Robotic Systems I

Lecture 6: Non-linear Filters and SLAM

Konstantinos Chatzilygeroudis - costashatz@upatras.gr

Department of Electrical and Computer Engineering
University of Patras

Template made by Panagiotis Papagiannopoulos

Bayes Filtering

$$
\begin{aligned}
\text{bel}(\boldsymbol{x}_t) &= p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) \\
&= \underbrace{\eta p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t})}_{\text{Bayes Rule}} \\
&= \underbrace{\eta p(\boldsymbol{z}_t|\boldsymbol{x}_t)}_{\text{Markov Assumption}} \; p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) \\
&= \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t) \underbrace{\int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) d\boldsymbol{x}_{t-1}}_{\text{Law of Total Probability}} \\
&= \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t) \underbrace{\int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}) d\boldsymbol{x}_{t-1}}_{\text{Markov Assumption}} \\
&= \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t) \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \text{bel}(\boldsymbol{x}_{t-1}) d\boldsymbol{x}_{t-1}
\end{aligned}
$$

Bayes Filtering in Two Steps

Prediction Step

$$\overline{\text{bel}}(\boldsymbol{x}_t) = \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)\text{bel}(\boldsymbol{x}_{t-1})d\boldsymbol{x}_{t-1}$$

Correction Step

$$\text{bel}(\boldsymbol{x}_t) = \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t)\overline{\text{bel}}(\boldsymbol{x}_t)$$

Kalman Filtering in Two Steps

Prediction Step

$$\overline{\text{bel}}(\boldsymbol{x}_t) = \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)\text{bel}(\boldsymbol{x}_{t-1})d\boldsymbol{x}_{t-1}$$

Correction Step

$$\text{bel}(\boldsymbol{x}_t) = \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t)\overline{\text{bel}}(\boldsymbol{x}_t)$$

- $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ is Gaussian
- $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is Gaussian
- $\text{bel}(\boldsymbol{x}_0)$ is Gaussian
- $\text{bel}(\boldsymbol{x}_t)$ **stays Gaussian!**

Kalman Filter - Linear Models

$$\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t + \boldsymbol{\epsilon}_t^m$$
$$\boldsymbol{z}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \boldsymbol{\epsilon}_t^o$$

where $\boldsymbol{\epsilon}_t^m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_t)$, $\boldsymbol{\epsilon}_t^o \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t)$.

Linear models with zero mean Gaussian noise!

Kalman Filter - Algorithm

1. Input: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$
2. $\overline{\boldsymbol{\mu}}_t = \boldsymbol{A}_t \boldsymbol{\mu}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t$
3. $\overline{\boldsymbol{\Sigma}}_t = \boldsymbol{A}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{R}_t$
4. $\boldsymbol{K}_t = \overline{\boldsymbol{\Sigma}}_t \boldsymbol{C}_t^T \left(\boldsymbol{C}_t \overline{\boldsymbol{\Sigma}}_t \boldsymbol{C}_t^T + \boldsymbol{Q}_t \right)^{-1}$
5. $\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{C}_t \overline{\boldsymbol{\mu}}_t)$
6. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}_t) \overline{\boldsymbol{\Sigma}}_t$
7. Output: $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$

Steps 2 and 3 are the *Prediction Step*

Steps 4, 5, 6 and 7 are the *Correction Step*

Kalman Filter - Non Linear Models?

What happens if the motion or observation model is not linear?

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{\epsilon}_t^m$$
$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t^o$$

where $\boldsymbol{\epsilon}_t^m \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_t), \boldsymbol{\epsilon}_t^o \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_t)$.

Kalman Filter - Non Linear Models?

What happens if the motion or observation model is not linear?

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{\epsilon}_t^m$$
$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t^o$$

where $\epsilon_t^m \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_t), \epsilon_t^o \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_t)$.

$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ and $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ **are NO longer Gaussians!**

Kalman Filter - Non Linear Models?

What happens if the motion or observation model is not linear?

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{\epsilon}_t^m$$
$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t^o$$

where $\epsilon_t^m \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_t), \epsilon_t^o \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_t)$.

$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ and $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ **are NO longer Gaussians!**

What can we do?

Extended Kalman Filter

When in doubt, linearize!

Extended Kalman Filter

When in doubt, linearize!

$$f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \approx f(\boldsymbol{\mu}_{t-1}, \boldsymbol{u}_t) + \underbrace{\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{\mu}_{t-1} \\ \boldsymbol{u} = \boldsymbol{u}_t}}}_{\boldsymbol{F}_t} (\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1})$$

$$h(\boldsymbol{x}_t) \approx h(\overline{\boldsymbol{\mu}}_t) + \underbrace{\left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\overline{\boldsymbol{\mu}}_t}}_{\boldsymbol{H}_t} (\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t)$$

Extended Kalman Filter

When in doubt, linearize!

$$f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \approx f(\boldsymbol{\mu}_{t-1}, \boldsymbol{u}_t) + \underbrace{\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{\mu}_{t-1} \\ \boldsymbol{u} = \boldsymbol{u}_t}}}_{\boldsymbol{F}_t} (\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1})$$

$$h(\boldsymbol{x}_t) \approx h(\overline{\boldsymbol{\mu}}_t) + \underbrace{\left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\overline{\boldsymbol{\mu}}_t}}_{\boldsymbol{H}_t} (\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t)$$

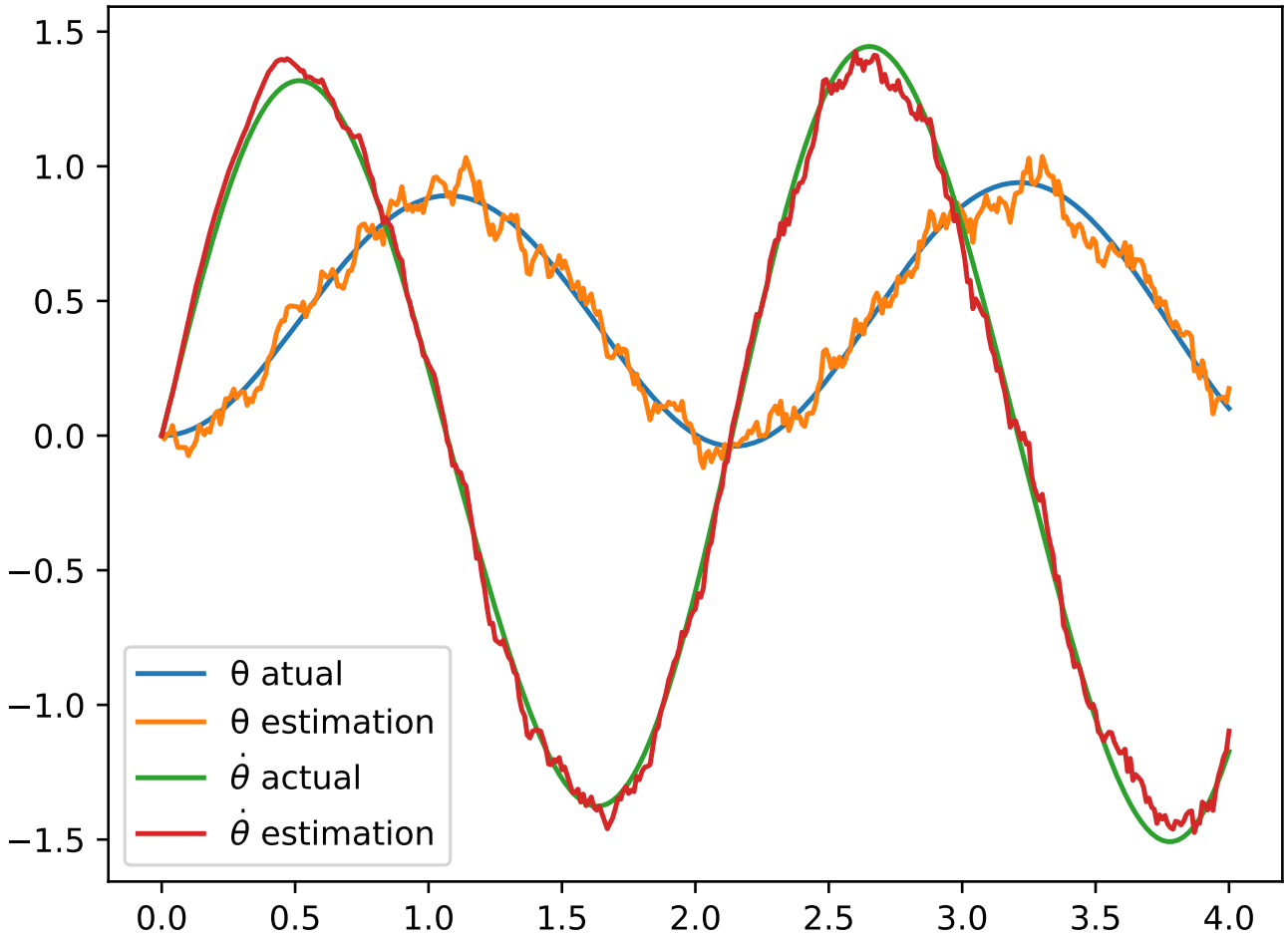
How does this affect our algorithm?

Extended Kalman Filter - Algorithm

1. Input: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$
2. $\overline{\boldsymbol{\mu}}_t = \boldsymbol{f}(\boldsymbol{\mu}_{t-1}, \boldsymbol{u}_t)$
3. $\overline{\boldsymbol{\Sigma}}_t = \boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{F}_t^T + \boldsymbol{R}_t$
4. $\boldsymbol{K}_t = \overline{\boldsymbol{\Sigma}}_t \boldsymbol{H}_t^T \left(\boldsymbol{H}_t \overline{\boldsymbol{\Sigma}}_t \boldsymbol{H}_t^T + \boldsymbol{Q}_t \right)^{-1}$
5. $\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t))$
6. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \overline{\boldsymbol{\Sigma}}_t$
7. Output: $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$

- We use the non-linear functions
- We use $\boldsymbol{F}_t$ instead of $\boldsymbol{A}_t$
- We use $\boldsymbol{H}_t$ instead of $\boldsymbol{C}_t$

Extended Kalman Filter - Code Example



Odometry

- *Odometry* is the process of estimating the robot base body pose $\boldsymbol{x}$ from the joint motions

Odometry

- *Odometry* is the process of estimating the robot base body pose $\boldsymbol{x}$ from the joint motions
- In wheeled robots, we want to estimate the chassis body pose from the wheel motions
- In more concrete terms, we are given $\boldsymbol{x}_t$, Δw_i (change of wheel angle between steps) and we want to estimate $\boldsymbol{x}_{t+1}$

Odometry

- *Odometry* is the process of estimating the robot base body pose $\boldsymbol{x}$ from the joint motions
- In wheeled robots, we want to estimate the chassis body pose from the wheel motions
- In more concrete terms, we are given $\boldsymbol{x}_t$, Δw_i (change of wheel angle between steps) and we want to estimate $\boldsymbol{x}_{t+1}$
- **How can we do this?**

Omnidirectional Mobile Robots - Reminder!

We have:

$$h_i(\theta) = \frac{1}{r_i \cos\gamma_i} \begin{bmatrix} x_i \sin(\beta_i + \gamma_i) - y_i \cos(\beta_i + \gamma_i) \\ \cos(\beta_i + \gamma_i + \theta) \\ \sin(\beta_i + \gamma_i + \theta) \end{bmatrix}^T$$

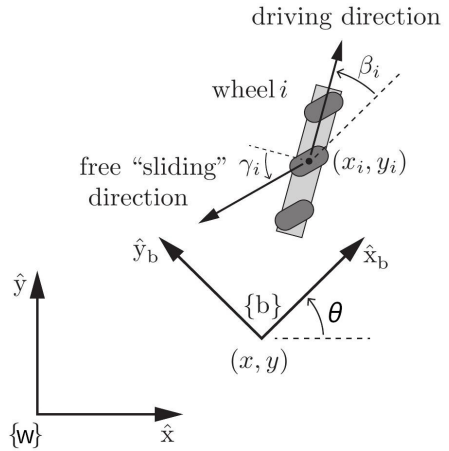
If we have m wheels, we can write something as follows:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} = \boldsymbol{H}(\theta)\dot{\boldsymbol{x}} = \begin{bmatrix} h_1(\theta) \\ \vdots \\ h_m(\theta) \end{bmatrix} \dot{\boldsymbol{x}}$$

Also:

$$\boldsymbol{u} = \boldsymbol{H}(0)\mathcal{V}_b = \begin{bmatrix} h_1(0) \\ \vdots \\ h_m(0) \end{bmatrix} \begin{bmatrix} \omega \\ v_x \\ v_y \end{bmatrix}$$

Independent of θ!



Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

Odometry for Omnidirectional Robots

- u_i are the wheel velocities
- We only have access to the wheel encoder measurements!
- In other words, we have access to the wheel position at time t, w_i^t, and at time $t+1$, w_i^{t+1}
- **How can we find the next body pose x_{t+1}?**

Odometry for Omnidirectional Robots

- u_i are the wheel velocities
- We only have access to the wheel encoder measurements!
- In other words, we have access to the wheel position at time t, w_i^t, and at time $t+1$, w_i^{t+1}
- **How can we find the next body pose $\boldsymbol{x}_{t+1}$?**
- $u_i \approx \frac{w_i^{t+1} - w_i^t}{dt}$
- **Thus:** $\overline{\mathcal{V}}_b = \boldsymbol{H}(0)^{-1}\boldsymbol{u}$

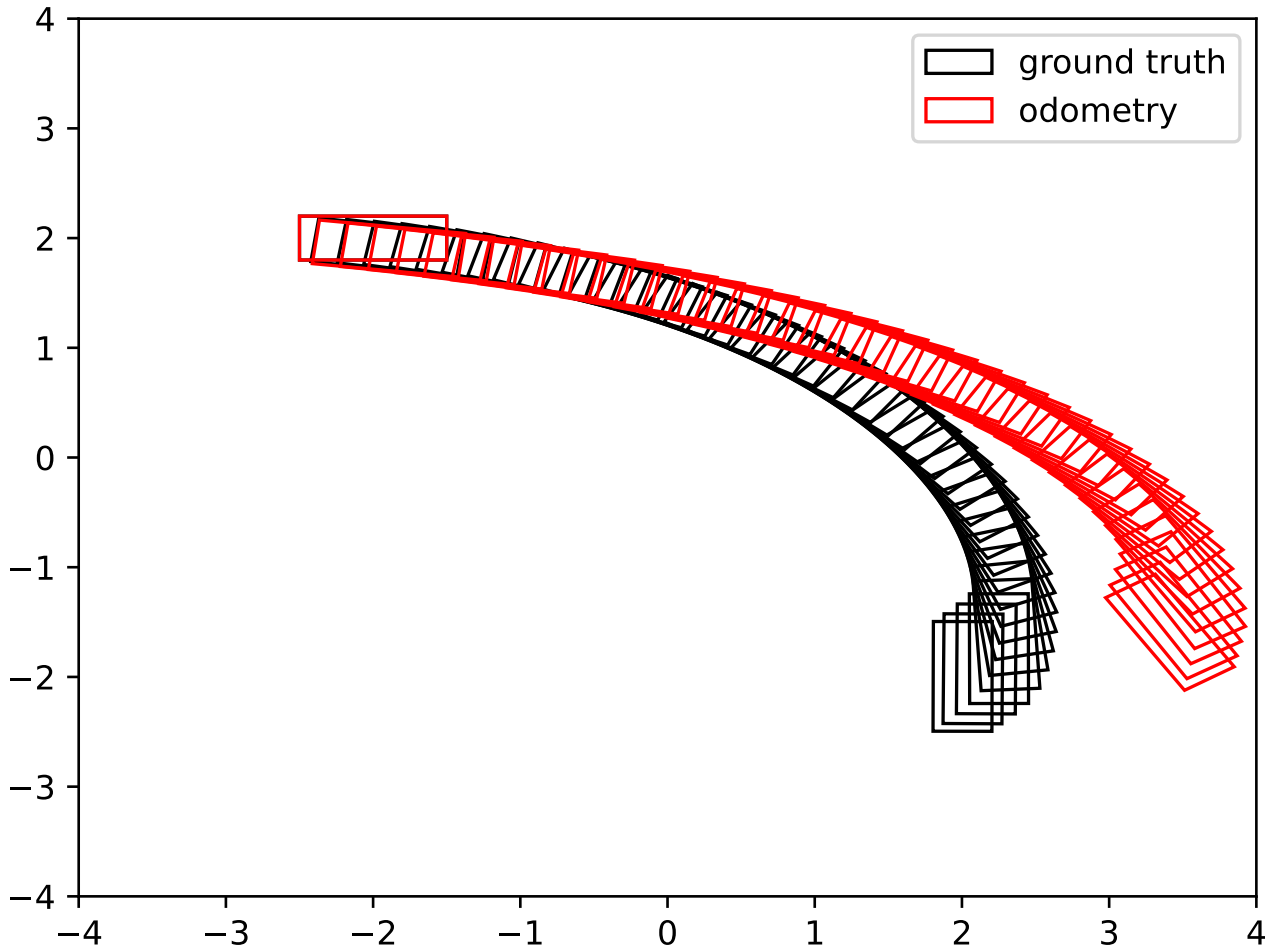
Odometry for Omnidirectional Robots

- u_i are the wheel velocities
- We only have access to the wheel encoder measurements!
- In other words, we have access to the wheel position at time t, w_i^t, and at time $t+1$, w_i^{t+1}
- **How can we find the next body pose $\boldsymbol{x}_{t+1}$?**
- $u_i \approx \frac{w_i^{t+1} - w_i^t}{dt}$
- **Thus:** $\overline{\mathcal{V}}_b = \boldsymbol{H}(0)^{-1}\boldsymbol{u}$
- Now we can integrate using the previous $\boldsymbol{x}_t$ and $\overline{\mathcal{V}}_b$

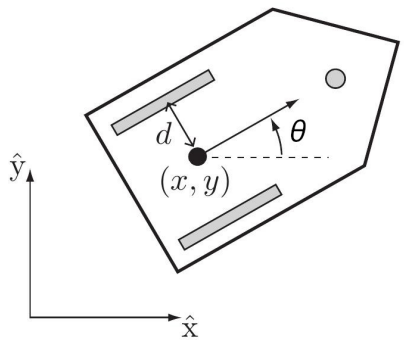
Odometry for Omnidirectional Robots

- u_i are the wheel velocities
- We only have access to the wheel encoder measurements!
- In other words, we have access to the wheel position at time t, w_i^t, and at time $t+1$, w_i^{t+1}
- **How can we find the next body pose $\boldsymbol{x}_{t+1}$?**
- $u_i \approx \frac{w_i^{t+1} - w_i^t}{dt}$
- **Thus:** $\overline{\mathcal{V}}_b = \boldsymbol{H}(0)^{-1}\boldsymbol{u}$
- Now we can integrate using the previous $\boldsymbol{x}_t$ and $\overline{\mathcal{V}}_b$
- $\boldsymbol{H}(0)$ is a $m \times 3$ matrix! Not always invertible! We use the pseudo-inverse: $\boldsymbol{H}(0)^{\dagger}$

Odometry - Code Example



Differential Drive Robots - Reminder!



Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

Kinematics:
The state of a differential drive robot is $\boldsymbol{x} = [\theta, x, y, \theta_L, \theta_R]^T$. And the kinematic equations are given by:

$$\begin{bmatrix} \dot{\theta} \\ \dot{x} \\ \dot{y} \\ \dot{\theta_L} \\ \dot{\theta_R} \end{bmatrix} = \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$$

where (θ, x, y) is the pose of the robot in world space, r are the wheel radii, d is the distance of the wheels from the robot center, θ_L, θ_R are the wheel rotation angles (values from encoder of the motors) and u_L, u_R are the wheel angular velocities (commands).

Odometry for Differential Drive Robots

- u_L, u_R are the wheel angular velocities (commands)
- We only have access to the wheel encoder measurements!
- $u_i \approx \frac{\theta_i^{t+1} - \theta_i^t}{dt}$ for $i \in \{L, R\}$
- **Thus:** $\overline{\mathcal{V}}_w = \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$
- But this is in world space (**why?**)! How can we transform this in body space?

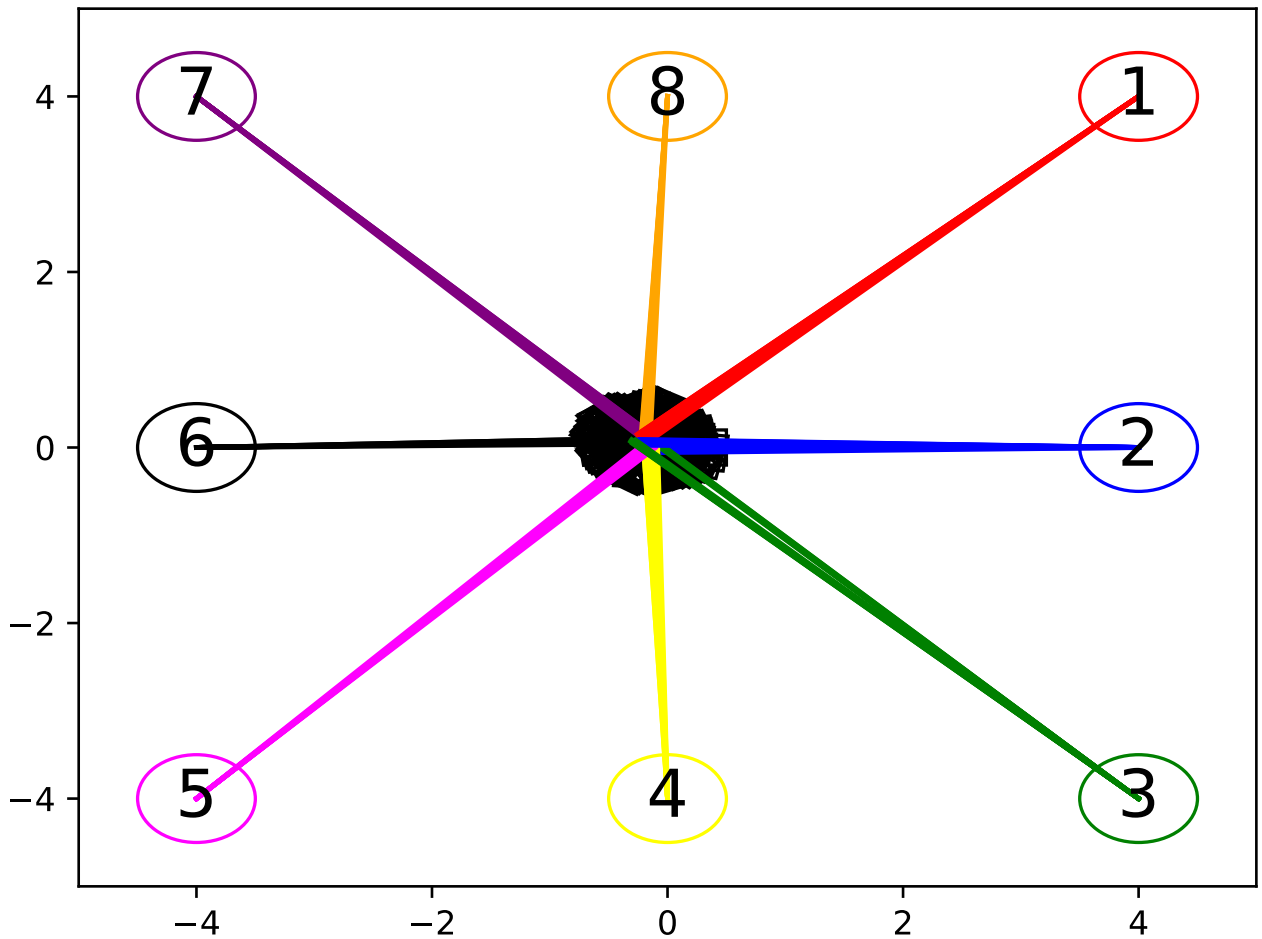
Odometry for Differential Drive Robots

- u_L, u_R are the wheel angular velocities (commands)
- We only have access to the wheel encoder measurements!
- $u_i \approx \frac{\theta_i^{t+1} - \theta_i^t}{dt}$ for $i \in \{L, R\}$
- **Thus:** $\overline{\mathcal{V}}_w = \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$
- But this is in world space (**why?**)! How can we transform this in body space?
- We set $\theta = 0$ (**why?**) and thus $\overline{\mathcal{V}}_b = \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$

2D Range-Bearing Observations

- **Sensors:**
  - Range sensors
  - LIDARs
  - ...
- We have K beams: $\boldsymbol{z}_t = \left[z_t^1, z_t^2, \cdots, z_t^K\right]^T$
- $z_t^i = \left[r_t^i, \phi_t^i\right]^T$
- r_t^i is distance to the object, ϕ_t^i relative angle to the sensor/robot

Range-Bearing Sensors - Code Example



SLAM Problem

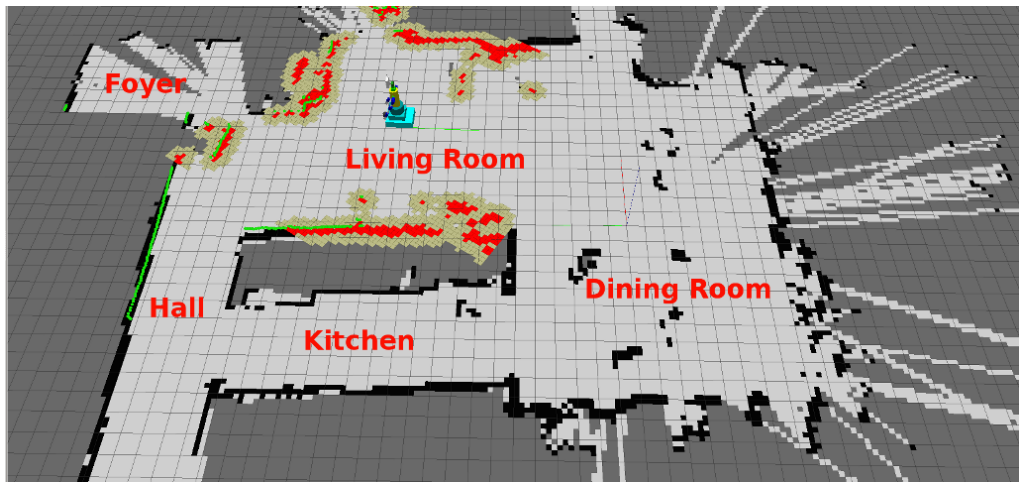
Simultaneous Localization and Mapping (SLAM):

- Build a map and locate the robot in the map at the same time

SLAM Problem

Simultaneous Localization and Mapping (SLAM):

- Build a map and locate the robot in the map at the same time
- We are given control inputs $\boldsymbol{u}_{1:K}$ and observations $\boldsymbol{z}_{1:K}$
- We want to estimate the *map of the environment* $\boldsymbol{m}$ and the trajectory of the robot $\boldsymbol{x}_{0:K}$



Source: `https://sifrobot.com/clinical-apps/simultaneous-localization-and-mapping-slam/`

EKF SLAM

EKF SLAM:

- What if we include the map inside the state $\boldsymbol{x}$?
- For example, $\boldsymbol{x} = \left[\theta, x, y, m_x^1, m_y^1, \cdots, m_x^N, m_y^N\right]^T$

EKF SLAM

EKF SLAM:

- What if we include the map inside the state $\boldsymbol{x}$?
- For example, $\boldsymbol{x} = \left[\theta, x, y, m_x^1, m_y^1, \cdots, m_x^N, m_y^N\right]^T$
- N landmarks
- A landmark is an *"important"* place/descriptor of the environment

EKF SLAM

EKF SLAM:

- What if we include the map inside the state $\boldsymbol{x}$?
- For example, $\boldsymbol{x} = \left[\theta, x, y, m_x^1, m_y^1, \cdots, m_x^N, m_y^N\right]^T$
- N landmarks
- A landmark is an *"important"* place/descriptor of the environment
- Now we can use EKF *almost as is!!!*

EKF SLAM

EKF SLAM:

- What if we include the map inside the state $\boldsymbol{x}$?
- For example, $\boldsymbol{x} = \left[\theta, x, y, m_x^1, m_y^1, \cdots, m_x^N, m_y^N\right]^T$
- N landmarks
- A landmark is an *"important"* place/descriptor of the environment
- Now we can use EKF *almost as is!!!*
- **Assumption:** We know which observation corresponds to which landmark!
- **Belief representation:**

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{m} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{x}_R\boldsymbol{x}_R} & \boldsymbol{\Sigma}_{\boldsymbol{x}_R\boldsymbol{m}} \\ \boldsymbol{\Sigma}_{\boldsymbol{m}\boldsymbol{x}_R} & \boldsymbol{\Sigma}_{\boldsymbol{m}\boldsymbol{m}} \end{bmatrix}\right)$$

EKF SLAM Algorithm

We basically perform EKF! We can make it faster/better by exploiting the problem structure:

- The *Prediction Step* only affects the robot variables!
- The *Correction Step* initializes new landmarks and/or updates already visited ones!

High-Level Steps

1. Predict next state
2. Predict observations (only for visible landmarks!)
3. Compare predicted observations with real ones (*data association*)
4. Update EKF

EKF SLAM Algorithm (2)

High-Level Pseudocode

1. Compute $\overline{\mu}_t$ only for the robot states given the motion model
2. Compute $\overline{\Sigma}_t$ as

$$\overline{\Sigma}_t = \begin{bmatrix} F_t \Sigma_{x_R x_R} F_t^T + R_t & F_t \Sigma_{x_R m} \\ \Sigma_{m x_R} F_t^T & \Sigma_{mm} \end{bmatrix}$$

3. For each visible landmark, compute *expected* observation: $h(\overline{\mu}_t)$
4. Compute H_t and proceed with the rest of EKF!

What if the landmarks are unknown!?

EKF SLAM Algorithm (2)

High-Level Pseudocode

1. Compute $\overline{\mu}_t$ only for the robot states given the motion model
2. Compute $\overline{\Sigma}_t$ as

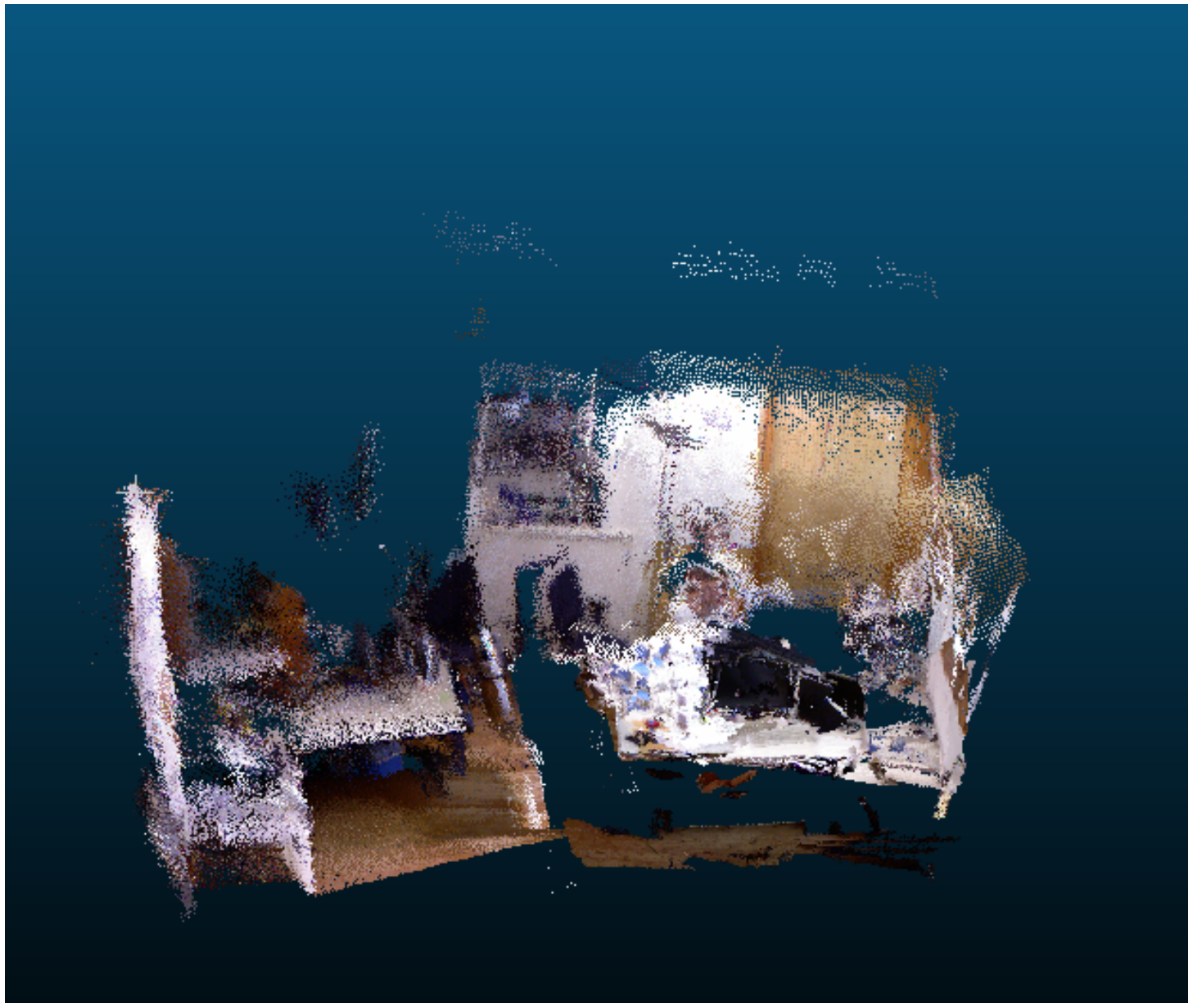
$$\overline{\Sigma}_t = \begin{bmatrix} F_t \Sigma_{x_R x_R} F_t^T + R_t & F_t \Sigma_{x_R m} \\ \Sigma_{m x_R} F_t^T & \Sigma_{mm} \end{bmatrix}$$

3. For each visible landmark, compute *expected* observation: $h(\overline{\mu}_t)$
4. Compute H_t and proceed with the rest of EKF!

What if the landmarks are unknown!?

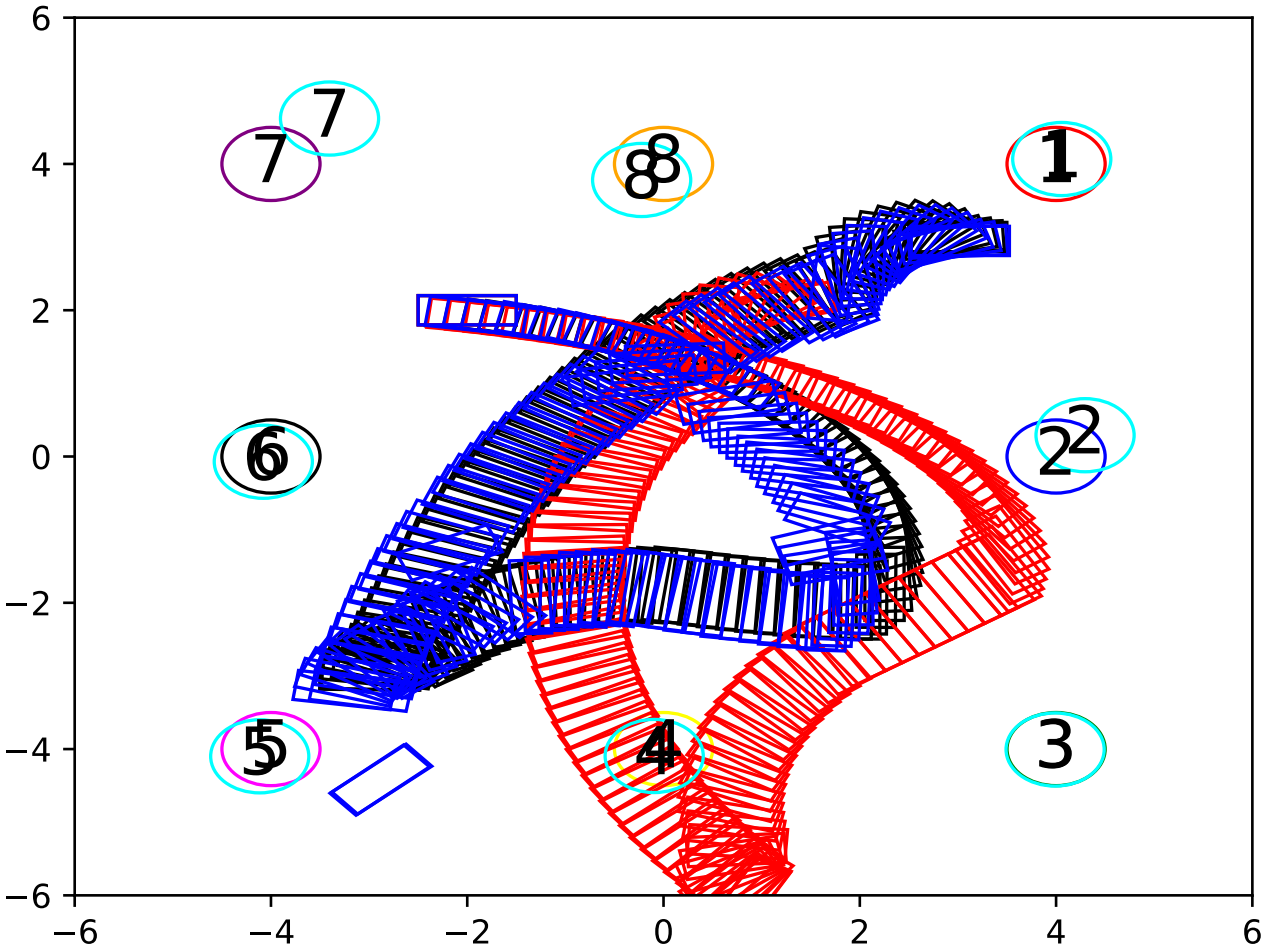
1. We need to keep a "dictionary" of landmark descriptions
2. We add a new description for new landmarks that have very low probability of belonging to an existing one

EKF SLAM - Results

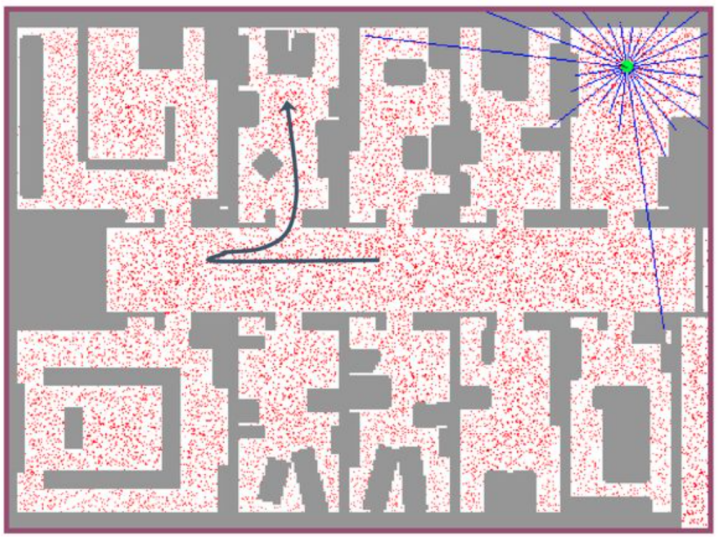


Source: `https://github.com/sjnarmstrong/ekf-slam`

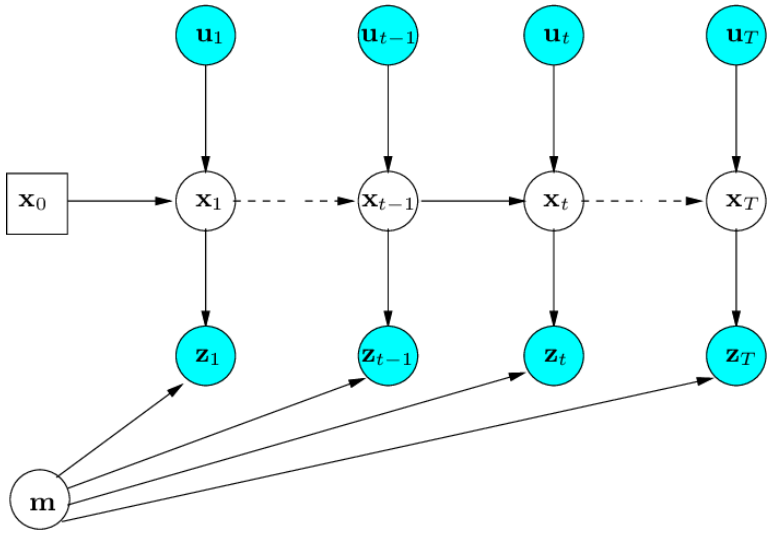
EKF SLAM - Code Example



Other solutions to SLAM?



Particle Filters



Graph-based Optimization[1]

[1] Source: *Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W., 2010. A tutorial on graph-based SLAM. IEEE Intelligent Transportation Systems Magazine*

State-of-the-art SLAM

Bibliography

Chapter 4 and Section 8.2 from **State Estimation for Robotics**, *Timothy D. Barfoot*, 2022, Cambridge University Press. ebook

Sections 3.3, 4.2, 6.3, 6.6, and 10.2 from **Probabilistic Robotics**, *Sebastian Thrun, Wolfram Burgard, Dieter Fox*, 2005, The MIT Press. online

Thank you

- **Any Questions?**
- **Office Hours**:
  - **Tue-Wed** *(10:00-12:00)*
  - 24/7 by email (`costashatz@upatras.gr`, subject: *ECE_RSI_AM*)
- **Material and Announcements**

Laboratory of Automation & Robotics