Autonomous Robotics

Localization

Josiah Hanna
University of Wisconsin — Madison

Some figures taken from Probabilistic Robotics and Drew Bagnell's course notes.

Announcements

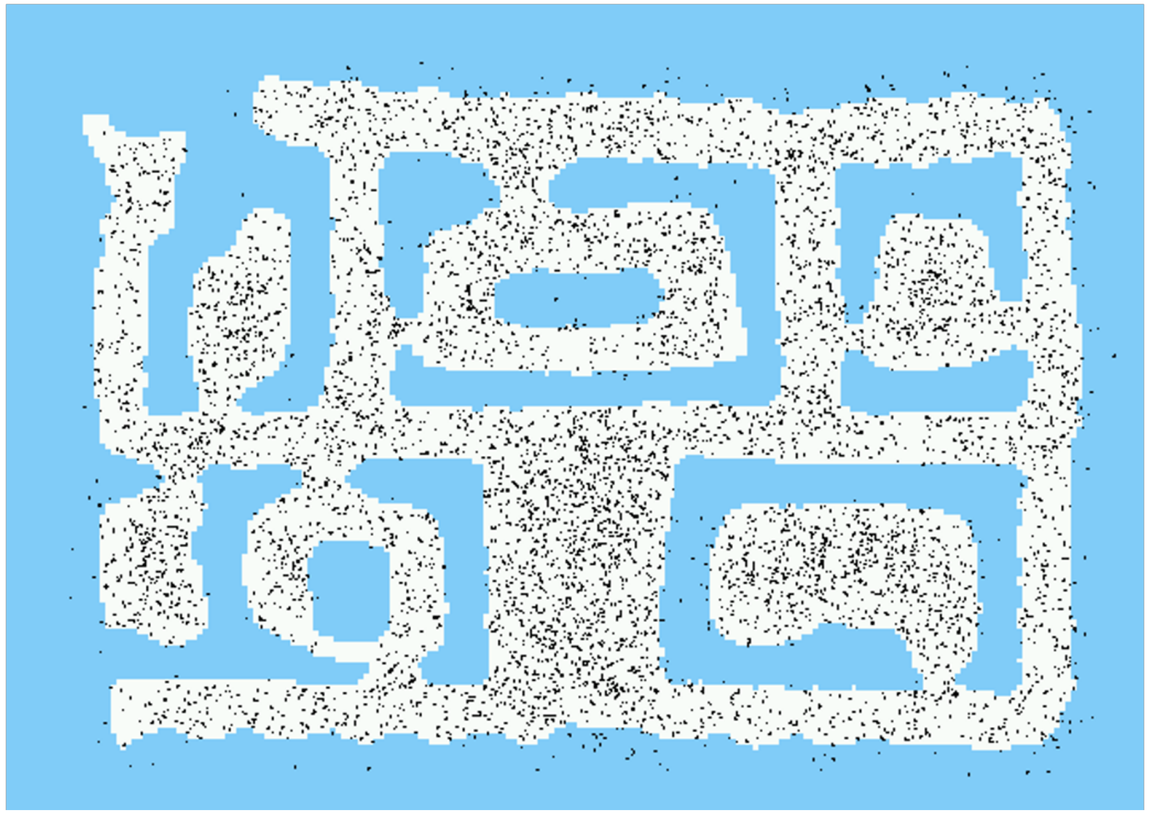
- Work on homework #2
- Reading assignment for next week (SLAM) has been posted.
- Asynchronous lecture (no in-person class) on Thursday (2/19).

Learning Outcomes

After today's lecture, you will:

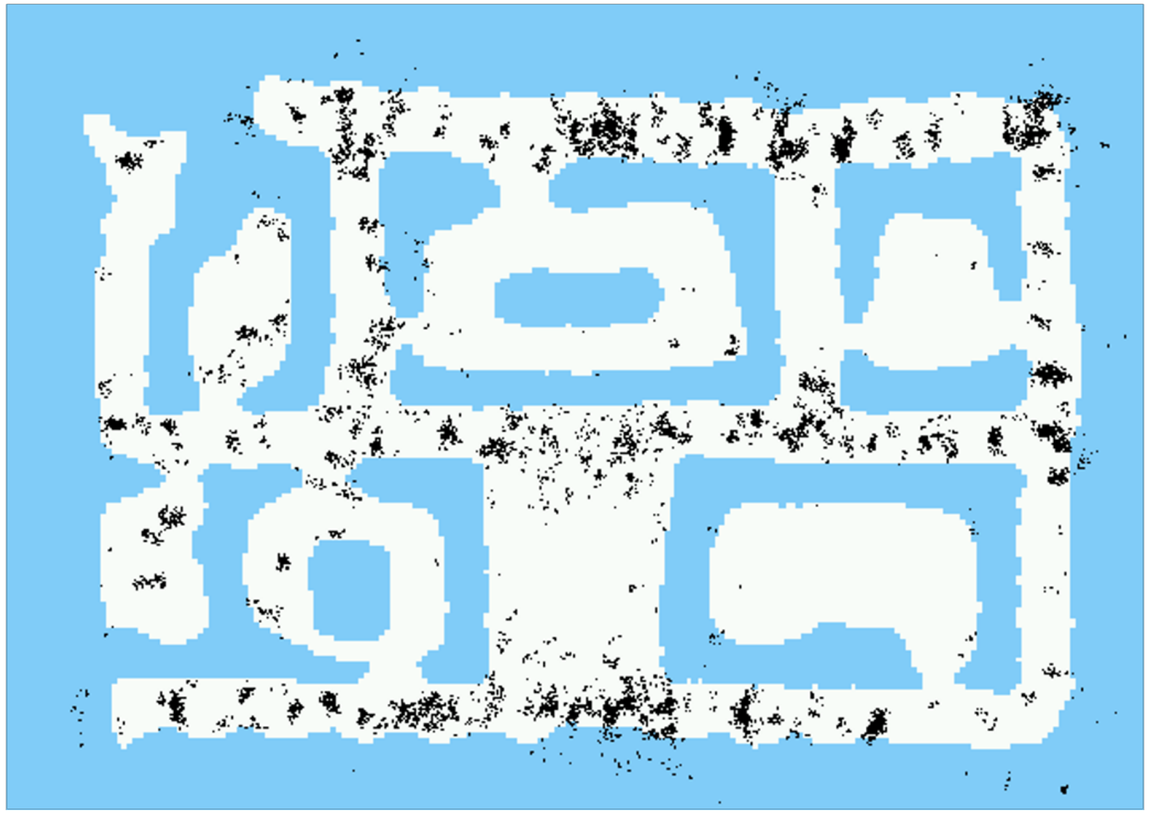
- Review the particle filter
- Be able to describe and formalize the robot localization problem.
- Be able to implement localization algorithms.

Particle Filter Applications



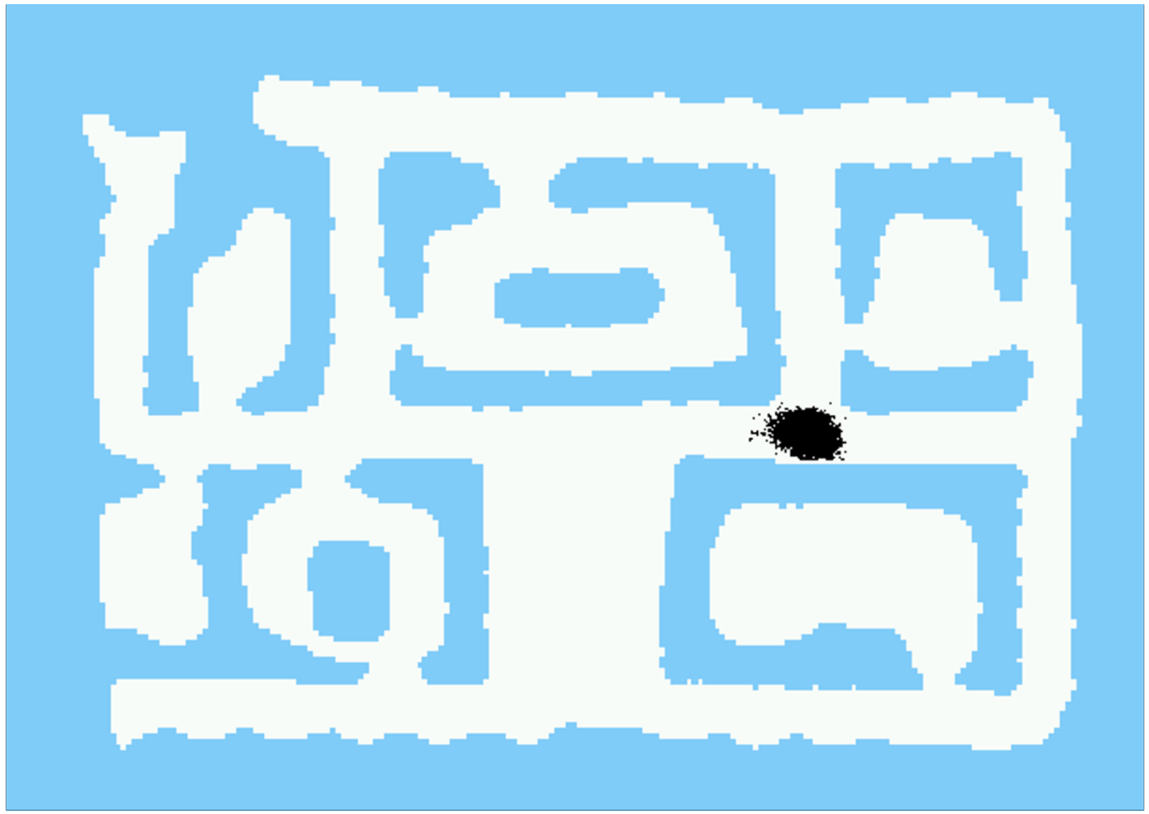
Initial particles

Particle Filter Applications



10 scans

Particle Filter Applications



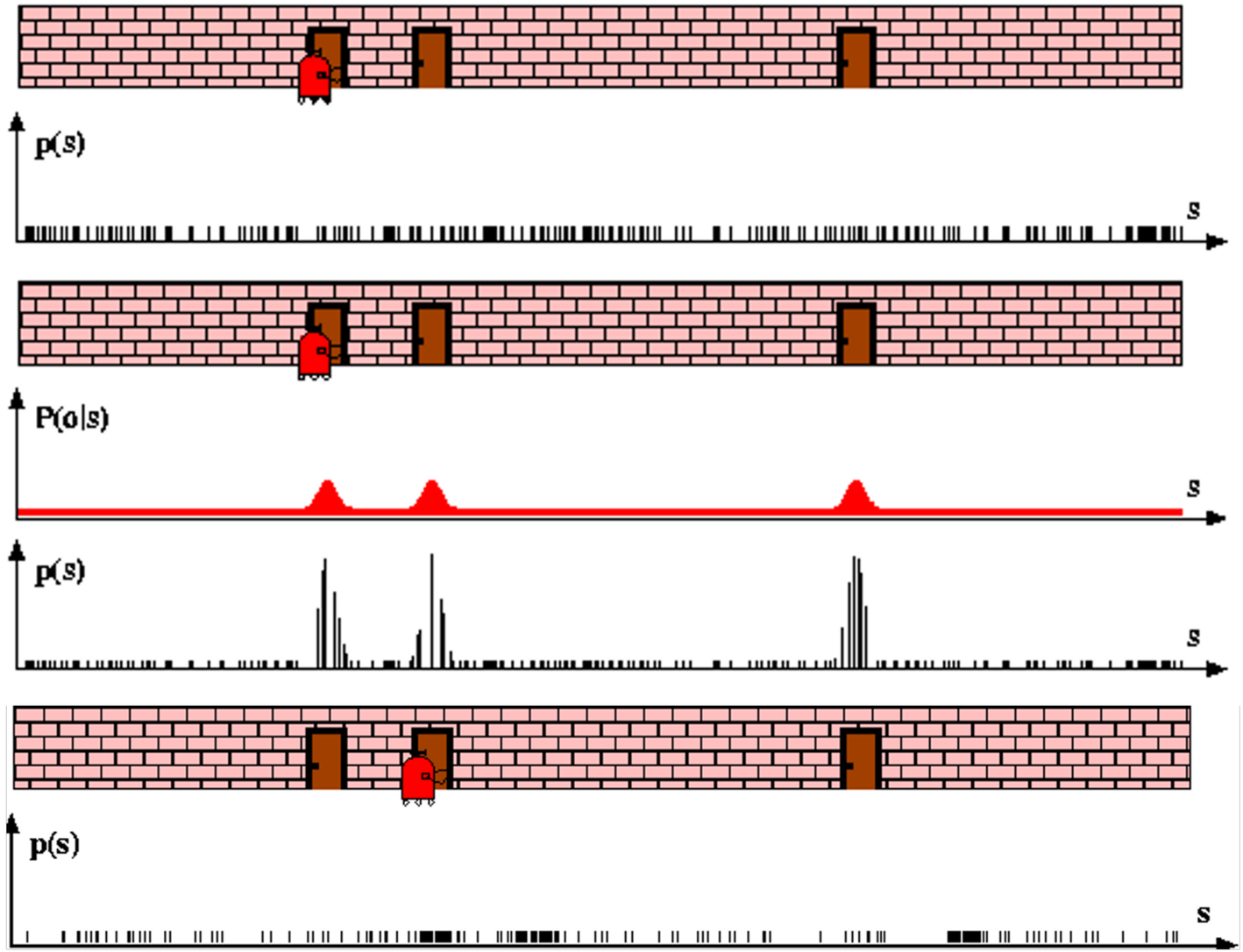
65 scans

Particle Filters

- Belief is represented by a set of particles, $\{(x_t^i, w_i)\}$.
- Robot takes action u_t and then observes z_t. Set $w_i \leftarrow \frac{1}{N}$.
- Update particles:
  - $x_{t+1}^i \sim p(\cdot \mid x_t^i, u_t)$
  - $w_i \leftarrow w_i * p(z_t \mid x_{t+1}^i)$
  - Normalize weights so that $\sum_{i=1}^{N} w_i = 1$.
  - Sample N new particles (with replacement) to form a new particle set.

$$\texttt{bel}(x_t) = \sum_{i=1}^{N} w_i \cdot \mathbf{1}\{x_t^i = x_t\}$$

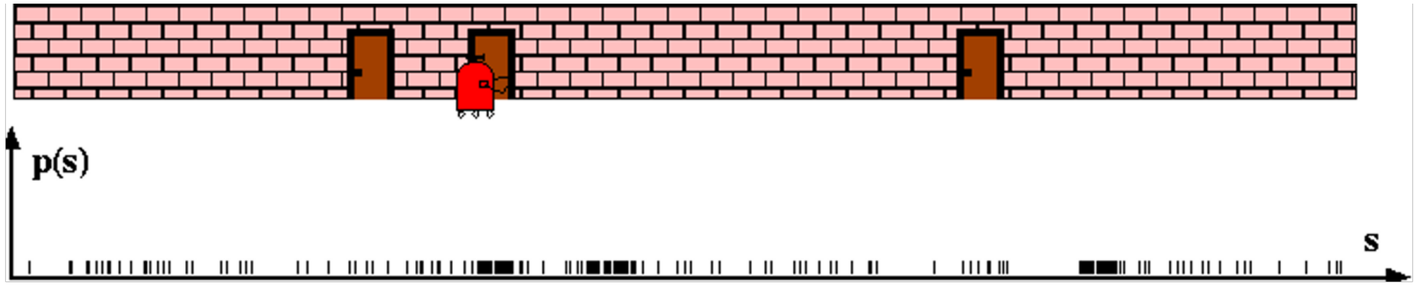
Particle Filter Illustration

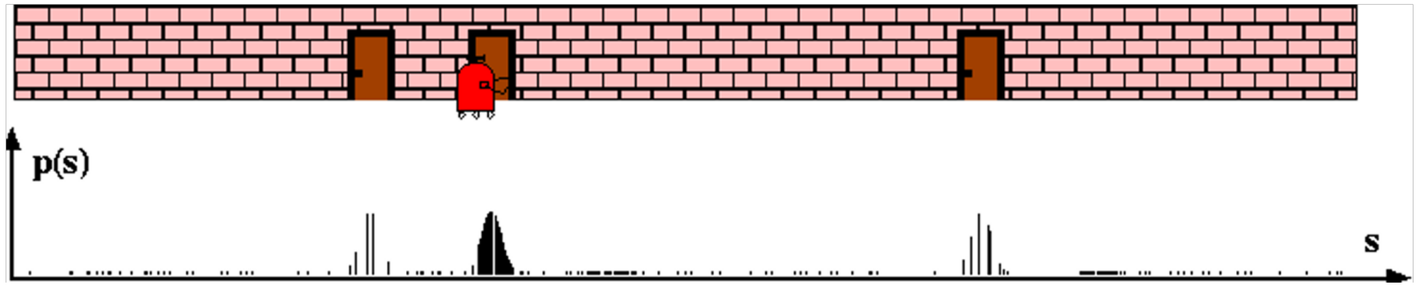


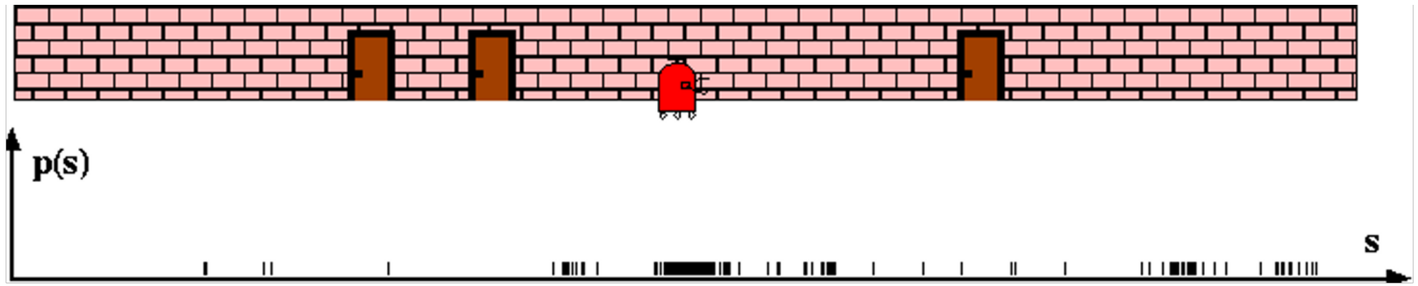
Explicit weight: the particle weight (height in this illustration)

Implicit weight: the density of similar particles

Particle Filter Illustration







Comparison to Kalman Filters

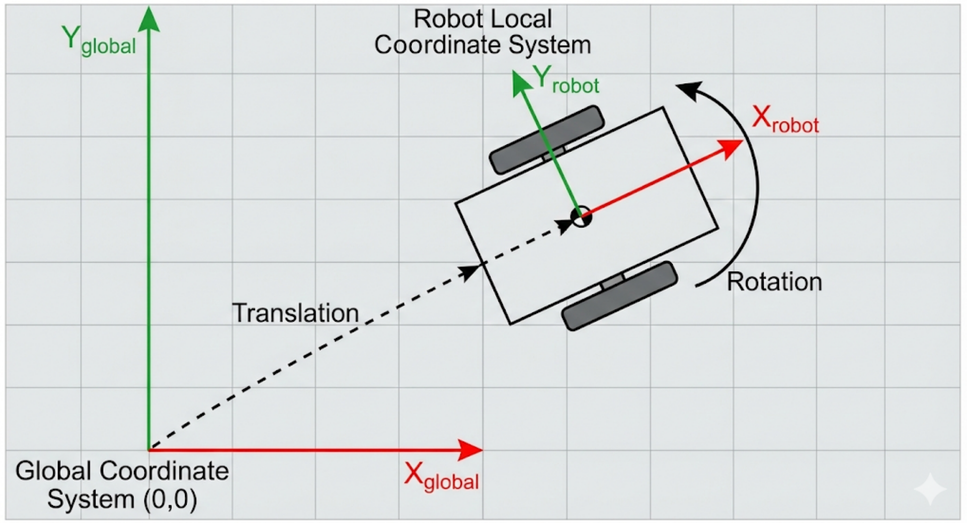
- Both filters can work in continuous state spaces.
- Particle filters > (Extended)KF:
  - Can approximate any belief distribution (compare to Kalman/EKF).
  - Approximate inference that scales with computation.
- (Extended)KF > Particle filters
  - Gaussian noise and dynamics are linear or can be linearized.
  - Computationally efficient.
  - Exact inference with fixed computation.

Localization Problem

- Estimate a robot's pose as it moves in an environment.
- Small changes to our formal model of the autonomous robot problem:
  - x_t will refer to the robot's pose in some global coordinate system.
  - m is the robot's map of the environment — more details in next lecture.
  - $p(x_t \mid x_{t-1}, u_t, m)$ is the robot's motion model, formalizing changes to the pose after an action is taken.
  - State and transition model are no longer attempting to capture all relevant factors.
- Markov localization: Bayes filters and their extensions applied to the localization problem.
- Pose is usually **not** sensed directly.

Coordinate Frames

- Localization is always done with respect to a given **global** coordinate frame.
  - E.g., longitude and latitude.
- Typically, the robot cannot directly measure location of itself and other objects in this coordinate frame.
  - Much easier to measure location within the robot's own coordinate frame.



Motion Models in Localization

- Common convention: define motion model in terms of odometry readings rather than true robot control.
  - Odometry: measurement of the robot's change in pose.
  - Example: for a differential drive robot, the control is a velocity set on each wheel while odometry is the robots forward motion and rotation (or forward and angular velocity).
- Odometry-based motion model is simple to calculate:

$$\theta_{t+1} = \theta_t + \omega \cdot \Delta t \qquad x_{t+1} = x_t + (v \cdot \Delta t)\cos(\theta)$$

$$y_{t+1} = y_t + (v \cdot \Delta t)\sin(\theta)$$

Notation:

v: forward velocity

ω: angular velocity

Motion Models in Localization

- Motion model depends upon map: $p(x_t \mid x_{t-1}, u_t, m)$.
  - Could be difficult to implement this motion model as would need to consider the interaction of the robot with different parts of the map.
  - Easier to implement a map-independent motion model, $p(x_t \mid x_{t-1}, u_t)$.

$$
\begin{aligned}
p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}, \boldsymbol{m}) &= \frac{p(\boldsymbol{m} \mid \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}{p(\boldsymbol{m} \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}, \\
&= \eta' p(\boldsymbol{m} \mid \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \\
&\approx \eta' p(\boldsymbol{m} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t), \\
&= \eta \frac{p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_t \mid \boldsymbol{m})}{p(\boldsymbol{x}_t)},
\end{aligned}
$$

Map-dependent motion model is the composition of map-indepdent model and a consistency check on the pose x_t

Multiple Measurement Models

- Robots often have multiple sensors. Instead of a single observation per time-step, z_t, we have multiple observations, z_t^i.
- Each sensor type will have its own measurement (observation) model.
  - $g_i(z_t^i \mid x_t, m)$.
- To integrate into the framework of Bayes filters, we will assume conditional independence given the current robot pose and map.

$$g(z_t \mid x_t, m) = \prod_i g_i(z_t^i \mid x_t, m)$$

- Conceptually, just apply the correction step for each observation sequentially.

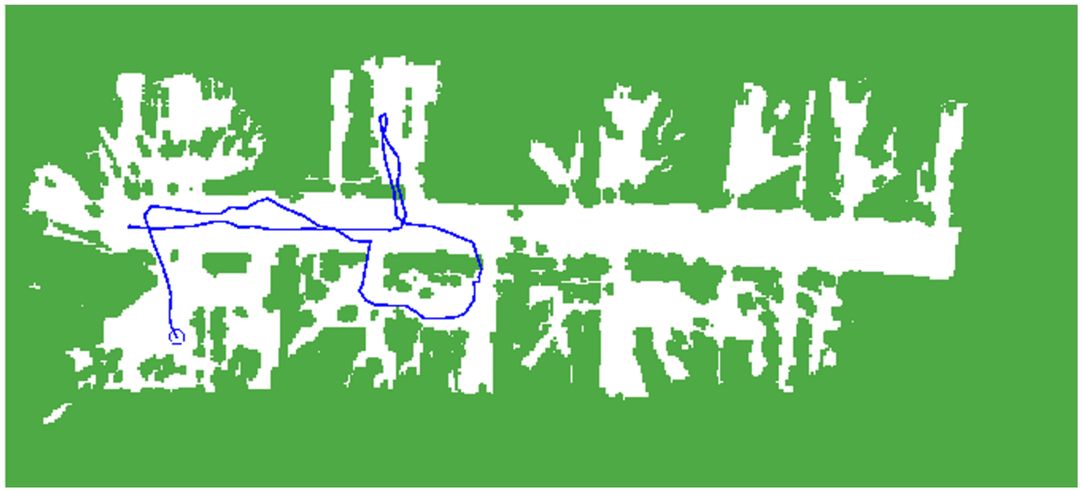
Localization Taxonomy

- Position-tracking: estimate the robot's current pose given observations, controls, and knowledge of the initial state.
  - $\texttt{bel}(x_t) := p(x_t \mid x_0, z_{1:t}, u_{1:t}, m)$
- Global localization: estimate the robot's current pose given observations and controls.
  - $\texttt{bel}(x_t) := p(x_t \mid z_{1:t}, u_{1:t}, m)$
- Kidnapped robot problem: the robot is teleported to some other location during operation and must recognize this and then relocalize.

Position-Tracking

Actual Trajectory

Trajectory from Odometry



Trajectory from position-tracking

EKF-Localization

- EKF-localization is most effective for position-tracking. Why?

Algorithm 2: Extended Kalman Filter Localization Algorithm

Data: $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, m$

Result: μ_t, Σ_t

$\bar{\mu}_t = g(u_t, \mu_{t-1})$ <— update mean with odometry

$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

foreach z_t^i **do**

Landmark given

$j = c_t^i$

$S_t^i = H_t^j \bar{\Sigma}_t [H_t^j]^T + Q_t$

$K_t^i = \bar{\Sigma}_t [H_t^j]^T [S_t^i]^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - h(\bar{\mu}_t, j, m))$

$\bar{\Sigma}_t = (I - K_t^i H_t^j) \bar{\Sigma}_t$

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

Algorithm 3: EKF Localization Algorithm, Unknown Correspondences

Data: $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$

Result: μ_t, Σ_t

$\bar{\mu}_t = g(u_t, \mu_{t-1})$ <— update mean with odometry

$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

foreach z_t^i **do**

foreach *landmark k in the map* **do**

Select most likely landmark

$\hat{z}_t^k = h(\bar{\mu}_t, k, m)$

$S_t^k = H_t^k \bar{\Sigma}_t [H_t^k]^T + Q_t$

$j = \arg\min_k \; (z_t^i - \hat{z}_t^k)^\top [S_t^k]^{-1} (z_t^i - \hat{z}_t^k)$

$K_t^i = \bar{\Sigma}_t [H_t^j]^T [S_t^j]^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^j)$

$\bar{\Sigma}_t = (I - K_t^i H_t^j) \bar{\Sigma}_t$

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

Notation: i is the index of the observed landmark; j indexes the actual landmark.

Static vs Dynamic Localization

Static vs. Dynamic

- Static: only the robot's pose changes in the environment.

- Dynamic: other factors change in the environment.

How to handle dynamic factors?

- Add new state variables to track

- Treat as exogenous noise (build into model stochasticity)

https://www.epfl.ch/labs/vita/research/planning/crowd-robot-interaction/

Passive vs Active Localization

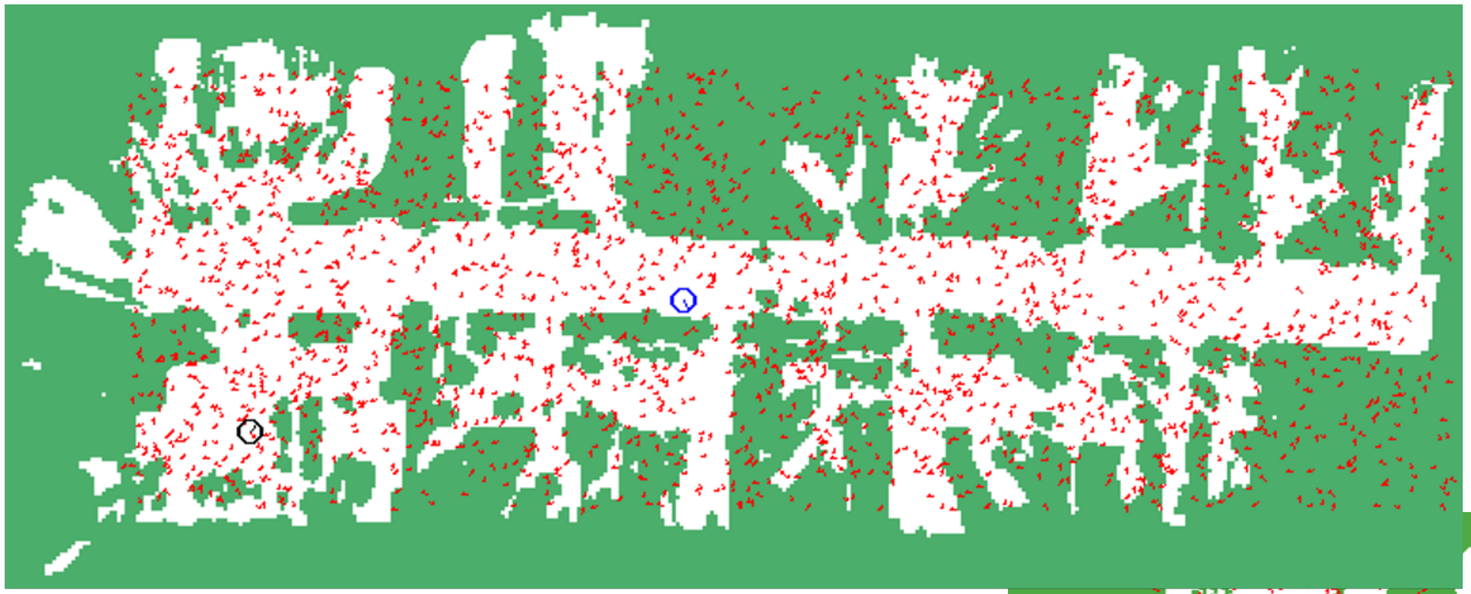
- Active: the robot takes actions that help localization. How?
  - Information-gathering actions
  - Remaining in easy to localize areas, e.g., wall-following.
- Passive: the robot's actions are guided by some other goal.

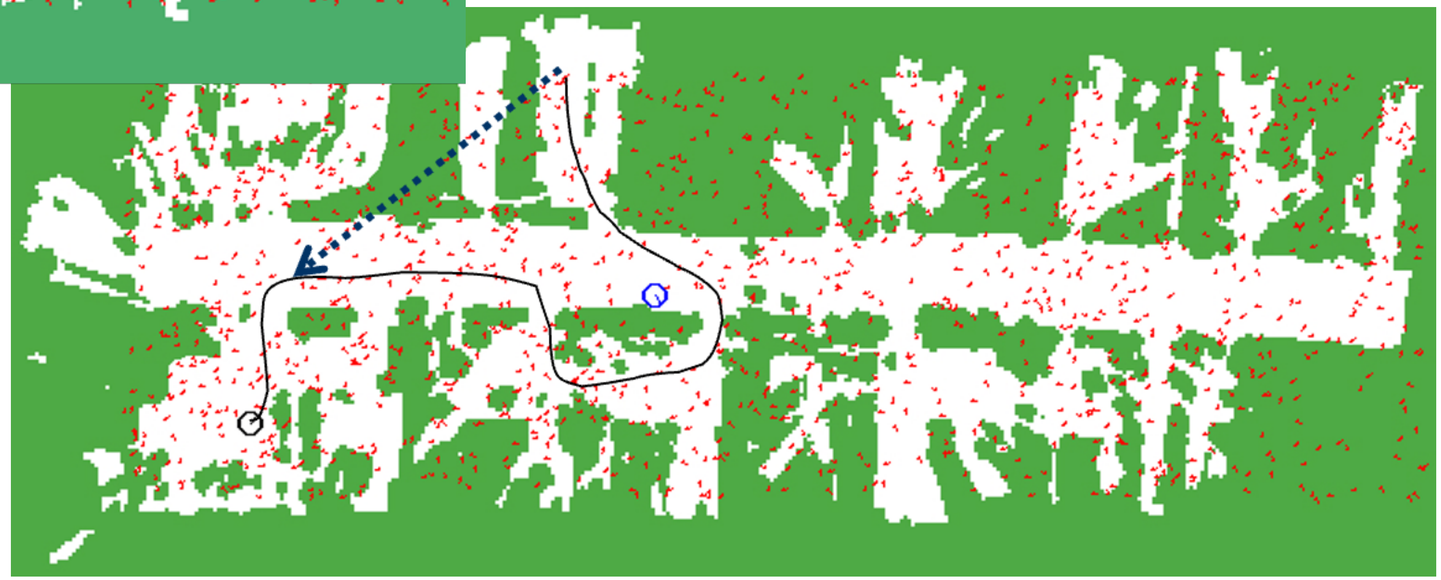
https://www.epfl.ch/labs/vita/research/planning/crowd-robot-interaction/

Monte Carlo Localization

- Particle filter applied for Markov localization.
- How to initialize particles?
- How to handle failures?
  - Add random particles. How many? How to choose them?

Monte Carlo Localization





Summary

- Reviewed the particle filter.
- Introduced the localization problem.
- Discussed considerations of the particle filter for localization problems.

Action Items

- Work on programming assignment #2.
- Read on SLAM for next week; send a reading response by 12 pm on Monday.