Motion Tracking using IMU Sensors (2)

Hua Huang

In this lecture

- Quaternion for orientation representation

- Motion Tracking using IMU sensors

Quaternion Representations of Orientation

Orientation representations

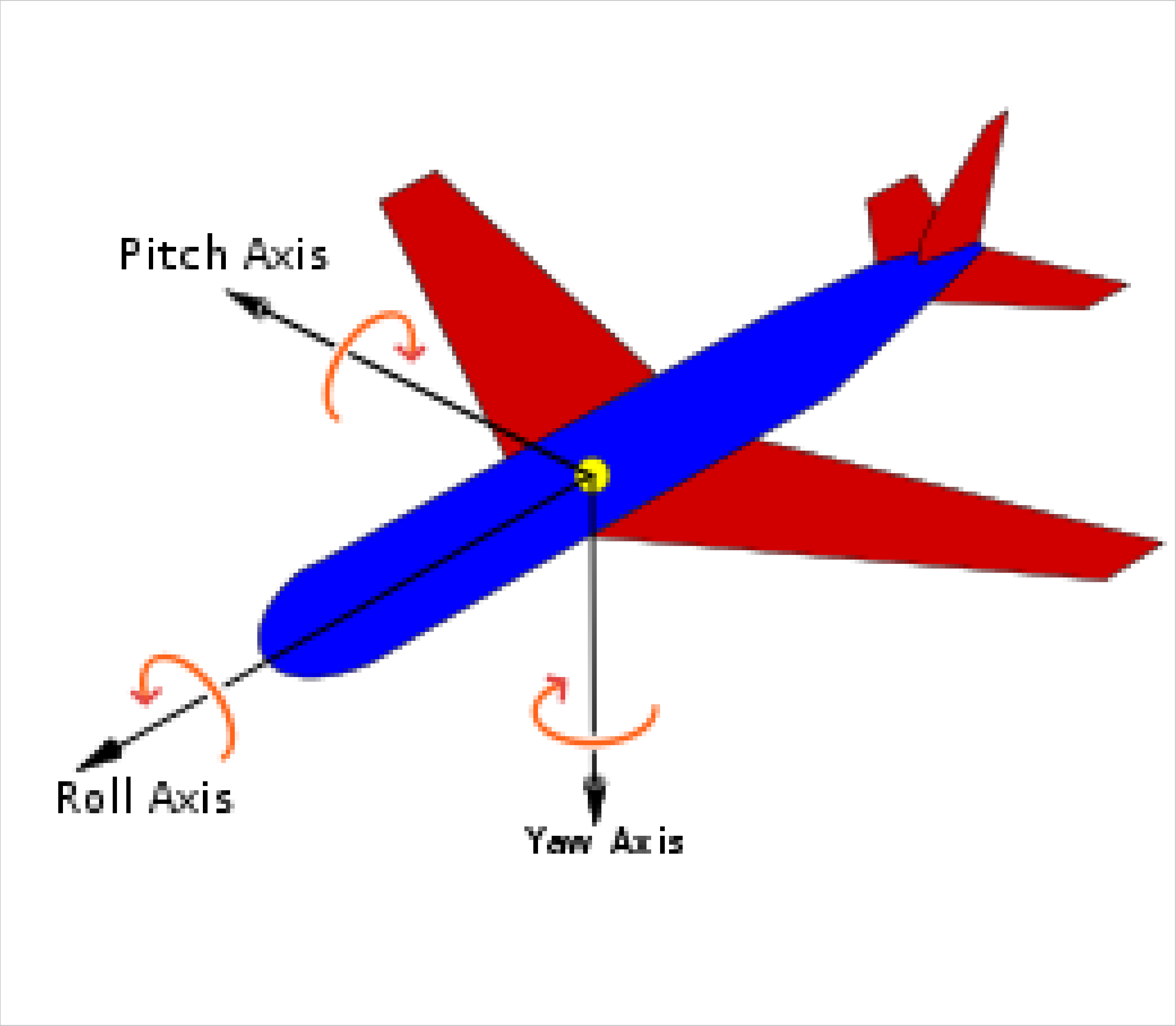
- Euler angles
- Rotation vectors (axis/angle)
- 3x3 matrices
- Quaternions

Direct Matrix Representation

- Recall that the 3*3 matrix can represent the orientation of the object.
- Rotation matrix can actually perform the rotation of vectors
- Why consider other representation?
  - Numerical issues
  - Storage issues
  - User interaction issues
  - Interpolation issues

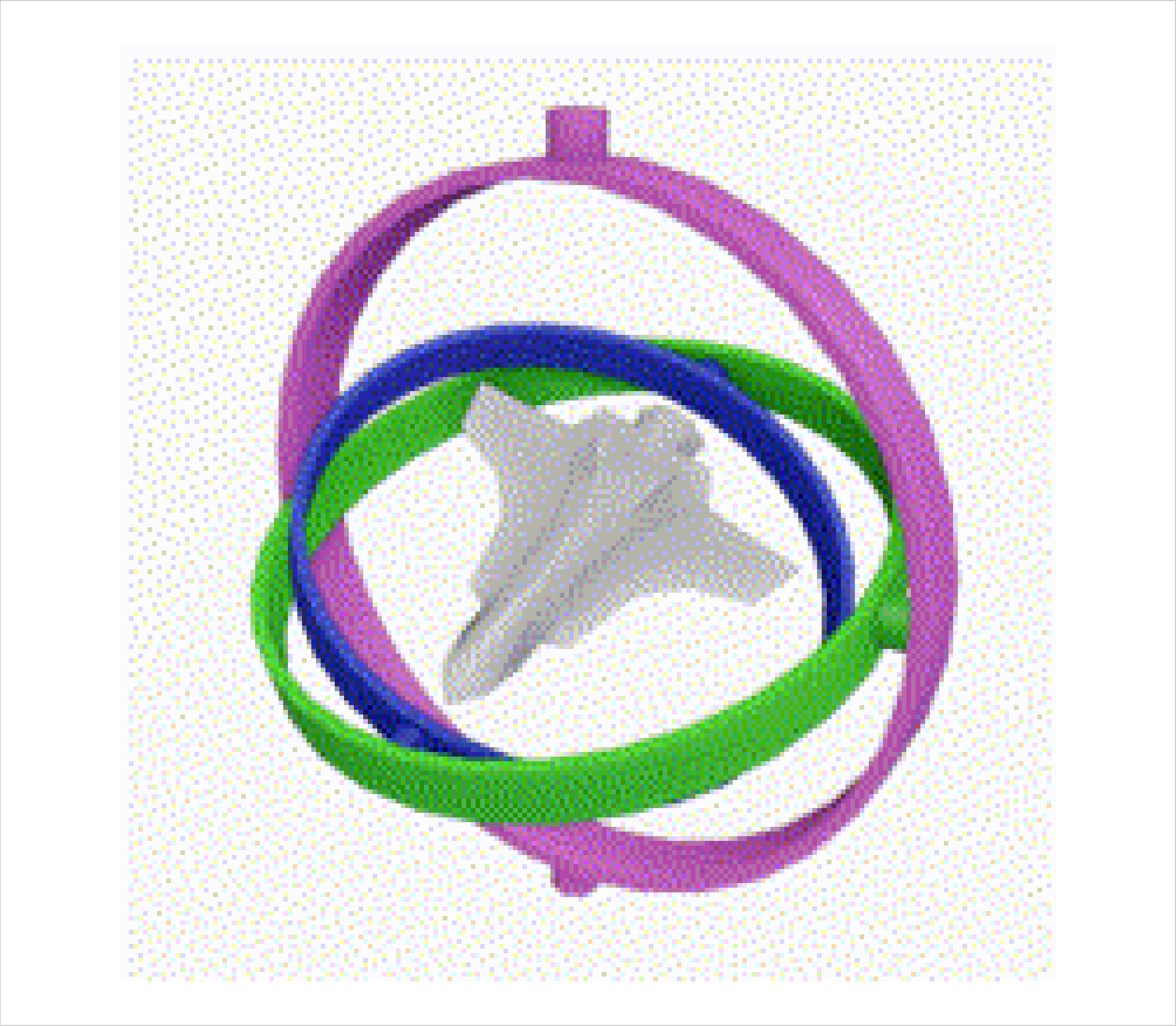
Euler Angle

- Generally, for vehicles, it is most convenient to rotate in roll (z), pitch (y), and then yaw (x)
- This is quite intuitive where we have a well-defined up direction

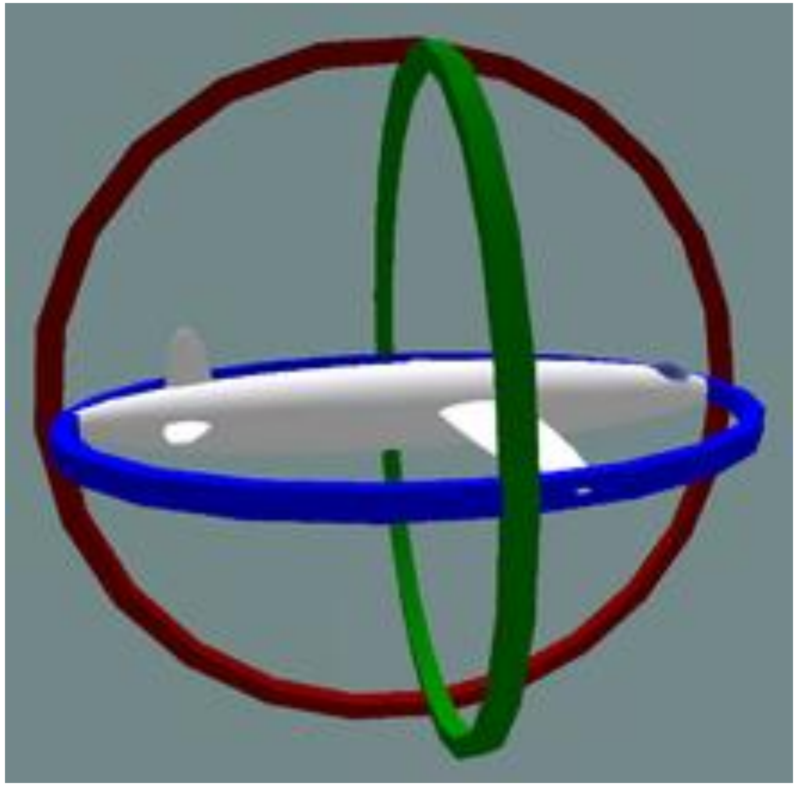


Gimbal Lock

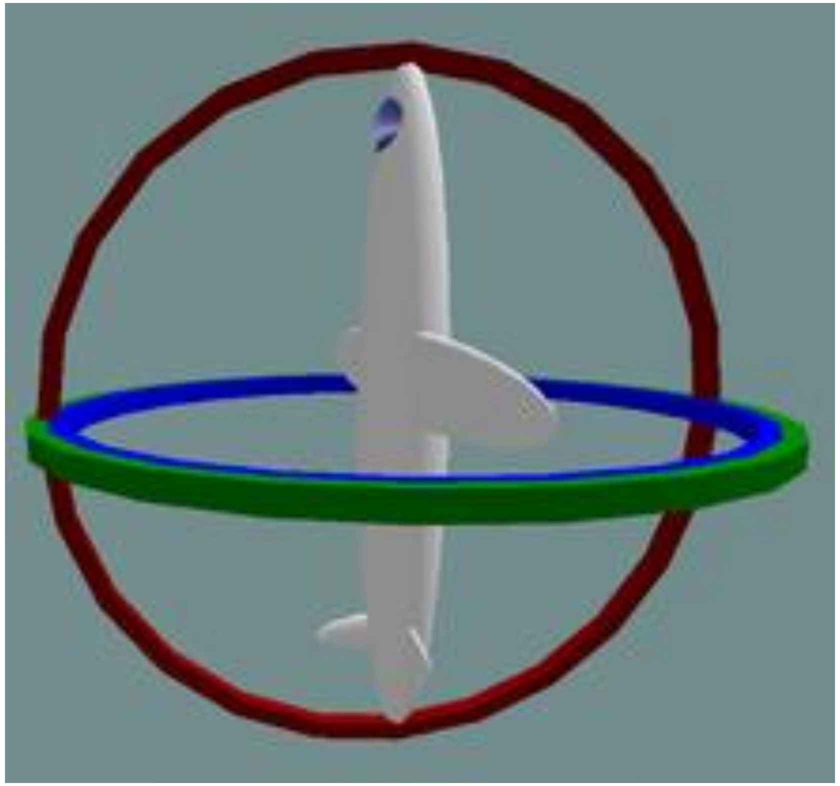
- A major problem with Euler angles is gimbal lock
- when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space.



Normal situation: three independent gimbals



Gimbal Lock: two gimbals are parallel

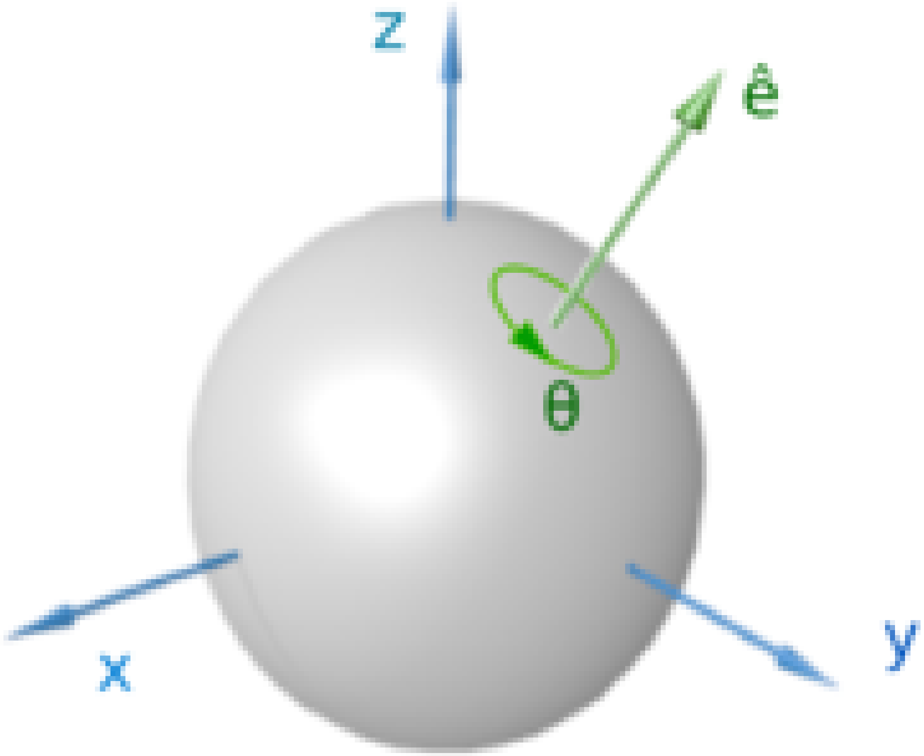


Pros and Cons of Euler Angles

- Pro
  - Human readable
  - Compact (3 numbers)
- Con
  - Gimbal lock
  - Not simple to concatenate rotations

Euler’s rotation theorem

- Any combination of rotations are equivalent to a single rotation about some axis that runs through the fixed point.
- This means that we can represent an arbitrary orientation as a rotation about some unit axis vector by some angle (4 numbers)



Complex numbers

- A complex number is a pair written formally as $a + bi$, where a and b are real numbers and i is the symbol for imaginary part.
- $i^2 = -1$
- Complex number algebra:

$$\begin{aligned}(a+bi)+(c+di) &= (a+c)+(b+d)i \\ (a+bi)(c+di) &= ac+adi+bci+bdi^2 \\ &= (ac-bd)+(ad+bc)i.\end{aligned}$$

Quaternions

- Quaternions are an extension of complex numbers that provide a way of rotating vectors
- In computer science, they are most useful as a means of representing orientations
  - Android uses quaternions to represent orientation

The algebra of quaternions

- A quaternion is a 4-tuple written as:

$$\boldsymbol{q} = s + q_1 i + q_2 j + q_3 k$$

- Where:

$$i^2 = j^2 = k^2 = -1$$
$$ij = k, ji = -k$$
$$jk = i, kj = -i$$
$$ki = j, ik = -j$$

The conjugate and norm of a quaternion

- The conjugate of $\boldsymbol{q} = s + q_1 i + q_2 j + q_3 k$ is:
$$\boldsymbol{q}^* = s - q_1 i - q_2 j - q_3 k$$

- The norm can be calculated as follows
$$|q| = \sqrt{qq^*} = \sqrt{s^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}$$

- Exercise: deduct the calculation

Unit Quaternions

- For representing rotations or orientations, 4 numbers is 1 too many, so as with axis/angle we use only unit length quaternions

$$|\mathbf{q}| = \sqrt{s^2 + q_1^2 + q_2^2 + q_3^2} = 1$$

Unit Quaternions as Rotations

- A unit quaternion represents a rotation by an angle θ around a unit axis vector $\boldsymbol{a}$ as

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\theta}{2} & a_x \sin\frac{\theta}{2} & a_y \sin\frac{\theta}{2} & a_z \sin\frac{\theta}{2} \end{bmatrix}$$

- Exercise: this q is of unit length. Why?

Quaternion to represent a vector

- Quaternions can represent vectors by setting the scalar part to 0 (i.e. the axis vector with 0 rotation)
- This vector (quaternion) needn't be unit length

i.e., the vector $\boldsymbol{a} = [a_x, a_y, a_z]$ can be represented as:

$$\boldsymbol{q} = a_x i + a_y j + a_z k$$

Rotation using Quaternion

- How to rotate the vector v counterclockwise by quaternion q, which represents rotating angle θ about axis $\boldsymbol{a}$ **?**
- Use the following equation

$$\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^{-1}$$

- q: unit quaternion
- v: the original vector
- v’: rotated vector

Conversion: Quaternion and rotation angles

- Question: what’s the quaternion to rotate along x axis for angle α?
  - Rotation axis: (1,0,0)
  - Rotation angle: α

- By definition: $\boldsymbol{q} = cos\frac{\alpha}{2} + sin\frac{\alpha}{2} * 1 * i + sin\frac{\alpha}{2} * 0 * j + sin\frac{\alpha}{2} * 0 * k$

- Same method for y and z axes

Example

- How to rotate a point (1,0,0) +90 degrees around the z axis?
- $\boldsymbol{q} = cos\frac{\alpha}{2} + sin\frac{\alpha}{2} * 0 * i + sin\frac{\alpha}{2} * 0 * j + sin\frac{\alpha}{2} * 1 * k$
- $\boldsymbol{q} = cos45 + sin45 * 0 * i + sin45 * 0 * j + sin45 * 1 * k$
- $\boldsymbol{q} = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}k$
- v=(1,0,0)

v'=qvq*= $\left(\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}k\right) * i * (\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}k) = \left(\frac{\sqrt{2}}{2}i + \frac{\sqrt{2}}{2}ki\right) * (\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}k)$

$= \frac{1}{2}i + \frac{1}{2}j - \frac{1}{2}ik - \frac{1}{2}jk = \frac{1}{2}i + \frac{1}{2}j + \frac{1}{2}j - \frac{1}{2}i = j$

Convert unit quaternion to Euler angle

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix}$$

Convert unit quaternion to rotation Matrix

- They are equivalent and can be converted to each other

$$
\begin{matrix}
1-2q_2^2-2q_3^2 & 2q_1q_2+2sq_3 & 2q_1q_3-2sq_2 \\
2q_1q_2-2sq_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3+2sq_1 \\
2q_1q_3+2sq_2 & 2q_2q_3-2sq_1 & 1-2q_1^2-2q_2^2
\end{matrix}
$$

Quaternion Summary

- Quaternions are 4D vectors that can represent 3D rigid body orientations
- We use unit quaternions for orientations (rotations)
- Quaternions are more compact than matrices to represent rotations/orientations
- Avoids gimbal locks
- Not human readable

ArmTrak: Tracking the user’s arm movements using a wearable sensor

Understanding human arm motion

How is the arm moving?

What is the meaning of this motion?

Gesture Recognition

What is the meaning of this motion?

Gesture Recognition

Running

Smoking

Drinking

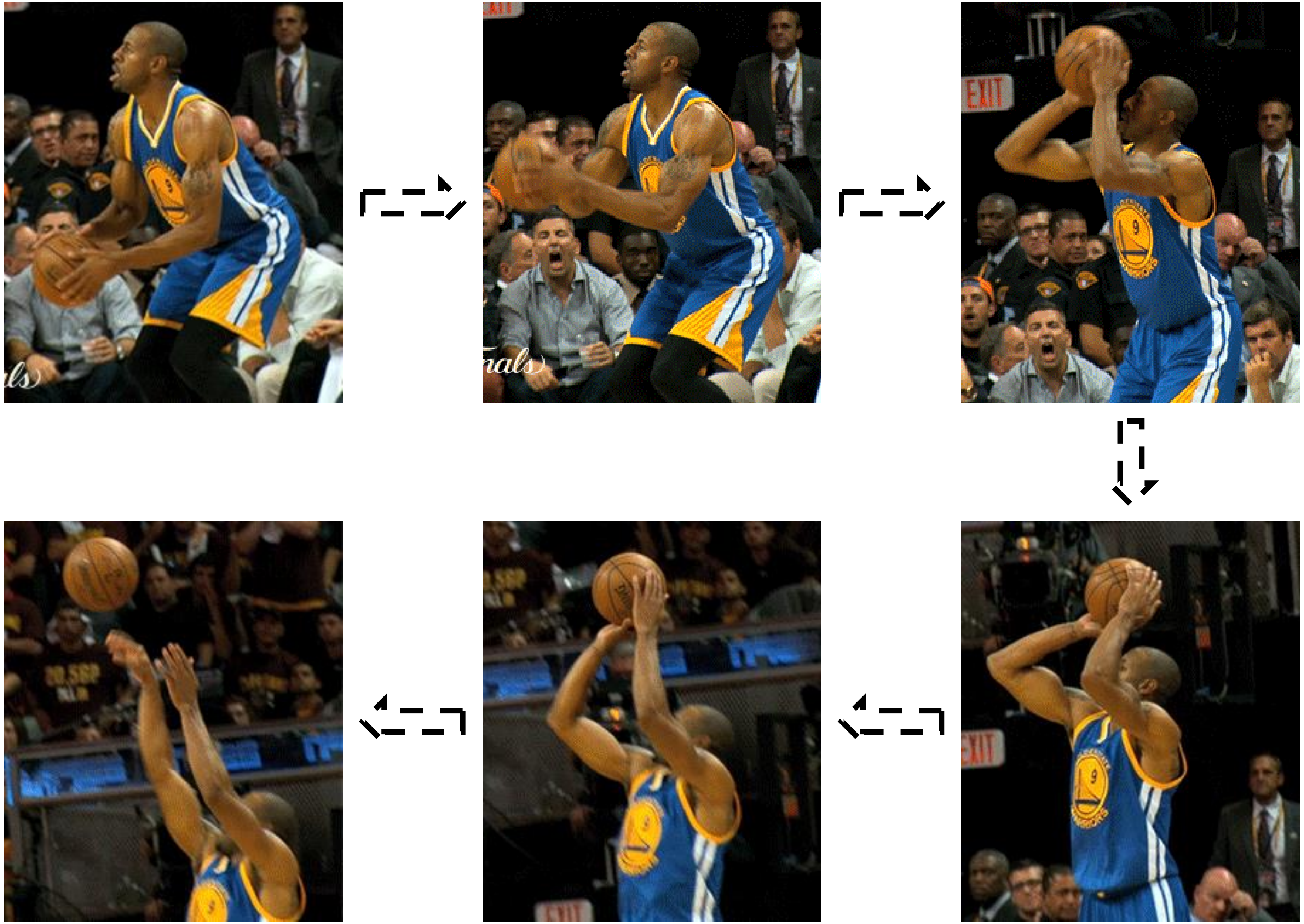
Driving

Posture Tracking

How is the arm moving?

Gesture Recognition

What is the meaning of this motion?


EXIT

Arm Posture Tracking - Applications

Natural User Interface

Sports Analytics

Can we track arm postures with a smartwatch alone?

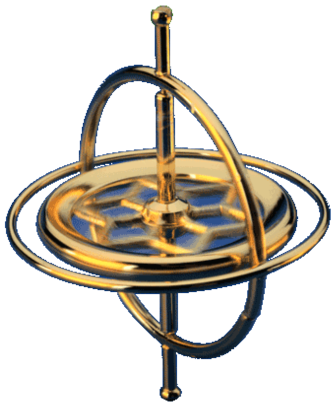
Can we track arm postures with a **smartwatch** alone?

- What is inside a smartwatch?

Accelerometer

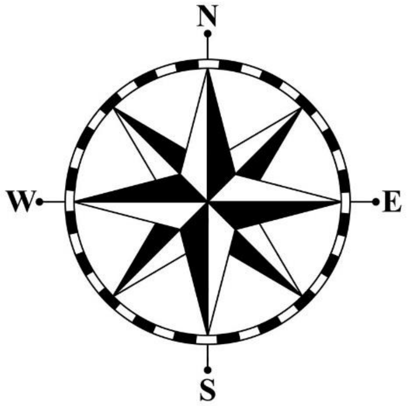
Acceleration along 3 axes

Gyroscope



Rotation speed around 3 axes

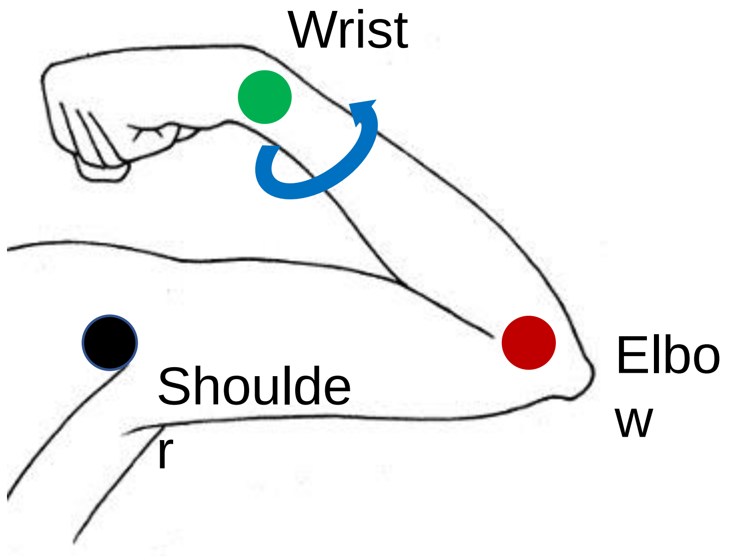
Compass



North vector projected to 3 axes

Can we track **arm postures** with a smartwatch alone?

- What do we need to track?



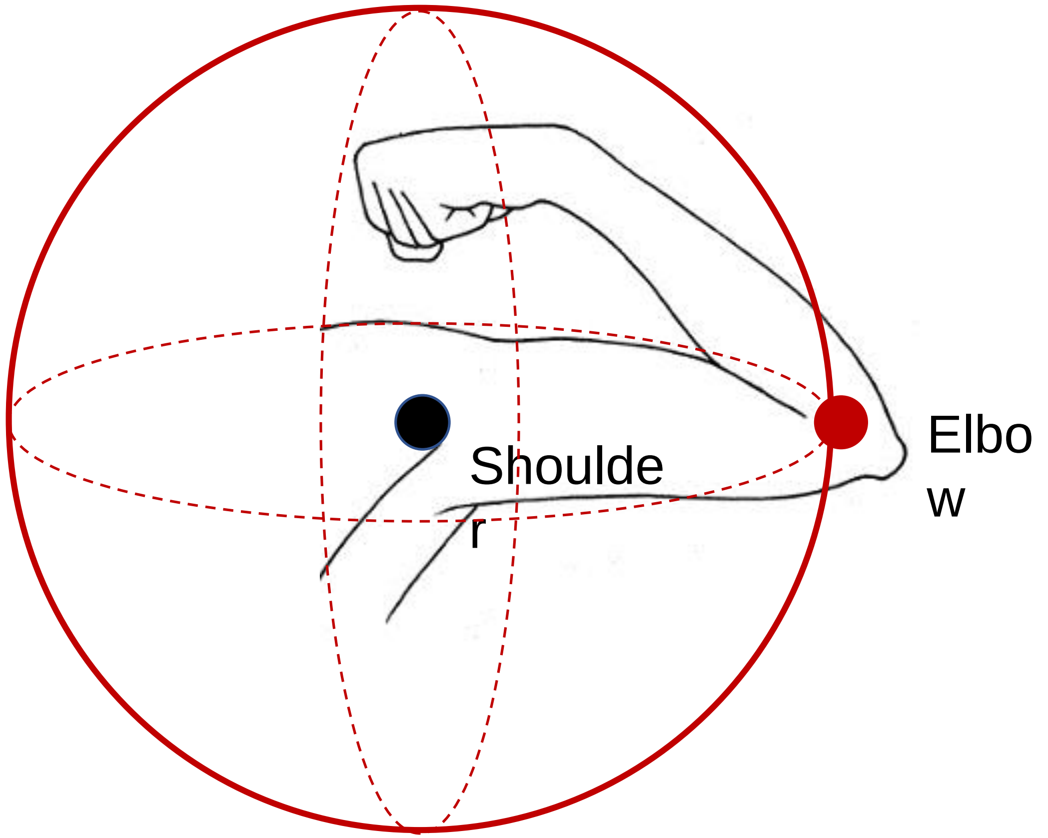
Posture = < Elbow Location, Wrist Location, Wrist Rotation >

Can we track **arm postures** with a smartwatch alone?

- What do we need to track?

Elbow
Location

3D Sphere
(DoF: 2)



Can we track **arm postures** with a smartwatch alone?

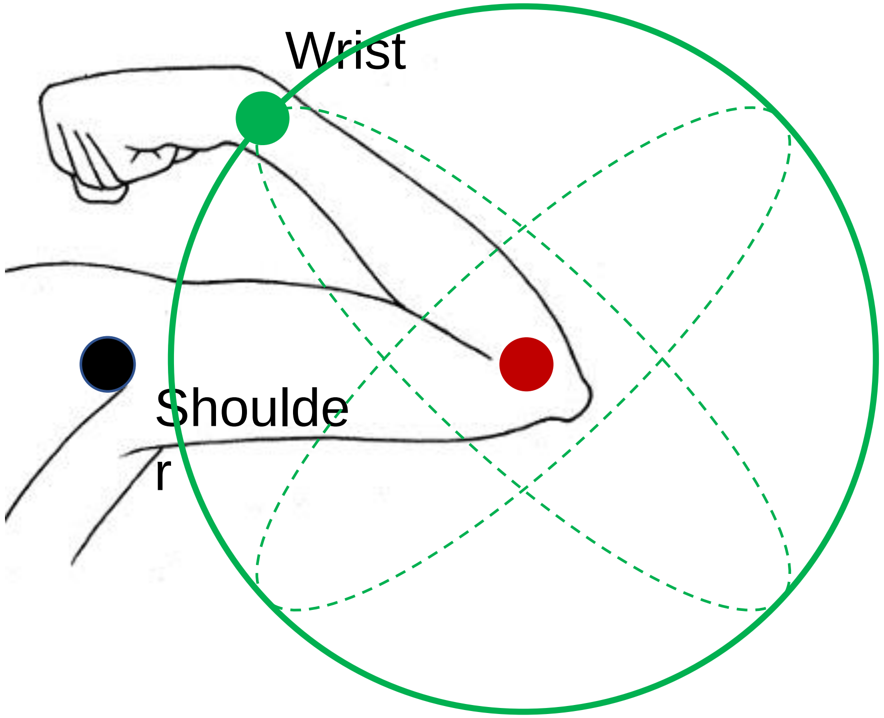
- What do we need to track?

Elbow Location

3D Sphere (DoF: 2)

Wrist Location

3D Sphere (DoF: 2)



Can we track **arm postures** with a smartwatch alone?

- What do we need to track?

Elbow Location

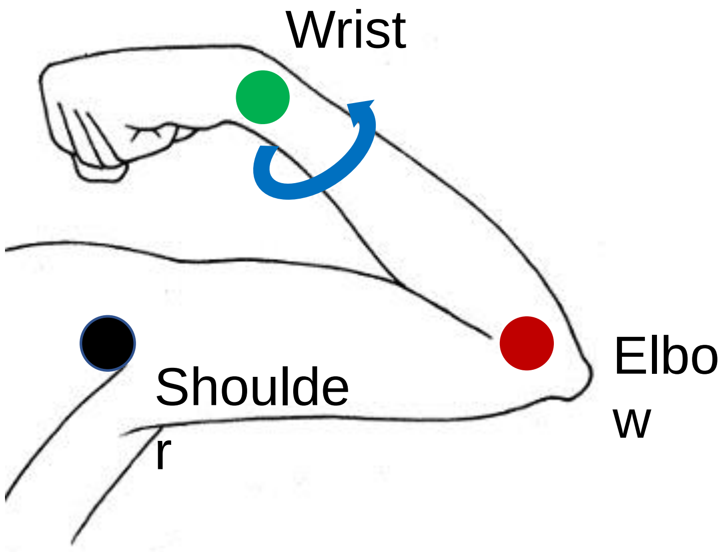
3D Sphere
(DoF: 2)

Wrist Location

3D Sphere
(DoF: 2)

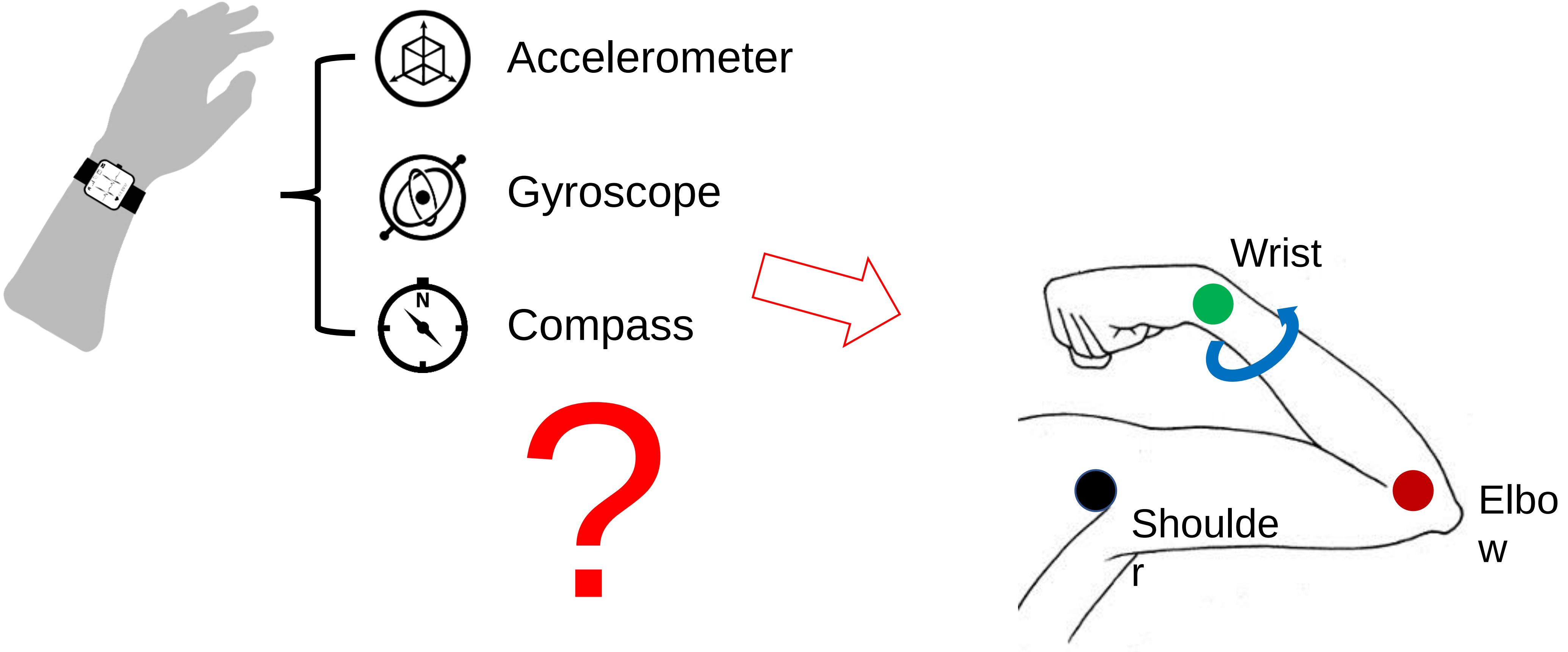
Wrist Rotation

1D Angle
(DoF: 1)



Can we track **arm postures** with a smartwatch alone?

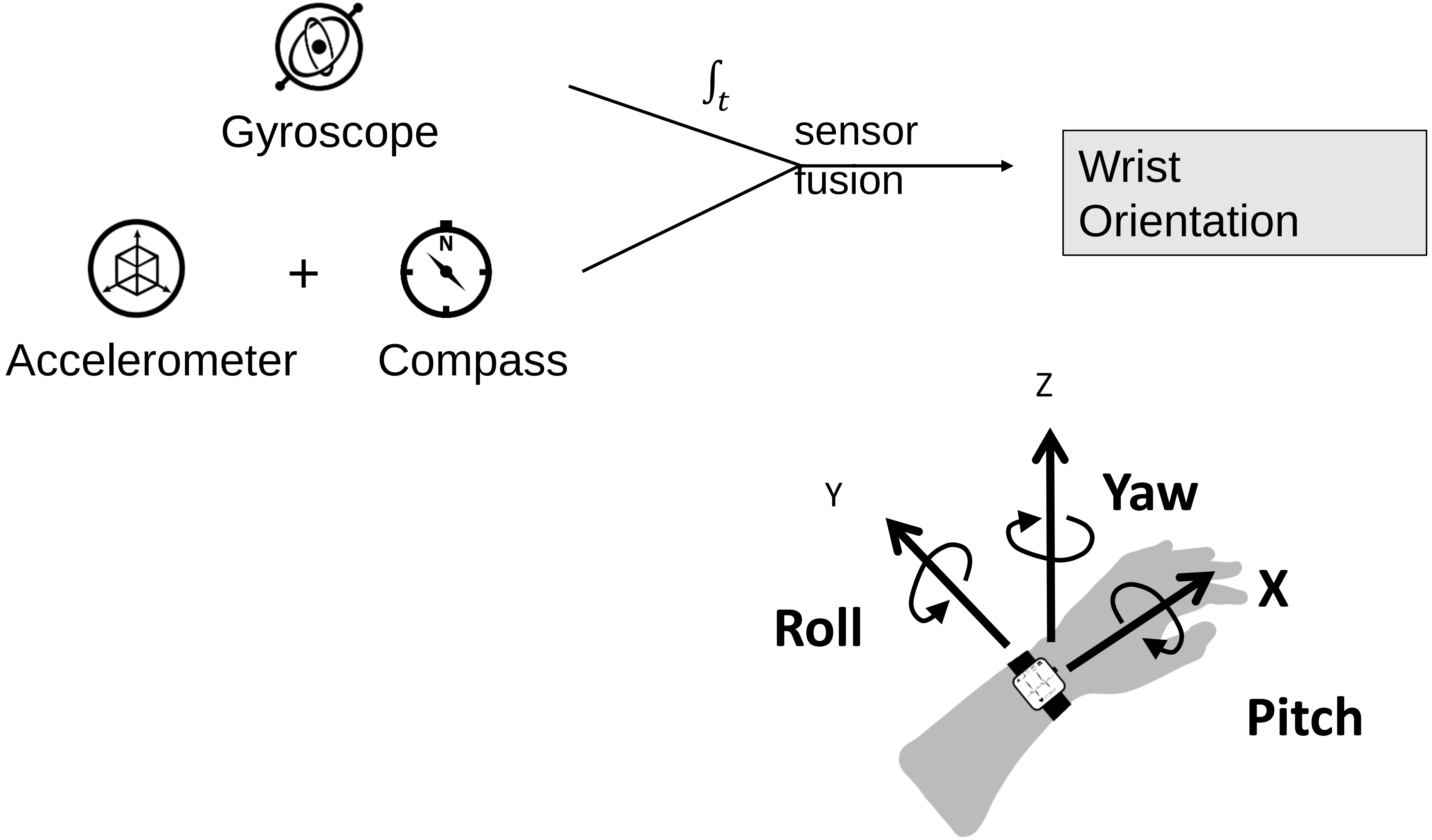
Smartwatch = < Accelerometer, Gyroscope, Compass>



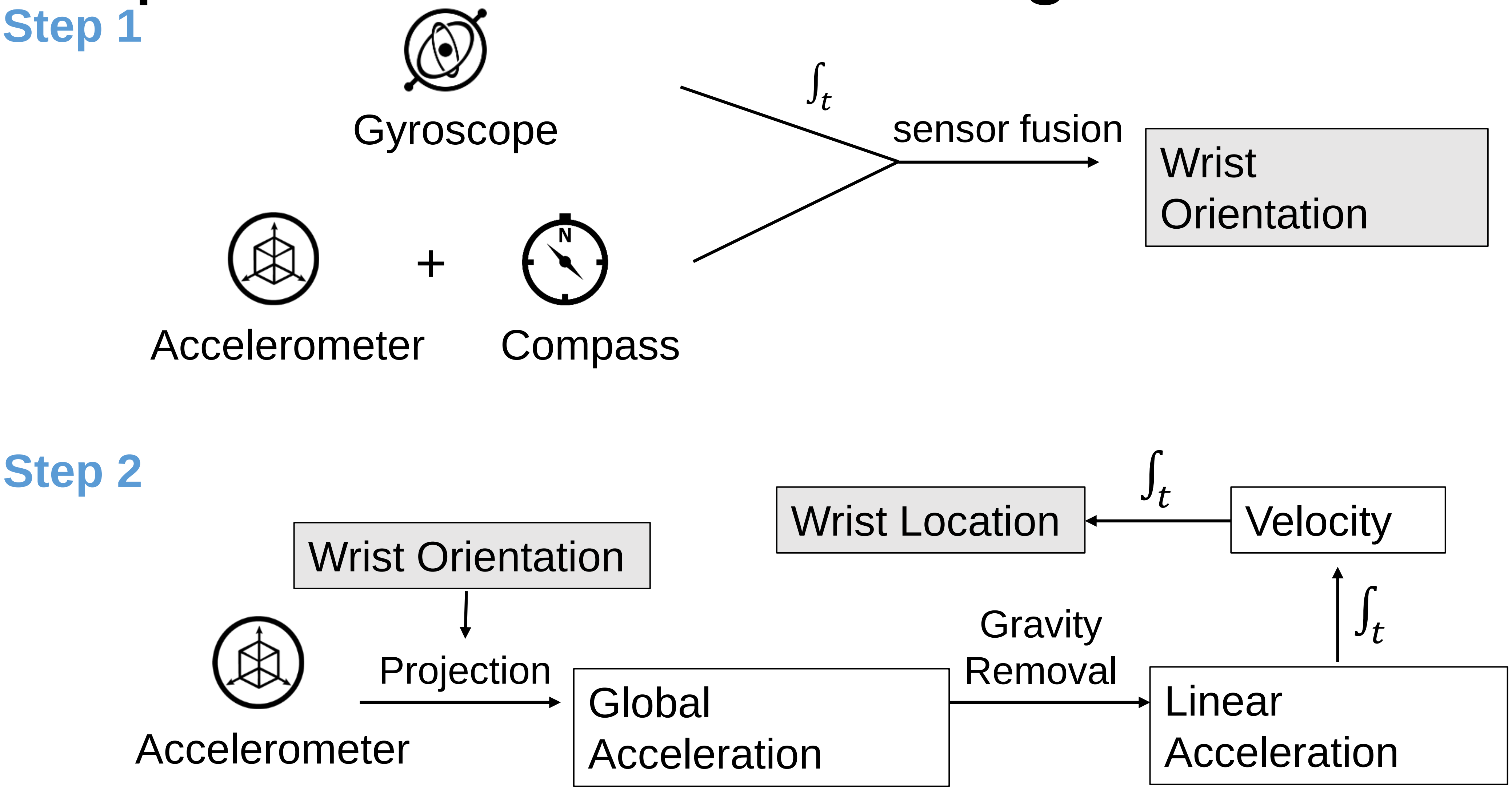
Posture = < Elbow Location, Wrist Location, Wrist Rotation >

Experiment 1: Double Integration

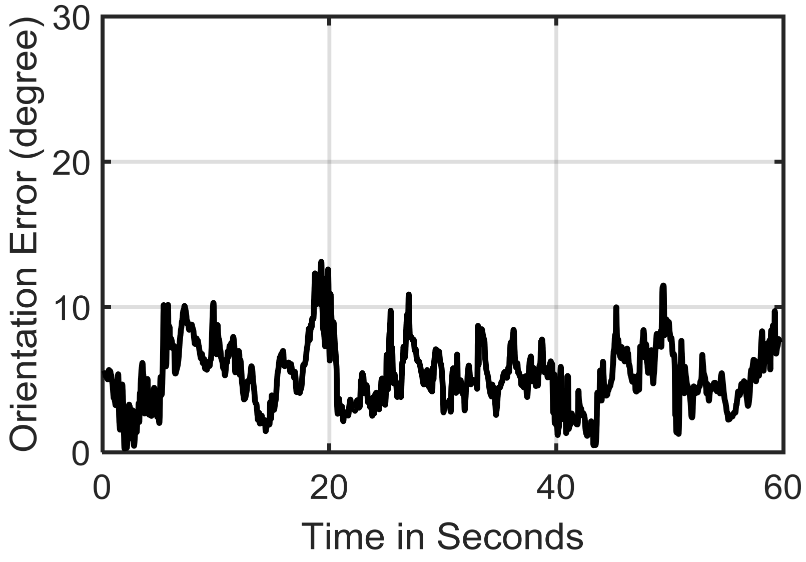
Step 1

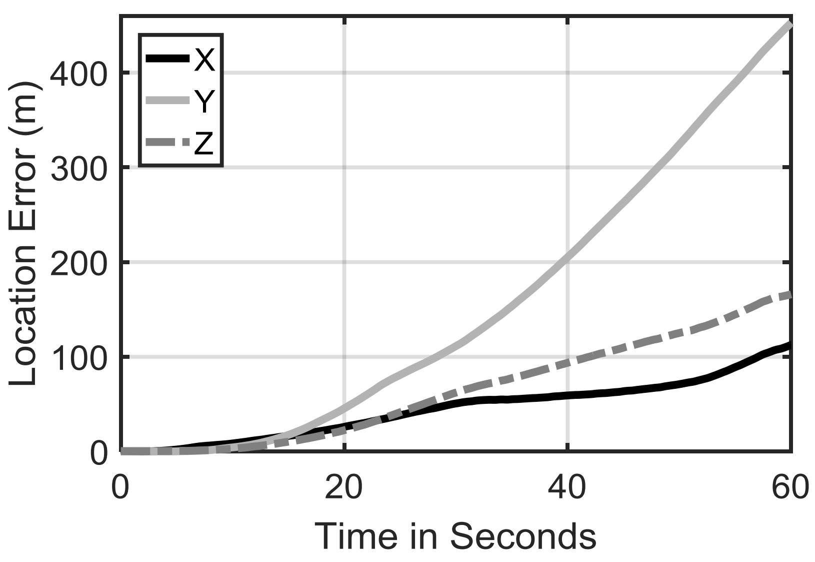


Experiment 1: Double Integration



Experiment: Double Integration





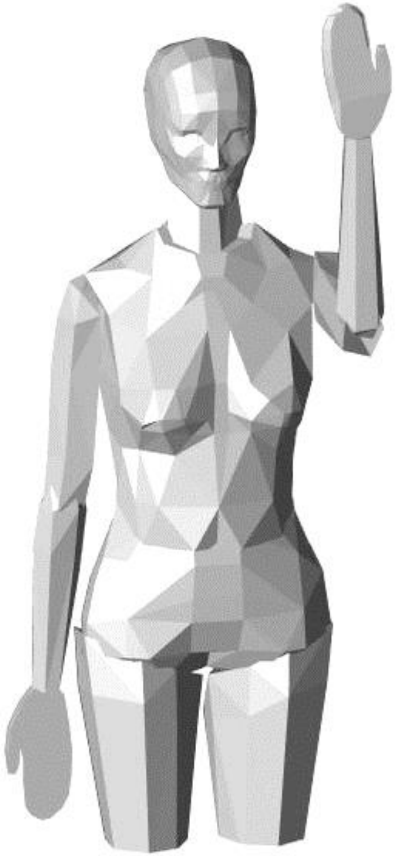
Wrist Orientation

Wrist Location

- Wrist orientation error is okay…
- Wrist location error goes unbounded!

Double integration won't work in unconstrained space

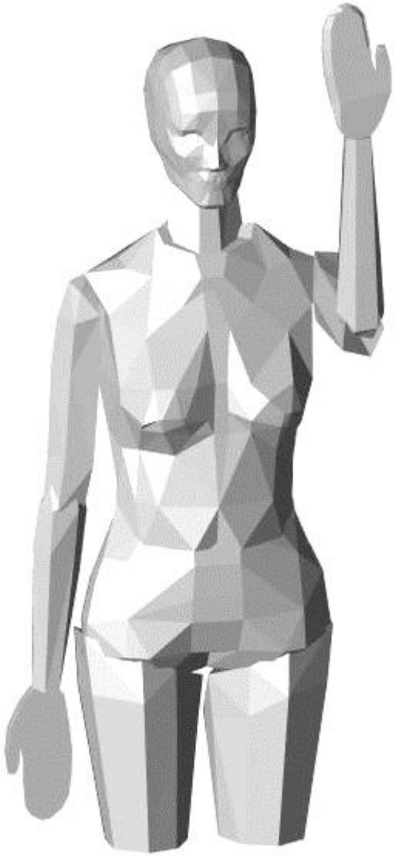
What happens if wrist orientation is fixed…



Forearm pointing upward
Palm facing towards yourself

What happens if wrist orientation is fixed…

Elbow

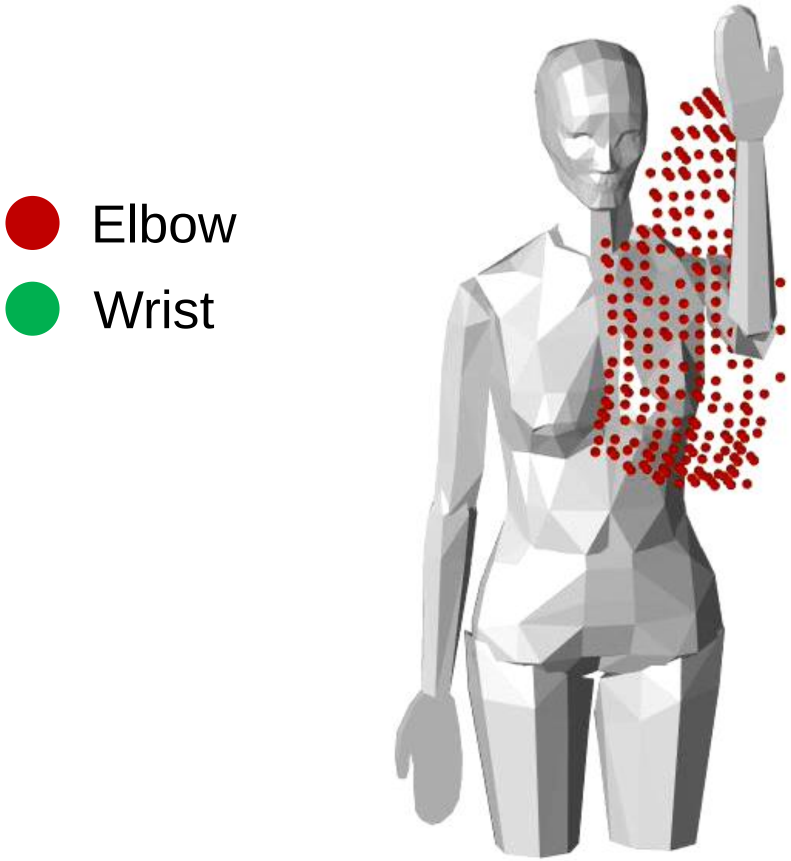


Forearm pointing upward
Palm facing towards yourself

Elbow Point Cloud:
A subset of elbow sphere

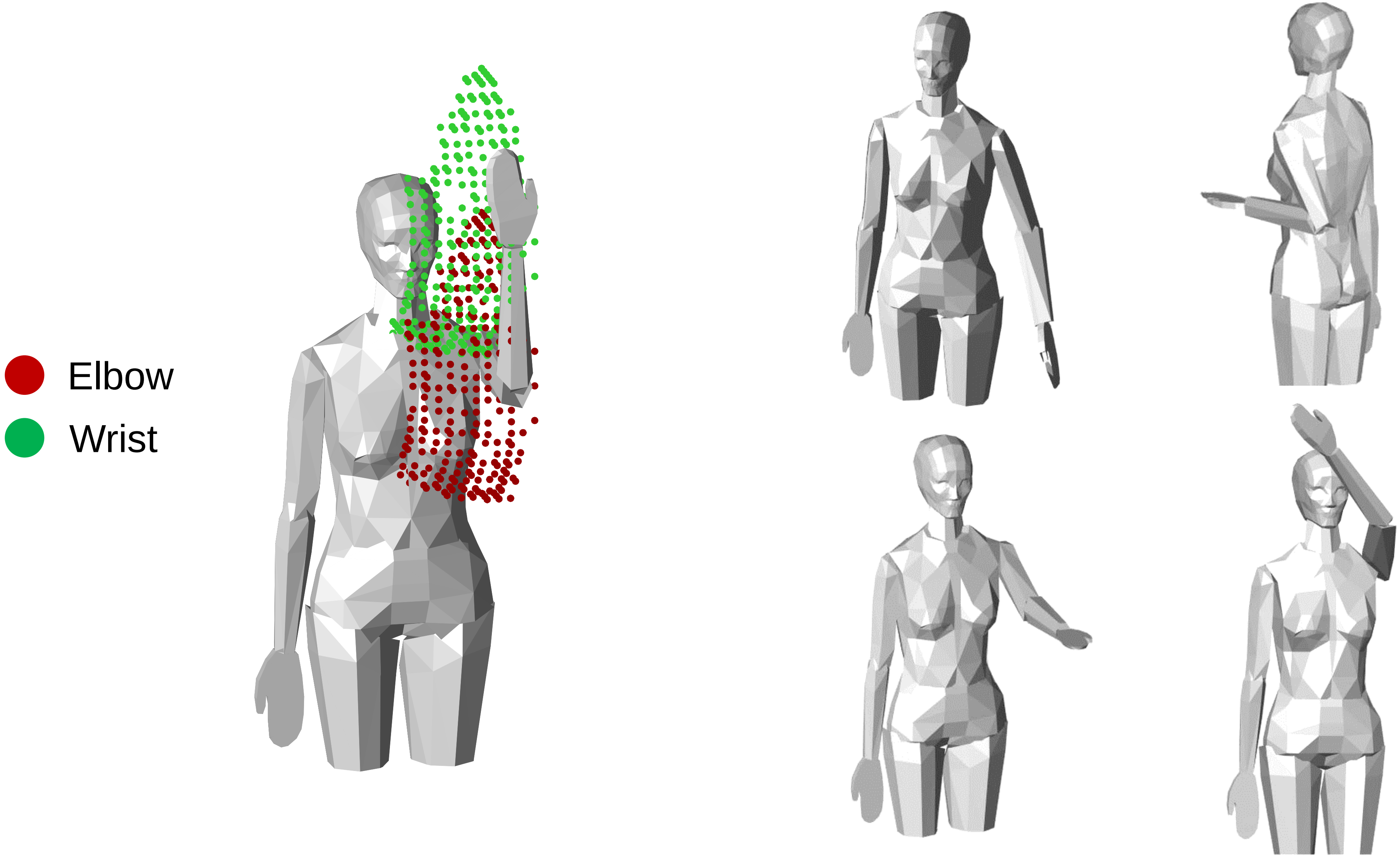
What happens if wrist orientation is fixed…

Forearm pointing upward
Palm facing towards yourself



Elbow Point Cloud:
A subset of elbow sphere

Wrist Point Cloud:
A shift of elbow point cloud, along forearm direction


Elbow
Wrist

What happens if wrist orientation is fixed…

- For a fixed <u>wrist orientation</u>, arm posture space is small!
- This is promising, as we already estimate <u>wrist orientation</u> reasonably well…
- But how can we derive this point cloud for each <u>wrist orientation</u>?

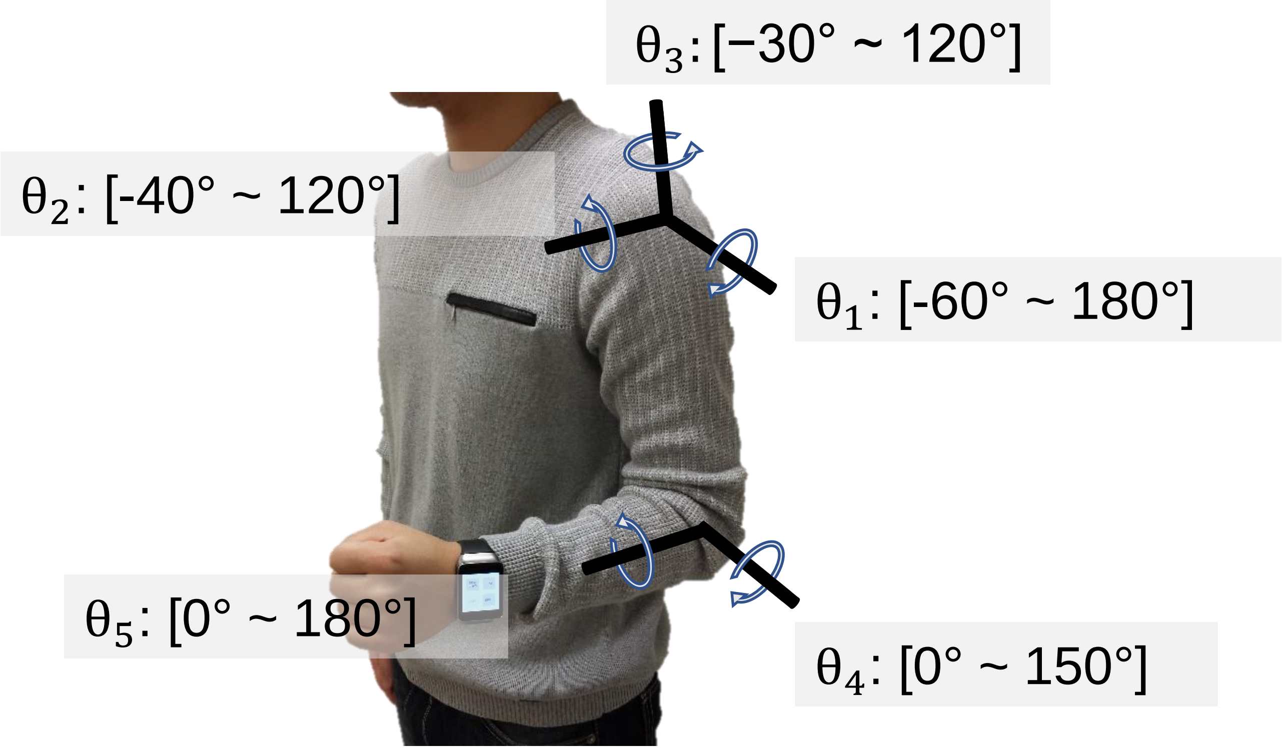
Human Arm Model

Shoulder: $\theta_1, \theta_2, \theta_3$

Human Arm Model

Shoulder: $\theta_1, \theta_2, \theta_3$

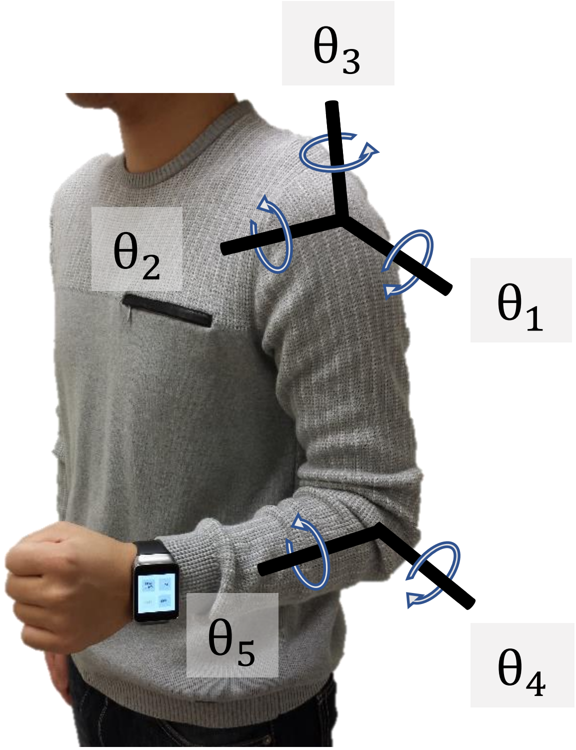
Elbow: θ_4, θ_5



Human Arm Model

Shoulder: $\theta_1, \theta_2, \theta_3$

Elbow: θ_4, θ_5



$$\text{Elbow Location} = f(\theta_1, \theta_2) = \mathrm{l_u} \begin{pmatrix} \cos(\theta_2)\sin(\theta_1) \\ \sin(\theta_2) \\ -\cos(\theta_1)\cos(\theta_2) \end{pmatrix}$$

$$\text{Wrist Location} = g(\theta_1, \theta_2, \theta_3, \theta_4)$$

$$\text{Wrist Orientation} = h(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$$

Orientation – Location Mapping

1-N Mapping for each orientation

Wrist Orientation

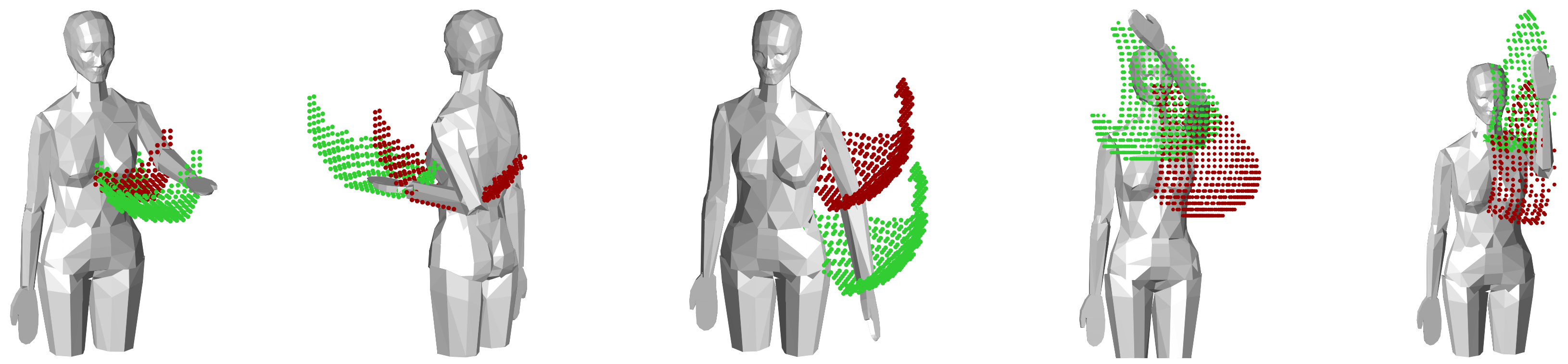
$h(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$

Elbow Location

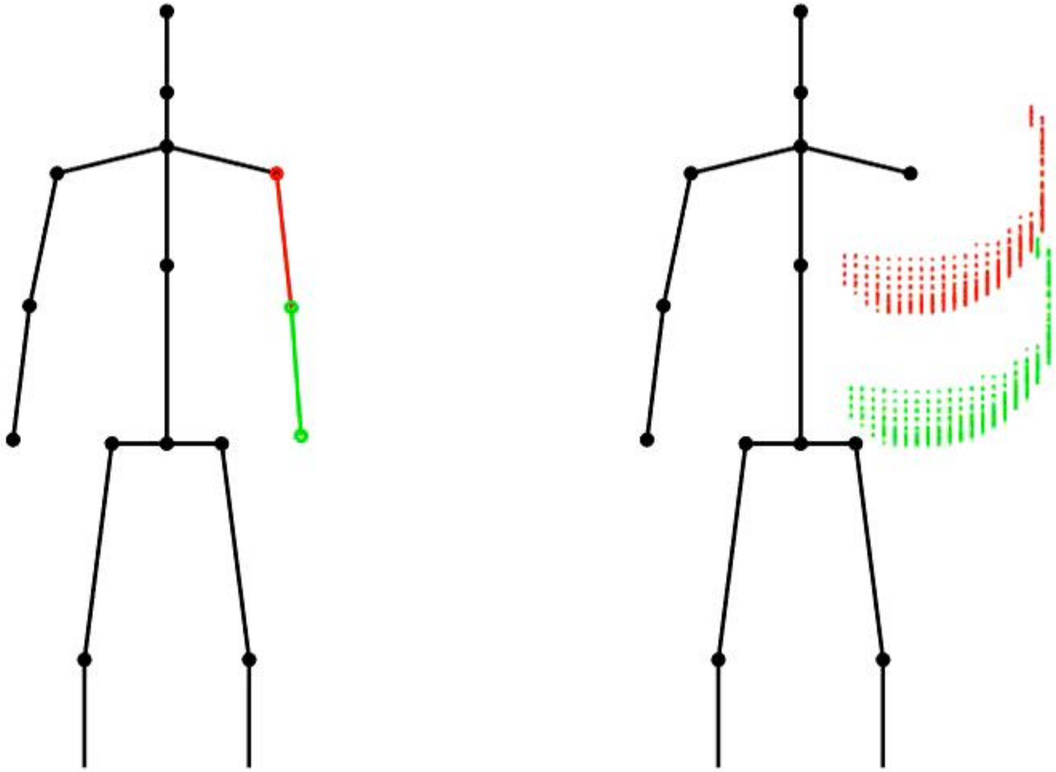
$f(\theta_1, \theta_2)$

Wrist Location

$g(\theta_1, \theta_2, \theta_3, \theta_4)$



Video: Point Cloud Tracking

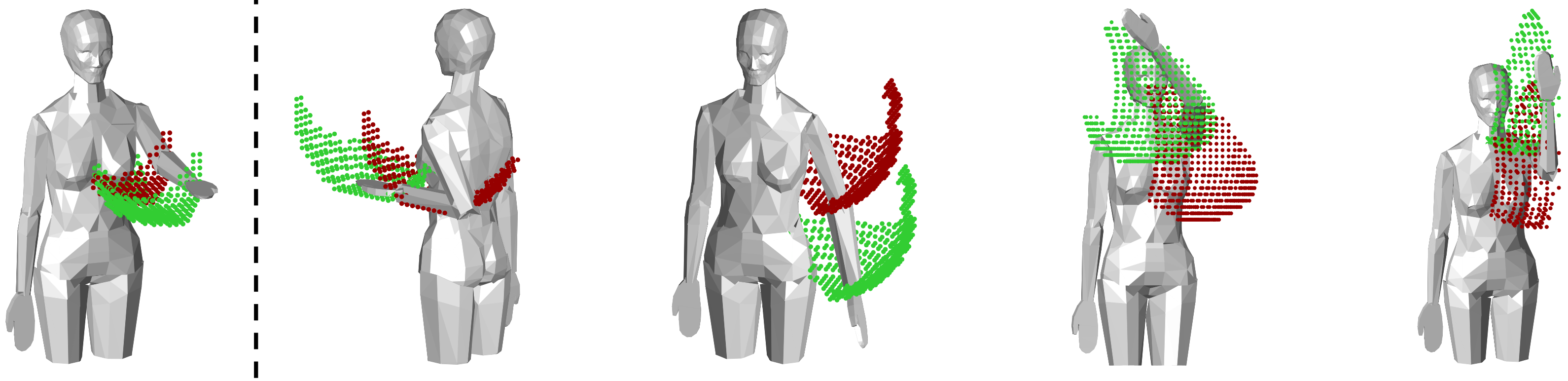


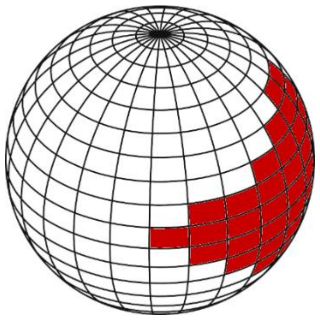
RGB
Video

Kinect
Groundtruth

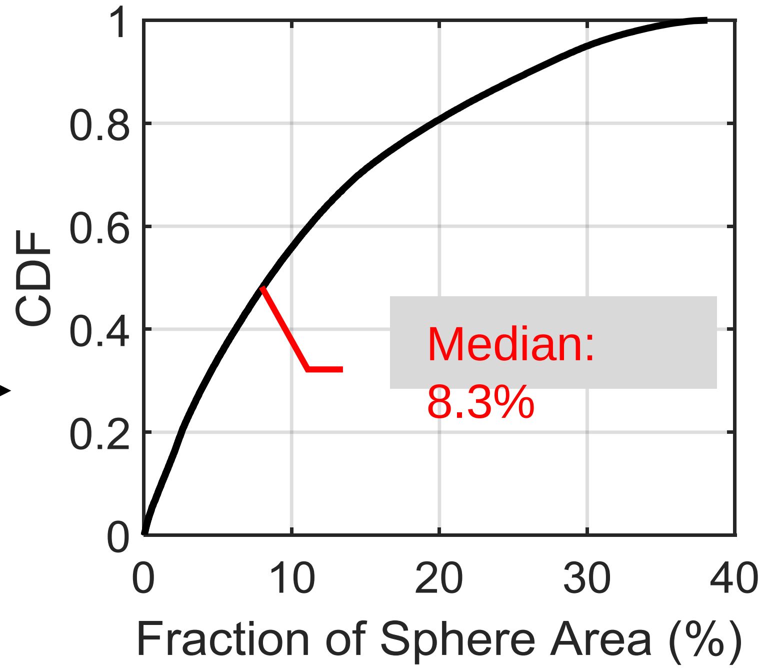
Elbow/Wrist Point
Clouds

How large are the point clouds?

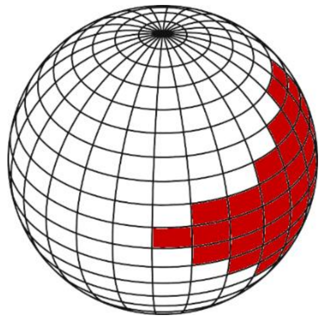




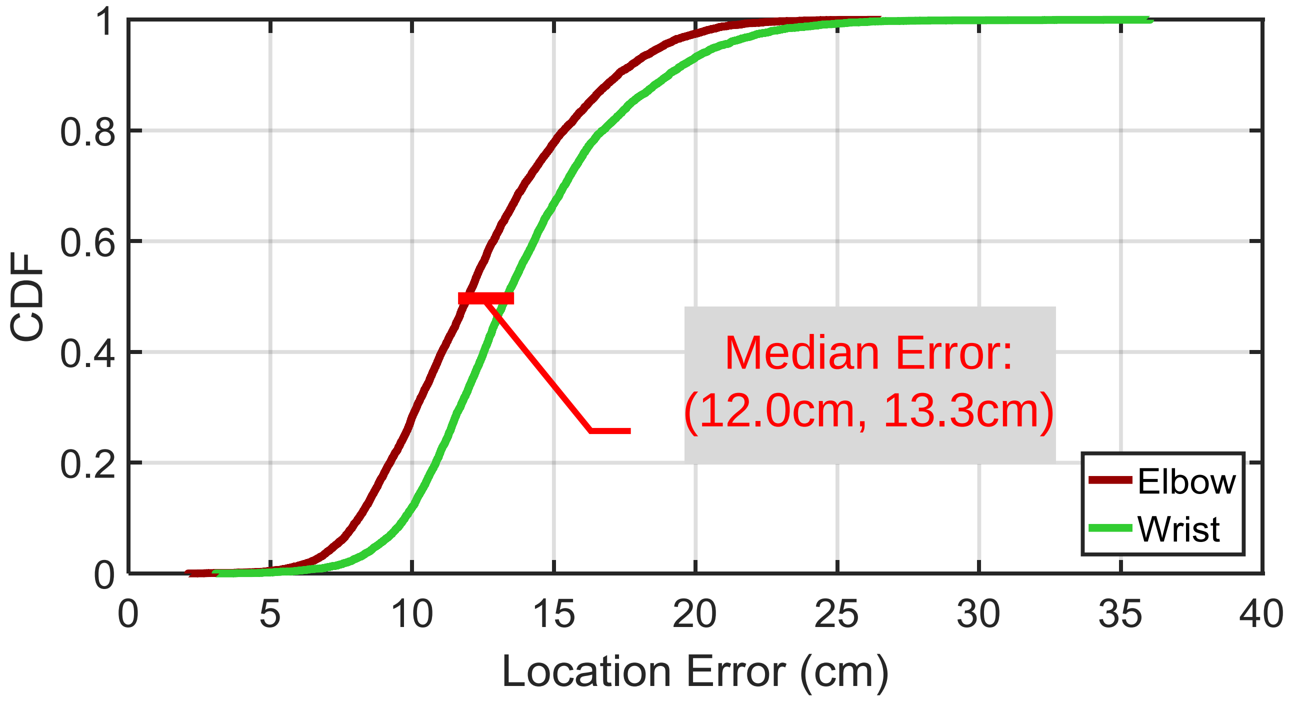
$$\text{Fraction} = \frac{\text{Area of Red}}{\text{Area of Sphere}}$$



How large are the point clouds?



Since they are small, what if we simply take an average?



Video: Write in the Air

Limitation

- Facing direction
  - Need to express arm posture in torso coordinate system

- Tracking on the move
  - Body motion will pollute accelerometer signal

Conclusion

- Tracking arm postures using motion sensors on a smartwatch alone

- <12cm, 13cm> tracking error for <elbow, wrist>