Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ECE 4160/5160
MAE 4910/5910

Fast Robots

Monte Carlo Localization
Brief intro to SLAM

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Lab 10

Lab 10 – Localization in Simulation

- https://cei-lab.github.io/FastRobots-2023/Lab10.html
- Pass/Fail
- To dos
  - Read the full lab and the notebook (before you show up to lab!)
  - Perform Grid localization for the sample trajectory
  - Video demo
  - Linda Li's example from 2022: https://lyl24.github.io/lyl24-ece4960/lab11
  - Discuss…
    - Control
    - Motion model and the prediction step
    - Sensor model and the update step
    - Choosing parameters, what effect do the parameters have
    - Ways to mitigate the computational load
    - Evaluate how well the Bayes filter works
    - How well will it work for your robot

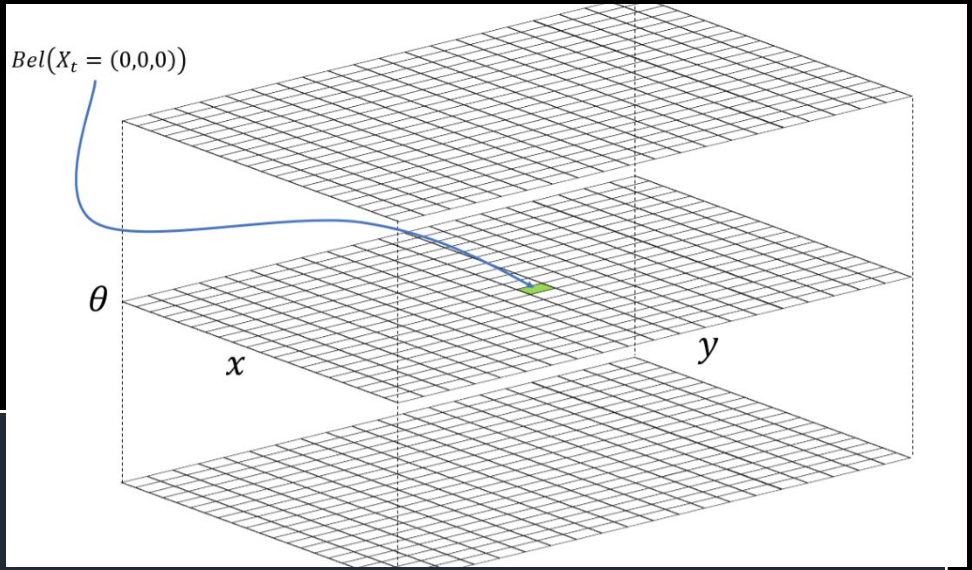
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots

Localization

Grid-Based Localization

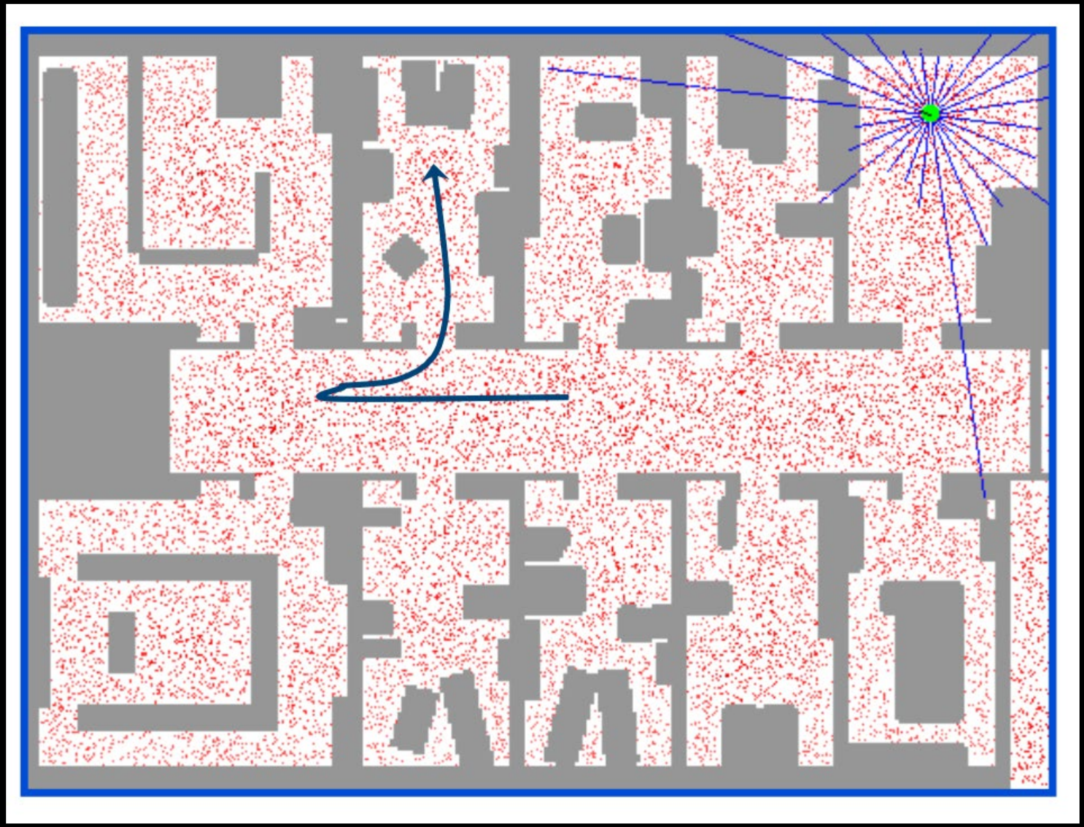
- Simple
- …but is computationally expensive for large workspaces



1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t | u_t, x_{t-1})\ bel(x_{t-1})$
4. $bel(x_t) = \eta\, p(z_t | x_t)\ \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

Monte Carlo Localization

- Non-parametric approach based on Particle Filters
- Model the distribution by samples
  - Prediction step
    - Draw from the samples
    - (Move forward based on motion model)
  - Update step
    - Weigh samples by their importance
    - (Sensor model)
  - Resample based on their weight
- The more samples we use, the better the estimate!



Monte Carlo Localization

- Non-parametric approach based on Particle Filters
- Model the distribution by samples

1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t\ , z_t)$:
2. for all x_t do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$
4. $bel(x_t)\ = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$
5. endfor
6. return $bel(x_t)$

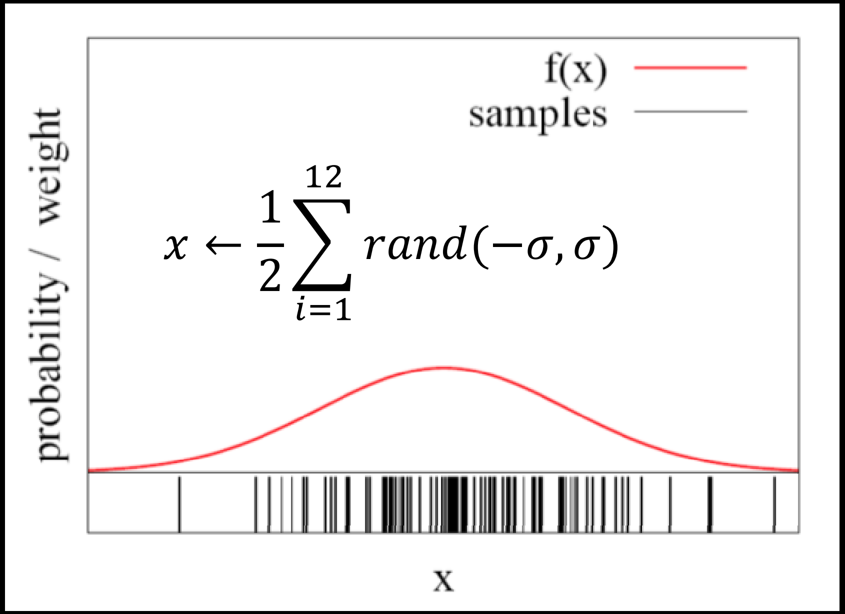
Prior samples

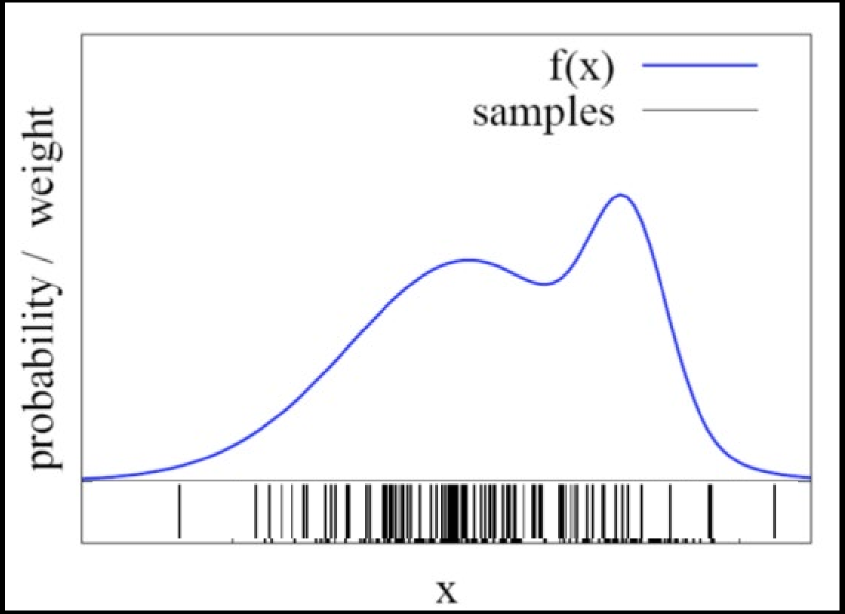
Draw x_t^i from $p(x_t|u_t, x_{t-1}^i)$

Importance factor for x_t^i: $w_t^i\ \alpha\ p(z_t|x_t)$

Monte Carlo Localization

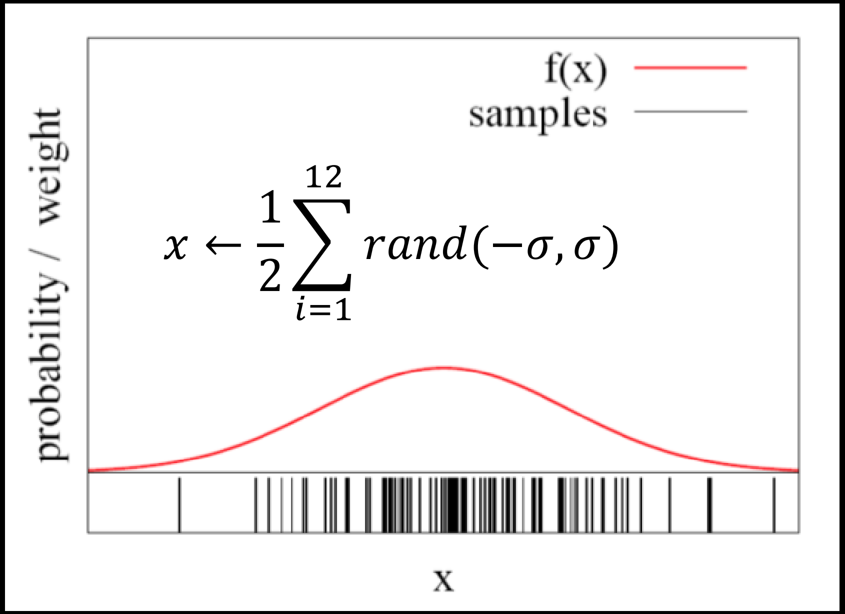
- *How do you obtain samples from an arbitrary distribution?*
  - Closed form solution for a uniform distribution
  - Closed form solution for Gaussian distribution

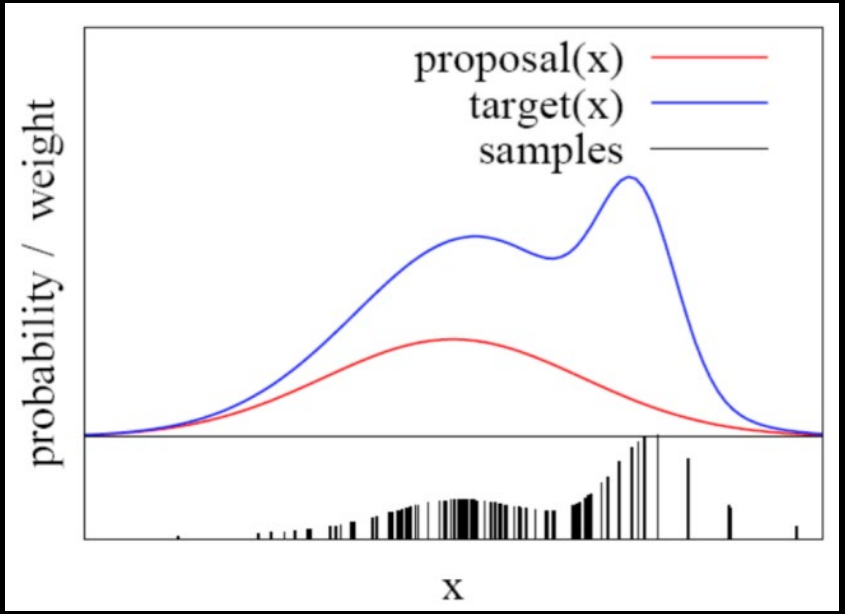




Monte Carlo Localization

- *How do you obtain samples from an arbitrary distribution?*
  - Closed form solution for a uniform distribution
  - Closed form solution for Gaussian distribution
- Use a *proposal distribution* to generate samples from the *target distribution*
- Account for differences using a weight
  - *w = target/proposal*





Monte Carlo Localization

https://www.cs.uml.edu/~holly/teaching/4510and5490/fall2018/Lecture-Particle-Filters.pdf

- Each particle, *j*, is a pose hypothesis
- *Proposal distribution* from the motion model

$$x_t^{[j]} \sim p(x_t | x_{t-1}, u_t)$$

- *Correction* via the observation model

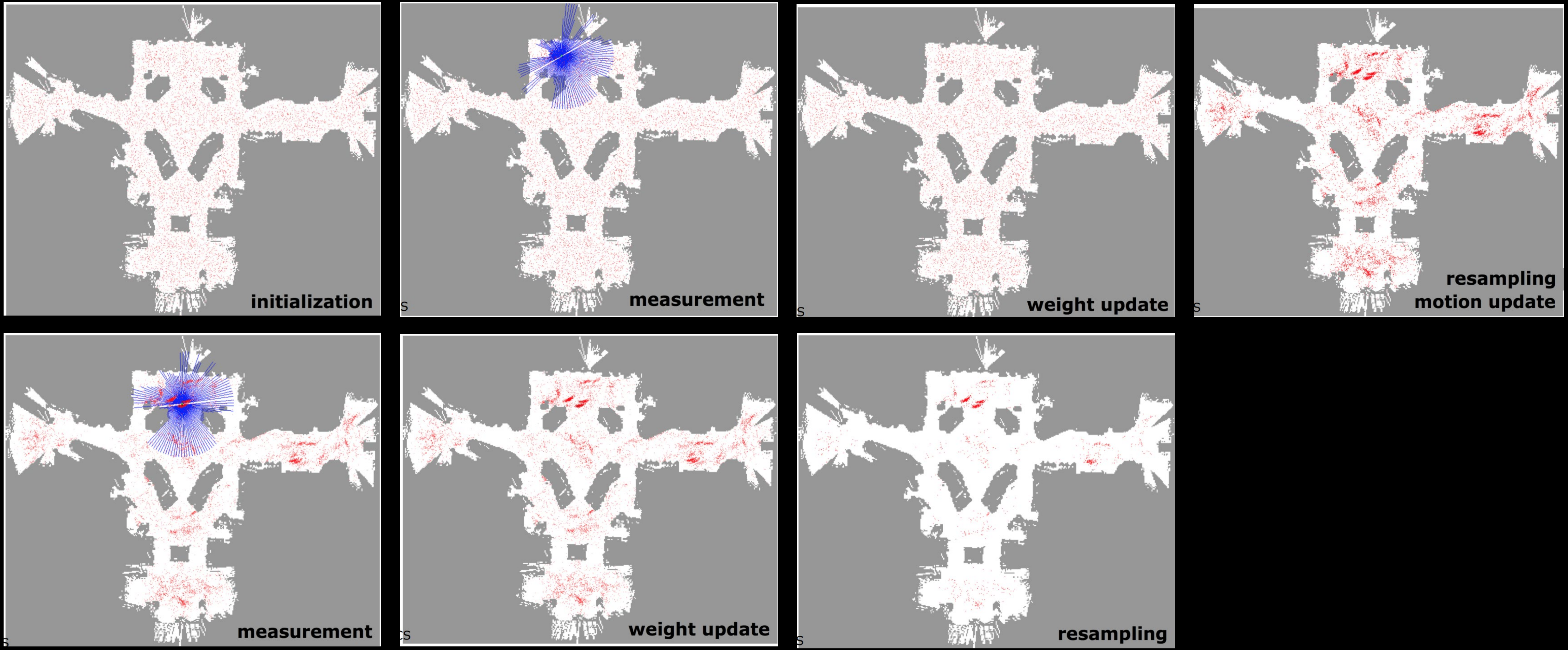
$$w_t^{[j]} = \frac{target(x_t^{[j]})}{proposal\left(x_t^{[j]}\right)} = p(z_t | x_t)$$

- *Resample*
  - Draw sample *I* with probability $w_t^{[j]}$ and repeat *J* times

Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $j = 1$ to J do
    sample $x_t^{[j]} \sim p(x_t \mid u_t, x_{t-1}^{[j]})$
    $w_t^{[j]} = p(z_t \mid x_t^{[j]})$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[j]}, w_t^{[j]} \rangle$
endfor
for $j = 1$ to J do
    draw $i \in 1, \ldots, J$ with probability $\propto w_t^{[i]}$
    add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$

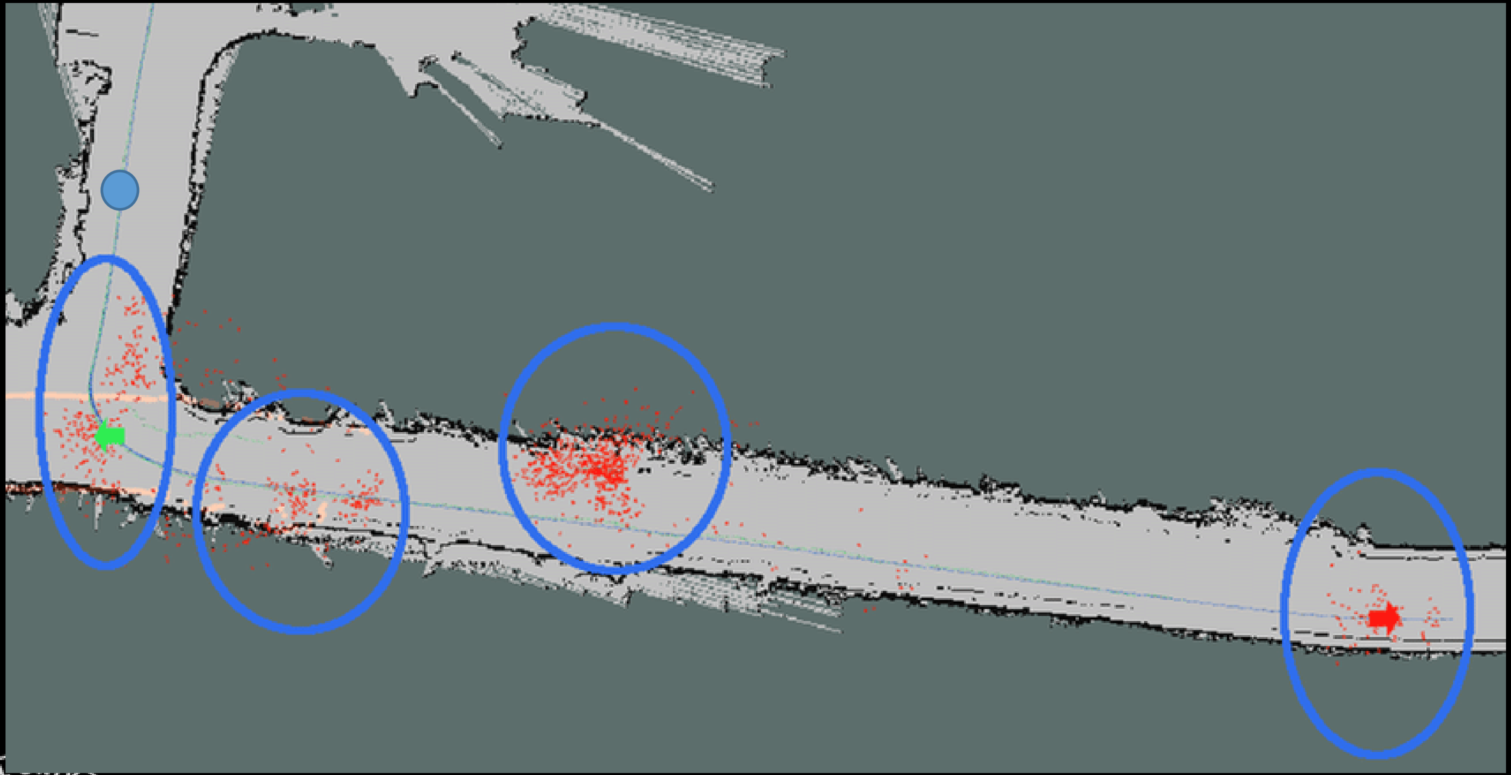
Monte Carlo Localization



https://www.cs.uml.edu/~holly/teaching/4510and5490/fall2018/Lecture-Particle-Filters.pdf

Monte Carlo Localization

- How would you deal with a kidnapped robot situation?
  - Randomly insert samples proportional to the average likelihood of the particles



- Pros
  - Works well for high-uncertainty scenarios
  - Much more efficient that the grid cells
- Cons
  - Scales poorly with higher dimensional workspaces

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ECE 4160/5160
MAE 4910/5910

Fast Robots

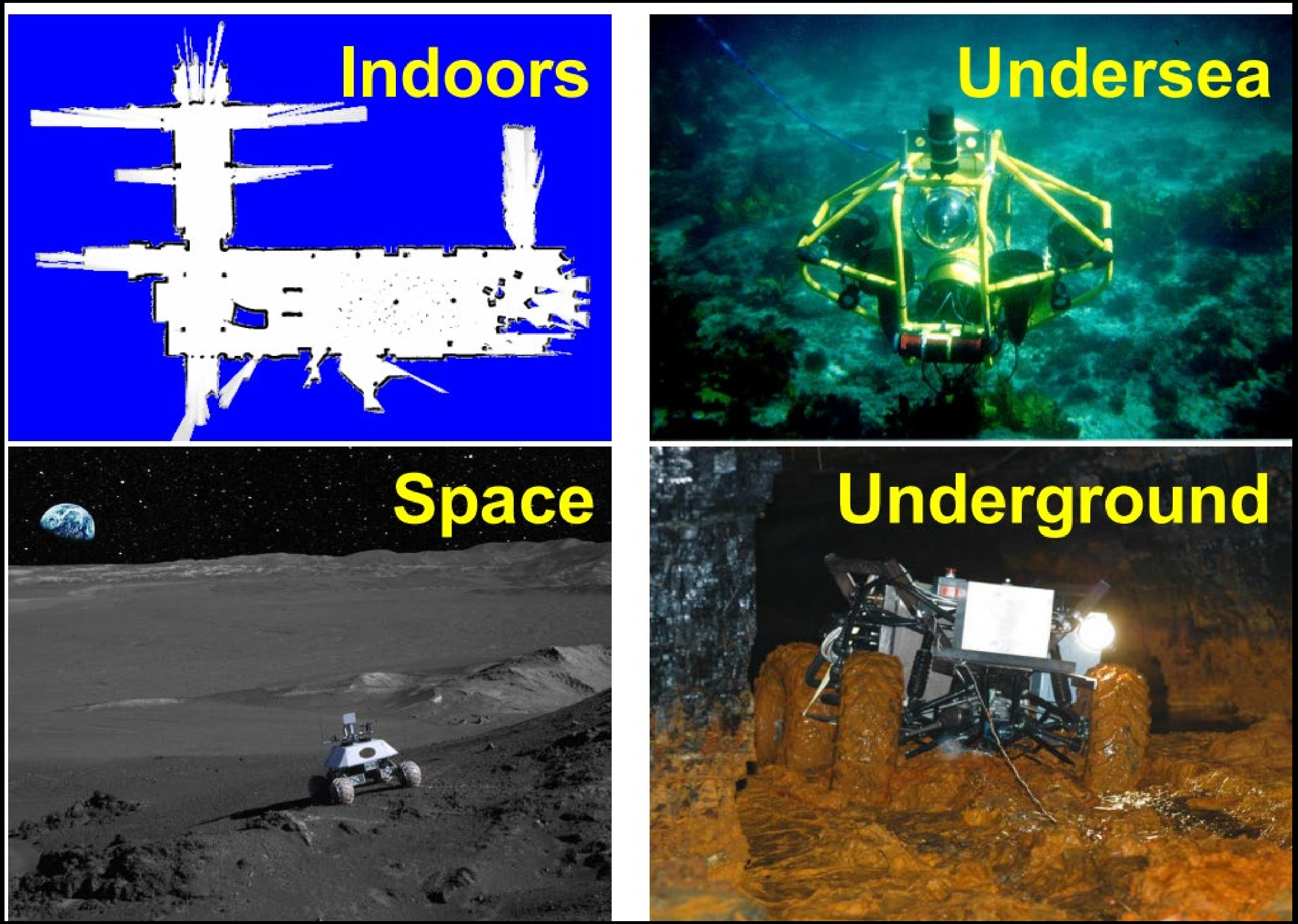
Brief intro to SLAM

Related Terms

- State estimation
- Localization → Inferring a location given a map
- Mapping → Inferring a map given a location
- SLAM → Learning a map and locating the robot simultaneously
- Navigation
- Motion planning

Related Terms

- State estimation
- Localization
- Mapping
- SLAM
- Navigation
- Motion planning



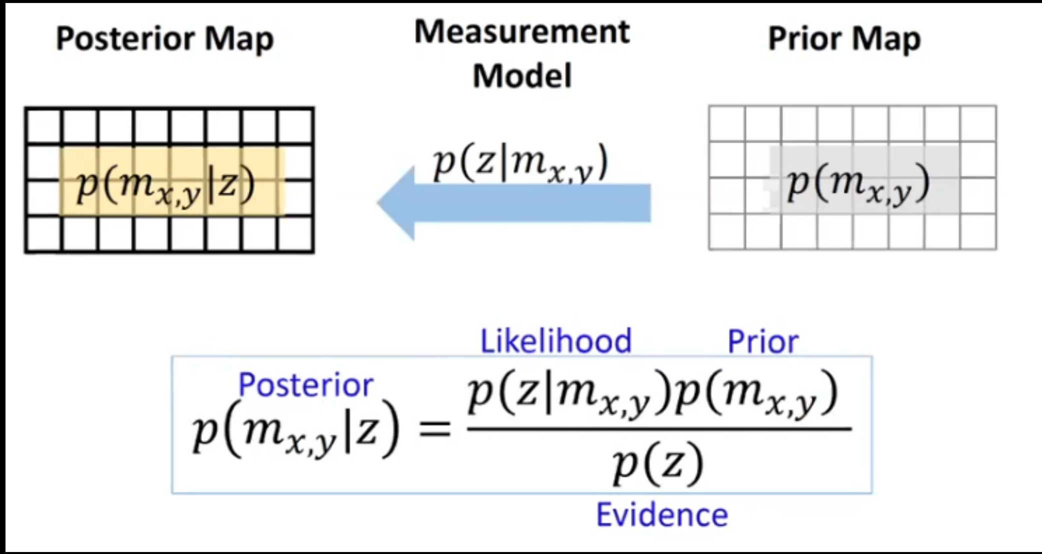
Given all we have learned…

- Transformation matrices
- Sensor and actuators (and probabilistic models)
- Controllers (PID, LQR)
- Observers (KF) → Include the map into the state
- Mapping
- Localization
  - Bayes Filter and grid-localization → Add grid-occupancy
  - Particle Filter → Let particles represent both pose and map
- Graph Search and Planning

…how would you implement SLAM?
(where could your estimate of the map fit in?)

Given all we have learned…

- Markov localization in a grid
  - Localization: Estimate your cell pose within the map
  - Mapping: Estimate if cells are occupied or not
    - Every grid cell is a random variable
  - SLAM: Estimate pose *and* if cells are occupied or not
    - 100x100 grid cells (pretty small map)
    - Localization: (x,y,theta) = 100x100x100 states
    - Map: (x,y) = 10,000 states
    - SLAM: 100x100x100 x10,000 states
- Same issue for particle filters…
    - Balance parametric and non-parametric approaches



Why is SLAM hard?

- Robot pose/path and map are both unknown
  - Not independent…
  - Map and pose estimates are correlated

Why is SLAM hard?

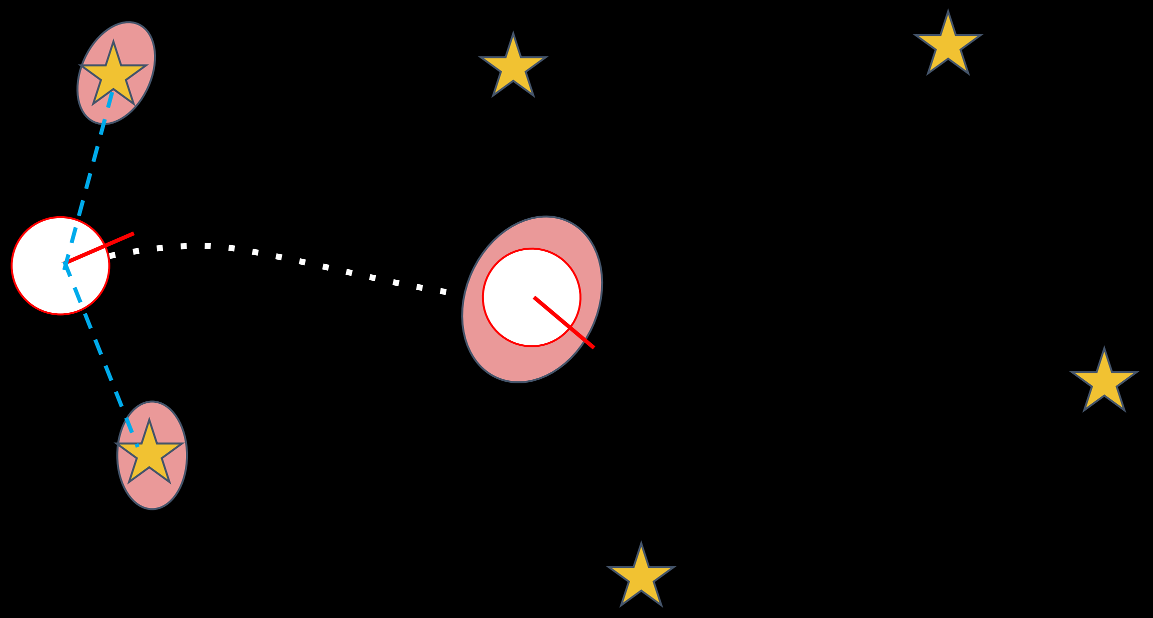
- Robot path and map are both unknown
- Map and pose estimates are correlated

Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated

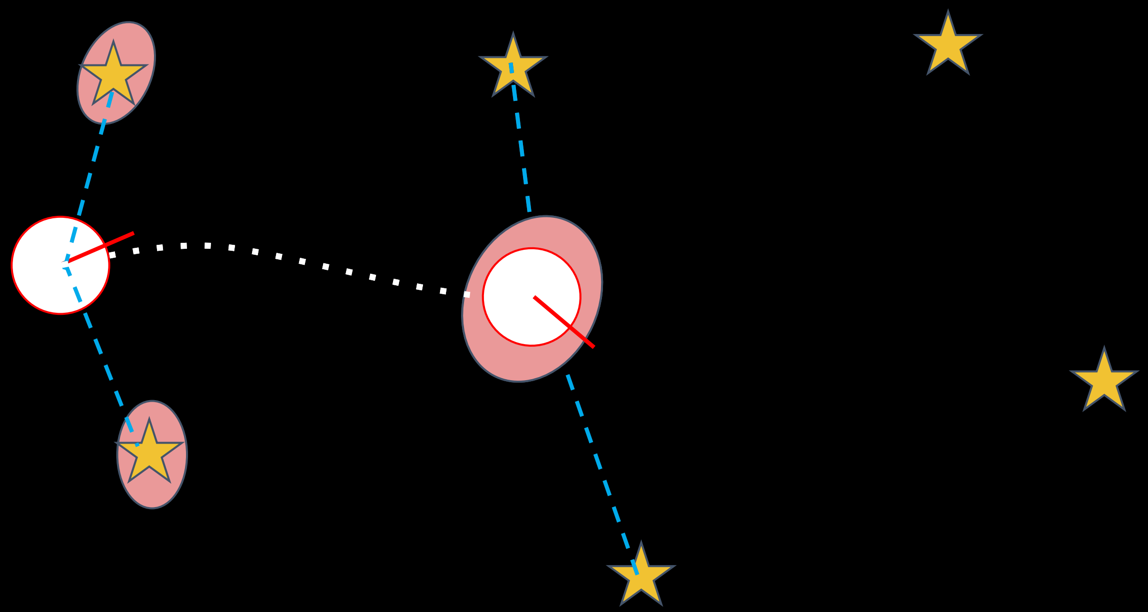
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated



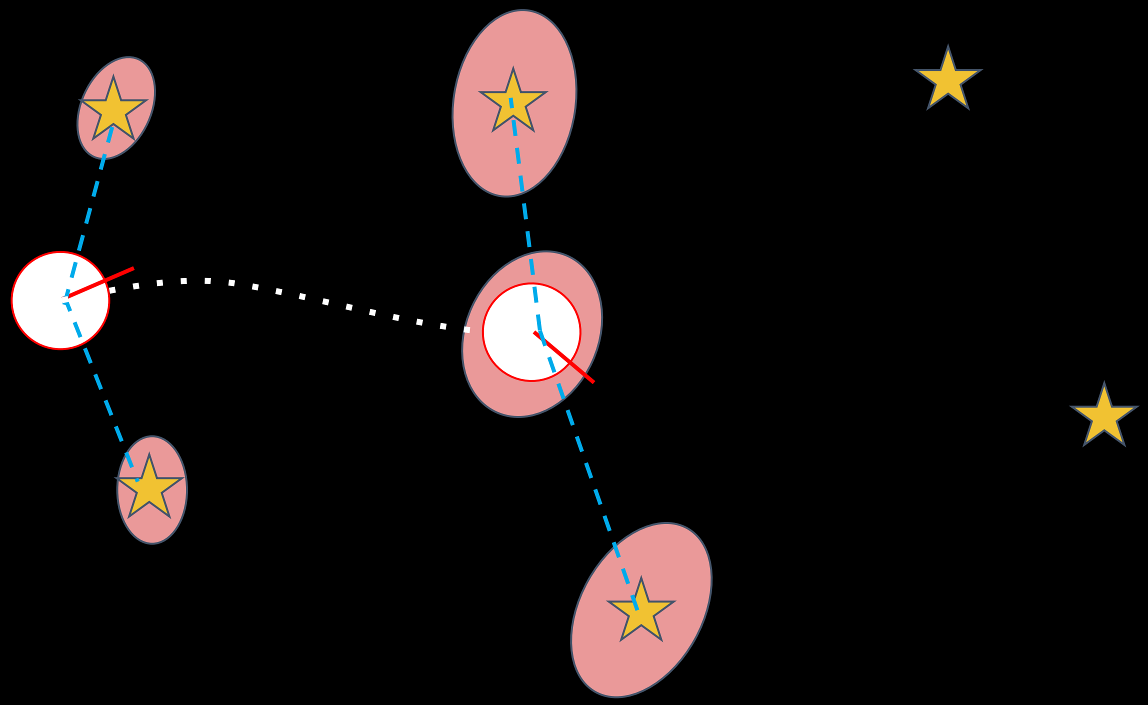
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated



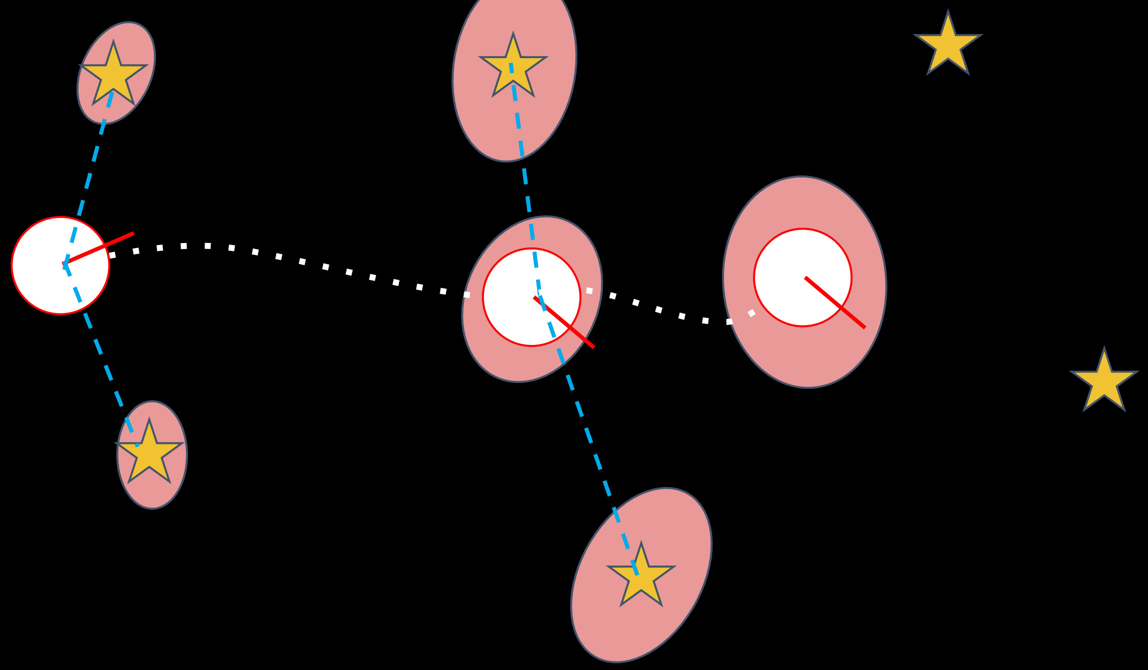
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated



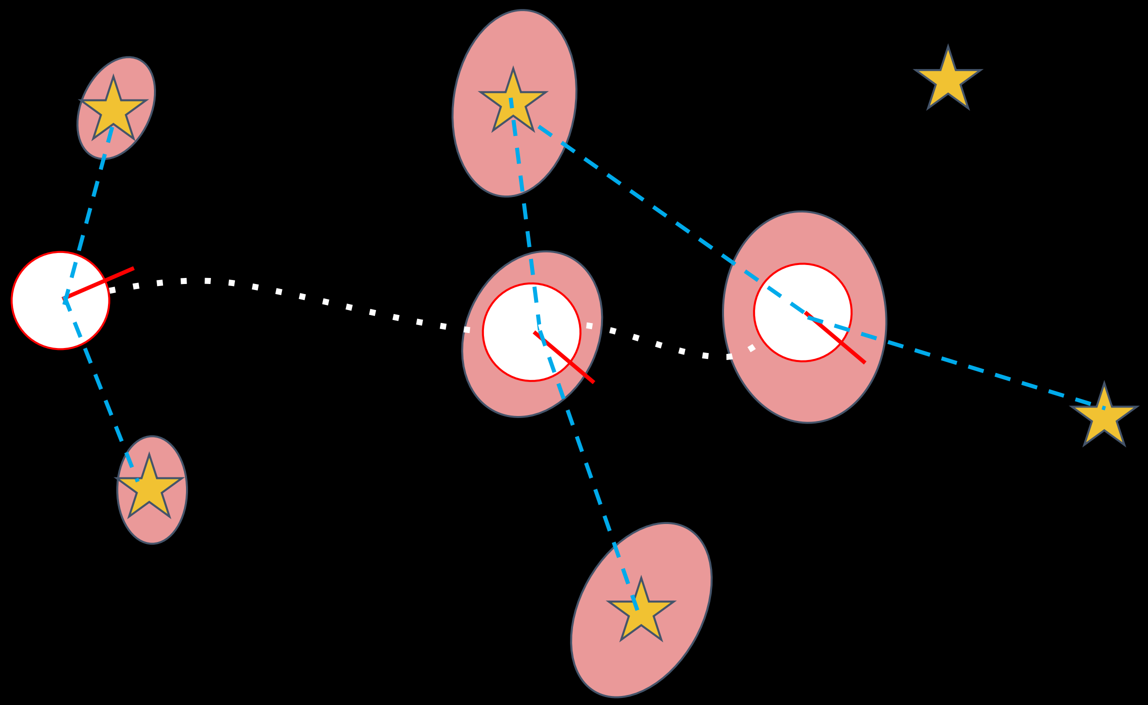
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated



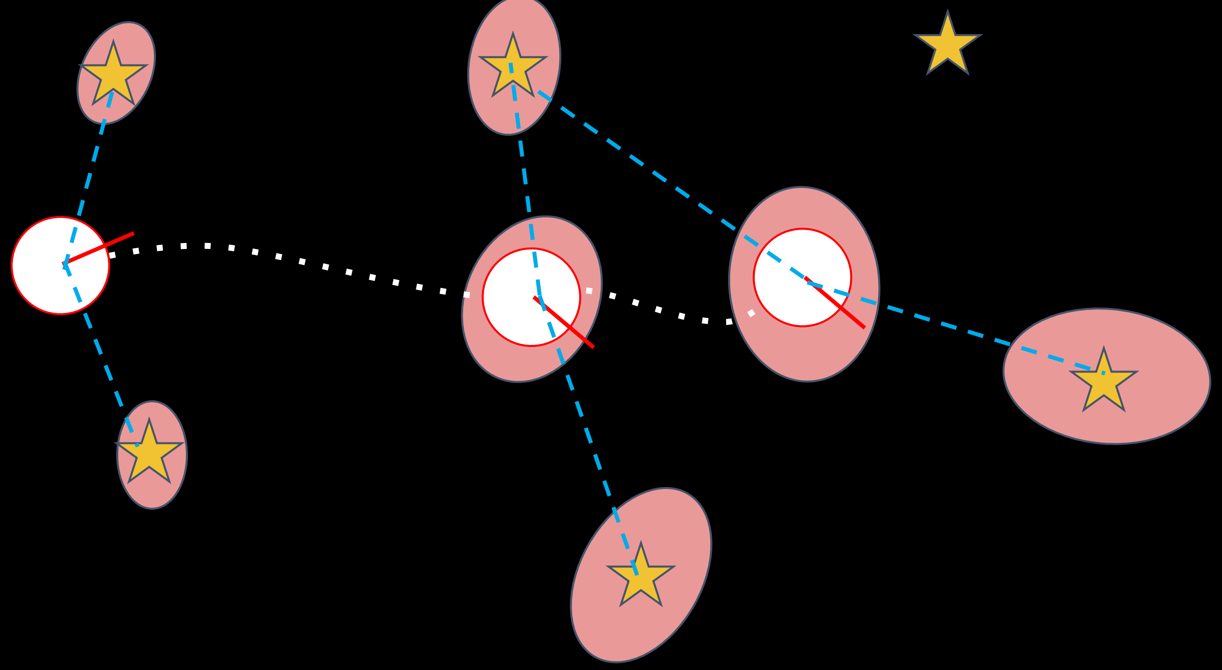
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated



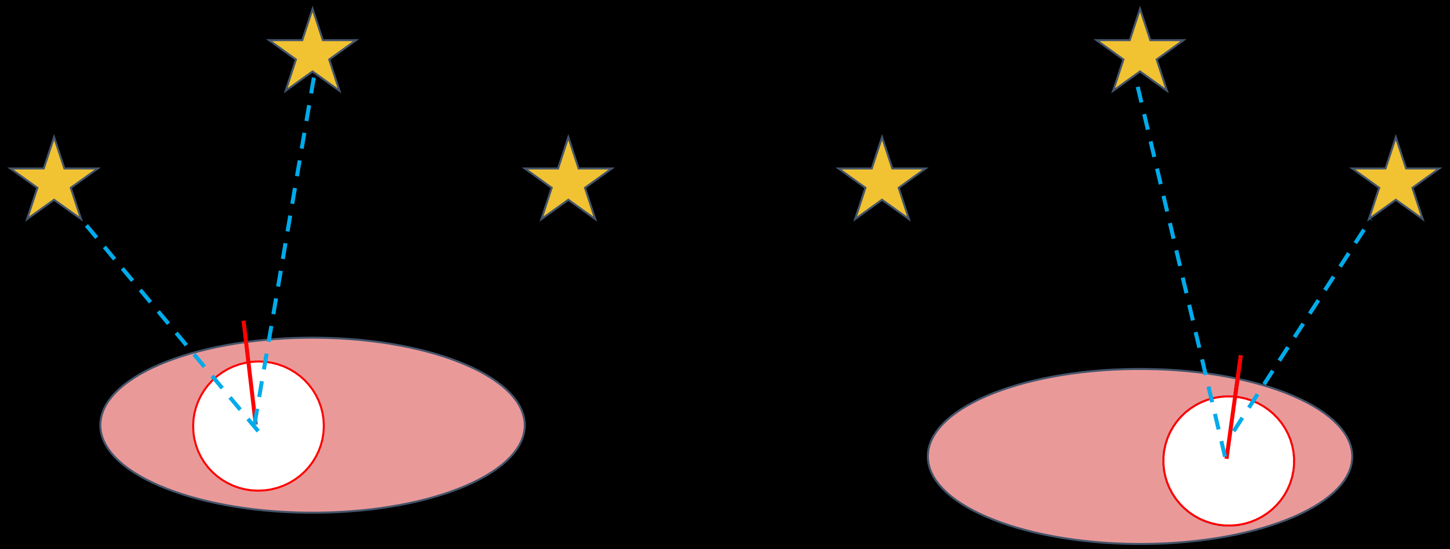
Why is SLAM hard?

- Robot path and map are both unknown
- Map and pose estimates are correlated
- Good data association is key



Why is SLAM hard?

- The mapping between observations and the map is unknown
- Picking the wrong data association can cause map divergence

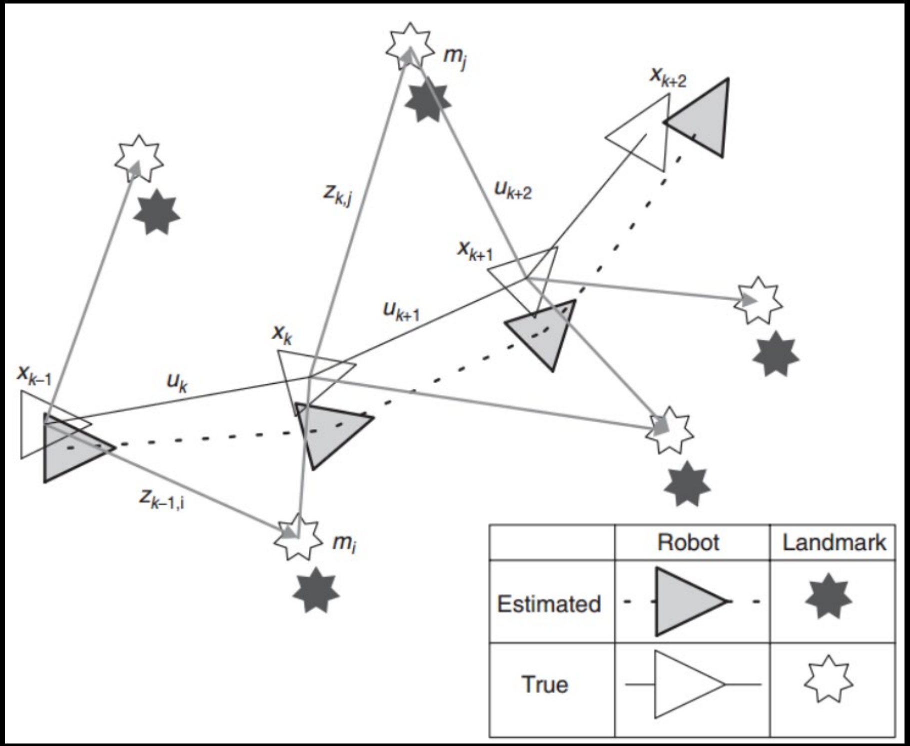


Related Terms

- State estimation
- Localization
- Mapping
- SLAM
- Navigation
- Motion planning

- Given
  - Control inputs
    - $U_{o:k} = \{u_1, u_2, \ldots u_k\}$
  - Relative observations
    - $Z = \{z_1, z_2, \ldots z_n\}$
- Compute
  - Map of the environment
    - $m = \{m_1, m_2, \ldots m_n\}$
  - Robot path (seq. of poses)
    - $X_{o:k} = \{x_0, x_1, \ldots x_k\}$

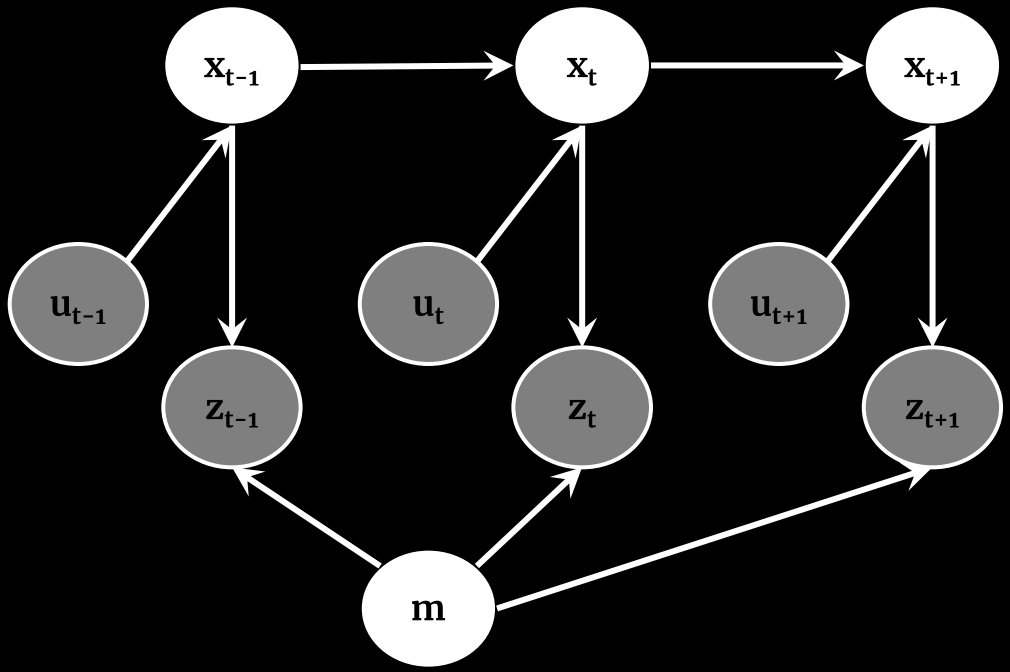
- Error in pose
- Error in observation
- Error in mapping
- Errors accumulate



(Landmarks are considered motionless)

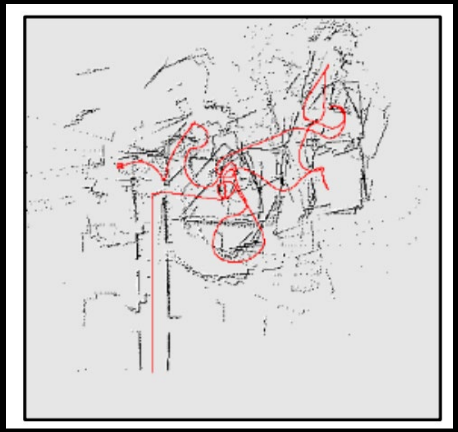
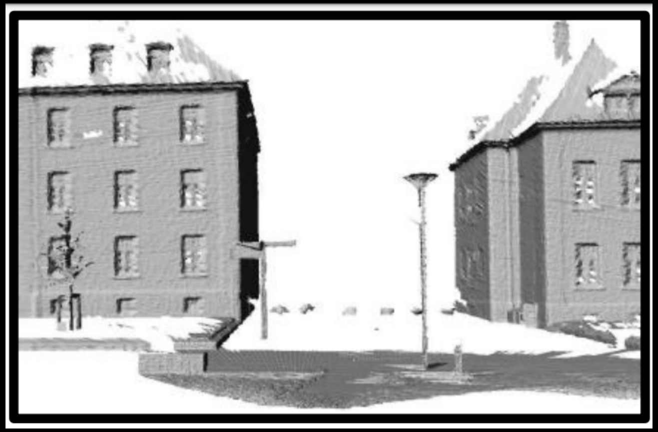
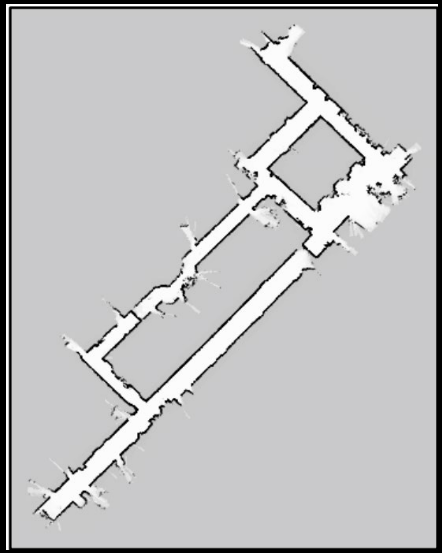
Simultaneous Localization and Mapping (graphical model)

- Nodes are random variables
- Directed edges are variable dependencies
- Gray nodes
  - Observed or directly measured variables
- White nodes
  - Inferred latent variables



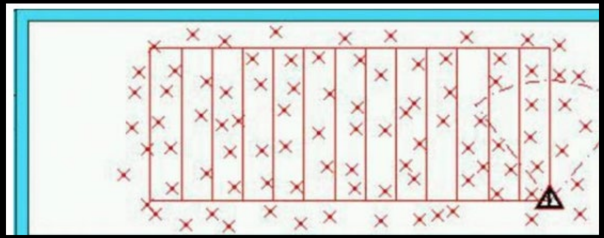
SLAM Representations

- Grid maps or scans

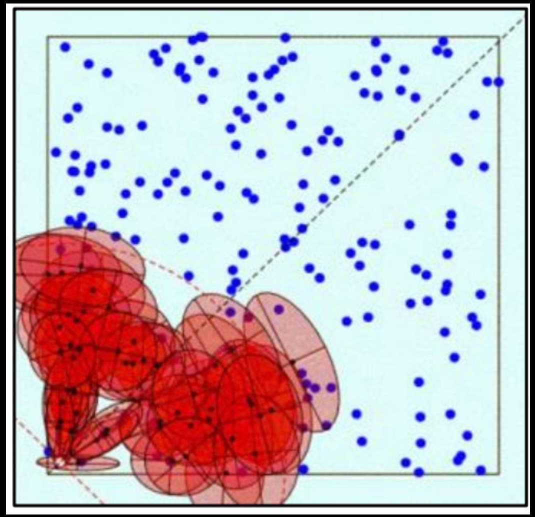
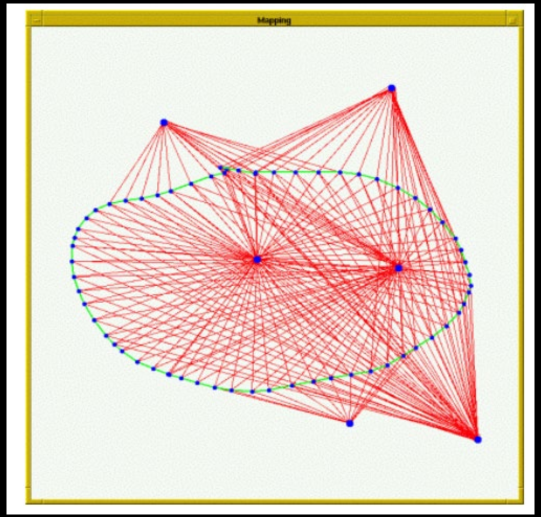




[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;...]

- Landmark-based



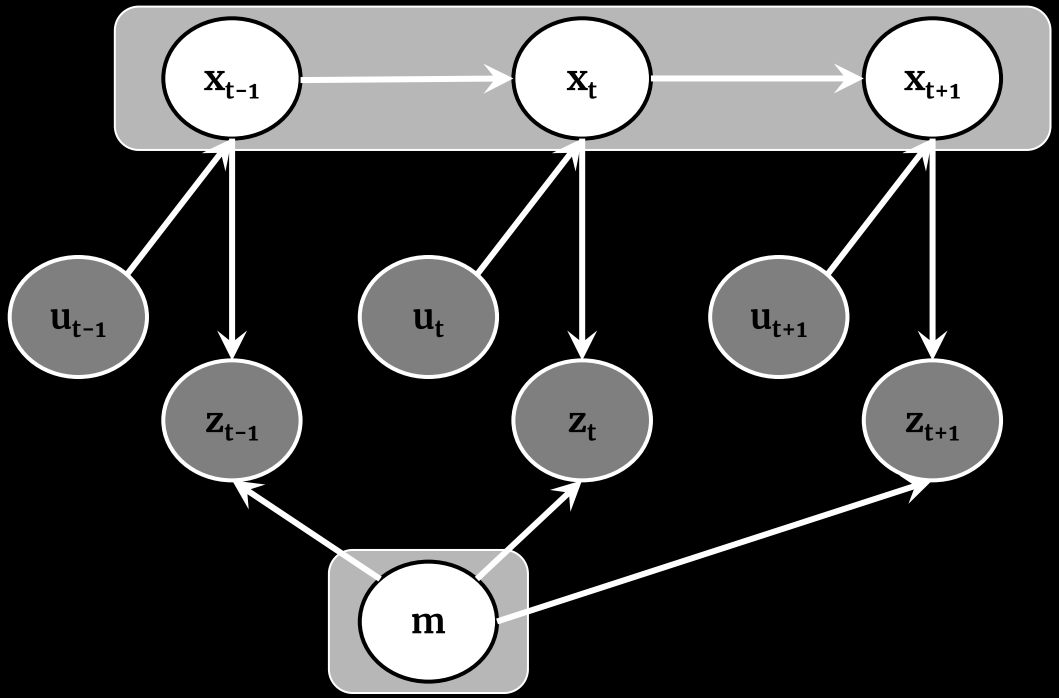
[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;...]

Simultaneous Localization and Mapping

- Nodes are random variables
- Directed edges are variable dependencies
- Gray nodes
  - Observed or directly measured variables
- White nodes
  - Inferred latent variables

- Full SLAM
  - Compute a joint posterior belief over the whole path of the robot and the map
- Online SLAM

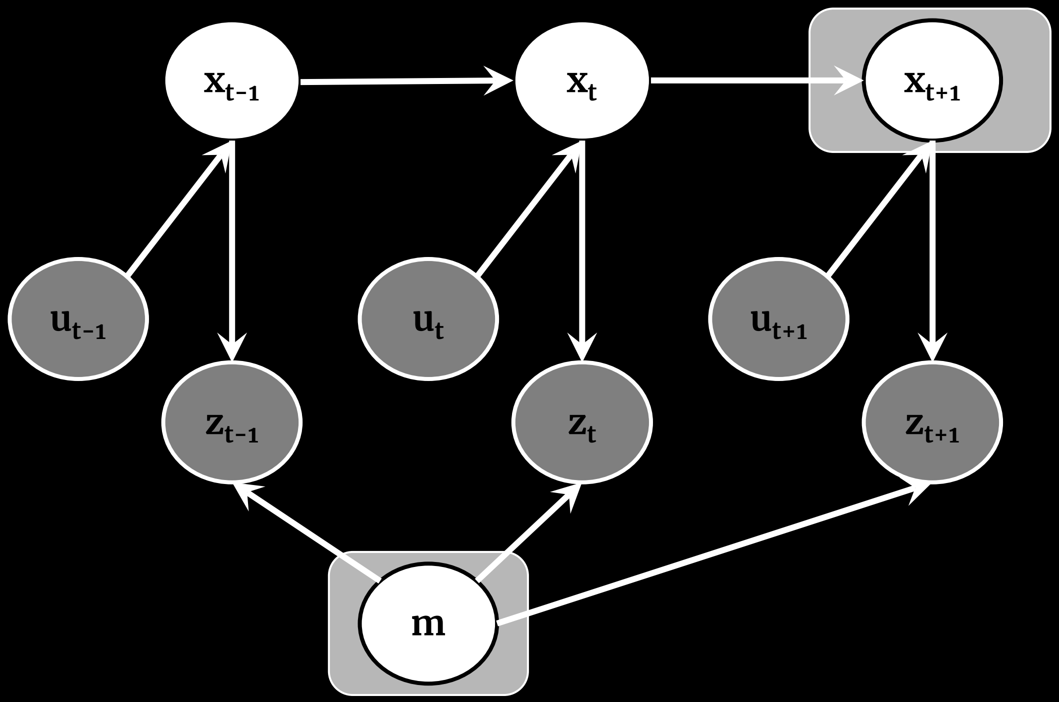


$$p(x_{1:t}, m | z_{1:t}, u_{1:t}, x_0)$$

Simultaneous Localization and Mapping

- Nodes are random variables
- Directed edges are variable dependencies
- Gray nodes
  - Observed or directly measured variables
- White nodes
  - Inferred latent variables

- Full SLAM
  - Compute a joint posterior over the whole path of the robot and the map
- Online SLAM
  - Compute a posterior over the current pose along with the map



$$p(x_t, m|z_{1:t}, u_{1:t})$$

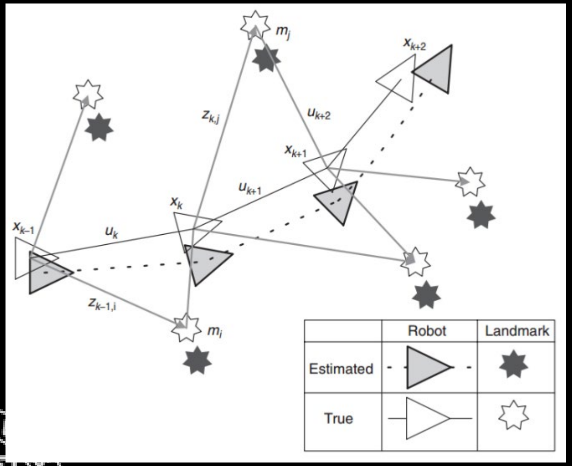
Simultaneous Localization and Mapping

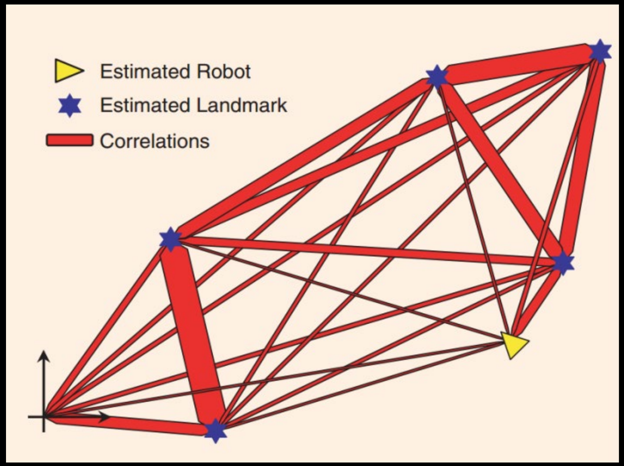
- Prediction (prediction step):
  - $p(x_t, m|z_{0:t}, u_{1:t}x_0) = \sum_{t-1} P(x_t|x_{t-1}, u_t)P(x_{t-1}, m|Z_{0:t-1}, U_{1:t}, x_0)$
- Correction (update step):
  - $p(x_t, m|z_{0:t-1}, u_{0:t}, x_0) = \eta P(z_t|x_t, m)P(x_t, m|Z_{0:t}, U_{1:t}, x_0)$

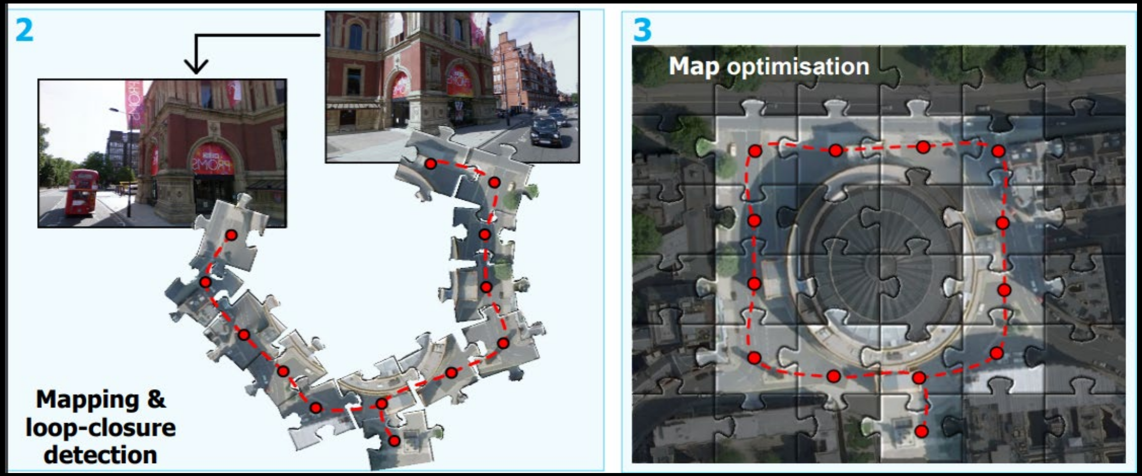
- We can solve the localization problem with the assumption that we know the map
  - $P(x_t|Z_{0:t}, U_{0:t}, m)$
- We can solve the mapping problem with the assumption that we know the location
  - $P(m|X_{0:t}, Z_{0:t}, U_{0:t})$

Simultaneous Localization and Mapping

- Robot observations of the relative landmark locations can be considered nearly independent, because the relative landmark locations are independent from the robot's coordinate frame
- Robot observations of the absolute landmark locations is less certain, because the absolute landmark location is strongly related to the robot's coordinate frame
- Because landmarks are correlated even unobserved landmarks can be updated, such that correlations are increased for every observation we make
- The accuracy of the relative map increases for more observations

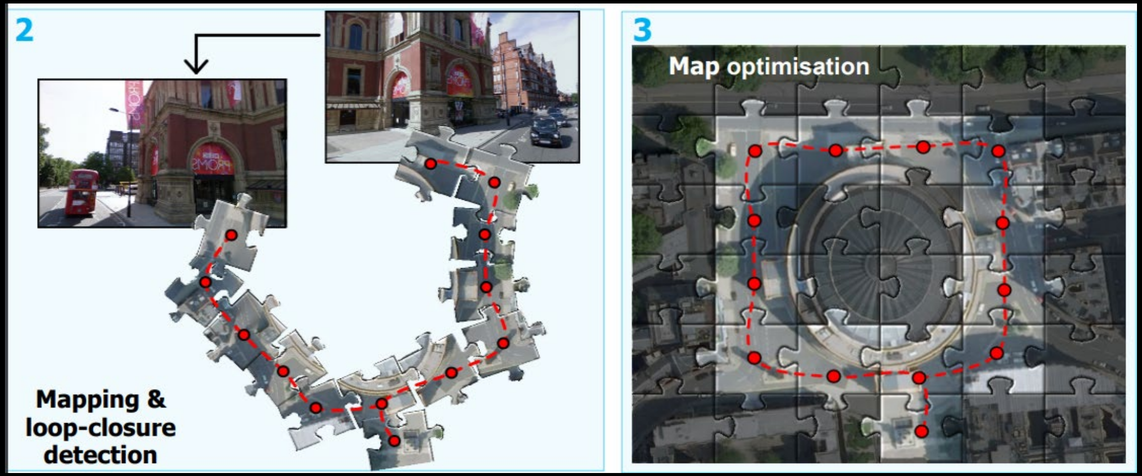






Simultaneous Localization and Mapping

- Why is it hard?
  - Map size
    - The larger the environment relative to the robot's perceptual range, the more difficult it is to acquire the map
  - Perceptual Ambiguity
    - The more different places look alike, the more difficult it is to establish correspondence between different locations traversed at different points in time
  - Cycles
    - Motion-cycles are particularly difficult to map



SLAM Solutions

- The trick is to find an appropriate representation for the observation and the motion problem
  - Graph SLAM → Global optimization: outputs the most likely map and trajectory
  - EKF SLAM
  - Fast SLAM

→ Probability distribution over landmarks and the most recent pose (online SLAM)

ECE 4160/5160

MAE 4910/5910

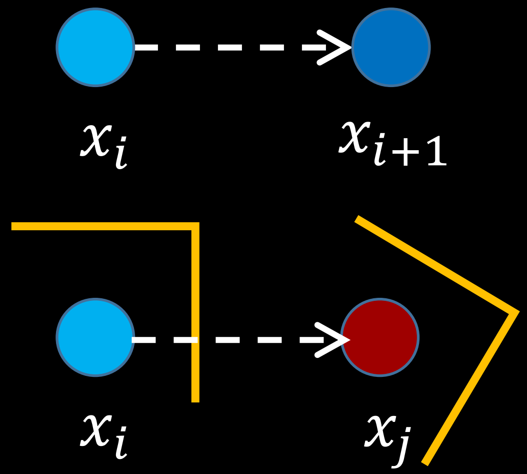
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots

Graph SLAM

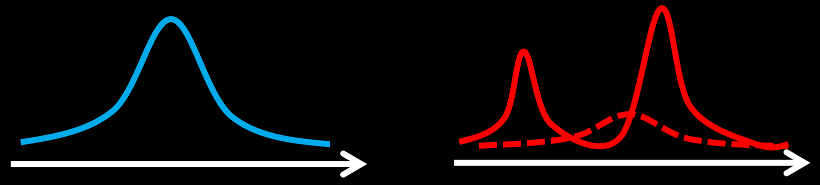
Graph SLAM

- Graph represents a set of objects where pairs of objects are connected by links encoding relations between them
- Create an edge if…
  - …the robot moves from x_i to x_{i+1}
    - (edge corresponds to odometry measurement)
  - …the robot observes the same part of the environment from x_i and from x_j
- Edges represent constraints
- Nodes represent the state (poses and landmarks)
  - Given a state, we can compute predicted observations
  - *Find a configuration of the nodes so that the real and predicted constraints are as similar as possible*
  - Minimize the Least Square Error over all constraints

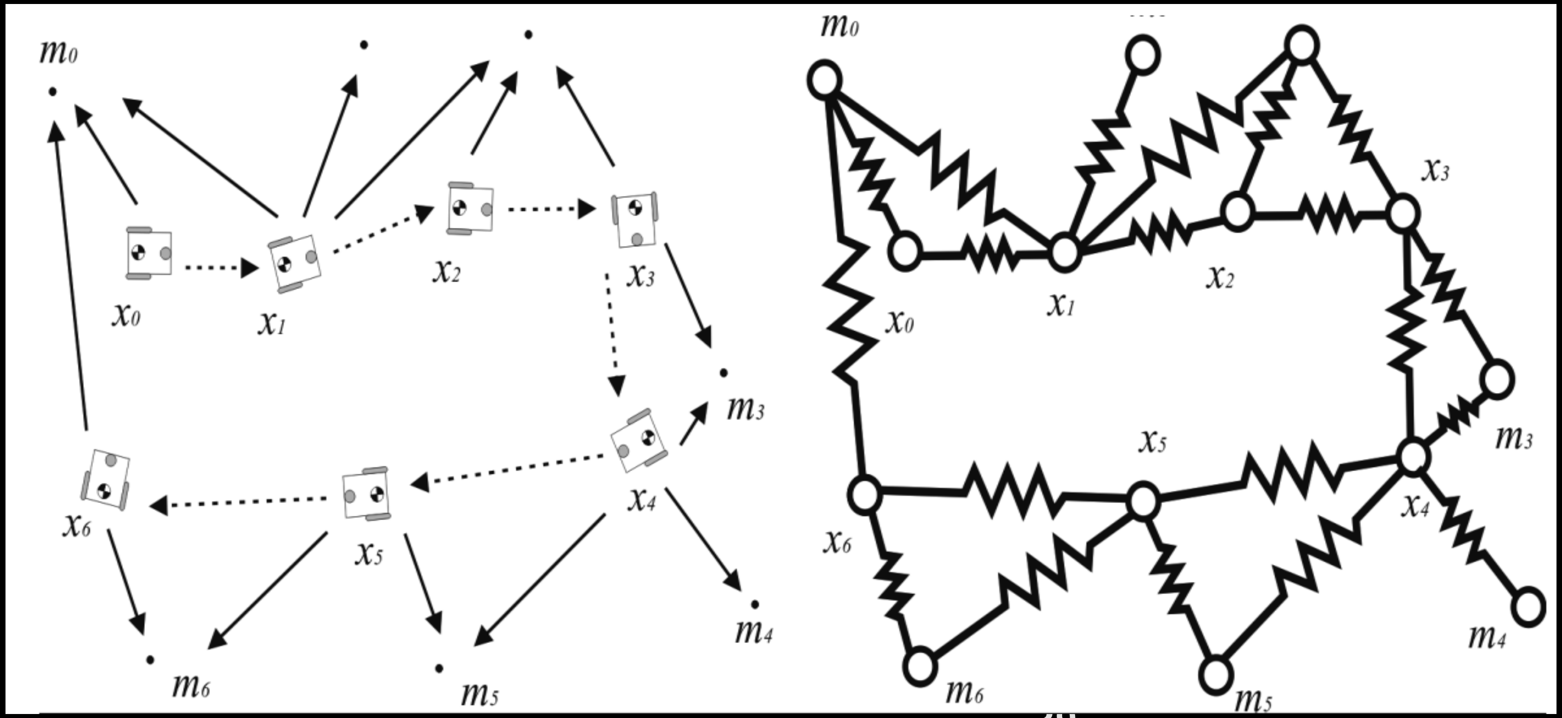


Graph-Based SLAM

- Treat constraints (generated by motions and observations) as elastic springs
- Minimize the energy in all the springs
- Any modern SLAM implementation has some version of this
  - Pro: Globally optimal
  - Con: BIG optimization problem, only one output



- Tricks
  - Combine poses over many time steps into single nodes to make the graph smaller
  - If you see the same landmark from several poses, you can get rid of the pose and add a stronger constraint between those landmarks



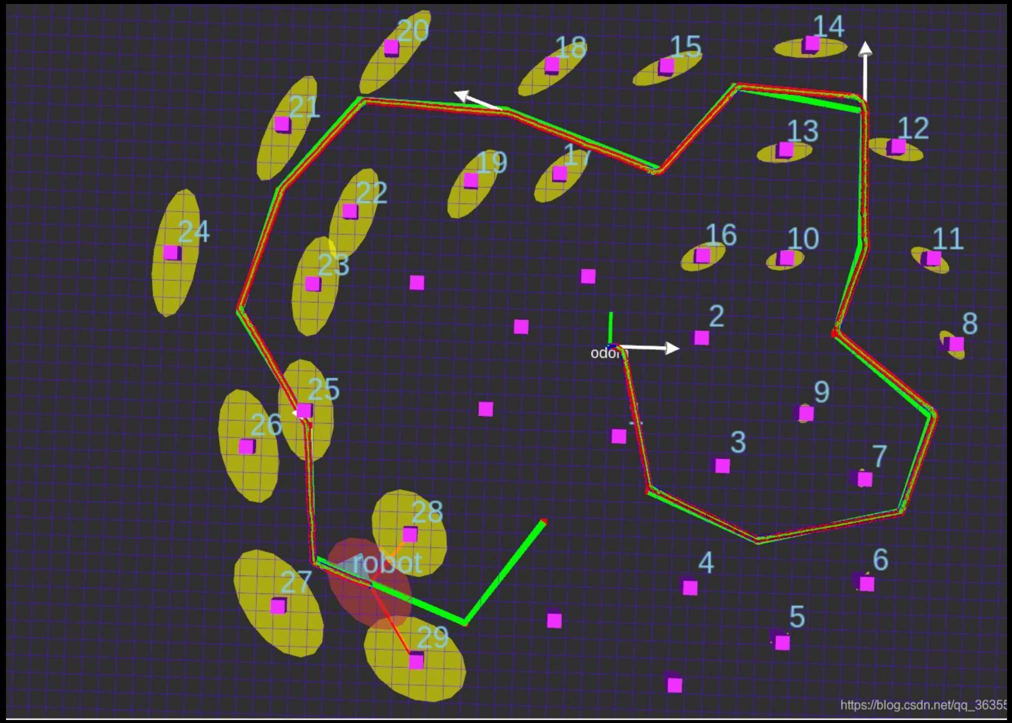
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots

EKF SLAM

EKF SLAM

- Goal: Estimate $p(x_k, m | u_{1:k}, z_{1:N})$
- Assume all noise is Gaussian
- Track a Gaussian belief of the current state and landmarks
- Apply the Kalman Filter…



Kalman Filter Implementation

State estimate: $\mu(t)$
State uncertainty: $\Sigma(t)$
Process noise: Σ_u
Kalman filter gain: K_{KF}
Measurement noise: Σ_z

Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, $u(t)$, $z(t)$)

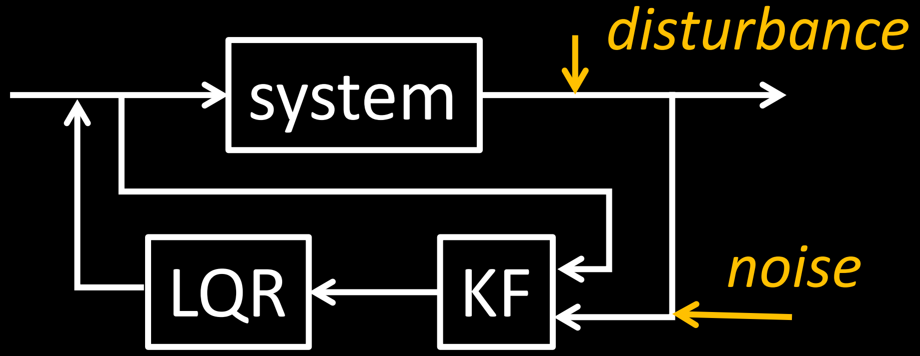
1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$

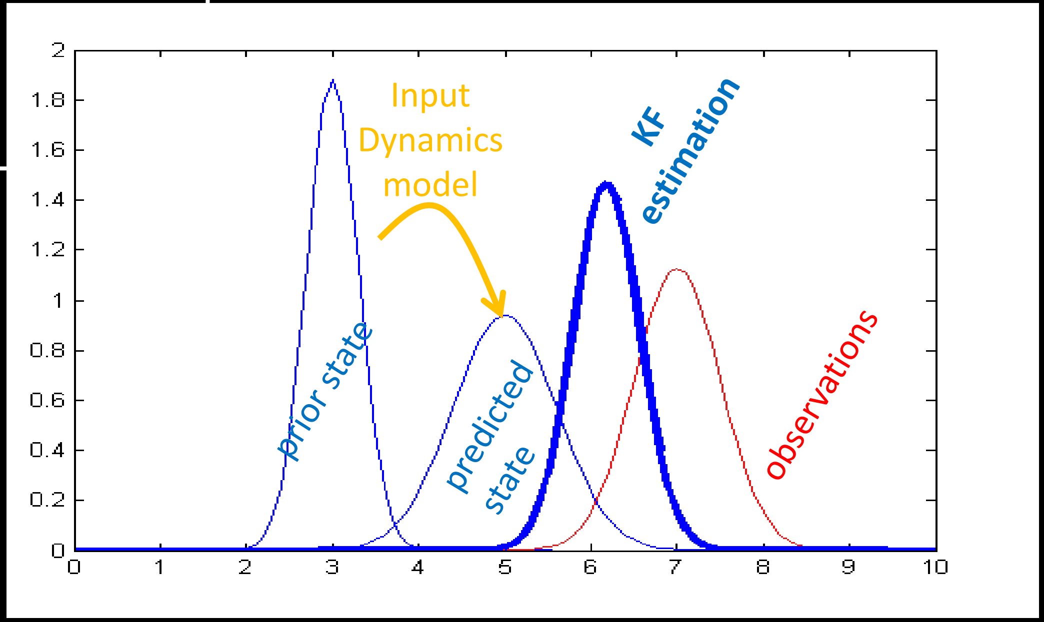
prediction

3. $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
5. $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$

update

6. Return $\mu(t)$ and $\Sigma(t)$





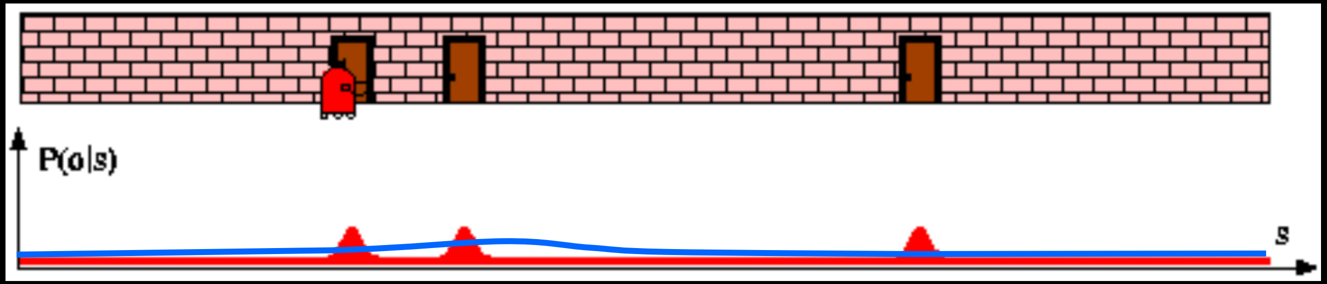
EKF SLAM

- Goal: Estimate $p(x_k, m | u_{1:k}, z_{1:N})$
- Assume all noise is Gaussian
- Track a Gaussian belief of the state and landmarks
- Linearize around every state and run the Kalman Filter
  - $x = \begin{bmatrix} \bar{\wp} \\ \bar{\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \wp \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix}$
  - $P = \begin{bmatrix} P_{\wp\wp} & P_{\wp\mathcal{M}} \\ P_{\mathcal{M}\wp} & P_{\mathcal{M}\mathcal{M}} \end{bmatrix} = \begin{bmatrix} P_{\wp\wp} & P_{\wp\mathcal{L}_1} & \cdots & P_{\wp\mathcal{L}_n} \\ P_{\mathcal{L}_1\wp} & P_{\mathcal{L}_1\mathcal{L}_1} & \cdots & P_{\mathcal{L}_1\mathcal{L}_n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{\mathcal{L}_1\wp} & P_{\mathcal{L}_n\mathcal{L}_1} & \cdots & P_{\mathcal{L}_n\mathcal{L}_n} \end{bmatrix}$

- Landmark matrix grows, making the inversion step costly!
- (in Full SLAM the trajectory matrix grows even faster)

EKF SLAM

- Goal: Estimate $p(x_k, m | u_{1:k}, z_{1:N})$
- Track a Gaussian belief of the state and landmarks
- Assume all noise is Gaussian
- Linearize around every state and run the Kalman Filter
- Pros
  - Super easy, well understood, runs online
  - Works well for low-uncertainty problems
- Cons
  - Works poorly for high-uncertainty problems
  - (States must be well-approximated by a Gaussian)



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ECE 4160/5160
MAE 4910/5910

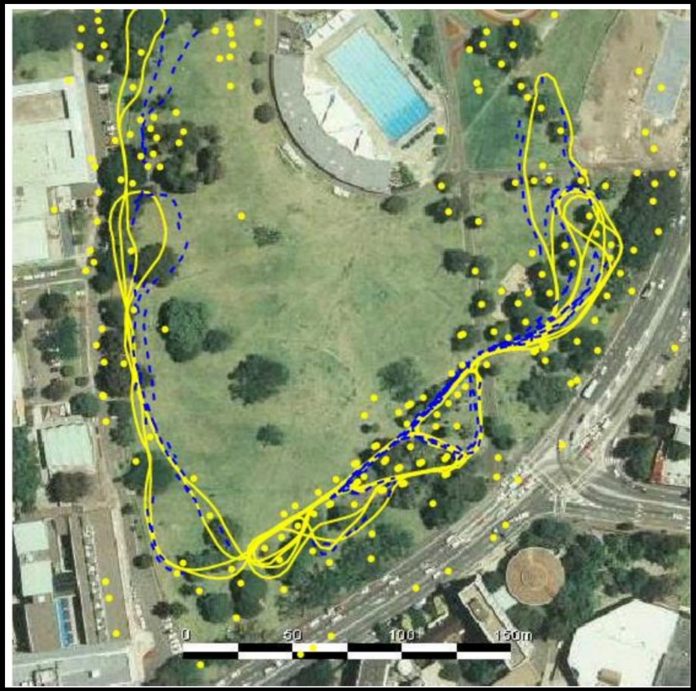
Fast Robots

Fast SLAM

Fast SLAM

- Half sample-based solution
  - Particle filter
    - Every particle has its own version of the map with a given trajectory
- Half analytical solution
  - Landmark-based
    - Each pose and map of independent features is updated analytically through EKF
  - Grid-map based
    - Occupancy of each grid cell is estimated by Bayes Filter

Victoria Park dataset
University of Sydney



GPS
FastSLAM

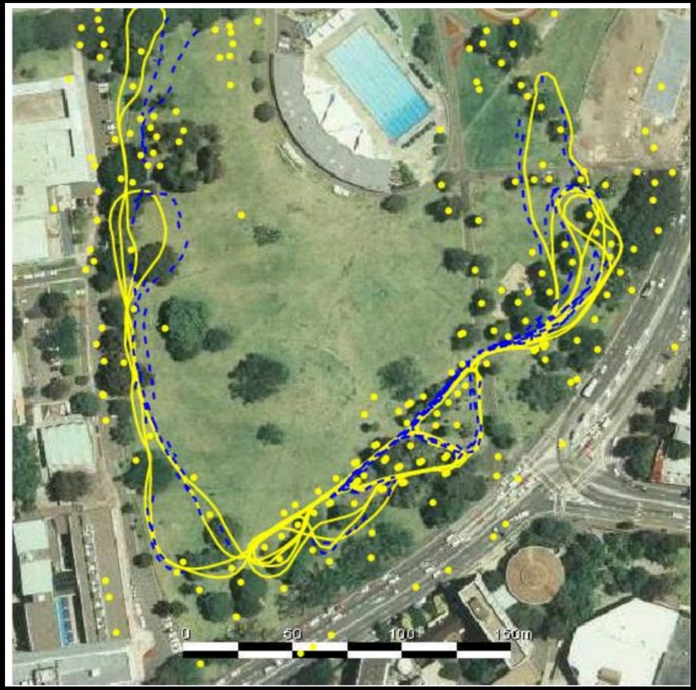
4km traverse
100 particles
<5m RMS position error

Fast SLAM

- Key idea: factorize the posterior
  - $p(x_{1:k}, m | z_{1:k}) = p(m | x_{1:k}, z_{1:k}) p(x_{1:k} | z_{1:k})$
- $p(x_{1:k} | z_{1:k})$: pose estimation is approximated by the Particle Filter
  - (can represent multiple hypotheses)
- $p(m | x_{1:k}, z_{1:k})$: classic mapping problem, appr. using EKF
  - (efficient at representing belief in high dimensions)

- Outcome is a Marginalized Particle Filter (MPF)
  - Each particle is a pose trajectory with an attached map corresponding to mean and covariance of each landmark

Victoria Park dataset
University of Sydney

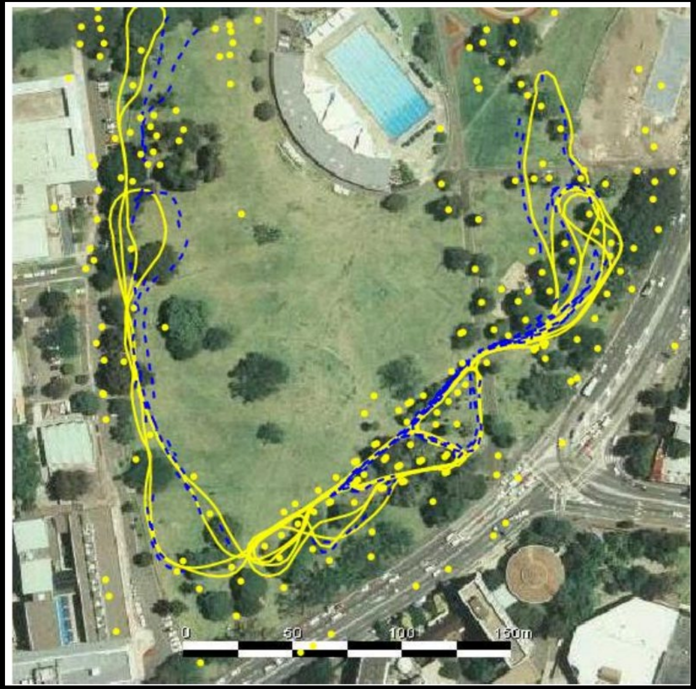


GPS
FastSLAM
4km traverse
100 particles
<5m RMS position error

Fast SLAM

- Distribution is estimated by a fixed number of particles
  - Each particle, k, contains an estimate of robot path and the mean and covariance of each of the n features
    - $P^{[k]}(x_t^{[k]}; \mu^{[k]}, \Sigma_1^{[k]}; \dots \mu^{[k]}, \Sigma_n^{[k]})$
- **Step 1:** Update particle trajectory (motion model)
- **Step 2:** Update particle landmarks with EKF (sensor model)
  - Linearize the observation model at $(x_t^{[k]}, m)$
  - Only updated associated landmarks
- **Step 3:** Update weights based on $p(z_t | x_t^{[k]}, m^{[k]})$
- **Step 4:** Resample distribution

Victoria Park dataset
University of Sydney



GPS
FastSLAM
4km traverse
100 particles
<5m RMS position error

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

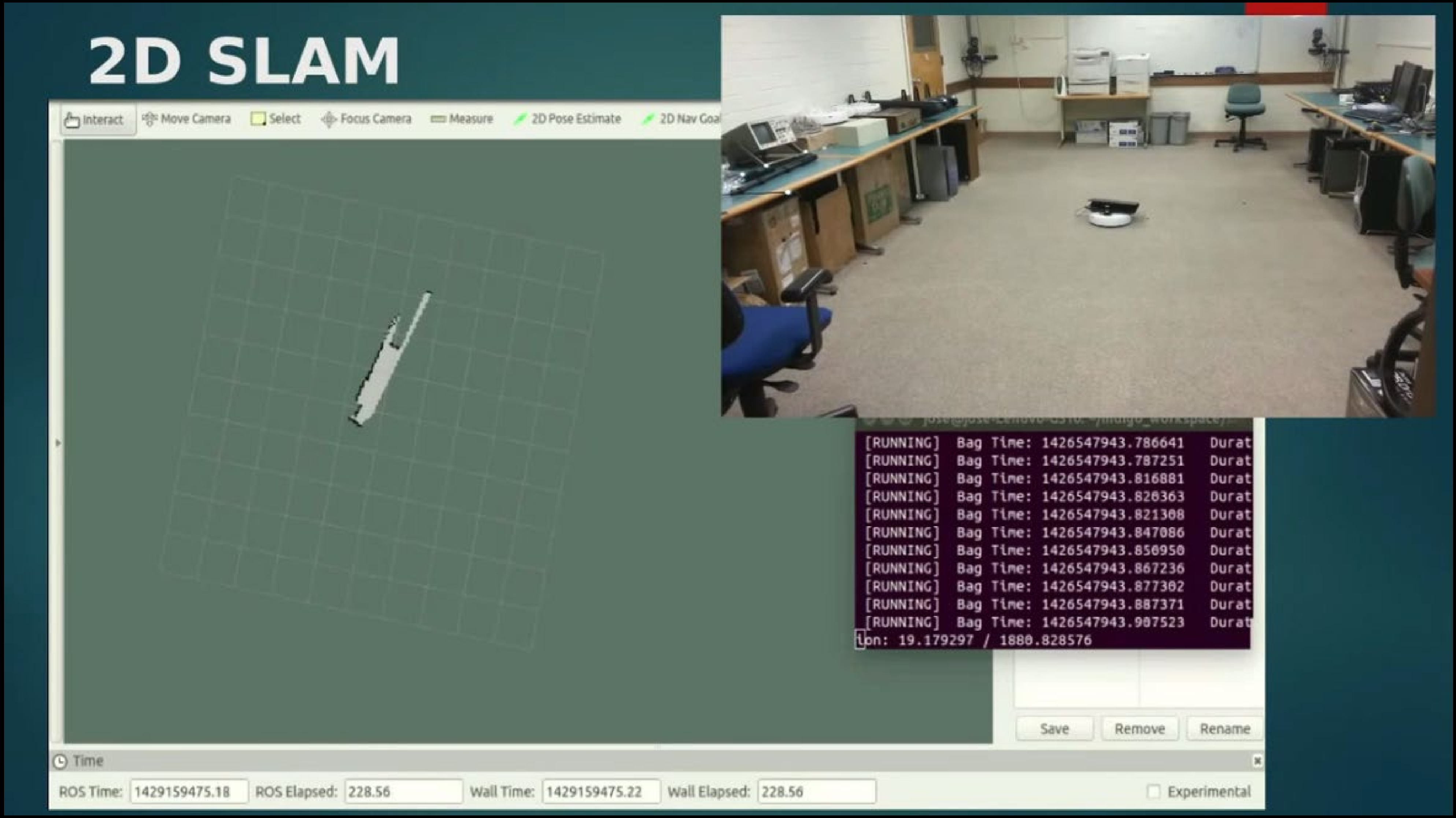
ECE 4160/5160
MAE 4910/5910

Fast Robots

SLAM State of the Art

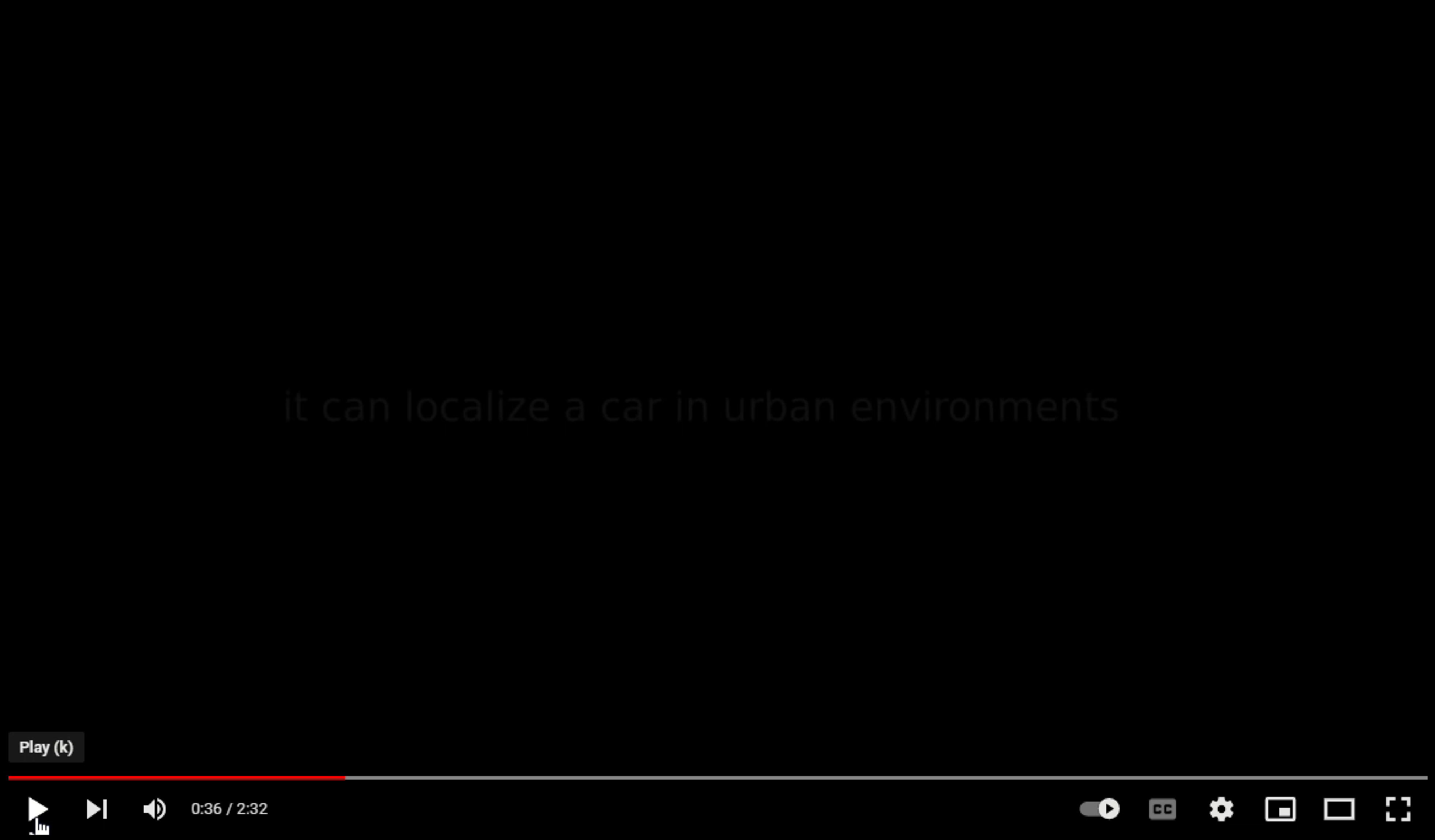
https://www.youtube.com/watch?v=6pRAhfBMW8w

State of the Art in SLAM



https://www.youtube.com/watch?v=ufvPS5wJAx0

State of the Art in SLAM

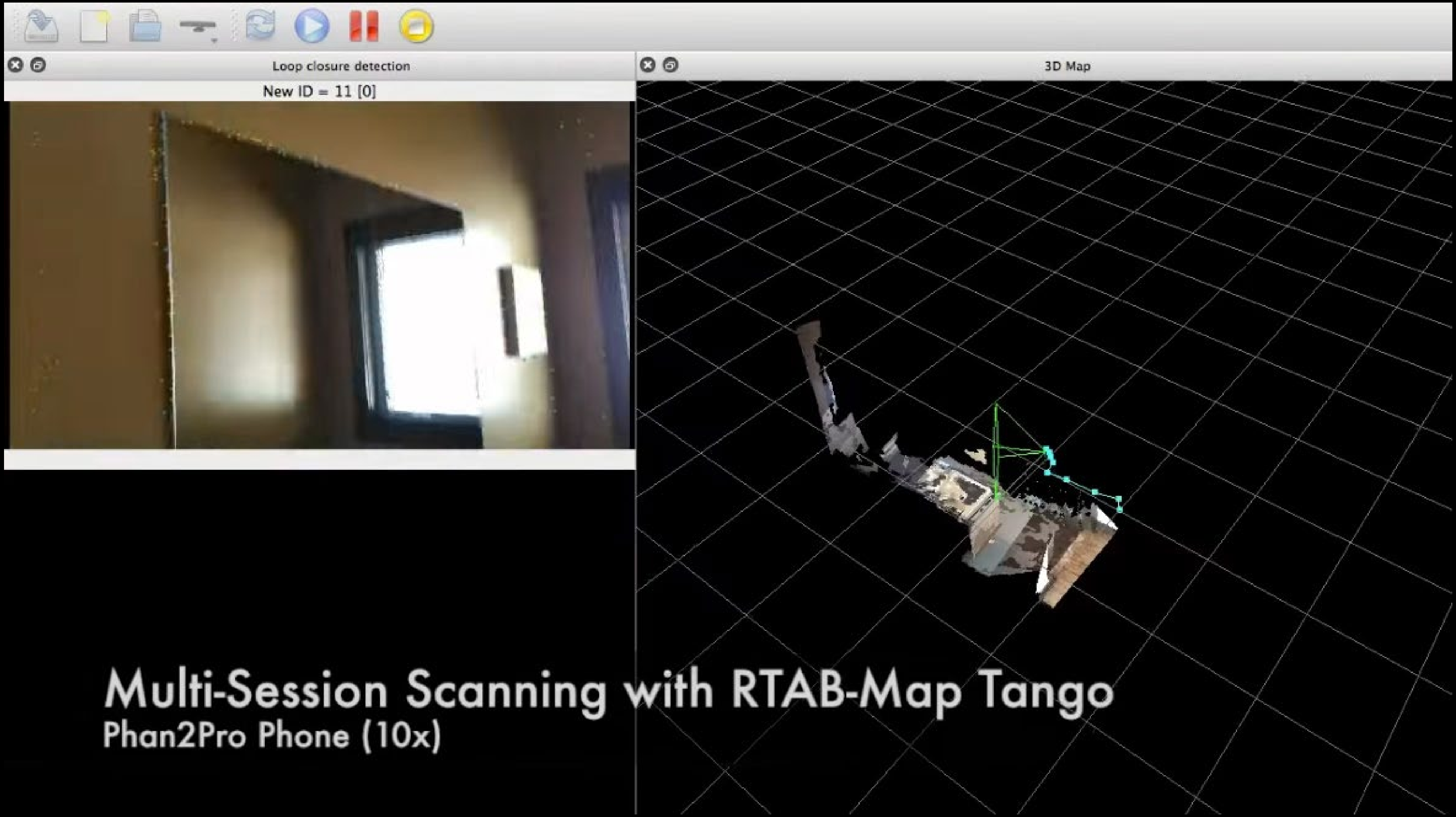


State of the Art in SLAM

- Robotics
  - 3D cameras with depth maps and high frame rates and resolution
  - Dense 3D models of the world
  - Uses ROS and deep learning to recognize features
  - Come built-in in a range of robots
  - Inherent to e.g. the RealSense tracking cameras
- 3D scanning/reconstruction
- Virtual and augmented reality

State of the Art in SLAM

https://www.youtube.com/watch?v=71eRxTc1DaU



Logistics

- Thursday Apr 20th – Tuesday Apr 25th:
  - Ethics I and II
    - Justice, Utilitarian, and Totalitarian methods
    - Case studies
- Thursday Apr 27th:
  - Trivia
- Tuesday May 2nd – Thursday May 4th:
  - ASML
  - Vecna Robotics
- Tuesday May 9th
  - Final Showcase 8.30-11am