Module D: SLAM& Pure Pursuit

Part 1. Introduction to SLAM

Part 2: Particle Filter & SLAM

Part 3. Cartographer & Pure Pursuit

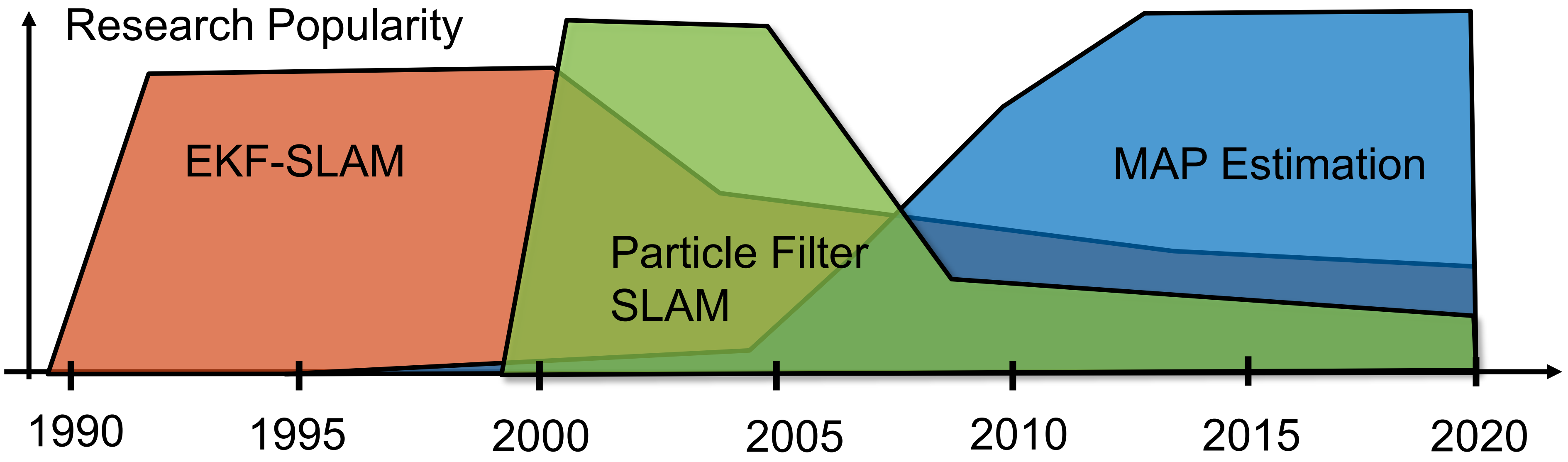
References:

https://f1tenth.org and UPenn ESE 680 Slides

https://google-cartographer.readthedocs.io/en/latest/

Recall: Major SLAM Algorithms

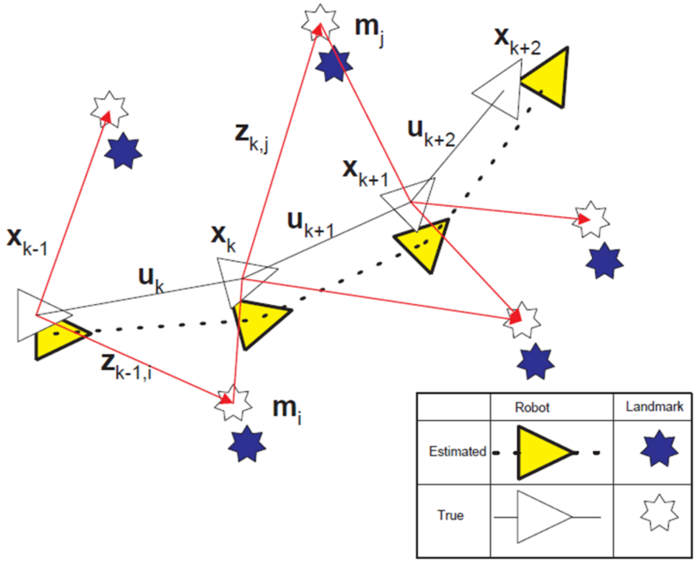
- Three major SLAM algorithms:



- Typical SLAM procedures consist of multiple iterations through:
  - Localization and Mapping: by motion update and observation updates
  - Scan matching to sub-maps or data association to landmarks
  - Loop closure: build global map and register the re-entered scenes

Math Formula of SLAM

- Definitions:
  - History of robot states: $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\mathbf{X}_{0:k-1}, \mathbf{x}_k\}$
  - History of control inputs: $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$
  - Set of landmarks: $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n\}$
  - Set of observations: $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$



- Probabilistic Models
  - Observation model: $P(\boldsymbol{z}_k|\mathbf{x}_k, \boldsymbol{m})$
  - Motion model: $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \boldsymbol{u}_k)$
  - Time update: $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)\, d\mathbf{x}_{k-1}$
  - Measurement update: $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})}$
- SLAM solves for joint conditional probability: $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$

EKF-SLAM

- EKF-SLAM formulation:
  - Motion model: $P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) \Longleftrightarrow \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$
  - Observation model: $P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{m}) \Longleftrightarrow \mathbf{z}(k) = \mathbf{h}(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k,$
- EKF-SLAM solution:
  - Time update (prediction):

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \\ \mathbf{P}_{xx,k|k-1} &= \nabla\mathbf{f}\ \mathbf{P}_{xx,k-1|k-1}\nabla\mathbf{f}^T + \mathbf{Q}_k \end{aligned}$$

  - Observation update (correction):

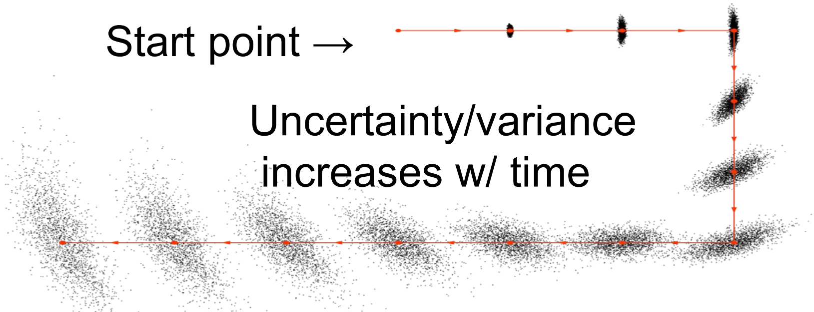
$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{m}}_{k-1} \end{bmatrix} + \mathbf{W}_k \overbrace{\left[\mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \hat{\mathbf{m}}_{k-1})\right]}^{\text{innovation}}$$

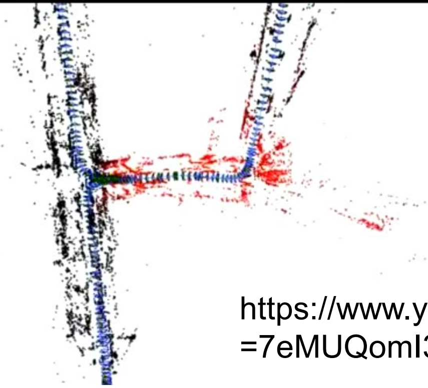
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^T$$

where

$$\mathbf{S}_k = \nabla\mathbf{h}\mathbf{P}_{k|k-1}\nabla\mathbf{h}^T + \mathbf{R}_k$$

Kalman Gain $\quad \mathbf{W}_k = \mathbf{P}_{k|k-1}\nabla\mathbf{h}^T\mathbf{S}_k^{-1}$

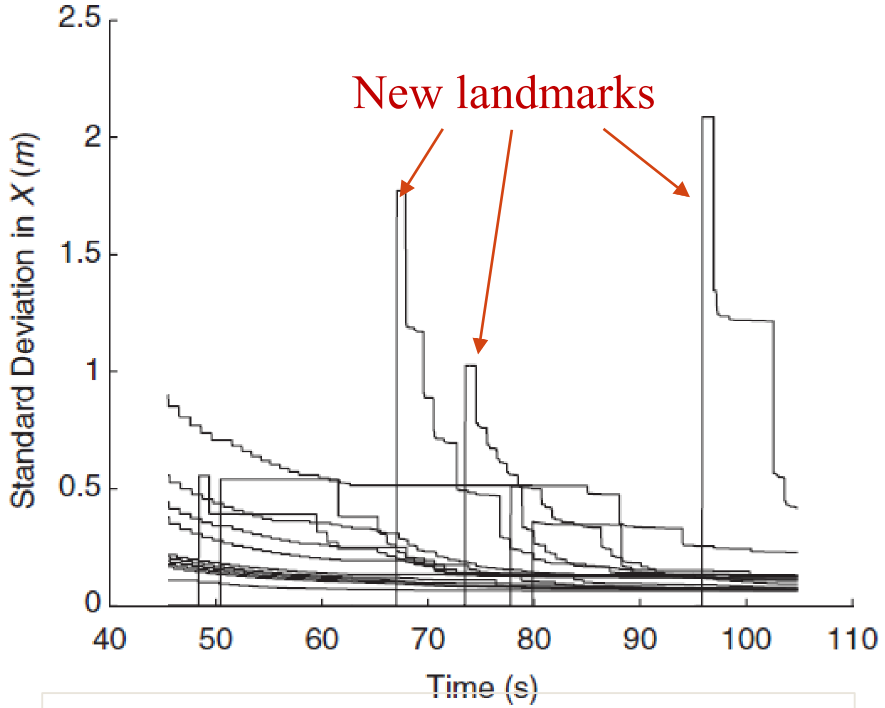




https://www.youtube.com/watch?v=7eMUQomI36Q

Challenges of EKF-SLAM

- Convergence:
  - Convergence is measured by the determinant of covariance matrices. Monotonic convergence is shown in landmark locations.
- Computational Complexity:
  - The prediction step: $O(N^2)$ with N – dimension of states
  - The correction step: $O(M^{2.8})$ with M – dimension of landmarks
  - Other variants aiming to reduce complexity:
    - Unscented KF (UKF) SLAM: sparse EKF
    - Extended Information Filter (EIF)-SLAM
- Data Association:
  - Difficult to correctly associate observations to landmarks.
  - Loop closure after a large traverse is particularly difficult.
- Nonlinearity and non-Gaussian distribution:
  - Nonlinear motion and observation models are linearized in EKF-SLAM → degrade the convergence and consistency dramatically.
  - Robot pose is typically non-Gaussian distribution → divergence.

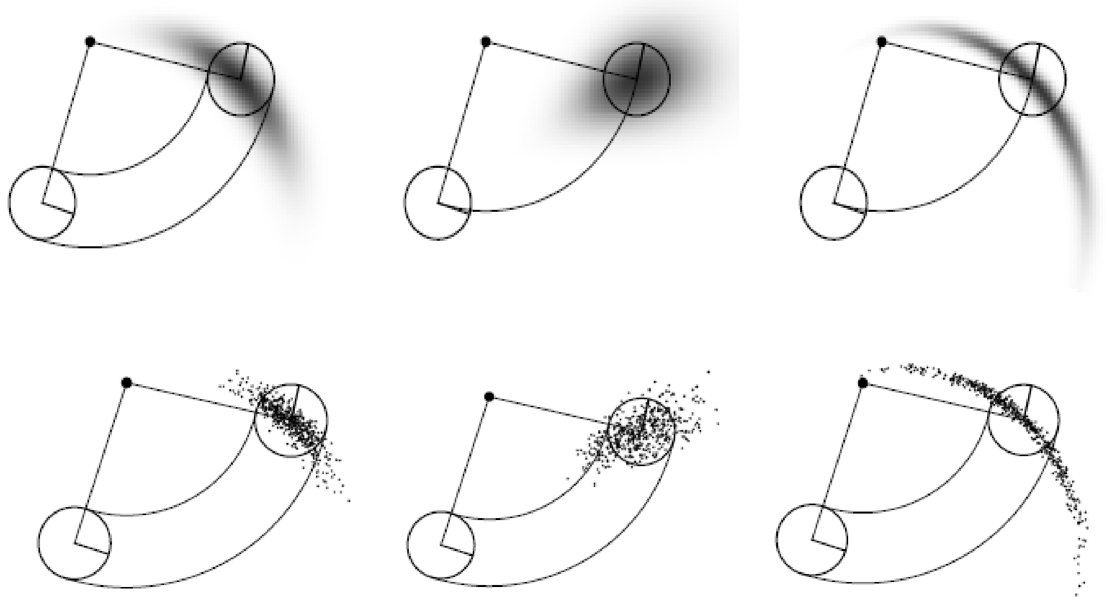


The convergence in landmark uncertainty. A landmark is initially observed with uncertainty inherited from the robot location and observation, and the standard deviations will reduce monotonically over time.

Figure from: G. Dissanayake et. al. IEEE Trans. Robotics and Automation,17(3):229-241, 2001.

Why Particle Filter SLAM?

- Why particle filters are good for SLAM?
  - Especially effective for non-Gaussian models
  - In many indoor environment, map models:
    - Gaussian distribution: ~ 75%
    - Non-Gaussian single modal dist.: 10 – 20 %
    - Non-Gaussian multi-modal dist.: 5-10%
  - PF-SLAM uses particles to resample the non-Gaussian distributions.
- Types of particle filter SLAM:
  - MCL: Monte Carlo Localization: feature or grid based
  - FastSLAM: ver 1.0, 2.0, and grid-based FastSLAM



Different motion models and their sampling when the robot makes a 90 degree turn: noises in angular velocity and accelerometer measurements are Gaussian, but the resulting poses are non-Gaussian.
Figure from: S. Thrun et. al. Probablistic Robotics, 2005. Chapter 5.

What is a Particle-Filter?

- What is a particle filter?
  - Denote $\tau(x)$ -- target distribution, $\pi(x)$ -- proposal distribution
  - Samples of the proposal distribution are called particles, denoted as $\chi_p = \{ \mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \ldots, \mathrm{x}^{(J)} \}$
  - A particle filter is to obtain samples χ_τ for the target distribution from the samples χ_p of the proposal distribution. This is achieved by sequential importance sampling (SIS): resample with replacement and their importance weights

$$w^{(j)} = \frac{\tau(x^{(j)})}{\pi(x^{(j)})}, \qquad j = 1, 2, \ldots, J$$

  - Importance weights are often calculated or derived from the relationship between $\tau(x)$ and $\pi(x)$

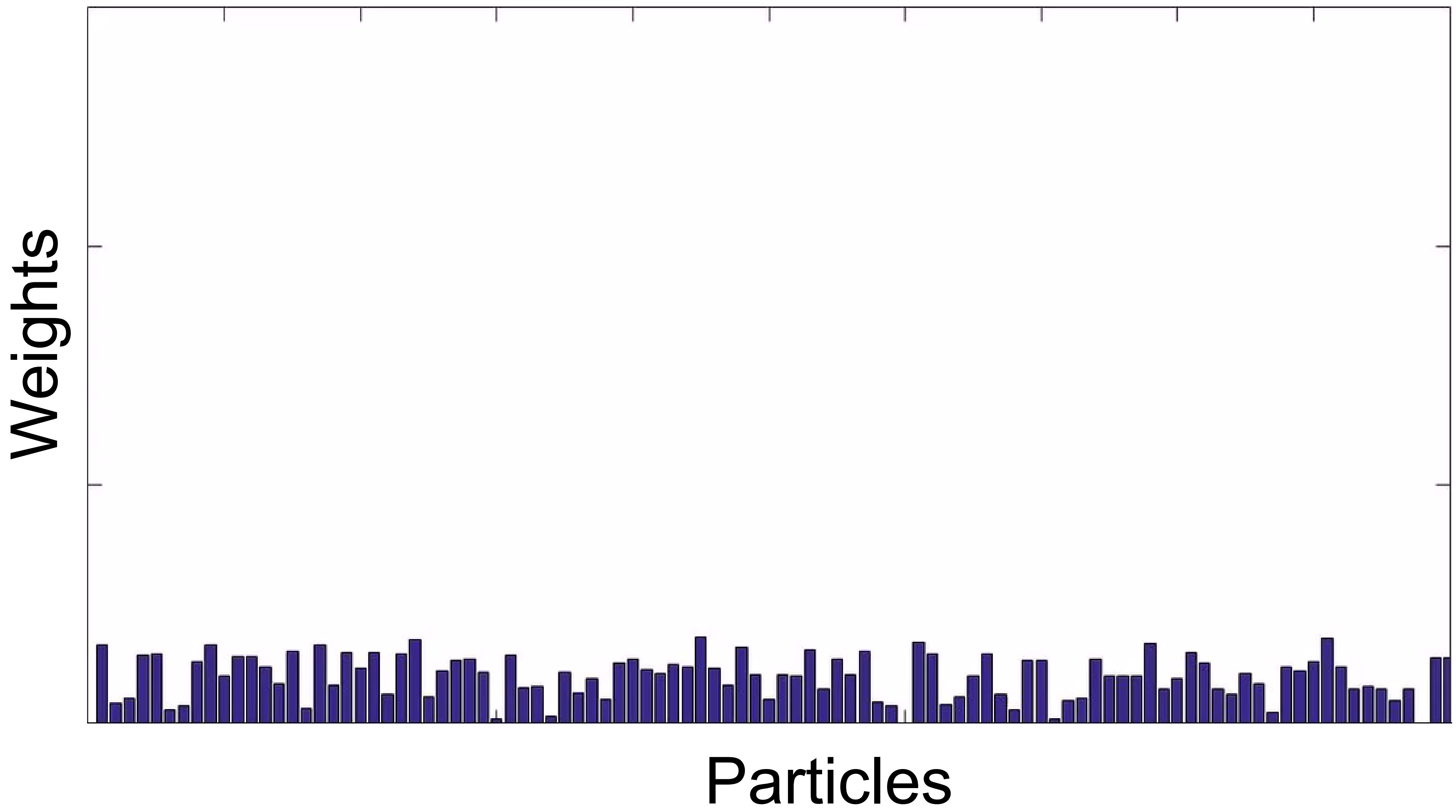
The PF algorithm consists of three steps:

1. Sample the proposal distribution
2. Compute importance weights
3. Resample by SIS

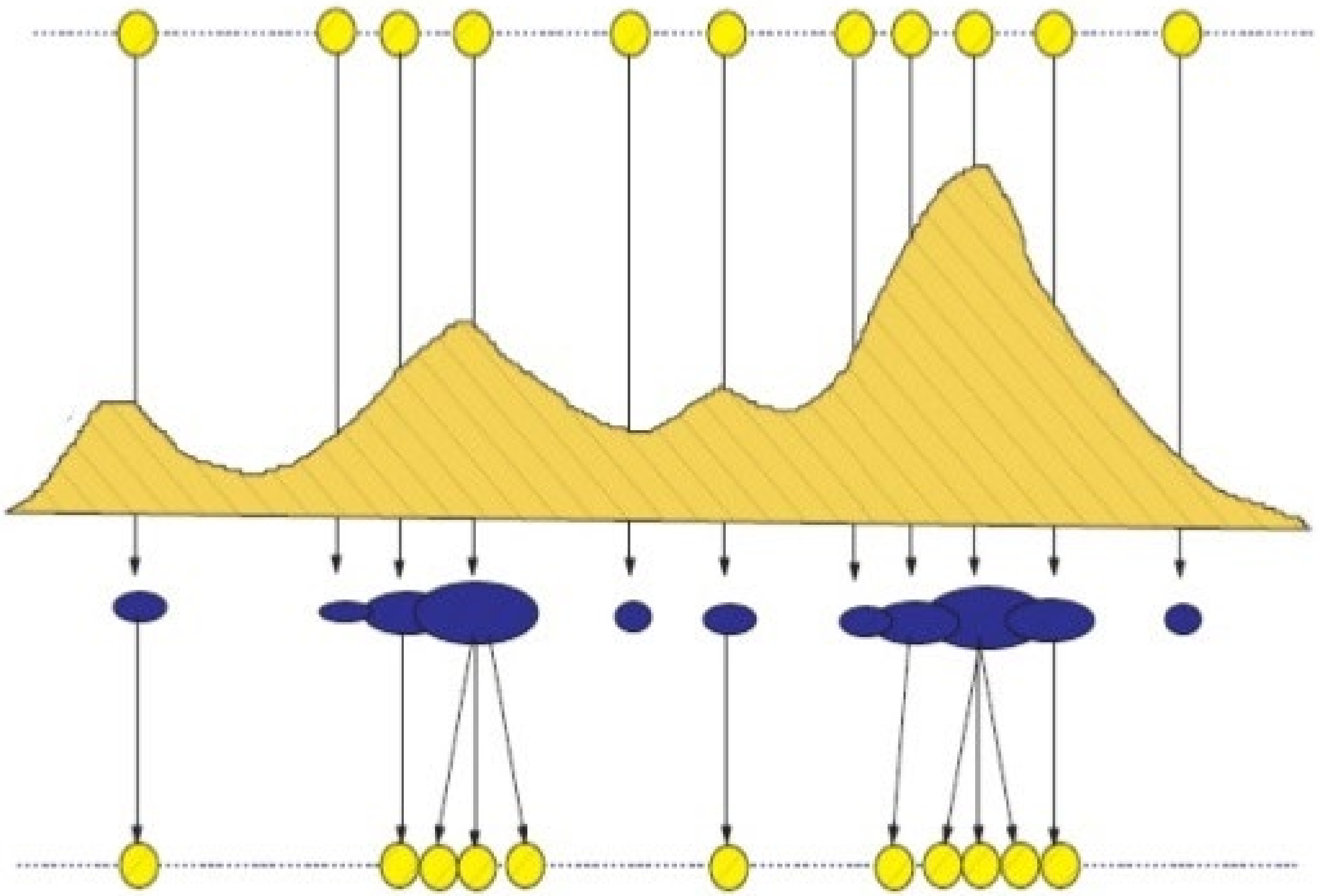
Psuedo-code: ParticleFilter(, ,)

```
    χ_t = χ = ∅
    for j = 1 : J do
        sample x^(j) ~ π(x)
        compute   w^(j)
        χ_t = { X_t,  [x^(j), w^(j)] }
    endfor
%resampling
    for j = 1 : J do
        draw  x^(j) with probability ∝ w^(j)
        add x^(j) to χ
    endfor
    return χ
```

Particles and Importance Weights



Particle Resampling



Original Particles

Target Distribution

Importance Weights

Resampling

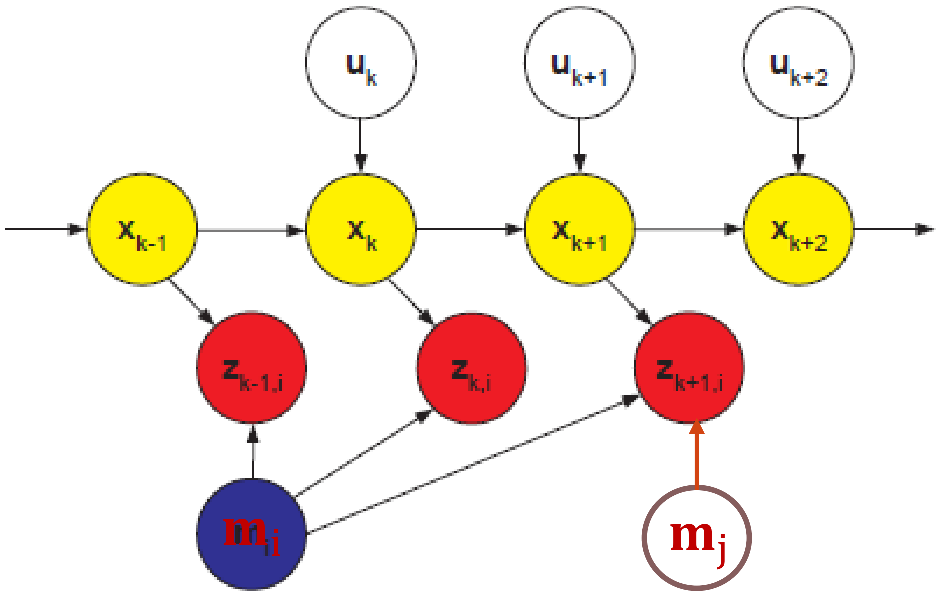
Applying PF to SLAM

- SLAM is to solve for joint conditional probability: $P(\mathbf{x}_{\mathrm{k}}, \mathbf{m} | \mathbf{Z}_{0:\mathrm{k}}, \mathbf{U}_{0:\mathrm{k}}, \mathbf{x}_0)$
  - Direct resampling is difficult due to high dimensions of robot states and landmarks
  - Factor the joint pdf into conditional pdf :

    $$P(\mathrm{X}_{0:k}, \mathrm{m} \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{x}_0)$$
    $$= P(\mathrm{m} \mid \mathrm{X}_{0:k}, \mathrm{Z}_{0:k}) P(\mathrm{X}_{0:k} \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{x}_0).$$
  - Use PF to sample the trajectory $\mathbf{X}_{0:\mathrm{k}}$ and compute their weights $\{w_k^{(i)}, \mathrm{X}_{0:k}^{(i)}\}_i^N$
  - Given trajectory particles, the landmarks are independent from each other (Rao-Blackwellisation).

    $$P(\mathbf{m} \mid X_{0:k}^{(i)}, \mathbf{Z}_{0:k}) = \prod_j^M P(\mathbf{m}_j \mid X_{0:k}^{(i)}, \mathbf{Z}_{0:k})$$

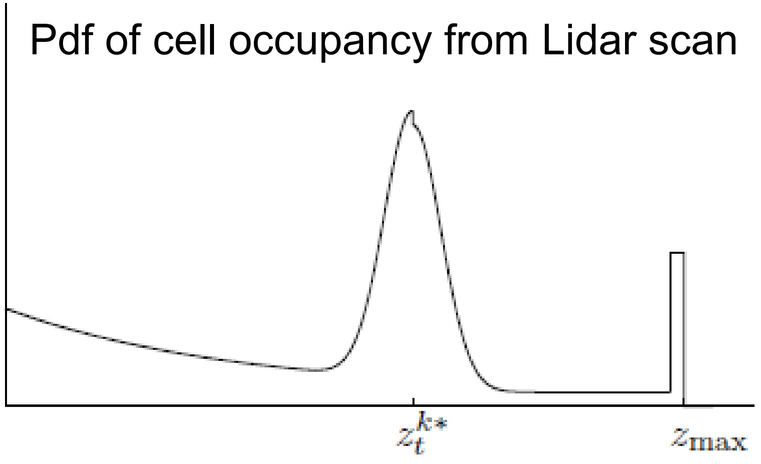
→ reduce complexity by computing M low-dimensional EKFs



Rao-Blackwellisation: A graphical model of SLAM: if the history of pose states are known exactly, then the map state $\mathbf{m}_i$ is independent from $\mathbf{m}_j$ → separate graph for each landmark.
Figure from: G. Dissanayake et. al. IEEE Trans. Robotics and Automation,17(3):229-241, 2001.

Hybrid MCL- used in simulator

- Proposal distribution: $x_k^{(j)} \sim P(\mathbf{x}_k|\mathbf{x}_{k-1}, \boldsymbol{u}_k)$ motion model → prediction step
- Importance weights: $w_k^{(j)} \propto P(\mathbf{z}_k|\mathbf{x}_k, \boldsymbol{m})$ observation model → correction step
- Resample and store over time [0:k] to get sample of $P(\mathrm{X}_{0:k} \mid \mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{x}_0)$.
- Compute map distribution by EKF. If applied to grid-maps:
  - Cell occupancies are independent from each other, occupied/empty prob. by log-likelihood ratio



From Thrun 2005 Chapter 5.

https://www.youtube.com/watch?v=h98sFFNrwa8

Feature-Based FastSLAM

LEHIGH UNIVERSITY

FastSLAM 1.0:

- Proposal distribution: $\mathbf{x}_k^{(j)} \sim P(\mathbf{x}_\mathrm{k} | \mathbf{x}_{\mathrm{k}-1}^{(j)}, \boldsymbol{u}_\mathrm{k})$
  motion model with past trajectory particles
- Trajectory particles: $\mathbf{X}_{0:k}^{(i)} \triangleq \{\mathbf{X}_{0:k-1}^{(i)}, \mathbf{x}_k^{(i)}\}$
- Importance weights:

$$\mathrm{w}_k^{(j)} \propto P\left(\boldsymbol{z}_\mathrm{k} \middle| \mathbf{X}_{\mathbf{0}:\mathrm{k}}^{(j)}, \boldsymbol{Z}_{\mathbf{0}:\boldsymbol{k}}\right)$$

  Can be derived from joint-pdf/motion-model

  Can be updated iteratively;

- Resample to keep the most probable trajectories.
- Performs well with feature based SLAM.

FastSLAM 2.0:

- Proposal distribution:

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{X}_{0:k-1}^{(i)}, \mathbf{Z}_{0:k}, \mathbf{u}_k)$$

motion model with past trajectory particles and observation history

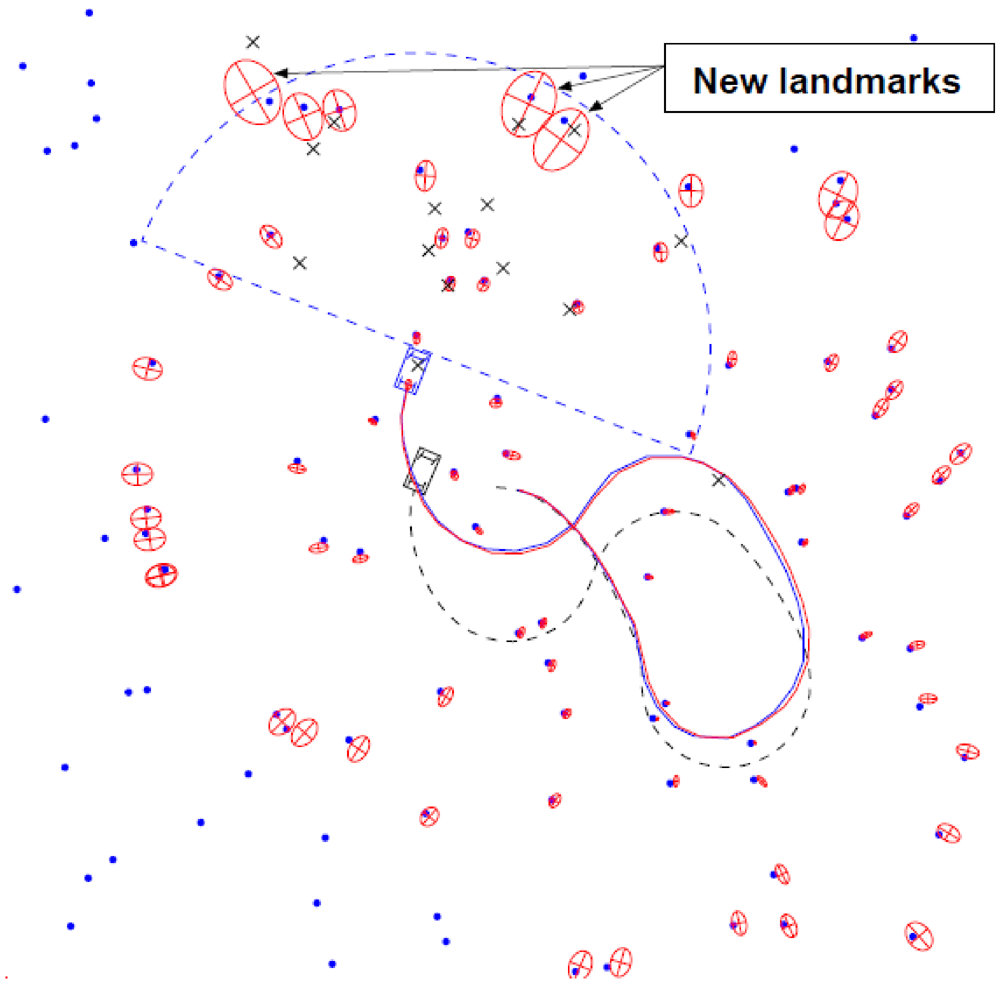
- Importance weights:

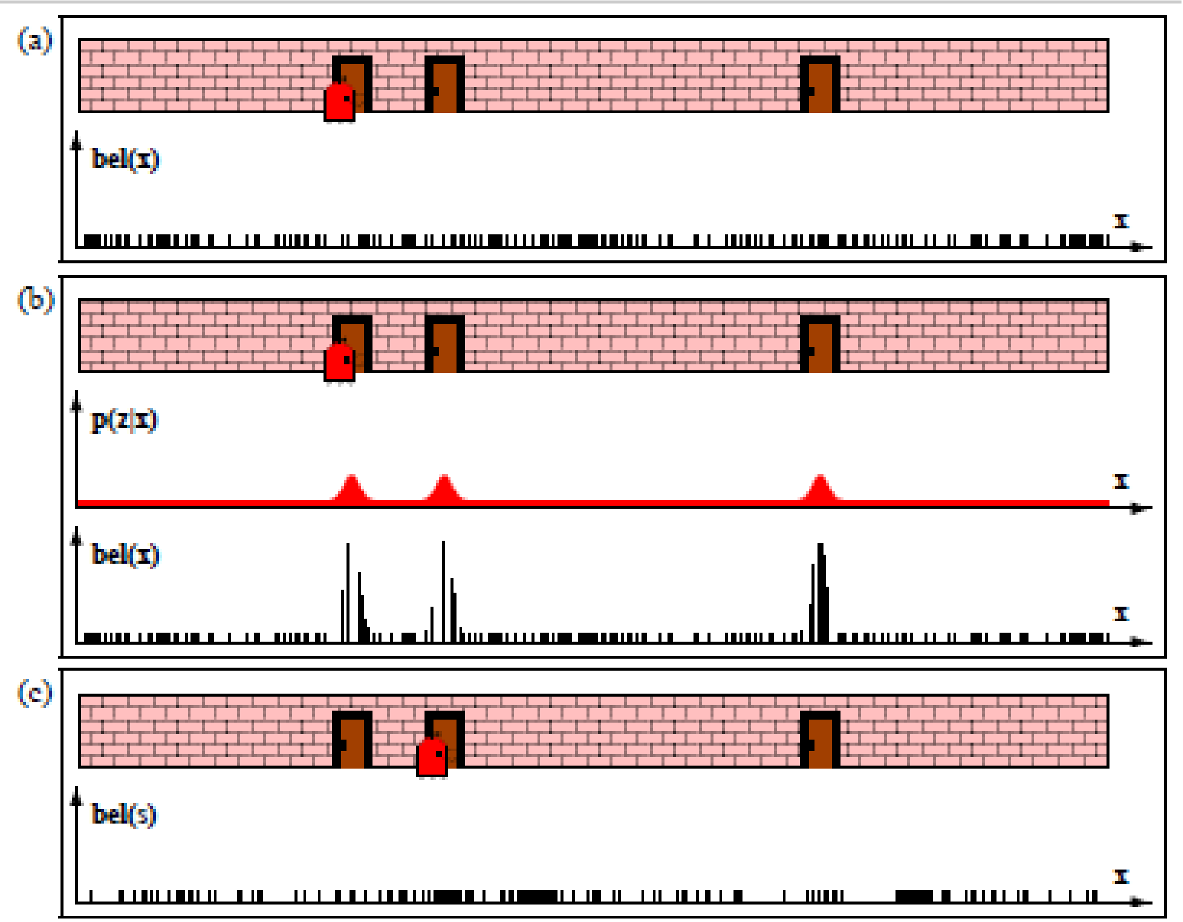
$$w_k^{(i)} = w_{k-1}^{(i)} \frac{P(\mathbf{z}_k \mid \mathbf{X}_{0:k}^{(i)}, \mathbf{Z}_{0:k-1}) P(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{u}_k)}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{X}_{0:k-1}^{(i)}, \mathbf{Z}_{0:k}, \mathbf{u}_k)}$$

  Updated iteratively

- Resample to keep the most probable trajectories.

Performance of FastSLAM 2.0

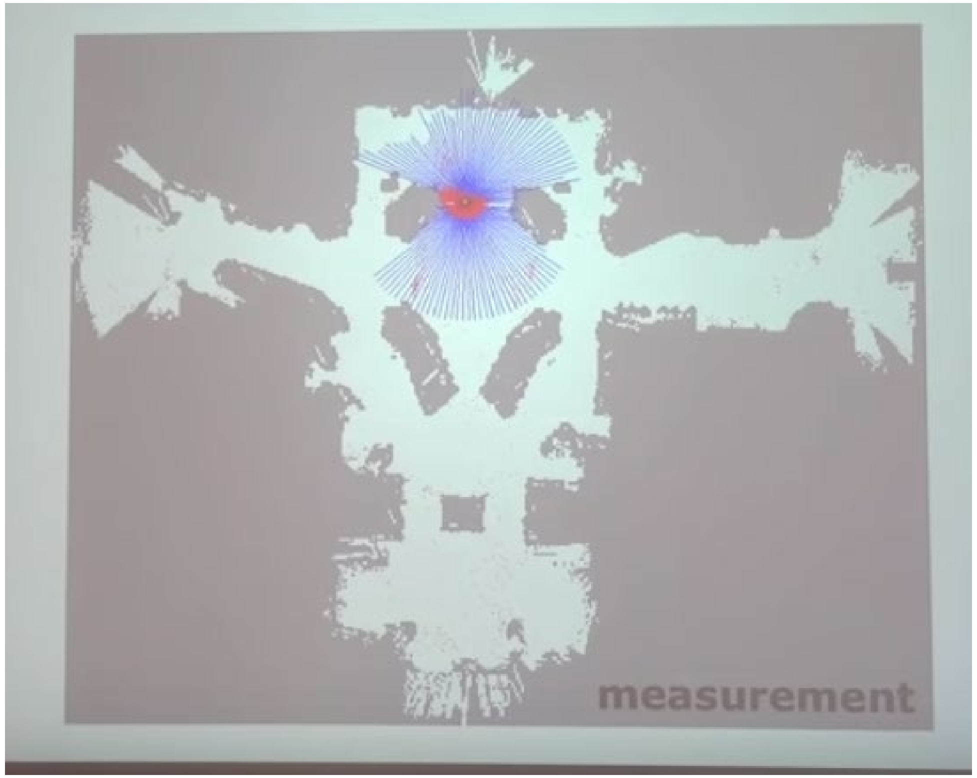




Figures are from Thrun 2005 Chapter 8.

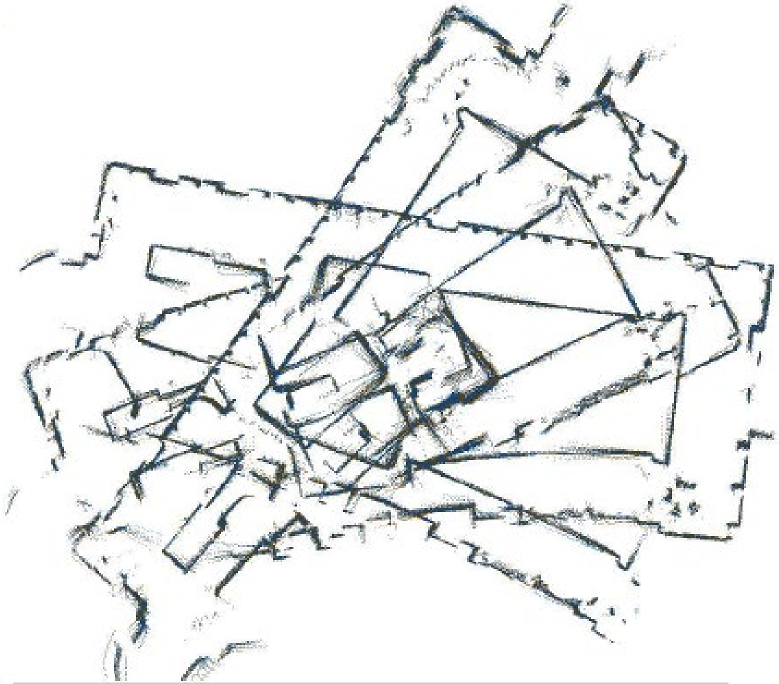
Grid-Based FastSLAM

- Need scan matching to align submaps

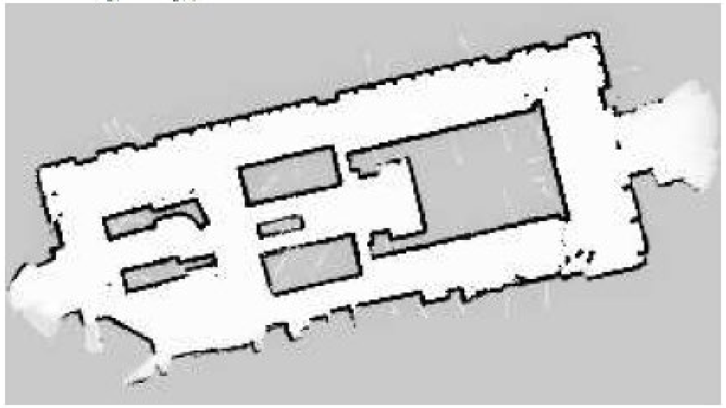


Grid-based PF SLAM example:
https://youtu.be/eAqAFSrTGGY?t=1635

(a)



(b)



References for Particle Filter SLAM

LEHIGH UNIVERSITY

- S. Thrun, D. Fox, W. Burgard and F. Dellaert. "Robust Monte Carlo Localization for Mobile Robots." Elsevier Artificial Intelligence Journal. 2001. https://doi.org/10.1016/S0004-3702(01)00069-8
- S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics." MIT Press, 2005. Chapter 4 & 8.
- M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. "Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," International Joint Conference on Artificial Intelligence, pp. 1151-1156, 2003.
- Dirk H¨ahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun, "An Efficient FastSLAM Algorithm for Generating Mapsof Large-Scale Cyclic Environments from Raw Laser Range Measurements," IEEE International Conference on Robotics and Automation, 2003.
- A. Doucet and A. M. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen years later," https://www.stats.ox.ac.uk/~doucet/doucet_johansen_tutorialPF2011.pdf
- C. Walsh, S. Karaman. "CDDT: Fast Approximate 2D Ray Casting for Accelerated Localization." Arxiv, 2017. http://arxiv.org/abs/1705.01167