CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University

Lecture 07
EKF, UKF, Particle Filters, and SLAM

Instructor: Jingjin Yu

Outline

Kalman filter recap

Extended Kalman filter (EKF)

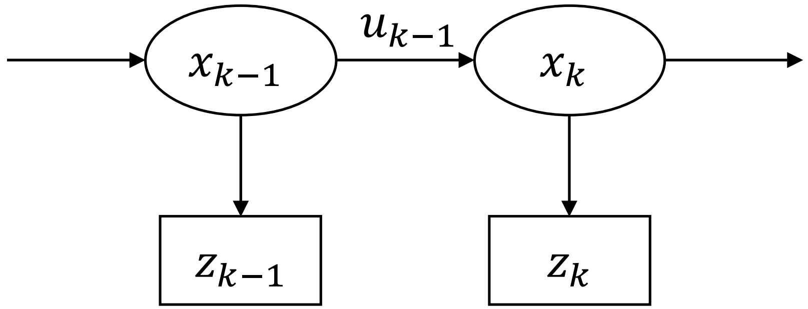
Unsented Kalman filter (UKF) and particle filters

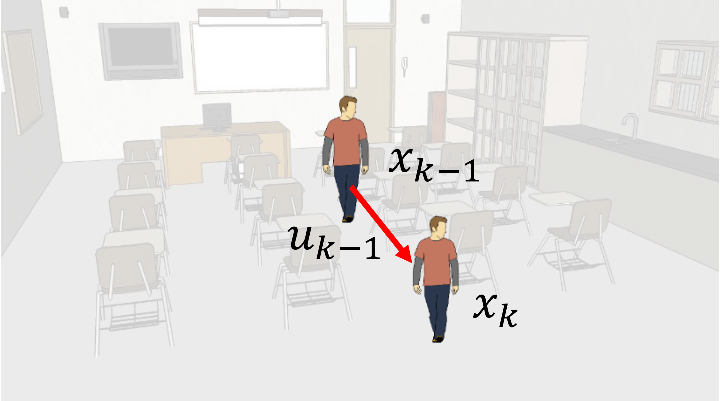
Simultaneous localization and mapping (SLAM)

Sensing review

Kalman Filter Review

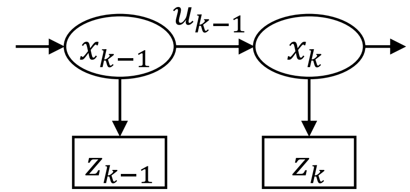
Kalman filter is a type of **Bayesian filters** over a **Hidden Markov model**





- ⇨The x_is are **hidden (actual)** system states that are **not directly known**
- ⇨We can **only observe** x_i using sensors to get observations z_i
- ⇨The (discrete) process is often modeled as a two-step iterative one
  - ⇨ Noisy state change: $x_k = f(x_{k-1}, u_{k-1}) + \omega_{k-1}$
  - ⇨ Noisy measurement after state change: $z_k = h(x_k) + v_k$
- ⇨The sequence of "data" is $u_0, z_1, u_1, z_2, u_2, z_3, \ldots$
- ⇨The goal is to derive an $\hat{x}_k$ as an accurate **estimate** of x_k

Kalman Filter Review – Assumptions



Stochastic, discrete-time **linear** system

$$x_k = Ax_{k-1} + Bu_{k-1} + \omega_{k-1}, \qquad \omega_{k-1} \sim N(0, Q) \qquad (1)$$

⇨ $x_k, \omega_{k-1} \in \mathbb{R}^n, u_{k-1} \in \mathbb{R}^m, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$

Linear observer (sensor)

$$z_k = Hx_k + \nu_k, \qquad \nu_k \sim N(0, R) \qquad (2)$$

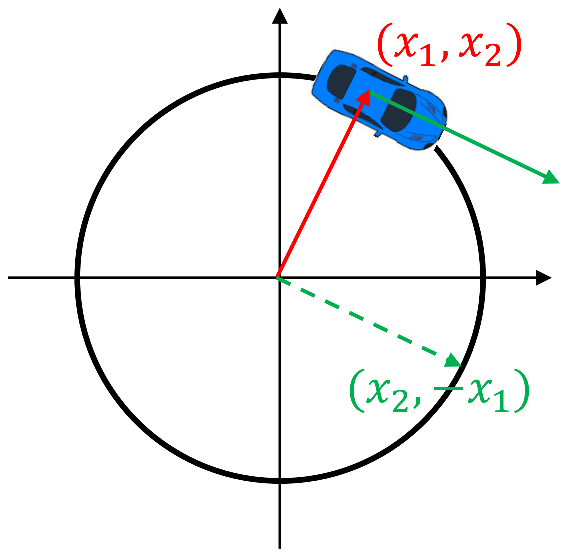
⇨ $z_k, \nu_k \in \mathbb{R}^\ell, H \in \mathbb{R}^{\ell\times n}$

Both ω_{k-1} and ν_k are **zero mean Gaussians** and are **uncorrelated**

⇨ i.e., $Cov(\omega, \nu) = 0$

Kalman Filter Review – Linear System Example

"Real" examples of **linear** dynamical systems



⇨Continuous

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

⇨Discrete (1D actual), no control

$$\begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1,k-1} \\ x_{2,k-1} \end{pmatrix}$$

⇨Discrete (1D actual), with "control"

$$\begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1,k-1} \\ x_{2,k-1} \end{pmatrix} + \begin{pmatrix} 0 & \frac{1}{2}(\Delta t)^2 \\ 0 & \Delta t \end{pmatrix} \begin{pmatrix} 0 \\ a \end{pmatrix}$$

Kalman Filter Review – Formulas

We have the iterative update algorithm

Time update

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$
$$P_k^- = AP_{k-1}A^T + Q$$

Measurement update

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$
$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R}$$
$$P_k = (I - K_k H)P_k^-$$

To run the algorithm

- ⇨The values of A, B, and H are known, u_{k-1} and z_k are also known
- ⇨The values of Q and R are estimated (**system identification** or **sys ID**)
- ⇨Initial values $\hat{x}_0$ and P_0 are guessed
- ⇨Usually P_k and K_k will quickly converge with the right Q and R

Extended Kalman Filter (EKF) – Assumptions

Kalman filter requires linearity, i.e.,

$$x_k = Ax_{k-1} + Bu_{k-1} + \omega_{k-1}, \qquad \omega_{k-1} \sim N(0, Q) \qquad (1)$$
$$z_k = Hx_k + v_k, \qquad v_k \sim N(0, R) \qquad (2)$$

These **nice assumptions** often do not hold in practice!

⇨ More realistic assumptions are

$$x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}), \qquad \omega_{k-1} \sim N(0, Q) \qquad (1^*)$$
$$z_k = h(x_k, v_k), \qquad v_k \sim N(0, R) \qquad (2^*)$$

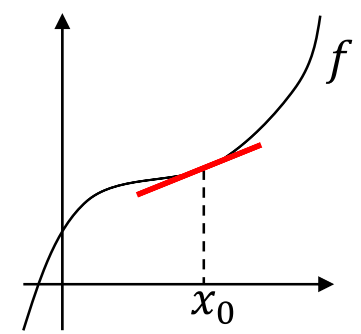
⇨ The functions f and h are **non-linear**

⇨ Here, we know f, h, u_{k-1}, z_k, and estimate Q and R

⇨ But, locally, f and h may be **linearly approximated**

⇨ This is achieved using **Taylor series**

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \ \ldots$$



Extended Kalman filter provides an **ad-hoc** extension to Kalman filters based on these assumptions, to compute the estimate, $\hat{x}_k$.

Update Equations for EKF

We have the iterative update algorithm

Time update

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$$
$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$$
$$A_k = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1}, u_{k-1}, 0}$$
$$W_k = \frac{\partial f}{\partial \omega}\Big|_{\hat{x}_{k-1}, u_{k-1}, 0}$$

Measurement update

$$\hat{x}_k = \hat{x}_k^- + K_k\big(z_k - h(\hat{x}_k^-, 0)\big)$$
$$K_k = \frac{P_k^- H_k^T}{H_K P_k^- H_K^T + V_k R V_k^T}$$
$$P_k = (I - K_k H_k) P_k^-$$
$$H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_k^-, 0}, \quad V_k = \frac{\partial h}{\partial v}\Big|_{\hat{x}_k^-, 0}$$

To run the algorithm

- ⇨Again, estimate Q and R offline (sys ID)
- ⇨Start filter with some initial $\hat{x}_0$ and P_0
- ⇨The values for A, W, H, and V change in each iteration
- ⇨Similar to Kalman filter, P_k and K_k can converge quickly if the model is right

A Note on the Two Update Steps

Both Kalman filter and EKF have time and measurement updates

⇨Kalman filter

Time update T_1	Measurement update M_1
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$ $P_k^- = AP_{k-1}A^T + Q$	$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$ $K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}, P_k = (I - K_k H)P_k^-$

⇨Extended Kalman filter

Time update T_2	Measurement update M_2
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$ $P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$ $A_k = \frac{\partial f}{\partial x}\vert_{\hat{x}_{k-1}, u_{k-1}, 0}$ $W_k = \frac{\partial f}{\partial \omega}\vert_{\hat{x}_{k-1}, u_{k-1}, 0}$	$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - h(\hat{x}_k^-, 0)\right)$ $K_k = P_k^- H_k^T\left(H_K P_k^- H_K^T + V_k R V_k^T\right)^{-1}$ $P_k = (I - K_k H_k)P_k^-$ $H_k = \frac{\partial h}{\partial x}\vert_{\hat{x}_k^-, 0}, V_k = \frac{\partial h}{\partial \nu}\vert_{\hat{x}_k^-, 0}$

One can mix and match these!

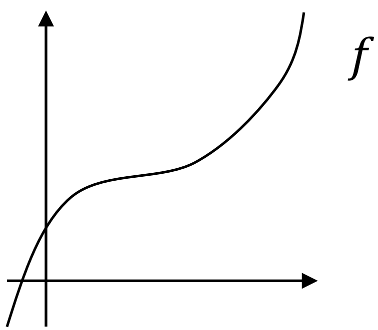
⇨E.g., one can build a filter with T_2 and M_1. Or T_1 and M_2

⇨This generally applies to two-stage filters including later ones

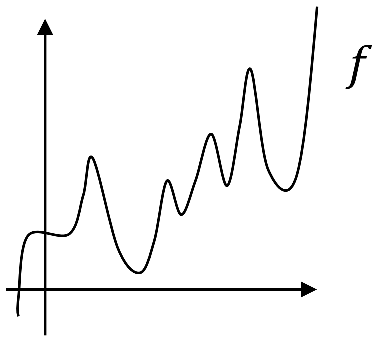
Issues with the Extended Kalman Filter

There are many issues with EKF

- ⇨Can perform poorly with highly non-linear f and h
  - ⇨ i.e., Taylor expansion may not capture f or h well enough
  - ⇨ This can be particularly true for f



"good" f



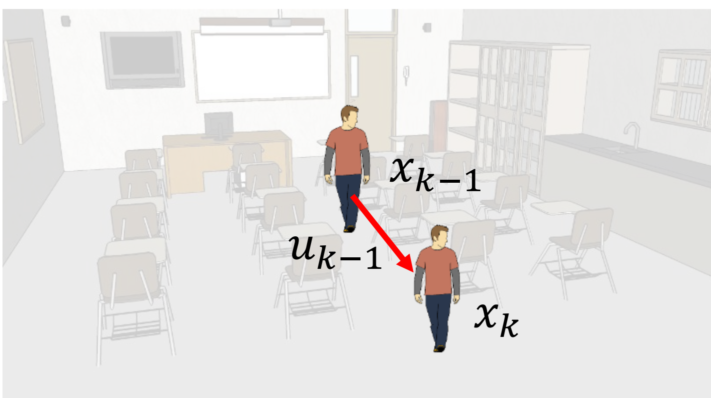
"bad" f

- ⇨Also, computing $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial \omega}$, and $\frac{\partial h}{\partial \nu}$ may be difficult or impossible
  - ⇨ E.g., f may be very complex and may not even have a closed form

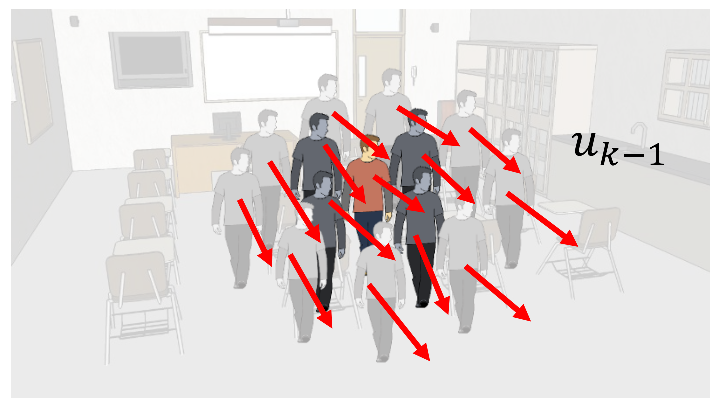
Unscented Kalman Filter and Particle Filter

Unscented Kalman filter (UKF) and Particle filter avoid such problems

- ⇨ For time update
  - ⇨ Directly **sample** $\hat{x}_{k-1}$ and obtain a certain number of samples $\hat{x}_{k-1}^{i}$ with weights
  - ⇨ Directly "push" the samples through f
  - ⇨ Compute $\hat{x}_k^-$ and P_k^- from these updated samples
  - ⇨ This can be imagined as running many Kalman filters
- ⇨ Similar steps for measurement update
- ⇨ Comparison to Kalman filter/EKF



Kalman filter/EKF



UKF/particle filters

- ⇨ Difference between UKF and particle filters
  - ⇨ UKF uses deterministic samples (so called "unscented" transformation)
  - ⇨ Particle filters use Monte Carlo sampling, usually with more samples than UKF

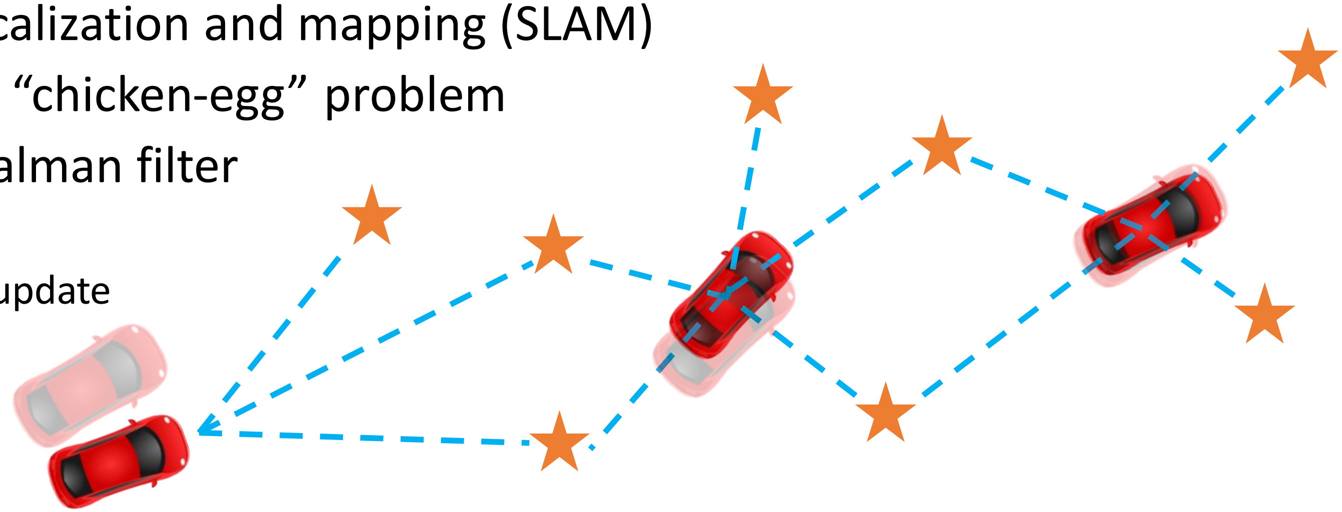
Simultaneous Localization and Mapping

Suppose you arrived a new town (e.g., travel, or playing an RPG)

- ➩How do you explore?
- ➩Move around and look for landmarks
  - ➩ Houses, buildings, roads, etc.
- ➩Build map based on the landmarks
- ➩Localize yourself on the map

Robots would need to do the same

- ➩Build a map using landmarks
- ➩Localize using the map
- ➩Simultaneous localization and mapping (SLAM)
- ➩This is partially a “chicken-egg” problem
- ➩Very similar to Kalman filter
  - ➩ Time update
  - ➩ Measurement update



SLAM, More Formally

Problem setup

- A robot moves in an environment with states x_k
- Make relative observation of $\mathbf{m} = m_1, \ldots, m_n$ **landmark** locations
- State history $X_k = \{x_0, \ldots, x_k\} = X_{k-1} \cup \{x_k\}$
- Control u_k applied at x_{k-1}
- $U_k = \{u_1, \ldots, u_k\} = U_{k-1} \cup \{u_k\}$
- z_{ik}: an observation of the i-th landmark at time step k
- $\mathbf{z}_k = (z_{1k}, \ldots, z_{nk})$ and $Z_k = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\} = Z_{k-1} \cup \{\mathbf{z}_k\}$
- Motion model: $P(x_k \mid x_{k-1}, u_k)$
  - This is similar to $x_k = f(x_{k-1}, u_k, \omega)$
  - Note the indexing is different from Kalman filter – this is due to convention
- Observation model: $P(\mathbf{z}_k \mid x_k, \mathbf{m})$

SLAM is to compute the following probability distribution

What about Kalman filter?

$$P(x_k, \mathbf{m} \mid Z_k, U_k, x_0)$$

x_k: location

m: map

SLAM also has time and observation (measurement) updates

SLAM Time Update

The time update makes predictions based on x_{k-1} and u_k

End goal: $P(x_k, \mathbf{m} \mid Z_k, U_k, x_0)$

$$
\begin{aligned}
&P(x_k, \mathbf{m} \mid Z_{k-1}, U_k, x_0) \\
&= \int P(x_k, x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_k, x_0)\mathrm{d}x_{k-1} \quad \text{(marginalization)} \\
&= \int \frac{P(x_k, x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(Z_{k-1}, U_k, x_0)}\mathrm{d}x_{k-1} \\
&= \int \frac{P(x_k, x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)} \frac{P(x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(Z_{k-1}, U_k, x_0)}\mathrm{d}x_{k-1} \\
&= \int P(x_k \mid x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0) P(x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_k, x_0)\mathrm{d}x_{k-1} \\
&= \underline{\int P(x_k \mid x_{k-1}, u_k) P(x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_{k-1}, x_0)\mathrm{d}x_{k-1}}
\end{aligned}
$$

- ⇨ The last step applies two conditional independences
  - ⇨ $P(x_k \mid x_{k-1}, u_k) = P(x_k \mid x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)$
  - ⇨ $P(x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_{k-1}, x_0) = P(x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_k, x_0)$
- ⇨ The term $P(x_k \mid x_{k-1}, u_k)$ is provided
  - ⇨ E.g., $x_k = Ax_{k-1} + Bu_k + \omega_{k-1}$ in a Kalman filter
- ⇨ The term $P(x_{k-1}, \mathbf{m} \mid Z_{k-1}, U_{k-1}, x_0)$ is from previous iteration or x_0
- ⇨ So this step is basically the same as the time update of a Kalman filter

SLAM Observation Update

Last slide: $P(\, x_k, \mathbf{m} \mid Z_{k-1}, U_k, x_0 \,)$

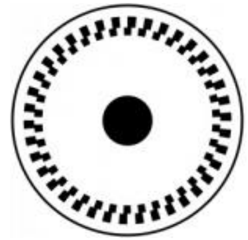
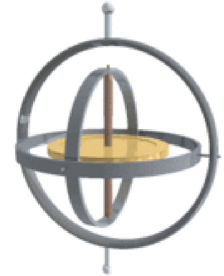
The observation update estimate $x_k, \mathbf{m}$ based on time update and $\mathbf{z}_k$

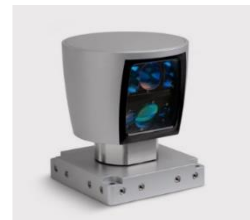
$$
\begin{aligned}
&P(\, x_k, \mathbf{m} \mid Z_k, U_k, x_0 \,) \\
&= P(\, x_k, \mathbf{m} \mid \mathbf{z}_k, Z_{k-1}, U_k, x_0 \,) \\
&= \frac{P(x_k, \mathbf{m}, \mathbf{z}_k, Z_{k-1}, U_k, x_0)}{P(\mathbf{z}_k, Z_{k-1}, U_k, x_0)} \\
&= \frac{P(x_k, \mathbf{m}, \mathbf{z}_k, Z_{k-1}, U_k, x_0)}{P(x_k, \mathbf{m}, Z_{k-1}, U_k, x_0)} \frac{P(x_k, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(Z_{k-1}, U_k, x_0)} \frac{P(Z_{k-1}, U_k, x_0)}{P(\mathbf{z}_k, Z_{k-1}, U_k, x_0)} \\
&= \frac{P(\mathbf{z}_k | \mathbf{m}, x_k, Z_{k-1}, U_k, x_0) P(x_k, \mathbf{m} | Z_{k-1}, U_k, x_0)}{P(\mathbf{z}_k | Z_{k-1}, U_k, x_0)} \\
&= \frac{P(\mathbf{z}_k | x_k, \mathbf{m}) P(x_k, \mathbf{m} | Z_{k-1}, U_k, x_0)}{P(\mathbf{z}_k | Z_{k-1}, U_k, x_0)} \qquad \text{(conditional independence)} \\
&\propto \underline{P(\mathbf{z}_k \mid x_k, \mathbf{m}) P(x_k, \mathbf{m} \mid Z_{k-1}, U_k, x_0)}
\end{aligned}
$$

- ⇨The term $P(\mathbf{z}_k \mid Z_{k-1}, U_k, x_0)$ can be normalized and does not matter
- ⇨The term $P(\mathbf{z}_k \mid x_k, \mathbf{m})$ is based on observation
  - ⇨ In extended Kalman filter, this is just $z_k = h(x_k, v_k)$
  - ⇨ In SLAM this is the challenging step
    - ⇨ Uses many techniques, e.g., iterative closest point fitting (ICP)
    - ⇨ Not part of the focus of this course – mostly computer vision techniques
- ⇨The term $P(x_k, \mathbf{m} \mid Z_{k-1}, U_k, x_0)$ is from time update

Sensing Review

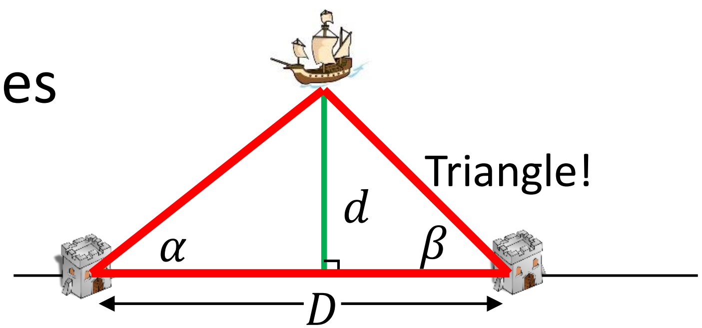
Sensor mechanisms

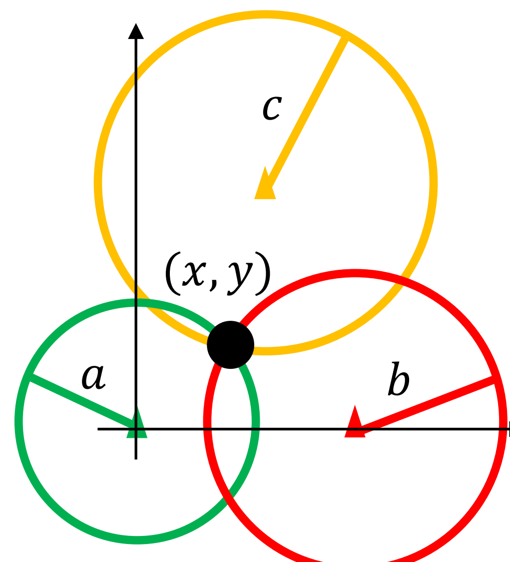





Localization techniques

- ⇨Triangulation
- ⇨Trilateration





Bayesian filters

- ⇨Kalman filter, EKF
- ⇨UKF/particle filters
- ⇨Simultaneous localization and mapping (SLAM)