Location, Gesture and Activity Sensing

Part II: Odometry to Motion Tracking

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

What We Have Learnt So Far?

- IMU: Accelerometer, Gyroscope, Magnetometer
- The IMU accelerometers measure true acceleration
  - A static accelerometer placed on Earth's surface will read the gravitational acceleration in its vertical axis
- To measure the coordinate acceleration, we need to subtract gravitational offset
  - Need to compute the roll, pitch and yaw angles when the accelerometer is in any arbitrary orientation
- Using only accelerometer data, we can compute two of the three angles
  - Need gyroscope to measure the yaw angle
- Combine the readings from all the three sensors to estimate the true orientation of an object

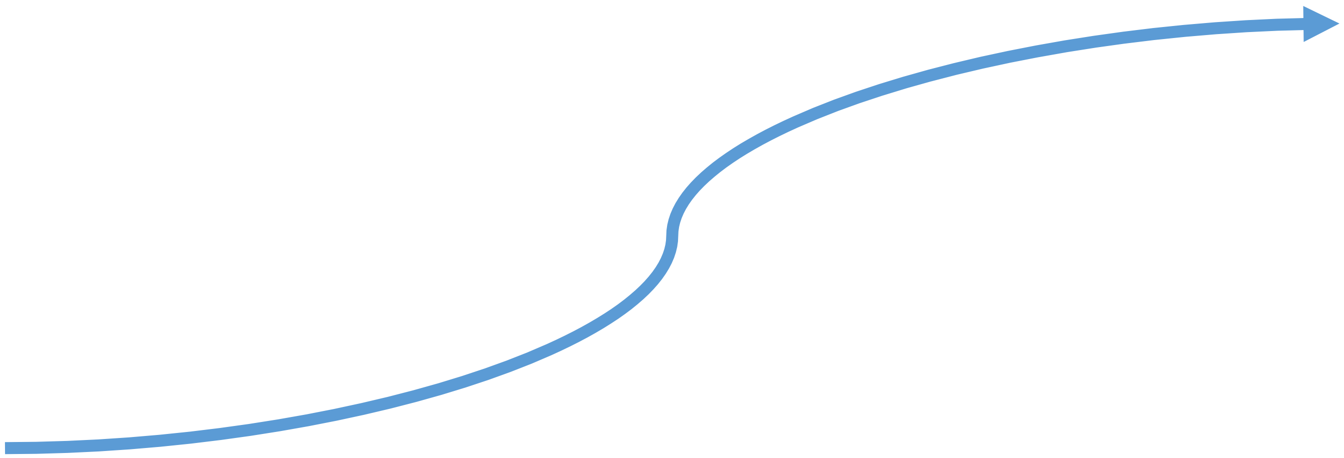
Our Next Problem

- Let we calculate the orientation of an object using the IMU measures and the Euler angles method, and then subtract the gravitational offset computed over the three axes to obtain the coordinate acceleration of an object
- Can we use this instantaneous values of the coordinate acceleration to estimate the movement trajectory of the object?

Animation Source: https://medium.com/@rocchokcoco/motion-path-magical-path-animation-6f9f36c621b3

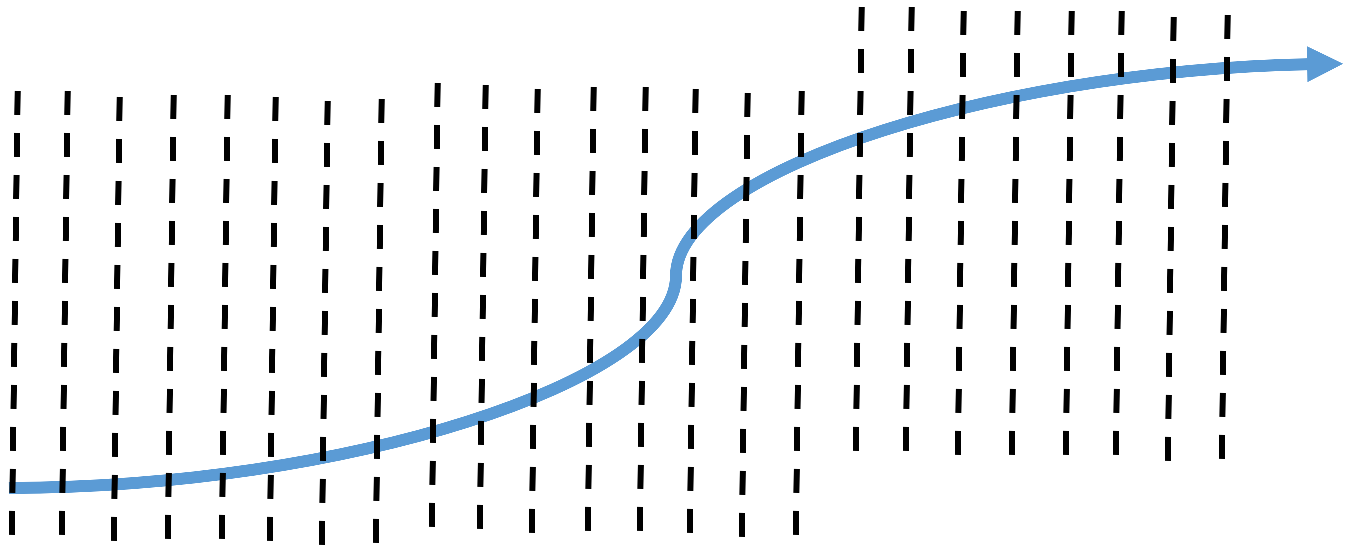
The Naïve Approach

- Let a_t be the measured acceleration at time t; $t \in [t_1, t_2, t_3, \ldots, t_n]$
  - Let we sample at small time gaps $\Delta t = t_2 - t_1 = t_3 - t_2 = \ldots = t_n - t_{n-1}$
- Use numerical integration over a_t to measure the velocity v_t at each time instances $t_2, t_3, \ldots, t_n$
- Finally, use numerical integration over v_t to measure the displacement x_t at each of the time instances $t_2, t_3, \ldots, t_n$



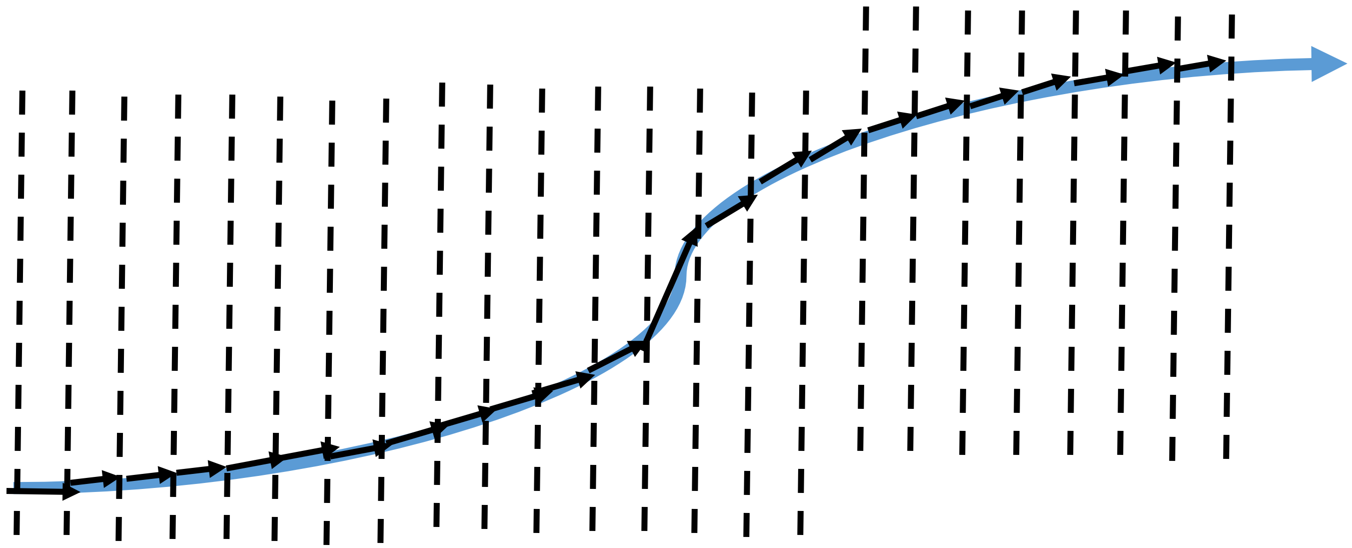
The Naïve Approach

- Let a_t be the measured acceleration at time t; $t \in [t_1, t_2, t_3, \ldots, t_n]$
  - Let we sample at small time gaps $\Delta t = t_2 - t_1 = t_3 - t_2 = \ldots = t_n - t_{n-1}$
- Use numerical integration over a_t to measure the velocity v_t at each time instances $t_2, t_3, \ldots, t_n$
- Finally, use numerical integration over v_t to measure the displacement x_t at each of the time instances $t_2, t_3, \ldots, t_n$



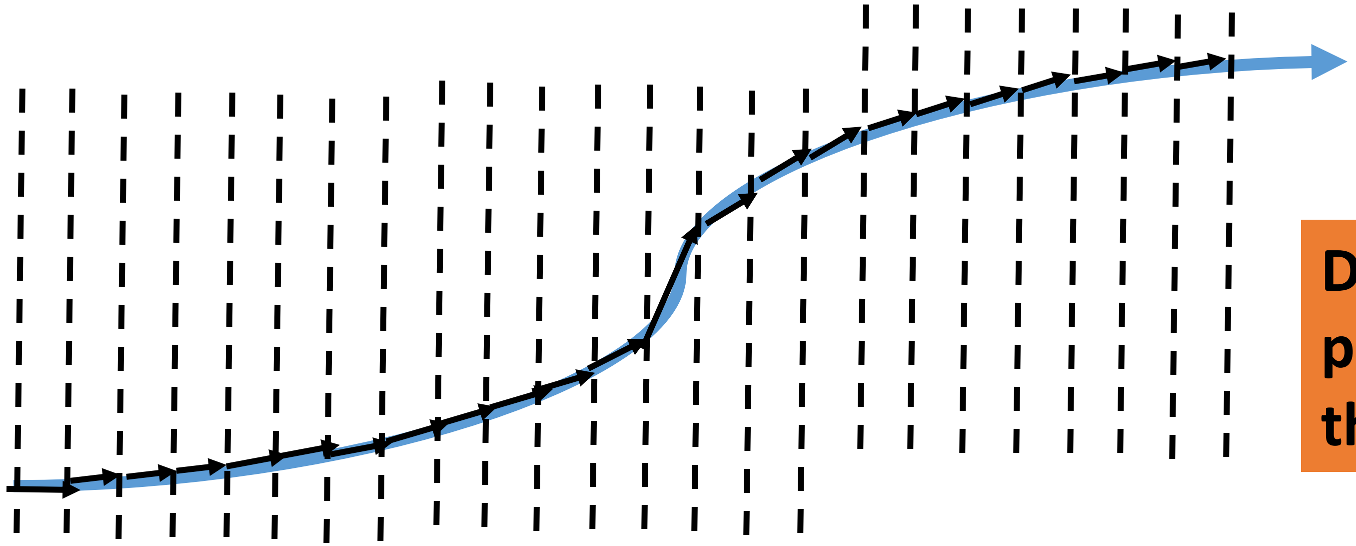
The Naïve Approach

- Let a_t be the measured acceleration at time t; $t \in [t_1, t_2, t_3, \ldots, t_n]$
  - Let we sample at small time gaps $\Delta t = t_2 - t_1 = t_3 - t_2 = \ldots = t_n - t_{n-1}$
- Use numerical integration over a_t to measure the velocity v_t at each time instances $t_2, t_3, \ldots, t_n$
- Finally, use numerical integration over v_t to measure the displacement x_t at each of the time instances $t_2, t_3, \ldots, t_n$



The Naïve Approach

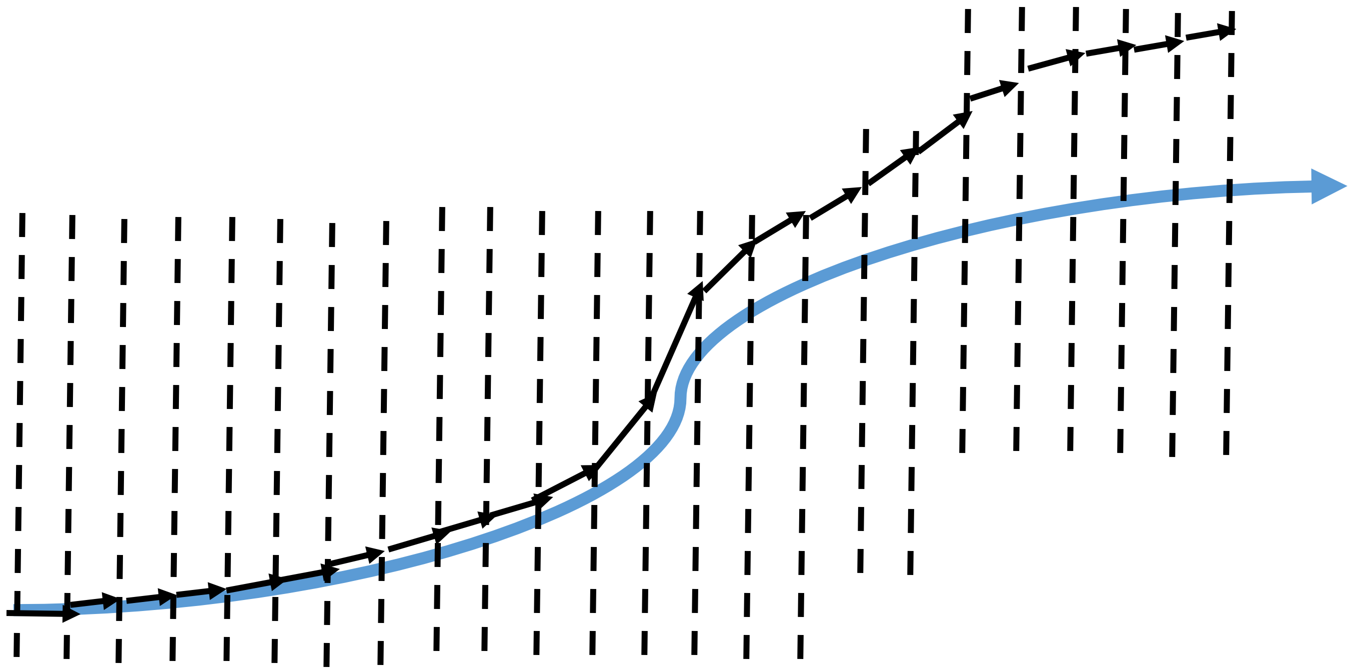
- Let a_t be the measured acceleration at time t; $t \in [t_1, t_2, t_3, \ldots, t_n]$
  - Let we sample at small time gaps $\Delta t = t_2 - t_1 = t_3 - t_2 = \ldots = t_n - t_{n-1}$
- Use numerical integration over a_t to measure the velocity v_t at each time instances $t_2, t_3, \ldots, t_n$
- Finally, use numerical integration over v_t to measure the displacement x_t at each of the time instances $t_2, t_3, \ldots, t_n$



Do you see any potential issue with this approach?

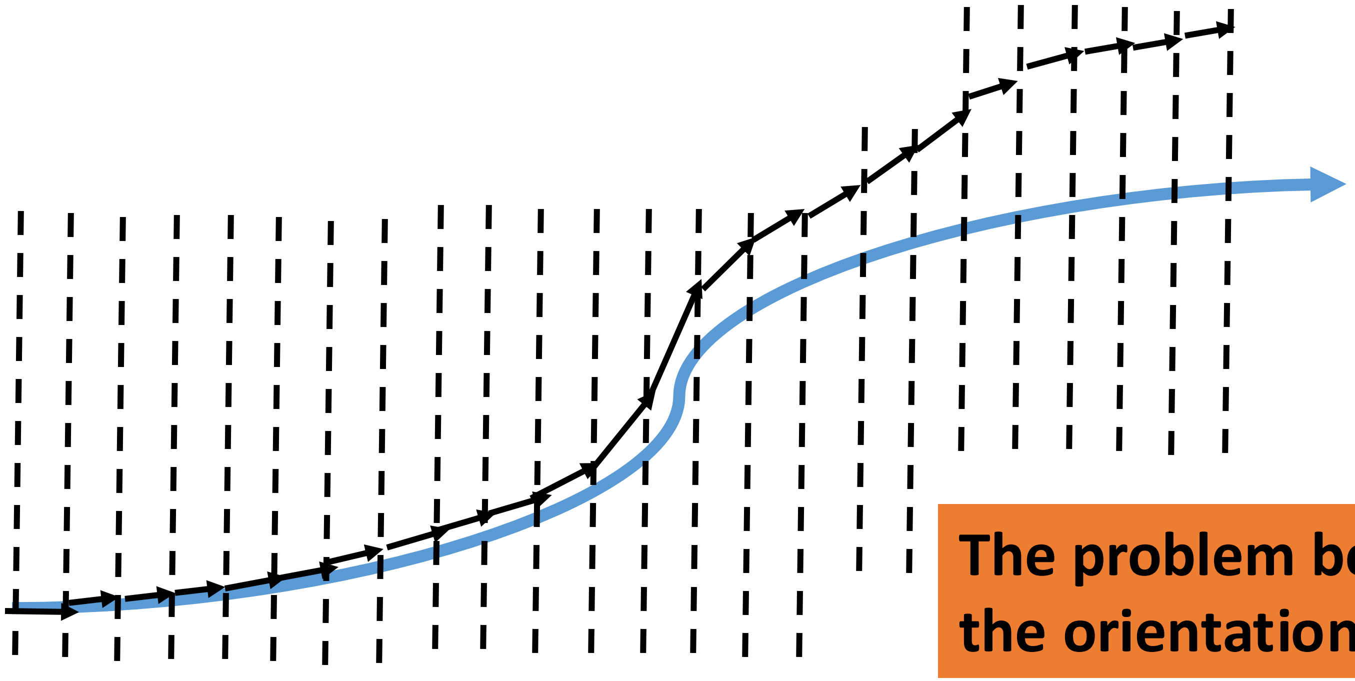
Fallacy in Trajectory Estimation

- The double integration induces significant errors in the estimation, the error will keep on added up when you add the respective displacement values



Fallacy in Trajectory Estimation

- The double integration induces significant errors in the estimation, the error will keep on added up when you add the respective displacement values



The problem becomes much severe if the orientation keeps on changing

How Can You Reduce the Error?

- Can you think of a parameter that will reduce the integration error?

How Can You Reduce the Error?

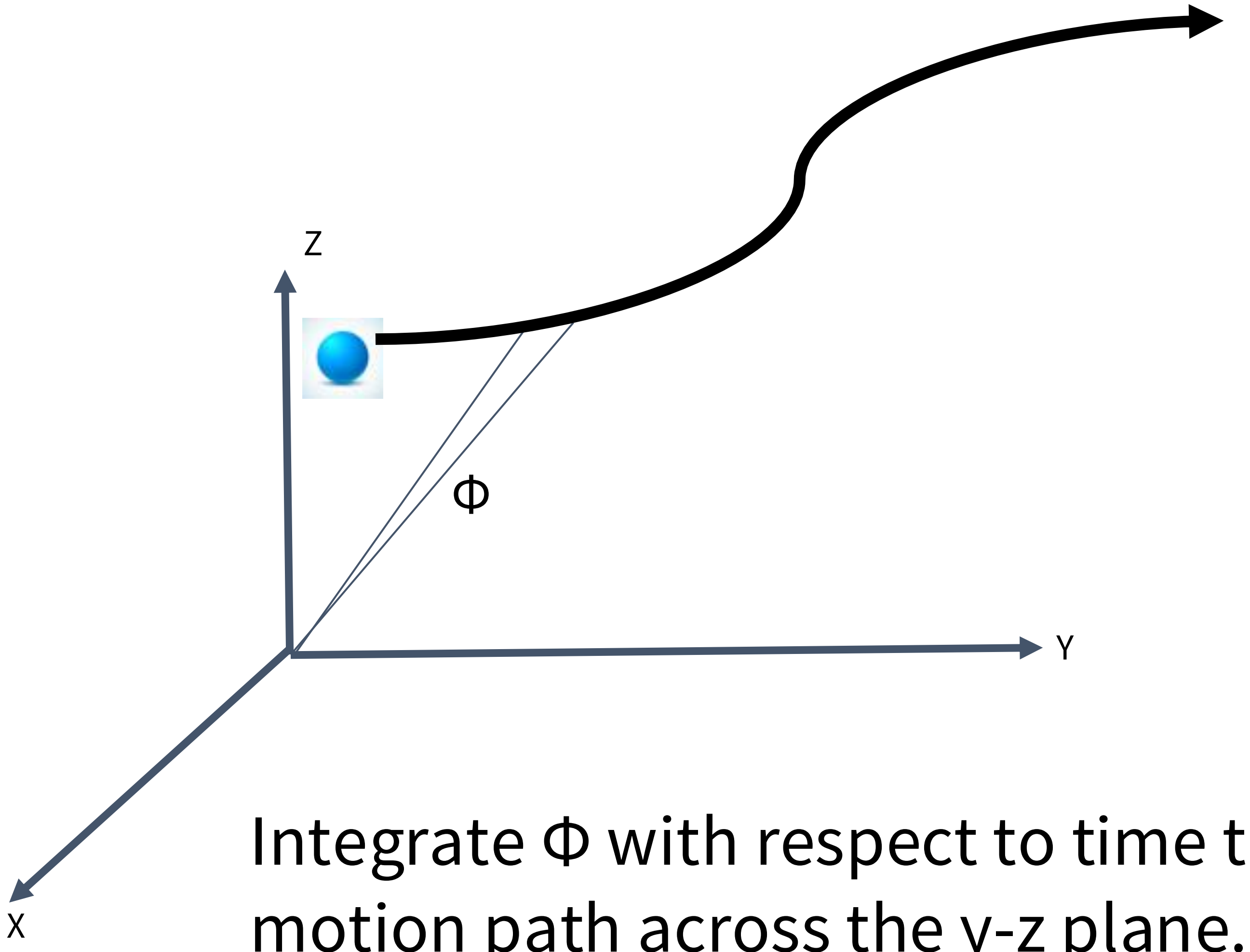
- Can you think of a parameter that will reduce the integration error?
  - Angular velocity: We get it from gyroscope
  - We can use the Euler angle method to subtract the gravitational offset
  - A single integration over the coordinate angular velocity will give angular displacement
  - But how do we convert angular displacement to linear displacement?

How Can You Reduce the Error?

- Can you think of a parameter that will reduce the integration error?
  - Angular velocity: We get it from gyroscope
  - We can use the Euler angle method to subtract the gravitational offset
  - A single integration over the coordinate angular velocity will give angular displacement
  - But how do we convert angular displacement to linear displacement?

- **Our conjecture:** At infinitesimally small time-scale, the angular displacement with respect to one axes gives the linear displacement with respect to the plane perpendicular to it (constructed by the other two axes)

Angular Displacement to Linear Displacement



Integrate Φ with respect to time to compute the motion path across the y-z plane.

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

ACM MobiSys 2016

Let's See a Video First ...

Basic Insights

- Gyroscope is good in capturing the orientation of the arm, but accelerometer fails to capture the arm location (as we have seen earlier …)

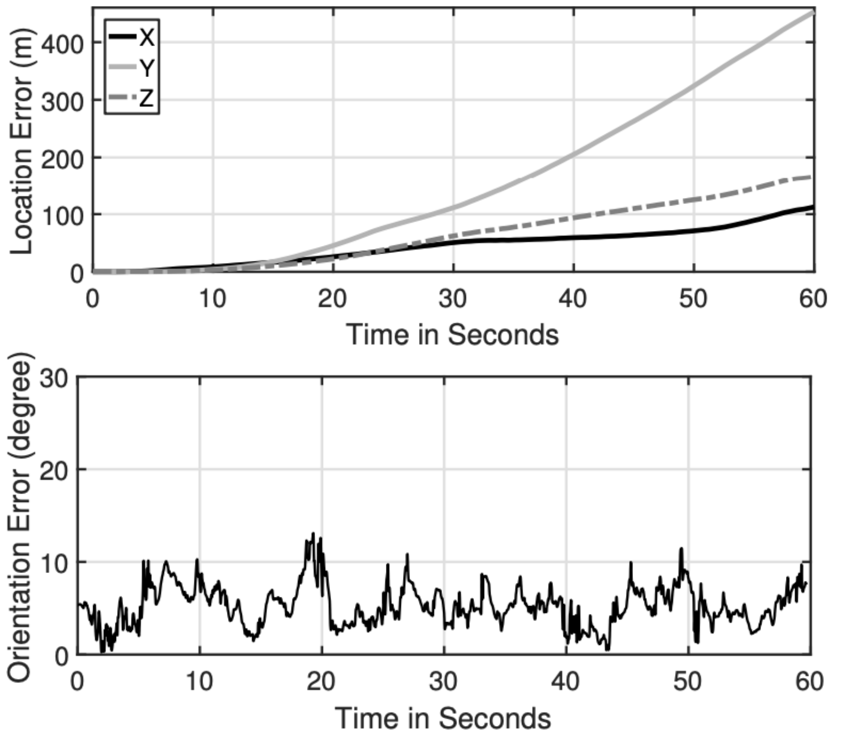
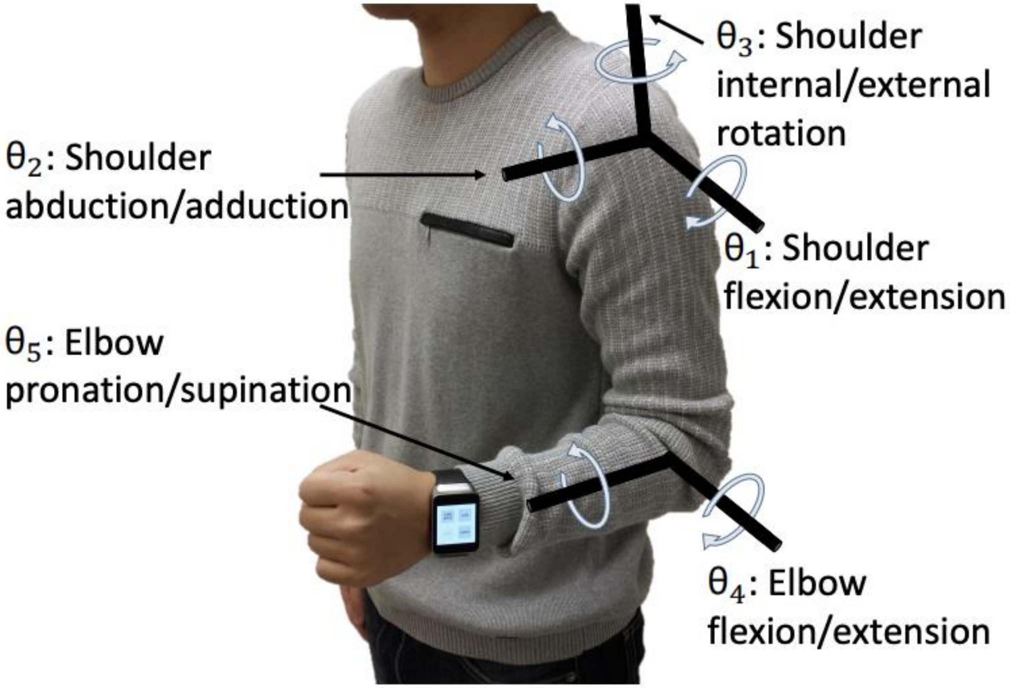


Figure 2: (a) Wrist location error diverges with double integral. (b) Wrist orientation error remains small over time.

Basic Insights

- Gyroscope is good in capturing the orientation of the arm, but accelerometer fails to capture the arm location (as we have seen earlier …)
- The arm movement is not fully random
  - There are predefined (5) degrees of freedom



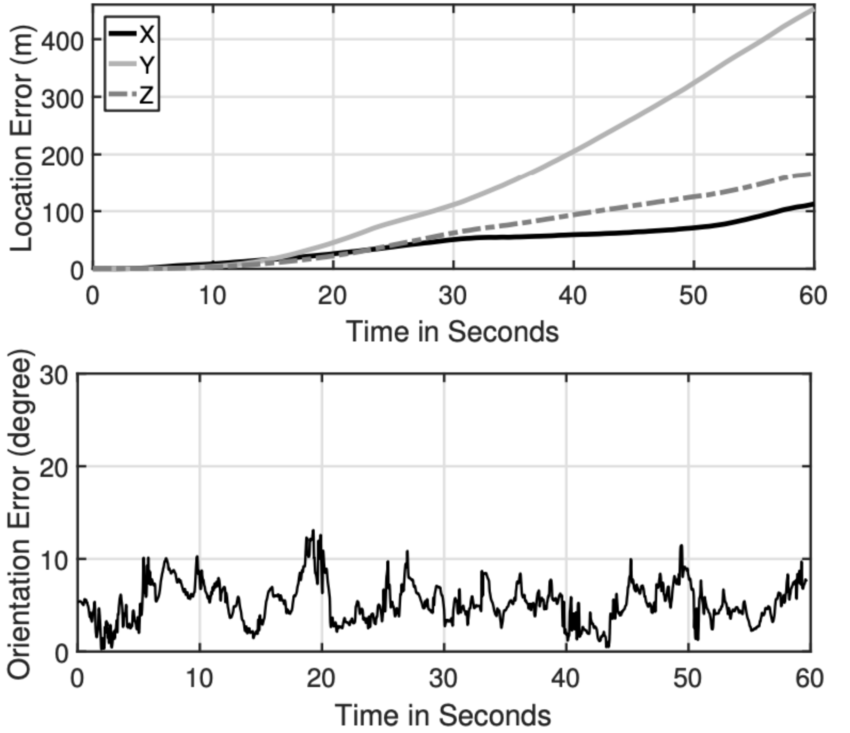
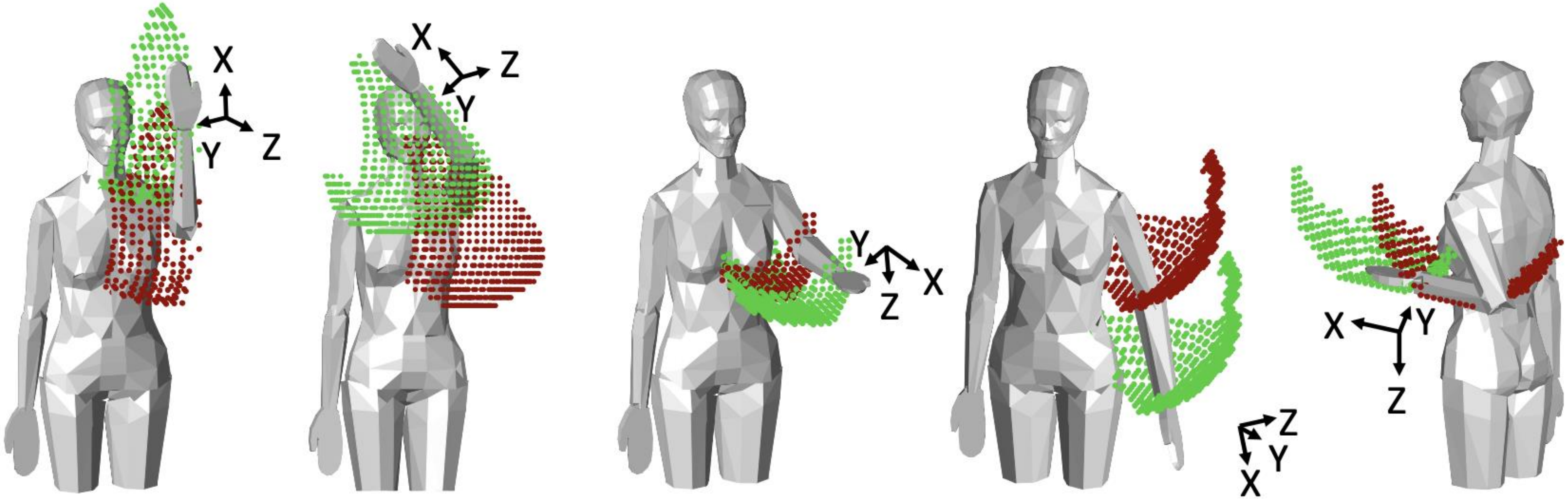


Figure 2: (a) Wrist location error diverges with double integral. (b) Wrist orientation error remains small over time.

Basic Insights

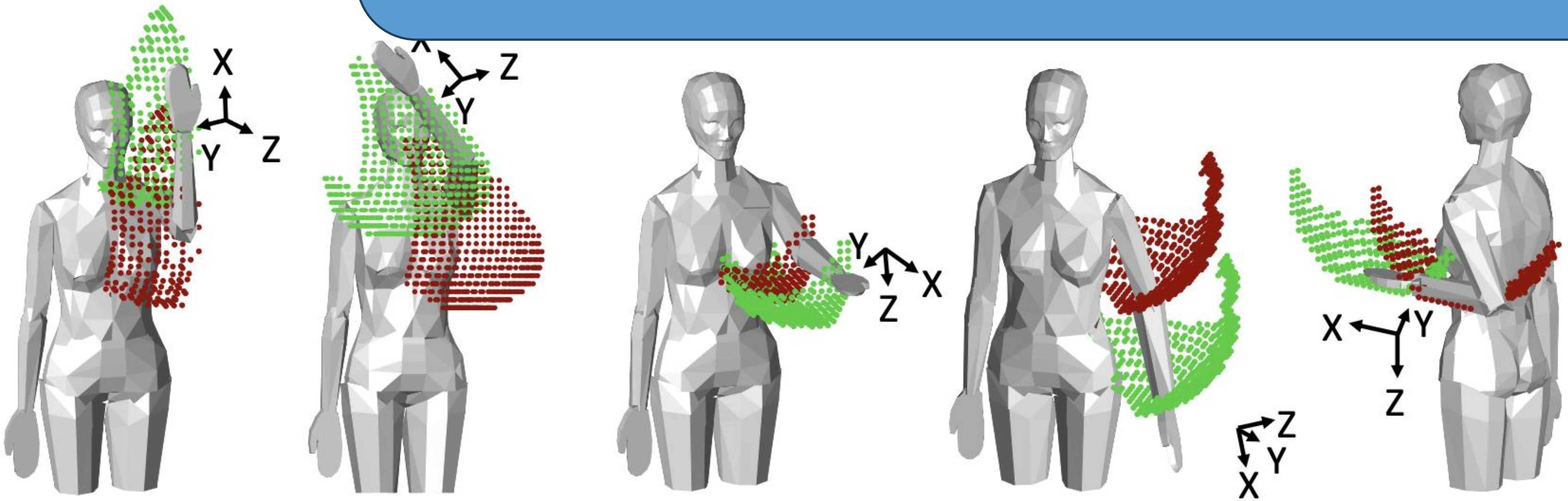
- Gyroscope is good in capturing the orientation of the arm, but accelerometer fails to capture the arm location (as we have seen earlier …)
- The arm movement is not fully random
  - There are predefined (5) degrees of freedom
  - Accordingly, there can be possible gestures



Basic Insights

- Gyroscope is good in capturing the orientation of the arm, but accelerometer fails to capture the ar
- The arm movement i
  - There are predefined
  - Accordingly, there ca

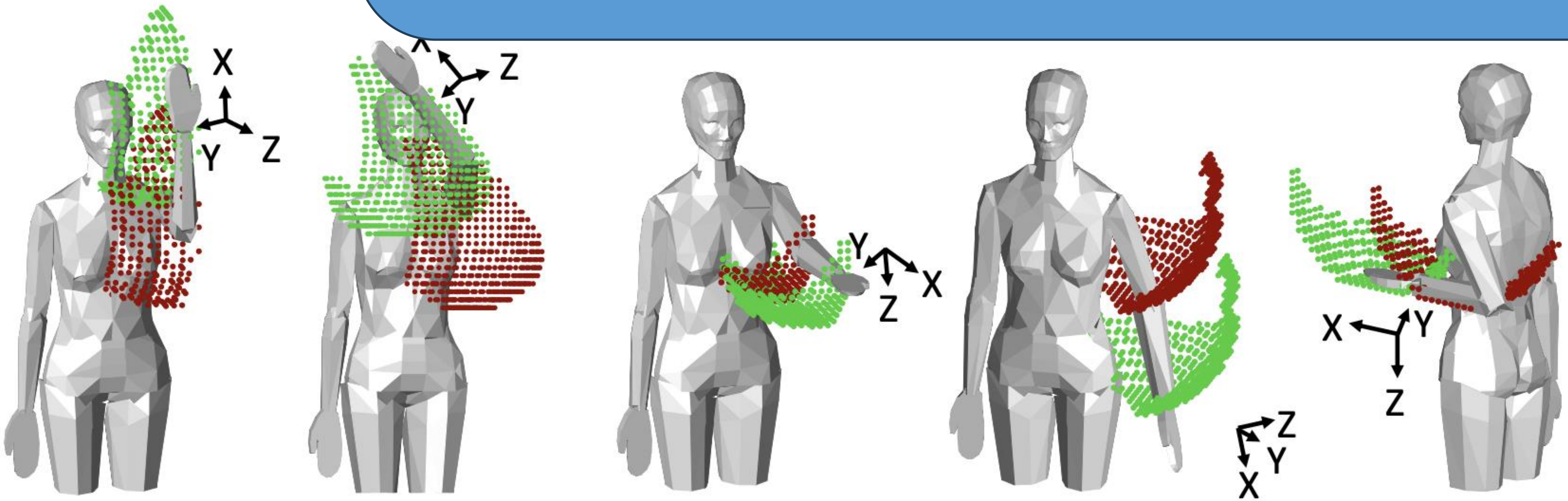
Can we use watch's orientation to detect/predict these gestures?



Basic Insights

- Gyroscope is good in ... er fails to capture the a...
- The arm movement i...
    - There are predefined...
    - Accordingly, there ca...

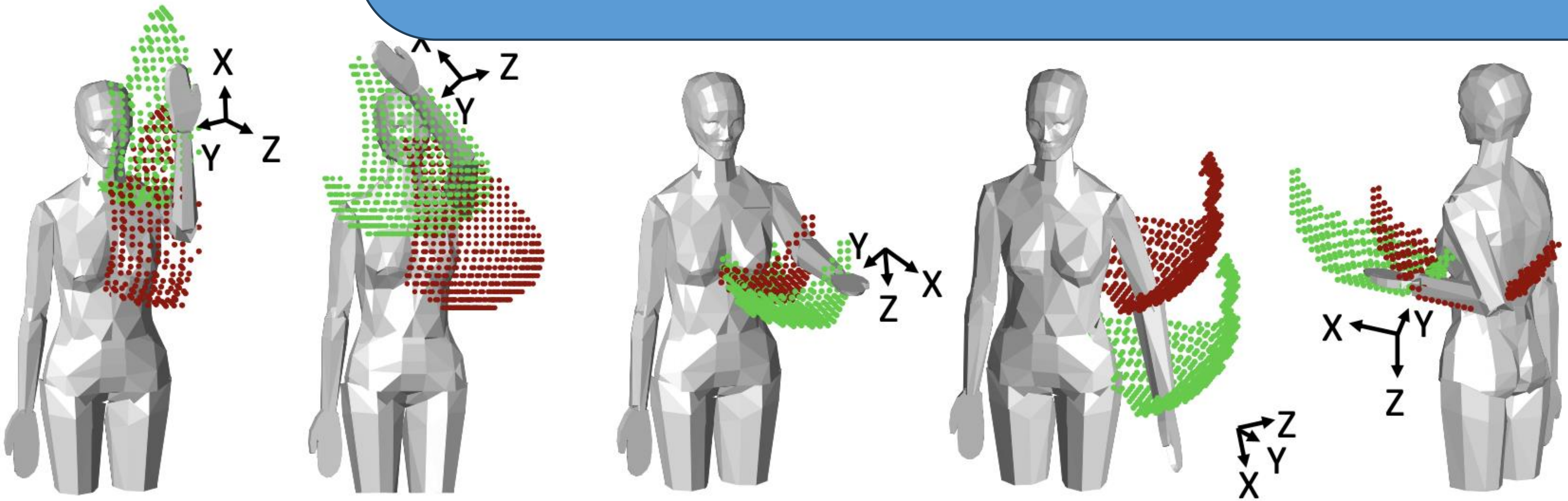
Core Idea: If you fix the orientation of the smartwatch, then there can be a fixed number of possible movements



Basic Insights

- Gyroscope is good in ... er
  fails to capture the a...
- The arm movement i...
  - There are predefine...
  - Accordingly, there ca...

However, this number is large!
Can we use the orientation information to fix the localization error?



We Need a Fixed Coordinate System

- **Torso Coordinate System (TCS)**
  - The left shoulder is the origin
  - The plane of the user's torso (i.e., the chest) is the XY plane
  - The Z axis will be the line emanating from the left shoulder in the frontward direction, perpendicular to the torso

We Need a Fixed Coordinate System

- **Torso Coordinate System (TCS)**
  - The left shoulder is the origin
  - The plane of the user's torso (i.e., the chest) is the XY plane
  - The Z axis will be the line emanating from the left shoulder in the frontward direction, perpendicular to the torso

- **An interesting system:** Can you convert the TCS to the Global Coordinate System (GCS)?
  - Some application needs GCS; for example, pointing a TV remote towards the TV

We Need a Fixed Coordinate System

- **Torso Coordinate System (TCS)**
  - The left shoulder is the origin
  - The plane of the user's torso (i.e., the chest) is the XY plane
  - The Z axis will be the line emanating from the left shoulder in the frontward direction, perpendicular to the torso

- **An interesting system:** Can you convert the TCS to the Global Coordinate System (GCS)?
  - Some application needs GCS; for example, pointing a TV remote towards the TV
  - The compass in the watch can be used to map TCS to GCS, but you need the facing direction
  - *ArmTrack* assumes that the facing direction is available!

We Need a Fixed Coordinate System

- **Torso Coordinate System (TCS)**
  - The left shoulder is the origin
  - The plane of the user's torso (i.e., the chest) is the XY plane
  - The Z axis w ction, perpendicu

- **An interesti** e
  System (GCS
  - Some appl
  - The compa g direction
  - *ArmTrack* assumes that the facing direction is available!

Both the orientation and the location of the smartwatch can be represented with respect to this TCS. We need the gravity offset subtraction to map the IMU readings to the TCS.

The Physics

- **Hypothesis**: A motion of a body can be decomposed into translational and rotational motions
- Movement of a watch from Point A to Point B
  - A translational motion from A to B
  - A rotational motion to change the orientation at B

The Physics

- **Hypothesis**: A motion of a body can be decomposed into translational and rotational motions
- Movement of a watch from Point A to Point B
  - A translational motion from A to B
  - A rotational motion to change the orientation at B

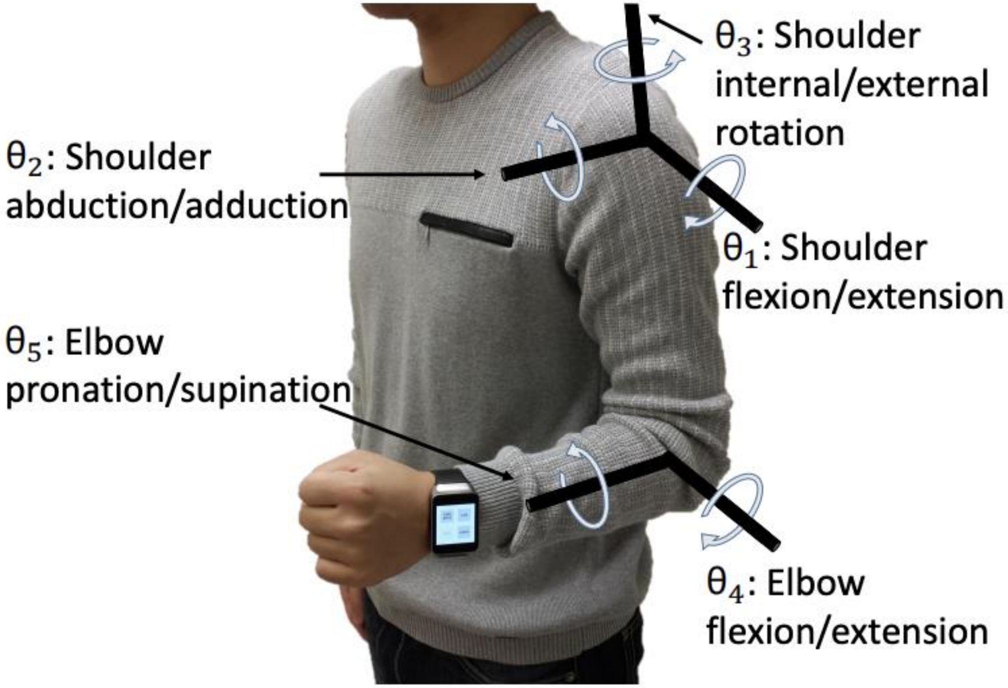
- Assume that you know the coordinate of Point A at TCS, and the accelerometer and gyroscope readings are super accurate!
  - Use double integration on accelerometer (after gravity offset subtraction) to estimate the translational shift
  - Use gyroscope to estimate the orientation at Point B (we have seen this earlier)

The Physics

- **Hypothesis**: A motion of a body can be decomposed into translational and rotational motions
- Movement of a watch from Point A to Point B
  - A translational motion from A to B
  - A rotational motion to change the orientation at B
- Assume that you know the coordinate of Point A at TCS, and the accelerometer and gyroscope readings are super accurate!
  - Use double integration on accelerometer (after gravity offset subtraction) to estimate the translational shift
  - Use gyroscope to estimate the orientation at Point B (we have seen this earlier)
  - **But unfortunately, this method does not work because of the reasons we discussed!**

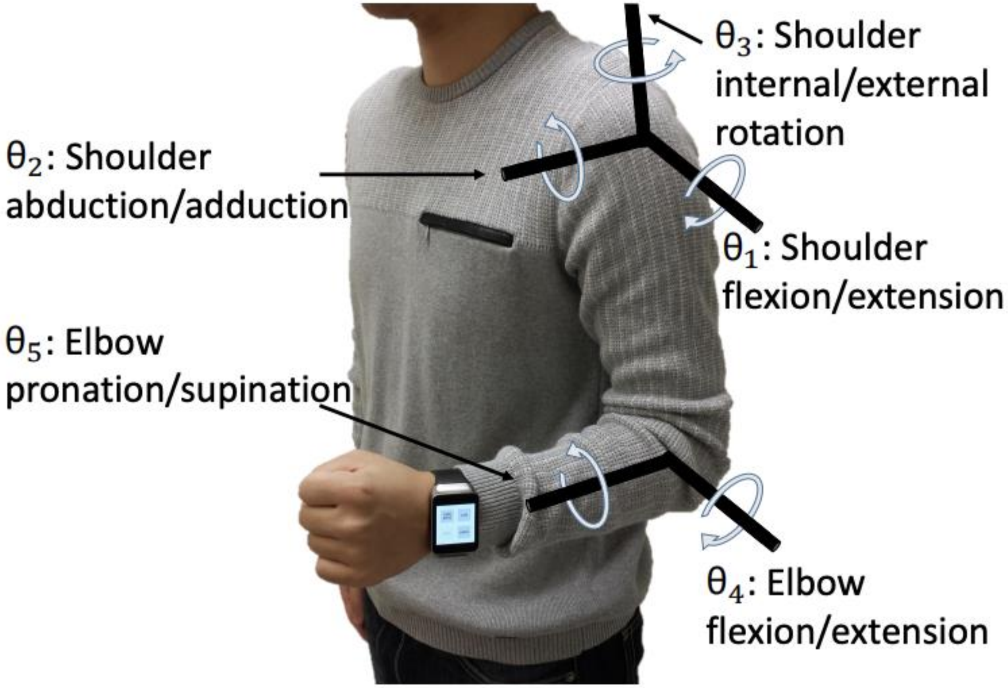
Using the Human Arm Kinematics

- Considering the 5-DoF, we have three parameters to model:
  - Wriest orientation
  - Wrist location
  - Elbow location



Using the Human Arm Kinematics

- Considering the 5-DoF, we have three parameters to model:
  - Wriest orientation
  - Wrist location
  - Elbow location
- Let l_u and l_f define the length of the upper arm and the forearm
- The elbow location can be estimated using Denavit-Hartenberg transformation:

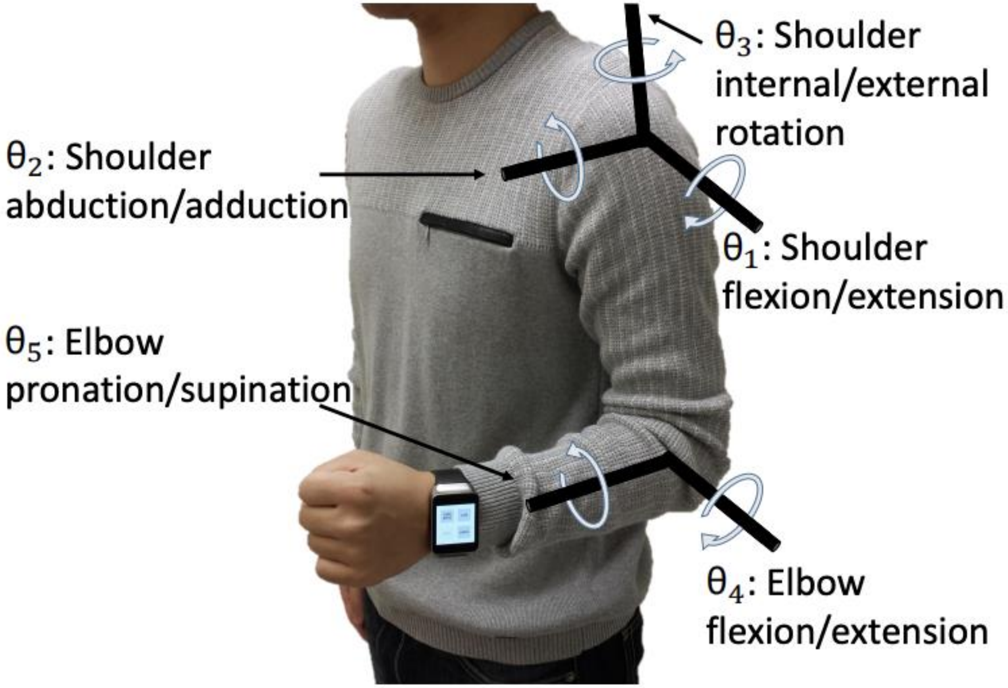


$$\mathrm{loc}_{\mathrm{elbow}} = f(\theta_1, \theta_2) = l_u \begin{pmatrix} \cos(\theta_2)\sin(\theta_1) \\ \sin(\theta_2) \\ -\cos(\theta_1)\cos(\theta_2) \end{pmatrix} \qquad \| \mathrm{loc}_{\mathrm{elbow}} \| = l_u$$

https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters

Using the Human Arm Kinematics

- Considering the 5-DoF, we have three parameters to model:
  - Wriest orientation
  - Wrist location
  - Elbow location
- Let l_u and l_f define the length of the upper arm and the forearm
- The elbow location can be estimated using Denavit-Hartenberg transformation:



$$\mathrm{loc}_{\mathrm{elbow}} = f(\theta_1, \theta_2) = l_u \begin{pmatrix} \cos(\theta_2)\sin(\theta_1) \\ \sin(\theta_2) \\ -\cos(\theta_1)\cos(\theta_2) \end{pmatrix} \qquad \|\mathrm{loc}_{\mathrm{elbow}}\| = l_u$$

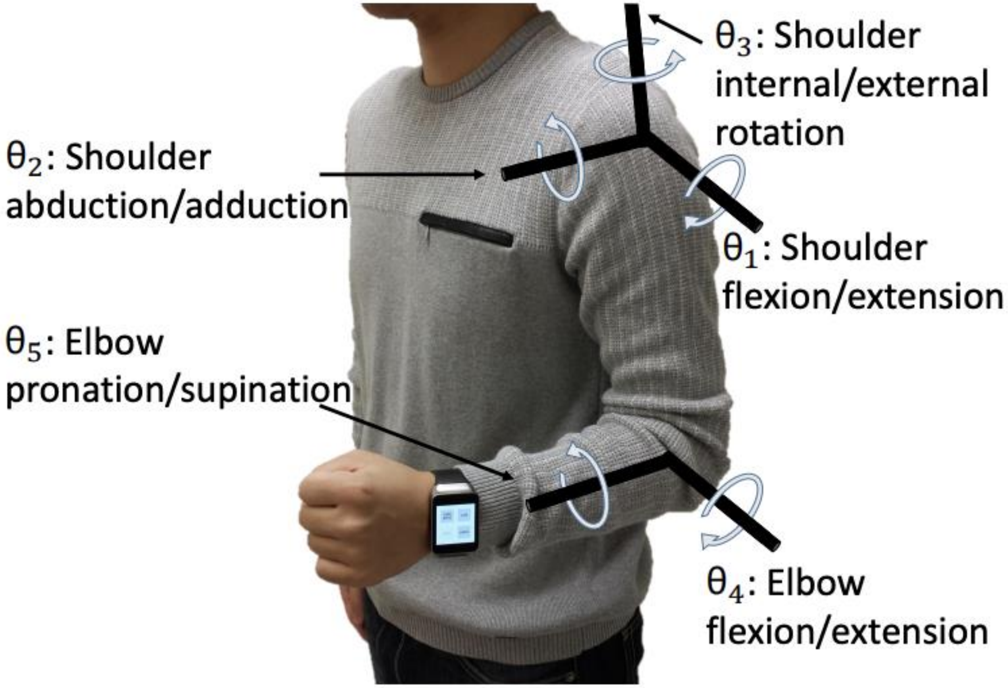
https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters

Using the Human Arm Kinematics

- Considering the 5-DoF, we have three parameters to model:
  - Wriest orientation
  - Wrist location
  - Elbow location
- Let l_u and l_f define the length of the upper arm and the forearm
- Similarly, the wrist's relative location to elbow can also be computed:

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

$$\| \mathrm{loc}_{\mathrm{wrist-to-elbow}} \| = l_f$$



https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters

Using the Human Arm Kinematics

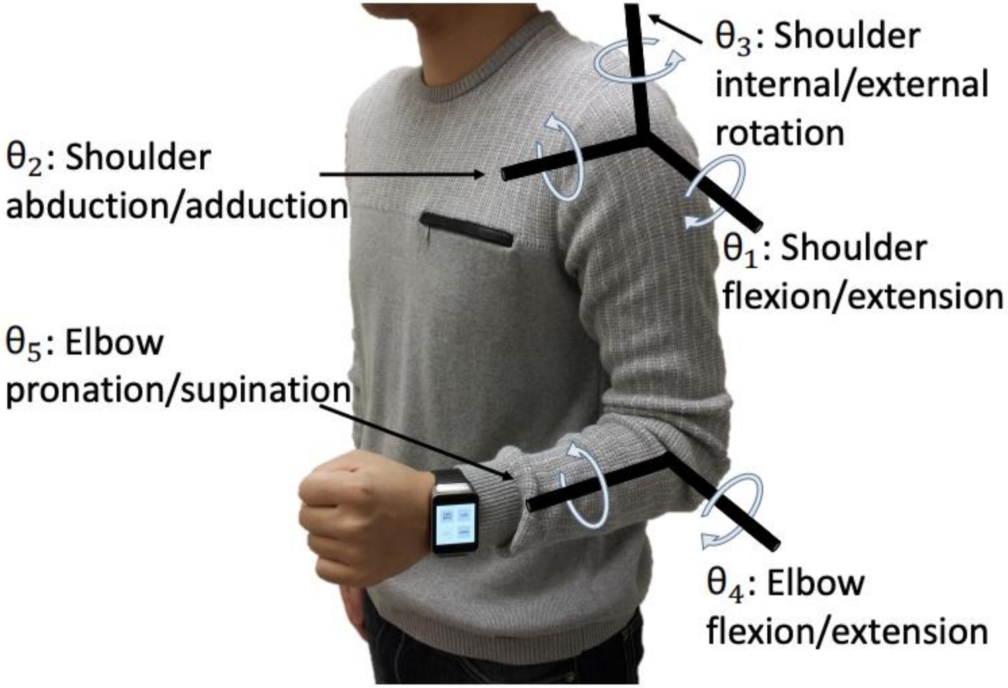
- Finally, the location of the wrist becomes a function,

$$\text{loc}_{\text{wrist}} = \text{loc}_{\text{elbow}} + \text{loc}_{\text{wrist-to-elbow}}$$

- The orientation of the wrist is also a function of the 5-DoFs

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

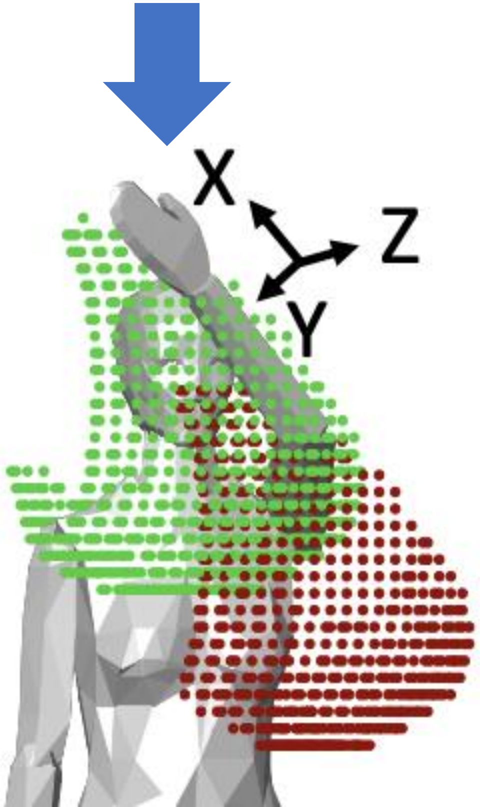
- So, if we know the five θs, we can solve the entire arm posture; wrist orientation, wrist location, and elbow location



https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters

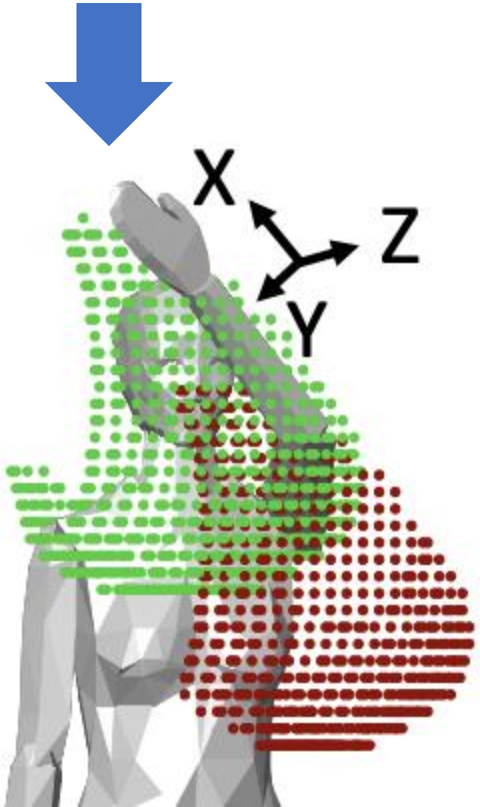
Mapping Orientation to Point Cloud

- **Input:** Watch Orientation
- **Output:** Elbow and Wrist's location point cloud



Mapping Orientation to Point Cloud

- **Input:** Watch Orientation
- **Output:** Elbow and Wrist's location point cloud
- **Hypothesis**: The average range of motion for each joint angle is fixed!



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°

- So, we know what can be their possible ranges! Use the previous set of equations to generate all possible combinations

Watch Orientation to Point Cloud Mapping

Algorithm 1 Watch Orientation to Point Cloud Mapping

1: ElbowPointCloud = Empty Dictionary
2: WristPointCloud = Empty Dictionary
3: **for all** $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \in \text{ROM}$ **do**
4: $\quad \text{loc}_{\text{elbow}} = f(\theta_1, \theta_2)$
5: $\quad \text{Rot}_{\text{watch}} = h(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$
6: $\quad \text{loc}_{\text{wrist-to-elbow}} = \text{Rot}_{\text{watch}}(t) \begin{pmatrix} l_f \\ 0 \\ 0 \end{pmatrix}$
7: $\quad \text{loc}_{\text{wrist}} = \text{loc}_{\text{elbow}} + \text{loc}_{\text{wrist-to-elbow}}$
8: $\quad \text{ElbowPointCloud}[\text{Rot}_{\text{watch}}].\text{Add}(\text{loc}_{\text{elbow}})$
9: $\quad \text{WristPointCloud}[\text{Rot}_{\text{watch}}].\text{Add}(\text{loc}_{\text{wrist}})$
10: **end for**

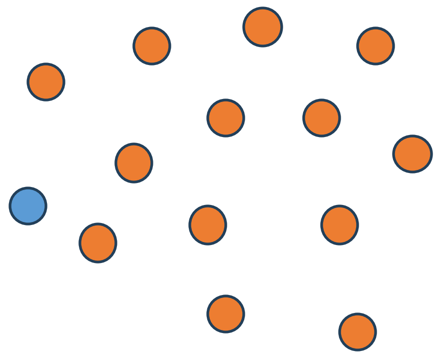
From Point Cloud to Elbow Location Estimation

- Once the orientation of the watch is known, posture estimation boils down to the elbow tracking problem
  - Wrist location can be computed as a static shift
- We have the following information
  - Reasonable estimate of orientation (may not be fully precise)
  - Point cloud of all possible elbow locations, for a given orientation

- **At any given time, where is the elbow in the point cloud?**

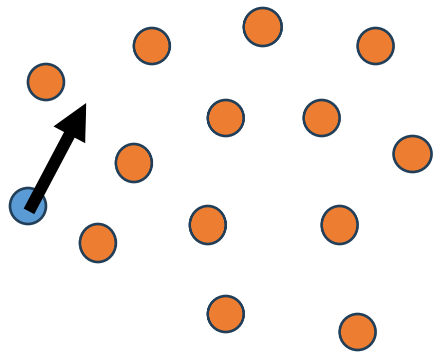
From Point Cloud to Elbow Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



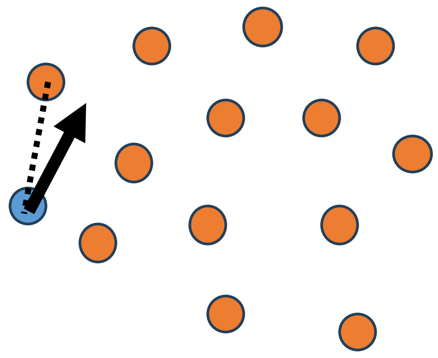
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



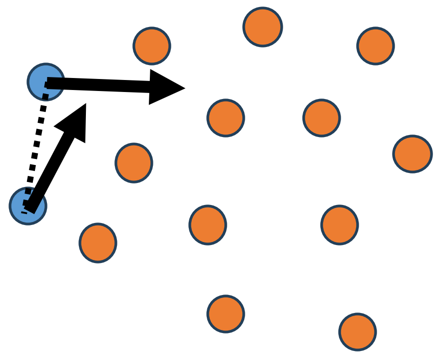
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



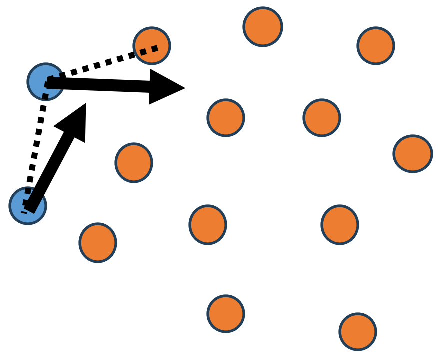
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



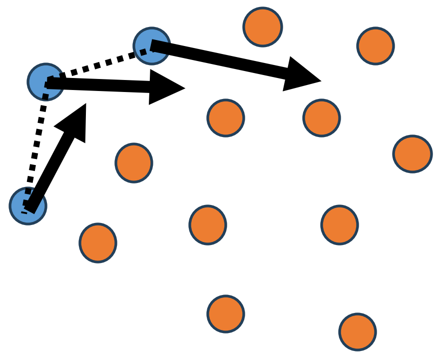
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



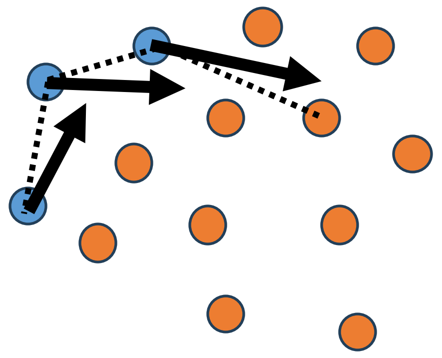
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



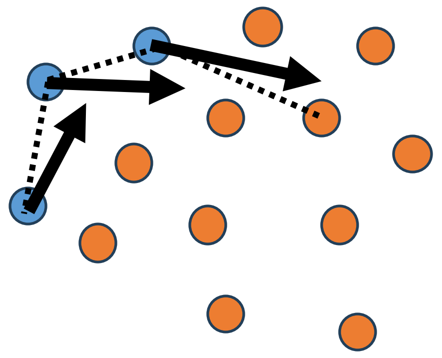
From Point Cloud to Location Estimation

- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?



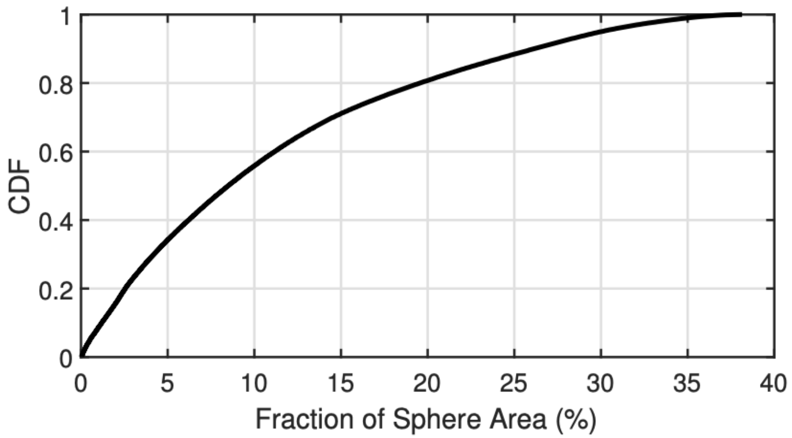
From Point Cloud to Location Estimation

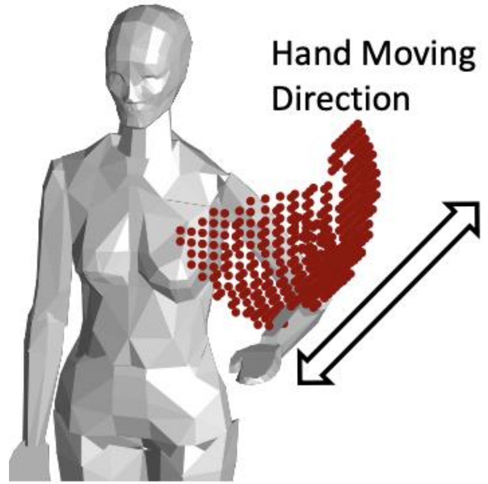
- Can be thought of as the state estimation framework
  - Given the current location, you know the next possible locations. If you know the possible displacement direction, can you estimate what would be the next location?
- Motivates a discrete space hidden Markov model
  - With Elbow's point cloud as the prior
- Note that all the points in the point cloud are not equally likely
  - Depends on the general human arm movement kinematics, depending on the location of the elbow
  - Use a *Posture Prior Module (PPM)* to determine priors



A Simple Solution May Work for Many Problems

- You may not need a very high accuracy all the time (say, eating episode detection)
- Note that, the point cloud is often quite small
    - Covers less than 10% of the sphere around the shoulder
- Simply using the average location of the point cloud may provide a good estimate
- However, this may not work for all the gestures
    - Consider the case of punching





Scope for Improvement

- As we discussed earlier, smartwatch sensors can work as an estimate of the elbow's movement
    - For punching, the accelerometer shows accleration to a straight direction

Scope for Improvement

- As we discussed earlier, smartwatch sensors can work as an estimate of the elbow's movement
  - For punching, the accelerometer shows accleration to a straight direction

- However, this is complicated in the reality, as the elbow may also have rotational motions
  - Acceleration has to be computed as a fusion of accelerometer and gyroscope

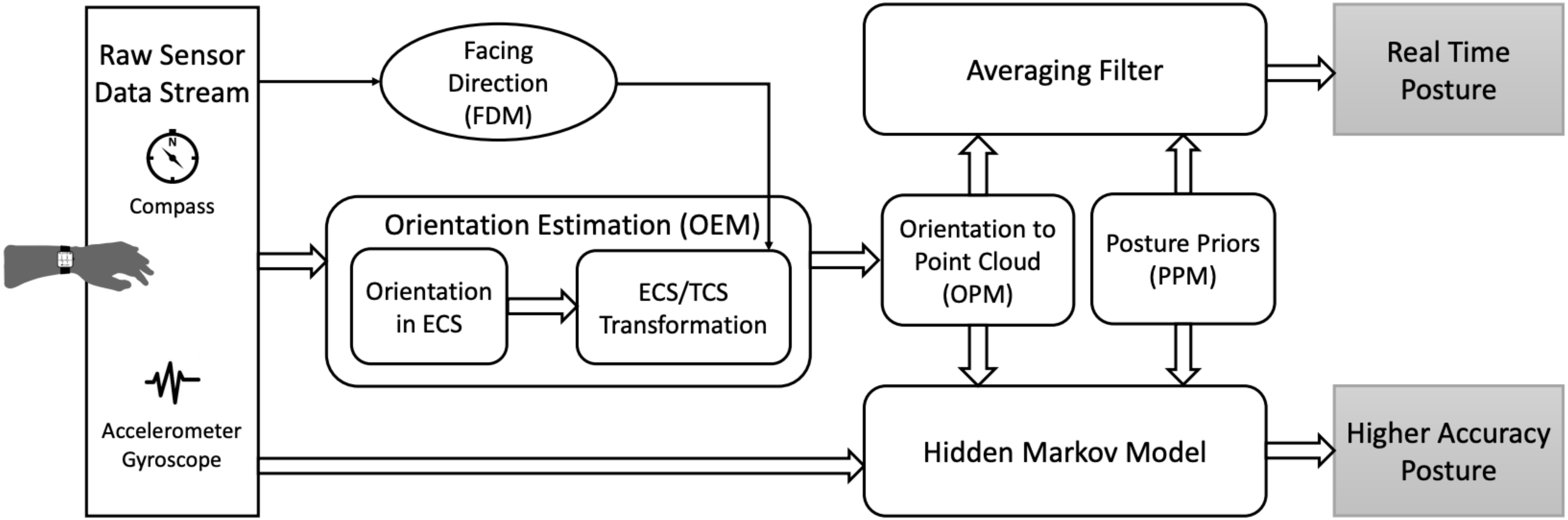
Using the Motion Vectors for Fine Corrections

- We have seen earlier how to estimate the acceleration in the presence of rotational motion.
- Use that formulation to ask the question: ***Which sequence of elbow locations best matches the measured elbow acceleration?***
- A **dynamic programming problem:** Given the point cloud at each time step, find out the location that matches with the target acceleration.
  - Use a Hidden Markov Model (HMM) to solve this problem: Iteratively search the optimal solution in the search space (point cloud locations)

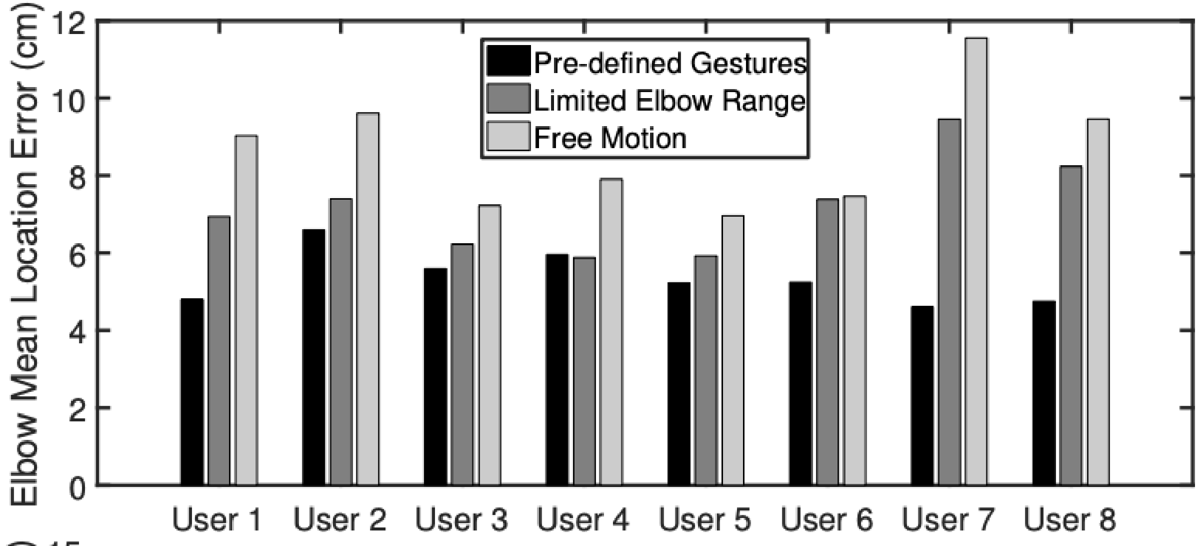
The Hidden Markov Model

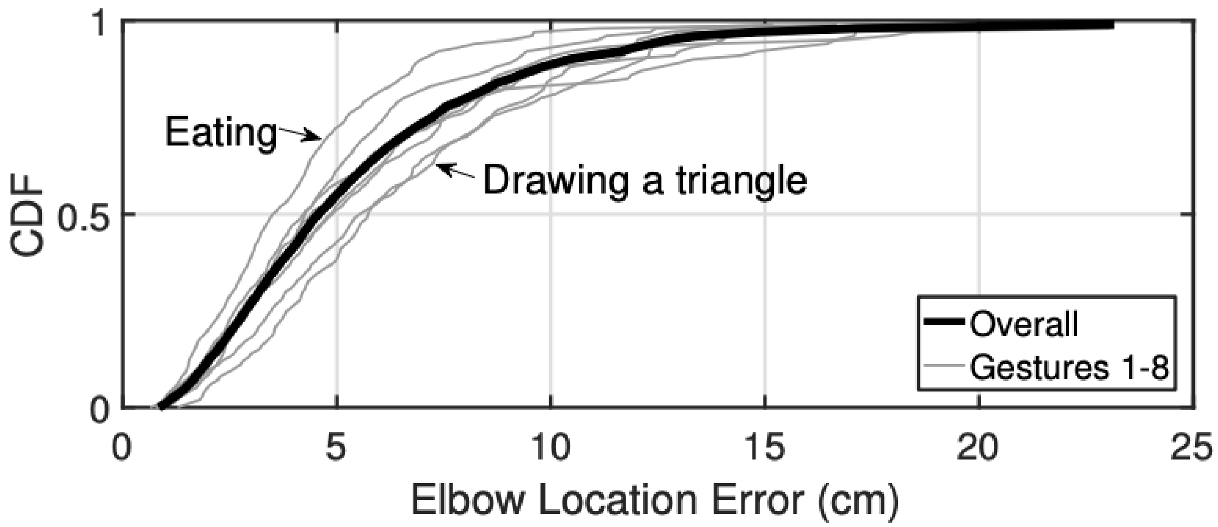
- **State Space**: <Elbow's previous location, Elbow's current location>
- **Prior Probability**: Uniform probability
  - As we do not know the initial elbow location
- **Transition Probability:** Product of three probabilities
  - Since the elbow transition is continuous, two successive locations will nearly be the same (considering time to be infinitesimally small).
    - An indicator variable denoting whether two successive location is the same
  - The acceleration computed between two successive states (location -> velocity -> acceleration) should be equal to the target observed acceleration from the IMU. Assume that the error difference follows Gaussian distribution
  - The current elbow location in the new state must be inside the point cloud inferred at that point of time: Probability that the current location is inside the computed point cloud

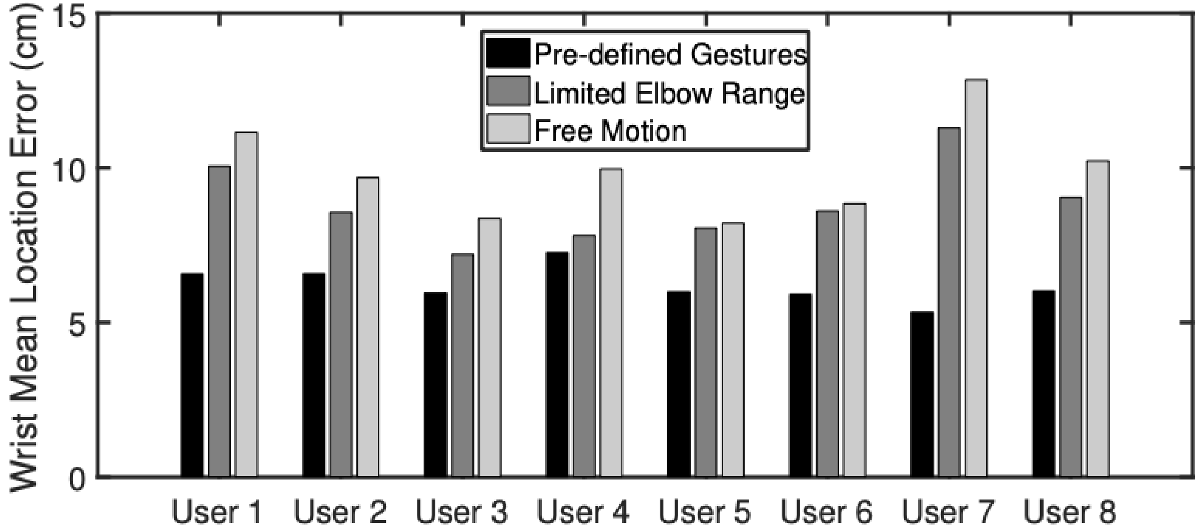
ArmTrak Architecture

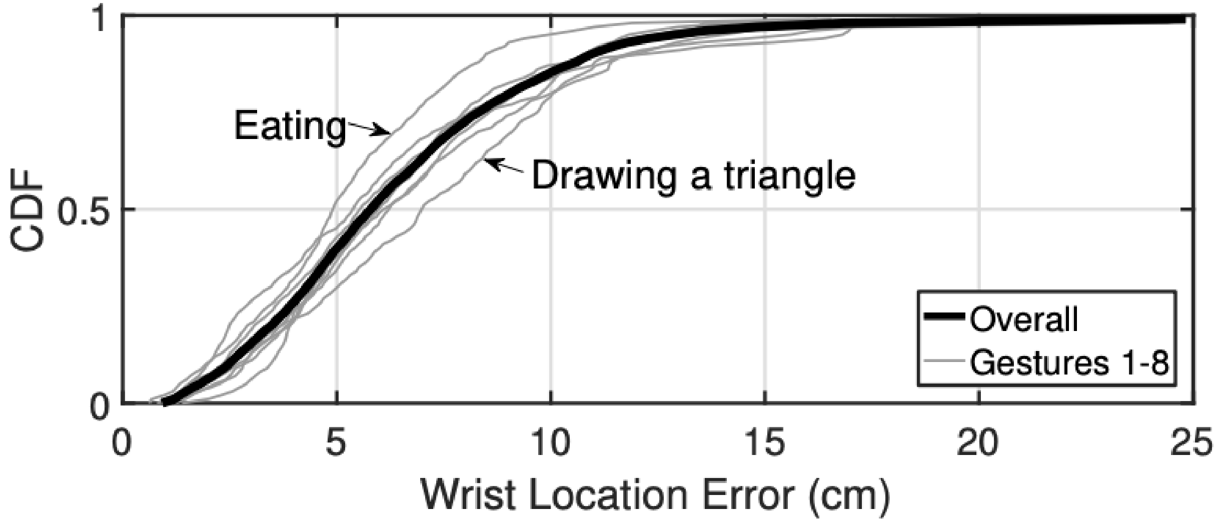


Performance of *ArmTrak*

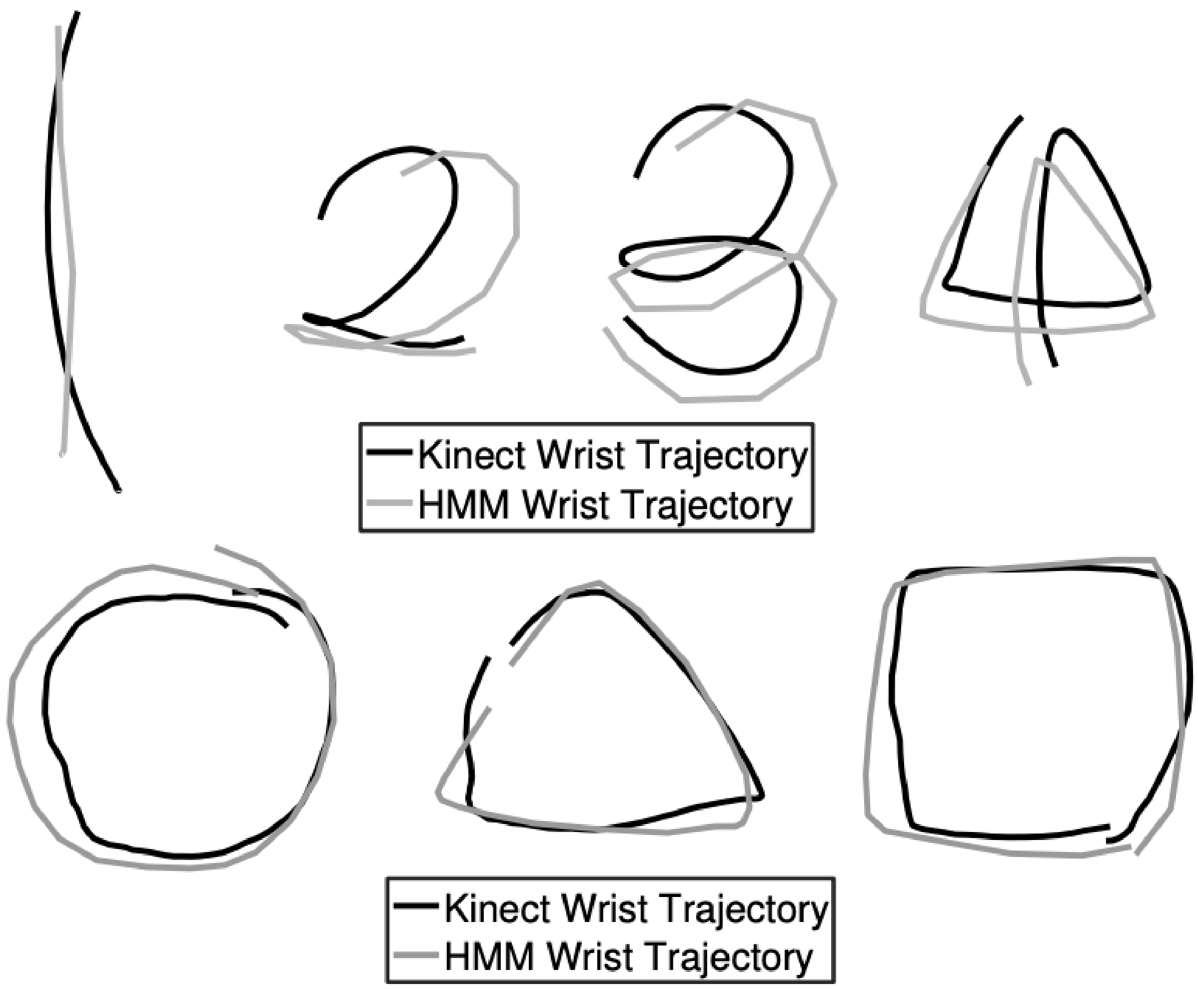


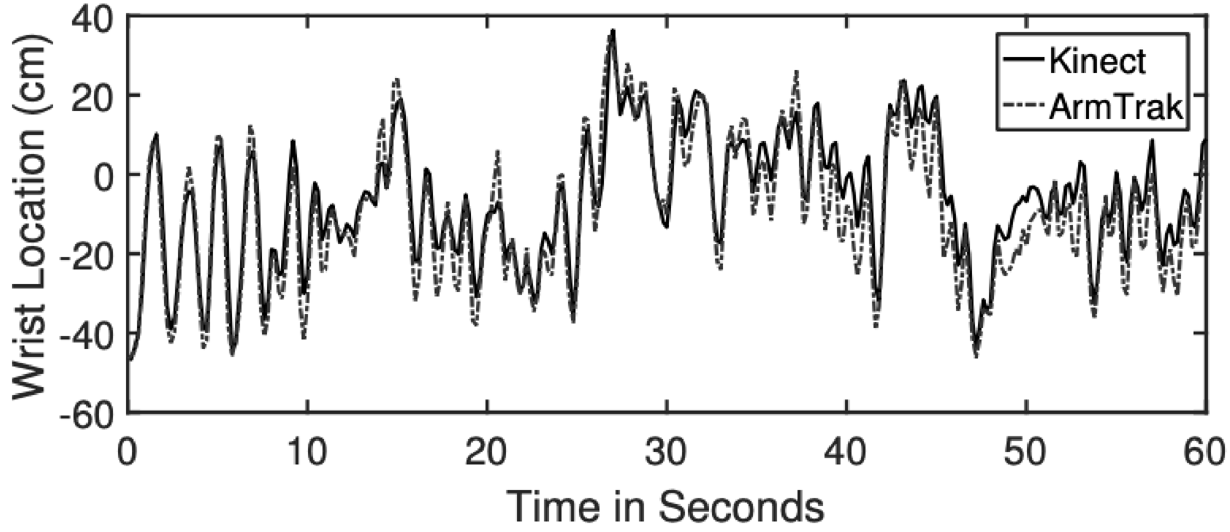






Performance of *ArmTrak*





Happy Learning!

Some resources related to this topic