I am a Smartwatch and I can Track my User's Arm

Sheng Shen, He Wang, Romit Roy Choudhury
University of Illinois at Urbana-Champaign
{sshen19, hewang5, croy}@illinois.edu

ABSTRACT

This paper aims to track the 3D *posture* of the entire arm – both wrist and elbow – using the motion and magnetic sensors on smartwatches. We do not intend to employ machine learning to train the system on a specific set of gestures. Instead, we aim to trace the geometric motion of the arm, which can then be used as a generic platform for gesture-based applications. The problem is challenging because the arm posture is a function of both elbow and shoulder motions, whereas the watch is only a single point of (noisy) measurement from the wrist. Moreover, while other tracking systems (like indoor/outdoor localization) often benefit from maps or landmarks to occasionally reset their estimates, such opportunities are almost absent here.

While this appears to be an under-constrained problem, we find that the pointing direction of the forearm is strongly coupled to the arm's posture. If the gyroscope and compass on the watch can be made to estimate this direction, the 3D search space can become smaller; the IMU sensors can then be applied to mitigate the remaining uncertainty. We leverage this observation to design *ArmTrak*, a system that fuses the IMU sensors and the anatomy of arm joints into a modified hidden Markov model (HMM) to continuously estimate state variables. Using Kinect 2.0 as ground truth, we achieve around 9.2 cm of median error for *free-form* postures; the errors increase to 13.3 cm for a real time version. We believe this is a step forward in posture tracking, and with some additional work, could become a generic underlay to various practical applications.

Keywords

Gesture, arm posture, kinematics, anatomy, tracking, smartwatch, hidden Markov model, accelerometer

1. INTRODUCTION

Analytics on human leg motion has fueled an industry on mobile health and well being. Nowadays, walking, running, biking and various other activities can be recognized from motion sensors embedded in smartphones and wearable devices. Understanding upper limb motion seems like the logical next step, and various research groups/start-ups have already made progress. Authors in [11, 21, 33, 35], for example, have employed various machine learning algorithms to detect meaningful arm and hand gestures – smoking, eating, typing, writing – on wearable wrist bands. Rithmio [4], perhaps the most advanced start-up in this space, is eliminating the need for training, so long as the user performs repetitive tasks, such as bouncing a basketball or exercises in the gym. Finally, for applications requiring full arm posture reconstruction (e.g., golf swing analysis, animation movie characters), today's solutions paste multiple sensors on the arm, or adopt computer vision based analytics [3, 5, 6, 8, 12, 13, 15–17, 20, 23, 28–31, 36, 38, 39]. This paper aims to construct the 3D arm posture using smartwatch sensors alone, and track the posture continuously over time, without any training. In addition, we desire the tracking techniques to be light-weight so they can be amenable to real time applications.

Before further discussion, we briefly clarify the notion of "postures" from "gestures". By posture, we mean the 3D geometric model of the arm. For a fixed shoulder location, arm posture is uniquely defined by 3 parameters – *elbow location*, *wrist location*, and *wrist rotation*. The wrist rotation captures the rotation of the wrist around the axis of the forearm[1]. A gesture, on the other hand, is a specific sequence of arm postures that carry semantic meaning (somewhat analogous to how words are meaningful sequences of alphabets). Hand gestures typically refer to gestures of the wrist, not necessarily the arm. This paper tracks the entire arm posture in 3D space over time (similar to a Kinect), and is expected to serve as building blocks to any application-defined gesture.

Designing *ArmTrak* entails 3 key research questions:

(1) The state space of the entire arm is large, meaning that the elbow and wrist could take up many configurations around the body. Without any pre-defined patterns to search for, the arm posture tracking problem translates to a Bayesian tracking problem in continuous space. While tracking is a mature area in signal processing, most of the problems are either guided by good motion models or able to obtain measurements directly from the object of interest. In our case, smartwatch sensors do not offer direct measurements from the elbow, are noisy, and lack models of how the arm is expected to move. To the best of our knowledge (based on literature survey in signal processing, robotics, and mobile computing), this still remains an unaddressed problem.

[1]For a fixed elbow and wrist location, the wrist rotation changes the palm's facing direction.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiSys'16, June 25-30, 2016, Singapore, Singapore
© 2016 ACM. ISBN 978-1-4503-4269-8/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2906388.2906407

(2) It is possible that continuous space techniques, such as Particle Filters or an appropriate variant, map to posture tracking. However, such techniques incur high complexity and latency – when considering scalability to many users, or the real-time requirements of certain applications, the approaches prove prohibitively expensive. Low complexity and fast run-time are important factors for a practical end to end system.

(3) The final problem pertains to expressing the arm posture in different coordinate systems. For applications where a user is pointing to a TV to turn it on, it is important to understand the direction of pointing in the global reference frame. For other applications, like golf-swing analysis, hand posture needs to be tracked in the torso's coordinate system. The core problem is rooted in detecting the human's facing direction from the watch sensors. *ArmTrak* must resolve this problem to cater to various application needs.

The perspective we bring to the problem pertains to a synthesis of anatomy, sensor fusion, and Bayesian inference. From the anatomical models of shoulder and elbow joints, we observe that for a given 3D orientation of the wrist (which is estimated via sensor fusion using accelerometer, compass and gyroscope), the space of possible elbow locations is quite constrained. Given that the elbow is also constrained on a sphere around the shoulder point, we can further reduce the search space for the elbow – called a *point cloud*. Now, using the (rotation polluted) accelerometer data, we estimate the translational motion of the elbow through a hidden Markov model (HMM) framework, but apply the point cloud as a prior. Once the elbow location is known, the wrist location is computed as a simple shift along the (forearm pointing) direction prescribed by the wrist orientation. To cater to applications, we make a series of optimizations, resulting in an option to prioritize either accuracy or latency. On one extreme, Viterbi Decoding yields the best results but after offline processing; on the other extreme, we compute an averaging on the point cloud to operate in real-time. Finally, we use a combination of the watch orientation and the compass to opportunistically estimate the user's facing direction, ultimately yielding the arm posture in the desired coordinate system.

We evaluate *ArmTrak* using Samsung Gear Live smartwatches, with the sensor data processed on the watch (in real time) as well as on the cloud (running MATLAB). Recruited volunteers stand in front of a Kinect 2.0 sensor and perform various kinds of gestures, starting from simple wrist movements all the way to random, free-form arm gestures. The skeletal models from the Kinect serve as ground truth, and we report *ArmTrak*'s accuracy as a function of both the wrist location and elbow location errors. We also report the degradation in our accuracy in exchange for the improvement in latency. On average, our $\langle elbow,\ wrist \rangle$ posture tracking results are $\langle 7.9cm,\ 9.2cm \rangle$ respectively in the offline setting, and drop to $\langle 12.0cm,\ 13.3cm \rangle$ when performed in the "fast" mode. More importantly, the tracking errors remain bounded over time, allowing for continuous gesture recognition We encourage our readers to watch the video demonstration of our system here: http://synrg.csl.illinois.edu/posture/.

Besides what is achievable, we must also discuss the shortcomings of the current system. (1) We believe that ferromagnetic materials in indoor environments can present important ramifications on accuracy; our experiments were performed in our lab with stable magnetic ambience. (2) Our techniques falter when the user performs gestures while on the move – the sensor data from the motion pollutes both posture tracking and facing-direction estimation. (3) Finally, gyroscopes are known to consume energy – we have ignored the energy considerations in developing *ArmTrak*. In view of these capabilities and deficiencies, we summarize the contribution in this paper as follows.

- *Using sensor data from smartwatches to track the posture of the entire arm.* Using observations from anatomical models to constrain the search space for the elbow, a key enabler for 3D posture tracking.

- *Using the accelerometer data as an input to a (modified) hidden Markov model, ultimately tracking the motion of the elbow (and the wrist).* Parameterizing the system to achieve different tradeoffs between accuracy and latency, and offering them as a single knob to application developers.

2. PROBLEM SETUP

This section discusses the basics of smartwatch motion and posture tracking, and the deliberations that led us to the proposed ideas.

2.1 Torso Coordinate System

We will define the posture of the arm, and its motion, in the *torso coordinate system*. In this system, the left shoulder will serve as the origin, and the plane of the user's torso (i.e., the chest) will serve as the XY plane. The Z axis will be the line emanating from the left shoulder in the frontward direction, perpendicular to the torso. The Kinect also models its skeleton tracking data in a similar coordinate system – since we use the Kinect as ground truth, aligning our coordinate system with Kinect simplifies our evaluation process.

The torso coordinate system (TCS) is desirable to most applications, although some need the arm posture to be expressed in a global (North-East) coordinate system (GCS). For instance, analysis of gym exercises, golf swing analysis, smoking recognition, etc., can all be performed in the TCS framework. However, when controlling devices in a room (e.g., pointing to a TV to turn it on), the posture of the arm needs to be modeled in global coordinates. The compass on the watch offers the necessary information to estimate postures in GCS, however, its translation to TCS requires knowledge of the user's facing direction. We will develop the overall *ArmTrak* system assuming knowledge of the facing direction, and then relax the assumption through a facing-direction estimator.

2.2 Wrist/Elbow Orientation and Location

Figure 1 shows the X, Y, and Z axes of a smartwatch when a user wears it on her left wrist. These axes, that are local to the watch's coordinate system, can easily be expressed as vectors in the torso coordinate system, denoted as $\vec{X}_t$, $\vec{Y}_t$, and $\vec{Z}_t$ (the subscript means the vector changes over time as arm moves). We define the "orientation" of the watch (same as the orientation of the wrist) as this tuple: $\langle \vec{X}_t, \vec{Y}_t, \vec{Z}_t \rangle$. $\vec{X}_t$ always aligns with the pointing direction of the forearm in 3D space, and for a fixed forearm pointing direction, both $\vec{Y}_t$ and $\vec{Z}_t$ change along with the rotation of the forearm around the X axis.

Similarly, location of the elbow or the wrist (same as the location of the watch) can also be expressed as a separate 3D tuple, $\langle x_t, y_t, z_t \rangle$, in the torso coordinate system (TCS). Once the wrist orientation is known, the elbow location and the wrist location

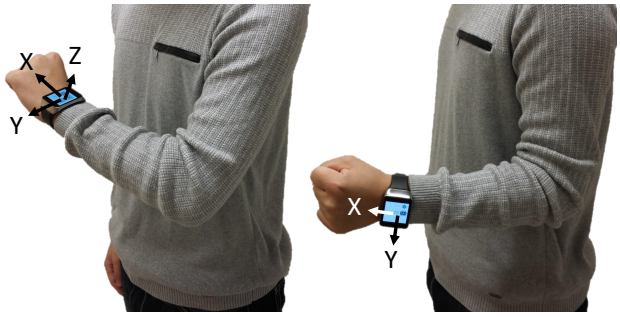

Figure 1: Smartwatch orientation changes with the pointing direction of the hand – the orientation is measured in the torso's reference frame.

are simply a *static shift* of each other, along the positive or negative forearm pointing direction (wrist's $\vec{X}_t$ orientation). The static shift is the length of the forearm, and thus needs to be known only once. With this, the posture of the entire arm can be determined in the torso coordinate system. The natural question is: how can the arm posture be tracked over time? Actually, we can ask a simpler sub-question: can the wrist location even be tracked over time? We make a few relevant observations next.

2.3 Noise: The Fundamental Problem

From basic physics, any motion of a body can be decomposed into translational and rotational motion. Thus, when a wrist moves from point A to a very close point, B, one can model this as a translational motion of the watch from A to B, followed by a change in orientation to reflect the orientation at B. Now, assume that the initial location and orientation are known at point A, and the accelerometer and gyroscope are super accurate. Then, the gyroscope can measure the angular velocity around each of the $\vec{X}_t$, $\vec{Y}_t$, and $\vec{Z}_t$ directions, and precisely estimate the orientation at point B. The accelerometer, on the other hand, measures a combination of translational and rotational motion (hence, plain double integration will not work). Instead, the double integration can be performed at infinitesimally small time steps, and after each step, the orientation of the device can be updated based on the corresponding gyroscope data. In other words, the system will be able to compute the linear displacements and rotations at extremely fine granularity, and concatenating them should result in perfect tracking.

Of course, the above is true under the assumption of perfect IMU sensors. With noisy sensors, we implemented the same algorithm (and appropriately subtracted gravity) to quantify the extent of divergence. Figure 2 shows the results – the orientation divergence is somewhat reasonable[2], but the translation error is excessive, more than 100 meters within 1 minute. In other words, deterministic techniques will always be affected by the randomness of noise; stochastic inference techniques are likely to be the appropriate approach.

2.4 An Estimation Problem: Particle Filter

The problem we are facing is obviously not new – robotics and signal processing researchers routinely face and solve these kinds of estimation problems (also called filtering). Briefly, the *state* of the object is modeled as variables and the range of these

[2]We cannot measure all 3 dimensions of orientation using Kinect, hence plot the error from 2 dimensions, calculated as the angular difference between the forearm's pointing direction and the ground truth.

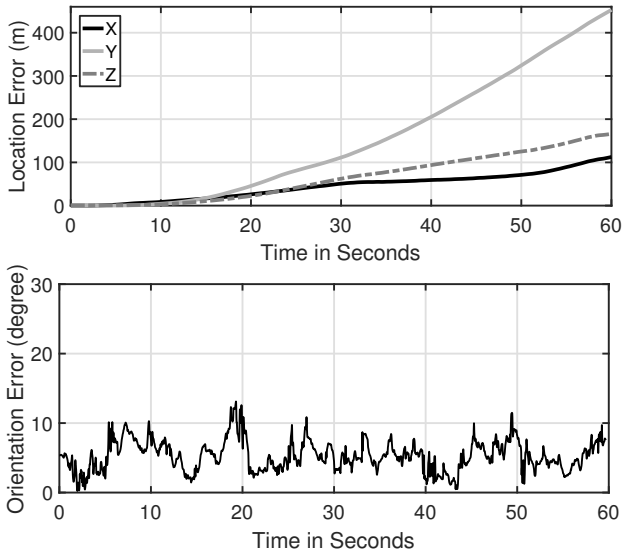

Figure 2: (a) Wrist location error diverges with double integral. (b) Wrist orientation error remains small over time.

variables together describe the *space* in which the object can exist. Then, based on some model of how the object is expected to move, and what the data reveals about its actual motion, these estimation algorithms compute the most likely state of the object. We implemented a particle filter, one of the popular state estimation algorithms in continuous space. However, given that the accelerometer and gyroscope data are differentials of location and orientation, the state of particles had to be defined with many variables to capture the entire arm posture. This resulted in a high-dimensional system and the estimator could hardly converge. We aborted the effort and focused on reducing the state space of the system for good tracking accuracy.

3. OPPORTUNITY AND VALIDATION

Following the failure of the particle filter, we focused on opportunities to reduce the state space of the system, i.e., constraining the possible postures of the arm. This seemed intuitive, i.e., since the arm joints have limits in their *range of motion* (RoM) [7, 14], they should constrain the arm postures as well. In exploring the arm joints and measurement data, we made the following empirical observation. Assume the shoulder location is fixed. *It appeared that for a fixed wrist orientation, the possible space of wrist locations is quite limited.* In other words, if one moves her wrist around without changing the wrist orientation, there are not many locations to which she can take her wrist.

To understand this intuition, let's first assume that we keep the elbow location fixed. Consider how the motion of the forearm will influence the wrist location and orientation. Since the elbow does not move, any forearm motion will change the wrist's orientation, no matter it is the twist of the wrist/forearm (which will change $\vec{Y}_t$ and $\vec{Z}_t$ of orientation) or the rotational motion around the elbow (which will change $\vec{X}_t$ of orientation and also change the wrist's location). Conversely, for a given wrist orientation, only one wrist location is possible (as it has to be along the $\vec{X}_t$ direction emanating from the elbow), under these artificial assumptions.

Of course, once the elbow starts moving, the wrist can move to multiple locations while preserving the same orientation. However, the elbow can only move on a sphere around the shoul-

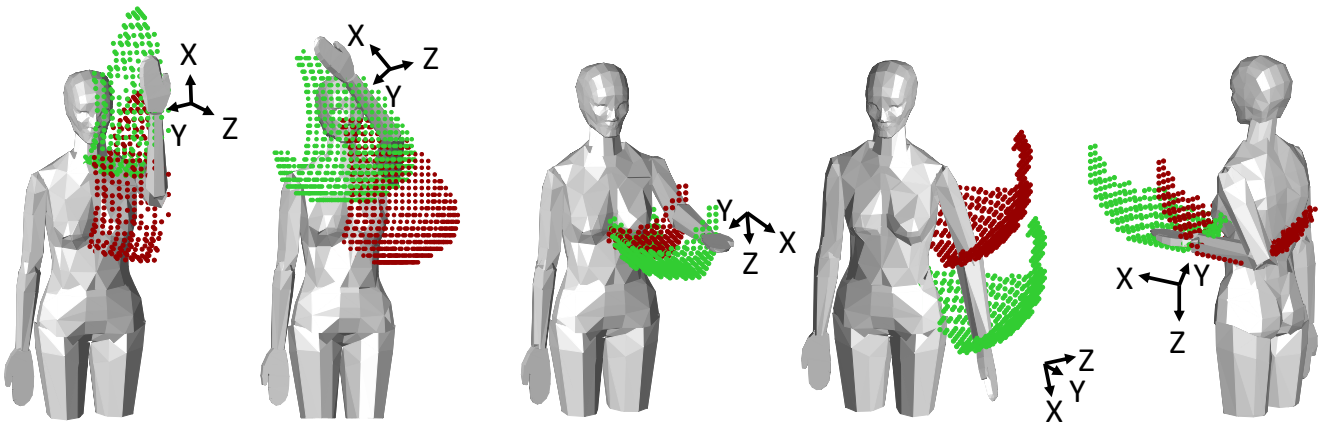

Figure 3: 5 different watch orientations and the corresponding wrist and elbow point clouds (shown in light green and dark red, respectively). The point clouds (essentially the space of feasible wrist and elbow locations for the given orientation) are relatively small and narrow down the uncertainty of the user's arm posture.

der, and the forearm's ability to twist is relatively limited. This suggests that for a given wrist orientation, the possible space of wrist locations may be reasonably restricted. The space will also vary across orientations, i.e., some wrist orientations will allow the wrist to move to more locations than others.

3.1 Preliminary Validation

As preliminary validation, we visualized the space of wrist locations for some wrist orientations. Figure 3 shows 5 example wrist orientations, along with the corresponding wrist and elbow location point clouds, marked in (light) green and (dark) red, respectively. The findings exhibit promise – the wrist location space is indeed a small fraction of the entire 3D space around the shoulder. Also, the elbow and wrist point clouds exhibit a 1:1 mapping, since they are simply a static shift of one another.

3.2 Formalizing through Arm Models

Figure 3 indicates the opportunity, but we need to generalize our observation. Therefore, we derived models from human arm kinematics, and reorganized them to formally express the relationship of wrist orientation, wrist location and elbow location. We describe the models here, followed by a quantification of state space reduction.

In robotics, a human arm is often modeled using 7 rotational degrees of freedom (DoF) [22] – 3 for the shoulder, 2 for the elbow, and 2 for the wrist. Since the watch is worn on the forearm near the wrist, the DoFs of the wrist are not manifested in the watch's sensor data. The remaining 5 DoFs define the state of the watch, as shown in Figure 4. When these 5 values are combined with the *known* lengths of the upper arm and forearm, the watch's location and orientation can be estimated uniquely.

Modeling this mathematically, let θ_1, θ_2, θ_3, θ_4 and θ_5 denote the 5 DoFs; let l_u and l_f denote the lengths of the upper arm and forearm. Using these, the Denavit-Hartenberg transformation [9] actually outputs the posture of the entire arm. For example, the elbow location is a function of θ_1 and θ_2, and can be expressed as:

$$\text{loc}_{\text{elbow}} = f(\theta_1,\theta_2) = l_u \begin{pmatrix} \cos(\theta_2)\sin(\theta_1) \\ \sin(\theta_2) \\ -\cos(\theta_1)\cos(\theta_2) \end{pmatrix} \quad (1)$$

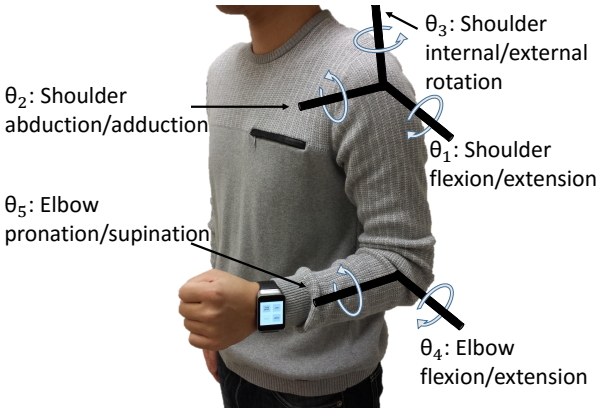

Figure 4: 5-DoF Arm Model showing the possible angular rotations.

and it satisfies

$$\| \text{loc}_{\text{elbow}} \| = l_u \quad (2)$$

Similarly, the wrist's relative location to the elbow can also be computed as

$$\text{loc}_{\text{wrist-to-elbow}} = g(\theta_1,\theta_2,\theta_3,\theta_4) \quad (3)$$

where the function $g()$ is a long equation omitted in the interest of space, but it of course satisfies

$$\| \text{loc}_{\text{wrist-to-elbow}} \| = l_f \quad (4)$$

Thus, the wrist's absolute location can be written as the vectorial addition of the elbow location and the wrist's relative location (to the elbow).

$$\text{loc}_{\text{wrist}} = \text{loc}_{\text{elbow}} + \text{loc}_{\text{wrist-to-elbow}} \quad (5)$$

Like location, the orientation of the wrist, expressed in the form of rotation matrix, can also be computed through a rotational function on the 5 θs (the function $h()$ omitted in the interest of space).

$$\text{Rot}_{\text{watch}} = h(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5) \quad (6)$$

In summary, knowing these 5 θs can solve the entire state of the arm posture, i.e., wrist orientation, wrist location and elbow location.

3.3 Mapping Orientation to Point Cloud

For a given watch orientation, we intend to map it to the elbow and wrist's location point clouds. We derive the mapping to the elbow first, because it lies on a sphere around the shoulder which will later make the model mathematically easier. The translation from the elbow to the wrist will be a static shift.

Now, to derive the elbow's point cloud, we first referred to some medical papers [7, 14, 24] and summarized the average range of motion (ROM) for each joint angle in Table 1. Here $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = 0°$ refers to the posture where the left arm is in free-fall on the left side of the torso, with the palm facing front. Then, for each watch orientation, we find all combination of $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$'s within the ROM that can generate that orientation according to (6). Each combination will map to one elbow location according to equation (1). Thus, we obtain a mapping from $\text{Rot}_{\text{watch}}$ to possible $\text{loc}_{\text{elbow}}$'s. We can also derive the mapping from $\text{Rot}_{\text{watch}}$ to possible $\text{loc}_{\text{wrist}}$'s easily, because for each $\text{Rot}_{\text{watch}}$, possible wrist locations are simply a shift of possible elbow locations, along the forearm's pointing direction – the equation described below.

$$\text{loc}_{\text{wrist-to-elbow}} = \text{Rot}_{\text{watch}} \begin{pmatrix} l_f \\ 0 \\ 0 \end{pmatrix} \quad (7)$$

Algorithm 1 presents the pseudo code.

Joint Angle	Min. Value	Max. Value
θ_1	-60°	180°
θ_2	-40°	120°
θ_3	-30°	120°
θ_4	0°	150°
θ_5	0°	180°

Table 1: Range of motions for each joint angle

Algorithm 1 Watch Orientation to Point Cloud Mapping

```
ElbowPointCloud = Empty Dictionary
WristPointCloud = Empty Dictionary
for all {θ1,θ2,θ3,θ4,θ5} ∈ ROM do
    loc_elbow = f(θ1,θ2)
    Rot_watch = h(θ1,θ2,θ3,θ4,θ5)
    loc_wrist-to-elbow = Rot_watch(t) (l_f, 0, 0)^T
    loc_wrist = loc_elbow + loc_wrist-to-elbow
    ElbowPointCloud[Rot_watch].Add(loc_elbow)
    WristPointCloud[Rot_watch].Add(loc_wrist)
end for
```

To quantify the reduction in uncertainty due to this mapping, Figure 5 plots the CDF of the elbow's point cloud, as a fraction of the surface area of the sphere around the shoulder. In 90% cases, the elbow can only reach $\frac{1}{4}$ of whole sphere area, and the median fraction is 8.3%. Of course, the fraction can be further reduced if we utilize the fact that these 5 DoFs are not entirely independent and thus model their RoMs jointly (instead of setting an upper/lower bound for each of the joint angle). We leave this optimization to future work.

If the mapped point cloud is moderately accurate (assuming that the orientation estimation is reasonably error-free), the next step

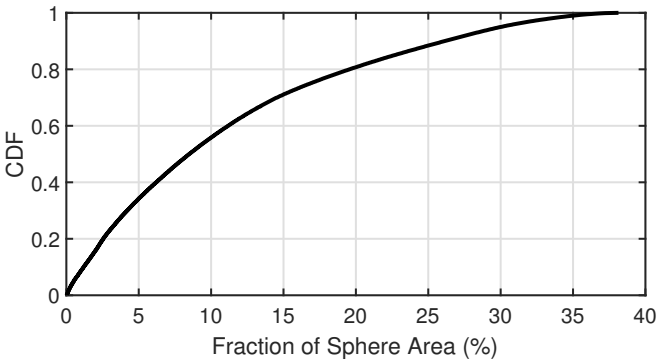

Figure 5: Elbow location subspace as a fraction of the sphere area (around the shoulder) – the state space is certainly smaller.

is to leverage the point cloud in a state estimation framework. This motivates a discrete space hidden Markov model, with the elbow's point cloud as the prior. We describe these techniques next, as a part of a full posture recognition system.

4. ARCHITECTURE

Figure 6 illustrates *ArmTrak*'s overall architecture. The raw sensor data from the smartwatch – composed of the accelerometer, gyroscope, and compass samples – are passed through an *Orientation Estimation Module* (OEM). This module computes the watch's orientation in the earth's coordinate system (ECS), using a borrowed technique called A^3 from MobiCom 2014 [40]. Since the user's facing direction is unknown, the transformation between ECS to TCS is still unknown. The *Facing Direction Module* (FDM) scans the sensor data stream and opportunistically recognizes samples that reveal the facing direction. The orientation is now transformed to TCS and forwarded to the *Orientation to Point Cloud Module* (OPM).

OPM consists of a pre-loaded mapping between orientation and point clouds (the mapping derived from our arm-joint models described earlier). Using the incoming orientation as an index, OPM outputs the corresponding point cloud. Not all candidates may be equally likely, therefore, a separate *Posture Priors Module* (PPM) analyzes general human arm motions and extracts priors. This information is used to bias the posture estimation process towards the sequences that are more likely in humans.

For applications that require high accuracy in arm posture estimation, the outputs of both OPM and PPM are forwarded to a Hidden Markov Model (HMM), along with the raw sensor data. The HMM observes sequences of data and estimates the most likely arm posture sequence. The outputs are favorable to applications that need higher accuracy and can tolerate latency in several seconds (even though the HMM has been carefully optimized for lower complexity). However, if some applications require a real-time arm posture, the outputs of OPM and PPM are sent together into an Averaging Filter. This filter computes a weighted average of all candidate arm postures for the current orientation, where the weights are guided by the priors. This serves as a faster but less accurate arm posture tracker.

4.1 Design for Higher Accuracy Postures

We first design for accuracy without latency considerations. As described earlier, once the orientation of the watch is known, posture estimation boils down to an elbow tracking problem. If

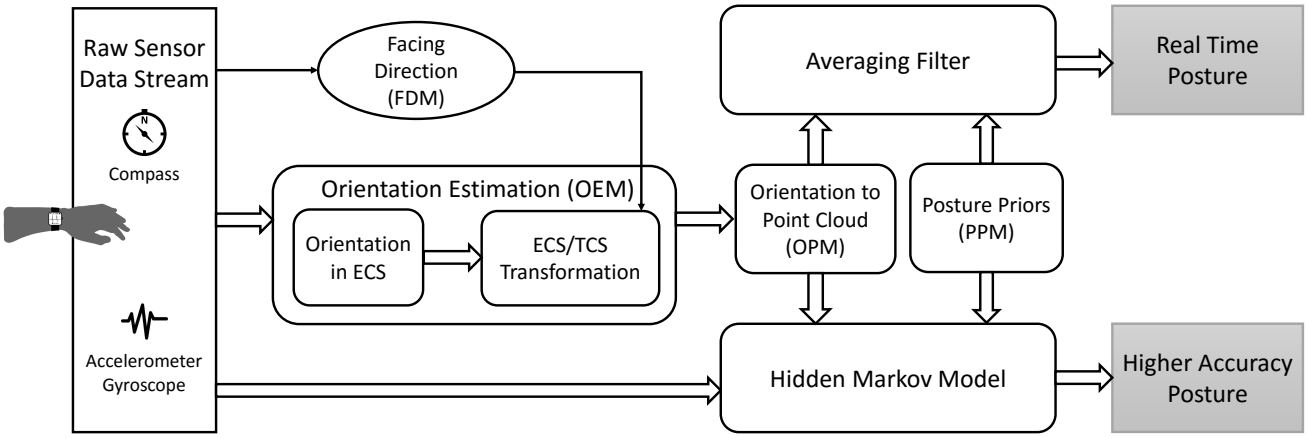

Figure 6: System Architecture. The raw sensor data is processed to obtain watch orientation and possible arm postures, and together with posture priors, they are sent to two different filters to obtain different posture estimations.

the elbow can be tracked, the wrist location can be computed as a static shift, yielding the complete arm posture. The resources we have (in addition to sensor data) are two-fold: (1) reasonable estimates of orientation, even though not precise, and (2) point cloud of all possible elbow locations, for a given orientation. The question then is: *at any given time, where is the elbow in the point cloud?*

Recall that the point cloud is often quite small – on average, it covers less than 10% of the sphere around the shoulder (Figure 5). Simply using the average location of the point cloud could result in a reasonably good estimate of the elbow location. However, in testing this simple averaging method with various kinds of gestures, we found much room for improvement. For instance, consider the punching gesture in Figure 7 – the forearm moves forward and backward while the orientation remains the same. The averaging scheme always shows the center of the point cloud (see Figure 8) since the point cloud remains almost the same for the entire gesture.

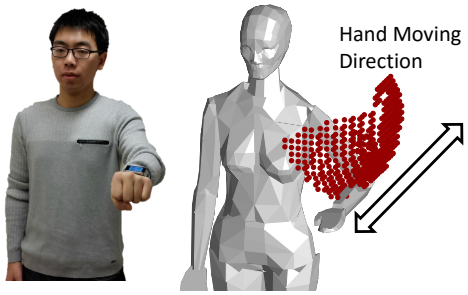

Figure 7: Punching: (a) video frame; (b) point cloud remains almost the same during punching.

The room for improvement (over simple averaging) arrives from using the smartwatch sensors as an estimate of the elbow's motion. In this specific punching case, the accelerometer data from the watch can be used to estimate the elbow's acceleration (in a straight line), which in turn can be converted to the wrist. In general, however, this is more complicated since the elbow will also experience rotational motion – in such cases, its acceleration has to be computed through a fusion of the gyroscope and accelerometer. To this end, we first introduce techniques to estimate acceleration, and then combine this elbow acceleration with the point cloud constraints to estimate elbow location.

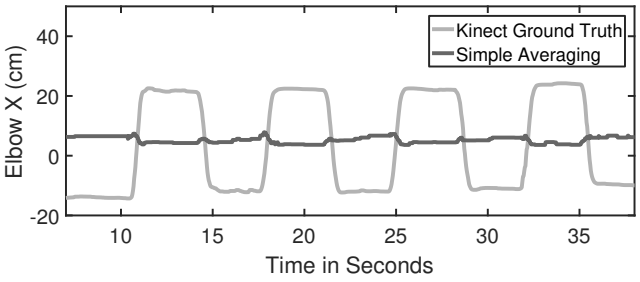

Figure 8: X coordinate of elbow location. During punching gesture, the orientation of the watch remains almost the same; therefore the point cloud doesn't change much over time. As a result, averaging the point cloud cannot follow the location of the elbow.

Estimate Elbow Acceleration

For any physical object, the acceleration is simply the second derivative of the location time series. For the elbow, we have

$$\text{accel}_\text{e}(t) = \frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{e}(t) \tag{8}$$

Applying Equation 5, we have:

$$\text{accel}_\text{e}(t) = \frac{\text{d}^2}{\text{d}t^2}\left[\text{loc}_\text{w}(t) - \text{loc}_\text{we}(t)\right] \tag{9}$$

$$= \frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{w}(t) - \frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{we}(t) \tag{10}$$

The first term in Equation 10, $\frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{w}(t)$, is simply wrist's acceleration in torso coordinate system, which we can get by projecting watch's accelerometer readings into torso coordinate system using estimated watch's orientation, Rot_watch:

$$\frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{w}(t) = \text{Rot}_\text{watch}(t)\ \text{accel}_\text{watch}(t) \tag{11}$$

The second term in Equation 10, $\frac{\text{d}^2}{\text{d}t^2}\text{loc}_\text{we}(t)$, is the acceleration caused by wrist's relative motion to the elbow. According to equa-

tion 7, we can express this as:

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathrm{loc}_{\mathrm{we}}(t) = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left[\mathrm{Rot}_{\mathrm{watch}}(t)\begin{pmatrix} l_f \\ 0 \\ 0 \end{pmatrix}\right] \tag{12}$$

Now, combining Equation 11 and Equation 12, we can re-write Equation 10 as

$$\begin{aligned}\mathrm{accel}_{\mathrm{e}}(t) = \mathrm{Rot}_{\mathrm{watch}}(t)\,\mathrm{accel}_{\mathrm{watch}}(t) \\ - \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left[\mathrm{Rot}_{\mathrm{watch}}(t)\begin{pmatrix} l_f \\ 0 \\ 0 \end{pmatrix}\right]\end{aligned} \tag{13}$$

Equation 13 shows that given watch's accelerometer data and our estimation on the watch's orientation, the elbow acceleration can be inferred.

Estimate Elbow Location

Given the measured elbow acceleration, as well as the point clouds (on the sphere) on which the elbow must be located, we now ask the following question: *which sequence of elbow locations best matches the measured elbow acceleration?* To intuitively understand this problem, let's assume there are N locations that the elbow can possibly reach – this can be viewed as the union of all point clouds for the elbow. Assume the motion sequence contains T time steps. Since the elbow can be at any of the N locations at a given step, the possible number of sequences is N^T, which is the search space for tracking the elbow. One of the sequences is optimal and our goal is to find this sequence. Hidden Markov Models (HMM) are well suited to solve this problem due to its dynamic programming construction for efficiently searching the state space.

Modified HMM for Elbow Tracking

If we group 3 locations (at 3 consecutive time steps) as the state of the elbow, then elbow acceleration can be encoded in one state. By using the estimated acceleration as observation and properly designing transition probabilities, finding the best elbow location sequence reduces to the Viterbi algorithms (solvable in polynomial time). Viterbi Decoding has a time complexity of $O(|S|^2 T)$, where $|S|$ is the state space size and T is the total number of time steps. In the above HMM formulation, the state space size is N^3, since each state is a location triple. As a result, the time complexity is $O(N^6 T)$. In trying to reduce the running time, we reorganized the state definitions. Specifically, we apply the continuity constraint, move the emission probability into the transition probability, and $|S|$ can be greatly reduced by allowing each state to contain only two locations. The time complexity is reduced to $O(N^3 T)$. The details are below.

For the ease of the description, let's assign each possible elbow location on the sphere a location ID, ranging from 1 to N. Let's also denote the T time steps of the motion sequence as $t_1, t_2, ..., t_T$, and the time step length as ΔT.

• **State definition**: Each state is defined as a pair of elbow locations:

$$\mathrm{state}_i = < \mathrm{loc_e}^{(i_1)}, \mathrm{loc_e}^{(i_2)} > \tag{14}$$

where $loc_e^{(i_1)}$ and $loc_e^{(i_2)}$ are the locations with location IDs i_1 and i_2, which together uniquely define the i-th state.

In our HMM formulation, we use a state to represent the elbow's previous location and current location. In this representation, if the state at time t_k is state_i, it means that the elbow location is $loc_e^{(i_1)}$ at t_{k-1} and $loc_e^{(i_2)}$ at t_k.

At first glance, the size of state space is N^2. However, observe that the human's elbow movement is limited to a maximum speed, thus given a small time step, the elbow can only move within a small range. We can actually eliminate those states whose location pairs are separated by larger than this range. In this way, the size of state space is reduced to αN^2, where α is much smaller than 1.

• **Prior probability**: Since we do not know the initial elbow location, we set the prior probability to be uniform:

$$\Pi(\mathrm{state}_i) = \frac{1}{\alpha N^2} \qquad \text{for any } i \tag{15}$$

• **Transition probability**: From current time t_k to next time t_{k+1}, the transition probability from state_i to state_j , denoted as $Pr(\mathrm{state}_j \mid \mathrm{state}_i; t_k, t_{k+1})$, contains three terms.

First, since the elbow trajectory is continuous, state_i and state_j must share the same location at the common time step t_k. This continuity constraint actually helps reduce the time complexity from $O(N^4 T)$ to $O(N^3 T)$.

$$\begin{aligned}\mathrm{state}_i &= < \mathrm{loc_e}^{(i_1)}, \mathrm{loc_e}^{(i_2)} > \\ \mathrm{state}_j &= < \mathrm{loc_e}^{(j_1)}, \mathrm{loc_e}^{(j_2)} > \\ \mathrm{loc_e}^{(i_2)} &= \mathrm{loc_e}^{(j_1)}\end{aligned} \tag{16}$$

We can express this limitation as an indicator function:

$$Pr_1 = I_{\mathrm{loc_e}^{(i_2)} = \mathrm{loc_e}^{(j_1)}} \tag{17}$$

Second, instead of using measured elbow acceleration as an observation in the location triple, here we directly model that probability into the transition probability between two location tuples. To be more specific, we can calculate the speed encoded in each state and derive acceleration using the speed of the two states:

$$\begin{aligned}\mathrm{velocity}_j &= \frac{\mathrm{loc_e}^{(j_2)} - \mathrm{loc_e}^{(j_1)}}{\Delta T} \\ \mathrm{velocity}_i &= \frac{\mathrm{loc_e}^{(i_2)} - \mathrm{loc_e}^{(i_1)}}{\Delta T} \\ \mathrm{accel}_{i,j} &= \frac{\mathrm{velocity}_j - \mathrm{velocity}_i}{\Delta T}\end{aligned} \tag{18}$$

This acceleration, $\mathrm{accel}_{i,j}$, is expected to be close to our observed acceleration $\mathrm{accel}_{\mathrm{observe}}(t_k)$. We assume that the error distribution of the observed acceleration is a zero mean Gaussian distribution with a standard deviation of σ_{accel}. Therefore, we have:

$$Pr_2 = \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{accel}}} e^{(\mathrm{accel}_{i,j} - \mathrm{accel}_{\mathrm{observe}}(t_k))^2/(2\sigma^2_{\mathrm{accel}})} \tag{19}$$

Third, in the new state_j, the elbow location $\mathrm{loc_e}^{(j_2)}$ must be inside the point cloud inferred at time t_{k+1}.

$$Pr_3 = I_{\mathrm{loc_e}^{(j_2)} \in \mathrm{PointCloud}_{t_{k+1}}} \tag{20}$$

In sum, the transition probability is the product of these three probabilities:

$$Pr(\mathrm{state}_j \mid \mathrm{state}_i; t_k, t_{k+1}) = Pr_1 Pr_2 Pr_3 \tag{21}$$

• **Emission probability**: Since we have already integrated the observed acceleration into transition probability, emission probability is simply set as 1. Thus, we only use the output of Viterbi decoding – it is a sequence of states and the second element of each state is the estimated elbow location. Figure 9 shows the improved elbow tracking results (in this toy punching case) using HMM.

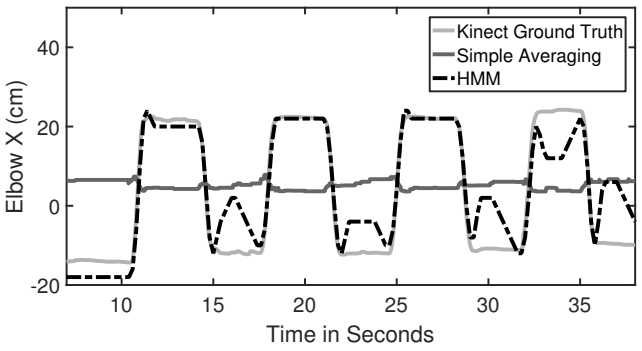

Figure 9: X coordinate of elbow location. Results from HMM better capture the location of the elbow.

• **Facing Direction**: We opportunistically sense the facing direction when the user's hand is in the vertical free-fall posture, or swinging through this position perhaps while walking. At this point, the X axis of the gyroscope is exactly pointed in gravity's direction, implying that the negative Y is the facing direction. We also observe that the hand swings to a small degree in this position and we utilize this to gain confidence.

4.2 Designing for Fast Posture Tracking

For the applications on smartwatches that require instantaneous posture information, the tracking algorithm must be light-weight in order to fit into the watch's limited computing power. As hinted earlier, we adopt a simple weighted averaging filter. This filter accepts (1) the point cloud of arm postures and (2) the corresponding prior from past measurements, and then outputs a weighted average of all the points in the cloud. The weights are determined by the probability density of each location – the more common postures will naturally bias the estimates. The weighted average is expressed as:

$$\text{loc}_\text{e} = \sum_i \text{loc}_\text{e}^\text{i} \frac{Pr(\text{loc}_\text{e}^\text{i})}{\sum Pr(\text{loc}_\text{e}^\text{i})} \tag{22}$$

$$\text{loc}_\text{w} = \text{loc}_\text{e} + \text{Rot}_\text{watch}(t) \begin{pmatrix} l_f \\ 0 \\ 0 \end{pmatrix} \tag{23}$$

As an outcome of averaging, the results are naturally quite smooth. Also, the averaged 3D location (for each point cloud) can be stored in a lookup table, indexed by the orientation corresponding to the point cloud. For a given orientation from the *Orientation Estimation Module*, the smartwatch simply looks up the elbow location from this table, computes the corresponding wrist location, and outputs the posture. Both memory and CPU footprint is marginal.

5. EVALUATION

This section discusses the experiment methodology and performance results of *ArmTrak*.

5.1 Implementation

ArmTrak is implemented on the Samsung Gear Live smartwatch using JAVA as the programming platform. The accelerometer and gyroscope are both sampled at $200Hz$ and the magnetic field sensor is sampled at $100Hz$. The smartwatch runs the light-weight real-time version of *ArmTrak* to report instantaneous arm posture. The sensor data and light-weight arm posture estimates are stored locally and transferred to the *ArmTrak* server for analysis. The server side code is written in MATLAB and implements the full version of *ArmTrak* to provide offline, higher accuracy, posture estimates.

5.2 Methodology

We recruited 8 volunteers, including 6 males and 2 females, for our experimentation – the volunteers are all students in CS/ECE. The volunteers were asked to wear a Gear Live smartwatch during our experiment. Their upper arm and forearm lengths, l_u and l_f, were measured beforehand.

Experiments with each volunteer were executed in 3 sessions. In the first session, volunteers were asked to move their arms totally freely for 3 minutes. Users performed random, meaningless, arm gestures – we requested them to not move their hands behind their backs to avoid losing ground truth from the Kinect. In the second session, we deliberately asked the volunteers not to put their elbows above the shoulder. The goal is to mimic real-world scenarios where most of the gestures need the elbow to be at lower heights. Under this constraint, the volunteers again moved their arms freely for 3 minutes. In the third session, volunteers were asked to repeat a set of pre-defined gestures for 10 times. The gesture set contains eating, drinking, boxing, bouncing a basketball, weight lifting, drawing a circle, drawing a triangle, drawing a square, and writing numeric digits in the air. During the whole experiment, a Kinect 2.0 was placed in front the volunteer to record ground truth. We prevent any movement in the background since that affects Kinect's ground truth calculations.

5.3 Performance Results

The following questions are of our interest in this section:

1. How well can *ArmTrak* track arm postures in general?
2. How does *ArmTrak*'s performance vary among different users and with pre-defined gestures? Are certain gestures better than others?
3. Will error accumulate and *ArmTrak*'s tracking diverge over time?
4. Accuracy and latency tradeoffs with the real-time version and the offline cloud version.
5. 2D shapes of different objects and digits drawn by users – a subjective measure.

In all these results, we measure error for every time step (i.e., output of HMM) and draw the CDF over all measurements.

(1) How well can *ArmTrak* track arm postures?

Figure 10 (a) and (b) show the CDF of tracking errors for the elbow and wrist, respectively. The results are for the higher accuracy HMM version. For free-form motion (where users performed completely random gestures), the median errors for the elbow and wrist are around 7.9cm and 9.2cm. Once the elbow

was restricted to remain below the shoulder, the error reduced further to 6.6cm and 8.3cm for the elbow and wrist, respectively. Finally, for pre-defined gestures like "eating", "weight lifting", etc., we use the ground truth information from the 7 other users as priors for the 8^{th} user, and perform cross-validation. Observe that the median error drops even further to 4.5cm and 5.7cm for elbow and wrist on average. Compared to the volunteers' average arm length of 50.2cm, we believe *ArmTrak*'s accuracy, with minimal prior information, makes it amenable to most gesture recognition applications, including gaming control, TV control, and daily activity patterns such as eating or exercising. Of course, the results can improve appreciably with more application-specific prior information.

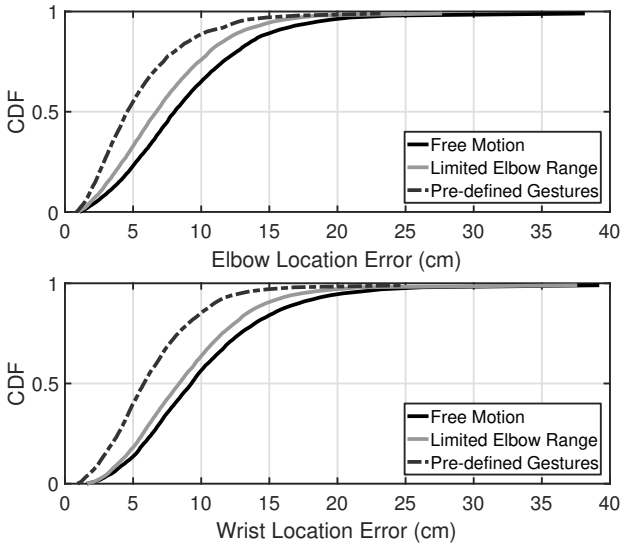

Figure 10: (a) *ArmTrak*'s performance on elbow tracking. (b) *ArmTrak*'s performance on wrist tracking.

(2) How does *ArmTrak*'s performance vary across different users and pre-defined gestures?

Figure 11 plots the performance across different users. We observe that performance is consistently high across all users, across all the 3 categories – pre-defined gestures, limited elbow range, and free motion. Also, the trend that pre-defined gestures are generally better than limited elbow range cases, and limited elbow range better than free-motion cases, holds across the users. The performance is worst for user 7. On examining the Kinect video, we find that user 7 moved that hand extremely fast. Still, the errors were around 10cm.

Figure 12 plots *ArmTrak*'s performance across all types of 8 gestures. Evidently, the performance did not show major variations across gestures, suggesting that the system is not biased to any patterns. Considering the fact that the prior is only obtained from 7 other users, we gained confidence that by improving the prior, *ArmTrak* can be generalized to and work well on other pre-defined gestures.

Upon comparing the performance among these gestures, we find that "Eating" has the best performance and "Drawing a triangle" is the worst. We again look into the Kinect video data and find that when volunteers were performing "Eating" gestures, their elbow only moved in a smaller region, while with "Drawing a triangle", the shape, size and position where they drew the triangle are all different, incurring a far greater elbow range.

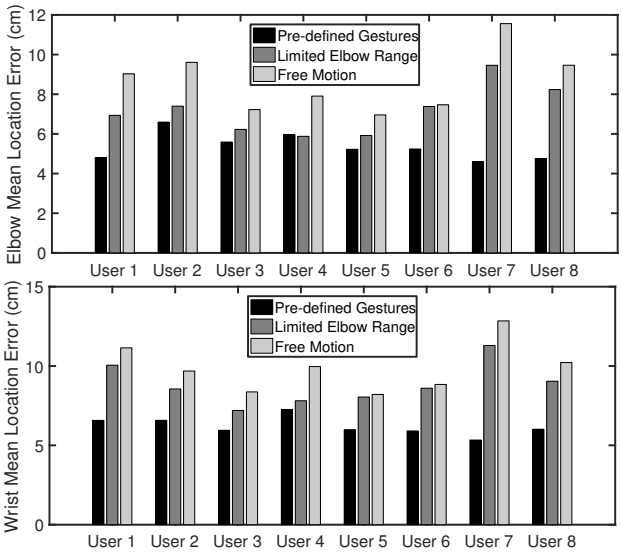

Figure 11: *ArmTrak*'s performance on (a) elbow and (b) wrist tracking, for different users.

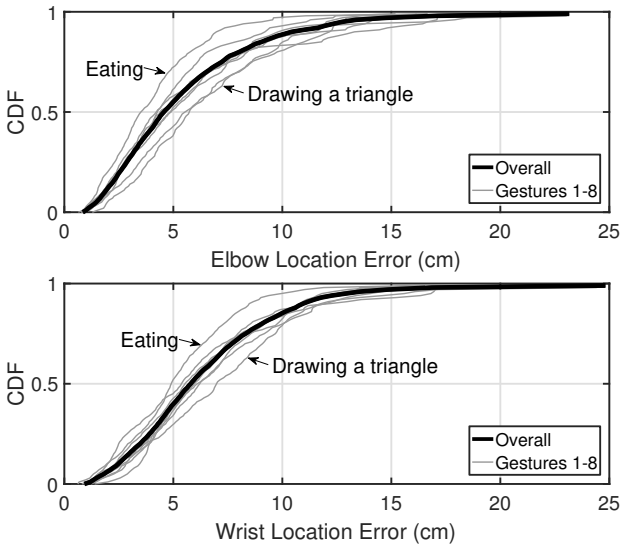

Figure 12: *ArmTrak*'s performance on (a) elbow and (b) wrist tracking, for different activities.

(3) Will error accumulate and *ArmTrak*'s performance degrade over time?

ArmTrak attempts to find a sequence of smartwatch locations (from the changing point clouds) that best matches with the observed acceleration data. This global optimization within point clouds ensures that the error will not accumulate over time, as it does with unconstrained double integration. One may argue that this optimization is performed offline over the whole motion sequence, thus intermediate states also benefit from future data. Therefore, characterizing the errors at every intermediate state, with no look-ahead into the future, is also of interest.

To understand the impact, we performed another experiment in

which we ask the HMM to traceback to each timestamp and report the instantaneous location estimate at that point. Figure 13 shows the general wrist tracking error trend for 3 volunteers. Although this error is higher than applying the global Viterbi Decoding over the whole sequence, the error still does not accumulate. We believe this is perhaps the most important property of *ArmTrak*.

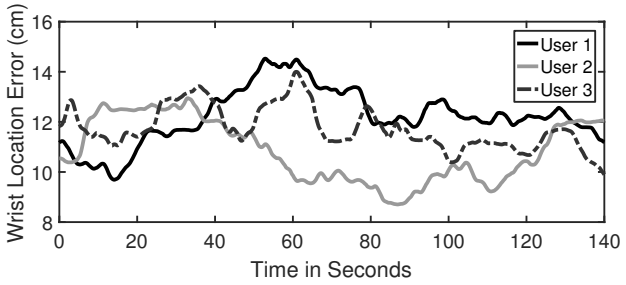

Figure 13: The general (smoothed) trend for wrist location error over time.

(4) Performance and latency tradeoffs between real-time version and offline version

Figure 14 shows the comparison of tracking accuracy for both elbow and wrist, using real-time and offline algorithms. Recall that the real-time algorithm computes a weighted average of the point cloud and stores a lookup table in the watch, indexed by 3D watch orientation. Compared with the offline algorithm, the median error of the real-time version increases to 12.0cm for the elbow and 13.3 cm for the wrist. Although high, the real-time version also remains stable over time. The reader is requested to visit the project website – http://synrg.csl.illinois.edu/posture/ – for a number of real-time and offline video demonstrations.

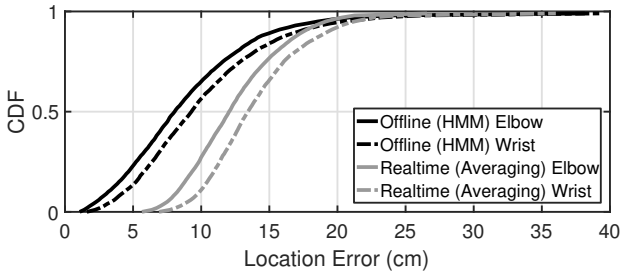

Figure 14: Performance comparison between offline (HMM) and real-time(simple averaging).

We computed the delay of both offline and real-time versions of *ArmTrak*. The real-time version running on the watch is essentially a look-up with orientation as index – even for very high update rate the lookup time is negligible. The delay of offline-version contains both the network upload/download delay of the sensor data, plus the computation time at the cloud (i.e., running MATLAB on a quad core graduate student desktop). The results are reported in Table 2 for 5Hz update frequency (i.e., HMM updates 5 times per second). Evidently, longer gesture data incurs almost a 10x increase in computation.

(5) What shapes have been inferred by *ArmTrak*?

Figure 15 shows some sample trajectories of the wrist, when users were asked to draw shapes and digits in the air. Although

Trajectory Time	10s	30s	1min	3min
Delay	98.2s	289.3s	9.1min	26.9min

Table 2: Latency of offline version increases with increase in the trace length. This is because Viterbi Decoding computes the globally optimal sequence of states, and incurs $O(N^3T)$ complexity.

the reproduction is not perfect, *ArmTrak* tracks the trend of the trajectory quite well. All users expressed satisfaction when they were shown the shapes that they drew in the air. For more complex situations, we asked the user to draw complicated shapes like a "star" or an "Olympic ring". Figure 15(c) shows one such case where the user drew for almost 1 minute – *ArmTrak* was consistently able to track the 3D shape.

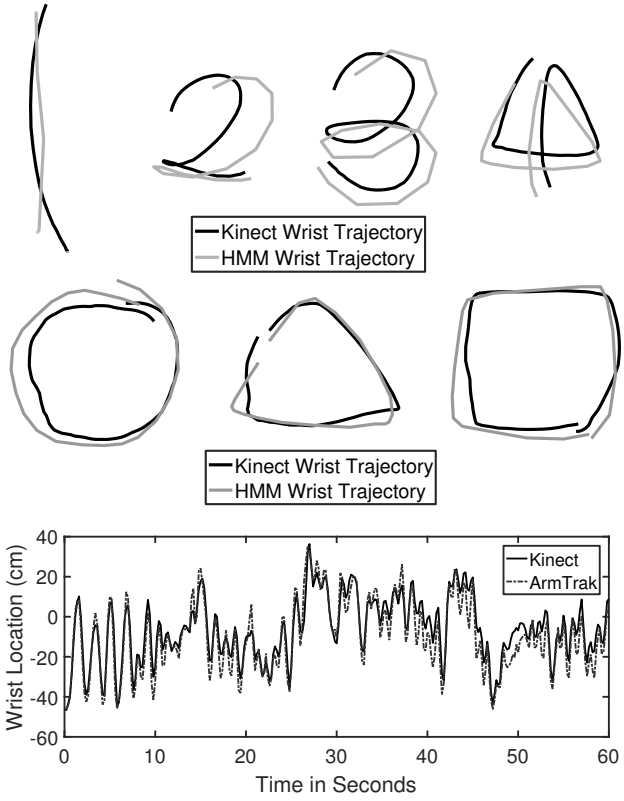

Figure 15: *ArmTrak*'s tracking result for (a) writing four digits, (b) drawing simple shapes, (c) a complicated 3D trajectory (only Z axis is shown).

6. LIMITATIONS AND NEXT STEPS

A few more technical pieces will need to come together before *ArmTrak* can be viewed as a usable technology – we discuss these below.

(1) Facing Direction and Tracking on the Move: We opportunistically estimate the user's facing direction when her hand is in the vertical free-fall posture, or swinging through this position perhaps while walking. Admittedly, we have not stress-tested this and are not confident this would scale in completely uncontrolled situations. For instance, if a user is sitting for a long duration, the free-fall opportunity may not arise, but the user may change her facing direction (perhaps by swiveling her chair). A deeper treatment of facing direction is necessary to

better ground the initial state of the watch. On related lines, *ArmTrak* will falter when the user is on the move – we have sidestepped this case, but plan to evaluate the extent of degradation in future.

(2) Need for More Speed: The Viterbi Decoding is running on a quad-core student desktop and is roughly producing results at 10x rate. This means that tracking the arm motion for τ seconds requires 10τ seconds of processing time. Of course, more hardware on a cloud can certainly bring down this latency, but the more important question pertains to whether more speedup is possible. We believe some degree of optimization (such as beam search) and parallelism would be possible inside the dynamic programming; we also believe a marginal sacrifice in accuracy can offer considerable speedup. The latency-accuracy tradeoff proved far richer than we anticipated and we intend to investigate this thoroughly in future.

(3) Energy Consumption: We have ignored the energy implications of our technique. For the real time system, we expect the weighted averaging technique to impose minimal energy burden. For the HMM/Viterbi model, we expect the system to run on the cloud – thus the energy consumption mainly emerges from network uploads. We have not characterized this overhead, however, it appears that many offloading applications are viable under this model. This is perhaps because most applications are likely to be on-demand (e.g., the user turns on the app during her visit to the gym and turns off thereafter). Further, some apps can tolerate latency and can delay the uploads until the watch is connected to power.

7. RELATED WORK

Gesture/posture recognition has been studied from various perspectives. The literature is vast, but we sample the most relevant ones, mainly from computer vision and wearable motion sensors.

Computer vision: Camera data has been used to track and analyze human motion across different granularities [19]. At a lower granularity, humans can be automatically detected [10] and tracked with bounding boxes [34], using the video feeds from cameras. Beyond bounding boxes, human activity can also be recognized from camera data via machine learning [26, 27]. At a higher granularity, pose estimation is a classical problem in human motion analysis, where the common approach is applying probabilistic models on the static RGB image or video sequence [6, 13, 15–17, 20, 23, 28, 29, 31, 36]. More recently, depth information has also been leveraged in the pose estimation solution landscape [3, 30]. For instance, Microsoft Kinect [3] fuses RGB and depth image to track the locations of the human's joints for video gaming. One of the key differentiators between vision and sensing based approaches is that vision must estimate 3D motion from the 2D views – a challenging task. However, vision benefits from knowing the pixel locations far more precisely compared to the noise from sensor hardware.

Wearable motion sensors: Previous research has shown that embedded motion sensors on wearable devices can be used for human activity recognition [18]. Industry on mobile health and well being uses these sensors to recognize a user's leg motion such as walking, running, etc [1, 2]. To measure meal intake, Bite Counter [11] uses a watch-like device with a gyroscope to detect and record when an individual has taken a bite of food. RisQ [21] leverages motion sensors on wristband to recognize smoking gestures. MoLe [33] analyzes motion data of smartwatches from typing activity to infer what the user has typed. Xu et al. [35] classified hand/finger gestures and written characters from smartwatch motion sensor data. However, all these motion analysis systems are designed for recognizing specific pre-defined motion patterns, as opposed to blind estimation of free-form postures. Similarly, authors in [25, 32] tried to reconstruct full body motion from multiple wearable devices by comparing accelerometer data with those generated from motion capture databases. However, the reconstruction relies heavily on the similarity of training and testing accelerometer data and the disparity between different motion classes inside training databases, and as a result, neither can they track free-form arm motion.

Zhou et al. [38, 39], Cutti et al. [8], and El-Gohary et al. [12] studied general upper limb movement tracking using motion sensors. However, they require users to be instrumented with multiple sensors on the arm. Perhaps the closest to our work is [37], where authors claimed to be able to track the upper limb by only mounting motion sensors on the wrist. However, the system is only evaluated on 1 subject, moving his arm up-to-down in a plane perpendicular to the ground. The same gesture is repeated constantly and lasted less than 15s. As a follow up to this work, the authors published a subsequent paper with multiple sensors on the arm to scale to a larger vocabulary of gestures [38, 39]. In contrast, our system is tested with free-form motion, has been tested up to 3mins without signs of divergence, and has demonstrated robustness to all 8 test users. We believe this is an improvement over the state of the art.

8. CONCLUSION

This paper is an attempt to estimate/track the geometric motion of the human arm, using only the inertial sensors on the smartwatch. The problem is challenging primarily because the smartwatch is a single point of measurement on this otherwise large space of possibilities. Moreover, the measurements are noisy, making continuous tracking over longer time scales even more difficult. We develop *ArmTrak*, a system that distills observations from human kinematics, and uses them carefully inside a (modified) HMM framework. Our results are encouraging, and with some more effort, could become a useful underlay to a broad class of gesture-based applications.

9. ACKNOWLEDGMENTS

We sincerely thank our shepherd, Dr. Vishnu Navda, and the anonymous reviewers for their insightful comments and suggestions. We are also grateful to Intel, Google, Qualcomm, HP and NSF (grant NSF 1430064) for partially funding this research.

10. REFERENCES

[1] Apple watch. http://www.apple.com/watch/.
[2] Fitbit. https://www.fitbit.com/.
[3] Microsoft kinect. https://dev.windows.com/en-us/kinect.
[4] Rithmio. http://rithmio.com/.
[5] Xsens. https://www.xsens.com/.
[6] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.
[7] D. C. Boone and S. P. Azen. Normal range of motion of joints in male subjects. *J Bone Joint Surg Am*, 61(5):756–759, 1979.

[8] A. G. Cutti, A. Giovanardi, L. Rocchi, A. Davalli, and R. Sacchetti. Ambulatory measurement of shoulder and elbow kinematics through inertial and magnetic sensors. *Medical & biological engineering & computing*, 46(2):169–178, 2008.

[9] J. Denavit. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME. Journal of Applied Mechanics*, 22:215–221, 1955.

[10] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[11] Y. Dong, A. Hoover, J. Scisco, and E. Muth. A new method for measuring meal intake in humans via automated wrist motion tracking. *Applied psychophysiology and biofeedback*, 37(3):205–215, 2012.

[12] M. El-Gohary and J. McNames. Shoulder and elbow joint angle tracking with inertial sensors. *Biomedical Engineering, IEEE Transactions on*, 59(9):2635–2641, 2012.

[13] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[14] R. L. Gajdosik and R. W. Bohannon. Clinical measurement of range of motion review of goniometry emphasizing reliability and validity. *Physical Therapy*, 67(12):1867–1872, 1987.

[15] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision computing*, 1(1):5–20, 1983.

[16] M. P. Kumar, A. Zisserman, and P. H. Torr. Efficient discriminative learning of parts-based models. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 552–559. IEEE, 2009.

[17] X. Lan and D. P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 470–477. IEEE, 2005.

[18] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *Communications Surveys & Tutorials, IEEE*, 15(3):1192–1209, 2013.

[19] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[20] J. O'Rourke and N. I. Badler. Model-based image analysis of human motion using constraint propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):522–536, 1980.

[21] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 149–161. ACM, 2014.

[22] J. C. Perry and J. Rosen. Design of a 7 degree-of-freedom upper-limb powered exoskeleton. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 805–810. IEEE, 2006.

[23] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 206–213. IEEE, 2006.

[24] N. B. Reese and W. D. Bandy. *Joint range of motion and muscle length testing*. Elsevier Health Sciences, 2013.

[25] Q. Riaz, G. Tao, B. Krüger, and A. Weber. Motion reconstruction using very few accelerometers and ground contacts. *Graphical Models*, 79:23–38, 2015.

[26] P. C. Ribeiro and J. Santos-Victor. Human activity recognition from video: modeling, feature selection and classification architecture. In *Proceedings of International Workshop on Human Activity Recognition and Modelling*, pages 61–78. Citeseer, 2005.

[27] N. Robertson and I. Reid. A general method for human activity recognition in video. *Computer Vision and Image Understanding*, 104(2):232–248, 2006.

[28] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image understanding*, 59(1):94–115, 1994.

[29] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. *Computer Vision–ECCV 2010*, pages 406–420, 2010.

[30] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[31] V. K. Singh, R. Nevatia, and C. Huang. Efficient inference with multiple heterogeneous part detectors for human pose estimation. In *Computer Vision–ECCV 2010*, pages 314–327. Springer, 2010.

[32] J. Tautges, A. Zinke, B. Krüger, J. Baumann, A. Weber, T. Helten, M. Müller, H.-P. Seidel, and B. Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (TOG)*, 30(3):18, 2011.

[33] H. Wang, T. T.-T. Lai, and R. Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 155–166. ACM, 2015.

[34] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern recognition*, 36(3):585–601, 2003.

[35] C. Xu, P. H. Pathak, and P. Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 9–14. ACM, 2015.

[36] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011.

[37] H. Zhou and H. Hu. Inertial motion tracking of human arm movements in stroke rehabilitation. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1306–1311. IEEE, 2005.

[38] H. Zhou and H. Hu. Upper limb motion estimation from inertial measurements. *International Journal of Information Technology*, 13(1):1–14, 2007.

[39] H. Zhou, H. Hu, and Y. Tao. Inertial measurements of upper limb motion. *Medical and Biological Engineering and Computing*, 44(6):479–487, 2006.

[40] P. Zhou, M. Li, and G. Shen. Use it free: Instantly knowing your phone attitude. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 605–616. ACM, 2014.