lester et al. [30] 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. [48] 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 shares a similar spirit with ours by co-analyzing the motions of smartphone and smartwatch, the 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. In contrast, our method, besides having a different gesture, has several distinctive differences for correlating the motion: (i) instead of just the magnitude and frequency, we correlate the transformation between devices using orientation data fused from accelerometer data and gyroscope data; and (ii) we account for the motion difference between the hand and the forearm, and formulate an optimization and an algorithm to take away the wrist swing rotation when matching the motions; see again Figure 1.

3 TWISTIN: TANGIBLE AUTHENTICATION OF SMART DEVICES VIA MOTION CO-ANALYSIS WITH A SMARTWATCH

The input to our system consists of two series of 3D orientation data over time from the two devices. The orientations are calculated from the readings of the motion sensors in the devices, including an accelerometer and a gyroscope. Given these inputs, the TwistIn system checks if the two devices were moving in sync through an optimization, and decides if the motion corresponds to a TwistIn gesture.

3.1 Data Acquisition and Processing

The input 3D orientations from the devices are represented as quaternions; please refer to Appendix A for how quaternions represent rotations and orientations in 3D space. 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 [21].

However, the quaternion formulation represents the same orientation by $\vec{q}$ or $-\vec{q}$, which are equal in magnitude but opposite in directions. Hence, before we interpolate between two quaternions, say $\vec{q_A}$ and $\vec{q_B}$, we should flip

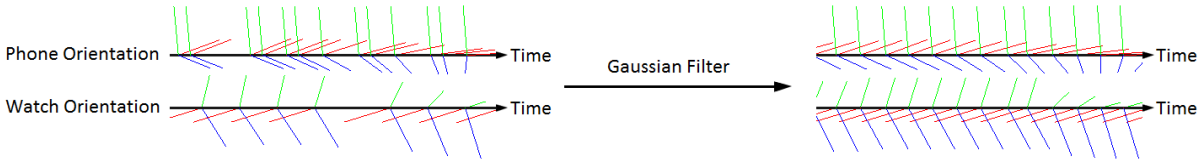

Fig. 4. Using a Gaussian filter to resample and smooth the orientation data.

the sign of $\vec{q_B}$, if the interpolation path (angular changes) between $\vec{q_A}$ and $-\vec{q_B}$ is shorter than that between $\vec{q_A}$ and $\vec{q_B}$. 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; see Appendix B for the details.

3.1.1 Resampling. Each orientation data is associated with a timestamp. However, 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 4 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 represented as quaternions q_A^t and q_B^t for devices A and B at time t.

3.1.2 Calculating Device Rotation. Each orientation data of a device is in fact a 3D rotation from a reference frame; see Figures 5a and 5b. 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. Hence, to facilitate the comparison between 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 5c):

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

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.

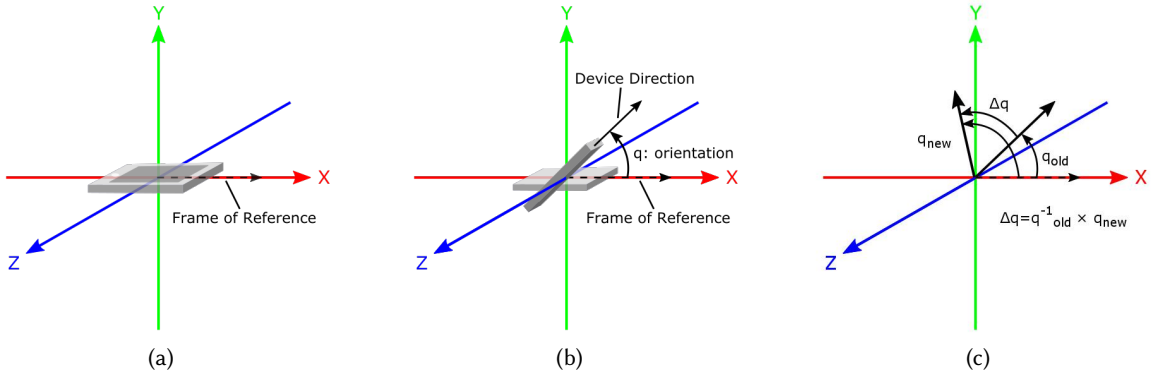

Fig. 5. (a) The device's reference frame. (b) The orientation of a device refers to the rotation from the reference frame to the device's current orientation. (c) Given q_{old} and q_{new}, we can compute the device's change in orientation Δq as shown above.

3.2 Optimization Model

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 2. Denoting r as the quaternion of the basis transformation, we should have

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

where T_{basis} is the period in which we estimate the basis transformation r.

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:

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

The above optimization model is, however, vulnerable for two reasons: (i) the wrist rotation, which would affect the device held on user's hand (see introduction and Figure 3); and (ii) the elbow movement, which would affect both devices. Hence, we filter out the unwanted secondary rotations by formulating a rotation decomposition [22] 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:

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

which can be geometrically interpreted as a projection of q to a, and then followed by a normalization.

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. (1) as:

$$\begin{aligned} \max_{\vec{a_A}, \vec{a_B}, r} \quad & \sum_{t\in T_{\text{basis}}} \langle p_A^t, r\;p_B^t\;r^{-1}\rangle \\ \text{subject to} \quad & ||\vec{a_A}|| = 1, ||\vec{a_B}|| = 1,\ \text{and}\ ||r|| = 1, \end{aligned} \tag{2}$$

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 $p_B^t = \frac{[w_B^t, (\vec{a_B}\cdot\vec{q}_B^t)\vec{a_B}]}{\sqrt{(w_B^t)^2 + (\vec{a_B}\cdot\vec{q}_B^t)^2}}$.

3.3 Deriving the Optimization Solution

The objective function to be maximized in Eq. (2) can be rewritten as

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

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:

$$\langle \Delta p_A^t\, r, r\, \Delta p_B^t \rangle = \frac{\begin{bmatrix} w_A^t & -a_x(\vec{a_A}\cdot\vec{q}_A^t) & -a_y(\vec{a_A}\cdot\vec{q}_A^t) & -a_z(\vec{a_A}\cdot\vec{q}_A^t) \\ a_x(\vec{a_A}\cdot\vec{q}_A^t) & w_A^t & a_z(\vec{a_A}\cdot\vec{q}_A^t) & -a_y(\vec{a_A}\cdot\vec{q}_A^t) \\ a_y(\vec{a_A}\cdot\vec{q}_A^t) & -a_z(\vec{a_A}\cdot\vec{q}_A^t) & w_A^t & a_x(\vec{a_A}\cdot\vec{q}_A^t) \\ a_z(\vec{a_A}\cdot\vec{q}_A^t) & a_y(\vec{a_A}\cdot\vec{q}_A^t) & -a_x(\vec{a_A}\cdot\vec{q}_A^t) & w_A^t \end{bmatrix} r}{\sqrt{(w_A^t)^2+(\vec{a_A}\cdot\vec{q}_A^t)^2}} \cdot \frac{\begin{bmatrix} w_B^t & -b_x(\vec{a_B}\cdot\vec{q}_B^t) & -b_y(\vec{a_B}\cdot\vec{q}_B^t) & -b_z(\vec{a_B}\cdot\vec{q}_B^t) \\ b_x(\vec{a_B}\cdot\vec{q}_B^t) & w_B^t & -b_z(\vec{a_B}\cdot\vec{q}_B^t) & b_y(\vec{a_B}\cdot\vec{q}_B^t) \\ b_y(\vec{a_B}\cdot\vec{q}_B^t) & b_z(\vec{a_B}\cdot\vec{q}_B^t) & w_B^t & -b_x(\vec{a_B}\cdot\vec{q}_B^t) \\ b_z(\vec{a_B}\cdot\vec{q}_B^t) & -b_y(\vec{a_B}\cdot\vec{q}_B^t) & b_x(\vec{a_B}\cdot\vec{q}_B^t) & w_B^t \end{bmatrix} r}{\sqrt{(w_B^t)^2+(\vec{a_B}\cdot\vec{q}_B^t)^2}}$$

$$= \frac{w_A^t I + (\vec{a_A}\cdot\vec{q}_A^t)M_A}{\sqrt{(w_A^t)^2+(\vec{a_A}\cdot\vec{q}_A^t)^2}} r \cdot \frac{w_B^t I + (\vec{a_B}\cdot\vec{q}_B^t)M_B}{\sqrt{(w_B^t)^2+(\vec{a_B}\cdot\vec{q}_B^t)^2}} r ,$$

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

$$\frac{w_A^t r \cdot w_B^t r + w_A^t r \cdot (\vec{a_B}\cdot\vec{q}_B^t)M_B r + w_B^t r \cdot (\vec{a_A}\cdot\vec{q}_A^t)M_A r + (\vec{a_A}\cdot\vec{q}_A^t)(\vec{a_B}\cdot\vec{q}_B^t)[(M_A r)\cdot(M_B r)]}{\sqrt{[(w_A^t)^2 + (\vec{a_A}\cdot\vec{q}_A^t)^2]\,[(w_B^t)^2 + (\vec{a_B}\cdot\vec{q}_B^t)^2)]}} .$$

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 $rM_A r$ and $rM_B r$, if we multiply them out, we will find that $rM_A r = rM_B r = 0$.

Therefore, we can simplify the optimization model in Eq. (2) as

$$\max_{\vec{a_A},\vec{a_B}} \quad \sum_{t\in T_{\text{basis}}} \frac{w_A^t w_B^t + (\vec{a_A}\cdot\vec{q}_A^t)(\vec{a_B}\cdot\vec{q}_B^t)}{\sqrt{[(w_A^t)^2 + (\vec{a_A}\cdot\vec{q}_A^t)^2]\,[(w_B^t)^2 + (\vec{a_B}\cdot\vec{q}_B^t)^2)]}} \tag{3}$$
$$\text{subject to} \quad ||\vec{a_A}|| = 1 \;\text{ and }\; ||\vec{a_B}|| = 1\,.$$

To solve the optimization problem, we apply Lagrangian multipliers and iteratively solve for $\vec{a_A}$ and $\vec{a_B}$ by fixing some variables; please refer to Appendix C for details.

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:

$$r_1 = \frac{[\vec{a_A}\cdot\vec{a_B} + 1, \vec{a_A}\times\vec{a_B}]}{|||[\vec{a_A}\cdot\vec{a_B} + 1, \vec{a_A}\times\vec{a_B}]|||} \quad\text{and}\quad r_2 = \frac{[\vec{a_A}\cdot -\vec{a_B} + 1, \vec{a_A}\times -\vec{a_B}]}{|||[\vec{a_A}\cdot -\vec{a_B} + 1, \vec{a_A}\times -\vec{a_B}]|||} .$$

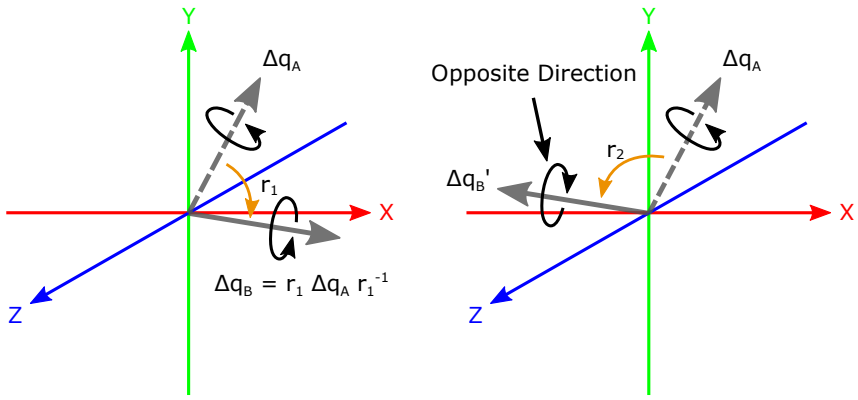

Fig. 6. Noting that Δq_A is device A's rotation and Δq_B is device B's rotation, and each is expressed as a rotation around an axis. Applying r_1 or r_2 to transform Δq_A can result in Δq_B or $\Delta q'_B$, where $\Delta q'_B$ is $[(\Delta q_B)_w, -(\Delta q_B)_x, -(\Delta q_B)_y, -(\Delta q_B)_z]$.

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 6.

3.4 Analyze the Turning Points in TwistIn

If the motion of the two devices are 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. Hence, we need to detect and see if the device motion is really a TwistIn gesture. The detection has two parts: (i) whether the two motions are in sync (see next subsection); and (ii) whether the motions consist of a number of twists (or turns) that correspond to the TwistIn gesture. Note again 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 3. We define T_{gesture} as the time period in which we detect the TwistIn gesture. This T_{gesture} is also used in the next subsection for checking if the device motions are in sync.

Concerning the second part in detecting the TwistIn gesture, for each device, we analyze the w-component of its resampled orientation data (quaternions) and locate the turning points over the gesture detection period T_{gesture}. 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 7(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_{gesture}, we regard the device as not performing the TwistIn gesture, e.g., see Figure 7(d). In practice, we empirically set N as seven. On the other hand, since there may be more than N turning points within T_{gesture}, 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_{gesture}. We denote $T_{\text{turning}} \subset T_{\text{gesture}}$ 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_{gesture}, 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.

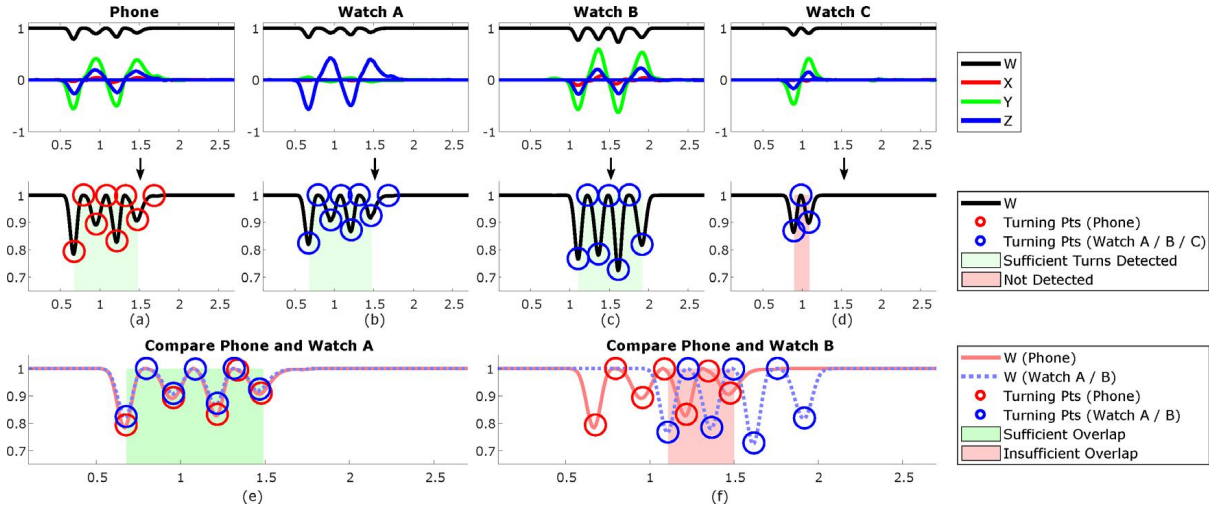

Fig. 7. We have two conditions to filter TwistIn gestures. First, there should be at least seven turning points within the gesture detection period (T_{gesture}); compare (a-c) with (d): watch C does not have sufficient turning points. Second, the turning point periods (T_{turning}) of the two devices 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.

3.5 Analyze r over Time

For a TwistIn gesture, apart from turnings, the motion of the two devices should also be in sync. To this end, we compute and analyze r over time by constructing a time series for r (denoted as r^t) over T_{gesture}, since $T_{\text{basis}} < T_{\text{gesture}}$. Note also that T_{gesture} can be divided into three sub-periods: (i) before the twisting, (ii) during the twisting, and (iii) after the twisting. Among the sub-periods, we focus on the middle sub-period, since the devices undergo major rotations during the twisting. 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:

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

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

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

In detail, if the motion of two devices are in sync, the variation of r over T_{gesture} should be small and the objective function (see Eq. (3)) 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_{gesture}. Afterwards, we check the results against their corresponding thresholds β_{err} and β_{var}:

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

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

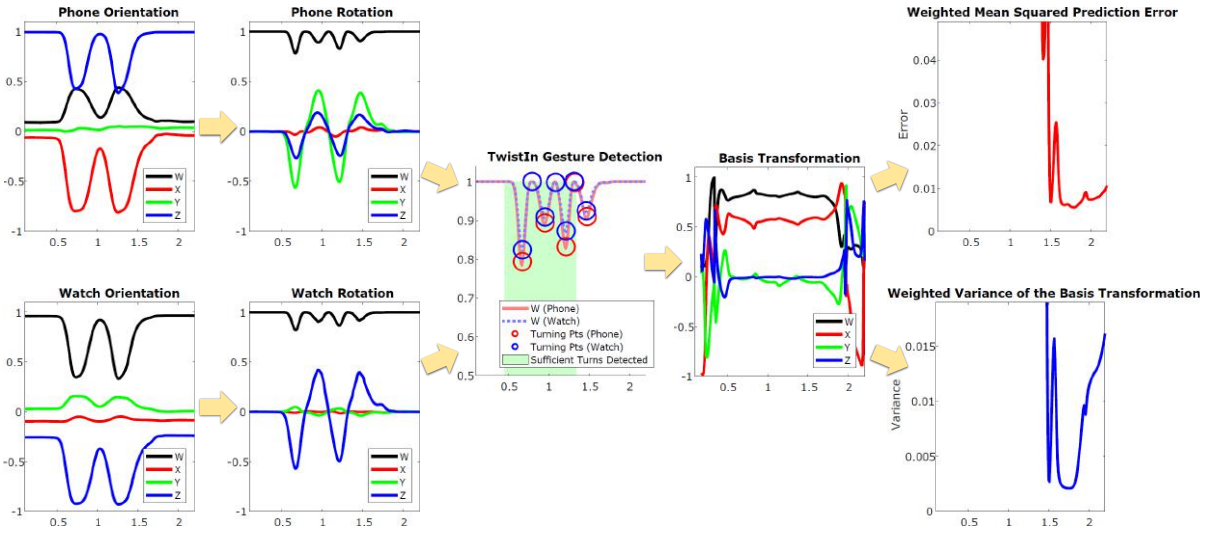

Fig. 8. A running example showing the time series processing in the TwistIn Authentication Procedure.

3.6 Overall TwistIn Authentication Procedure

In this subsection, we present the overall TwistIn authentication procedure, which is built on top of the various components we described earlier; see Algorithm 1 below. To summarize, we first collect a series of orientation data from each device and then resample and smooth these data using a Gaussian Filter. The smoothed time series are then used to calculate the devices' rotation. After that, we solve for a time series of basis transformation via our optimization model with rotation decomposition (see Eq. 3), and check if both devices have been twisted sufficiently by analyzing their turning points and periods. Finally, we analyzed the basis transformation time series 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 device will be authenticated. Figure 8 presents a running example showing that how the time series are calculated and processed in the whole process.

ALGORITHM 1: 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 our optimization model with rotation decomposition (r^t)
if *both devices have been sufficient twisted by analyzing their turning points and periods* **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

Fig. 9. Experiment setup using an iPhone 6S (iOS 10.3.3) and an Apple Watch 1 (watchOS 3.0).

4 IMPLEMENTATION DETAILS AND EXPERIMENT RESULT

In this section, we present the implementation details of our method and then a series of experiments that evaluate our method. In the first experiment, we determine how many twists for the TwistIn gesture. In the second experiment, we evaluate TwistIn with a user study with participants. In the third experiment, we additionally evaluate TwistIn by considering several challenging scenarios.

4.1 Hardware and Implementation Details

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 10.3.3) and an Apple Watch 1 (watchOS 3.0); see Figure 9. 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. Finally, we also archive the orientation time series from both devices as space-delimited files for further evaluation.

4.2 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. Typically, a twist is a back-and-forth rotation about the forearm (see the Figure 9), 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. Below, we recall the six parameters in our method:

N	––	minimum number of turning points in a TwistIn gesture; see Section 3.4;
T_{basis}	––	time period taken to optimize for the basis transformation; see Section 3.2 & 3.3;
T_{gesture}	––	time period taken to evaluate and detect the TwistIn gesture; see Section 3.4 & 3.5;
β_{overlap}	––	minimum overlap percentage between the devices' turning period (T_{turning}); see Section 3.4;
β_{err}	––	threshold for weighted mean squared prediction error; see Section 3.5, and;
β_{var}	––	threshold for weighted variance of basis transformation; see Section 3.5.

Table 1. EER achieved when the devices were twisted for different number of twists (#Twists).

#Twists	N	T_{basis}	T_{gesture}	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

Procedure. To compare the effectiveness of using 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 three more times but the participant would perform TwistIn with 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 × 2 devices × 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 $\binom{60}{2} = 1,740$. Note also 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 extremely low chance to produce the false positives observed in the results.

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 by using the optimal parameters to perform the authentication procedure on all the motion sample pairs. The optimal parameters were obtained by exhaustively searching the parameter space, such that the performance was maximized, i.e., minimizing the squared error $\sqrt{\text{FNR}^2 + \text{FPR}^2}$.

Table 2. Effect of prolonged twisting studied by using different N, T_{basis} and T_{gesture} on the data where the participants performed five twists.

#Twists	N	T_{basis}	T_{gesture}	β_{overlap}	β_{err}	β_{var}	TPR	FPR	FNR	TN	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

Discussion. We used the TwistIn algorithm to analyze the motion samples for each setting (i.e., one twist to five twists), and obtained the evaluation results shown in Table 1. From the results, 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 2. Since T_{gesture} depends on N and T_{basis} depends on T_{gesture}, we reused the T_{gesture} and T_{basis} from the corresponding row in Table 1; see the 3rd and 4th columns in Table 1 and Table 2. For parameters β_{overlap}, β_{err} and β_{var} listed in Table 2, we exhaustively searched for their values for optimal performance, for each case shown in each row in Table 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 2. In the results, we observe a high FPR (0.1092) for a short evaluation period and a small N when $T_{\text{gesture}} = 1.0s$ and $N = 3$. This shows that a small T_{gesture} with a small N is vulnerable to attacks when the devices were twisted for a short period of time. To balance between convenience and security, we decided to use two twists as the minimum requirement in the TwistIn gestures.

4.3 Experiment 2: Performance Evaluation and User Study

In the second experiment, we evaluated our method's performance and gathered feedback from the participants.

Procedure. We introduced our method and demonstrated the TwistIn gesture to the participants. We also reminded them to avoid wrist motions during 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 for 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 twice.
(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 twice.
(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.

Table 3. Parameters used in experiments two and three.

N	T_{basis}	T_{gesture}	β_{overlap}	β_{err}	β_{var}
7	0.10	1.20	0.60	0.061	0.0188

Table 4. Performance obtained for each scenario. Note that scenarios 1, 2, 3 and 5 consisted of positive cases while scenario 4 consisted of negative cases only.

Scenario		TPR	FPR	FNR	TNR
(i)	Sitting (w/ preferred hand)	0.9833	—	0.0167	—
(ii)	Sitting (w/ non-preferred hand)	0.9750	—	0.0250	—
(iii)	Standing (w/ preferred hand)	0.9833	—	0.0167	—
(vi)	Attacking (w/ preferred hand)	—	0.0583	—	0.9417
(v)	Sitting (w/ preferred hand)	0.9917	—	0.0083	—
Overall		0.9833	0.0583	0.0167	0.9417

(v) Scenario (i) was repeated again.

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

(i) The participant pressed the "Record" button on the smartphone.
(ii) After three seconds, the smartphone vibrated, signaling the start of the recording.
(iii) The participant performed the TwistIn gesture by twisting the devices twice according to the requirements provided for each scenario.
(iv) The participant waited briefly for a second and then pressed the "Complete" button.

Right after the experiments, we asked the participants 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.

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. 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.

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 $\times$ 2 devices $\times$ 10 times $\times$ 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.

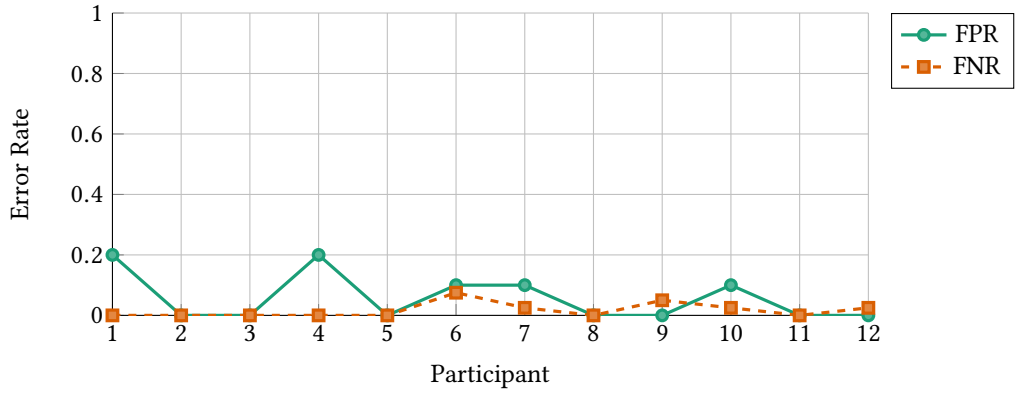

Fig. 10. Performance of each participant achieved. The FPR refers to the rate of gaining access to other's smartphone in scenario 4 and the FNR refers to the failing rate of authenticating the smartphone in scenarios 1, 2, 3, and 5 combined.

Table 5. Distribution of variables and the p-value calculated.

Variable	Mean	S-D	Minimum	Maximum	p-value	
					FPR	FNR
Hand Length (cm)	18.91	1.375	17	21	0.2293	0.6311
Hand Span (cm)	20.09	1.814	17	23	0.0646	0.4096
Hand Circumference (cm)	19.36	1.206	18	22	0.8090	0.3065
Wrist Size (cm)	15.54	1.157	14	17	0.7109	0.4068
Minimum Devices Distance (cm)	6.583	2.466	4	11	0.6568	0.6992

Performance Analysis. We evaluated the motion samples from each scenario using the empirical parameters shown in Table 3. We then calculated the evaluation metrics (see Section 4.2) and tabulated the results in Table 4. In summary, the result is satisfactory, since the mean TPR is high while the mean FPR and FNR are both very low.

Then, we compared the performance across scenarios. The differences of FNR between scenarios 1 and 2 suggested that authenticating using the preferred hand had slightly better performance than the non-preferred hand. Indeed, some participants mentioned and informed us that it was more difficult and less convenient to twist with the non-preferred hand. However, there was no obvious difference between the results for scenarios 1 and 3. Indeed, no participants reported any difference between sitting and standing. On the other hand, there was an 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 low FPR, which showed that it was difficult to mimic other's TwistIn gesture. Still, a few attack attempts were succeeded 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.

Correlation between Performance and User Preference. We analyzed the performance statistically to study if there were any correlation between performance and 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 FPR and FNR. The p-value for the FPR was 0.3604, meaning that the preference of the preferred hand did not affect the FPR. On the other hand, the p-value for FNR was 0.0331, which is lower than 0.05. 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 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 FPR and FNR, and Table 5 shows the results. Since the p-values 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.

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 FPR and FNR, and obtained the results tabulated in Table 5. Since all of the p-values are greater than 0.05, we can reject the null hypothesis. This suggested that the participants' performances had no correlation with their physical differences.

Feedback from participants. We interviewed the participants to rate our method and compare with others. The results are tabulated in Table 6, 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, 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

Table 6. Questions and responses of the interview on the scale of -5 (being strongly disagree) to +5 (being strongly agree).

Question	Mean	S-D	Minimum	Maximum
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

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.

4.4 Experiment 3: Challenging Scenarios

In the last experiment, we additionally tested the performance of our method for 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 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 × 2 devices × 10 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.

Table 7. Performance obtained by applying the parameters in Table 3 to detect TwistIn for the following scenarios.

	Scenario	TPR	FNR
1	walking	0.6800	0.3200
2	jogging	0.5000	0.5000
3	picking up from desk	0.8600	0.1400
4	taking out from pocket	0.7400	0.2600
5	on a bus trip	0.9600	0.0400

Table 8. TNR and FPR evaluated 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.

TNR	FPR
0.9852	0.0148

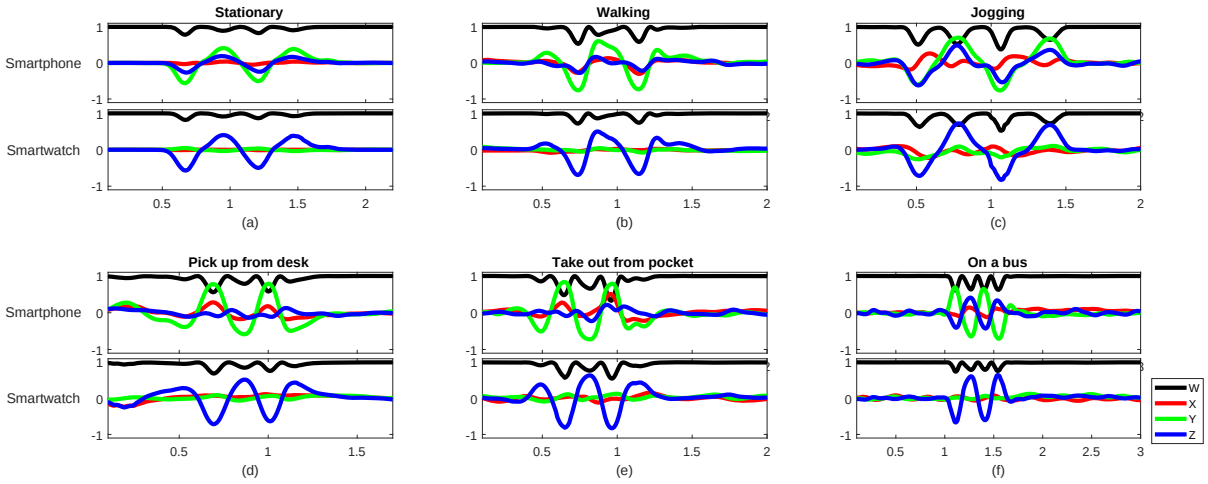

Fig. 11. Plotting the device rotations in various scenarios. (a) is obtained from scenario 1 in Experiment 2, while (b)-(f) are obtained from scenarios 1 to 5 in Experiment 3, respectively. Note that in all figures, x-axes denote the time and y-axes denote the value of the xyzw-components in the quaternion samples.

(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 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.

Discussion. Similar to the previous experiments, we calculated the TPR, FPR, FNR, and TNR to evaluate our method's performance as in Section 4.2. We used the parameters listed in Table 3 to evaluate each scenario. The

result is tabulated in Table 7. 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 8. 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. Obviously, 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 11(b) and (c), some additional rotations can be observed in motions for smartphone but not for 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. These match the results, where the participants performed the best in scenario 5 and performed poorly in scenario 2. While we have considered the rotation decomposition to our optimization model, it may not be able to filter out all kinds of additional rotations if the rotations align with the rotational axis of the TwistIn gesture (forearm). Therefore, these rotations would affect the calculation of the basis transformations. On the other hand, the results in Table 8 also shows that our method could minimize the chance of unauthorized access as the FPR is as low as 0.0148.

5 LIMITATION AND FUTURE WORK

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 solve this problem by considering motion samples over a longer time period and exploring the use of deep learning in decomposing the rotation and recognizing TwistIn.

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 include biometrics such as the twisting frequency and magnitude into the algorithm. This can potentially improve our method's performance.

Lastly, 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.

6 CONCLUSION

In this paper, we present the TwistIn gesture to authenticate devices using a smartwatch. We observe that two devices share the same rotation when they are interacted using a single hand, but 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 formulated an optimization based on a rotation decomposition to filter out unwanted rotation components and solve for the basis transformation between the two devices. 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 false positive and false negative rates are only 0.0583 and 0.0167 (see Table 4), suggesting that this method can be efficiently adopted in small devices for allowing fast access to the devices.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments that contributed to improving the paper. The work is supported in part by the CUHK strategic recruitment fund and Hong Kong Research Grants Council under General Research Fund - Project No. 14225616.

REFERENCES

[1] Ron Amadeo. 2015. Meet Google's "Eddystone" - a flexible, open source iBeacon fighter: The Ars Technica review. https://arstechnica.com/gadgets/2015/07/meet-googles-eddystone-a-flexible-open-source-ibeacon-fighter/ [Online; accessed 17-April-2018].

[2] Daniel Amitay. 2011. Most common iPhone passcodes. *Retrieved June* 15 (2011), 2011.

[3] Inc. Apple. 2017. About iBeacon on your iPhone, iPad, and iPod touch. https://support.apple.com/en-gb/HT202880 [Online; accessed 17-April-2018].

[4] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. 2010. Smudge Attacks on Smartphone Touch Screens. *Woot* 10 (2010), 1–7.

[5] Noam Ben-Asher, Niklas Kirschnick, Hanul Sieger, Joachim Meyer, Asaf Ben-Oved, and Sebastian Möller. 2011. 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*. ACM, 465–473.

[6] Manuele Bicego, Andrea Lagorio, Enrico Grosso, and Massimo Tistarelli. 2006. On the use of SIFT features for face authentication. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, 35–35.

[7] Jonathan Camhi. 2015. BI Intelligence projects 34 billion devices will be connected by 2020. http://www.businessinsider.com/bi-intelligence-34-billion-connected-devices-2020-2015-11 [Online; accessed 27-February-2017].

[8] Bing Chang, Ximing Liu, Yingjiu Li, Pingjian Wang, Wen-Tao Zhu, and Zhan Wang. 2017. Employing Smartwatch for Enhanced Password Authentication. In *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 691–703.

[9] Xiang Anthony Chen, Tovi Grossman, Daniel J Wigdor, and George Fitzmaurice. 2014. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 159–168.

[10] Andrew Cunningham and Lee Hutchinson. 2016. macOS 10.12 Sierra: The Ars Technica review. https://arstechnica.com/gadgets/2016/09/macos-10-12-sierra-the-ars-technica-review/4/#h4 [Online; accessed 16-April-2018].

[11] Yuan Du, Haoyi Ren, Gang Pan, and Shjian Li. 2011. Tilt & touch: Mobile phone for 3D interaction. In *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 485–486.

[12] Serge Egelman, Sakshi Jain, Rebecca S Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. 2014. Are you ready to lock?. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 750–761.

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

[14] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. 2013. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security* 8, 1 (2013), 136–148.
[15] Inc. Gartner. 2017. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. http://www.gartner.com/newsroom/id/3598917 [Online; accessed 10-May-2017].
[16] Avik Ghose, Chirabrata Bhaumik, and Tapas Chakravarty. 2013. 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*. ACM, 1135–1142.
[17] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors* 12, 9 (2012), 11734–11753.
[18] John J Guiry, Pepijn van de Ven, and John Nelson. 2014. Multi-sensor fusion for enhanced contextual awareness of everyday activities with ubiquitous devices. *Sensors* 14, 3 (2014), 5687–5701.
[19] Ken Hinckley. 2003. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*. ACM, 149–158.
[20] Ken Hinckley and Hyunyoung Song. 2011. Sensor synaesthesia: touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 801–810.
[21] Berthold KP Horn. 1987. Closed-form solution of absolute orientation using unit quaternions. *JOSA A* 4, 4 (1987), 629–642.
[22] Chao Hu, Max Q-H Meng, Mrinal Mandal, and Peter Xiaoping Liu. 2006. Robot rotation decomposition using quaternions. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*. IEEE, 1158–1163.
[23] William Hutama, Peng Song, Chi-Wing Fu, and Wooi Boon Goh. 2011. 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*. ACM, 3315–3318.
[24] Andrew H Johnston and Gary M Weiss. 2015. Smartwatch-based biometric gait recognition. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*. IEEE, 1–6.
[25] Ju-Whan Kim, Han-Jong Kim, and Tek-Jin Nam. 2016. 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*. ACM, 2307–2318.
[26] Florian Klompmaker, Karsten Nebe, and Julien Eschenlohr. 2013. Towards multimodal 3d tabletop interaction using sensor equipped mobile devices. *Mobile Computing, Applications, and Services* (2013), 100–114.
[27] Sven Kratz, Michael Rohs, and Georg Essl. 2013. 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*. ACM, 173–178.
[28] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.
[29] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic Sensors: Towards General-Purpose Sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 3986–3999.
[30] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. 2004. "Are You With Me?" ï£¡– Using Accelerometers to Determine if Two Devices are Carried by the Same Person. In *International Conference on Pervasive Computing*. Springer, 33–50.
[31] Kent Lyons. 2015. What can a dumb watch teach a smartwatch?: Informing the design of smartwatches. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ACM, 3–10.
[32] Takuya Maekawa, Daisuke Nakai, Kazuya Ohara, and Yasuo Namioka. 2016. 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*. ACM, 1088–1099.
[33] Simon Olberding, Sergio Soto Ortega, Klaus Hildebrandt, and Jürgen Steimle. 2015. 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*. ACM, 223–232.
[34] Mahsan Rofouei, Andrew Wilson, AJ Brush, and Stewart Tansley. 2012. 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*. ACM, 1915–1918.
[35] Jaime Ruiz and Yang Li. 2011. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2717–2720.
[36] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 197–206.
[37] Florian Schaub, Ruben Deyhle, and Michael Weber. 2012. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th international conference on mobile and ubiquitous multimedia*. ACM, 13.
[38] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 13–16.
[39] Corey Shemelya, Fernando Cedillos, Efrian Aguilera, David Espalin, Danny Muse, Ryan Wicker, and Eric MacDonald. 2015. Encapsulated copper wire and copper mesh capacitive sensing for 3-D printing applications. *IEEE Sensors Journal* 15, 2 (2015), 1280–1286.
[40] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. 2016. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors* 16, 4 (2016), 426.

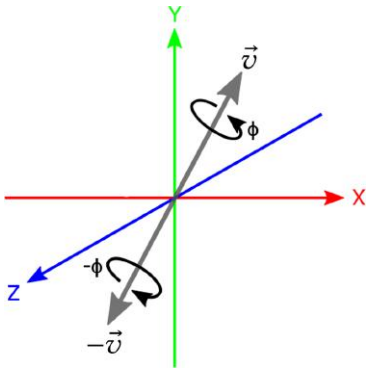

Fig. A1. A 3D rotation of ϕ degrees around an arbitrary axis $\vec{v}$ is the same as a 3D rotation of $-\phi$ degrees around $-\vec{v}$. Therefore, both quaternions q and $-q$ represents the same 3D rotation.

[41] Peng Song, Wooi Boon Goh, Chi-Wing Fu, Qiang Meng, and Pheng-Ann Heng. 2011. WYSIWYF: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1333–1342.

[42] Xing Su, Hanghang Tong, and Ping Ji. 2014. Activity recognition with smartphone sensors. *Tsinghua Science and Technology* 19, 3 (2014), 235–249.

[43] Vu H Tran, Kenny TW Choo, Youngki Lee, Richard C Davis, and Archan Misra. 2016. MAGI: Enabling multi-device gestural applications. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*. IEEE, 1–6.

[44] Sebastian Uellenbeck, Markus Dürmuth, Christopher Wolf, and Thorsten Holz. 2013. Quantifying the security of graphical passwords: the case of android unlock patterns. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 161–172.

[45] Dirk Van Bruggen, Shu Liu, Mitch Kajzer, Aaron Striegel, Charles R Crowell, and John D'Arcy. 2013. Modifying smartphone user locking behavior. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, 10.

[46] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal De Lara. 2007. Amigo: Proximity-based authentication of mobile devices. In *International Conference on Ubiquitous Computing*. Springer, 253–270.

[47] Gerard Wilkinson, Ahmed Kharrufa, Jonathan Hook, Bradley Pursglove, Gavin Wood, Hendrik Haeuser, Nils Y Hammerla, Steve Hodges, and Patrick Olivier. 2016. 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*. ACM, 2832–2844.

[48] Katrin Wolf and Jonas Willaredt. 2015. PickRing: seamless interaction through pick-up detection. In *Proceedings of the 6th Augmented Human International Conference*. ACM, 13–20.

[49] Junshuang Yang, Yanyan Li, and Mengjun Xie. 2015. MotionAuth: Motion-based authentication for wrist worn smart devices. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 550–555.

[50] Hongzi Zhu, Jingmei Hu, Shan Chang, and Li Lu. 2017. ShakeIn: Secure User Authentication of Smartphones with Habitual Single-handed Shakes. *IEEE Transactions on Mobile Computing* 16, 10 (2017), 2901–2912.

A REPRESENTING ROTATIONS USING QUATERNIONS

Quaternions represent 3D rotations using an axis-angle representation. 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:

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

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

B MINIMIZING ANGULAR CHANGES IN SUCCESSIVE QUATERNIONS IN A TIME SERIES

The same 3D rotation can be represented by quaternions $+q$ 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 A1. Since successive 3D orientations in a time series should be highly related, the angular changes between them should be small (less than π). Hence, we calculate the angle between two successive quaternions, say q_i and q_{i+1}, in a time series as

$$\theta \;=\; \cos^{-1}(\; 2\langle q_i, q_{i+1}\rangle^2 - 1\;)\,,$$

where $\langle q_i, q_{i+1}\rangle$ stands for the dot product of q_i and q_{i+1}. In other words,

$$\langle q_i, q_{i+1}\rangle = \pm\sqrt{\frac{\cos\theta + 1}{2}}\,.$$

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_{i+1}) accordingly; see Algorithm 2 below.

ALGORITHM 2: 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 $q_i \cdot q_{i+1} < 0$ **then**
    $q_{i+1} = -q_{i+1}$
  end
end

C SOLVING THE OPTIMIZATION

To solve the optimization model shown in Eq. (3), 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:

$$\vec{a}_A^{\,0} = \frac{1}{n}\sum_{t\in T_{\text{basis}}}\frac{\vec{q}_A^{\,t}}{||\vec{q}_A^{\,t}||} \quad\text{and}\quad \vec{a}_B^{\,0} = \frac{1}{n}\sum_{t\in T_{\text{basis}}}\frac{\vec{q}_B^{\,t}}{||\vec{q}_B^{\,t}||}\,,$$

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

$$\vec{a}_A^{\,k+1} = \begin{bmatrix} \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})\; x_A^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \\ \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})\; y_A^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \\ \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})\; z_A^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \end{bmatrix} \text{ and } \vec{a}_B^{\,k+1} = \begin{bmatrix} \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})\; x_B^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \\ \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})\; y_B^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \\ \sum_{t\in T_{\text{basis}}} \frac{(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})\; z_B^t}{\sqrt{[(w_A^t)^2+(\vec{a}_A^{\,k}\cdot\vec{q}_A^{\,t})^2]\;[(w_B^t)^2+(\vec{a}_B^{\,k}\cdot\vec{q}_B^{\,t})^2)]}} \end{bmatrix},$$

followed by normalizations

$$\vec{a}_A^{\,k+1} = \frac{\vec{a}_A^{\,k+1}}{||\vec{a}_A^{\,k+1}||} \quad\text{and}\quad \vec{a}_B^{\,k+1} = \frac{\vec{a}_B^{\,k+1}}{||\vec{a}_B^{\,k+1}||}\,.$$

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.

Received August 2017; revised February 2018; accepted June 2018