3-2018

Smartwatch-based early gesture detection & trajectory tracking for interactive gesture-driven applications

Tran Huy VU
Singapore Management University, hvtran.2014@phdis.smu.edu.sg

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Quentin ROY
Singapore Management University, quentinroy@smu.edu.sg

Kenny Tsu Wei CHOO
Singapore Management University, kenny.choo.2012@phdis.smu.edu.sg

Youngki LEE
Singapore Management University, YOUNGKILEE@smu.edu.sg

DOI: https://doi.org/10.1145/3191771

Smartwatch-based Early Gesture Detection & Trajectory Tracking for Interactive Gesture-Driven Applications

TRAN HUY VU, Singapore Management University, Singapore
ARCHAN MISRA, Singapore Management University, Singapore
QUENTIN ROY, Singapore Management University, Singapore
KENNY CHOO TSU WEI, Singapore Management University, Singapore
YOUNGKI LEE, Singapore Management University, Singapore

The paper explores the possibility of using wrist-worn devices (specifically, a smartwatch) to accurately track the hand movement and gestures for a new class of immersive, interactive gesture-driven applications. These interactive applications need two special features: (a) the ability to identify gestures from a continuous stream of sensor data early–i.e., even before the gesture is complete, and (b) the ability to precisely track the hand's trajectory, even though the underlying inertial sensor data is noisy. We develop a new approach that tackles these requirements by first building a HMM-based gesture recognition framework that does not need an explicit segmentation step, and then using a per-gesture trajectory tracking solution that tracks the hand movement only during these predefined gestures. Using an elaborate setup that allows us to realistically study the table-tennis related hand movements of users, we show that our approach works: (a) it can achieve 95% stroke recognition accuracy. Within 50% of gesture, it can achieve a recall value of 92% for 10 novice users and 93% for 15 experienced users from a continuous sensor stream; (b) it can track hand movement during such strokeplay with a median accuracy of 6.2 cm.

CCS Concepts: • **Human-centered computing → Gestural input**; **Ubiquitous and mobile computing systems and tools**;

Additional Key Words and Phrases: gesture recognition, wearable devices, hand tracking, low-latency, VR, immersive applications

ACM Reference Format:
Tran Huy Vu, Archan Misra, Quentin Roy, Kenny Choo Tsu Wei, and Youngki Lee. 2018. Smartwatch-based Early Gesture Detection & Trajectory Tracking for Interactive Gesture-Driven Applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 39 (March 2018), 27 pages. https://doi.org/10.1145/3191771

1 INTRODUCTION

Inertial sensing on wrist-worn devices, such as smartwatches, has recently been used to infer a variety of gesture-driven lifestyle activities, such as smoking [12] and eating [2, 17, 21]. These approaches typically focus on the problem of gesture recognition, i.e., using features defined over the inertial sensor data to identify specific gestures. Separately, researchers have investigated the use of such possibly-noisy inertial sensor data to track

Authors' addresses: Tran Huy Vu, Singapore Management University, 81 Victoria street, 188065, Singapore, hvtran.2014@smu.edu.sg; Archan Misra, Singapore Management University, 81 Victoria street, 188065, Singapore, archanm@smu.edu.sg; Quentin Roy, Singapore Management University, 81 Victoria street, 188065, Singapore, quentinroy@smu.edu.sg; Kenny Choo Tsu Wei, Singapore Management University, 81 Victoria street, 188065, Singapore, kenny.choo.2012@smu.edu.sg; Youngki Lee, Singapore Management University, 81 Victoria street, 188065, Singapore, youngkilee@smu.edu.sg.

2474-9567/2018/3-ART39 $15.00
https://doi.org/10.1145/3191771

the hand's 3-D location or movement trajectory, for applications such as pointing based interaction [14] and handwriting recognition [1].

In this paper, we investigate the possibility of using such wrist-based sensing (e.g., via a smartwatch) to enable a novel class of pervasive, real-time, gesture-driven interactive applications, such as immersive virtual reality (VR) games. For these applications, the objective is to accurately track the movement of the human hand, but with latencies low enough to preserve the interactivity of the application or the game. As an exemplar of such a gesture-centric game, consider a VR-based Virtual Table Tennis (**VTT**) application, where a user is competing or training against an opponent. While the user sees the table tennis board, the ball and the opponent in the VR display, she uses the real-world physical movement of her hand (wearing a smartwatch) to hit the ball. The hand's motions are tracked and integrated into the virtual world, and appropriately projected in the VR display. Accurate tracking of the hand movement is needed to faithfully replicate real-world mechanics—for example, the time instant when the racket hits the ball in VR should closely reflect the time that the user's hand would have struck the ball in the real world.

At present, such fine-grained hand tracking requires the installation and use of non-portable, custom infrastructure (such as the Kinect depth camera, the Wii Sensor Bar or the HTC-Vive laser tracker). Our goal is to untether the user from such infrastructural components. Instead, the user should be able to simply strap on a VR display and a wrist-mounted device such as a smartwatch, and perform this immersive application at any place.
Key Challenges: The key characteristic of our target class of immersive applications is that they require us to both (a) classify individual gestures and (b) to concurrently track the hand's movement trajectory, in real time. To provide the user with specific training on the type of stroke played, the system needs to correctly identify the stroke; moreover, to provide accurate visualization of *when* and *where* the user hits the ball, the system needs to track the hand's trajectory as well. A closer analysis of the representative interactive application reveals a couple of unique challenges:

- *Calculate the Trajectory Fast and Accurately:* Table Tennis is a very fast-paced game, with professional grade players often exchanging $\approx$ 120 strokes each per minute (see [24]) during a rally. To provide a truly interactive feel, the game must not exhibit *lag*—i.e., the game must detect the instant of contact between a player's racket and the ball instantly, so that the virtual reality game can proceed apace. This means that we must not only compute the hand's trajectory fast (in real time—e.g., with lag $\leqslant$ 102 msec - corresponding to the 80th percentile of lag tolerance reported in Figure 2), but also precisely (as the calculated point of intersection between the racket and the ball will affect the calculated time instant of contact as well).
- *Recognize the Gesture Before It is Completed:* Most stroke-based games (e.g., tennis, table tennis and badminton) involve a significant "follow-through"—i.e., the actual game gesture involves significant movement of the hand both before and after the act of striking of the ball. In fact, as our analysis later shows, a typical TT stroke gesture (the nucleus of gesture) lasts for about 296 ms, with the point of contact between the ball and racket occurring around 120 msec after the start of the stroke (i.e., at approx. the $39.7\%^{th}$ point of the gesture). Accordingly, we cannot afford to delay the execution of the recognition step till the end of the gesture, as this would seriously impact the interactive feel of the game.

The above requirements thus require an entirely new class of gesture recognition techniques, that can recognize gestures fast (even before the gesture is complete) and simultaneously track the hand's trajectory accurately, in real-time. In this paper, we develop an integrated smartwatch-based gesture recognition framework that tackles these two objectives: (a) gesture recognition and (b) hand tracking concurrently, and with low latency.
Key Contributions: We make the following key contributions:

- *Low-Latency, Preemptive Gesture Recognition:* We develop a gesture recognition technique that is able to not just identify gestures with low latency, but identifies a gesture even when it has not been entirely completed. Unlike conventional stream-oriented gesture recognition algorithms, the proposed technique

dispenses with a separate segmentation step (which can only be performed on the sensor stream after the gesture has been completed). Instead, we develop a unified Hidden Markov Model (HMM) based classifier, where an explicit *universal state* is used to reset the classification process whenever the evolving hand movement is judged as not likely to be part of the set of recognized gestures. We show that this proposed universal-state based HMM model for gesture recognition is accurate: person-independent models achieve a precision of 90.5% and a recall of 95.3% in recognizing 6 different TT strokes.

- *Gesture-aware Trajectory Tracking:* We next devise an inertial tracking algorithm that is able to track the hand trajectory accurately, in real-time, based purely on inertial sensors in commodity wearable devices. Unlike prior work, we do not track the entire hand trajectory at all times, but only during specific gestures. Our proposed approach utilizes a *gesture-specific regression model* to infer the hand's trajectory directly from the smartwatch's rotation vector (derived from the accelerometer and gyroscope sensor data). Compared to prior work that requires knowledge of the forearm length and an exhaustive training set for arbitrary hand movement, our approach is more light-weight and is able to track the movement accurately (with a median error of $\approx$ 6.2 cm), while the gesture is being performed.
- *Empirical Evidence on the Suitability of Table-Tennis Tracking:* We utilize a carefully-instrumented real-world setting, involving the VTT application and 25 distinct users (10 inexperienced & 15 experienced), to demonstrate both the technical accuracy for, and end-user benefit from, our proposed technology. We also show that this segmentation-oblivious algorithm is capable of early gesture detection, identifying 93.7% of gestures (for experienced users) within the first 50% of an individual gesture instance. We also demonstrate the sufficiency of the accuracy achieved for trajectory tracking: we are able to determine the stroke gesture with a median delay of 2.3 msec (after the ground truth time instant when the racket hit the ball). In addition, we show that users perceive our tracking to have high fidelity, with our system capturing the real-world *hit time* of strokes without perceptible delay more than 85% of the time.

2 REPRESENTATIVE APPLICATION & REQUIREMENTS

The requirements for an enhanced gesture recognition cum trajectory tracking solution come from our vision for a new class of multi-device immersive applications. In these applications, the user relies on multiple mobile & wearable devices, whose input and output interfaces are combined to offer an integrated, multi-modal experience. In particular, we consider the class of VR or AR (augmented reality) applications, where the user experiences a virtual/augmented world on her smartglasses, with other wearable devices providing fine-grained tracking of the user's gestural activities.

In the representative VTT application (illustrated in Figure 1), the wrist-worn smartwatch (or smartwatches) are used to monitor the fine-grained movement of the arm. The inertial sensors on the smartwatch can then be combined with the inertial sensing data from the head-mounted smartglasses, to obtain the movement trajectory of the arm *relative* to the user's body orientation, and this movement trajectory can then be embedded inside the virtual world displayed on the smartglasses (e.g., a point-of-view representation of the user's avatar). The VTT application can then be used to provide a realistic emulation of two (or four) players playing TT without being in physical proximity, with the players sharing a common view of the court and having to play actual strokes in the real world. Furthermore, to provide more realistic feedback of the experience of a real TT game, we can envision that the wrist-worn device provides some form of tactile feedback (e.g., by vibrating the wearable device) whenever the system detects that the user's racket has hit the ball. To provide a high-quality user experience, the tracking of each user's strokeplay should be low-latency (to ensure that the virtual world rendering and the tactile feedback do not perceptibly lag the physical world gestural activities) and accurate (to ensure that the outcome of the strokes reflect the physical world with high fidelity).

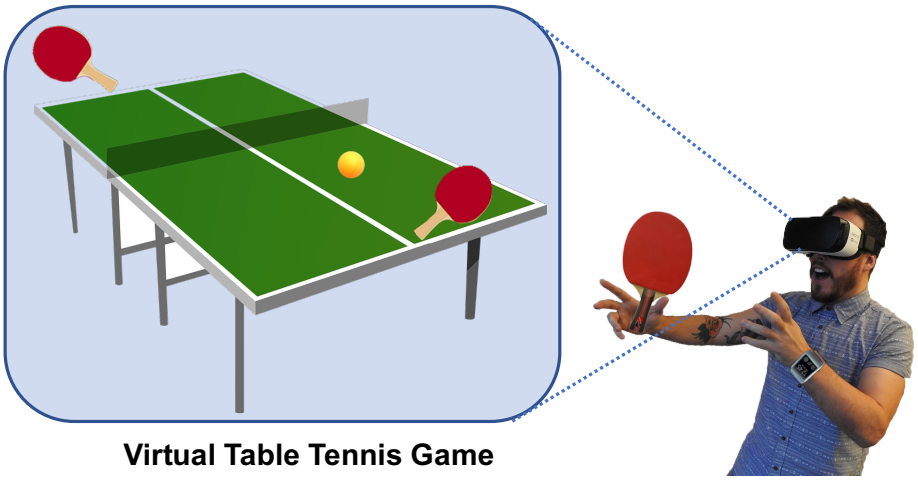


Fig. 1. The Virtual Table Tennis (VTT) application. A user makes real-world TT gestures while wearing a smartwatch, with the gestures being integrated into the virtual world displayed on the wearable VR device.

While the investigations in this paper are confined to this representative VTT application, the general requirement for low-latency gesture recognition and trajectory tracking apply to a much broader class of applications. For example, consider other VR-based applications such as Dancing Tutor (where a user's dance-related moves are tracked and projected into the virtual world) or Racing Emulator (where a driver is assumed to steer a racing car using a virtual steering wheel, with the wearable sensors providing real-time vibratory feedback about the road surface conditions). All of these applications exhibit the previously described characteristics of (i) a critical interaction occurring not at the end, but at an intermediate point, of a gesture, and (ii) the need to accurately track a user's arm movement, in real time, but only for a finite set of application-relevant gestures.

2.1 Perceiving Latency and Its Effects on Usability

Before proceeding further, it is useful to understand and quantify the desired performance characteristics of the representative VTT application. In particular, we are interested in examining the latencies generated from *gesture recognition* and *hand trajectory tracking*, and understand the point at which the latencies becomes noticeable to the user, thus impacting the user experience.

Jota et al. [7] examined the effects of latency on direct-touch pointing tasks. They found that no participant could notice latency below 20ms, with most participants (85%) not being able to differentiate between 1-40ms of latency. We conducted experiments with 12 participants to examine the noticeability of latency within the context of VTT. Participants were asked to hit a suspended table tennis ball with a table tennis bat 30 times. On detecting the hit event (this was measured via visual tracking with a high-speed camera (100 fps), attached to a powerful desktop that achieves near-zero frame processing latency), an audio alert was played after a randomly generated delay ranging from 0 to 500ms. We take the time difference between the audio alert and the hit event to be the *lag/latency,* and asked participants to indicate if they noticed the delay.

Figure 2 plots the cumulative distribution of the probability of user perception (i..e, the fraction of instances where the user indicated a perceptible delay) as a function of this lag/latency, for the experiment described above.Our results are similar to Jota et al., with no user being able to perceive latencies below 19ms, and only 5% of the noticed latencies being between 19-40ms (see Figure 2). We observe that 80% of users are completely

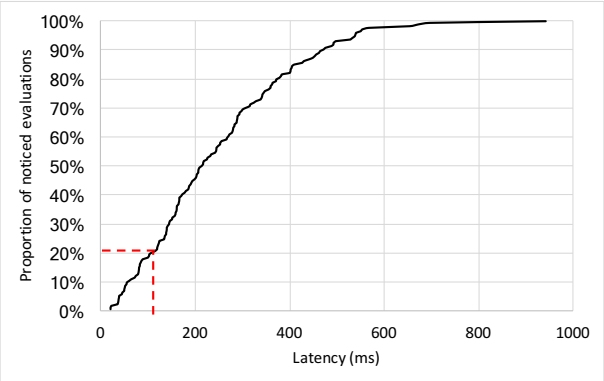


Fig. 2. Cumulative distribution of noticed latencies.

oblivious to lags less than or equal to 100 msec, and thus set ourselves a benchmark of developing a tracking system that can identify the specific stroke within 100 msec of the actual hit event.

3 SYSTEM OVERVIEW

To realize our vision of a practical gestural interface which can both recognize gestures early and track user's wrist accurately, we propose a system architecture (Figure 3) where a *Continuous Progressive Gesture Recogniser* cooperates with a *Gesture Based Trajectory Tracker* to enable early detection of gestures and infer the position of user's wrist during a gesture.

This system uses only sensors on a smartwatch without any environment/infrastructure installment. The *Data Collector* collects Accelerometer, Gyroscope, Magnetometer and additional derivative sensors (which are computed from the three physical sensors)–i.e., the Rotation Vector, Game Rotation Vector, Linear Acceleration and Gravity. The Rotation Vector, Linear Acceleration, and Gravity values are conveyed to the recognizer. The Rotation Vector is conveyed to the Trajectory Tracker.

- *Continuous Progressive Gesture Recogniser* (using a combination of HMM+ Classifier): Firstly, a Feature Extraction block extracts the Arm motion features including the velocity in vertical and horizontal direction and the angular orientation of the lower arm. These features are then combined as a feature vector and are quantized into discrete values by the Vector Quantization block. These discrete values are fed into a modified HMM model (StreamHMM) to compute the probability of gestures states. Finally, a classifier (Random Forest) is applied to enable the early detection of gestures.
- *Gesture-Based Trajectory Tracker:* The orientation (values of X-Axis of smartwatch) is extracted from the sensors by Data Collector, and is used as one feature of the Trajectory Tracker. It also uses the probabilities of the HMM states (of the Recogniser) to improve the tracking accuracy. A regression model automatically estimates the position of the wrist.

4 DATASET

We conduct two experiments to collect sensor and ground truth positional data, while participants perform the six basic table tennis strokes (see Figure 4). The sensor data was recorded using a smartwatch on a participant's wrist, and contain 3 hardware sensors: *Accelerometer*, *Magnetometer*, *Gyroscope*, and 4 derived sensors: *Linear acceleration*, *Gravity*, *Rotation vector*, and *Game rotation vector*. The ground truth positional data contain color video and the position corresponding to each pixel in the video. We use a ZED camera [19] mounted on the

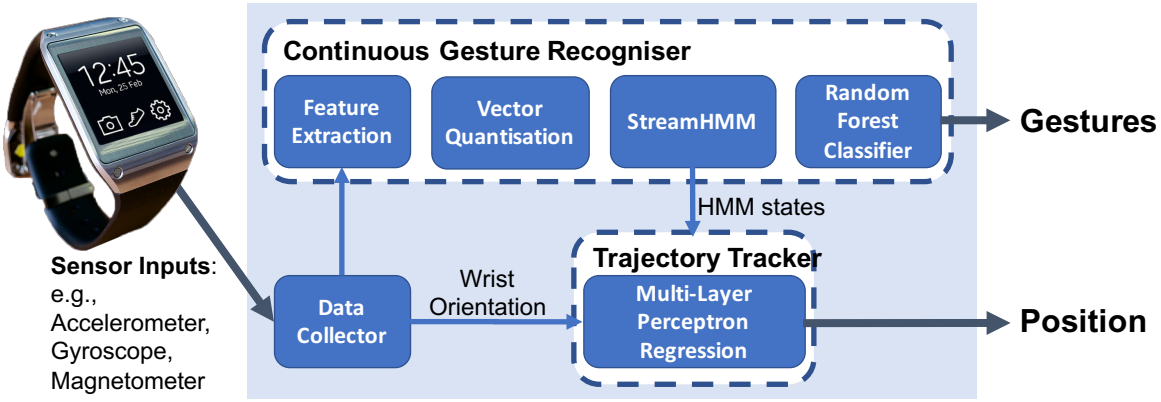


Fig. 3. System Overview.

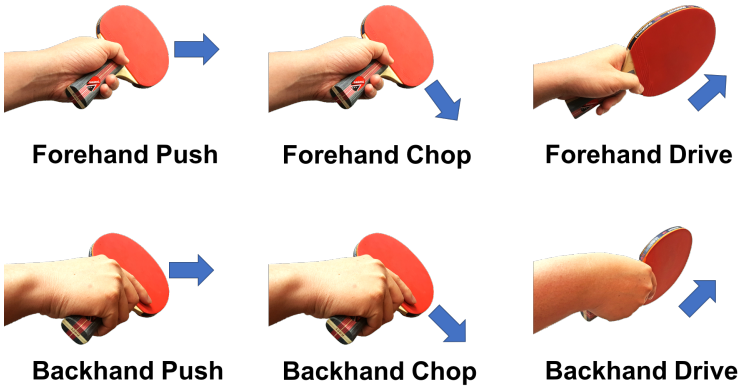


Fig. 4. The six table tennis strokes used in the study. The arrows show the direction of motion as applied to a table tennis ball coming from the right.

ceiling of the experiment room to record the positional ground truth data. To later extract the position of the wrist and head, we provided a yellow band and a pink band and asked participants to wear the yellow band on their head and the pink band on their wrist (wrapped around the smartwatch). *Note that the head/wrist bands, as well as the camera, are merely used to collect ground truth data for validation, and do not form part of our proposed wearable-based recognition technology.* We use a RoboPong 1050[1] table tennis robot to automatically serve balls to our participants. The robot was placed at the other end of the table (see Fig. 6).

In the first experiment, we recruited 10 participants, without any special focus on the experience or skill level of the participants. All participants were male university students aged between 21-30 years old and had a working understanding of how to play table tennis. Nine do not play on a regular basis, and one plays on average once a month. We used a *Samsung Gear Live* smartwatch to record sensor data at a sampling rate of 100Hz. At the beginning of the experiment, participants watched 6 training videos of the 6 gestures on a table tennis training channel [15]. Participants were given a short period of time to get familiar with the robot and gestures until they

[1]https://www.newgy.com/p-279-robo-pong-1050-plus.aspx

feel comfortable to play. For the study task, participants were asked to execute the strokes with balls served by the robot to achieve **24** successful returns per stroke. A stroke is deemed successful if the participant is able to return the ball to either the left or right halves of the opposite side based on our prompting from a balanced, randomized schedule (e.g., Left, Right, Left, Left, Right, Right). If a participant fails to return the ball to the correct half, he is asked to repeat that stroke until he is successful. The focus on successful stroke completion is intended to encourage participants to perform the gestures conscientiously, and to provide a sufficient number of usable gesture instances. As such, while a stroke may be deemed unsuccessful, the attempt can still be considered as a data point for that stroke. We examined the captured data, post-session, to remove invalid stroke instances (i.e., warm-up strokes). The table tennis robot was configured to serve balls at consistent spin, and to serve balls to two positions for each stroke to increase the variation of gestures.

Results from the first experiment achieved a high average gesture recognition accuracy, but also exhibited higher variance. Given that most participants had insufficient playing experience, it was unclear if the gestures performed were representative, and if the results obtained were affected by the likely variability in stroke-making among the participants. Accordingly, we conducted a second experiment, where we collected sensor data from 17 *experienced* players, recruited from an online table tennis group, where we explicitly indicated the need for a minimum of 3 years of playing experience. The group had only one lefthanded player and one penhold-style player, whom we've excluded from our studies due to the low sample size. Figure 5a plots the age and experience of the remaining 15 participants, who were 23–56 years old, with an average age of 32.8. The least experienced players have played table tennis for 3 years, while the most experienced players have 25 years of table tennis experience. Except for one player who plays table tennis daily, the others play table tennis weekly. Prior to playing, users were also asked to perform a self-evaluation of their proficiency in the 6 gestures (illustrated in Figure 5b), using a Likert scale ranging from 1-5: (1) corresponds to *I don't know how to play that stroke*, while (5) corresponds to *I am expert at it*. For these set of experiments, we used a more-modern *LG Urbane W150* smartwatch to record sensor data with the same sampling rate as in the first dataset. Participants were not asked to watch the video again because they had seen it during the online registration form. However, they were allowed to practice some shots to become familiar with the robot. Each participant in this study was asked to finish **30** successful instances of each of the 6 strokes.

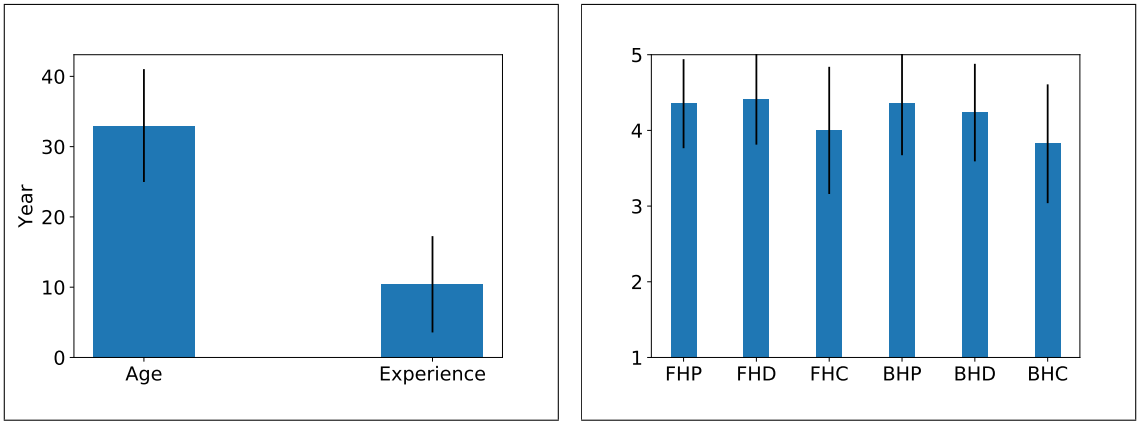
5 EARLY GESTURE RECOGNITION

In this section, we tackle the problem of recognizing gestures *accurately* and *early*–i.e., before the entire gesture has been completed. We shall see that the conventional approach for continuous gesture recognition, involving the cascaded steps of segmentation and classification, is insufficient, as it introduces too much latency in the recognition process. Accordingly, we propose and evaluate an alternative approach with two novel characteristics: (a) unified HMM model, with a novel *universal* state that allows a gesture to be recognized in streaming fashion, and (b) an explicit additional classifier, which uses features defined over the evolving HMM states, to recognize gestures early and robustly.

5.1 Inadequacy of Explicit Segmentation

Conventional gesture recognition techniques usually combine two stages: *segmentation* and *classification*. *Segmentation* first partitions the incoming sensor data into consecutive segments; each individual segment is then classified to assign it a particular gesture label. Gesture classification is typically performed using techniques such as Dynamic Time Warping (DTW) [10] or HMMs [13, 16].The overall accuracy of gesture recognition thus depends on both the segmentation and classification steps.

The basic segmentation technique is based on thresholds, with the gesture deemed to have started when some sensor-derived value exceeds a threshold and deemed to have ended when the value falls below another



(a) Participants' age and experience.

(b) Self evaluation of proficiency.

Fig. 5. (a): Average age and experience of participants. The least experienced participant has 3 years of experience. (b): Proficiency of participants in the 6 gestures based on self evaluation. (1) means "I cannot perform the stroke", (5) means "I am expert in this stroke".

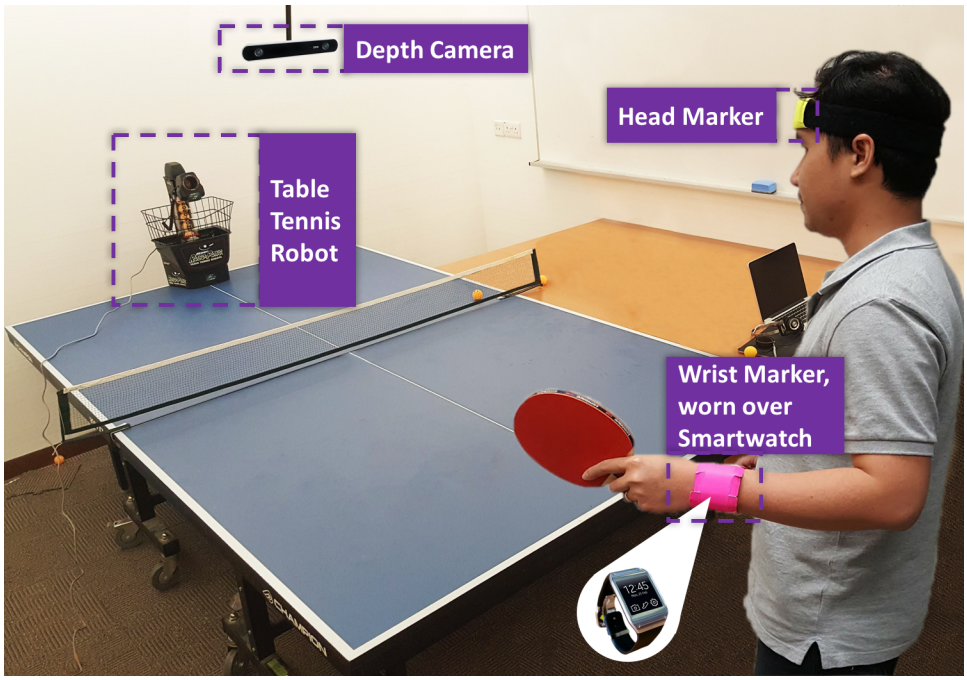


Fig. 6. Experiment Setup. The participant plays with a table tennis robot, while a high speed depth camera captures ground truth position data. A smartwatch worn on the hand provides sensor data on strokes performed.

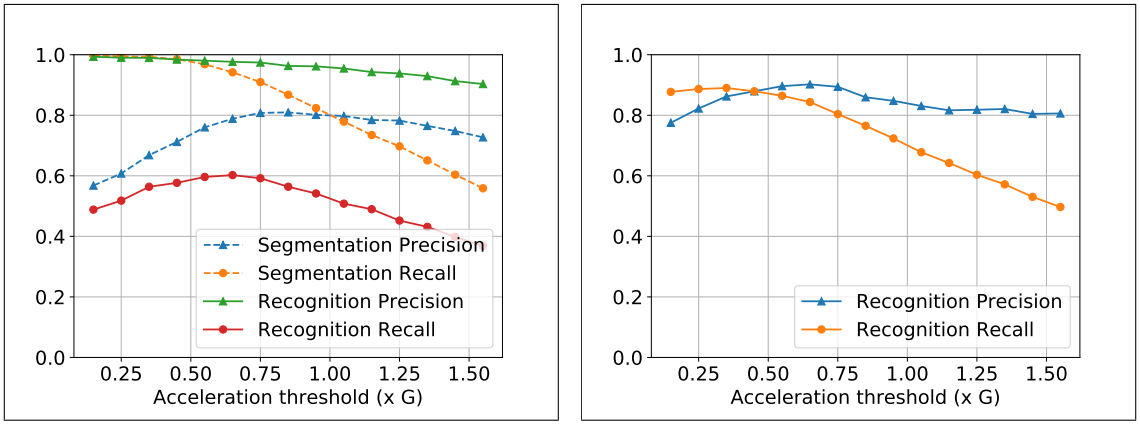
threshold. Typical examples include setting the threshold on the hand acceleration magnitude (derived from accelerometer data) and/or rotation values (derived from gyroscope readings). More sophisticated techniques can set these thresholds adaptively: for example, the authors in [13] propose a method of adaptively adjusting the acceleration threshold based on the observed false-positive (FP) and false-negative (FN) rates of the accelerometer segmentation compared with gyroscope segmentation.

This classification pipelined model of "segmentation followed by classification" suffers from three drawbacks:

(1) *Longer Latency:* By definition, a segment can be identified from a sensor stream only after the last sample for the segment has been collected—i.e. after the associated gesture has been completed [8]. Moreover, segmentation algorithms (e.g., [13]) often impose additional latency by requiring an additional observation period (beyond the end of the current gesture) to guard against the problem of 'gesture splitting', where a more complex gesture is inadvertently split into several segments. This approach is thus problematic for interactive gesture-based applications, where users expect the system to essentially react "instantaneously" (in real-time). More specifically, for our VTT application, we shall see, that a player typically hits the ball halfway (39.4%) into the gesture.

(2) *Poor Segmentation Accuracy Under Large Differences in Gesture Dynamics:* Threshold-based approaches typically work well when the different gestures have similar dynamics, such as similar overall duration or initial hand-force intensity. However, finding an appropriate threshold value is challenging, when the dynamics of gestures vary considerably (as is the case for table tennis, where certain strokes involve rapid and extended hand movement, while other strokes (such as Backhand Push) involve significantly lesser movements of the hand). To illustrate this point, we evaluated the accuracy of a single threshold-based segmentation technique (similar to the mechanism outlined in [13]) on table tennis gestures. Figure 7a shows the accuracy of our table tennis gesture data set as a function of the acceleration threshold.
The figure plots both the segmentation recall/precision (i.e., determining how many segments were identified compared to the ground truth), as well as the classifier recall/precision (what fraction of the segment duration was truly a part of the associated gesture?). Here precision is defined as: $\frac{\sum TruePositive}{\sum TruePositive + \sum FalsePositive}$, while recall is defined as: $\frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative}$. We notice that with low threshold, the segmentation step rarely misses gestures, but the segments contain redundant data which will lower the classification recall. With high threshold value, the segmentation step misses gesture more frequently, and thus also results in low classification recall. In general, using the segmentation method in [13], the system gets the highest recall of about 0.6 at a threshold of 0.65G. The precision is quite high and does not change considerably.

(3) *Inaccurate Segmentation, Leading to Poor Classification:* Even if the segmentation process can identify the occurrence of a gesture, it can misalign the start or end samples of the gesture–i.e., it can either truncate part of the entire gesture (in this case, the identified segment of samples is smaller than the true gesture duration) or it can falsely append spurious samples to a gesture (now, the segment duration is longer than the true gesture duration). This misalignment can reduce the accuracy of the downstream classification step. This can be observed by noting the classification recall and accuracy values presented in Figure 7a. We see that the classifier precision decreases slightly but still remains higher than 90% as the threshold is increased; however, the recall is fairly low (mostly below 60%) indicating that segmentation typically misses (truncates) a significant portion of individual gestures. A careful inspection of our dataset showed that a fixed-threshold segmenter tended to append spurious samples to gestures that involved more vigorous hand movement (such as Forehand Drive and Backhand Drive). In contrast, the segment was often truncated for less vigorous gestures (such as Forehand Push and Backhand Push).



(a) e-Gesture Performance

(b) Enhanced e-Gesture Performance

Fig. 7. (a): Applying E-Gesture technique [13] into our Table Tennis dataset. Threshold values range from 0.15G to 1.55G. The recall reach highest value of 0.6 at a threshold of 0.65G; it means only 60% true gestures are recognized. The precision is quite high, more than 0.9 accordingly, and slightly decreases when the threshold increases. *(*b): E-Gesture based enhanced classifier that uses the highest confidence value that an HMM's state sequence achieves at *any intermediate* point of the segment. Results in significantly higher recall, at the expense of reduction in precision.

5.1.1 Improved Baseline for Segmentation-based Classifier: Our initial results showed that the gesture classification accuracy was low, often because the segmentation step included spurious non-gesture accelerometer readings before and after the true gestural segment. More specifically, the HMM models in [13] output the final gesture only after the sensor data of the entire segment has been processed. Due to the spurious trailing non-gestural sensor data, the confidence level of the state sequence of each HMM classifier would fall below the confidence threshold.

To remedy this issue, we developed an alternative personalized model, where the confidence levels of the hidden states were tracked throughout the evolution of the identified gestural segment. More specifically, a per-HMM counter (each HMM corresponding to one of the 6 distinct gestures) was used to keep track of the *maximum confidence* value associated with the state sequence of each HMM, at *any* point during the segmented gesture. In the other words, this model computes the confidence value (probability) of each HMM on the current sub-segment until the end of the segment is detected. The segment was subsequently classified to be the gesture that matched the longest sub-segment with the highest confidence value.

Figure 7b plots the resulting precision and recall for this improved classifier. We observe a better balance of precision and recall. The recall value improves because this approach can effectively ignore the spurious motion artefacts that follow the true gesture. However, the precision drops (the false-positive rate increases) as this approach also results in several non-gesture segments being classified as a legitimate gesture, especially if parts of the segment had motion that bore a resemblance to one of the gestures. Based on empirical analysis, we found that a threshold of 0.45G served as an effective choice: yielding a recall value of 87.9% and a precision value of 87.9%.

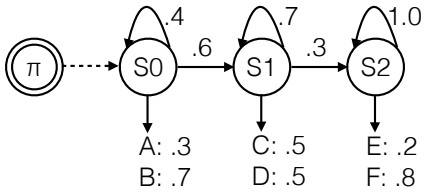


Fig. 8. Left_to_right HMM is useful to infer the states of a finite segment of data. {A-F} indicate the various output (i.e., observable) values–i.e., a vector of sensor values/attributes that are observed in each of the hidden states. The associated number denotes the emission probability.The initial state π serves as a starting point of the forward and Viterbi algorithm.

5.2 Segmentation-less Gesture Detection on Stream Data

We now describe our alternative approach for gesture detection that dispenses with the afore-mentioned segmentation step, and instead continually tries to identify gestures that occur within an incoming stream of sensor data.

Our approach is based on appropriate modifications to the basic Hidden Markov Model (HMM) based recognizers. HMMs have been widely used for gesture recognition (e.g., [13, 16]). The Left-To-Right HMM (See Figure 8) has been popularly used, as its structure matches well with the sequential evolution of gestures. In the traditional technique (which required a preceding segmentation step), each gesture is modeled as an HMM. Whenever a segment is identified (e.g., consisting of the inertial sensor samples $\{X_0X_1..X_t\}$, the probability of the segment with respect to each HMM is computed using Viterbi algorithm (1) as follows [2]:

$$P_s^0 = \pi_s * B[s][X^0]; P_s^t = \max_i P_i^{t-1} * A[i][s] * B[i][X^t] \quad (1)$$

This classical HMM approach requires an explicit starting state (corresponding to the start of the gesture) and operates on a finite length segment. We thus need to modify the HMM model to account for an infinite stream, with no explicitly delineated gesture segments.

In this work, we modify an HMM-based approach, as an example of a probabilistic state machine. Note that there are more sophisticated recent techniques, such as *Long Short Term Memory* (LSTM) neural networks, that have been proposed for processing sequential sensor data. In this work, our primary focus is on developing an overall framework for low-latency gesture detection and 3-D tracking; hence, we select the relatively low-complexity HMM as our candidate recognizer, as it can be executed on a smartwatch with very low processing overhead.

Our approach is based on the integration of a new *Universal State* into an existing HMM model (See Figure 9). The goal of the *Universal State* is to allow a "regular-expression like" matching of a gesture signature (an approach previously investigated in [20]), within a continuous sequence of observations with arbitrarily long non-gesture periods. In effect, the *Universal State* serves as a wildcard character (*), capturing the arbitrarily long sequence of non-gestural observables. The *Universal state* generates observations with the same probabilities of observations in the entire dataset–in the simplest case, this can be a uniform distribution, while it can also reflect the observed probability of a gesture occurring over the entire observation period. The *Universal state* also has a self-transition loop, implying that the HMM can stay at *Universal state* for an indefinite time; this corresponds to the arbitrary interval between two consecutive gestures.To reduce the chance that the observations are assigned to the final state of a gesture model for a long period, a return connection is added from the final state of each gesture model to the *Universal State*. Finally, whenever the system receives an observation (a vector of features),

[2] A[s][ns]: Probability of a transition from state s to state ns. B[s][O]: Probability of state s generating observation O

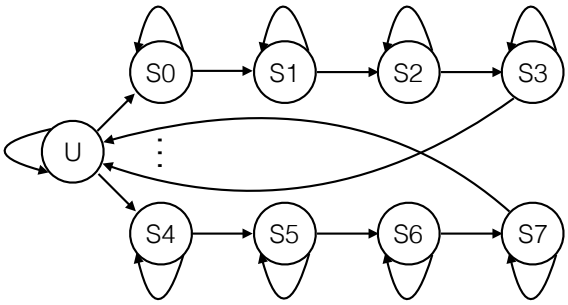


Fig. 9. Left_to_right HMM is useful to infer the states of a finite segment of data. {A-F} indicate the various output (i.e., observable) values–i.e., a vector of sensor values/attributes that are observed in each of the hidden states. The associated number denotes the emission probability.The initial state π serves as a starting point of the forward and Viterbi algorithm.

Table 1. Features used to define the output vector (O) for HMMs

Feature	Description
Horizontal velocity	Arm's angular velocity in horizontal plane
Vertical velocity	Arm's angular velocity in vertical plane
Wrist vertical angle	Wrist's rotation angle compared with vertical line

it updates the probabilities of states using equation 2 and 3. Initially, the probability of *Universal State* is 1.0; S_i^t is state i at time t, O^t is observation at time t.

$$P(S_i^{t+1}|O^{t+1}) = \frac{P(O^{t+1}|S_i^{t+1}) \times P(S_i^{t+1})}{P(O^{t+1})} \tag{2}$$

$$P(S_i^{t+1}|O^{t+1}) = \frac{B[i][O^{t+1}] \times \sum_1^n P(S_j^t) \times A[j][i]}{P(O^{t+1})} \tag{3}$$

5.2.1 Features Used in VTT Application. The unified HMM logic has so far been abstract in abstract terms. To build a working implementation of our VTT application, we need to define the sensor-driven set of *features* (i.e., the observation vector O in Equation 3) that we use to compute state transition probabilities. Prior work on HMMs for gesture recognition has defined O using relatively simple features over the raw data, such as discretization, quantization[10, 16], or taking differentials or integrals of accelerometer and gyroscope data[13]. In our work, we utilize a small set of more-complex *relative orientation* features, computed from the underlying accelerometer and gyroscope data as described in Table 1. These features are then quantized to 256 discrete levels, using a K-Means module with 256 centroids.

5.3 Early Detection of Gestures

We now describe the cascaded HMM-cum-classification technique that forms the core novel component of our early gesture detection algorithm.The Universal-state augmented unified HMM model allows us to detect the occurrence of candidate gestures in a streaming fashion, without requiring a separate and explicit segmentation step. However, in its base implementation, the unified HMM model (Figure 10) still identifies a gesture only when it is complete–i.e., only when the corresponding Left-to-Right model has its end-state exceed a specified

Table 2. Features used for early gesture classification (for the 6 VTT gestures)

Type	Feature	Description
Model	Normalized Expected State	$\frac{\sum_{i \in S} p_i * s_i)}{\sum_{i \in S} p_i)}$ (a total of 6 features, one for each HMM)
Model	Sum prob.	$\sum_{i \in S} p_i)$-a total of 6 features, one for each HMM
Model	Universal	The probability of the *Universal state*
Data	Acceleration	Instant magnitude of acceleration: $\sqrt{a_x^2 + a_y^2 + a_z^2}$
Data	Gravity	The value of X component of gravity
Time	Elapse time	For each HMM, the time elapsed since state 0 had the highest probability –a total of 6 features

likelihood threshold. Additional modifications to the gesture detection logic are needed to support our second objective: *of detecting a gesture early*, even before the entire gesture has been completed.

The simplest way of inference using this model is to define a threshold of probability. The system will select the model whose probability is the highest among gesture models. If this probability is higher than a threshold, the system outputs the corresponding gesture, otherwise, it will output *null-gesture*. However, the specification of a single explicit probability threshold value does not work across different gestures that have very different kinetic vectors.

Accordingly, we do not use an explicit threshold to perform early gesture detection. Instead, we develop a novel technique, where the *probability values of the hidden states of the HMM are used as features* by a supervised classifier to generate the output gesture. More specifically, for *each* individual Left-to-Right HMM (corresponding to a single gesture), after every incoming observation sample $O(t)$, we compute the following features, : (i) sum of the probabilities of each of its hidden states–i.e., $\sum_{i \in S} p_i)$; (ii) normalized expected value of the hidden state–i.e., $\frac{\sum_{i \in S} p_i * s_i}{\sum_{i \in S} p_i}$. Given N gestures, this leads to a $2N$-element vector of state-related features. In addition, we also consider the following additional features: (iii) the probability of the *Universal* state; (iv) the magnitude of the acceleration data (i.e., $\sqrt{a_x^2 + a_y^2 + a_z^2}$, where a_i represents the acceleration value along the i^{th} axis); (v) the X-component of the gravity vector (as this is an indirect indicator of the hand's orientation); and (vi) a vector $TS = [ts_1, \ldots, ts_N]$, where the i^{th} element expresses the time elapsed since the initial state (s_0) in the i^{th} HMM had the highest probability among all its hidden states. The acceleration feature helps distinguish deliberate strokes (which typically have higher acceleration) from other 'random' gestures, while the vector TS effectively guards against strokes being recognized too early (especially for the 'push' and 'drive' gestures).

Table 2 summarizes these features. This set of features is then used by a Random Forest classifier (trained in a supervised fashion) to determine the most likely gesture, for each incoming observation sample $O(t)$. Initially, the probability of the *Universal State* is the highest, and the classifier indicates that the inertial data (observed till this point) is most likely to be a 'non-gesture'. The classifier automatically outputs a gesture label, once sufficient data has arrived.

6 EXPERIMENTAL RESULTS ON EARLY GESTURE DETECTION

We first evaluate the efficacy of our proposed stream-compatible, early gesture detection technique using a *personalized* model–i.e., where a separate HMM (per gesture) is trained for each individual. This is similar to many prior studies on gesture recognition [10, 13, 16] that have also utilized personalized HMM models. The evaluation is done using the 'leave one gesture out' (LOGO) method. Because our system includes 2 stages: the HMM and the classification, we apply 2 training passes. At the first pass, we train the HMM model on the $N - 1$ instances of

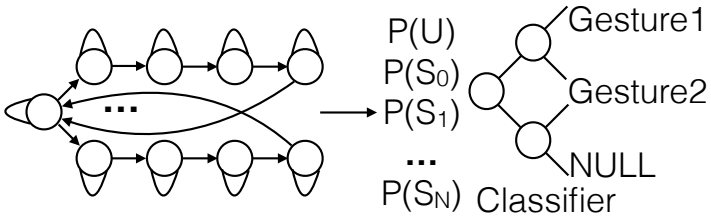


Fig. 10. A classifier is used to classify gestures, using probabilities of states as features

each gesture (N is the total number of each gesture). Then we feed the stream of the entire dataset into the HMM model, the features extracted from the training part (containing $N-1$ instances of gesture and non-gesture) are stored and used as training data for the downstream classifier (Random Forest). The features extracted from the testing part (contains the test gesture and non-gestures) are used to test the classifier. Of course, our classifier produces a label for every sensor sample. To make the classification more efficient, the *Universal State* is used to pre-filter the result. If the *Universal State* has the highest probability or the most probable gesture has not reached a state threshold (empirically chosen as 3), non-gesture is assigned; otherwise, the downstream classifier classifies the candidate gesture. Whenever the classifier declares a gesture A, our system considers the current and subsequent sensor sample as gesture A until the probability of the corresponding HMM state sequence drop below that of the *Universal State*.

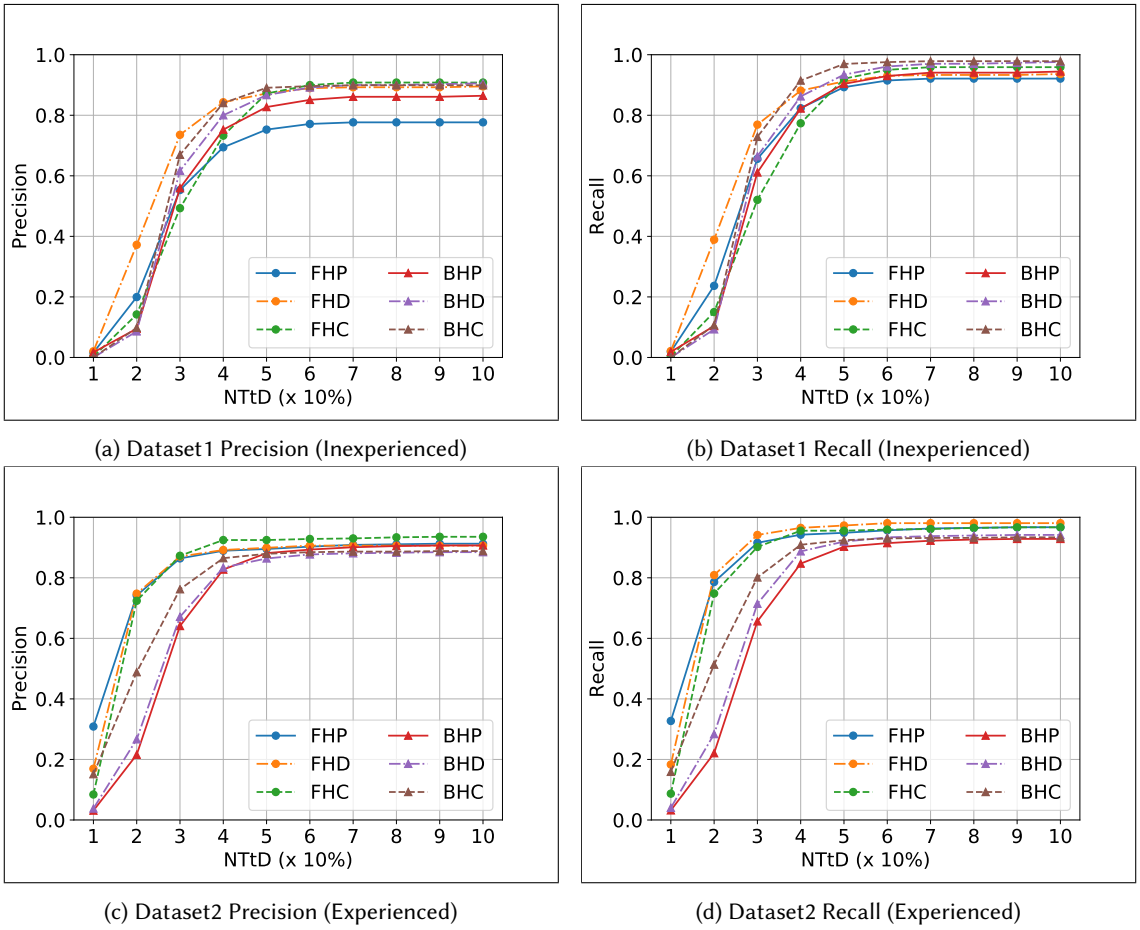
We sum up the total true positives, false positives, false negatives in each class across all participants; subsequently, we compute the average precision and recall values based on these numbers.

6.1 Accuracy vs. Fraction of Gesture Completed

Because our principal goal is to understand the early gesture detection capability, we compute the precision and recall of our model as a function of different *fraction of the total gesture duration.* More precisely, we compute the gesture detection accuracy for various values of *Normalized Time To Detect (NTtD)*[6] (in increments of 10%), which is the portion of gesture that has been completed. Figures 11 illustrate the overall precision and recall for each of the 6 strokes (gestures) as a function of the NTtD value, with the first 2 figures corresponding to study 1 (novice/inexperienced users) and the latter 2 figures corresponding to study 2 (experienced users). We see that:

- *Early Detection:* At NTtD values of 0.5 (i.e., at about 50% of the gesture), both precision and recall are higher than $\sim$ 84% for all the strokes. Moreover, after the NTtD value of 0.5, all the precision and recall values do not change considerably with an average precision of 87.4% and average recall of 95.2% (for inexperienced users).
- *Better Performance under Stable Gestures:* We see that, at identical NTtD values, the gesture recognition performance is superior for experienced users, compared to inexperienced users. In particular, at NTtD=30%, the precision & recall for users in study 2 is 64-87% and 65-94% respectively, while the comparative precision & recall is between 50-73% and 52-76%, respectively, for study 1. Clearly, experienced users exhibit more consistent stroke-making, implying that even a modest 30% samples of testing gesture are enough, whereas novice users tend to exhibit higher variability in performing the same stroke multiple times.

6.1.1 Variation Across Users: We observed that even person-specific models show significant differences in the cross-user variation in gesture recognition accuracy, for different gestures. Figures 12 plots the variation in the precision and recall, respectively, of gesture detection for each of the 6 VTT gestures, for both Study 1 and Study 2. (This is in contrast to Figures 11, which plot the *average* values, across all individuals.) We see that the median precision values (almost always above 87% for Study 1, and above 90% for study 2) and recall (above approx. 95%



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)

(c) Dataset2 Precision (Experienced)

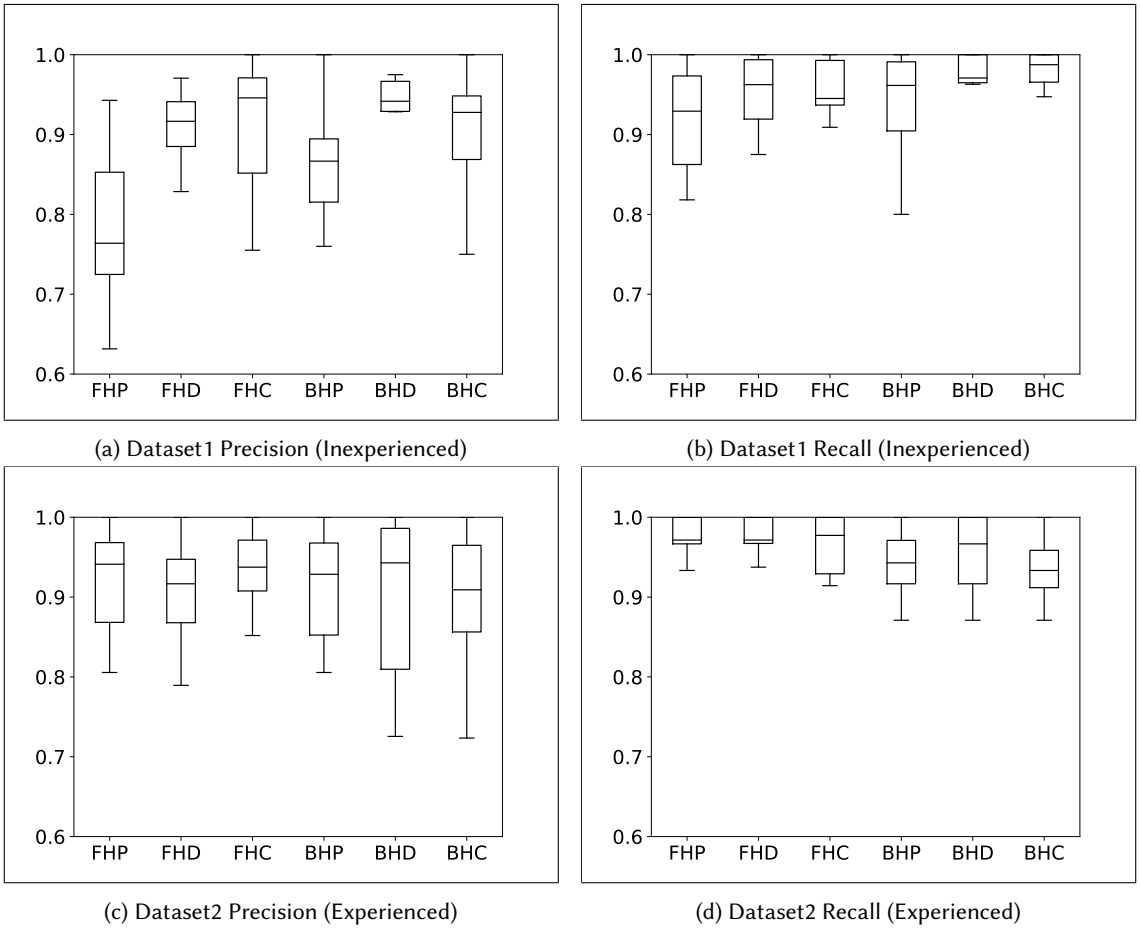
(d) Dataset2 Recall (Experienced)

Fig. 11. *Left*: Precision and *Right*: Recall w.r.t. Normalized Time To Detect (Across Both Studies)

for study 1 and 95% for study 2) values are fairly high. However, there was one individual user (in study 1) who exhibited significantly lower accuracy, due to the specific individual's inability to play the 6 strokes distinctly. In general, the variation across experienced users (Dataset 2) is significantly smaller than for inexperienced users (Dataset 1).

6.2 The Utility of the Classifier

For early gesture detection, our approach combines the streaming-oriented HMM with an HMM-state driven classifier. To understand the importance of the classifier, we studied the accuracy of gesture recognition both with and *without* the final HMM-based classifier. In general, HMM-based models can use a threshold (on the confidence value) to classify gestures [13, 16]. Similarly, we can classify gestures directly using our modified HMM model, where the *Universal State* serves as a dynamic threshold. Moreover, our model produces confidence value for every sample, as well as the most probable current state. Intuitively, we would choose the gesture that evolves to the higher state (a perfect gesture would, in fact, evolve to the final state). However, waiting for the



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)

(c) Dataset2 Precision (Experienced)

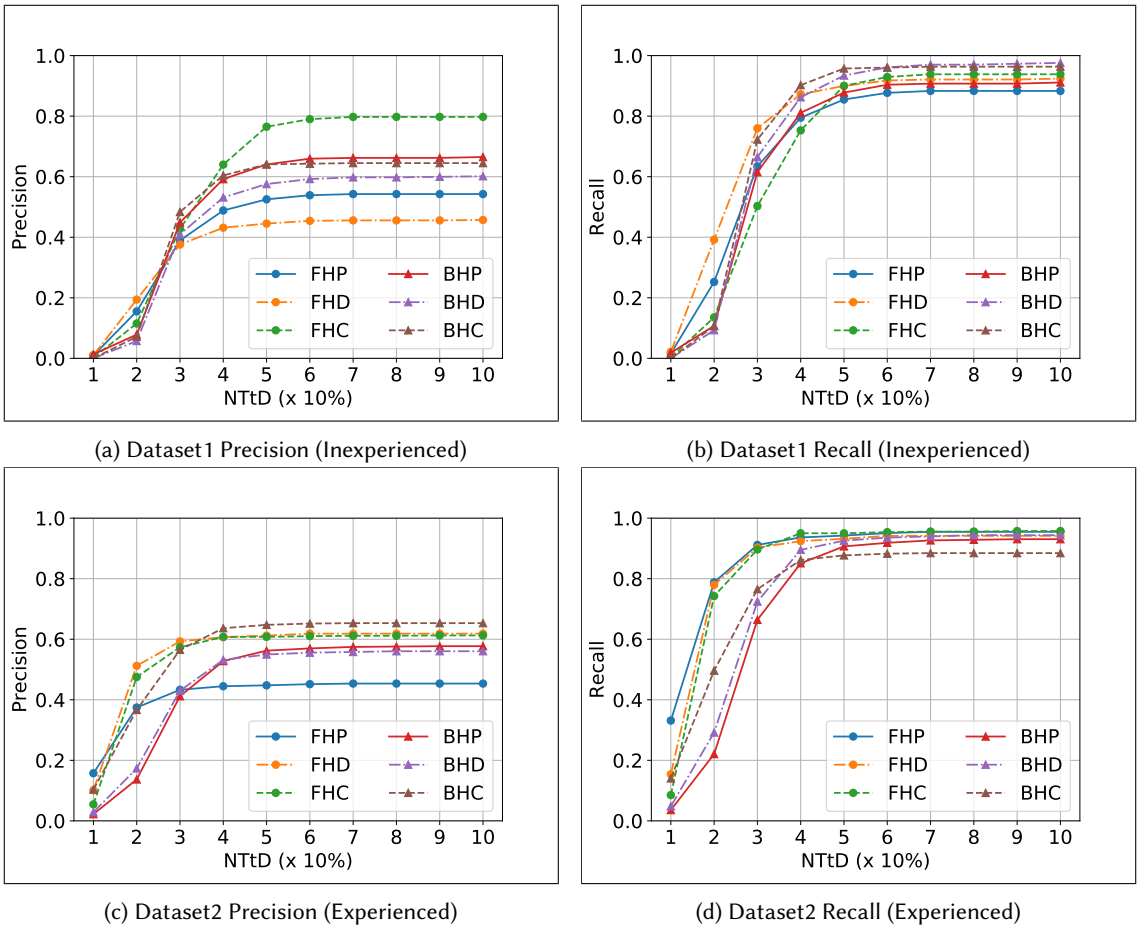
(d) Dataset2 Recall (Experienced)

Fig. 12. Box plot of *Left*: precision and *Right*: recall across users. X-axis corresponds to the 6 distinct gestures.

evolution to the final state incurs latency, and we cannot perform early detection of a gesture. On the other hand, if the system declares a gesture too early, it is more likely to produce a false positive.

A little reflection will show that determining the threshold is the main problem with the threshold method. Our addition of a classifier is intended precisely to tackle this conundrum and to reinforce the decision making of the system. In particular, the classifier observes the states, probabilities and gesture labels of every sample of training data, and can thus avoid recognizing a gesture under isolated samples of spurious observations. For example, in certain situations, a non-gesture may manifest in a high (close to final) HMM state and probability of *Forehand push* gesture; however, at that point, the hand may be pointing upward (x-axis value of gravity is high), making a *Forehand push* very unlikely. By using features corresponding to different evolutionary points of the gesture, our gesture recognizer becomes more *robust*.

Figure 13 demonstrates this point, by plotting performance of our stream-HMM model, but without the subsequent Random Forest classifier. We empirically set the state threshold to 3 (if *Normalized Expected State* is less than 3, the system decides that the sample belongs to a non-gesture). We see that our modified HMM



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)

(c) Dataset2 Precision (Experienced)

(d) Dataset2 Recall (Experienced)

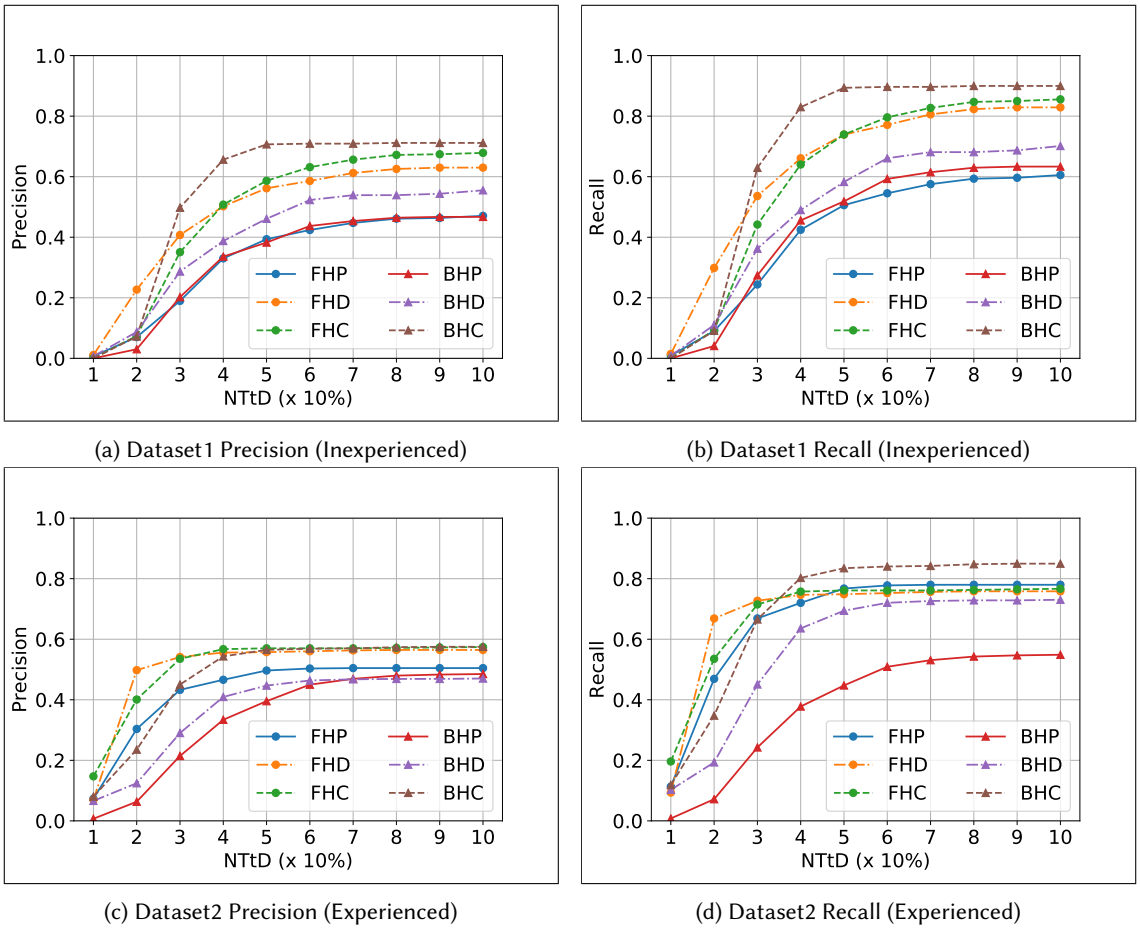
Fig. 13. Early detection ability of our stream-HMM model (without the subsequent classifier). It provides high recall. But the precision is quite low because many non-gestures have similar states and probabilities.

continues to be effective in detecting gestures correctly and early. However, it also detects many non-gestures as valid gestures (i.e., incurs false positives), thereby driving the precision of the system to be significantly lower.

6.3 Performance of Person-Independent Models:

To additionally understand the need for such personalized models, we also study the overall gesture recognition perform for a *person-independent* model. Note that, in general, the accuracy of person-independent gesture recognition algorithms is usually quite low–e.g., Thomaz et. al. [21] reported low values for precision and recall when detecting 'eating' gestures using a person-independent model.

We use the Leave One Person Out (LOPO) validation technique, whereby we train the model using $K - 1$ participants and test the system with the other participant, repeating this process for every distinct participant. As before, we measure the precision (See Figure 14a) and recall (See Figure 14b) as a function of NTtD, with a



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)

(c) Dataset2 Precision (Experienced)

(d) Dataset2 Recall (Experienced)

Fig. 14. Precision w.r.t. Normalized Time To Detect (Person-Independent Model).

step of 10%. As expected, the person-independent model gives rise to significantly lower accuracy. However, even then, we see that the early detection capability of the gestural model is relatively unaffected–e.g., for an NTtD value of 0.6 (i.e., at the time instant when a gesture is 60% complete), the recall of the person-independent model is higher than 50% for all gestures.

This LOPO validation shows that the development of a generalized model may not be impossible. All gestures show quite high recall and the precision is around 50%. The two gestures *Forehand Push* and *Backhand Push* have the lowest precision and recall in the study 1 (inexperienced players). As we mentioned before, the participants in this study are not professional players, so they usually play these two gestures very similarly to either *Drive* or *Chop* gestures. Even in study 2 (experienced users), the *Backhand Push* has significantly lower recall compared to other gestures. These results suggest that more sophisticated classification models (e.g., the Deep Learning-based LSTM models) may indeed offer more robust, person-independent classification. However, we do not pursue this investigation further in this paper, as the development of "improved classifiers" is not the core focus of our work.

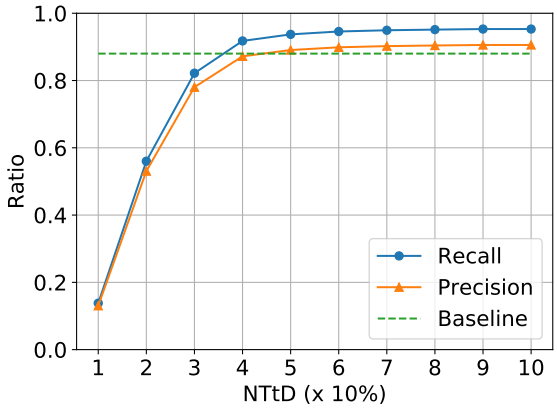


Fig. 15. Early detection capability of proposed method vs. enhanced E-Gesture baseline.

6.4 Comparison with E-Gesture Baseline

To evaluate how well our method can recognize gestures early, we compare the precision and recall of our proposed approach with a competitive baseline. As described in Section 5.1, the baseline is based on a modified version of the segmentation-cum-HMM model of [13], which we showed to provide better results. This baseline classifier operates on the data of each segment and outputs the gesture with the highest confidence value at *any* intermediate point of the gesture.

Figure 15 plots this comparative performance–with the horizontal line showing the precision and recall values for the baseline (which operates on the entire segment and thus does not vary with NTtD values). The figure shows that our proposed approach matches the precision and recall values of the baseline at an NTtD value of 0.4 (the precision is 87.0% compared to 88.0% of the baseline, and the recall is 92.0% compared to 88.0% of the baseline), implying *that we are able to detect gestures as accurately as the baseline (which needs to wait for the entire gesture segment) much earlier, using only the first 40% of a gesture.*

7 GESTURE-STATE-ENABLED TRAJECTORY TRACKING

While many studies rely on hand tracking to recognize gestures, hand tracking using sensors in a wearable device is not trivial because of the noise in sensory data. In this work, we explore the reverse problem: *Use knowledge of the gesture being performed to improve the accuracy of hand tracking.* This is based on the observation that during gestures of a specific type, a user's hand is likely to follow a more-limited "trajectory cone" in the 3-D space. In particular, if we can detect gestures accurately and early (which we've demonstrated in Section 5), we are likely to be able to estimate the position of user's hand based on the instant orientation of the hand more accurately.

7.1 Existing Approaches of Hand Tracking

A widely-used prior approach is to use dead-reckoning over the inertial sensing data. As we can extract the angle of the hand and its acceleration, we can intuitively apply the integral to calculate the 3D positions of the hand, assuming that we can obtain the initial reference position of the hand. However, this method often suffers from accumulated errors caused by noises in acceleration and orientation sensor readings. In addition, it is not easy to reset the reference point to offset the integral calculation once the table tennis play is started; note the periodic

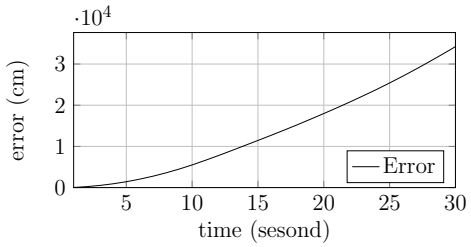


Fig. 16. Tracking error of the dead-reckoning approach.

reset of the reference point is essential for reckoning-based approaches to converge the accumulating errors. We implemented and evaluated this method over our dataset described in Section 4. Figure 16 shows the trajectory tracking errors as the time progresses. The tracking error increased to 300 meters after just 30 seconds, making the approach infeasible for our scenario.

ArmTrack [18] adopted a different approach to address this problem of error accumulation. Its key idea is to use a pre-built look-up table (LUT) that maps an orientation of the hand in a particular position to a specific 3-D location; in particular, they consider the possible angles of the shoulder and elbow joints to decide the mapping. This approach works well under the assumption that the user's shoulder is fixed. However, this method does not correspond well with our scenario, as table tennis interactions have complex kinematics and usually involve significant body movement during a game session. For the same stroke, the spine orientation, as well as the angles of the shoulder and elbow, could be highly variable, making it difficult to build an accurate look-up table. Moreover, for a VR-based game such as *VTT*, the wrist position should be computed, not relative to the body, but relative to the "virtual world" coordinates. For example, a user starting *VTT* on her VR display will see the board straight ahead, but as she moves her body, the location of her hand should shift, relative to the table (and thus the trajectory of an incoming ball), even if the hand doesn't move relative to the body. Accordingly, in our approach, (1) we assume the 'virtual world' coordinates to be defined by the user's orientation at the start of the game; (2) during subsequent game-play, we first compute the hand trajectory relative to the body (the user's forehead); (3) we separately track the forehead's motion, relative to the virtual world coordinates, using a smartphone (e.g., one rigidly mounted on a VR device such as Samsung Gear VR); and (4) finally, utilize the forehead motion vector to translate the hand's body-coordinates to the virtual world coordinates.

7.2 Gesture-State-Enabled Trajectory Tracking

To enable accurate, real-time hand tracking, we proposed a new method called *Gesture-state-enabled Trajectory Tracking*. The core idea behind this method is to utilize the intermediate states of the gesture's progress, along with the orientation of the hand. As described in Section 5, our gesture recognizer continuously outputs which gesture the player is likely performing as the gesture progresses. When the information of the gesture progress is combined with the hand orientation, it is possible to quickly narrow down the possible positions of the hand to a smaller, *plausible* area.

Figure 17 illustrates the trajectory tracking logic. As the first step, it computes two different features upon a sensor reading:

- *Orientation of the hand:* the trajectory tracker first computes the orientation of the hand, more specifically the orientation of the wrist-elbow limb. This can be calculated by computing the X-axis unit vector (1.0, 0.0, 0.0) of the smartwatch in the reference coordinate system (i.e., the coordinate system at the start of the game, when the table tennis board is directly in front of the user). The use of the x-axis component is

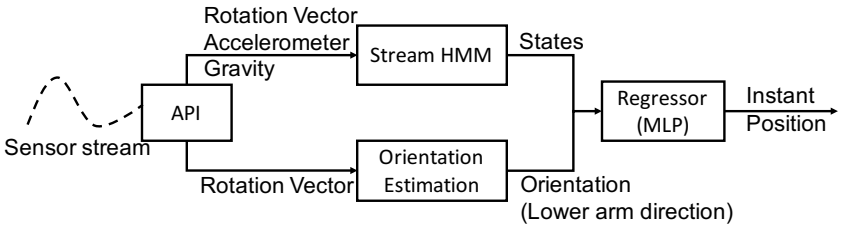
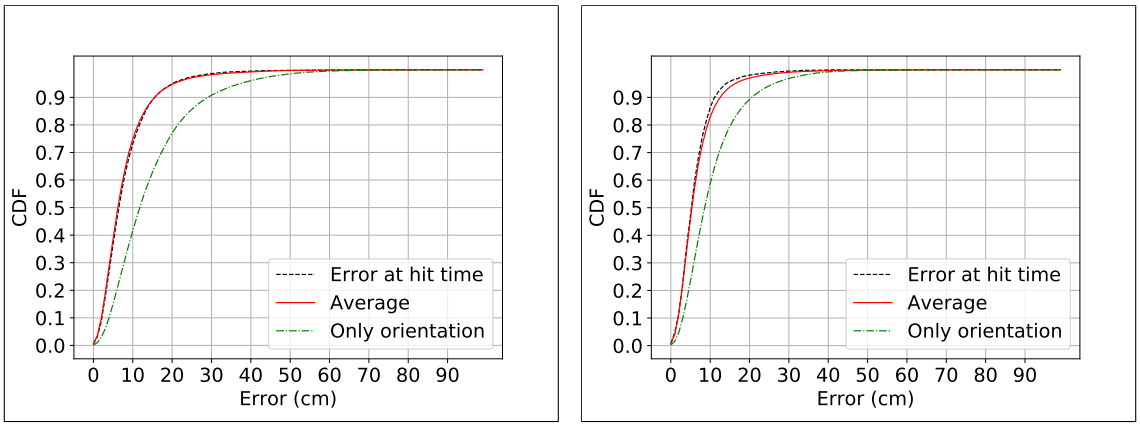


Fig. 17. The overall logic of trajectory tracking



(a) Dataset1 (Inexperienced)

(b) Dataset2 (Experienced)

Fig. 18. CDF of Average (and Hit Time) Trajectory Tracking Error.

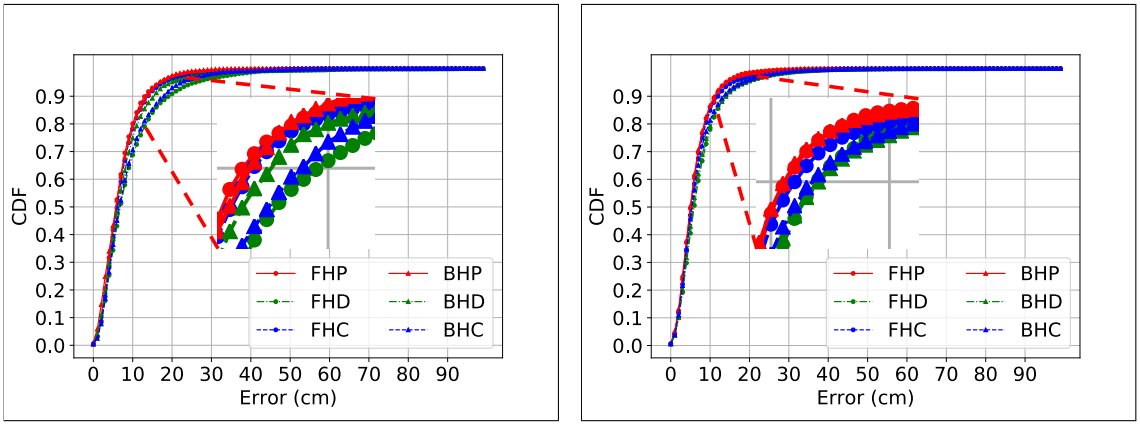
important as it is least affected by the wrist rotation; note that table tennis gestures involve wide rotations of the wrist, and thus, y-axis and z-axis components can vary hugely even at the same position of the hand.

- *Current state of the gesture:* it retrieves the current HMM states of the gesture, which indicates how much progress has been made for the gesture being currently performed. We still utilize the average states and probabilities which are described in section 5 as features to estimate the wrist position.

These two input values are streamed into a regression model, named *Multilayer Perceptron (MLP)*, which computes the possible location of the hand. We use a fully connected MLP (in Weka machine learning library [22]) with 3 layers of hidden units. Number of nodes in the 3 hidden layers are empirically set to 9, 18 and 9 correspondingly.The model is pre-trained with the hand trajectory data (computed using the "ground truth" video data of a player captured by the ZED camera). As mentioned before, the hand trajectory is defined relative to the 'virtual game' coordinates– this positional ground-truth is measured (and used in training the MLP) by tracking the user's head location, and adding the head-to-hand vector to this location.

7.3 Hand Tracking Performance

We evaluate the system using 10-fold cross validation (each gesture instance is a primitive unit). The data in this experiment is generated from the gesture recognition experiment. The states and wrist orientation of each test gesture are combined with the corresponding position ground truth to create a new hand tracking dataset.



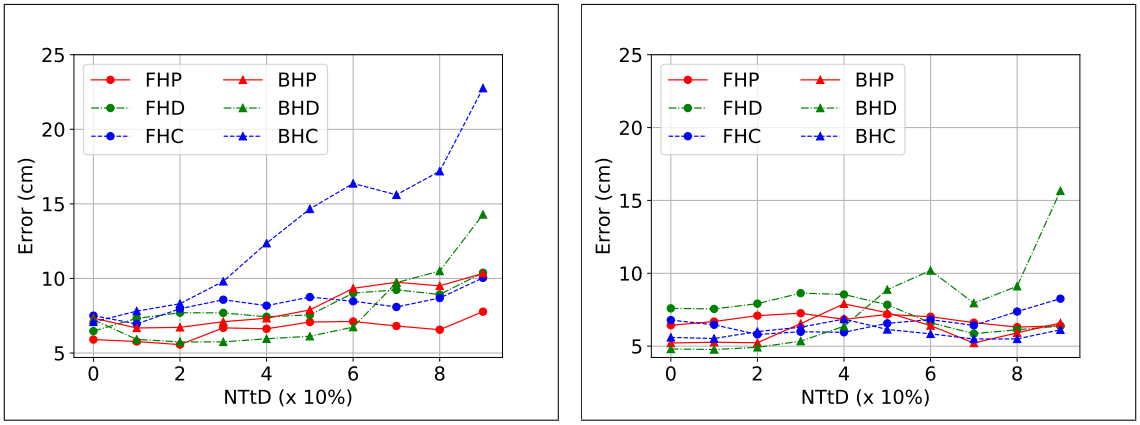
(a) Dataset1 (Inexperienced) (b) Dataset2 (Experienced)

Fig. 19. Stroke-specific tracking error distribution using (a): inexperienced user dataset, and (b): experienced user dataset.

Figure 18 shows the CDF of the average hand tracking error (the average computed over all points of a gesture instance), based on a personalized model. We compare the tracking error of our approach with a baseline (named "only-orientation") that performs regression, but without using the intermediate state of HMM. In addition, we also plot the CDF of the errors, computed only at the 'hit time' (when the bat hits the ball). While the baseline model achieves a median error of 9.5cm, our approach improves the tracking error significantly with a median error of 6.3 cm (for experienced users). The error at hit time (6.2cm) is slightly lower (but not by a significant amount) compared to the average error. As anticipated, the experiment with inexperienced users shows a slightly higher median error of 7.2cm. We can see a significant improvement (around 15-30cm) in tracking accuracy over the baseline at the 90^{th} percentile. The results show that using intermediate HMM-state information helps the regression model to estimate wrist position more accurately.

We further break down the tracking error per gesture type. Figure 19 shows the CDF of the average tracking error, for each of the 6 strokes. In this experiment (with experienced players), we observe a lower difference in the tracking error, across different gestures, for experienced users. However, at 90th percentile, we still see higher error values for the *Forehand Drive*, *Backhand Drive* and *Backhand Chop* gestures. The two *Drive* gestures are sometimes occluded in our training data because the player moves the racket towards the camera during the gesture, thus occluding the color marker on the wrist. The *Backhand Chop* also has the higher error, but less than the two *Drive* gestures. Both *Drive* gestures and *Chop* gestures usually involved fast movements of the arm, as a result of which the orientation and ground truth estimators are less accurate, compared to the slower hand movements in the other strokes.

We further investigate how the tracking error change as a gesture progresses. Figure 20 shows the tracking error as a function of the percentage progress of the gesture. Contrary to our expectation, there appears to be no discernible trend relating the tracking error to the progress of the gesture. There is an apparent significant increase in the error of *Backhand Drive* or *Backhand Chop* towards the end–however, note that these two gestures are often occluded, as a result of which the ground-truth samples of the hand's location are sometimes missing or not as accurate as the other strokes.



(a) Dataset1 (Inexperienced)

(b) Dataset2 (Experienced)

Fig. 20. Tracking error distribution vs. gesture progress, using (a): inexperienced user dataset, and (b): experienced user dataset.

8 USER PERCEPTUAL EXPERIENCE

The previous sections have demonstrated that our approach can recognize gestures early and accurately (with over 90% precision by the mid-point of a *VTT* gesture) and that it can then track the hand's trajectory, in real time, with an error of no more than 6-8cm. Clearly, such errors in tracking will manifest themselves in errors in estimating the *hit time*–i.e., when the racket makes contact with the ball. For a system such as *VTT* to be used, users must have the *perception* that the virtual experience faithfully recreates the "real world".

To study this perceptual effect, we recruited 5 players who are students in our university. We suspended a table tennis ball in the experiment room; each participant adjusted their body position, as well as the ball's height, such that they are comfortable in striking the suspended ball using the Forehand Push or Backhand Push strokes. Each participant wore a smartwatch (on their stroke-playing hand), which was instrumented with our gesture recognition software. The software was provided with the ball's *ground truth* location (computed using the Zed camera). The participant was then required to try to hit the ball with 25 Forehand Push shots and 25 Backhand Push shots. Whenever the smartwatch detected a gesture, it estimated the position of the wrist. If the estimated wrist position is within 8cm (which is about the diameter of the bat) of the suspended ball's position, then the system will emit a "beep" sound. We then asked the participant if they perceived the "beep" sound to be before, later or at the true *hit time* (the time when the user saw his hand actually hit the suspended ball). To avoid the need for building a personalized gesture model (which would have required additional training data from each user), we instead used a previously existing gesture recognition model derived from a participant of the prior studies. As expected, as this is a model trained on a different user, the gesture recognition accuracy is a bit lower than the results reported earlier (for personalized models).

As mentioned before, the user study measures users' perception of system response time. Figure 21 summarizes the users' response to our query: we see that for ≈ 85.7% of the gesture instances, users felt the "beep" sound happened concurrently with the true *hit time*. To further understand the likely usability of a system built using our technology, each user was asked (once each after the forehand and backhand sessions) about the "usability" of a future *VTT* training system built using this gesture recognition technology. The user rated the usability of the technique on a scale from varying from 1(unusable)-5 (absolutely usable). The average score across 5 users (based on sessions where the gesture recognizer was most accurate) is 4.2 out of 5. This result gives us confidence

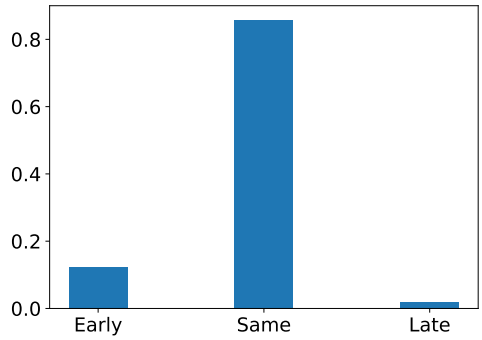


Fig. 21. User's perception of our system response time.

that our gesture recognition and hand tracking system can indeed be embedded in useful interactive applications (such as *VTT*).

9 RELATED WORK

Gesture recognition is, in general, an extensively studied problem. Similarly, there has been a spurt of recent activity investigating the use of smartwatches and other wearable devices to enable gestural interfaces. Given the large volume of research conducted on this topic, we focus primarily on the three aspects most relevant to our goals–(a) namely inertial-sensing based gesture recognition, (b) 3-D hand movement tracking and (c) early gesture recognition.

Smartwatch based Gesture Recognition & 3-D Tracking: One of the common applications of gesture recognition is touchscreen-based gesture input [3, 23], where patterns of touch trajectories are classified into different input commands. Like our problem, touch screen-based recognition also needs to be near-real time, to support interactive use. Touch screen based gesture recognition and inertial-sensing based gesture recognition can both be seen as general problems of pattern recognition. However, the inertial-sensing based approach need to cope with a more difficult segmentation problem because the system does not know when a gesture starts.

Park et al. [13] proposed an adaptive threshold based segmentation technique for recognition of custom gestures during immersive lifestyle activities. However, the use of a segmentation method implies that this system must wait for the entire gesture to be concluded before the gesture can be classified. Bevilacqua et al. [4] tackled the problem of recognizing the progressive evolution of gestures. However, they use the conventional HMM scheme on *phrases* of sensor readings, so the processing time is very high for long *phrases*, and is not suitable for applications requiring real-time recognition. Lee et al. [9] introduce a method which can recognize gestures from video without using any sliding window or segmentation. They introduce a loop-back transition to the *start* state. However, this *start* state does not generate any observation, so it is used to restart the model. Though this method does not require segmentation, it still requires the system to wait for a while after the gesture ends to confirm the gesture and restart the Viterbi algorithm.

Recently, Shen et al. [18] propose a technique which can compute full hand trajectories. They achieve it by modeling the space around the user as point clouds, then particle filtering algorithm is used to estimate hand position. This technique requires powerful hardware to run and imposes significant processing delay (a 1-second trajectory incurs about 2-seconds of processing delay).

Smartwatch based Daily Activity Recognition: Smartwatch-based sensing can be used to enable detection and analysis of a variety of gesturally-driven lifestyle activities. Typical studies include eating activity recognition [17, 21] or smoking recognition [12]. One of the problem in this type of application is the accuracy in uncontrolled environments because there can be many ambiguous hand motion that could be misclassified as gestures.

To improve the accuracy of gesture recognition, a technique called segmentation is also applied. Segmentation select potential segments of sensor data to classify, and thus reduce the false positive rate, but it may increase the false negative rate if the segmentation id not carefully crafted. Park et al. [13] propose adaptive threshold based segmentation which can change the threshold of acceleration based on mobility situation. However, this method requires the system to wait for a gesture to finish to avoid segmentation false. So the accuracy of the system depends on both classifier accuracy and segmentation accuracy. Furthermore, this technique is not applicable to applications that require early detection.

Gesture Recognition for Sports Activities: Smartwatch-based sensing has also been explored to detect a variety of sports-related gestures. Blank et al. [5] recently studied the classification of Table Tennis strokes. Instead of using smartwatch, they instrumented a racket with sensors accelerometer and gyroscope to capture the movement of the wrist and the rotation of the racket. This technique enabled the classifier to be able to differentiate advanced stroke with spin. However, the user has to have an instrumented racket to play Table Tennis. Moreover, they evaluate the system using only the different strokes without non-gesture presented. Therefore it is difficult to confirm if it can be generalized to real game situations.

Early Gesture Detection: Early detection of gesture is an interesting and open problem. Recently, researchers from NVIDIA Molchanov et al. [11] apply state-of-the-art machine learning techniques including both Convolutional Neural Network and Recurrent Neural Network to recognize hand gesture online from videos. They aim to recognize gesture with *negative lag*–i.e., before the gesture has ended.

10 DISCUSSION

Our experimental results show that our system outperforms competitive baselines by a significant margin–both in terms of the gesture detection latency and the 3-D hand tracking accuracy. However, there are several additional questions and issues that we must consider.

Limitation of Early Recognition: Our current approach for early detection is based on the features derived from the HMM state probabilities. The experimental table tennis data was, however, obtained from 10 familiar, but not expert, users. It is possible that the speed of the gestures may vary considerably across expertise level. To provide robustness against such significant variation in gesture execution speeds, we may need to explicitly consider the *elapsed time duration* from the start of the gesture as a feature, in addition to the HMM state probabilities. Moreover, the early detection approach implicitly assumes that the gestures are prefix-free–i.e., one gesture isn't a sub-trajectory of another gesture. If the gestures are not prefix-free, then the observation interval will need to be larger than the duration of the smallest prefix, to help disambiguate between multiple gestures.

Limitation of hand tracking: The current experimental study focused primarily on determining the trajectory of the hand, under the assumption that the body position and orientation was relatively unchanged. Clearly, for applications where the body position can vary, we will need to fuse the body orientation together with the smartwatch orientation to get improved 3-D tracking of hand movement. In particular, the gesture detection logic is currently based on features that are computed in the earth's reference frame, whereas the actual gesture trajectory may vary depending on the body orientation. Accordingly, accurate gesture recognition itself may require fusing of the hand orientation with the body orientation/posture data (e.g., obtained from the smartglasses)–this may be difficult to execute at the desired low latency.

Multi-Limb Gestures: Our current approach has focused on tracking the gesture and trajectory of a single arm. There may be applications that require simultaneous tracking of multiple limbs, or where the gesture itself is

based on the movement of both arms. In such cases, the current early gesture detection approach may not be suitable, as the detection of the *composite* gesture may require the fusion of sensor streams on multiple devices. The resulting communication latencies (e.g., from Bluetooth or WiFi Direct links) may be too high (often several hundred msec), negating the benefits of early gesture recognition.

11 CONCLUSION

Early detection of gestures and hand tracking are two problems which need to be solved to support natural gestural interaction in immersive, interactive applications. Moreover, full hand tracking is either inaccurate with wearable sensors, or is computationally expensive. To tackle these challenges, we proposed two new techniques: (i) one that supports early gesture recognition by using a classifier that uses features based on the probabilities of the states of multiple HMMs, and (ii) a 3-D trajectory tracking approach that uses a per-gesture regressor to infer the hand motion trajectory. Experimental results show that our approach provides significant improvements, in both latency and accuracy, over the conventional two-stage approach of segmentation followed by HMM-based gesture recognition. In ongoing work, we are embedding this technique as part of a new real-time gesture recognition library, and then exploring enhancements to support such low-latency tracking in multi-device, multi-limb scenarios.

ACKNOWLEDGMENTS

This work was supported by Singapore Ministry of Education Academic Research Fund Tier2 under research grant MOE2014-T2-1063. All findings and recommendations are those of the authors and do not necessarily reflect the views of the granting agency, or Singapore Management University.

REFERENCES

[1] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. 2011. Using mobile phones to write in air. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 15–28.

[2] Oliver Amft, Holger Junker, and Gerhard Troster. 2005. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*. IEEE, 160–163.

[3] Olivier Bau and Wendy E Mackay. 2008. OctoPocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 37–46.

[4] Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, and Nicolas Rasamimanana. 2009. Continuous realtime gesture following and recognition. In *Gesture in embodied communication and human-computer interaction*. Springer, 73–84.

[5] Peter Blank, Julian Hoßbach, Dominik Schuldhaus, and Bjoern M Eskofier. 2015. Sensor-based stroke detection and stroke type classification in table tennis. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ACM, 93–100.

[6] Minh Hoai and Fernando De la Torre. 2014. Max-margin early event detectors. *International Journal of Computer Vision* 107, 2 (2014), 191–202.

[7] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2291–2300. https://doi.org/10.1145/2470654.2481317

[8] Holger Junker, Oliver Amft, Paul Lukowicz, and Gerhard Tröster. 2008. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition* 41, 6 (2008), 2010–2024.

[9] Hyeon-Kyu Lee and Jin-Hyung Kim. 1999. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence* 21, 10 (1999), 961–973.

[10] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675.

[11] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4207–4215.

[12] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. 2014. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th annual international conference on Mobile systems, applications,*

and services. ACM, 149–161.
[13] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. 2011. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 260–273.
[14] Krzysztof Pietroszek, Liudmila Tahai, James R Wallace, and Edward Lank. 2017. Watchcasting: Freehand 3D interaction with off-the-shelf smartwatch. In *3D User Interfaces (3DUI), 2017 IEEE Symposium on*. IEEE, 172–175.
[15] Pingskills. 2017. Pinkskills. https://www.pingskills.com/tutorials/advanced-strokes/forehand-chop. (2017). [Online; accessed 15-November-2017].
[16] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. 2008. Gesture recognition with a Wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*. ACM, 11–14.
[17] Sougata Sen, Vigneshwaran Subbaraju, Archan Misra, Rajesh Krishna Balan, and Youngki Lee. 2015. The case for smartwatch-based diet monitoring. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 585–590.
[18] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I am a Smartwatch and I can Track my User's Arm. In *Proceedings of the 14th annual international conference on Mobile systems, applications, and services*. ACM, 85–96.
[19] StereoLabs. 2017. Zed stereo camera. https://www.stereolabs.com/. (2017). [Online; accessed 15-November-2017].
[20] Thomas Stiefmeier, Daniel Roggen, and Gerhard Troster. 2007. Gestures are strings: efficient online gesture spotting and classification using string matching. In *Proceedings of the ICST 2nd international conference on Body area networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 16.
[21] Edison Thomaz, Irfan Essa, and Gregory D Abowd. 2015. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1029–1040.
[22] Weka. 2017. Weka 3: Data Mining Software in Java. https://www.cs.waikato.ac.nz/ml/weka/downloading.html. (2017). [Online; accessed 15-November-2017].
[23] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 159–168.
[24] ZoomTT. 2017. The Fastest Sport. https://zoomtt.com/2017/04/15/fastest-sport-table-tennis-vs-badminton/. (2017). [Online; accessed 15-May-2017].

Received May 2017; revised November 2017; accepted January 2018