CS60055: Ubiquitous Computing

Location, Gesture and Activity Sensing

Part III: Rethinking the Fundamentals

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

What Fundamentals We Have Learnt So Far?

- Orientation is more reliable than location when measured over the IMU
  - Orientation is computed separately from Accelerometer, Gyroscope and Magnetometer, and a complimentary filter/ Kalman filter helps reduce the noise
  - To compute location, double integration over acceleration is a bad idea! We need to look for alternate methods

What Fundamentals We Have Learnt So Far?

- Orientation is more reliable than location when measured over the IMU
  - Orientation is computed separately from Accelerometer, Gyroscope and Magnetometer, and a complimentary filter/ Kalman filter helps reduce the noise
  - To compute location, double integration over acceleration is a bad idea! We need to look for alternate methods
- However, the basis of orientation calculation is the "**gravity**"
  - **Assumption:** The value and the direction of the gravitational acceleration is a known constant (9.81 m/s^2 towards the downward Z-axis in ECS which we assume as the Global Reference Frame or GRF)
  - For a stationary object with respect to the ECS, we can "observe" the X, Y and Z components of this acceleration (through the accelerometer) and then use the Euler Angle method to compute the orientation

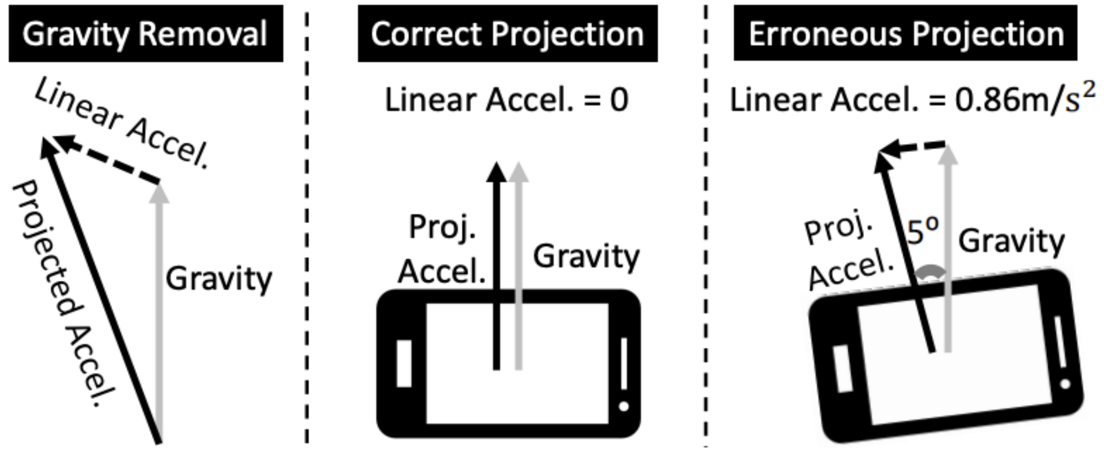
What Fundamentals We Have Learnt So Far?

- When the object is in motion, we can apply one of the two techniques
  - Compute the initial orientation of the object (**when it is static)**, and then integrate the gyroscope data to obtain subsequent orientations
  - Periodically recalibrate the orientation, when the object pauses in between

- Based on the predicted 3D orientation of the object, project the proper acceleration of the object on the target reference frame after subtracting the gravity offset, and then double integrate to get the location in 3D
  - However, double integrate may induce significant amount of noise in the estimation, so we have seen alternate methods

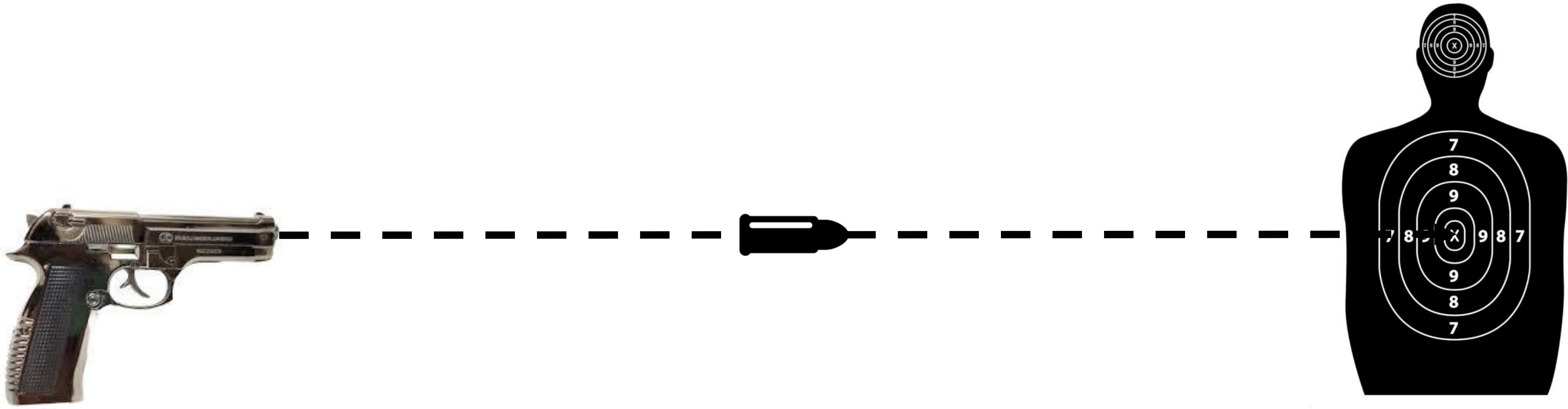
Is Gravity A Good Anchor in Real World?

- **Gravity Pollution**
  - Accelerometer can use gravity only when it is static; otherwise, the object's motion will mix with the gravity measurement
- **Magnetic Interference**
  - From accelerometer, we cannot measure yaw, we need magnetometer
  - However, nearby ferromagnetic materials may pollute magnetometer readings

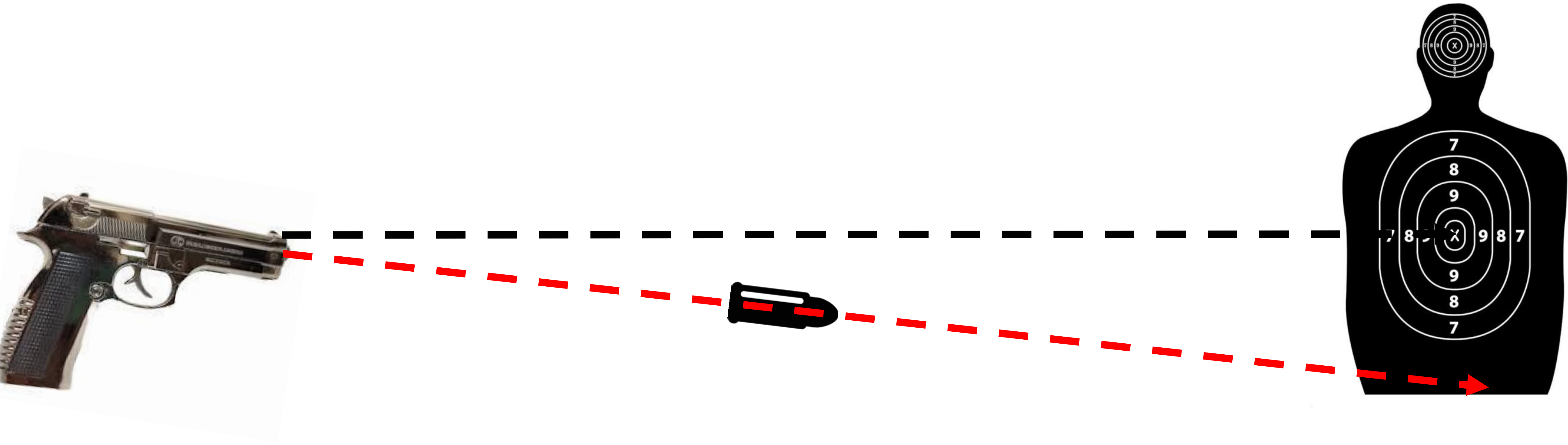


- **Inherent Sensor Noise**
  - Hardware noise is always there in the measurements, any integration (even a single integration over gyroscope) will accumulate noise

Orientation is Crucial for Location Measurement of a Moving Object



Orientation is Crucial for Location Measurement of a Moving Object



When Can We Use Gravity as an Anchor

- Gravity is unpolluted when
  - The object is static
  - The object has pure rotational motion

- Find out the instances when the gravity is unpolluted and recalibrate the GRF to LRF mapping during those instances
  - Check the A^3 paper*

- However, such instances are rare in many of the real scenarios
  - Say, the person is running

*P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in ACM MobiCom 2014

When Can We Use Gravity as an Anchor

- Gravity is unpolluted when
  - The object is static
  - The object has pure rotational motion

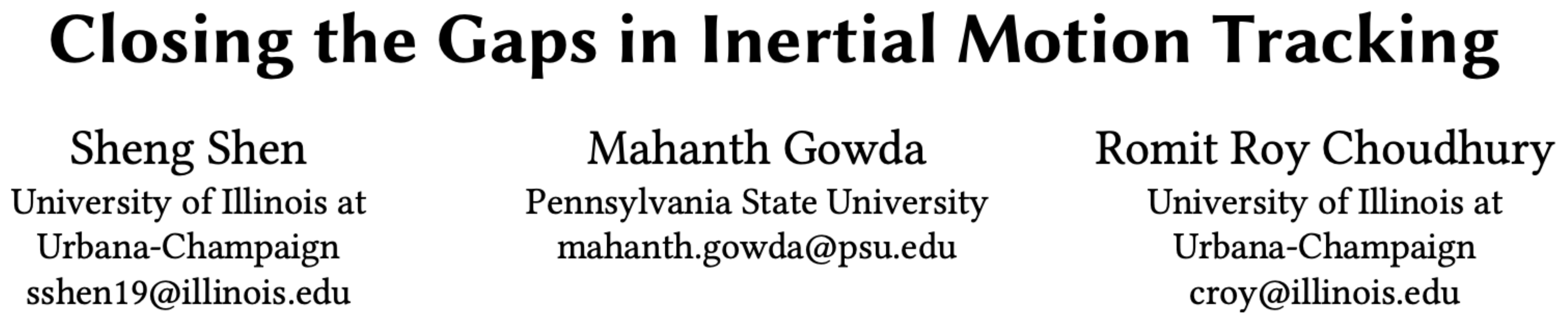
- Find out the instances when the gravity is unpolluted and recalibrate the GRF to LRF mapping during those instances
  - Check the A^3 paper*

- However, such instance
  - Say, the person is runni

Can we figure out a better anchor than gravity?

*P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in ACM MobiCom 2014

Towards a Better Anchor for Motion Tracking using IMU



Closing the Gaps in Inertial Motion Tracking

Sheng Shen
University of Illinois at Urbana-Champaign
sshen19@illinois.edu

Mahanth Gowda
Pennsylvania State University
mahanth.gowda@psu.edu

Romit Roy Choudhury
University of Illinois at Urbana-Champaign
croy@illinois.edu

ACM MobiCom 2018

Orientation Estimation

- Can you think of any other feature which is fixed in the global reference frame and can be measured using the IMU?

Orientation Estimation

- Can you think of any other feature which is fixed in the global reference frame and can be measured using the IMU?
  - **3D Magnetic North Vector**

- **Pros:** The magnetometer reading does not vary with linear acceleration

- **Cons**: The intensity and direction may vary across locations

Orientation Estimation

- Can you think of any other feature which is fixed in the global reference frame and can be measured using the IMU?
  - **3D Magnetic North Vector**
- **Pros:** The magnetometer reading does not vary with linear acceleration
- **Cons**: The intensity and direction may vary across locations
- **Key idea**: The object starts from a static moment, utilizes the unpolluted gravity to precisely estimate the 3D North vector, and thereafter uses the North vector as the anchor for orientation

The Initialization

- Accelerometer
  - Offers unpolluted gravity during initialization (when the object is static)
  - Determine the vertical tilt of the object (*roll*, *pitch*)

- Magnetometer
  - Measures the 3D magnetic vector in the local reference frame (LRF)
  - Project the computed magnetic vector to the horizontal plane to estimate the object's heading (*yaw*)

- Combine these two to determine the 3D orientation of the object in the GRF

Towards the New Anchor

- Once we know the initial 3D orientation ($O(t_0)$) of the object,
  - Project the current magnetometer's local measurement ($N^L(t_0)$)onto the GRF
  - Leads the global 3D magnetic north vector N^G
- Gives the anchor we need,

$$\mathbf{N}^G = \boldsymbol{O}(t_0)\mathbf{N}^L(t_0)$$

Towards the New Anchor

- Once we know the initial 3D orientation ($O(t_0)$) of the object,
  - Project the current magnetometer's local measurement ($N^L(t_0)$)onto the GRF
  - Leads the global 3D magnetic north vector N^G
- Gives the anchor we need,

$$\mathbf{N}^G = \mathbf{O}(t_0)\mathbf{N}^L(t_0)$$

- As the object starts moving, the magnetometer tracks N^G in its LRF
  - Provides 2 DoF of global orientation
- In parallel, gyroscope tracks all 3 DoFs of changing orientation
  - But the values can be polluted due to the motion of the object
- **How can we combine the 5 DoFs to accurately measure the 3-DoF orientation?**

Magnetometer + Gyroscope Fusion

- Gyroscope drift accumulates over time -> Use magnetometer to recalibrate
    - Gyroscope measures changes in 3-DoF orientation
    - Magnetometer measures 2-DoF of global orientation

Magnetometer + Gyroscope Fusion

- Gyroscope drift accumulates over time -> Use magnetometer to recalibrate
  - Gyroscope measures changes in 3-DoF orientation
  - Magnetometer measures 2-DoF of global orientation

- Assume that at current time *t*, the actual orientation of the object is *O(t)*
  - *O(t)* is a 3x3 rotation matrix that rotates the object's GRF to LRF
- Since, the magnetometer measure the global 3D magnetic North vector N^G in its LRF,

$$\mathbf{N}^L(t) = \mathbf{O}^{-1}(t)\mathbf{N}^G$$

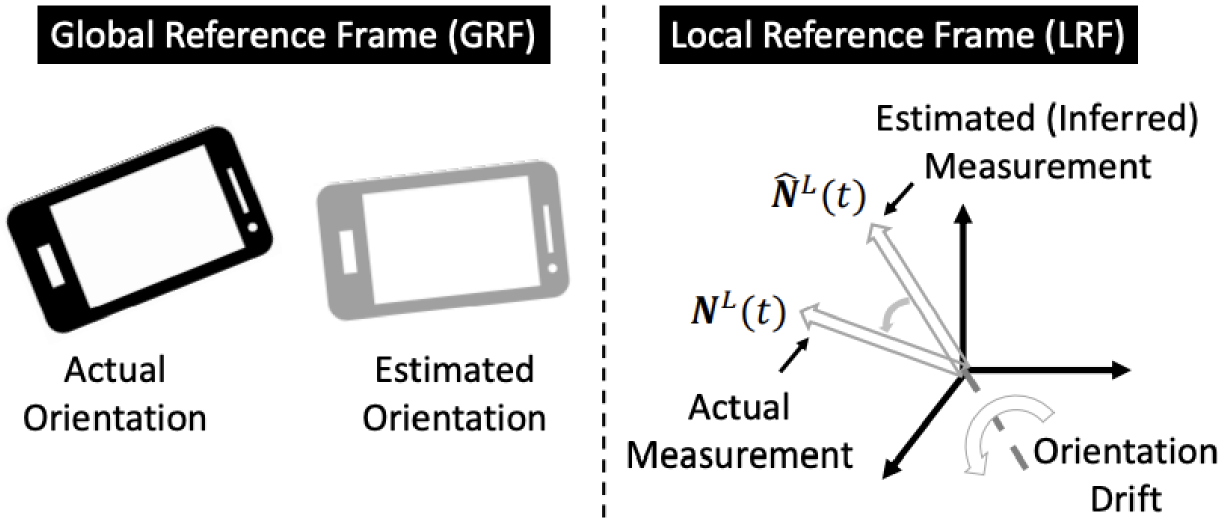
  - Note that we do not know the actual O(t), and the current measure from gyroscope *O'(t)* is erroneous.

Magnetometer + Gyroscope Fusion

- However, we can check this error by computing the $N'^{L}(t)$ and comparing it with $N^{L}(t)$,

$$N'^{L}(t) = O'^{-1}(t)N^{G}$$

- The difference between the actual magnetometer measurement ($N^{L}(t)$) and the computed measurement ($N'^{L}(t)$) reveals the drift in the orientation



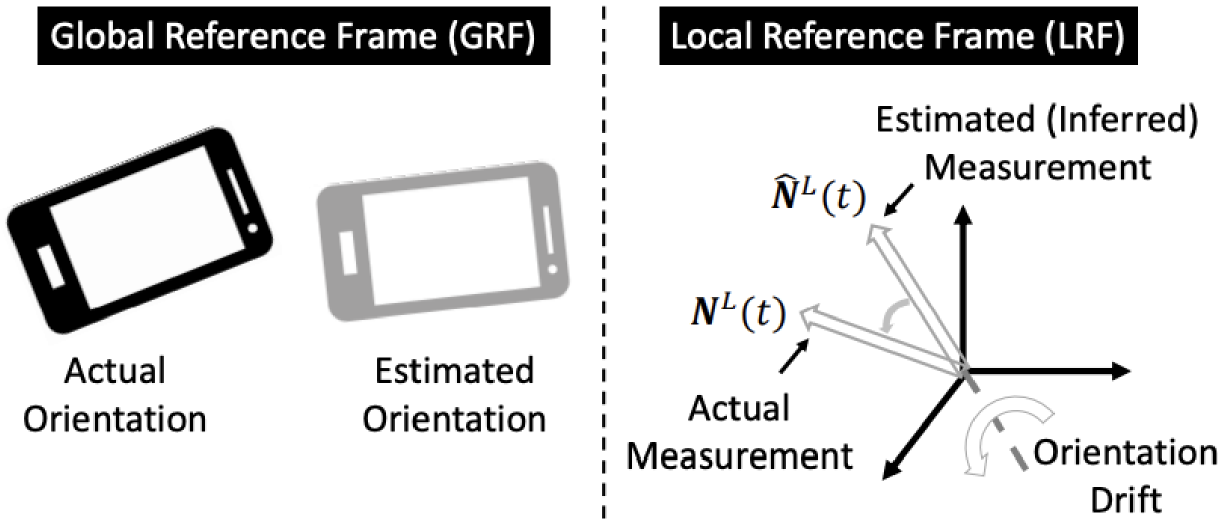
Magnetometer + Gyroscope Fusion

- However, we can check this error by computing the $N'^L(t)$ and comparing it with $N^L(t)$,

$N'^L(t) =$

- The difference between th d the computed measureme

How do we update orientation estimation using this disparity?



Updating Orientation

- **Key observation:** Noise properties of magnetometer and gyroscope are different
  - Gyroscopes observe a long-term integration drift (measures angular velocity in its LRF and compute the orientation based on calculating the angular displacement from the measured angular velocity)
    - **However, it is quite accurate in short-term**
  - Magnetometers have short-term noise from environmental fluctuations and sensor imperfection
    - **However, do not drift in the long run as they always measure the same global North vector and thus no integration is needed.**

Updating Orientation

- To fuse the best of both sensors, a *complimentary filter* is used
  - Computes the weighted combination of the two

- Gyroscope drifts less in the short term
  - Extract the high frequency components from the gyroscope

- Magnetometer is stable in long run
  - Extract the low frequency components from the magnetometer

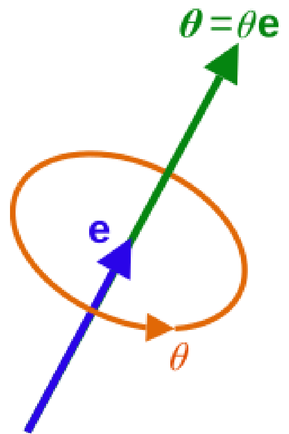
Assign larger weight to the gyroscope

Updating Orientation

- To fuse the best of both sensors, a *complimentary filter* is used
  - Computes the weighted combination of the two
- Gyroscope drifts less in the short term
  - Extract the high frequency components from the gyroscope
- Magnetometer is stable in long run
  - Extract the low frequency components from the magnetometer

$\}$ **Assign larger weight to the gyroscope**

MUSE uses axis-angle rotations rather than Euler angle rotation! Why?



Gimbal Lock

- A scenario when you lose one degree of freedom
  - When the axis of two of the gimbals become parallel
  - The third axis cannot make any further progress
- Depends on how you are triggering the Euler angles rotation
  - Need a fourth gimbal to manually break the lock
- Popular incident: Apollo 11 moon mission
  - The spacecraft had been manually operated to come out of the lock, using starts as reference
  - Mike Collins: "*How about sending me a fourth gimbal for Christmas?*"

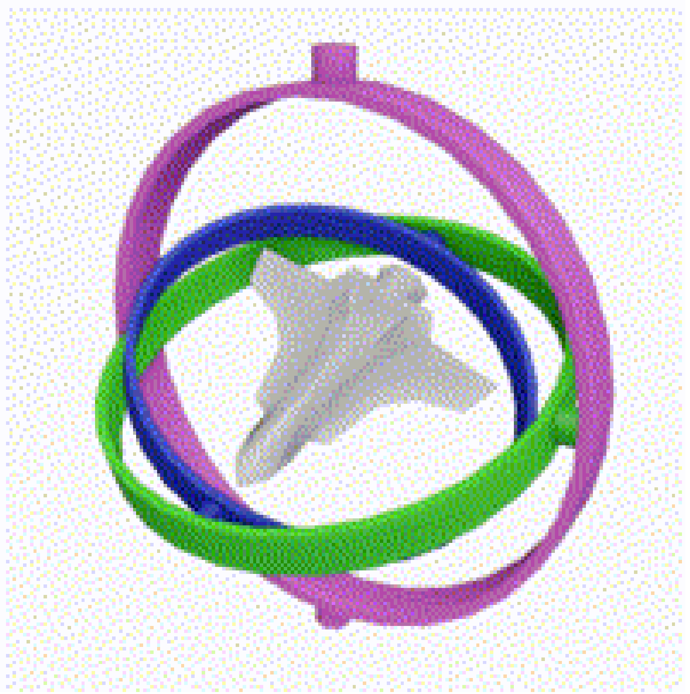


Image Source: Wikipedia

Gimbal Lock: An Example

- Direction Cosine Matrix for 3-2-1 Euler angles:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Let β = π/2, then

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gimbal Lock: An Example

- Performing matrix multiplication and applying the trigonometry formula:

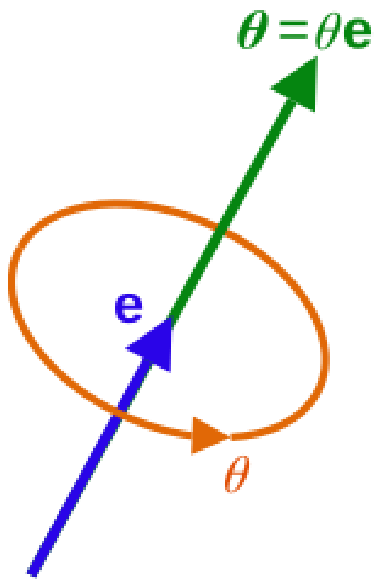
$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}$$

- Changing the values of α and γ in the above matrix has the same effects
  - o The rotation angle (α+γ) changes
  - o The rotation axis remains in the Z direction (the first row and the last column remains fixed with all values of α and γ)
- MUSE uses axis angles rather than Euler angles to avoid such Gimbal locaks.

Why Axis Angle Does Not Suffer from Gimbal Lock?

- Represented as an ordered pair

$$(\text{axis}, \text{angle}) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right)$$



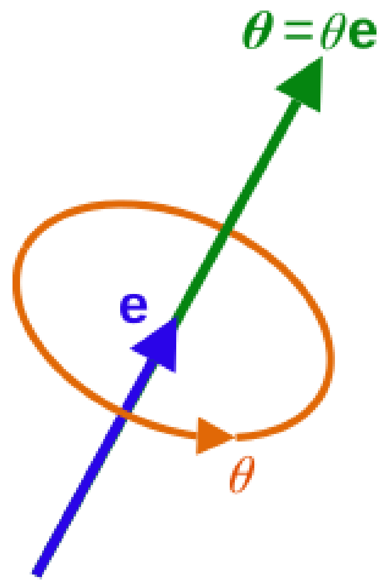
- Say, a π/2 left rotation while standing: $\left(\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \frac{-\pi}{2} \right)$

- Equivalent Euler angle representation: $\begin{bmatrix} 0 \\ 0 \\ \frac{\pi}{2} \end{bmatrix}$

Why Axis Angle Does Not Suffer from Gimbal Lock?

- Represented as an ordered pair

$$(\text{axis}, \text{angle}) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right)$$

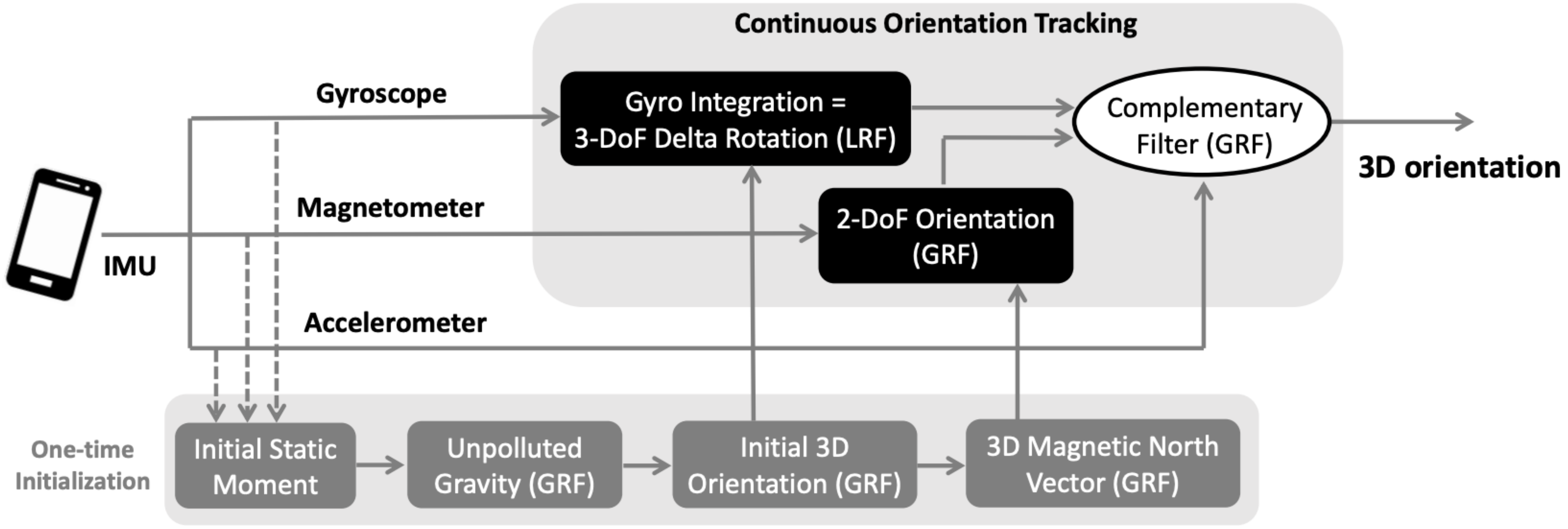


- Say, a π/2 left rotation while standing: $\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} \frac{-\pi}{\ } \right)$

- Equivalent Euler angle representation:

Axis angle represents the rotations with respect to each individual axes, whereas Euler angle provides rotation with respect to each previous rotations, make them locked when no movement possible with respect to the previous rotation

MUSE Processing Pipeline


Continuous Orientation Tracking
Gyroscope
Gyro Integration = 3-DoF Delta Rotation (LRF)
Complementary Filter (GRF)
3D orientation
Magnetometer
IMU
2-DoF Orientation (GRF)
Accelerometer
One-time Initialization
Initial Static Moment
Unpolluted Gravity (GRF)
Initial 3D Orientation (GRF)
3D Magnetic North Vector (GRF)

Some Interesting Issues

- MUSE calibrates 2 DoF using the magnetic north poles; however, it cannot correct the error for the other DoF when in motion.
    - Opportunistically detects static or slow-moving instances to recalibrate that DoF

Some Interesting Issues

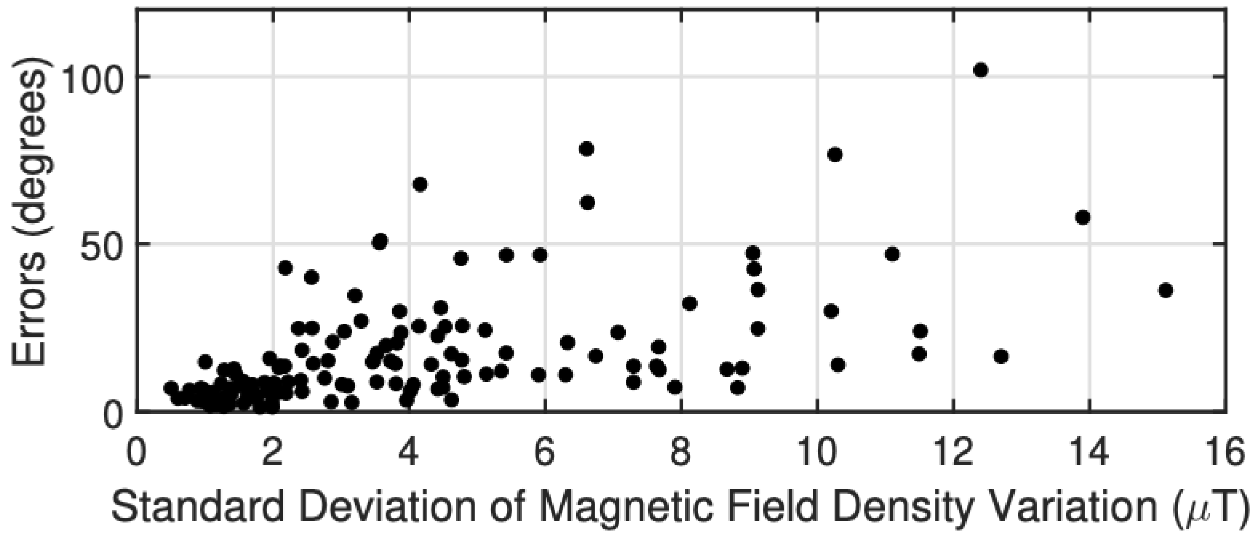
- MUSE calibrates 2 DoF using the magnetic north poles; however, it cannot correct the error for the other DoF when in motion.
    - Opportunistically detects static or slow-moving instances to recalibrate that DoF

- **When does MUSE fail?**

Some Interesting Issues

- MUSE calibrates 2 DoF using the magnetic north poles; however, it cannot correct the error for the other DoF when in motion.
  - Opportunistically detects static or slow-moving instances to recalibrate that DoF

- **When does MUSE fail?**
  - Rotation is only across the 3rd DoF (the global anchor – magnetic north axis)
  - No opportunity for initialization
  - Strong magnetic interference



Location Estimation

- ArmTrak provided a method for estimating location from orientation using 5 DoFs of arm's kinematics
  - What can be the possible issue with such a system?

Location Estimation

- ArmTrak provided a method for estimating location from orientation using 5 DoFs of arm's kinematics
  - What can be the possible issue with such a system?
- Orientation estimation with each DoF introduces divergence
  - The error margin may get expanded due to the divergence with each of the 5 DoFs
- In practical sense, orientation and locations are coupled with each other
  - There would be a location change with each of the orientation change (unless the orientation change is solely on one axis)
  - **Can we develop a joint orientation and location tracking approach?**

The Core Idea

- MUSE estimates the orientation of the IMU sensors
  - Among the 3 DoF orientation, two of them will be drift free
  - We need to estimate the remaining 1 DoF orientation and 2 DoFs of location from the 3 DoF accelerometer

The Core Idea

- MUSE estimates the orientation of the IMU sensors
  - Among the 3 DoF orientation, two of them will be drift free
  - We need to estimate the remaining 1 DoF orientation and 2 DoFs of location from the 3 DoF accelerometer

What sequence of watch orientations and locations will cause gravity and linear motion to (vectorially) add up in a way that matches the accelerometer measurements?

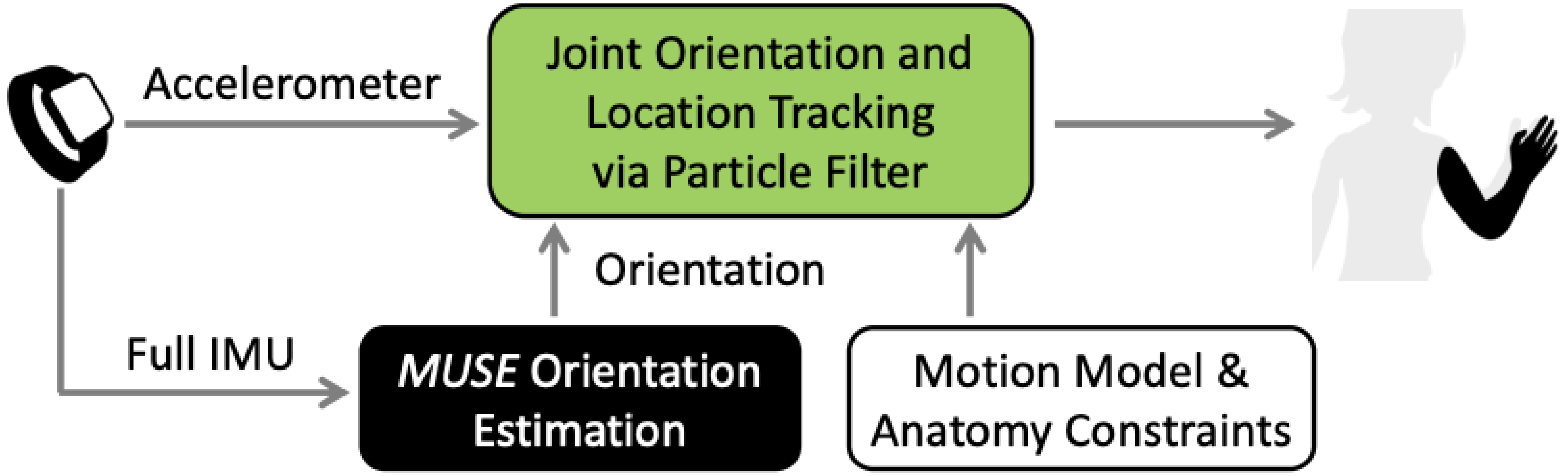
The Core Idea

- MUSE estimates the orientation of the IMU sensors
  - Among the 3 DoF orientation, two of them will be drift free
  - We need to estimate the remaining 1 DoF orientation and 2 DoFs of location from the 3 DoF accelerometer

What sequence of watch orientations and locations will cause gravity and linear motion to (vectorially) add up in a way that matches the accelerometer measurements?

Use a particle filtering-based approach to combine them together for a joint estimation of location and orientation

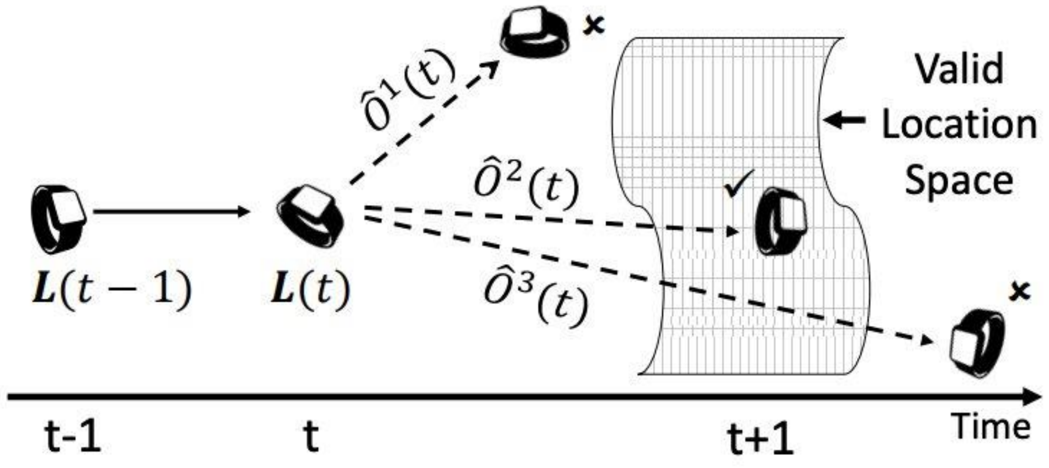
MUSE Location Estimation


Accelerometer
Joint Orientation and Location Tracking via Particle Filter
Orientation
Full IMU
MUSE Orientation Estimation
Motion Model & Anatomy Constraints

Joint Tracking through Particle Filter

- We have certain uncertainty in the orientation estimation
  - Possible drift in the 1-DoF towards magnetic north
  - However, different orientation may project to different locations

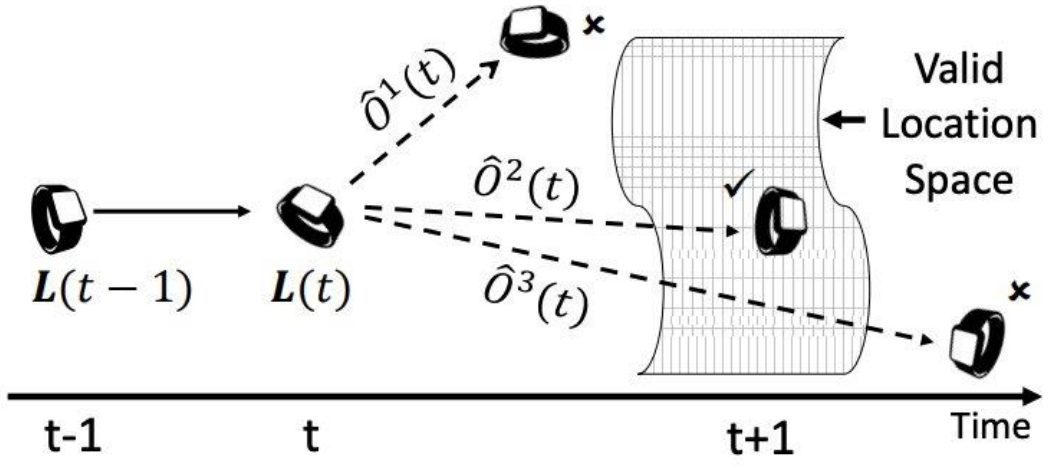
- The correct estimation of the orientation will project the location within the valid location space



- **Challenge:** In reality, the past locations are not knows and needs to be estimated
  - The valid location space depends on the smartwatch orientation – a nonlinear system

Joint Tracking through Particle Filter

- Model each possible *<orientation, location>* trajectory as a particle in the particle filter
  - Particles with wrong orientation will exhibit diminishing weights and will eventually get removed
  - Particles that are left behind are likely to be the good joint estimators for both location and orientation



Statistical Filters – A Primer

- Use a series of observed measurements over time to predict unknown variables, while considering statistical noises and other measurement inaccuracies
- **Kalman Filter**
  - a.k.a *Linear quadratic estimation* (estimates state of a linear dynamic system)
  - The filter uses a system's dynamic model (such as, the laws of motion), known control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state)
    - Expected to be better than the estimate obtained by using only a single measurement.
  - Estimate a joint probability distribution over the variables for each time frame
  - Constructed as a mean squared error minimizer
  - A recursive filter (uses the output to its input)
  - Various variations exist: Simple Kalman Filter, Kalman-Bucy Filter, Information filter, …

Kalman Filtering

- Assumes that the true state at time k evolves from the state (k-1)

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

  - F_k is the state transition model applied to the previous state x_{k-1}
  - B_k is the control-input model applied to the control vector u_k
  - w_k is the process noise, assumed to be drawn from a zero-mean multivariate normal distribution with covariance Q_k (which also needs an estimate from the observations)

- At time k, an observation of the true state is estimated as,

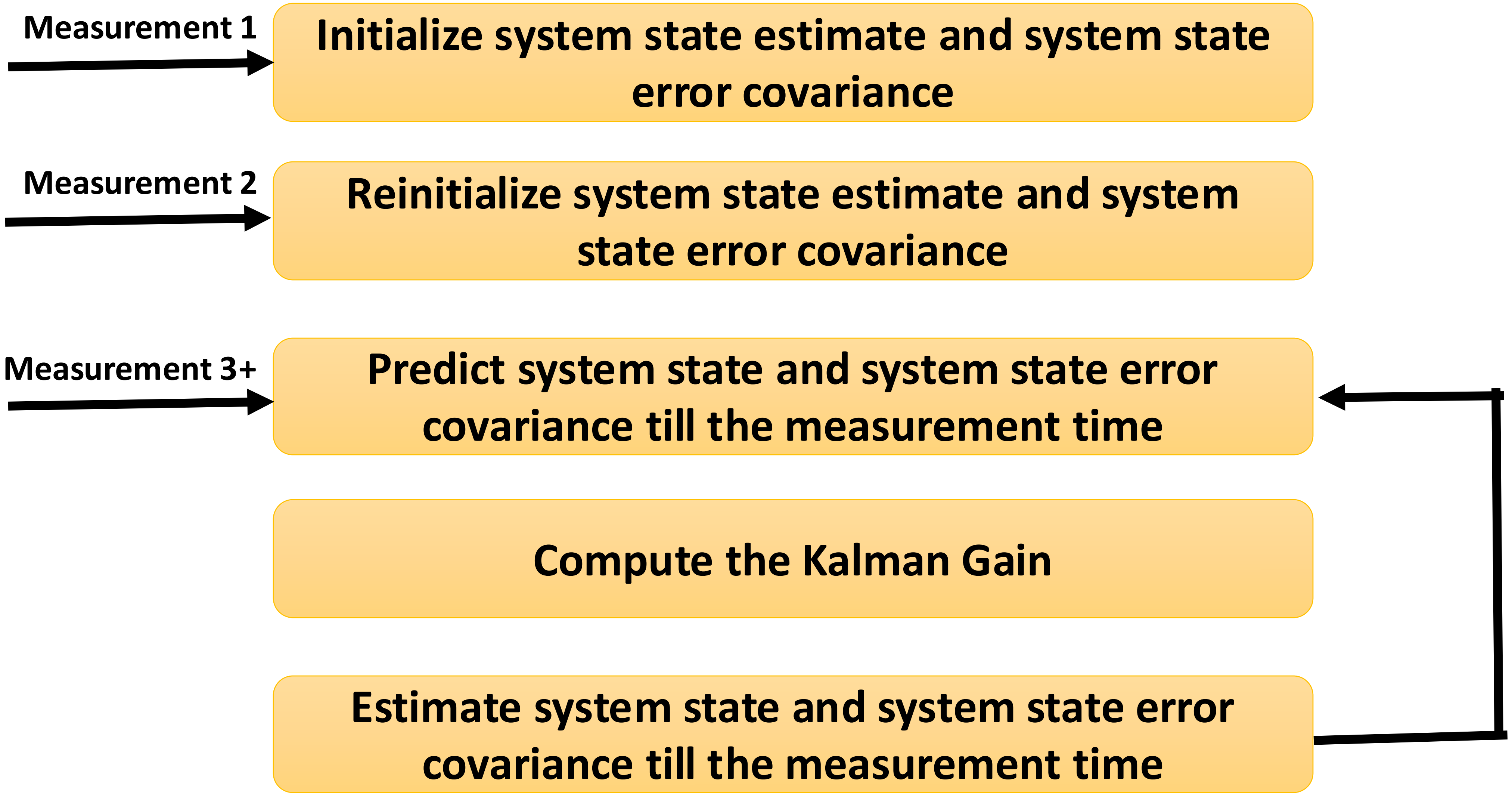
$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

  - H_k is the observation model, maps the true state space to the observed state space
  - v_k is the observation noise, assumed to be zero-mean Gaussian white noise

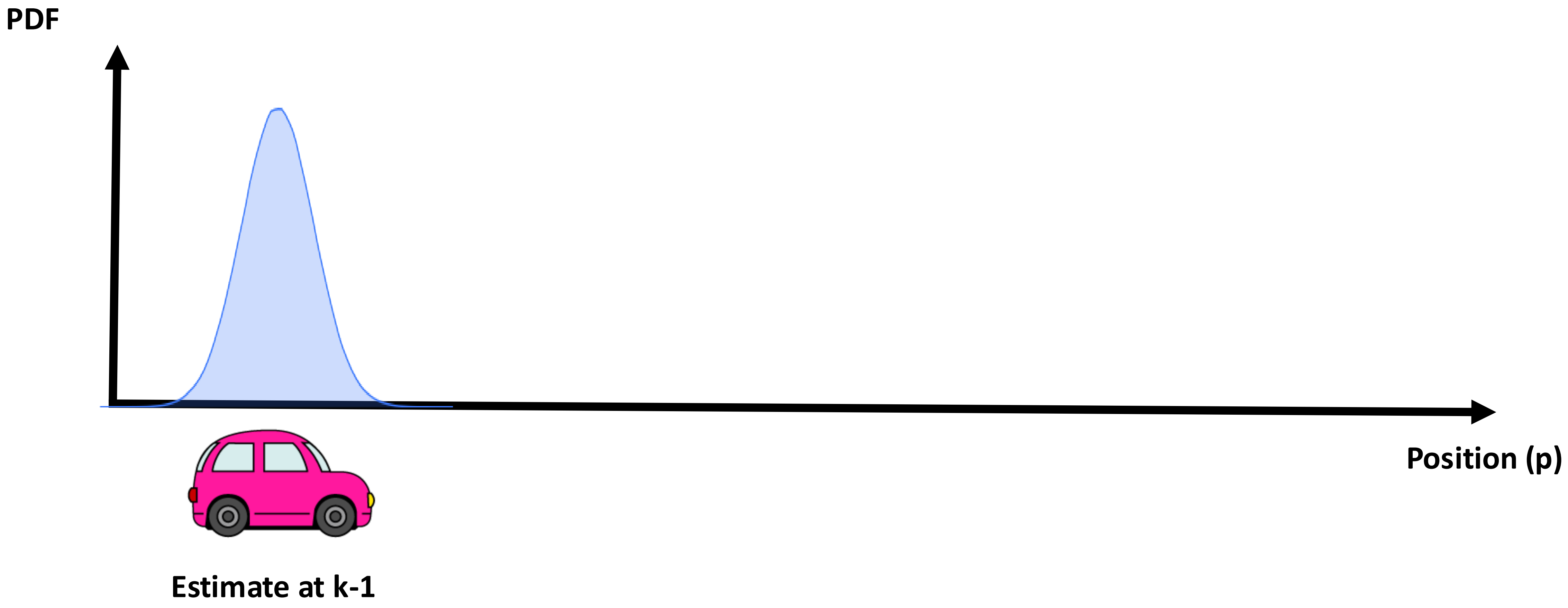
Kalman Filtering

- **Inputs**:
  - Measurements: *mx1* vector
  - Measurement Covariance matrix: *mxm* matrix
- **System Model**:
  - State transition matrix: *nxn* matrix
  - State-to-measurement matrix: *mxn* matrix
  - Process noise covariance matrix: *nxn* matrix
- **Internal Parameter**:
  - Kalman Gain: *nxm* matrix (the weight given to the measurements and current-state estimate)
- **Output:**
  - State variable: *nx1* vector
  - State Covariance matrix: *nxn* matrix

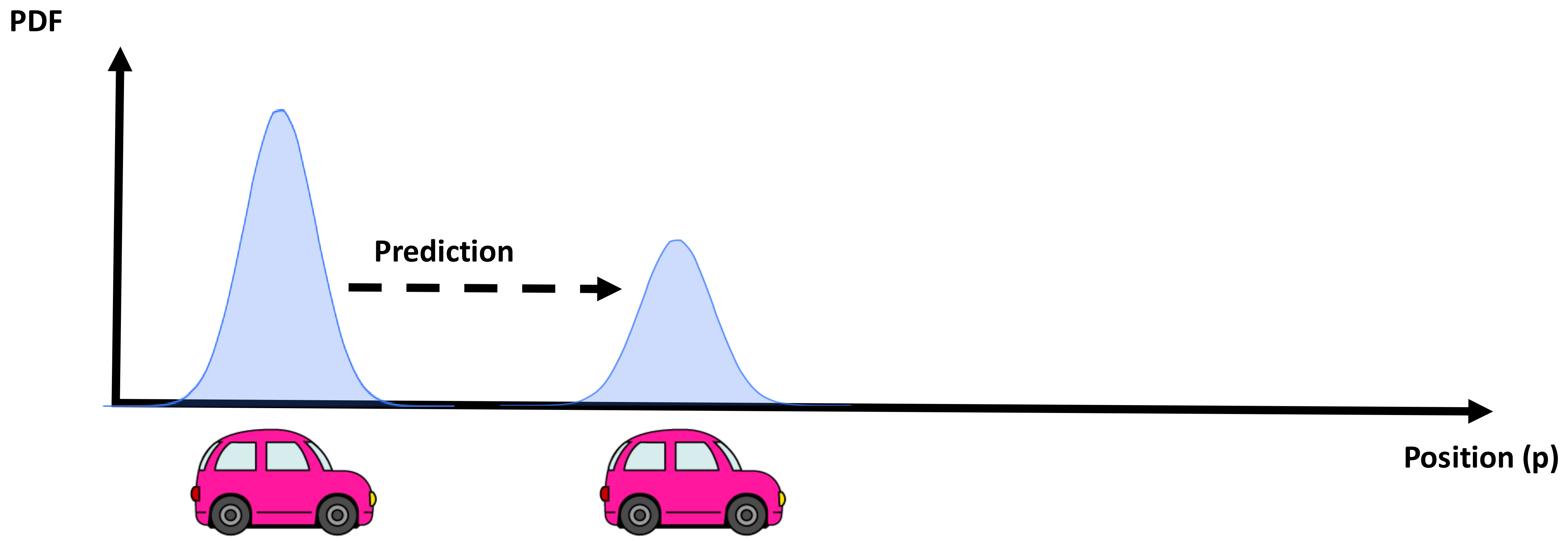
Kalman Filtering


Measurement 1
Initialize system state estimate and system state error covariance
Measurement 2
Reinitialize system state estimate and system state error covariance
Measurement 3+
Predict system state and system state error covariance till the measurement time
Compute the Kalman Gain
Estimate system state and system state error covariance till the measurement time

Kalman Filter: Example



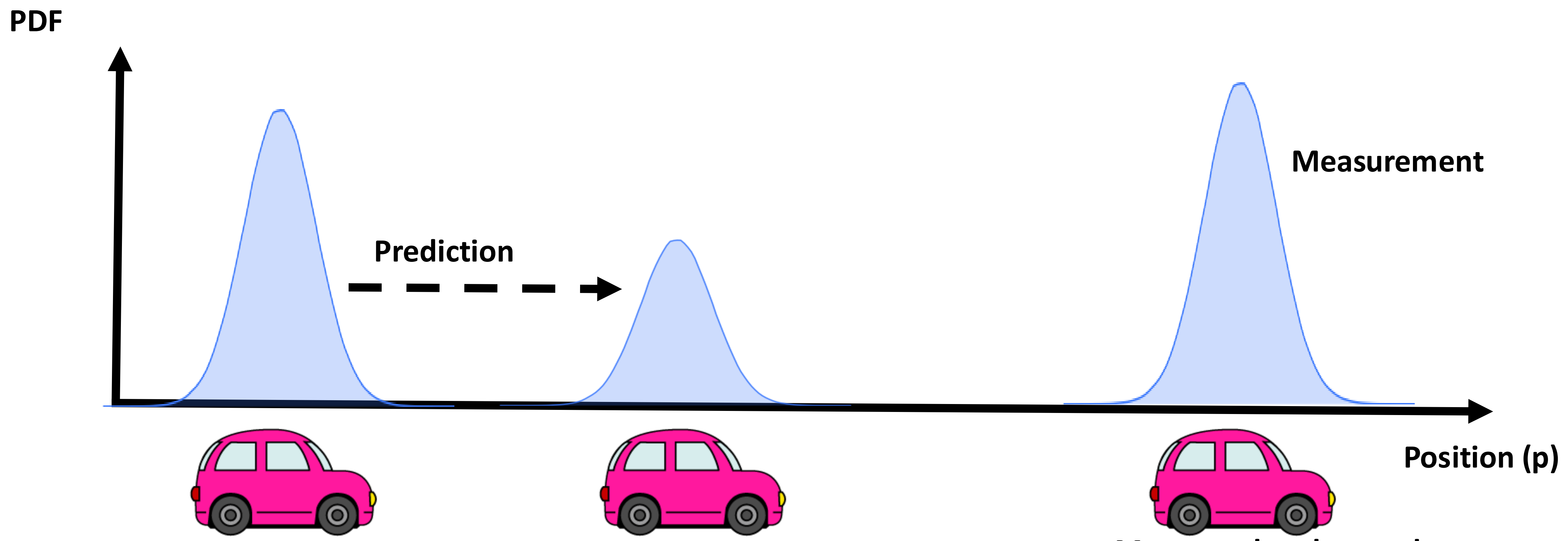
Kalman Filter: Example



Estimate at k-1

Apply motion model (say, odometry) for predicting the position at k

Kalman Filter: Example

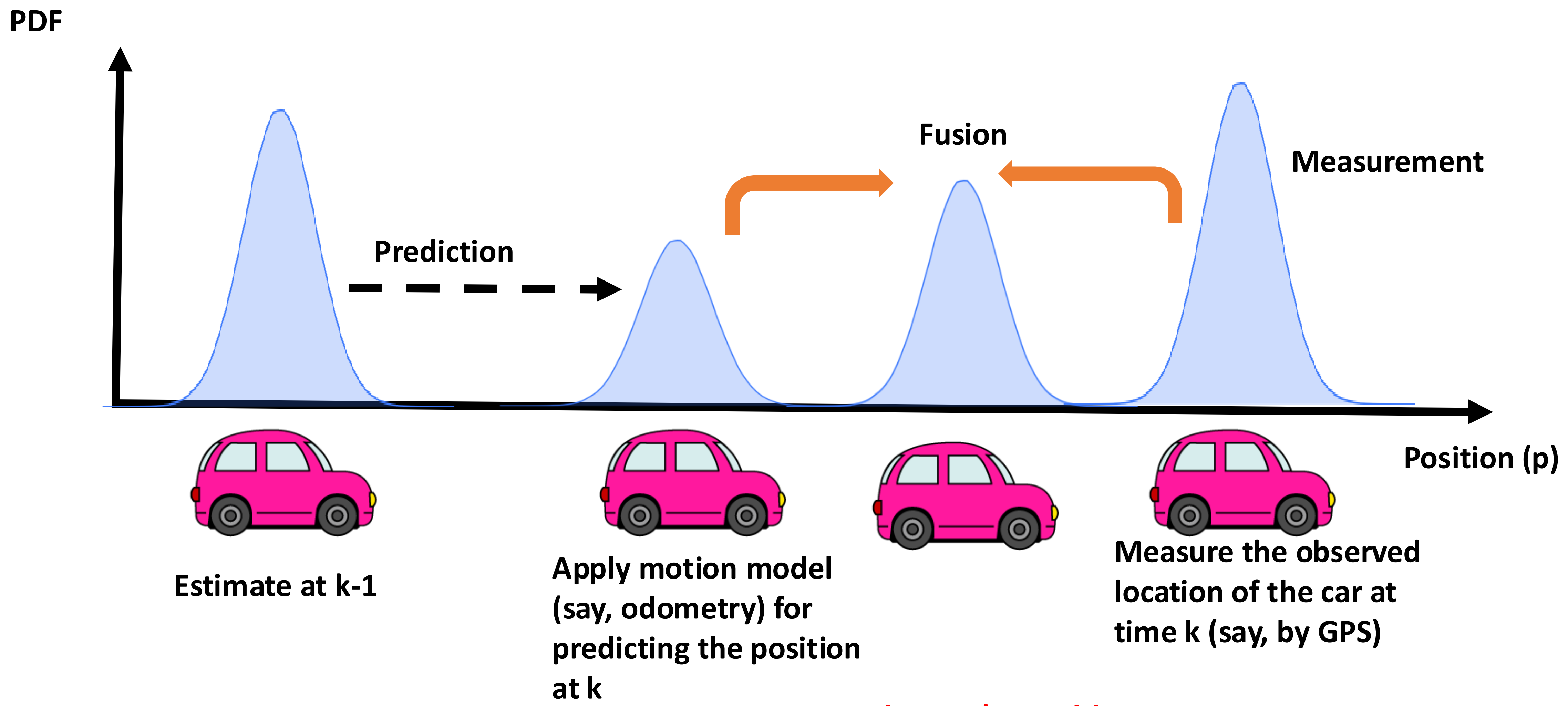


Estimate at k-1

Apply motion model (say, odometry) for predicting the position at k

Measure the observed location of the car at time k (say, by GPS)

Kalman Filter: Example


PDF
Prediction
Fusion
Measurement
Position (p)
Estimate at k-1
Apply motion model (say, odometry) for predicting the position at k
Measure the observed location of the car at time k (say, by GPS)

Estimate the position at k which is noise-free

Extended Kalman Filters (EKF)

- The simple Kalman filters use a linearity assumption in the state transition model F_k and the observation model H_k
  - Good for many applications
  - But not all applications can be modeled using liner motion

- Can we approximate a non-linear motion to a linear motion?

Extended Kalman Filters (EKFs)

- EKFs overcome the linearity assumptions
  - Considers that the next state probability and the measurement probabilities are giverned by non-linear functions g and h as,

    $x_k = g(u_k, x_{k-1}) + \varepsilon_k$

    $z_k = h(x_k) + \delta_k$

    where *g* and *h* are any arbitray function, and ε and δ are the process noise and observation noise
- However, with arbitrary functions, we cannot assume that the noises are drawn from a Gaussian distribution
  - We need a linear approximation of the functions

Extended Kalman Filters (EKFs)

- EKFs overcome the linearity assumptions
  - Considers that the next state probability and the measurement probabilities are giverned by non-linear functions g and h as,

    $x_k = g(u_k, x_{k-1}) + \varepsilon_k$

    $z_k = h(x_k) + \delta_k$

    where *g* and *h* are any arbitray fu
    observation noise
- However, with arbitrary functior
  drawn from a Gaussian distribut
  - We need a linear approximation o

How can we approcimate *g* and *h* to a linear function?

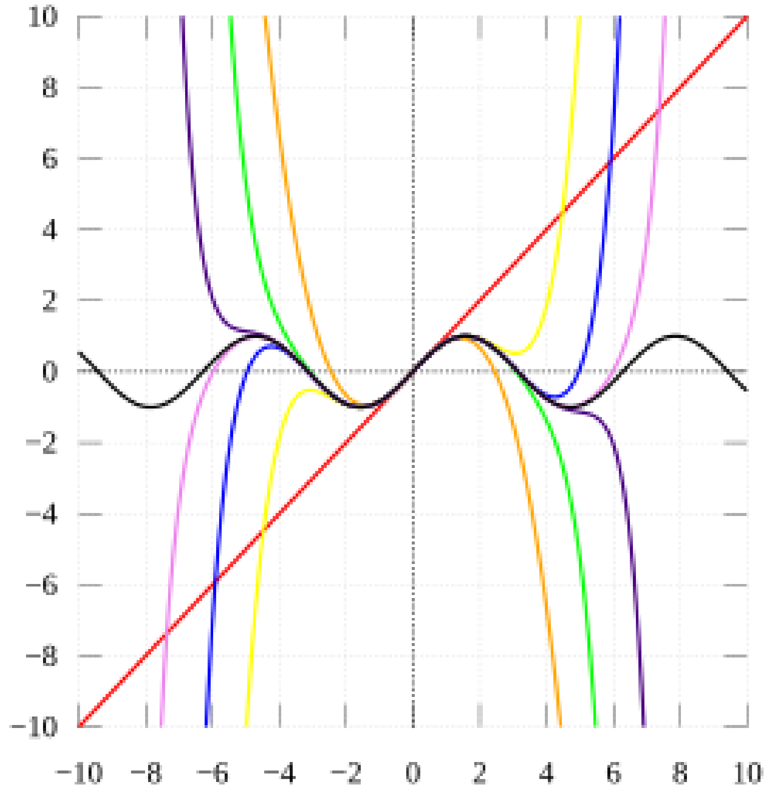
Extended Kalman Filters (EKF)

- Apply ***Taylor expansion*** to get a linear approximation of the function at any measurement point

- Use the partial derivative to get the slope:
  $g'(u_k, x_{k-1}) := \partial g(u_k, x_{k-1})/\partial x_{k-1}$

- We can approximate g using the Taylor expansion as follows:
  $g(u_k, x_{k-1}) \approx g(u_k, \mu_{k-1}) + g'(u_k, \mu_{k-1})(x_{k-1} - \mu_{k-1})$

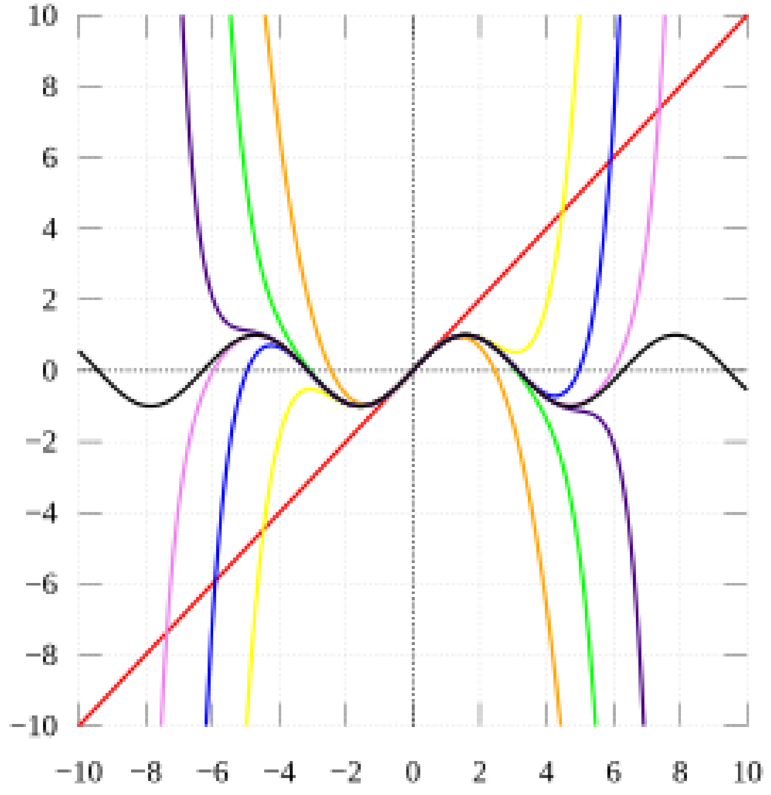
  where $g'(u_k, \mu_{k-1})$ is $g'(u_k, x_{k-1})$ evaluated at $x_{k-1} = \mu_{k-1}$



Extended Kalman Filters (EKF)

- We can approximate g using the Taylor expansion as follows:
  $g(u_k, x_{k-1}) \approx g(u_k, \mu_{k-1}) + g'(u_k, \mu_{k-1})(x_{k-1} - \mu_{k-1})$

  where $g'(u_k, \mu_{k-1})$ is $g'(u_k, x_{k-1})$ evaluated at $x_{k-1} = \mu_{k-1}$
- u_k is a constant here (based on the measurement -- the point at which you compute the slope), μ_{k-1} is also a constant. So, the first term returns a constant value
  - The approximation returns a linear function of x_{k-1}



Extended Kalman Filters (EKF)

- We define $G_k = g'(u_k, \mu_{k-1})$ (the Jacobian matrix of size n x n)
- Then the approximation is represented as follows:

$$g(u_k, x_{k-1}) \approx g(u_k, \mu_{k-1}) + G_k(x_{k-1} - \mu_{k-1})$$

- So, we can approximate the next state probability as follows:

$$p(x_k | u_k, x_{k-1}) := N(g(u_k, \mu_{k-1}) + G_k(x_{k-1} - \mu_{k-1}), Q_k)$$

Extended Kalman Filters (EKF)

- Similarly, we approximate

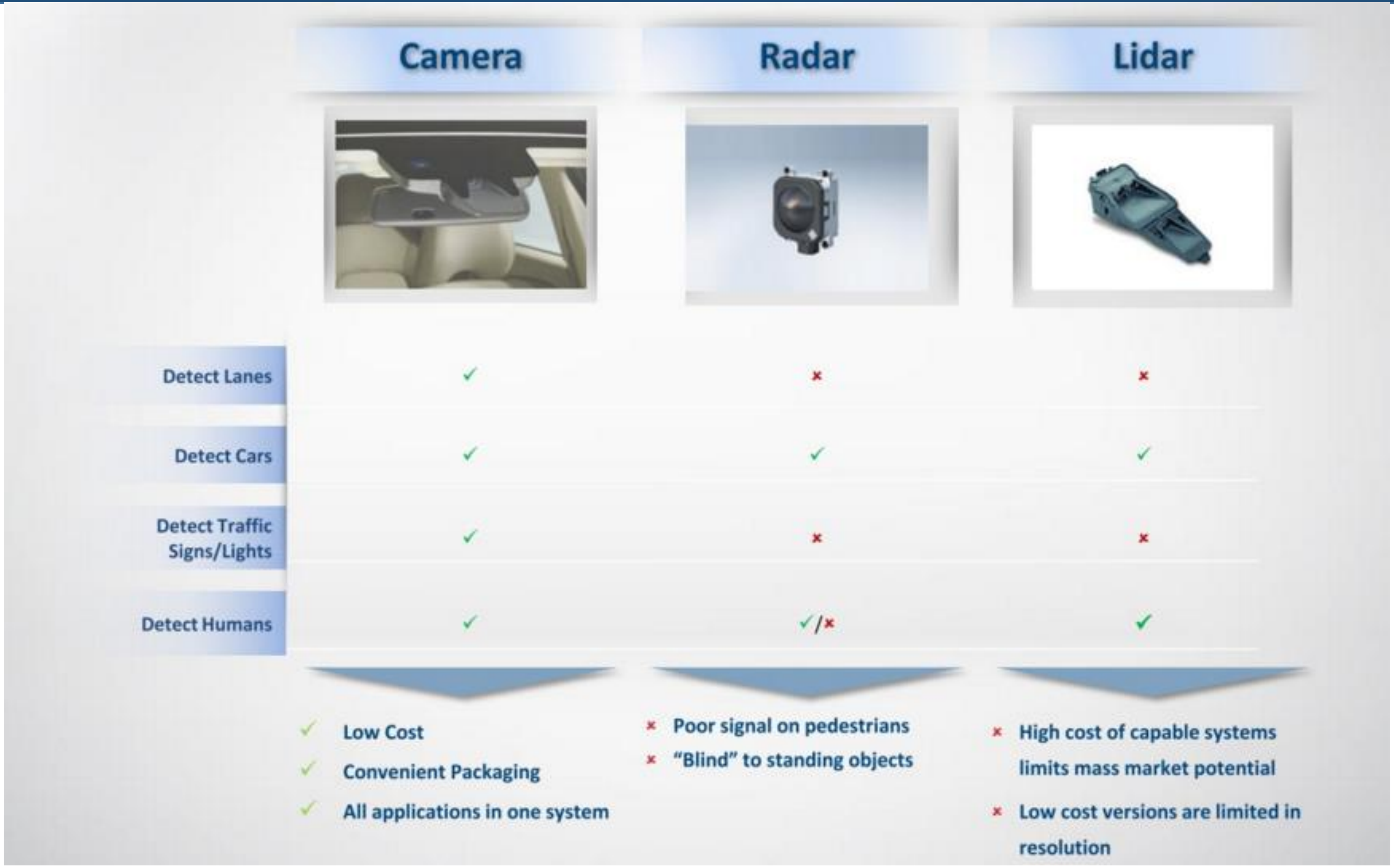
$$h(x_k) \approx h(\mu'_k) + h'(\mu'_k)(x_k - \mu'_k)$$
$$h(x_k) \approx h(\mu'_k) + H_k(x_k - \mu'_k)$$

where μ'_k is the state that is most likely at the time of linearizing h

- Finally, the measurement probability is approximated as:

$$z_k := N(h(\mu'_k) + H_k(x_k - \mu'_k), R_k)$$

Example: Tracking Vehicle Movements



	Camera	Radar	Lidar
Detect Lanes	✓	✗	✗
Detect Cars	✓	✓	✓
Detect Traffic Signs/Lights	✓	✗	✗
Detect Humans	✓	✓/✗	✓

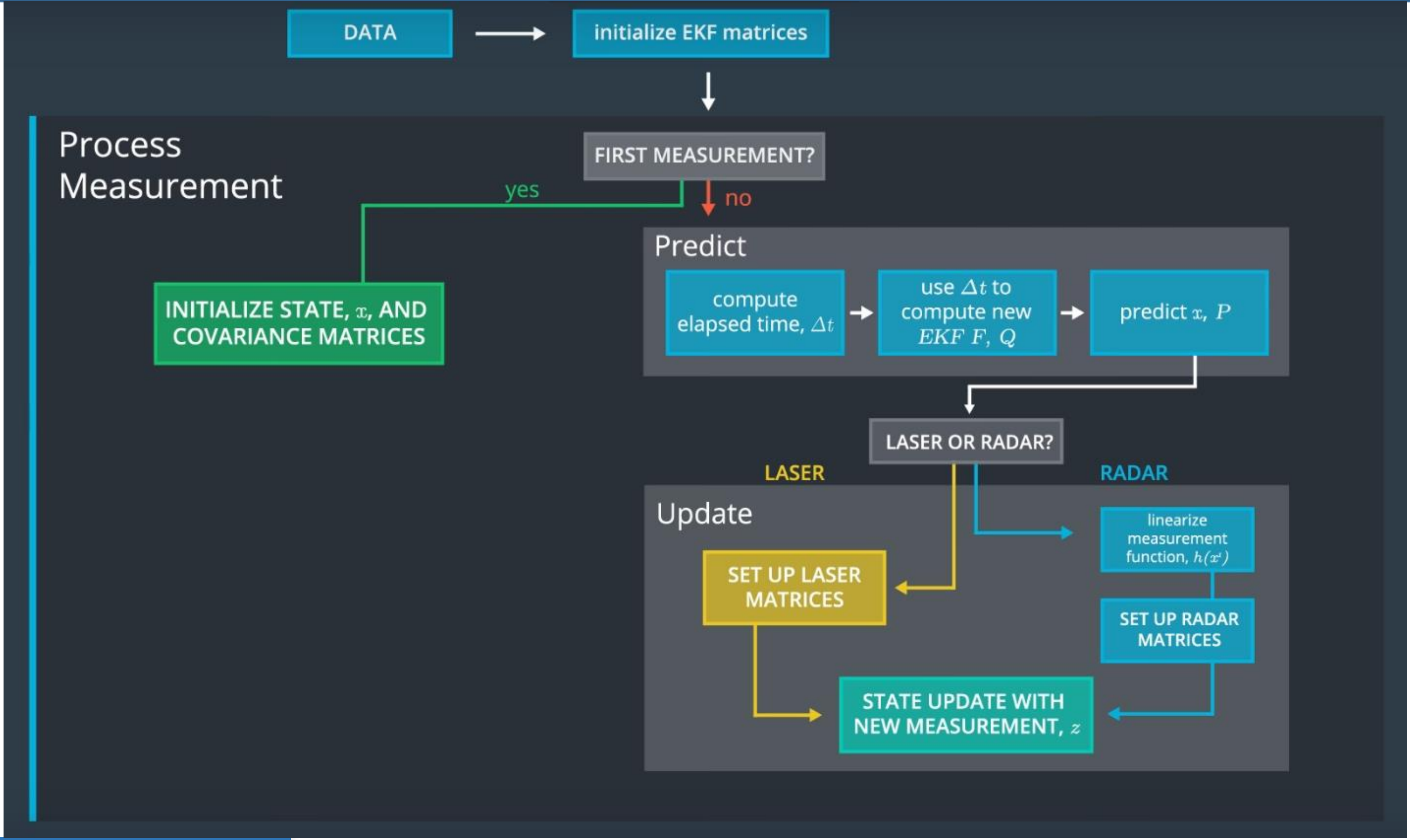
Camera:
- ✓ Low Cost
- ✓ Convenient Packaging
- ✓ All applications in one system

Radar:
- ✗ Poor signal on pedestrians
- ✗ "Blind" to standing objects

Lidar:
- ✗ High cost of capable systems limits mass market potential
- ✗ Low cost versions are limited in resolution

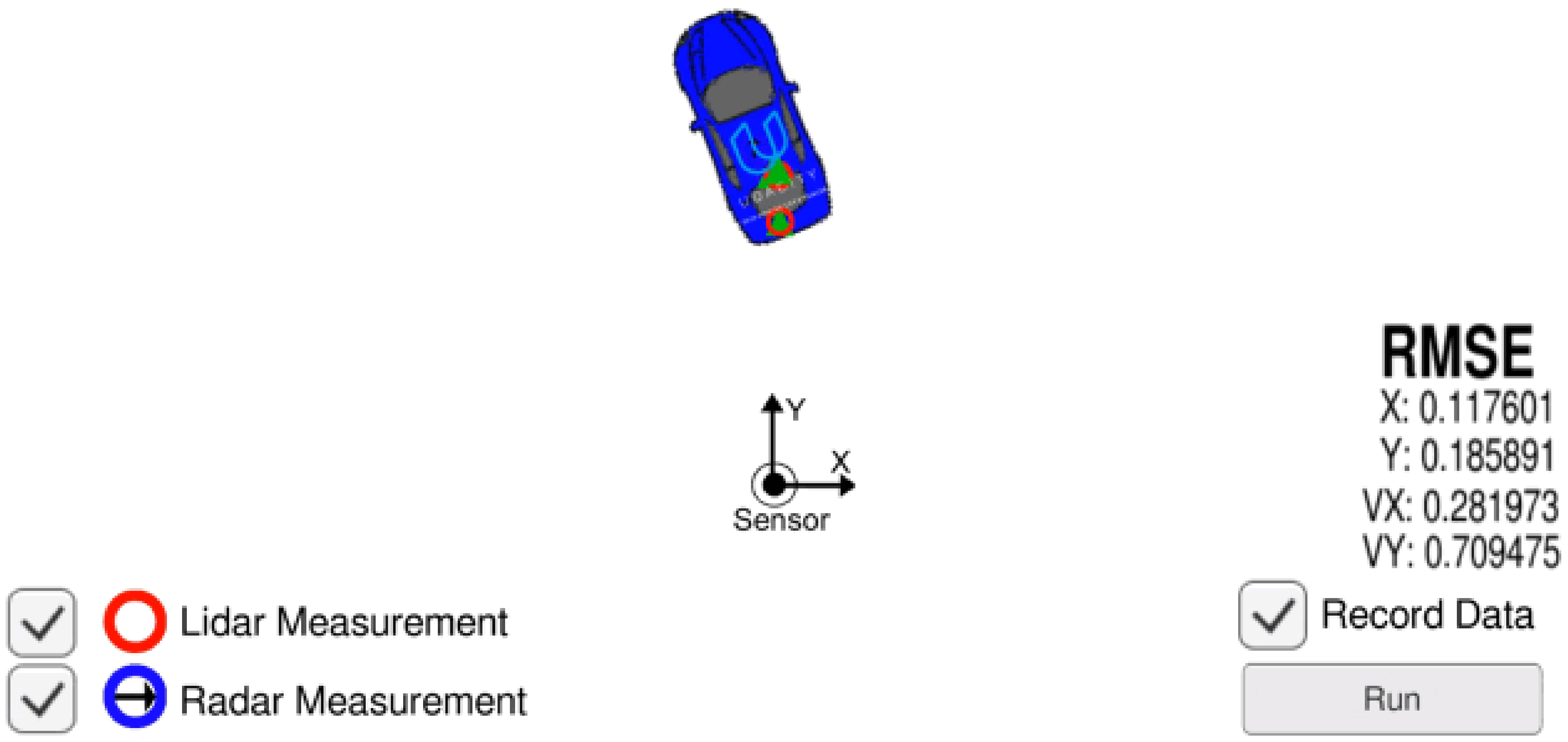
Source: https://junshengfu.github.io/tracking-with-Extended-Kalman-Filter/

Example: Tracking Vehicle Movements


DATA
initialize EKF matrices
Process Measurement
FIRST MEASUREMENT?
yes
no
INITIALIZE STATE, x, AND COVARIANCE MATRICES
Predict
compute elapsed time, Δt
use Δt to compute new EKF F, Q
predict x, P
LASER OR RADAR?
LASER
RADAR
Update
linearize measurement function, $h(x')$
SET UP RADAR MATRICES
SET UP LASER MATRICES
STATE UPDATE WITH NEW MEASUREMENT, z

Source: https://junshengfu.github.io/tracking-with-Extended-Kalman-Filter/

Example: Tracking Vehicle Movements



RMSE
X: 0.117601
Y: 0.185891
VX: 0.281973
VY: 0.709475

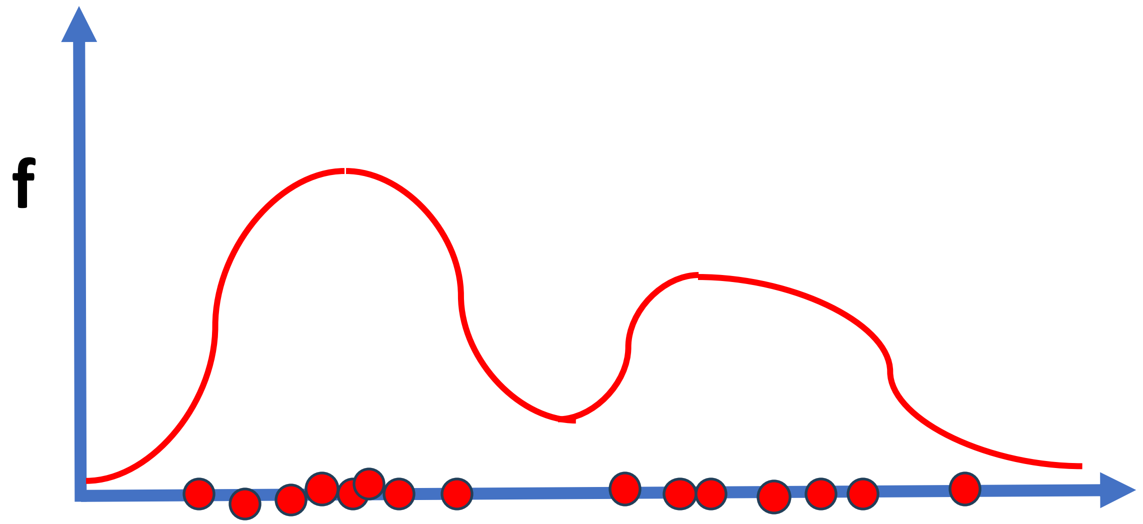
Source: https://junshengfu.github.io/tracking-with-Extended-Kalman-Filter/

Particle Filters

- **Non-parametric filters**: The representation of the state space is represented as a set of samples drawn from the underlying distribution
  - Remember that Kalman Filter uses a parametric form – use a function based on Gaussian distribution to model the state space
  - However, a function may not represent all possible samples in a state space, particularly when the state space is not based on some known form, like Gaussian

Particle Filters

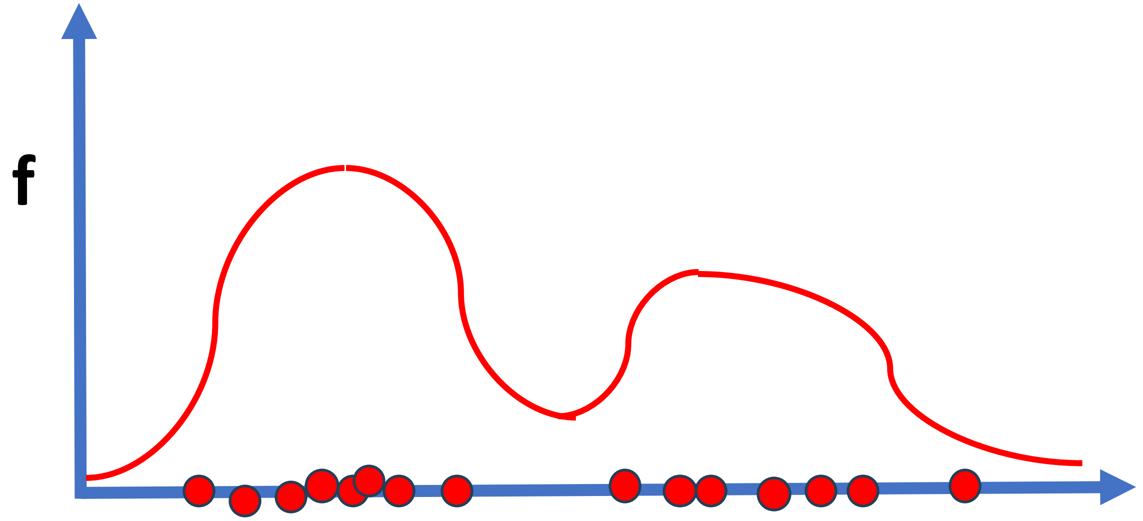
- **Non-parametric filter**: The representation of the state space is represented as a set of samples drawn from the underlying distribution
  - Remember that Kalman Filter uses a parametric form – use a function based on Gaussian distribution to model the state space
  - However, a function may not represent all possible samples in a state space, particularly when the state space is not based on some known form, like Gaussian



Particle Filters

- **Non-parametric filter**: The representation of the state space is represented as a set of samples dr
  - Remember that Kalm Gaussian distribution
  - However, a function r particularly when the

Rather than maintaining a set of parameters for the distribution (i.e., mean, covariance), the particle filter stores a set of samples that represent the distribution!



Particle Filters

- The samples drawn from the distribution are called the particles:

$$X_k = x_k^{[1]}, x_k^{[2]}, \ldots, x_k^{[M]}$$

  Each particle represents a hypothesis (a guess) that what the true world state might be at time *k* given the observations *z* and the actions *u*,

$$x_k^{[m]} \sim p(\, x_k \mid z_{1:k}, u_{1:k}\,)$$

- As $M \rightarrow \infty$, the denser a sub-region of the state space is populated by the samples drawn, the more likely that the true state falls within this expected sub-region

Particle Filter Algorithm

- Input: χ_{k-1}, u_k, z_k
- Initialize $\chi'_k = \chi_k = \emptyset$
- For m=1 to M, do
  - Sample $x_k^{[m]} \sim p(x_k \mid u_k, x_{k-1}^{[m]})$
  - $\omega_k^{[m]} = p(z_k | x_k^{[m]})$
  - $\chi'_k = \chi'_k + \langle x_k^{[m]}, \omega_k^{[m]} \rangle$
- For m=1 to M, do
  - draw *i* with probability proportional to $\omega_k^{[i]}$
  - Add $x_k^{[i]}$ to χ_k
- Return χ_k

Draw the initial set of particles based on the belief distribution; however, based on the observations, some particles might have a very low probability

Particle Filter Algorithm

- Input: χ_{k-1}, u_k, z_k
- Initialize $\chi'_k = \chi_k = \emptyset$
- For m=1 to M, do
  - Sample $x_k^{[m]} \sim p(x_k \mid u_k, x_{k-1}^{[m]})$
  - $\omega_k^{[m]} = p(z_k \mid x_k^{[m]})$
  - $\chi'_k = \chi'_k + \langle x_k^{[m]}, \omega_k^{[m]} \rangle$
- For m=1 to M, do
  - draw *i* with probability proportional to $\omega_k^{[i]}$
  - Add $x_k^{[i]}$ to χ_k
- Return χ_k

Importance Resampling

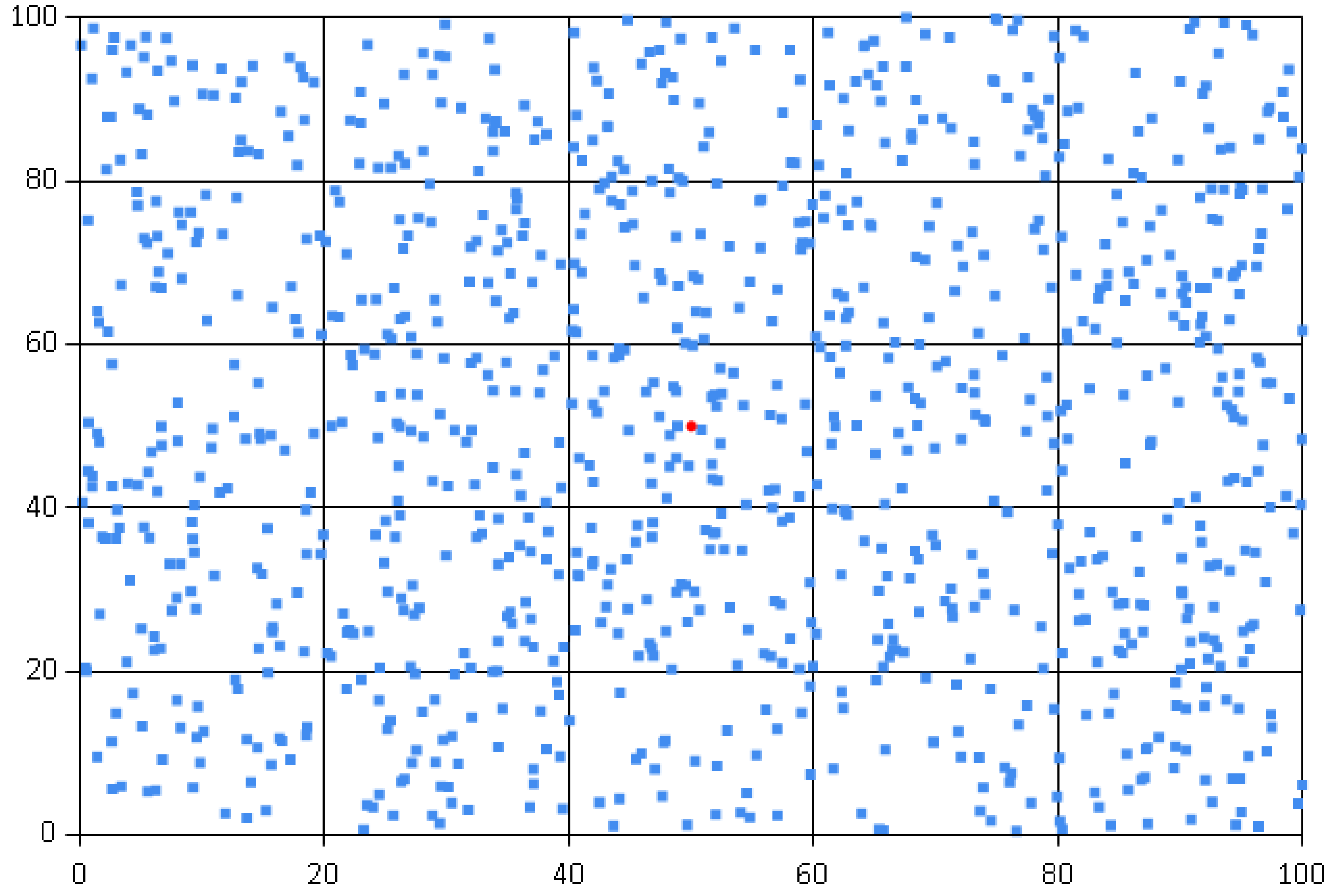
Resample the particles based on the probability: The particles having higher probability gets more number of instances, whereas the particles with very low probability vanishes

Particle Filter Algorithm

- Input: χ_{k-1}, u_k, z_k
- Initialize $\chi'_k = \chi_k = \emptyset$
- For m=1 to M, do
    - Sample $x_k^{[m]} \sim p(x_k \mid u_k, x_{k-1}^{[m]})$
    - $\omega_k^{[m]} = p(z_k | x_k^{[m]})$
    - $\chi'_k = \chi'_k + \langle x_k^{[m]}, \omega_k^{[m]} \rangle$
- For m=1 to M, do
    - draw *i* with probability proportional to $\omega_k^{[i]}$
    - Add $x_k^{[i]}$ to χ_k
- Return χ_k

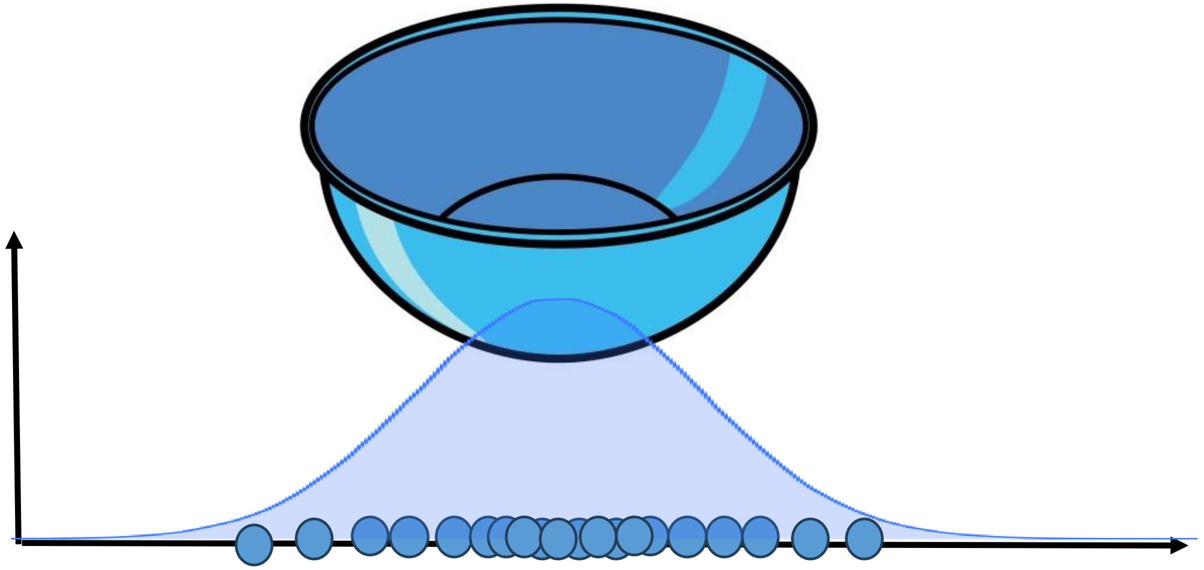
Particles which are more likely to represent the true state gets duplicated whereas the particles that has very low probabilities get vanished from the system!

Particle Filter - Visualization

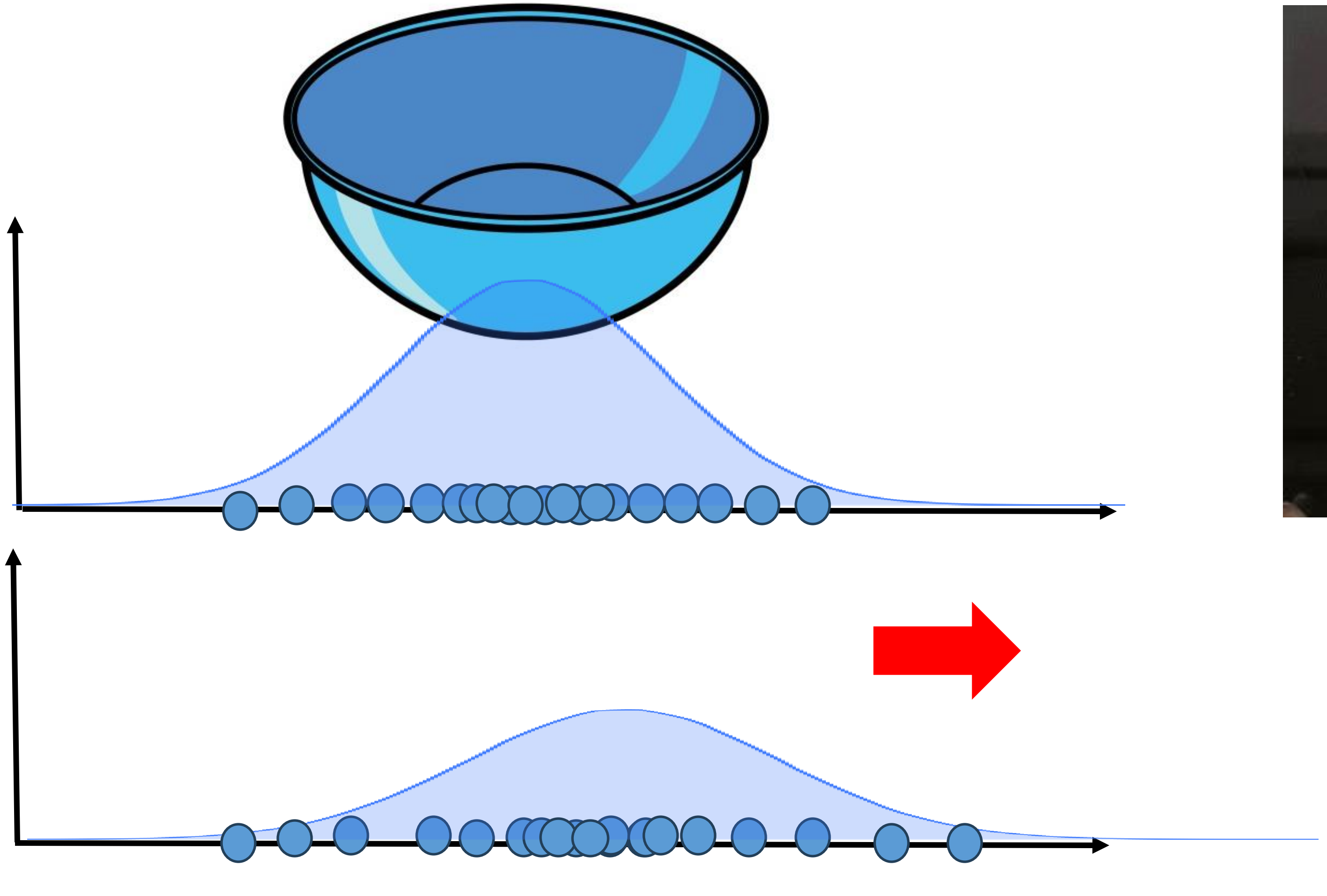


Source: https://towardsdatascience.com/particle-filter-a-hero-in-the-world-of-non-linearity-and-non-gaussian-6d8947f4a3dc

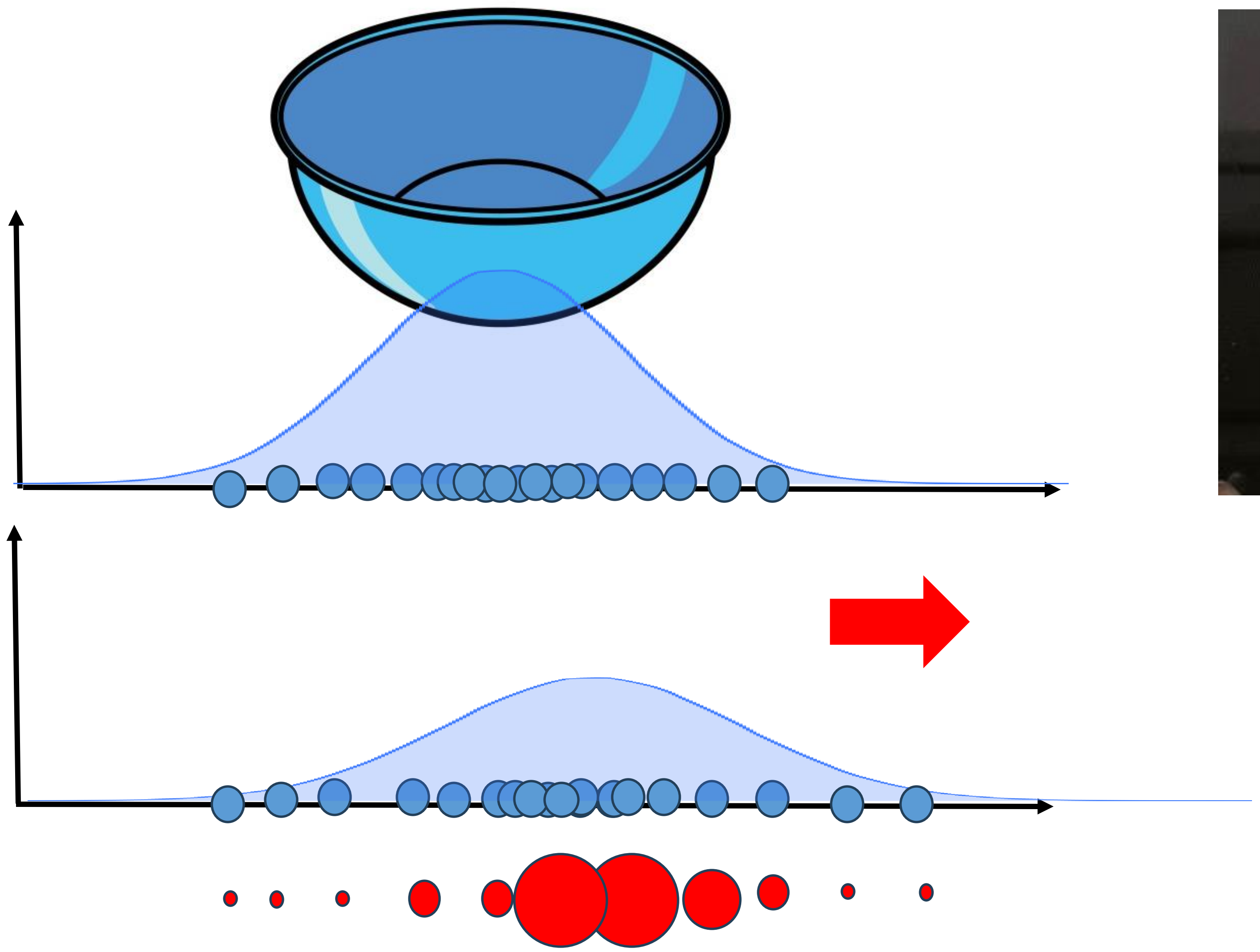
Particle Filters: An Example



Particle Filters: An Example

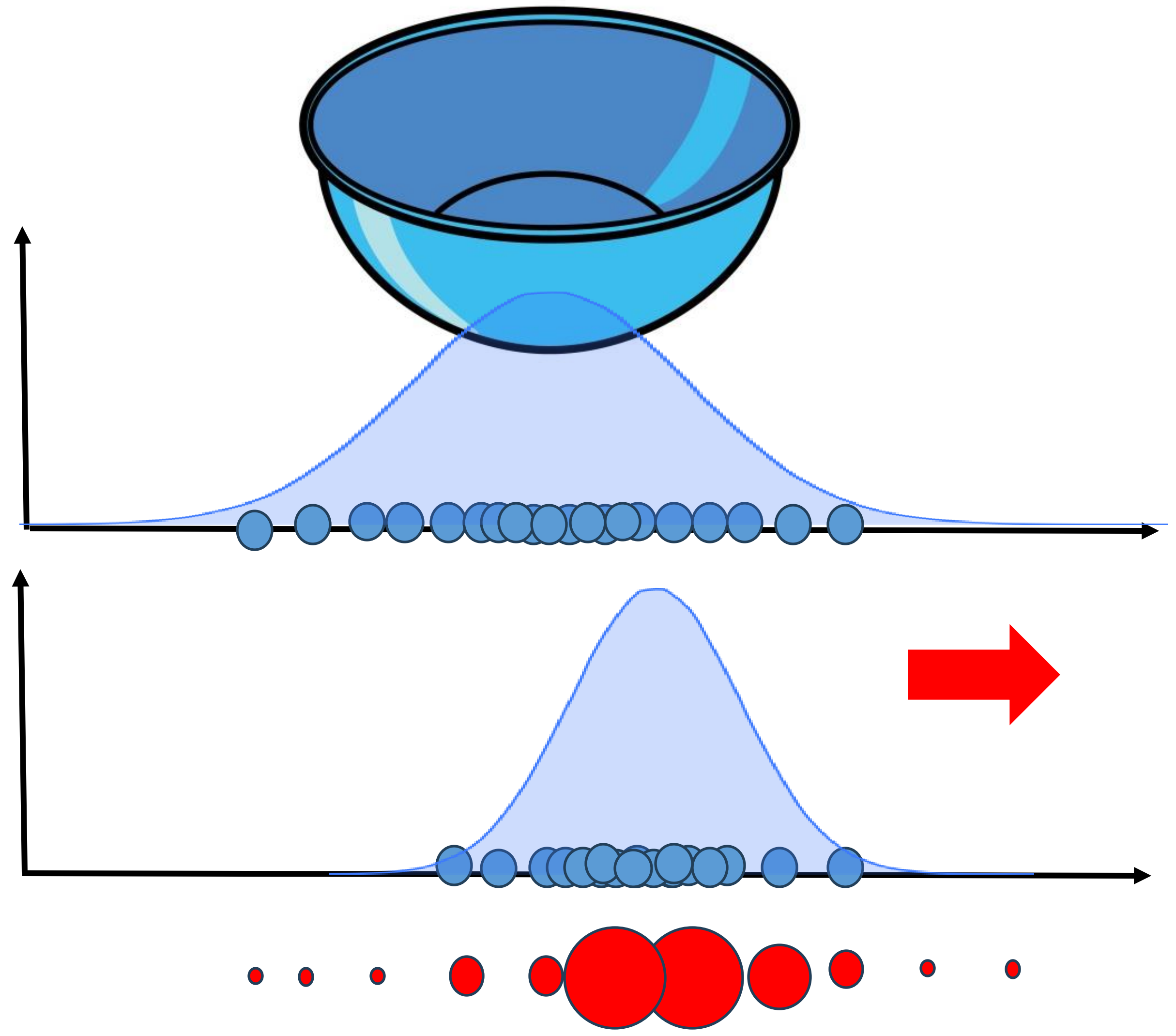


Particle Filters: An Example



Resampling

Particle Filters: An Example



Resampling

Design Challenges of Particle Filters

- **Number of particles**
  - Theoretically, the filter approximates true distribution when the number of particles is close to infinity
  - In practice, use a large value of M

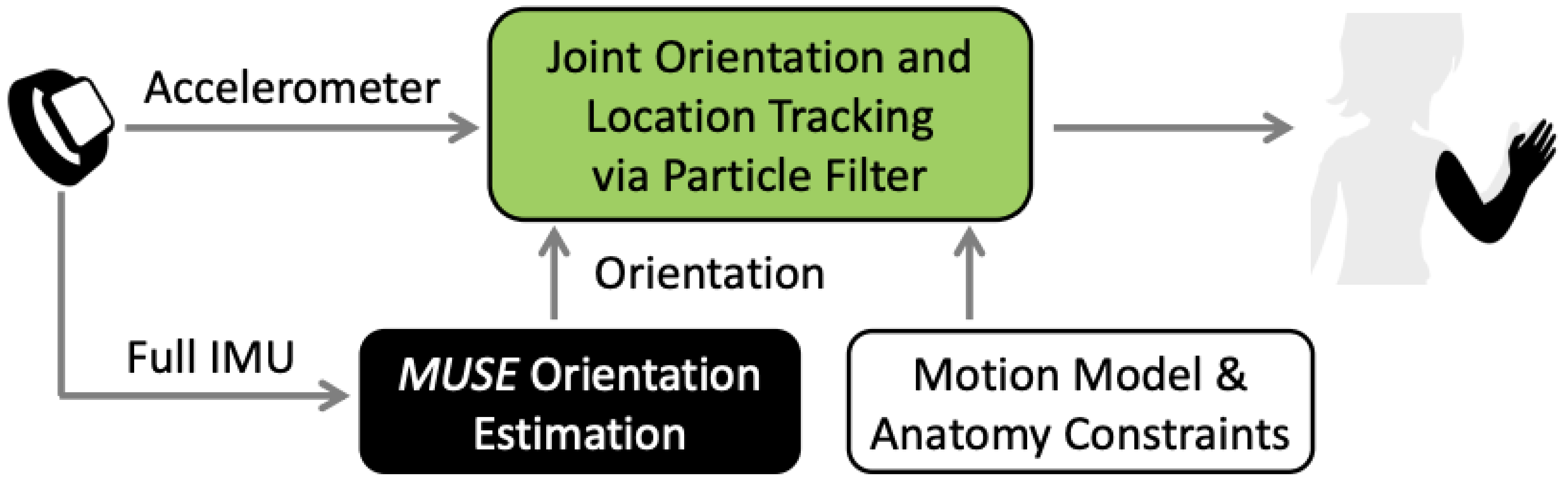
- **High variance**
  - Resampling introduces randomness -> the estimator results in a high variance
  - If we resample too frequently, we lose variety in the sample
  - Reduce the rate of resampling, or use a low-variance resampler

Design Challenges of Particle Filters

- **Divergence between proposed and target distribution**
    - What if the motion model is noisy but the sensors are correct?
    - **Inherent assumption:** The sensors are noisy, and the models are correct!

- **Particle deprivation**
    - Particles are sparse in high dimensional space
    - Particles near the true state may vanish due to random resampling
    - Periodically add a small number of new particles unformly over the space

Returning back to MUSE


Accelerometer
Joint Orientation and Location Tracking via Particle Filter
Orientation
Full IMU
MUSE Orientation Estimation
Motion Model & Anatomy Constraints

MUSE Particle Filtering

- Particles: <orientation, location>
- **State Space:** three recent location estimates along with the orientation drift around magnetic north vector

$$(i\text{-th particle})\ \boldsymbol{X}^i = \left[\boldsymbol{L}^i(t)\ \boldsymbol{L}^i(t-1)\ \boldsymbol{L}^i(t-2)\ \delta^i(t-1)\right]$$

- **Weight update:**
  - A good estimate of the state should ensure
    - It lies in the **valid location space**
    - It matches the accelerometer measurements
  - Compare the accelerometer sensor reading with the measured acceleration from particles' locations

Handling Acceleration and Orientation

- From the location of the particles, we can compute the global acceleration as follows:

$$\frac{\mathrm{d}^2\boldsymbol{L}^i(t-1)}{\mathrm{d}t^2} \approx \frac{\boldsymbol{L}^i(t) - 2\boldsymbol{L}^i(t-1) + \boldsymbol{L}^i(t-2)}{\Delta t^2}$$

- The accelerometer gives the true acceleration (impact of orientation + the gravitational acceleration)
  - MUSE orientation estimation can have a drift; subtract the drift as observed from the magnetic north vector

$$\hat{\mathbf{O}}^i(t-1) = \Delta R(\delta^i(t-1)) \times \hat{\mathbf{O}}(t-1)$$

  - Subtract the gravity component to get the global acceleration:

$$\frac{\mathrm{d}^2\boldsymbol{L}^i(t-1)}{\mathrm{d}t^2} = \hat{\mathbf{O}}^i(t-1)\cdot accel(t-1) - \begin{bmatrix}0 & 0 & \text{gravity}\end{bmatrix}^T$$

Handling Acceleration and Orientation

- From the location of the particles, we can compute the global acceleration as follows:

$$\frac{d^2 L^i(t-1)}{dt^2} \approx \frac{L^i(t) - 2L^i(t-1) + L^i(t-2)}{\Delta t^2}$$

- The accele the gravitation
  - MUSE o om the magneti
  - Subtract the gravity component to get the global acceleration.

$$\frac{d^2 L^i(t-1)}{dt^2} = \hat{O}^i(t-1) \cdot accel(t-1) - \begin{bmatrix} 0 & 0 & \text{gravity} \end{bmatrix}^T$$

Need to minimize the difference between these two. Assign the particle weight as a zero-mean Gaussian probability density function on these differences

Particle Filter for MUSE

- **Resampling:**
  - Resampled at each step to ensure that the particles are more concentrated to the ones with higher probability (better fitted to the accelerometer measurements)

- **Predictions:**
  - Takes the average drift collection for all the particles and use it to calibrate MUSE output
  - The location triples are updated after each step:

$$\begin{aligned} \boldsymbol{L}^i(t-2) &\leftarrow \boldsymbol{L}^i(t-1) \\ \boldsymbol{L}^i(t-1) &\leftarrow \boldsymbol{L}^i(t) \\ \boldsymbol{L}^i(t) &\leftarrow \text{(A random point from valid location space)} \end{aligned}$$

MUSE Particle Filtering

- **Resampling:**
  - Resampled at each step to ensure that the particles are more concentrated to the ones with higher probability (better fitted to the accelerometer measurements)

- **Predictions:**
  - Takes the average drift collection for all the particles and use it to calibrate MUSE output
  - The location triples are updated after each step:

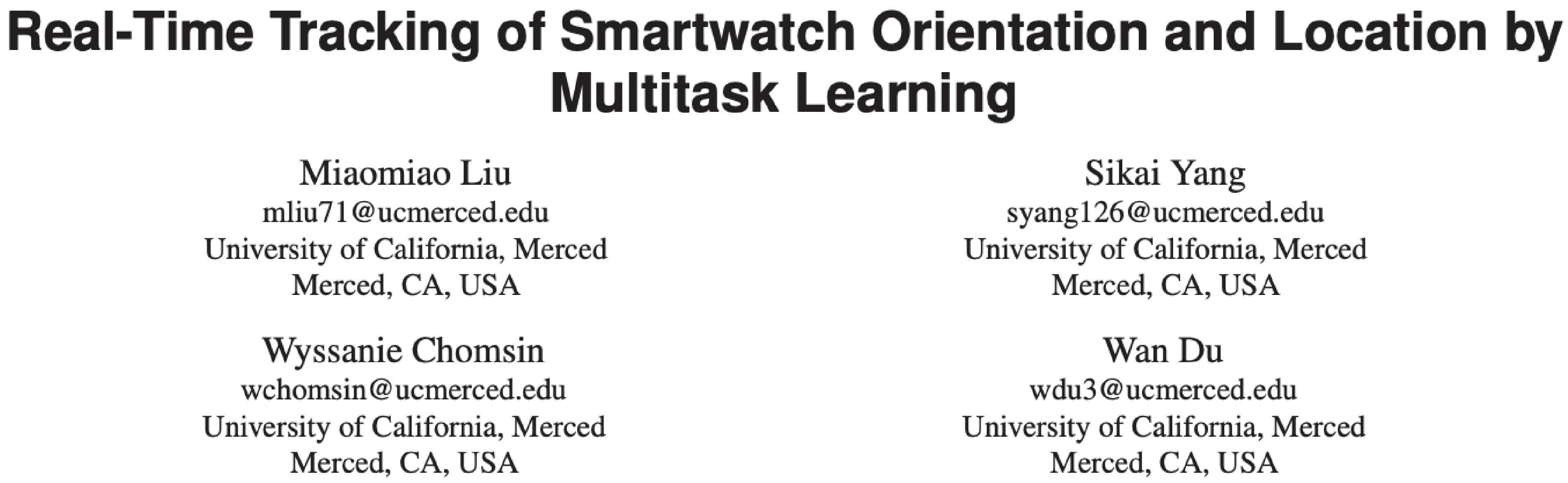
$$\begin{aligned} \boldsymbol{L}^i(t-2) &\leftarrow \boldsymbol{L}^i(t-1) \\ \boldsymbol{L}^i(t-1) &\leftarrow \boldsymbol{L}^i(t) \\ \boldsymbol{L}^i(t) &\leftarrow \text{(A random point from valid location space)} \end{aligned}$$

Preserves diversity in the system

In Summary

- IMU sensors are never correct
  - Even MUSE can suffer from strong magnetic interference
- Correction and calibrations are needed depending on application requirements
  - Not all applications need high accuracy: Think about the best optimization
  - Tradeoff across correctness and real-time performance
- We now know how to identify the gestures, how do we indentify activities from the gestures?

Further Reading



Real-Time Tracking of Smartwatch Orientation and Location by Multitask Learning

Miaomiao Liu
mliu71@ucmerced.edu
University of California, Merced
Merced, CA, USA

Sikai Yang
syang126@ucmerced.edu
University of California, Merced
Merced, CA, USA

Wyssanie Chomsin
wchomsin@ucmerced.edu
University of California, Merced
Merced, CA, USA

Wan Du
wdu3@ucmerced.edu
University of California, Merced
Merced, CA, USA

Liu, Miaomiao, et al. "Real-time tracking of smartwatch orientation and location by multitask learning." Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems. 2022. https://dl.acm.org/doi/pdf/10.1145/3560905.3568548

Some resources related to this topic