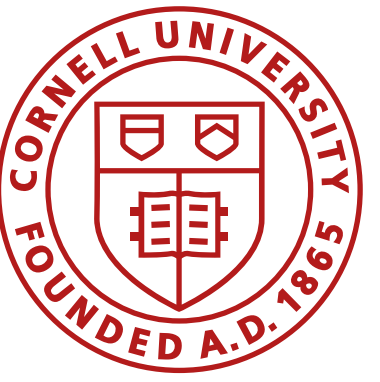


Particle Filters and SLAM

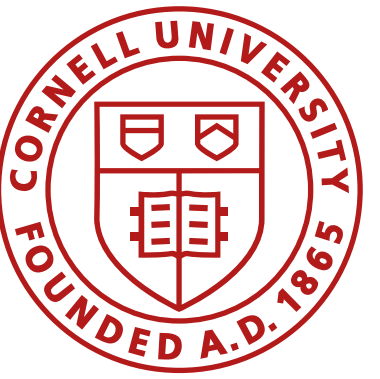
Fast Robots, ECE4160/5160, MAE 4190/5190

E. Farrell Helbling, 4/16/26



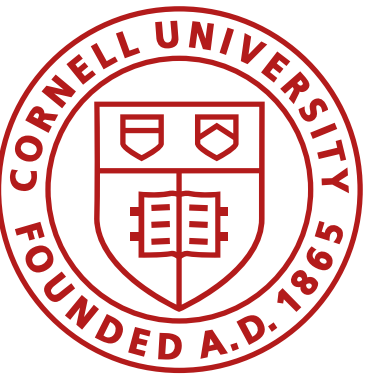
Class Action Items

- Lab 10: localization (sim) is this week!
 - The lab is graded S/U (all of the code is already posted on previous websites)
 - If you do get an unsatisfactory, you will need to redo it.
- Lab 11 is already posted if you want to get a jump start.
- Google Form sent yesterday
 - Lab 8 votes on stunts and bloopers



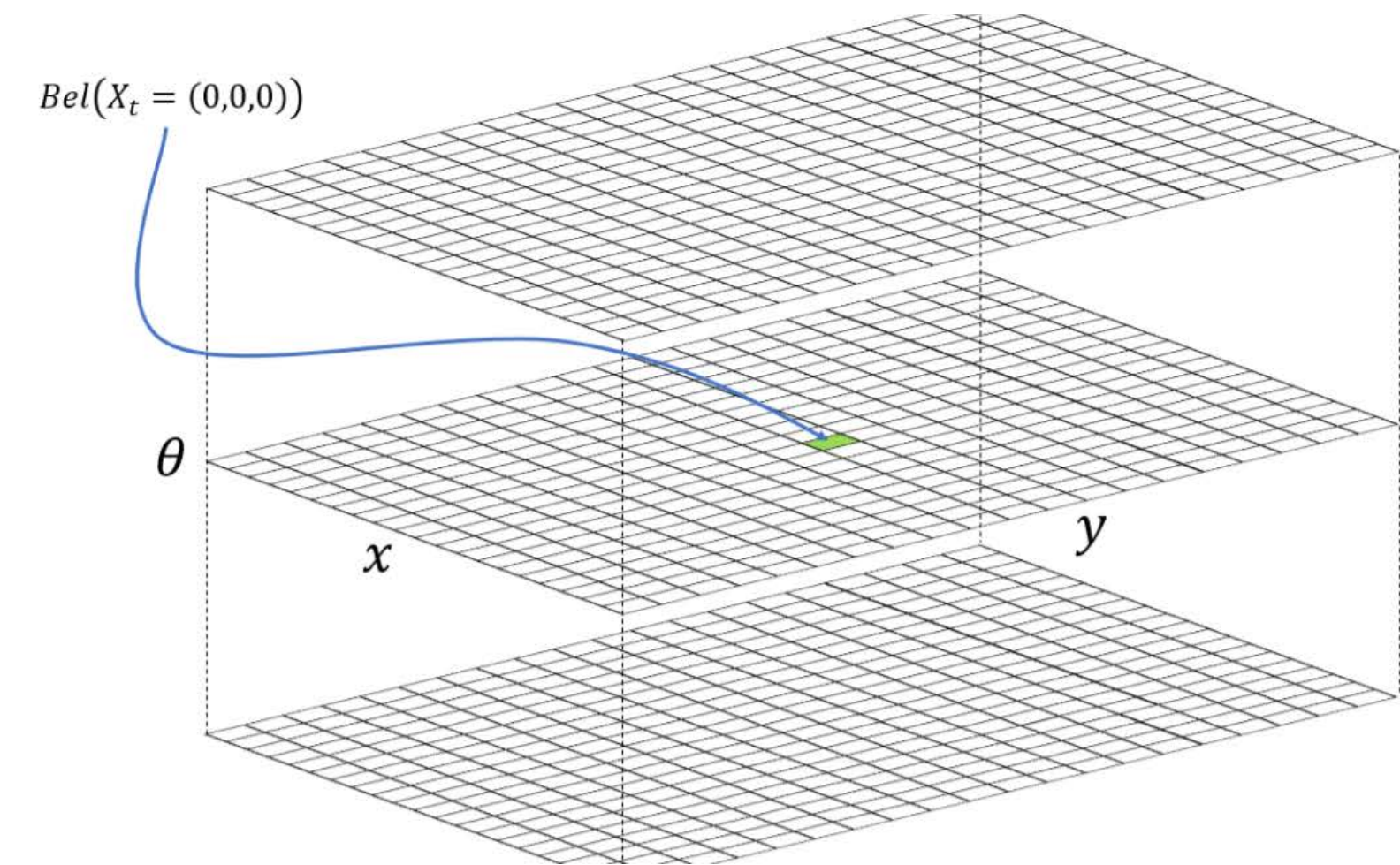
Lab 10 Localization

- This lab is graded S/U
- Tasks
 - Read the full lab and the notebook (before you show up to lab)
 - Perform grid localization for the sample trajectory
 - Video demo
 - Discuss:
 - Control
 - Motion model and the prediction step
 - Sensor model and the update step
 - Choosing parameters, effect of changing parameters
 - Ways to mitigate computational load
 - Evaluate the Bayes Filter
 - Evaluate how well this will work on your robot



Grid-based localization

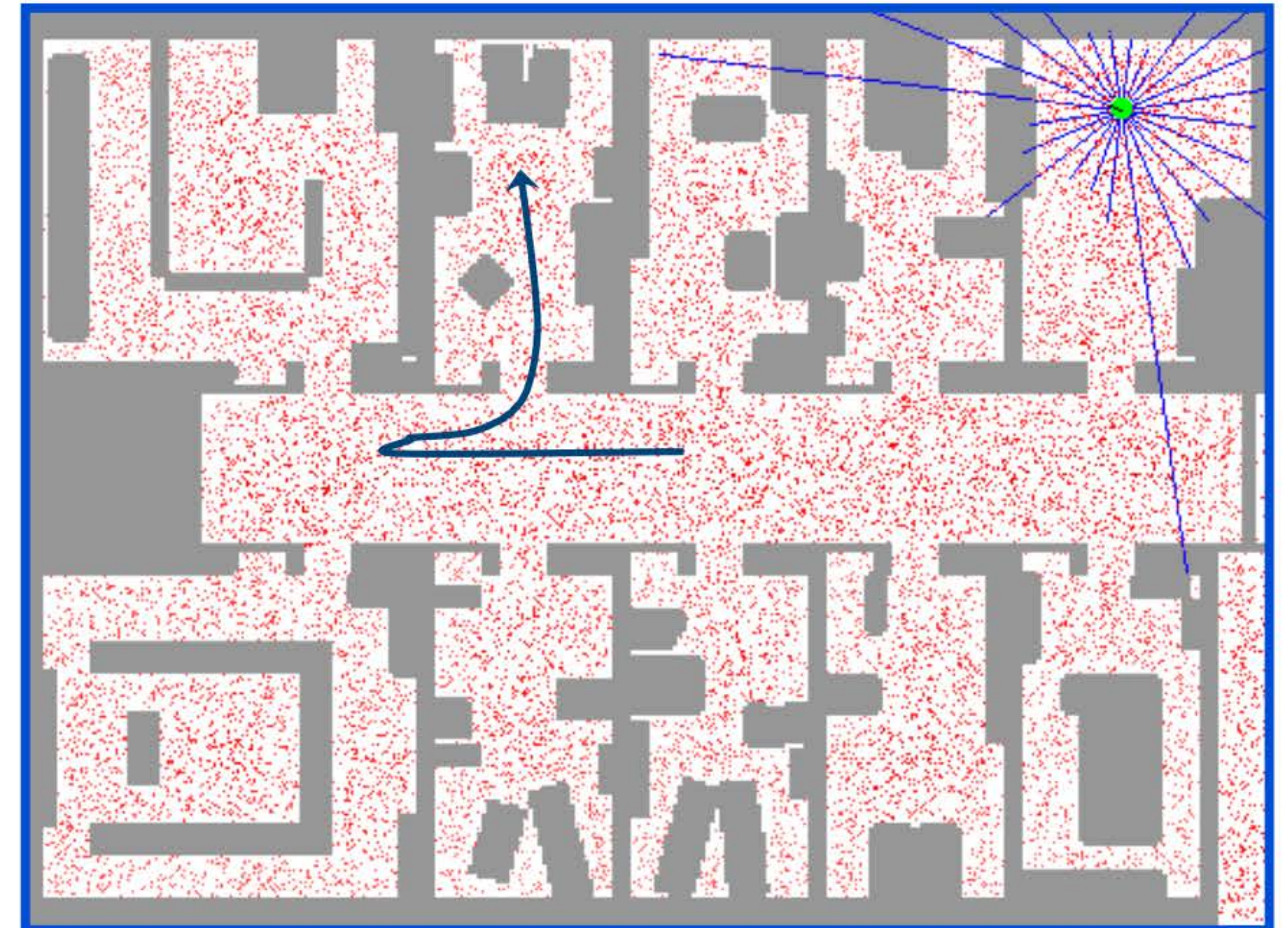
- Simple
- ... but it is computationally expensive for large workspaces

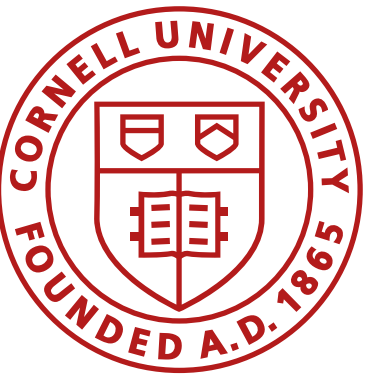


1. **Algorithm Bayes_Filter** ($bel(x_{t-1}), u_t, z_t$) :
2. **for** all x_t **do**
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t | u_t, x_{t-1}) bel(x_{t-1})$
4. $bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t)$
5. **end for**
6. **return** $bel(x_t)$

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**

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

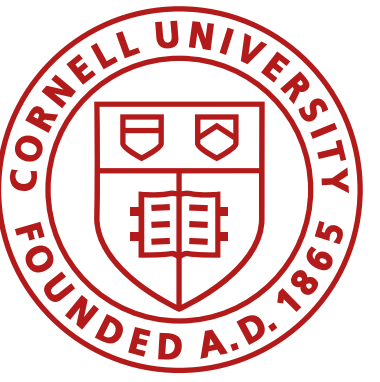
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t | u_t, x_{t-1}) bel(x_{t-1})$ ← Prior samples

4. $bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t)$

5. **end for**

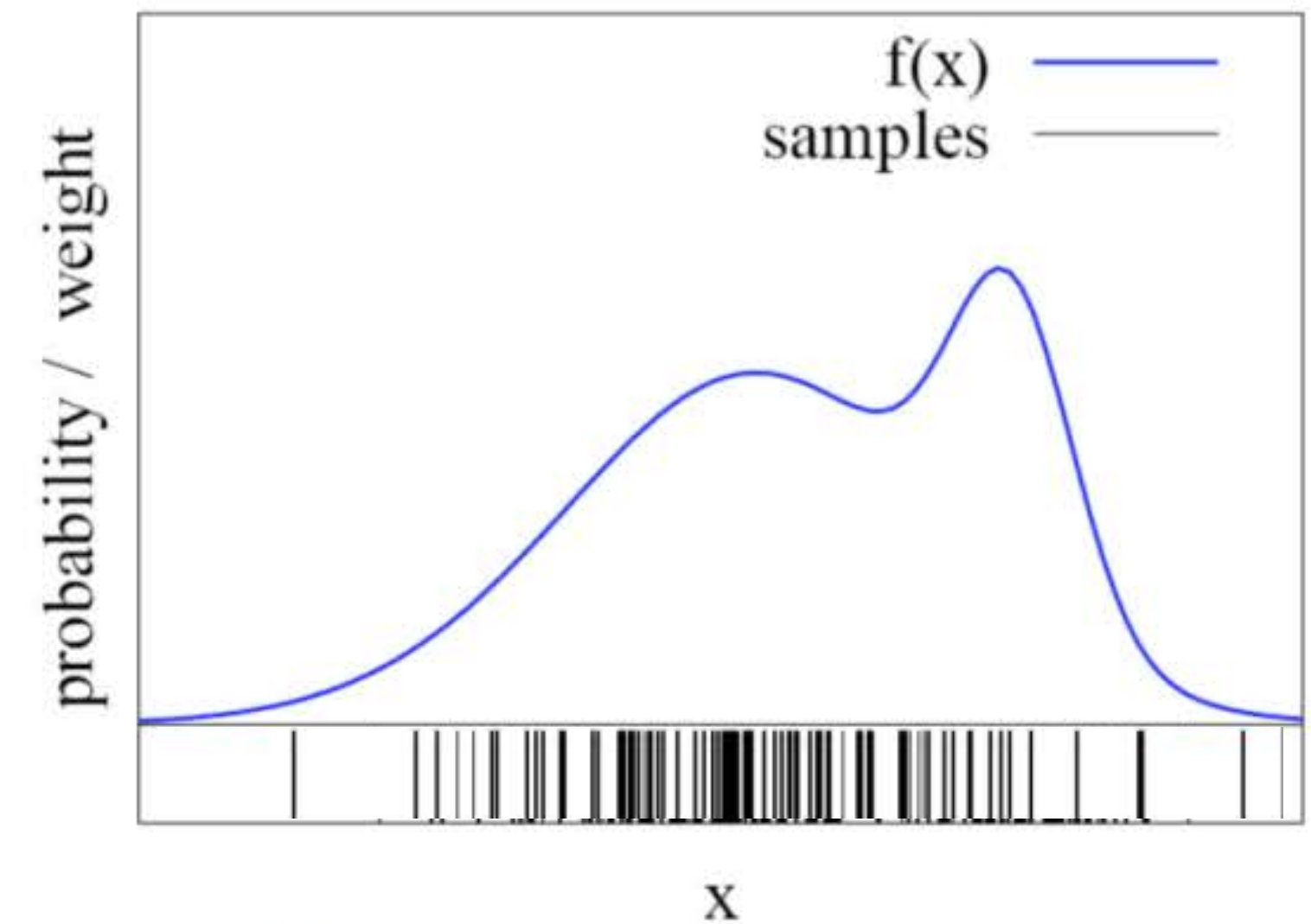
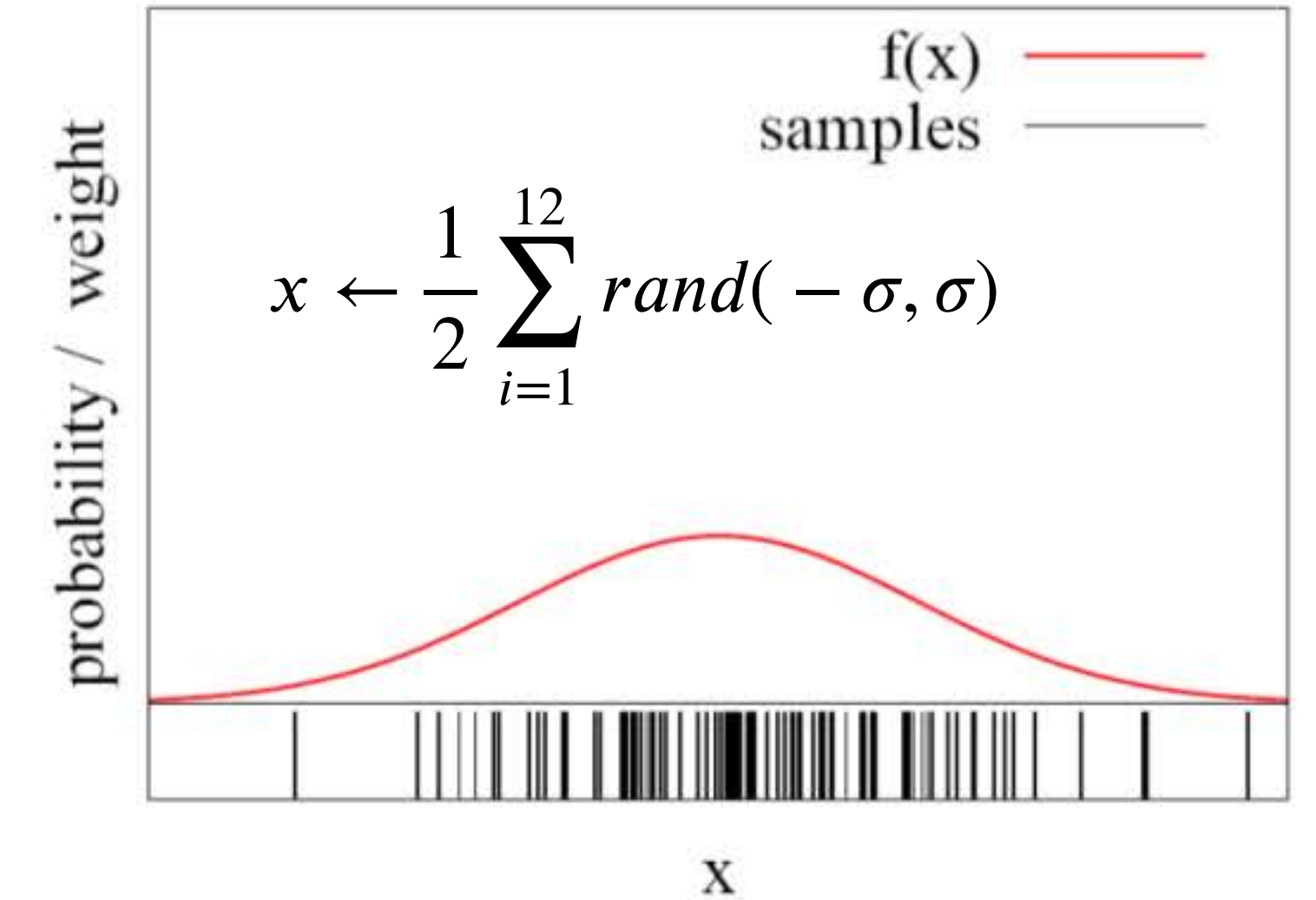
Importance factor $w_t^i \propto p(z_t | x_t)$

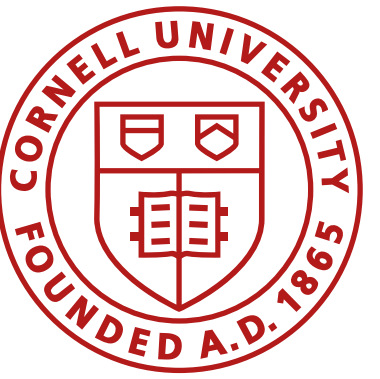
6. **return** $bel(x_t)$



Monte Carlo Localization

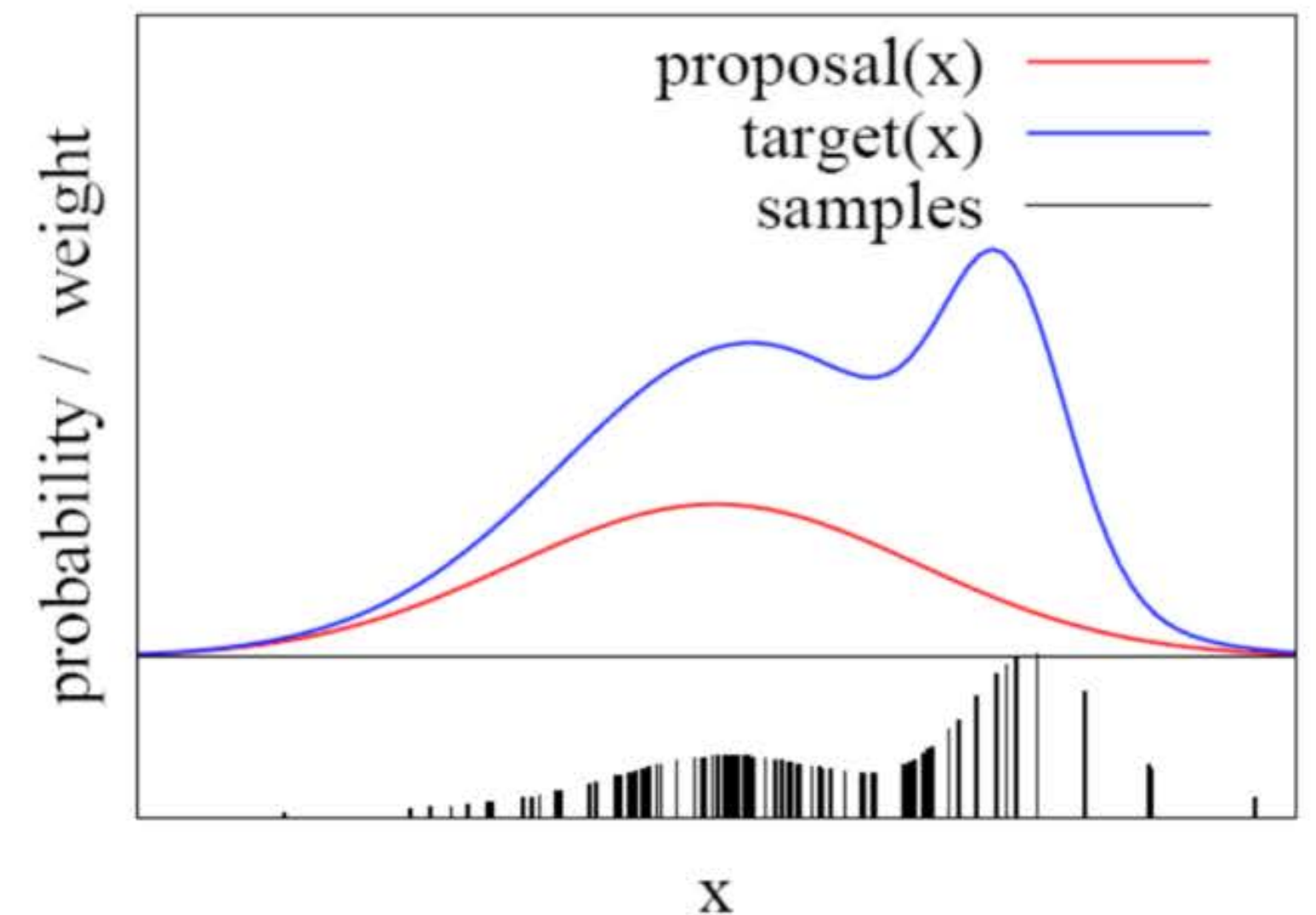
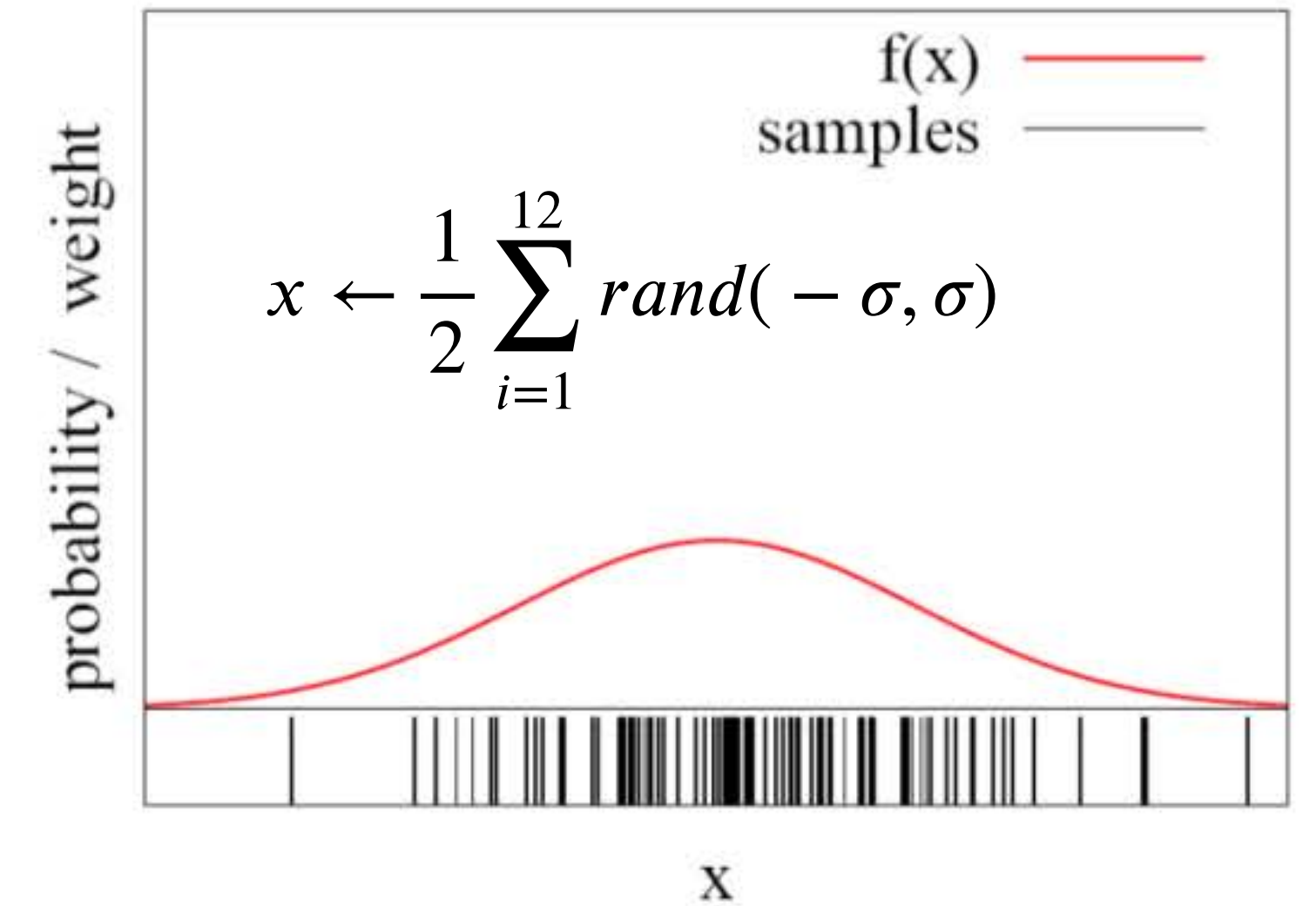
- How do you obtain samples from an arbitrary distribution?
 - Closed form solution for a uniform distribution
 - Closed form solution for a Gaussian distribution

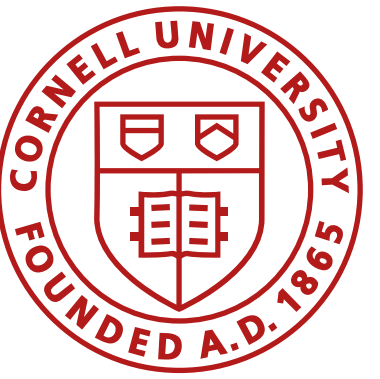




Monte Carlo Localization

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





Monte Carlo Localization

- 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

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

- Resample

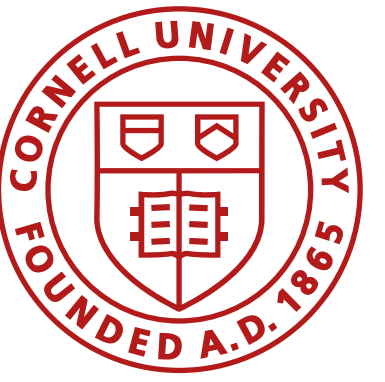
- Draw sample i with probability $w_t^{[j]}$ and repeat J times

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

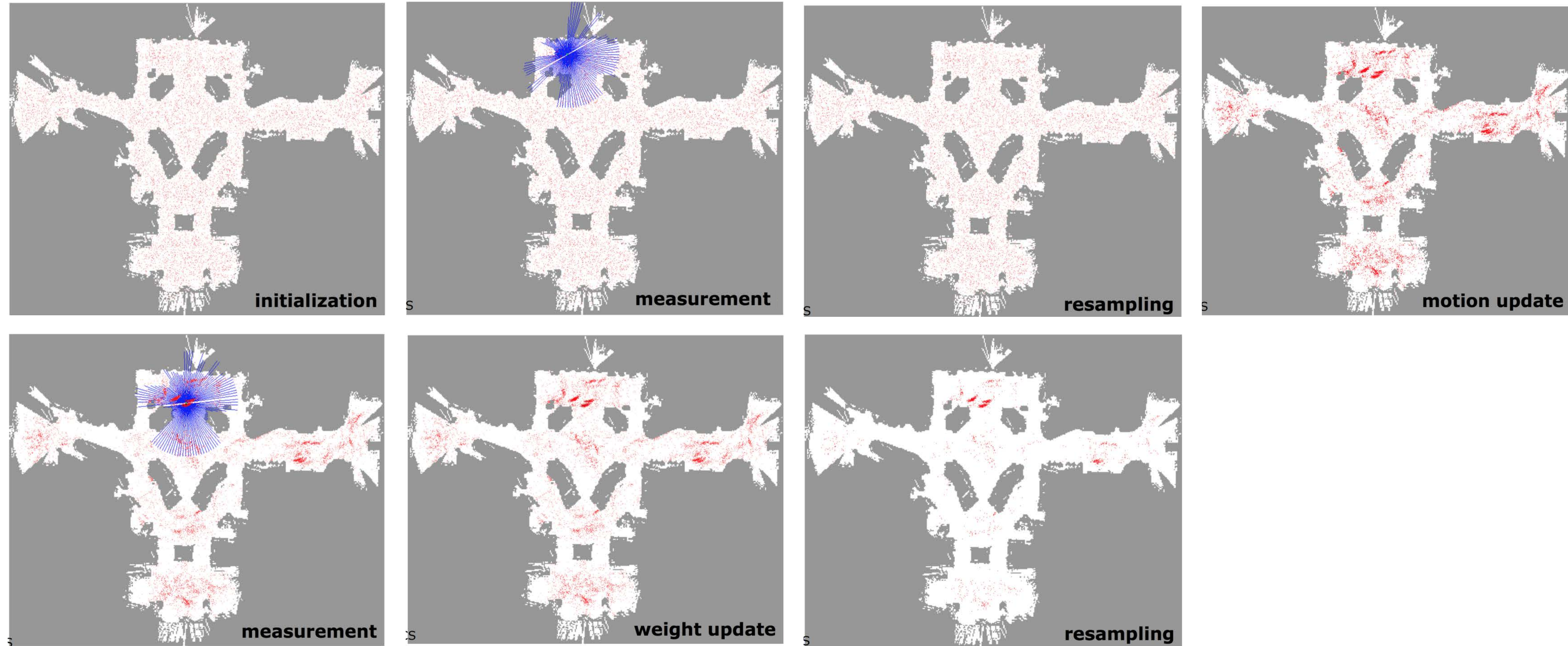
```

1:    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
2:   for  $j = 1$  to  $J$  do
3:       sample  $x_t^{[j]} \sim p(x_t | u_t, x_{t-1}^{[j]})$ 
4:        $w_t^{[j]} = p(z_t | x_t^{[j]})$ 
5:        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[j]}, w_t^{[j]} \rangle$ 
6:   endfor
7:   for  $j = 1$  to  $J$  do
8:       draw  $i \in 1, \dots, J$  with probability  $\propto w_t^{[i]}$ 
9:       add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
10:  endfor
11:  return  $\mathcal{X}_t$ 

```

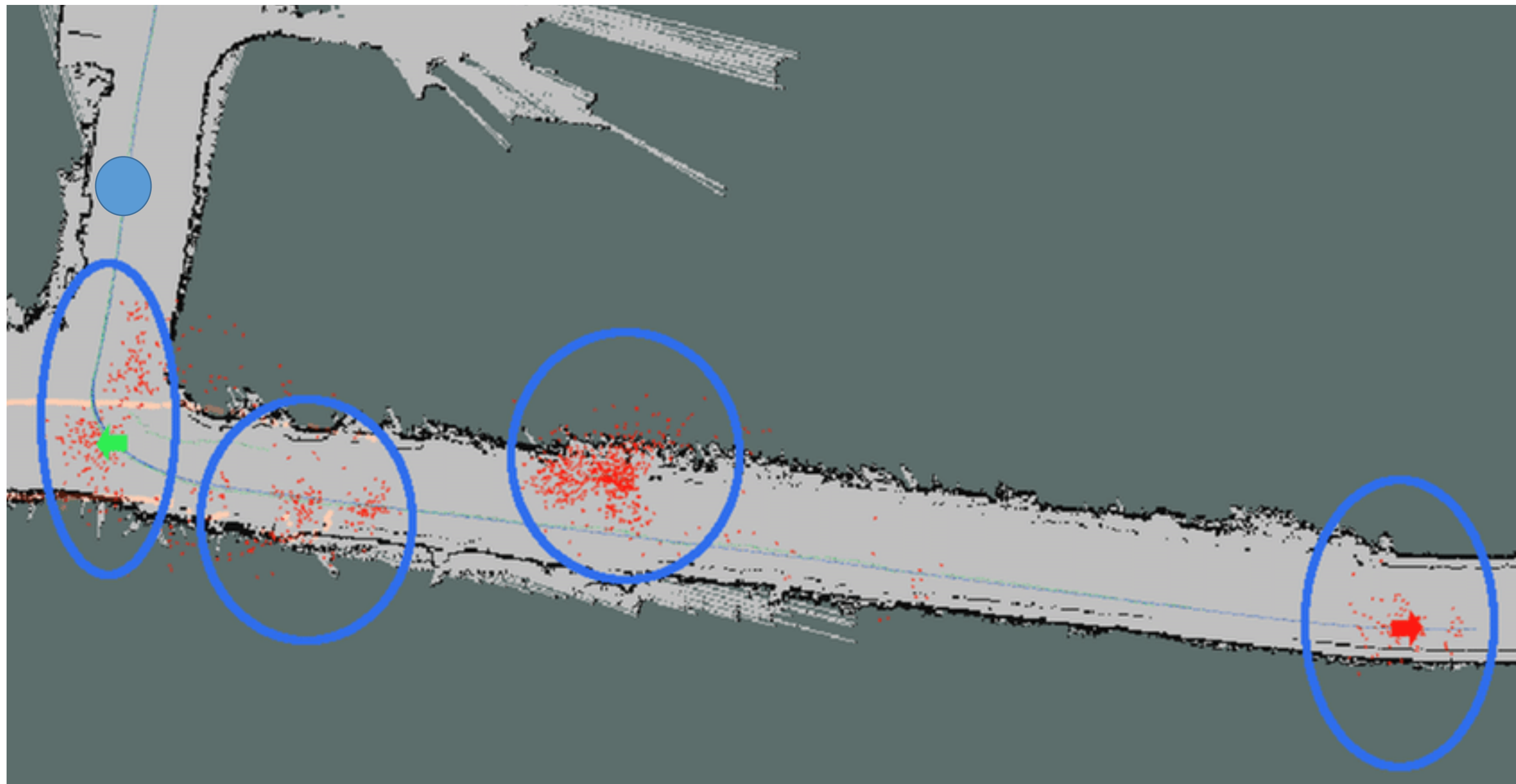


Monte Carlo Localization

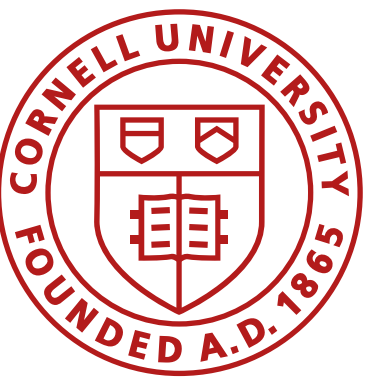


Monte Carlo Localization

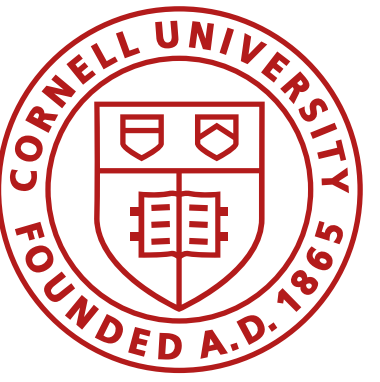
- How would you deal with a kidnapped robot?
 - Randomly insert samples proportional to the average likelihood of the particles



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



Brief intro to SLAM

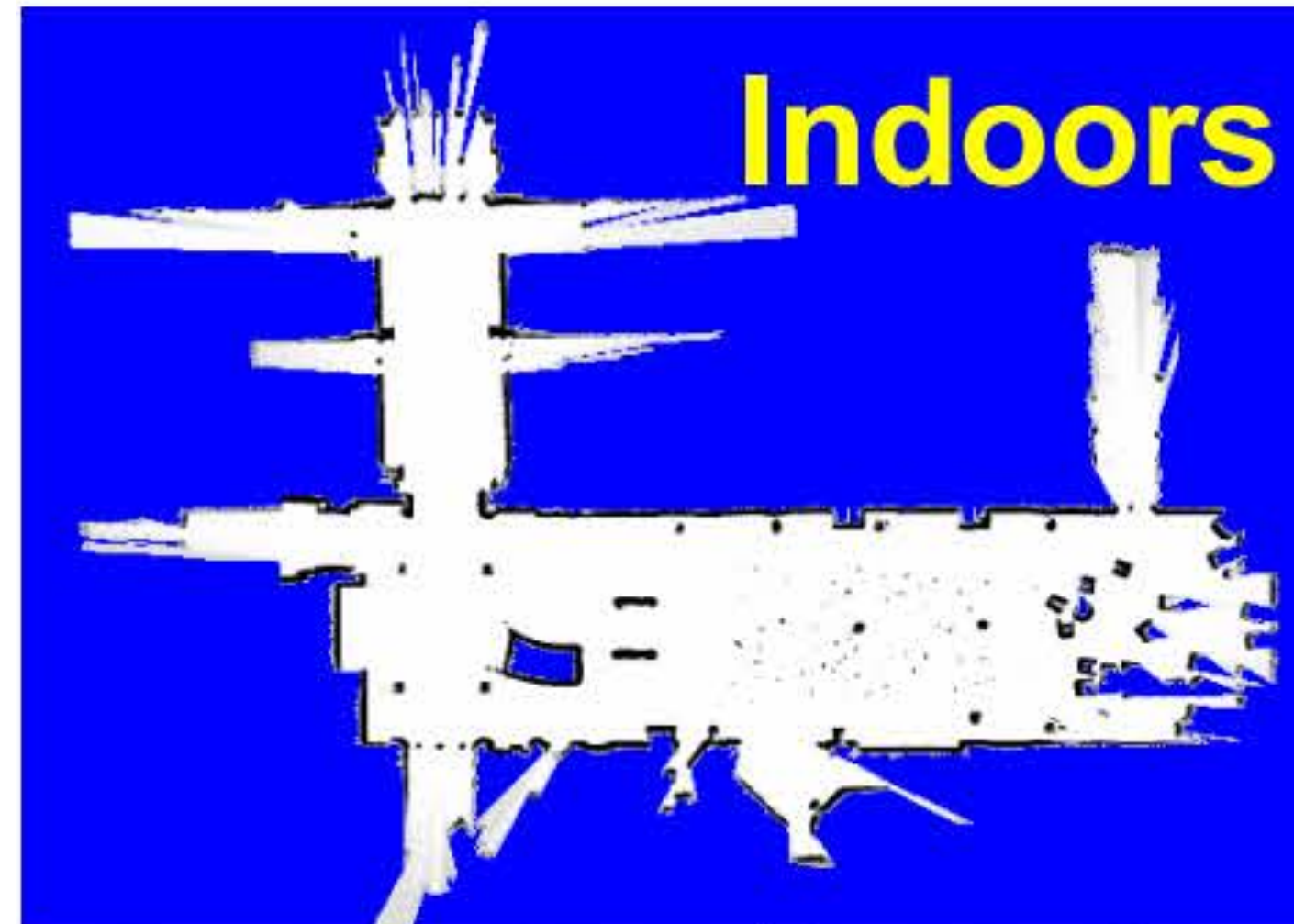


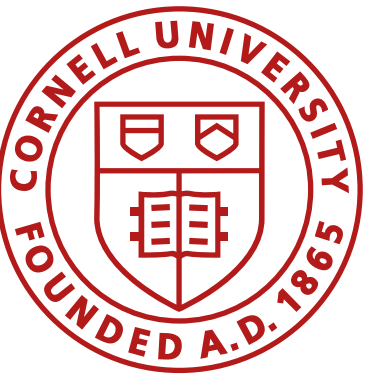
Related terms

- 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



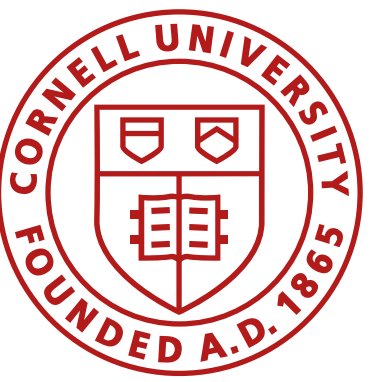


Given all we have learned...

- Transformation matrices
- Sensors and actuators (and probabilistic models)
- Controllers (PID, LQR)
- Observers (KF)
- Mapping
- Localization
 - Bayes Filter and grid-localization
 - Particle Filter
- Graph Search and Planning

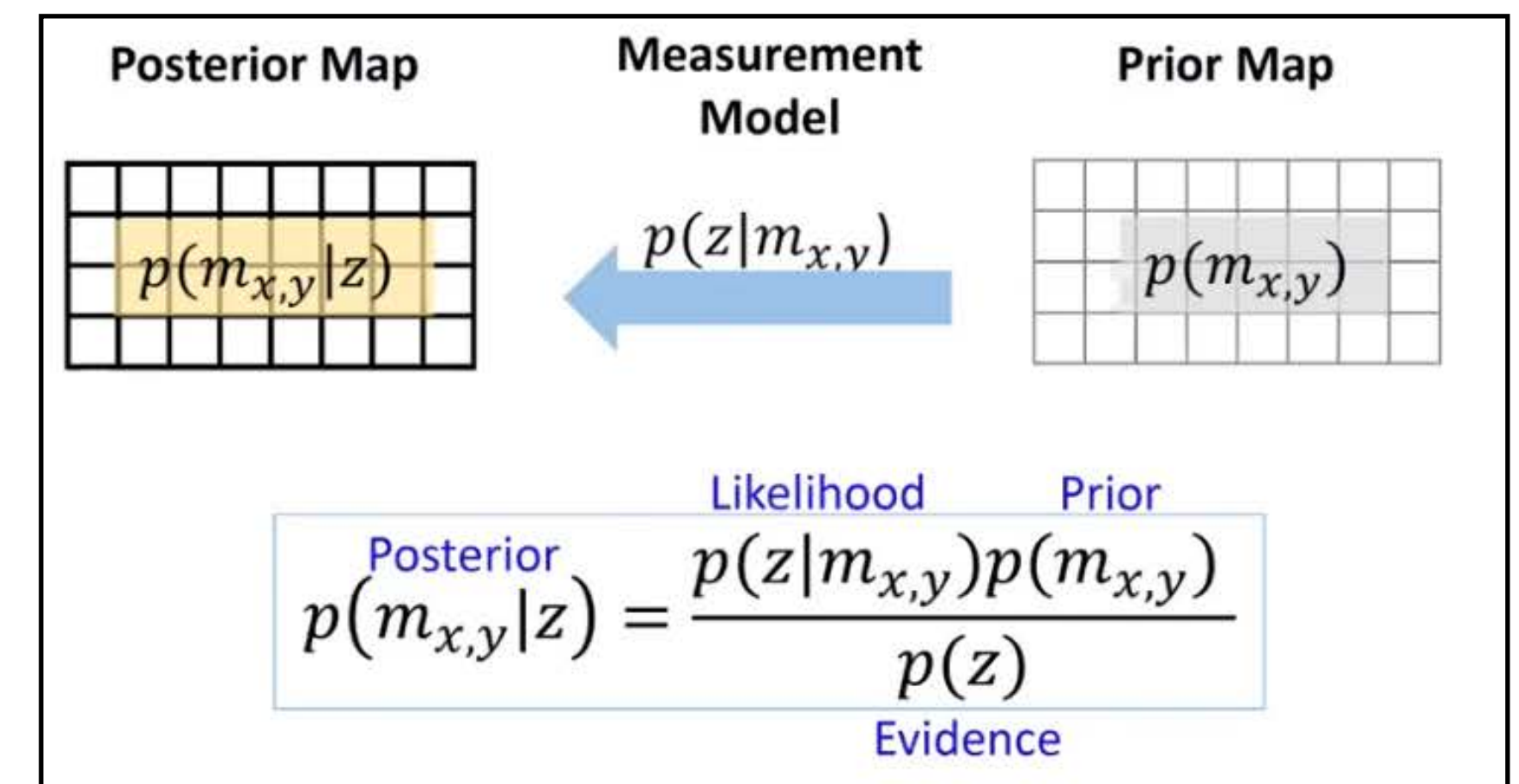
... how would you implement SLAM?

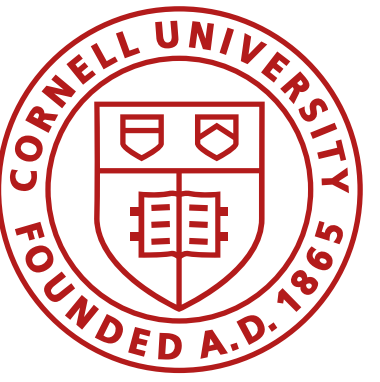
(where could your estimate of the map fit in?)



Given all we have learned...

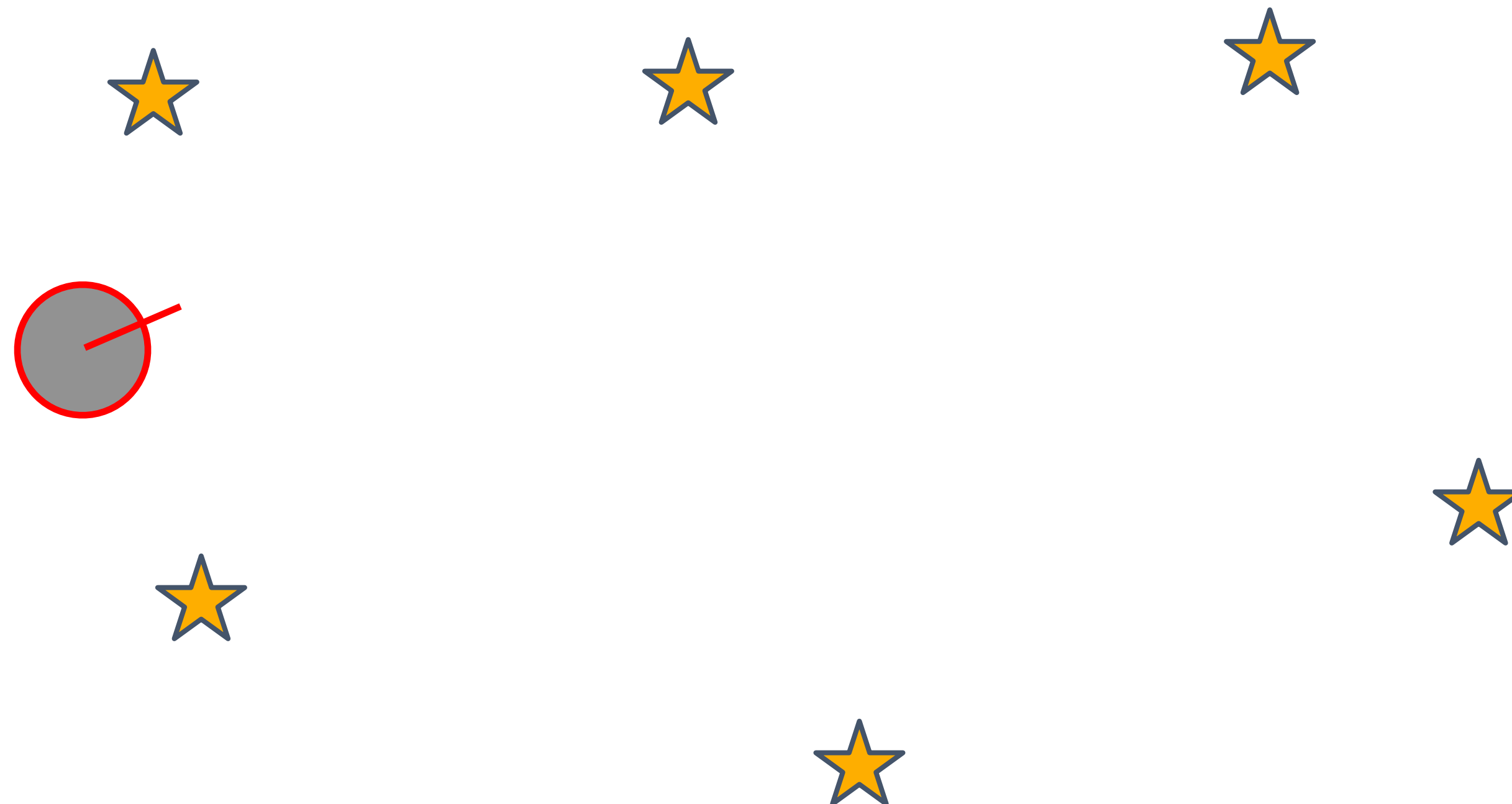
- Markov localization in a grid
 - Localization: estimate your cell pose within the map
 - Mapping: estimate if cells are occupied or not
 - Every grid cell is a random variable
 - SLAM: estimate pose *and* if cells are occupied or not
 - 100x100 grid cells (pretty small map)
 - Localization: (x, y, theta) = 100x100x100 states
 - Map: (x,y) = 10,000 states
 - SLAM 100x100x100x10,000 states
- Same issue for particle filters...
 - Balance parametric and non-parametric approaches

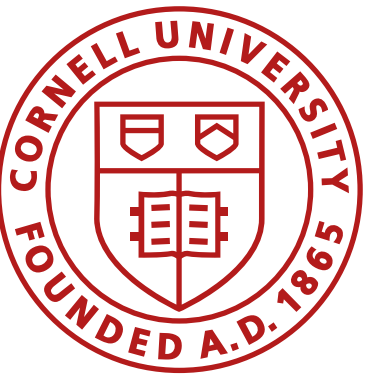




Why is SLAM hard?

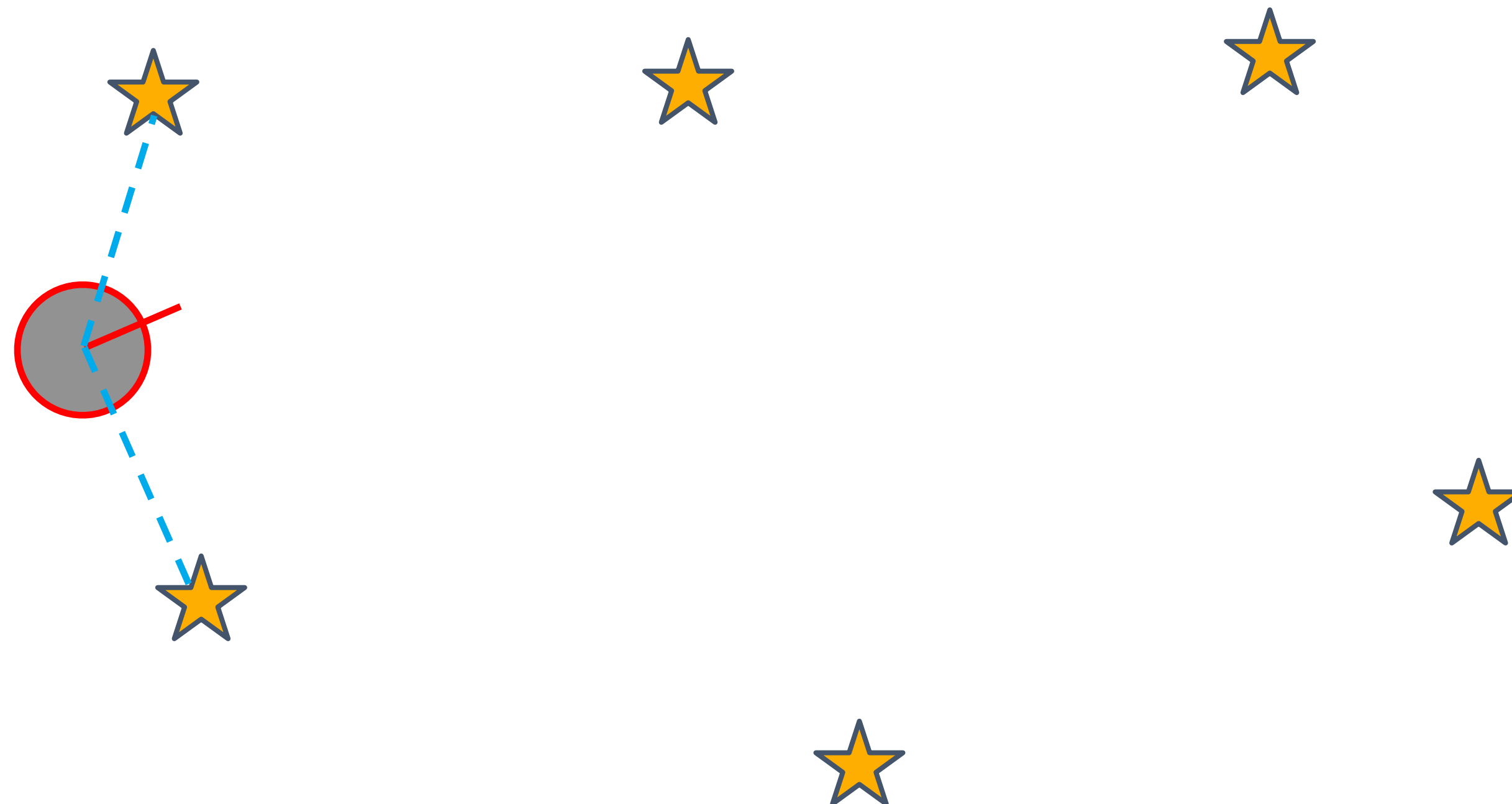
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

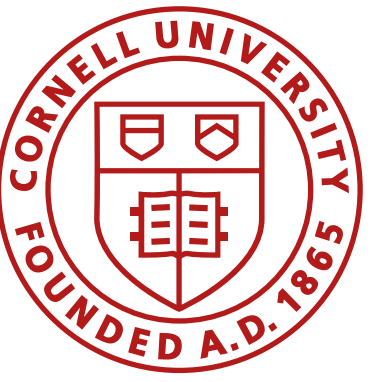




Why is SLAM hard?

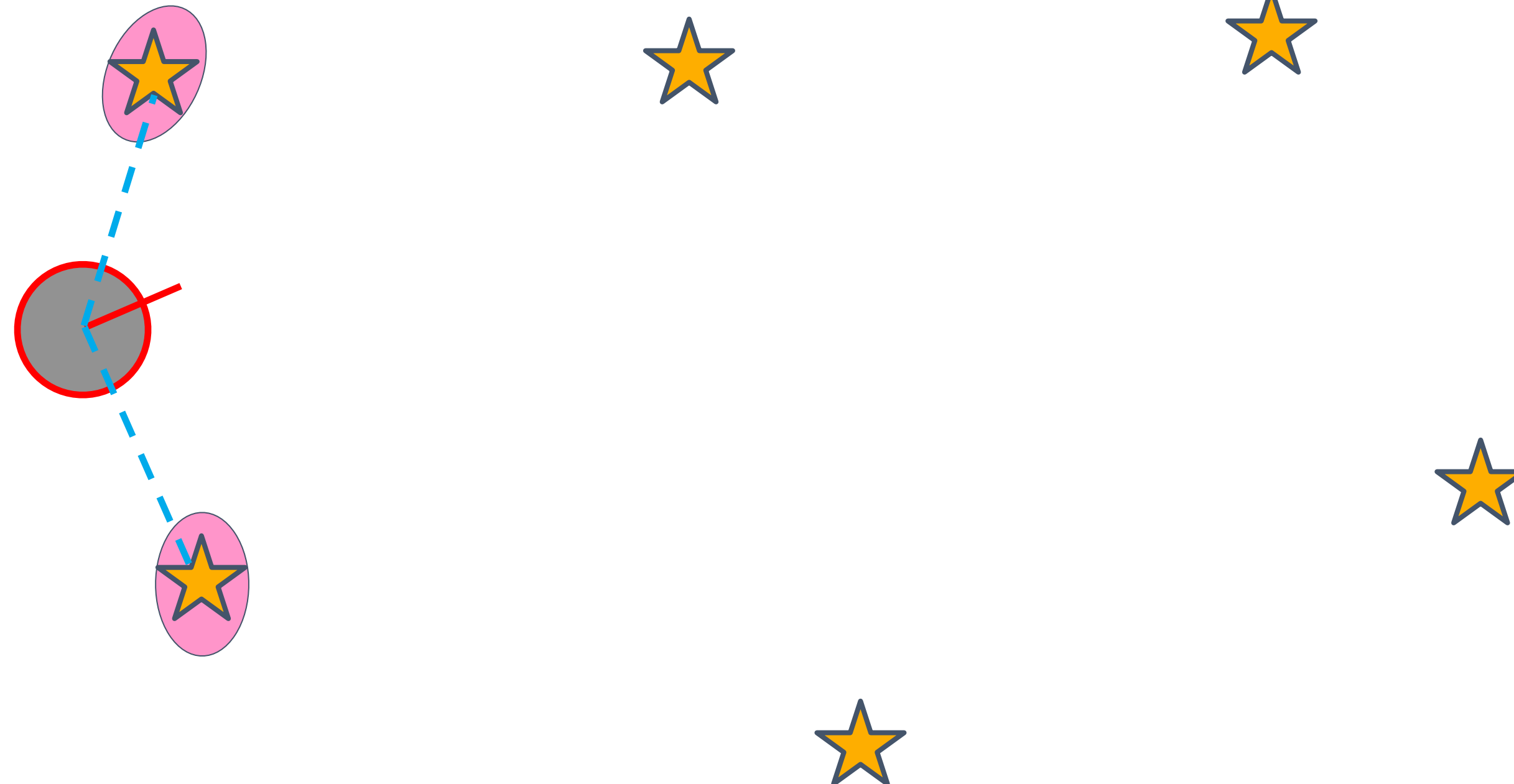
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

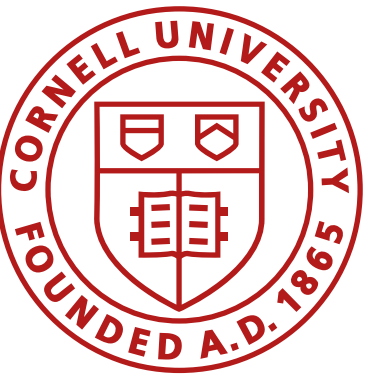




Why is SLAM hard?

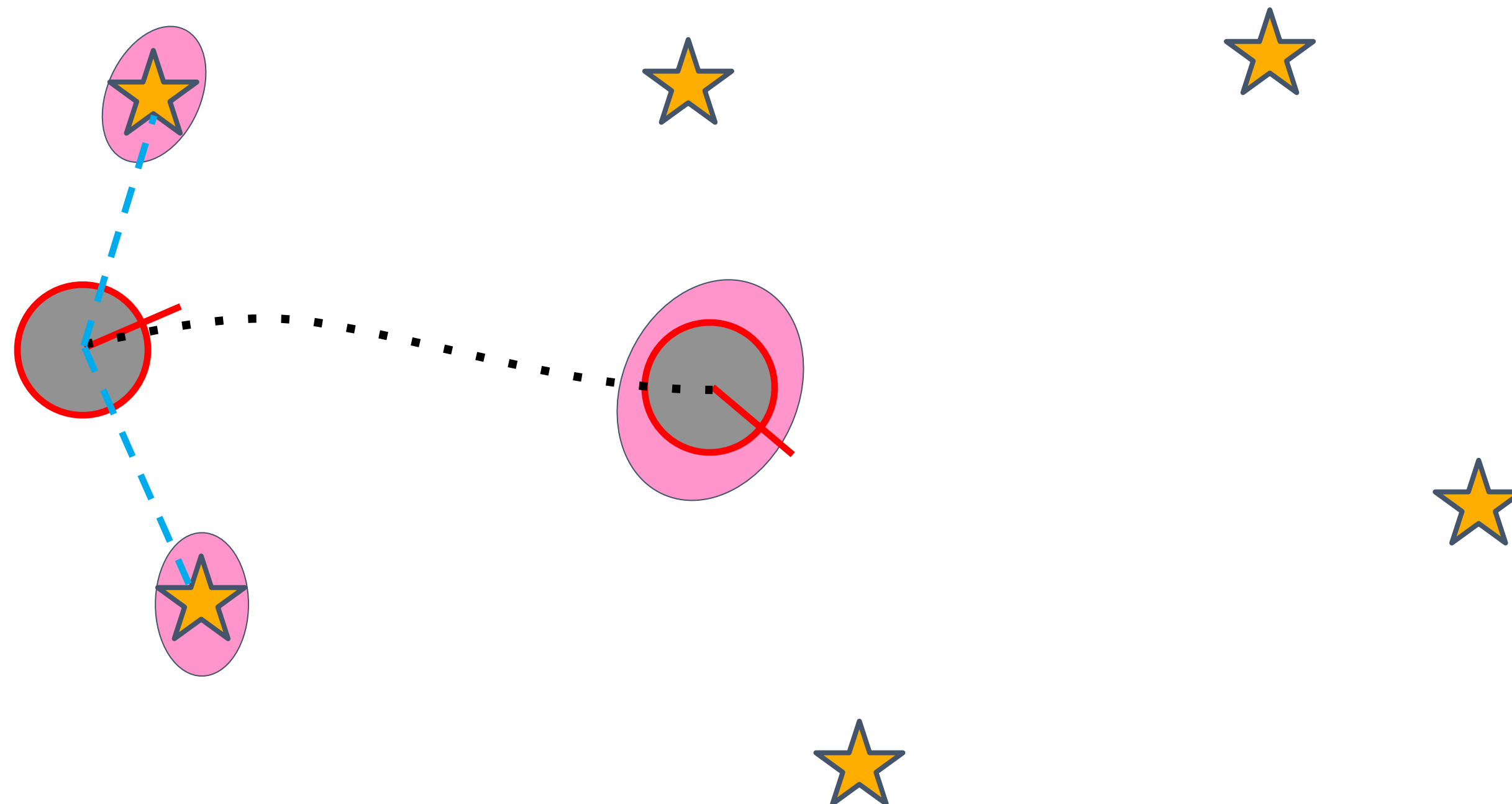
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

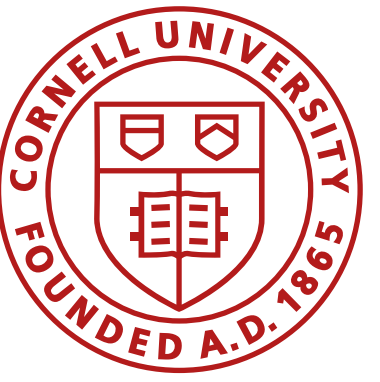




Why is SLAM hard?

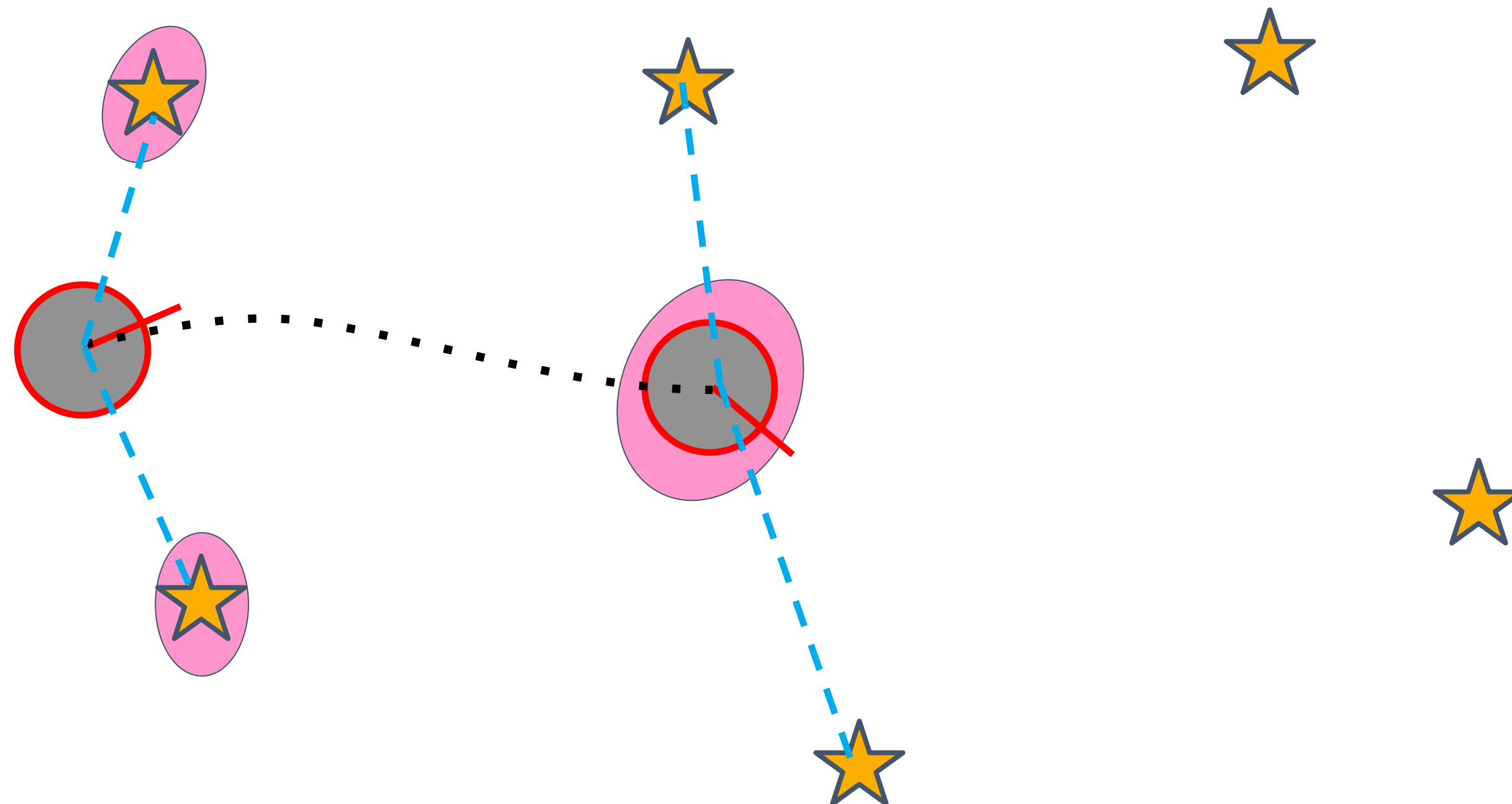
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

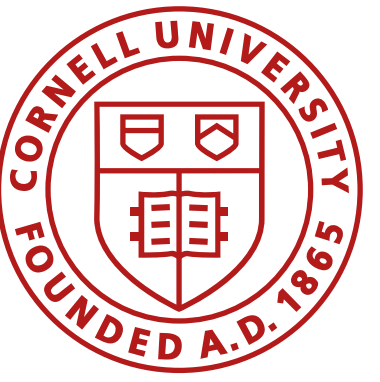




Why is SLAM hard?

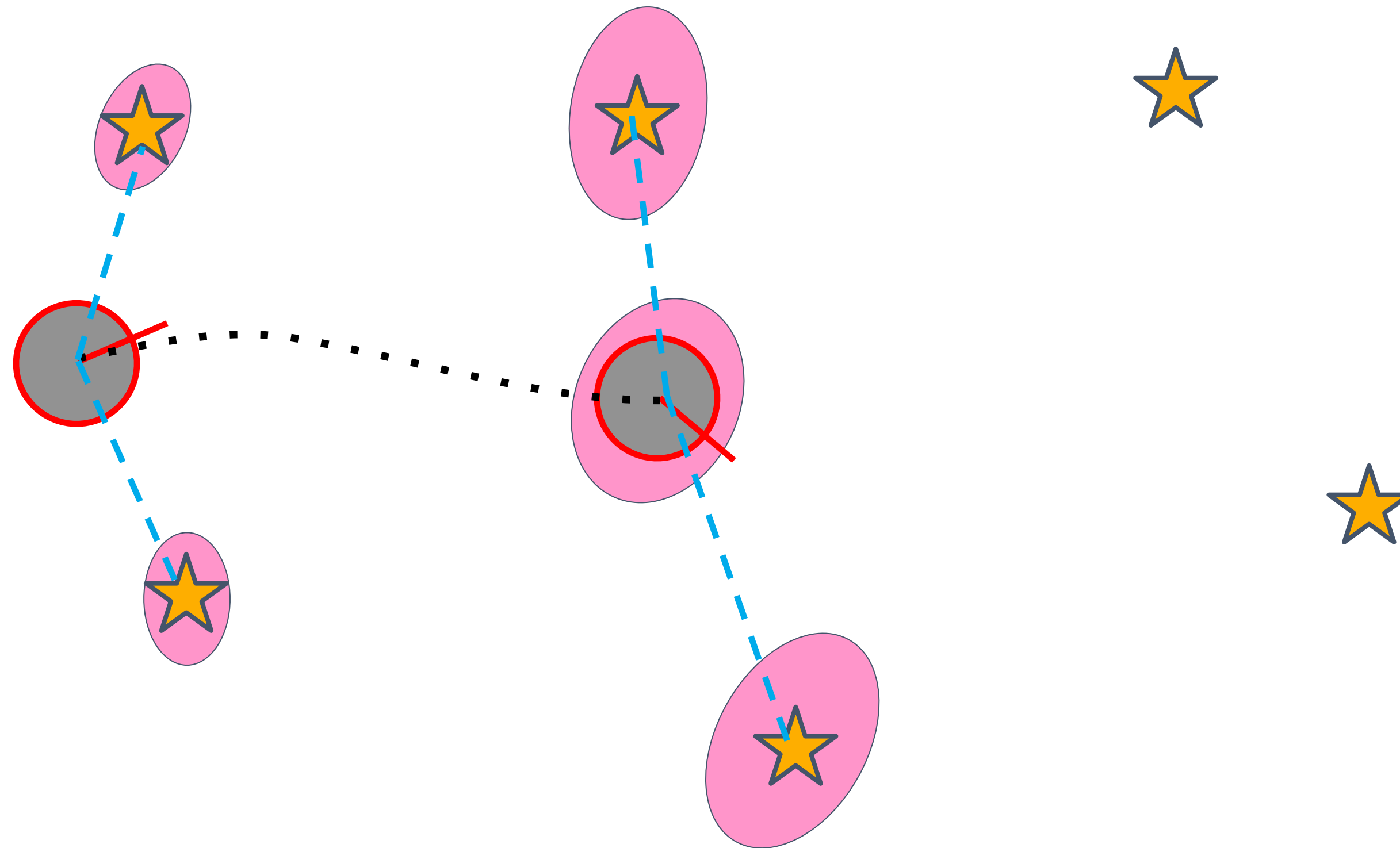
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

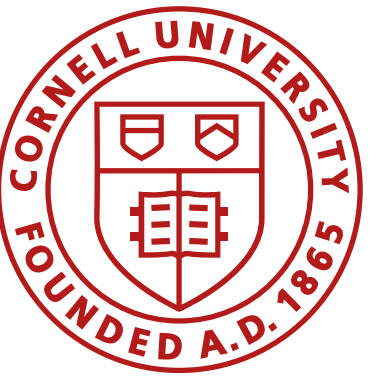




Why is SLAM hard?

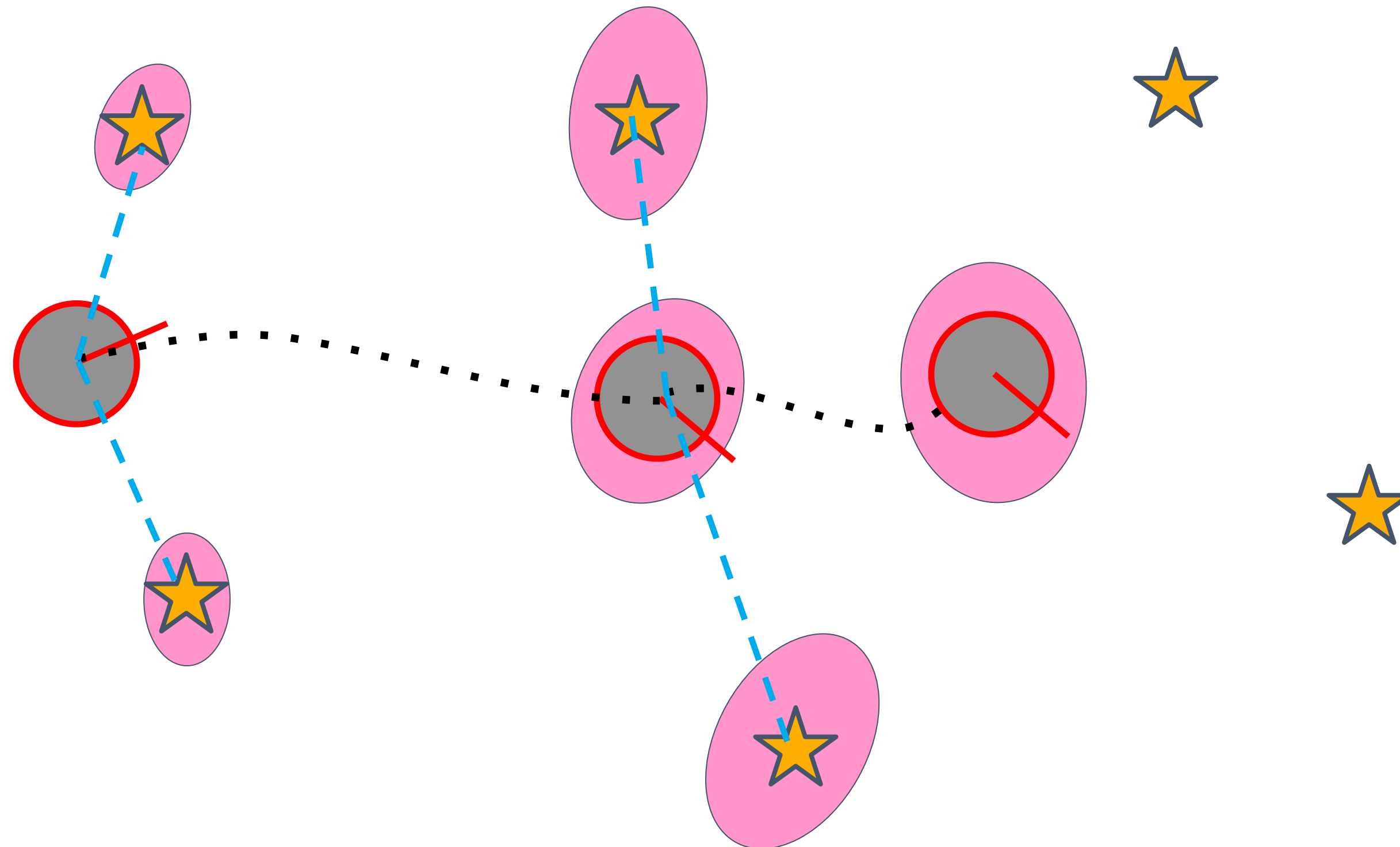
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

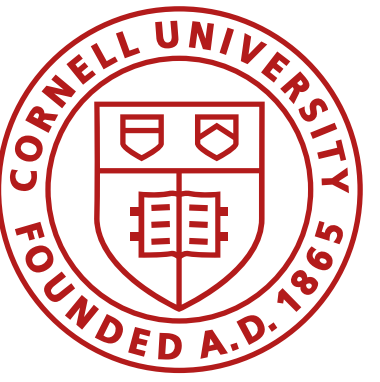




Why is SLAM hard?

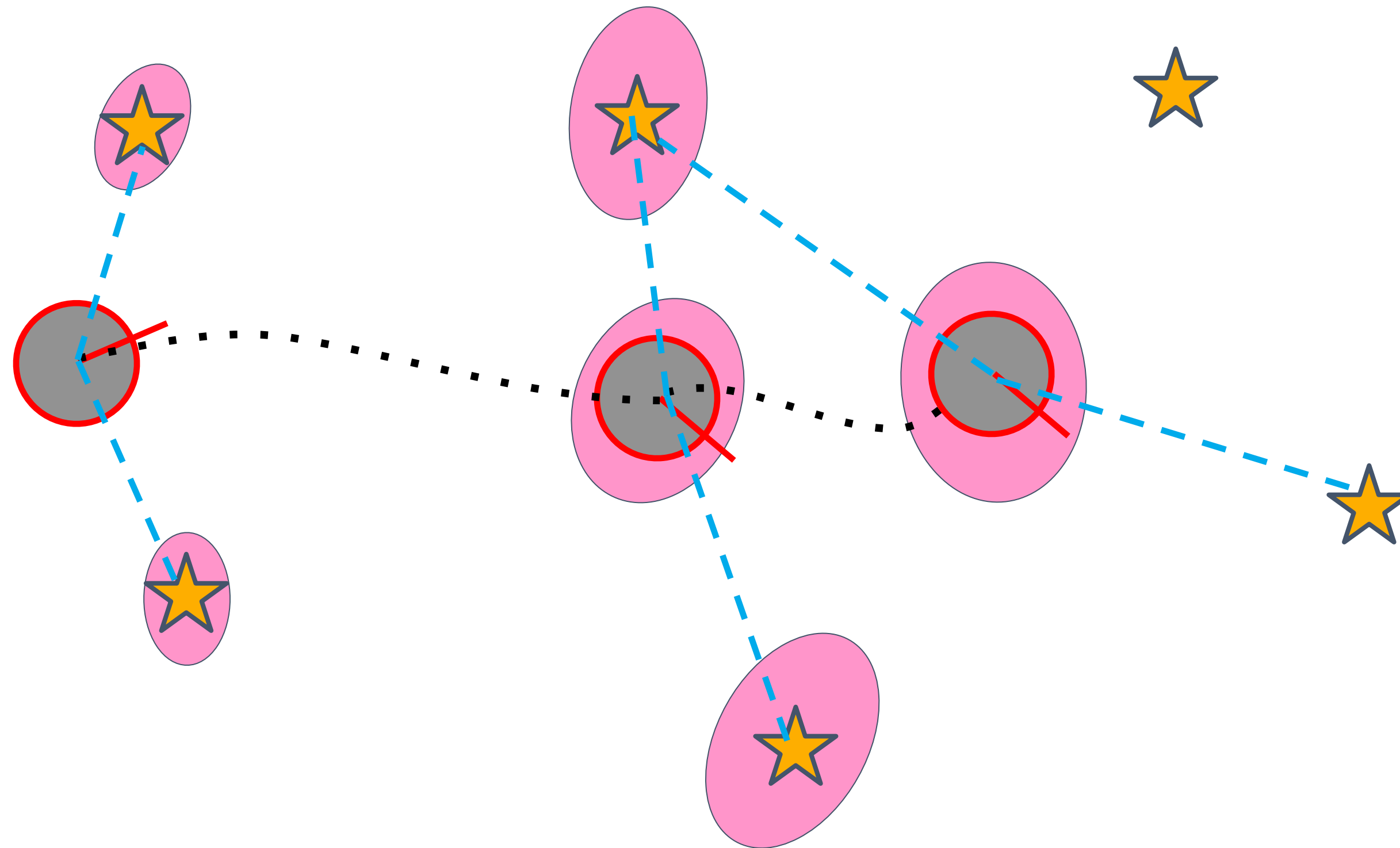
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated

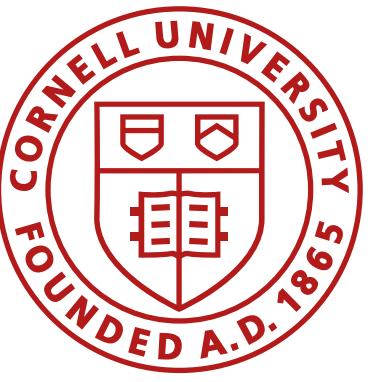




Why is SLAM hard?

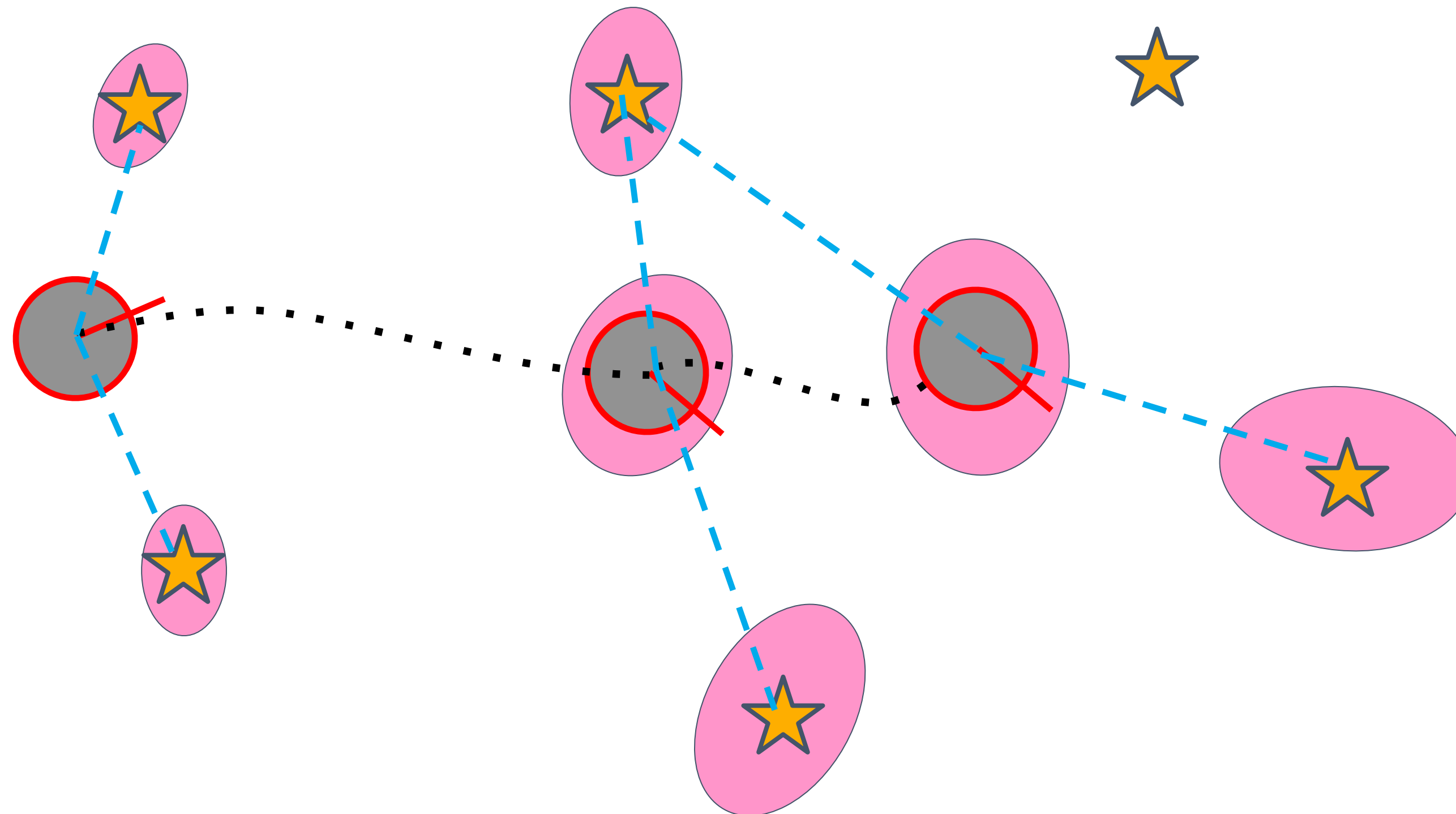
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated





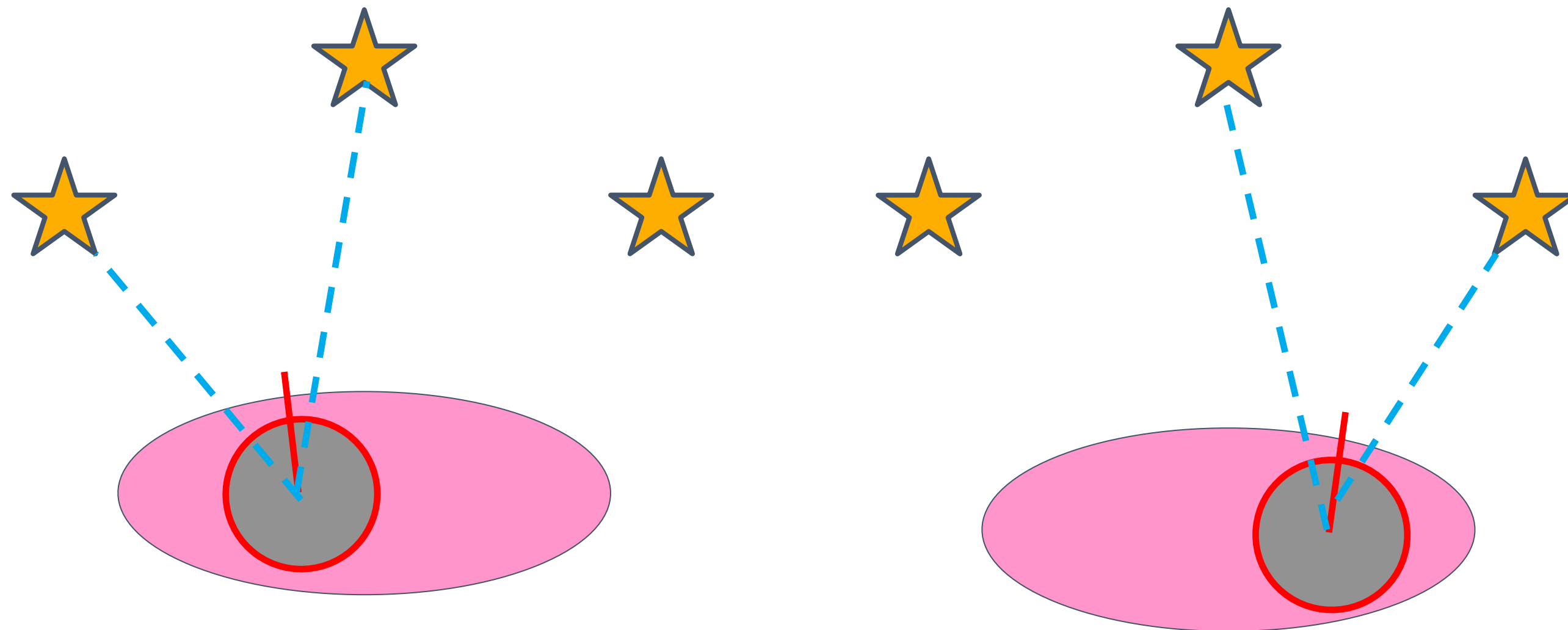
Why is SLAM hard?

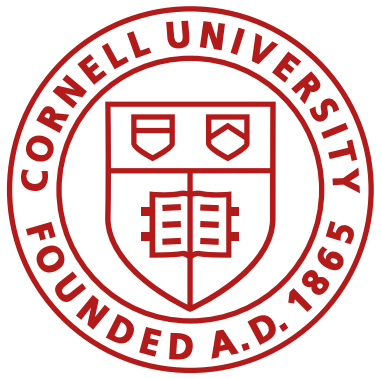
- Robot pose/path and map are both unknown (not independent)
- Map and pose estimates are correlated
- Good data association is key



Why is SLAM hard?

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



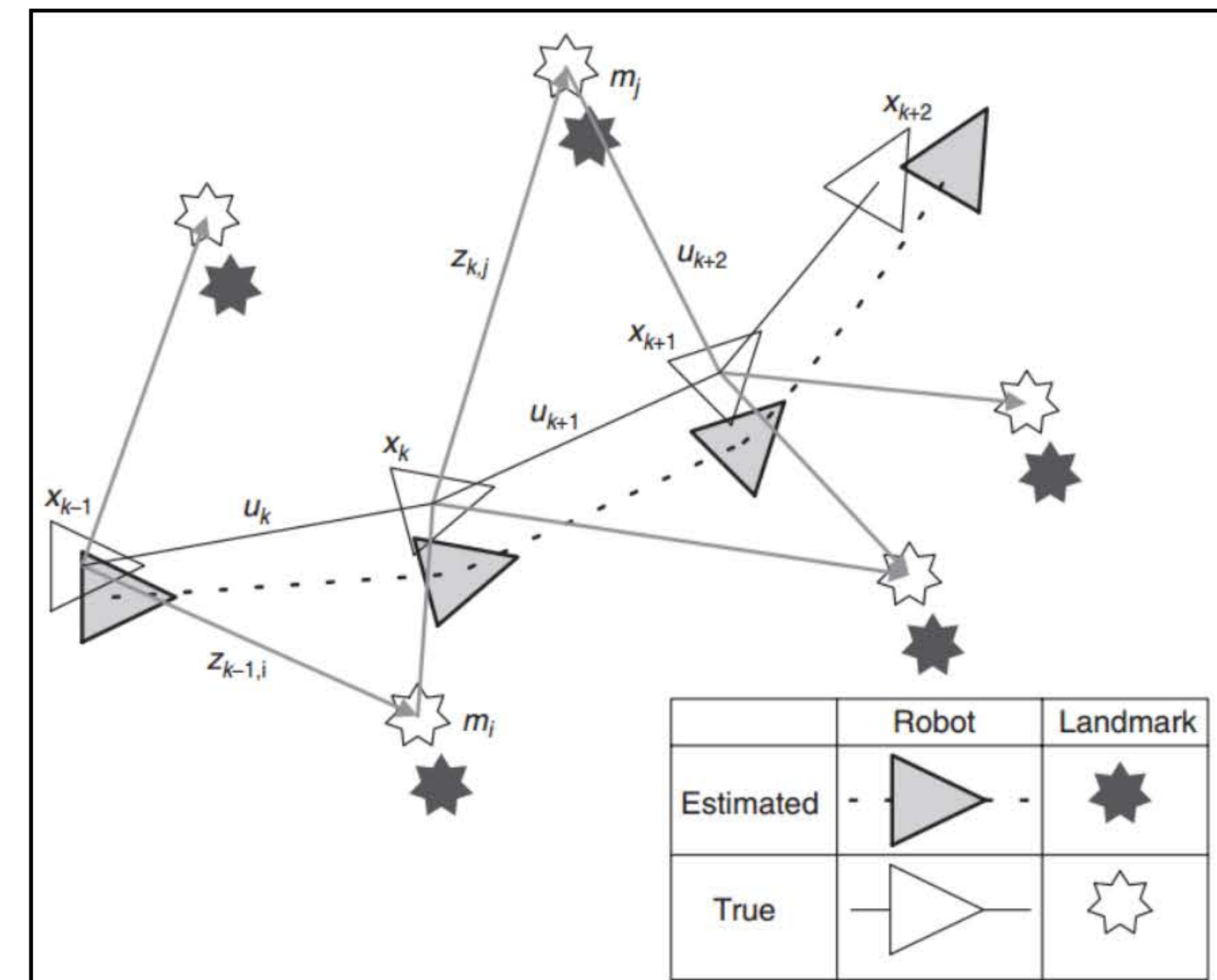


Related terms

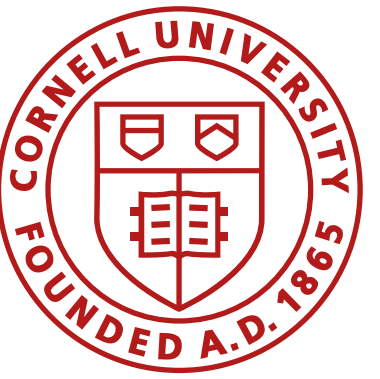
- State estimation
- Localization
- Mapping
- **SLAM**
- Navigation
- Motion planning

- Given
 - Control inputs
 $U_{0:k} = \{u_1, u_2, \dots, u_k\}$
 - Relative observations
 $Z = \{z_1, z_2, \dots, z_n\}$
- Compute
 - Map of the environment
 $m = \{m_1, m_2, \dots, m_n\}$
 - Robot path (seq. of poses)
 $X_{0:k} = \{x_0, x_1, \dots, x_k\}$

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



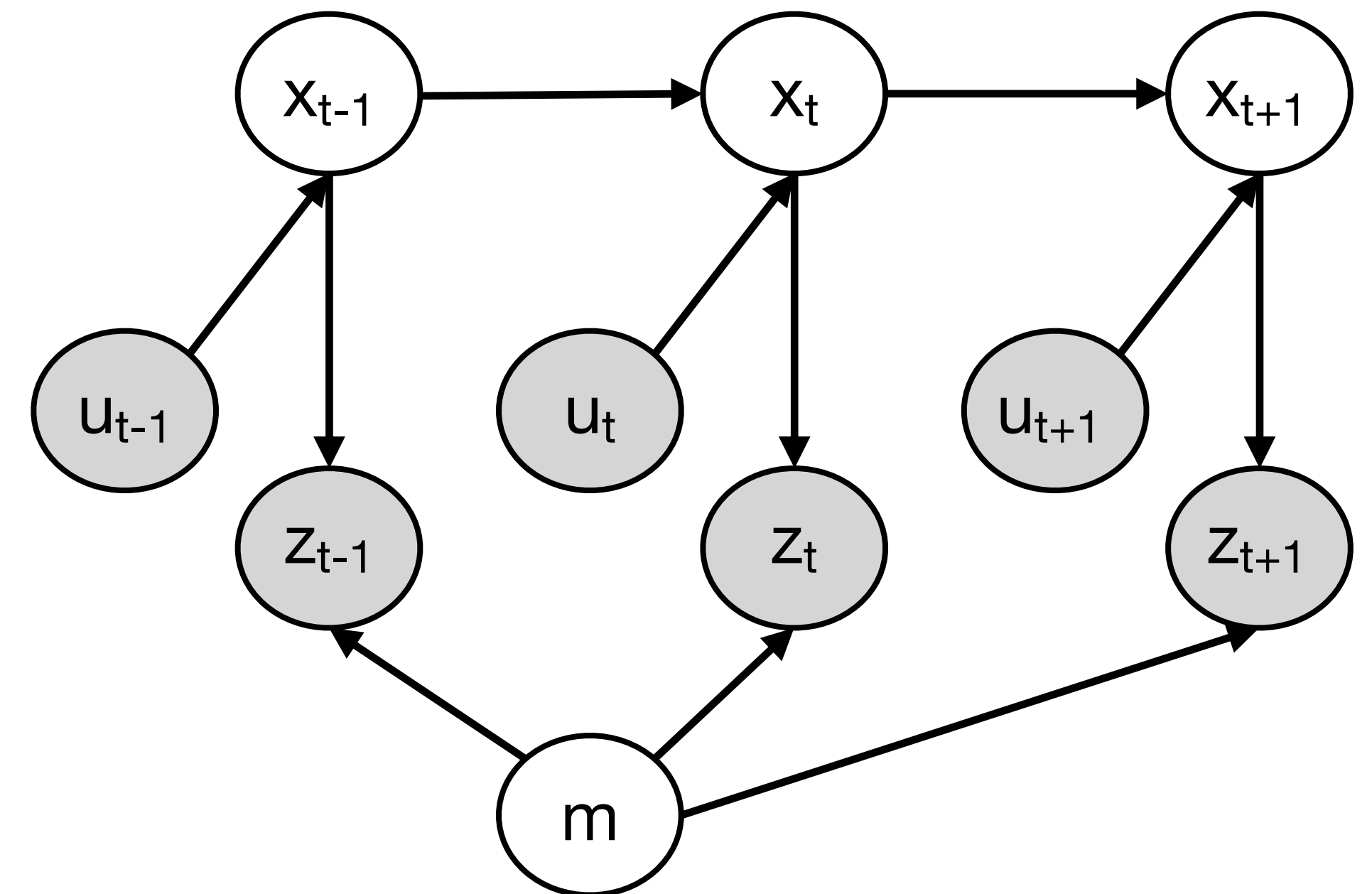
Landmarks are considered motionless



Simultaneous Localization and Mapping

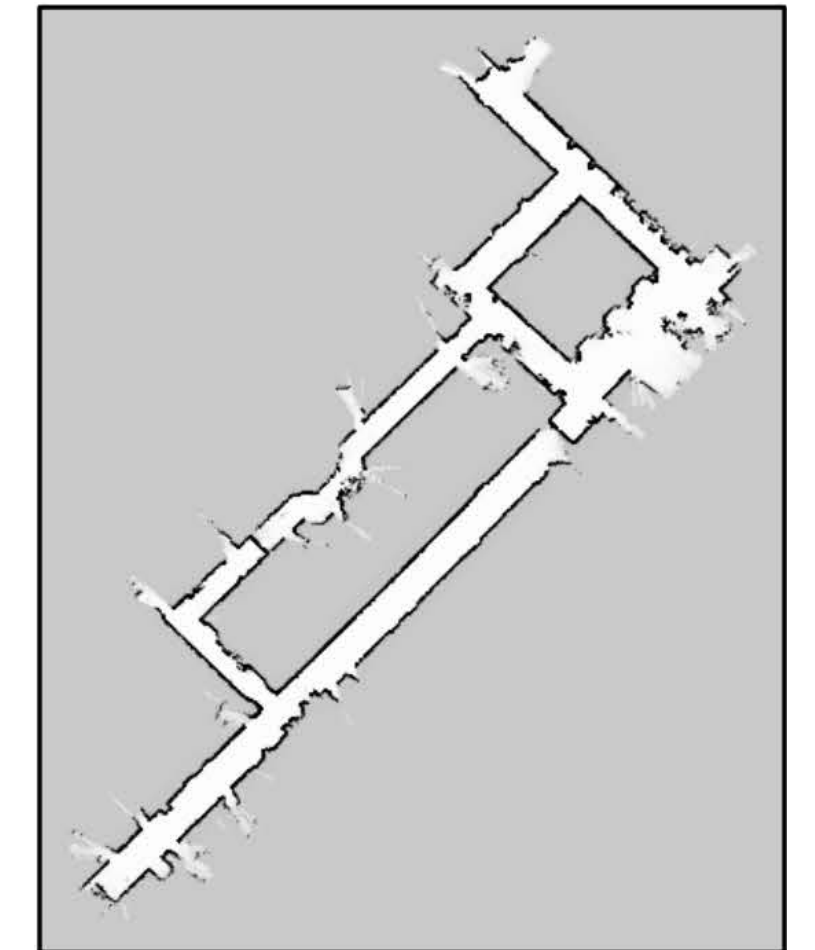
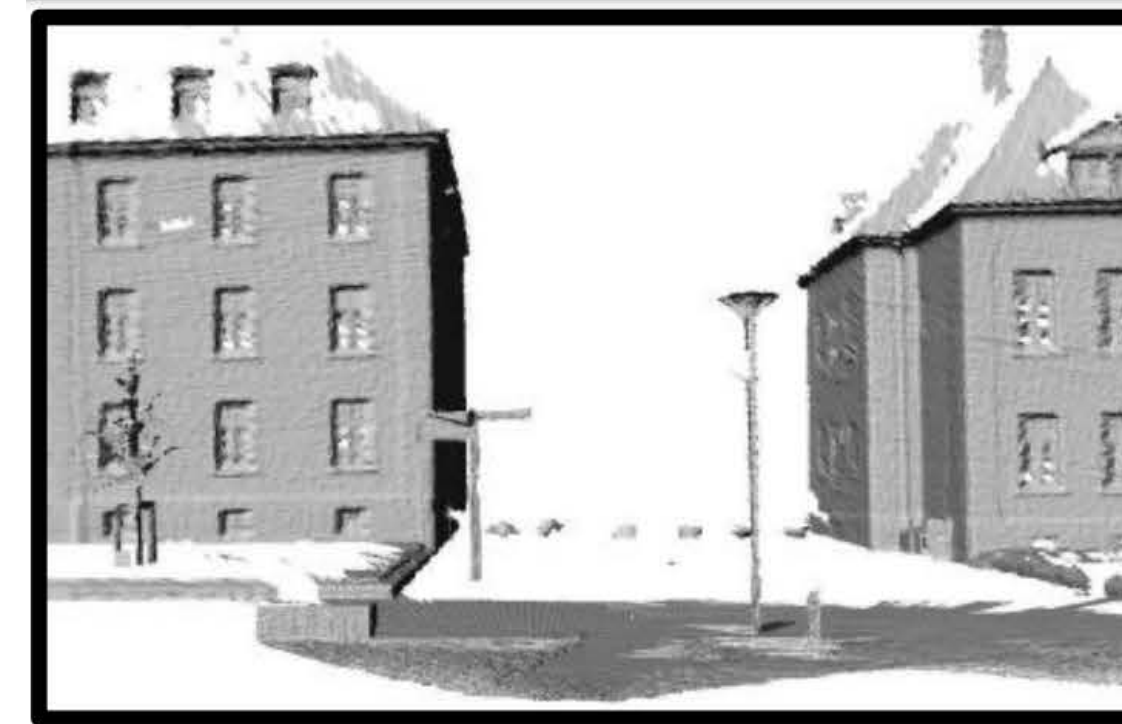
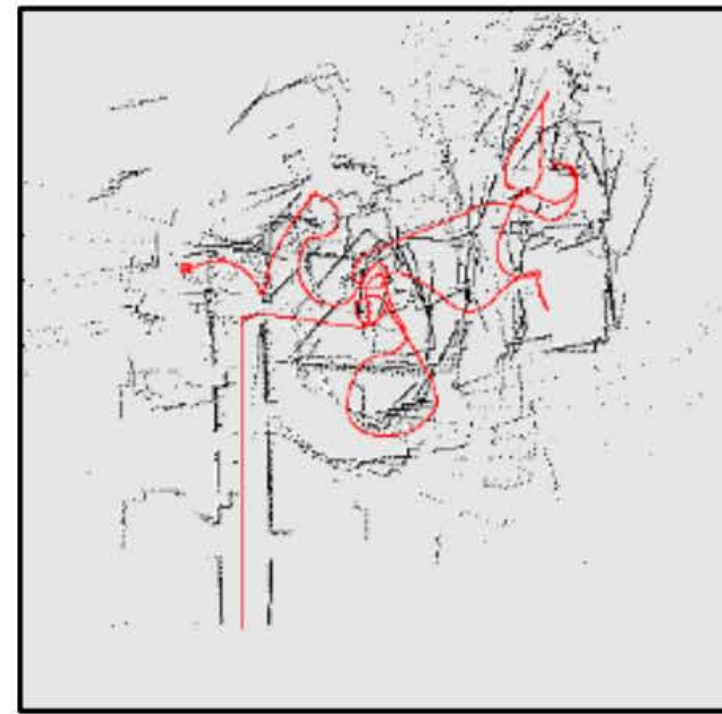
Graphical model

- Nodes are random variables
- Directed edges are variable dependencies
- Gray nodes: observed or directly measured variables
- White nodes: inferred latent variables



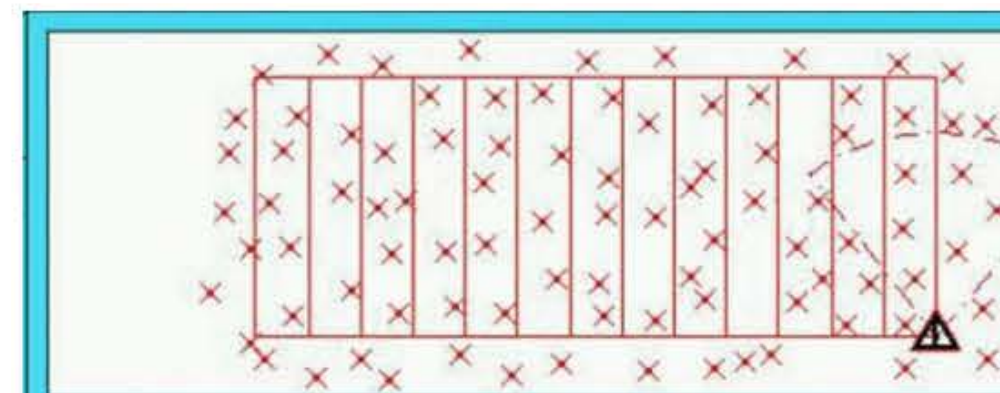
SLAM Representations

- Grid maps or scans

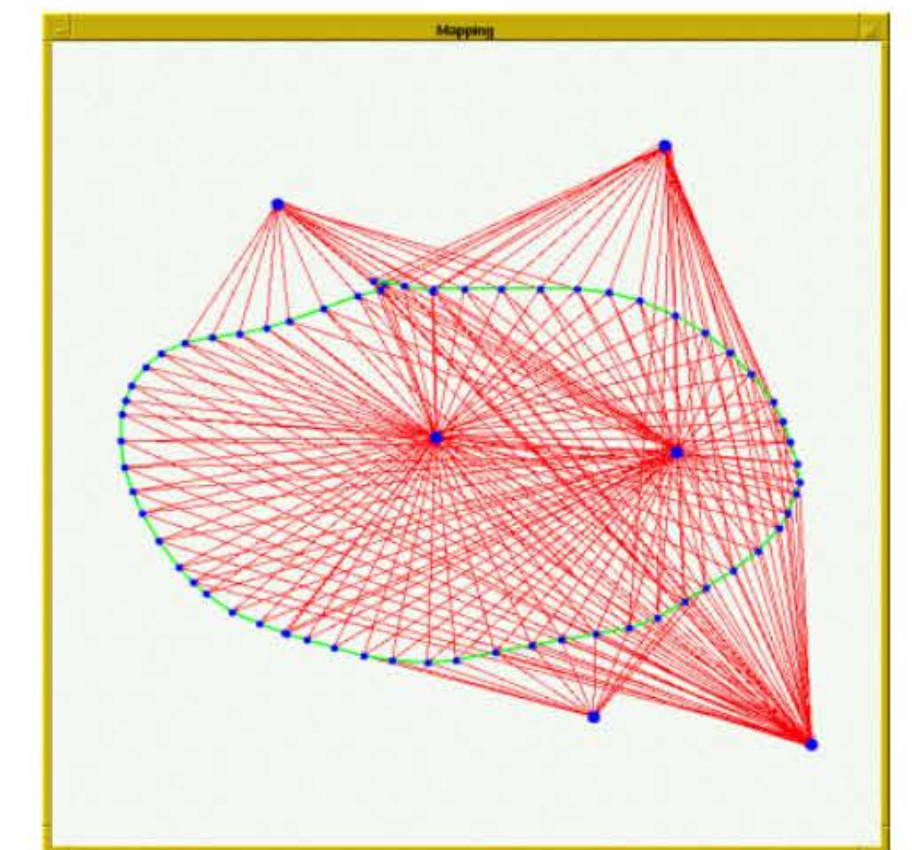
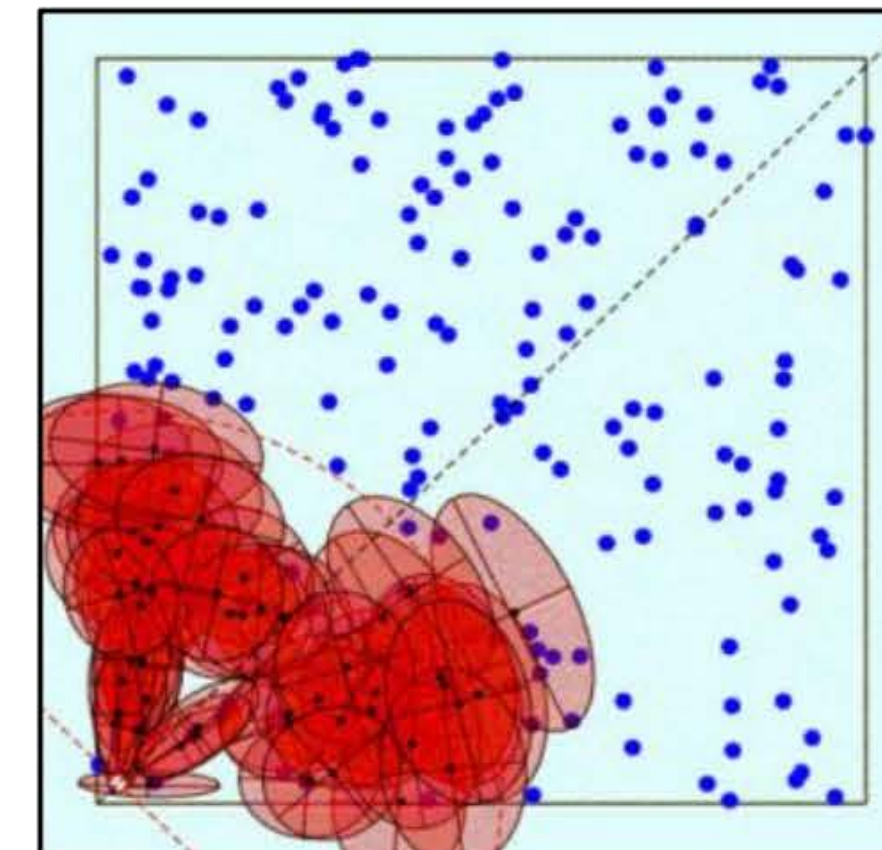


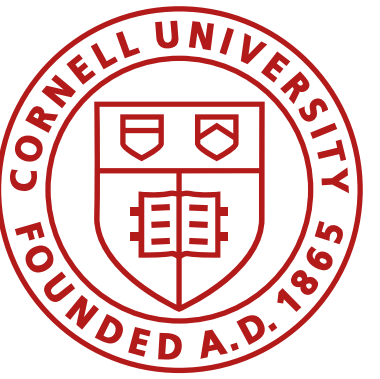
[Lu & Milios, 97; Gutmann, 98; Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;...]

- Landmark-based



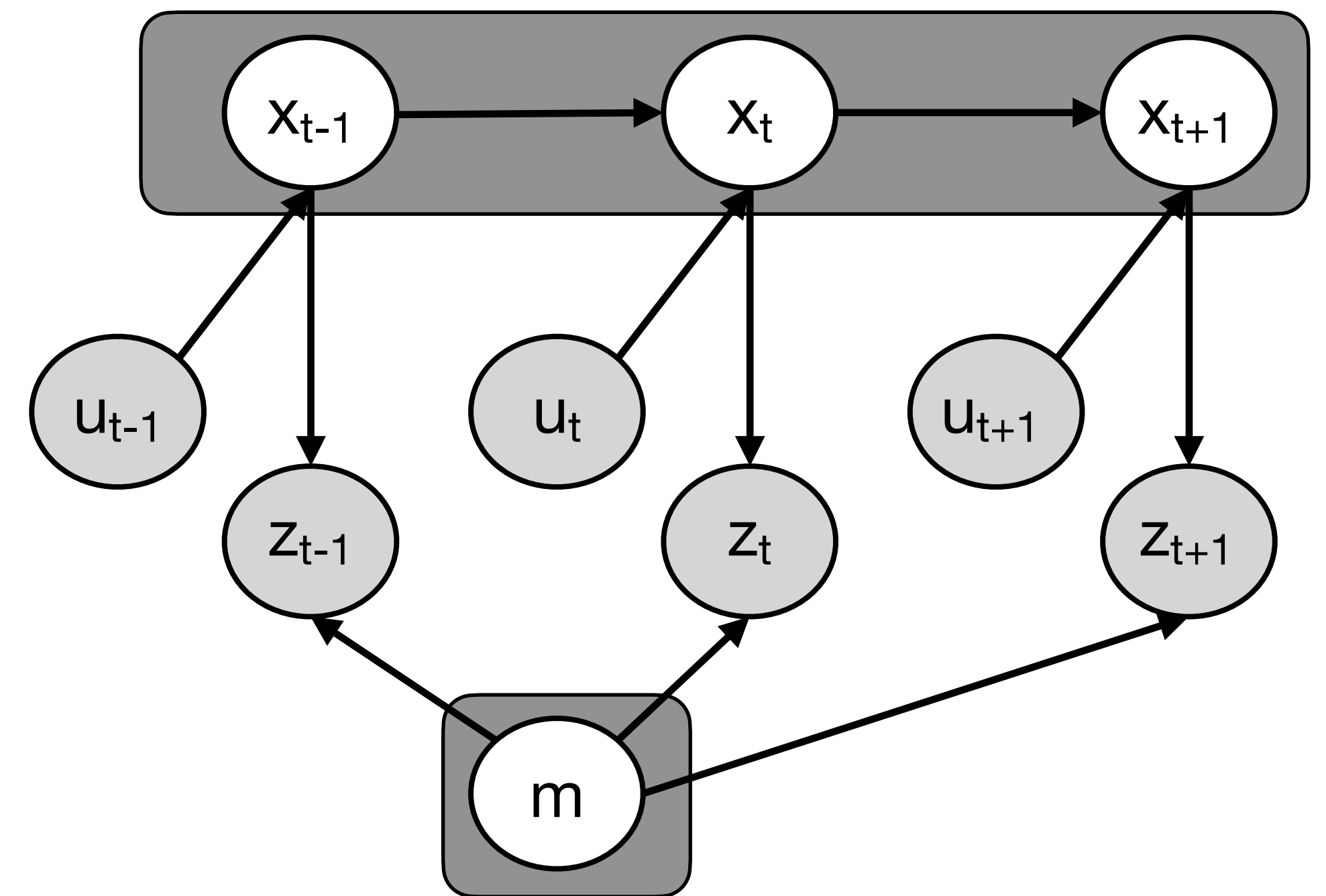
[Leonard et al., 98; Castelanos et al., 99; Dissanayake et al., 2001; Montemerlo et al., 2002;...]



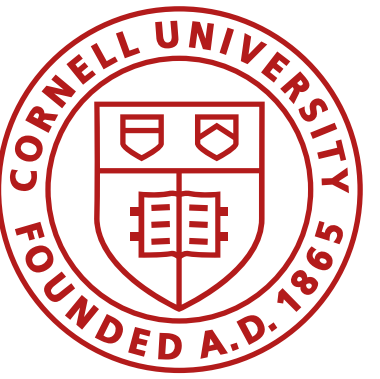


Simultaneous Localization and Mapping

- 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 over the whole path of the robot and the map

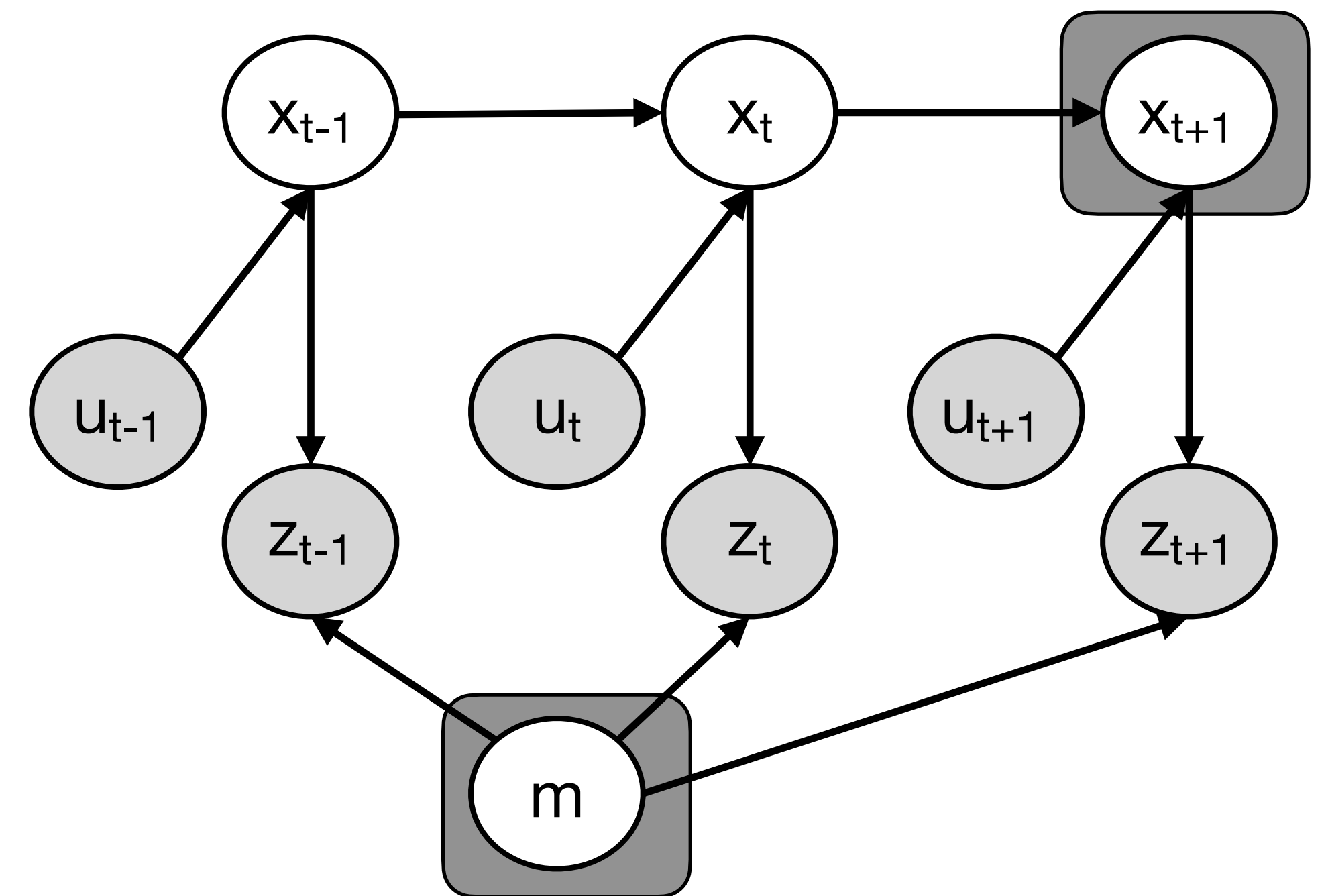


$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}, x_0)$$

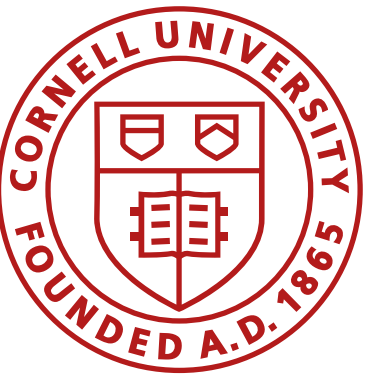


Simultaneous Localization and Mapping

- 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



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

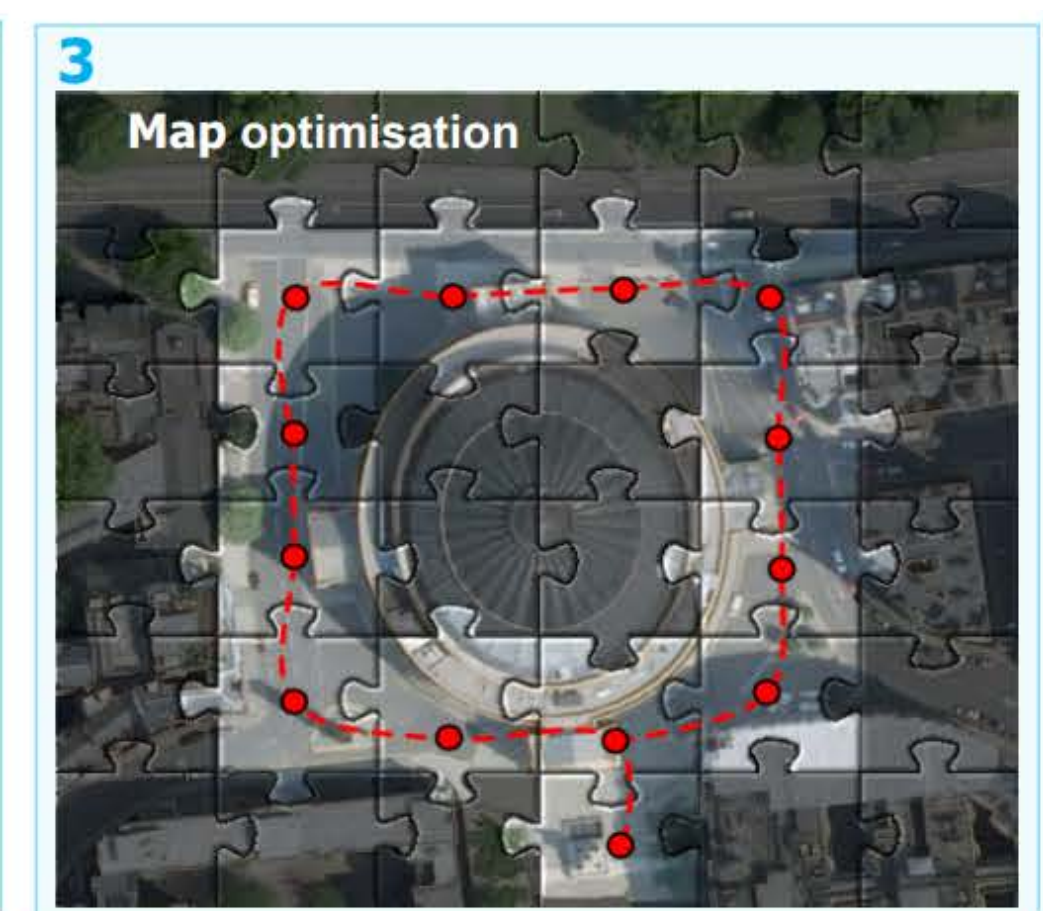
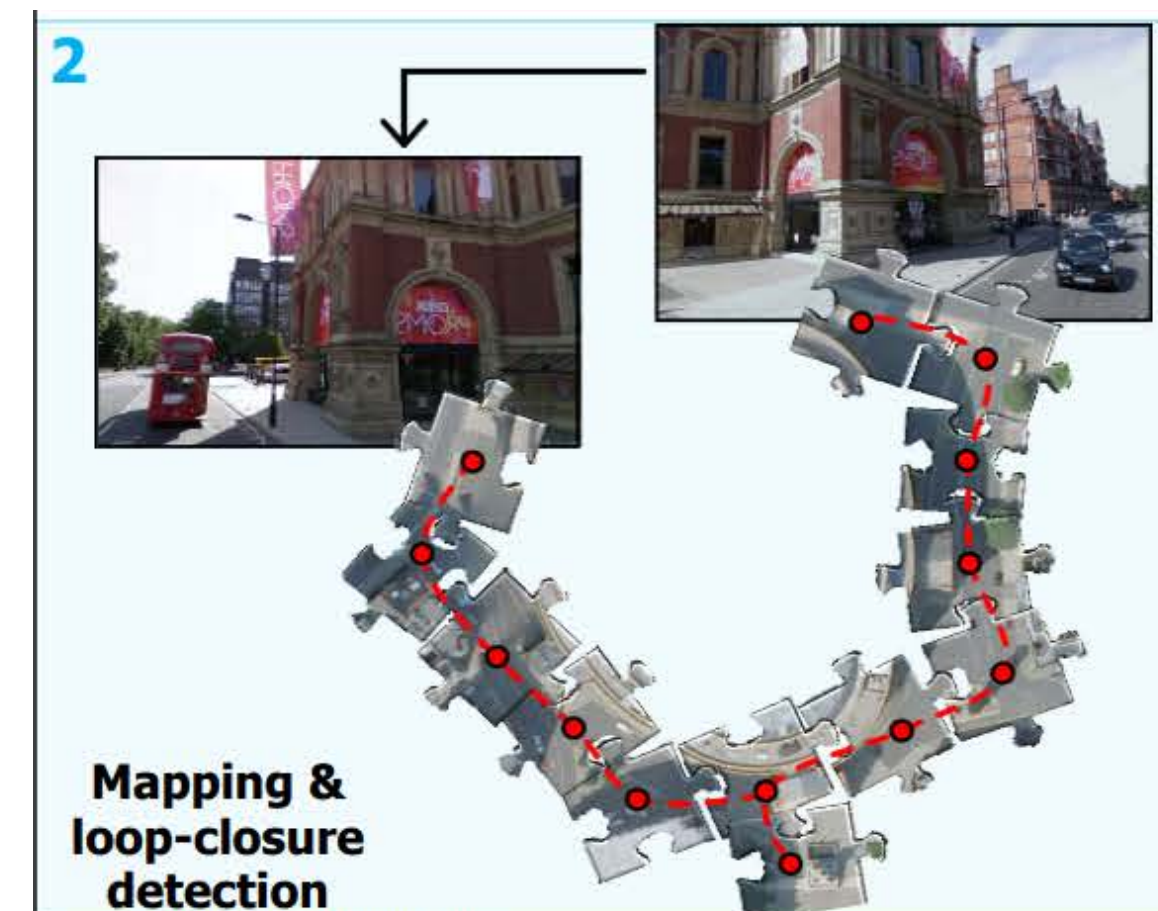
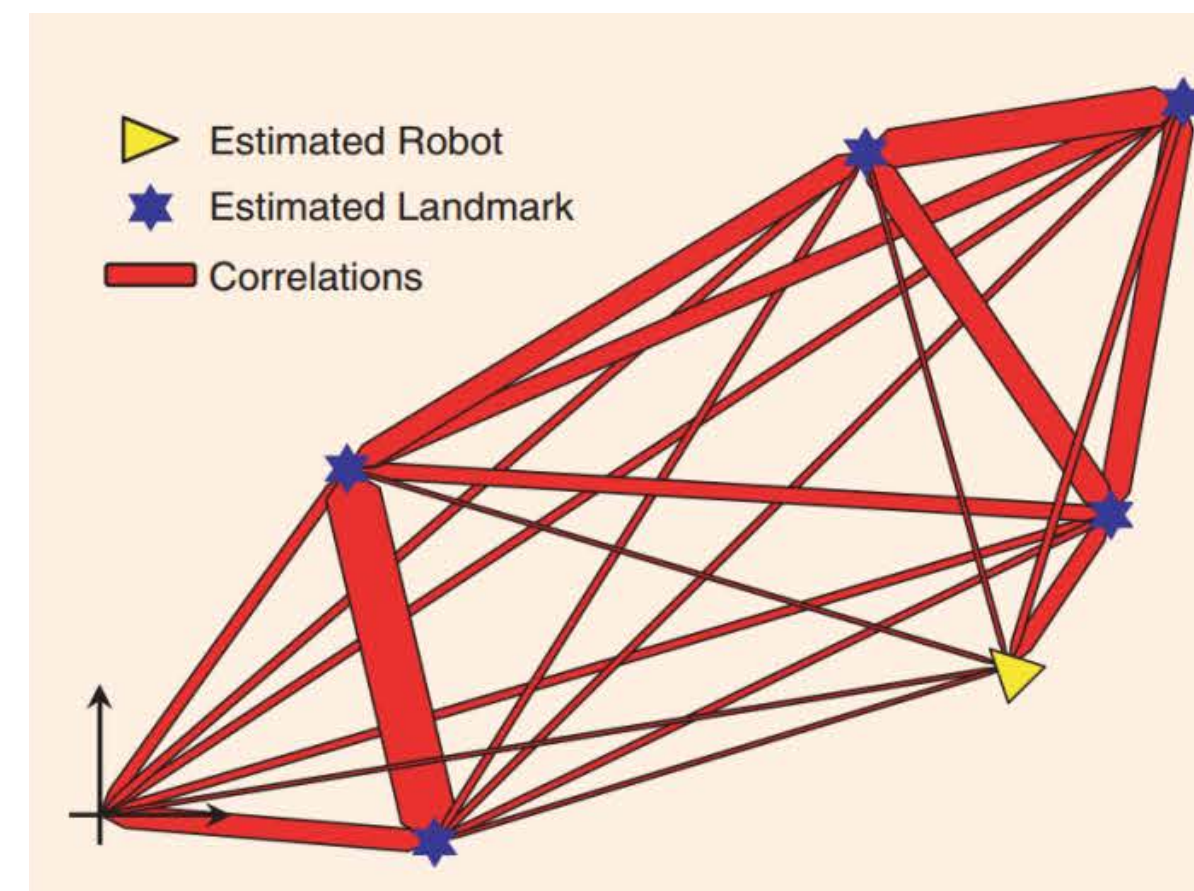
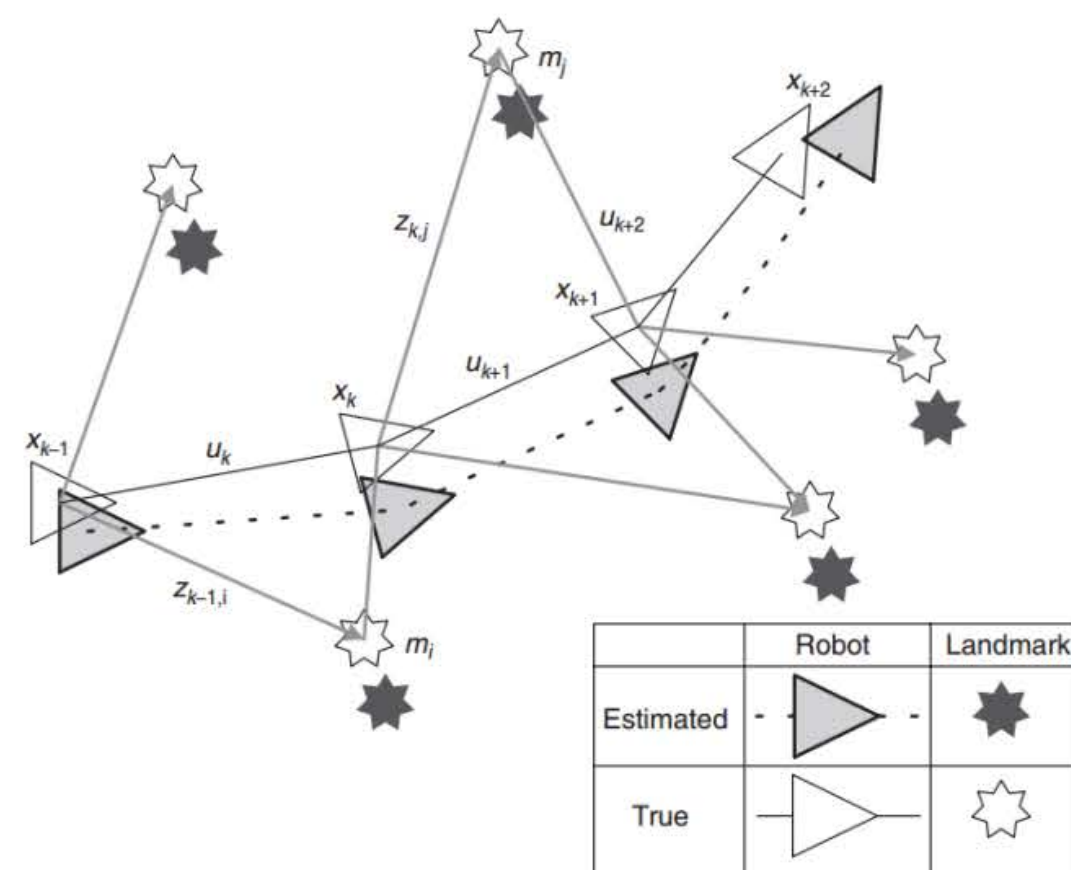


Simultaneous Localization and Mapping

- Prediction (prediction step):
 - $p(x_t, m \mid z_{0:t}, u_{1:t}, x_0) = \sum_{t-1} P(x_t \mid x_{t-1}, u_{1:t}) P(x_{t-1}, m \mid Z_{0:t-1}, U_{1:t}, x_0)$
- Correction (update step):
 - $p(x_t, m \mid z_{0:t-1}, u_{0:t}, x_0) = \eta P(z_t \mid x_t, m) P(x_t, m \mid Z_{0:t}, U_{1:t}, x_0)$
- We can solve the localization problem with the assumption that we know the map
 - $P(x_t \mid Z_{0:t}, U_{0:t}, m)$
- We can solve the mapping problem with the assumption that we know the location
 - $P(m \mid X_{0:t}, Z_{0:t}, U_{0:t})$

Interesting Features

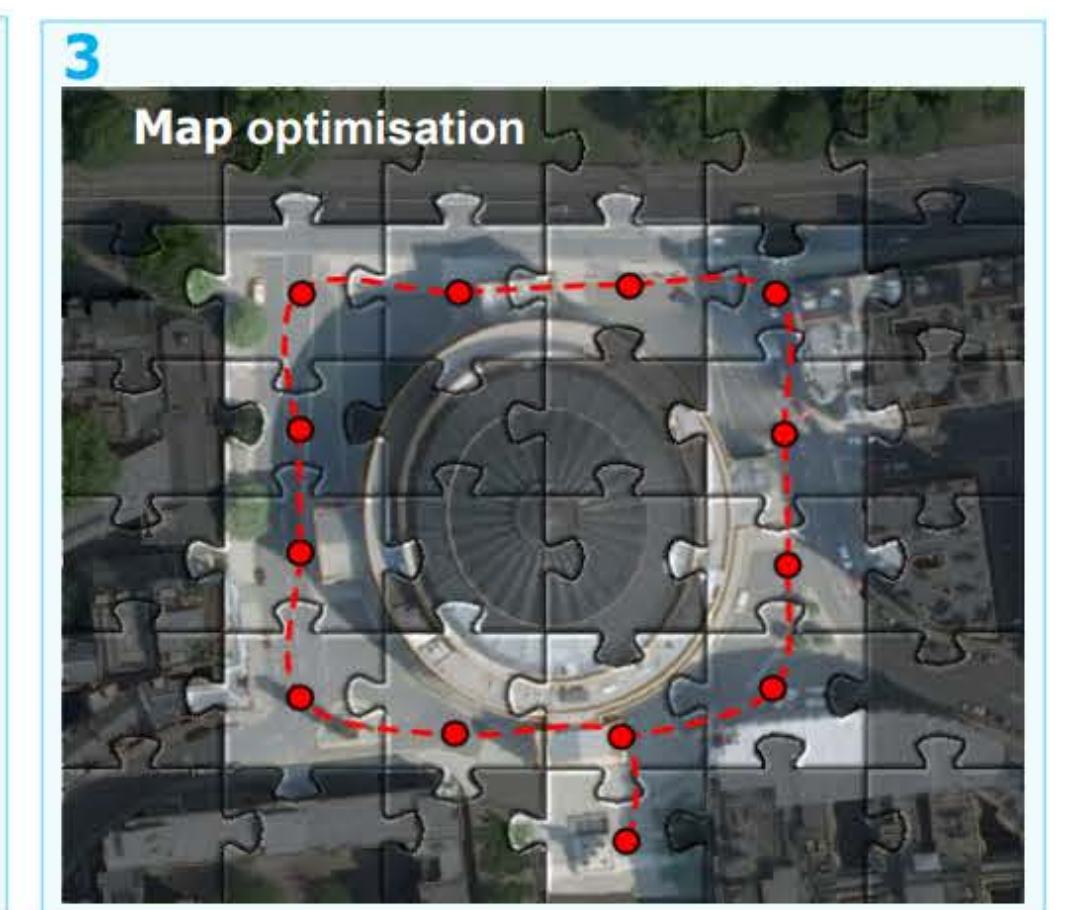
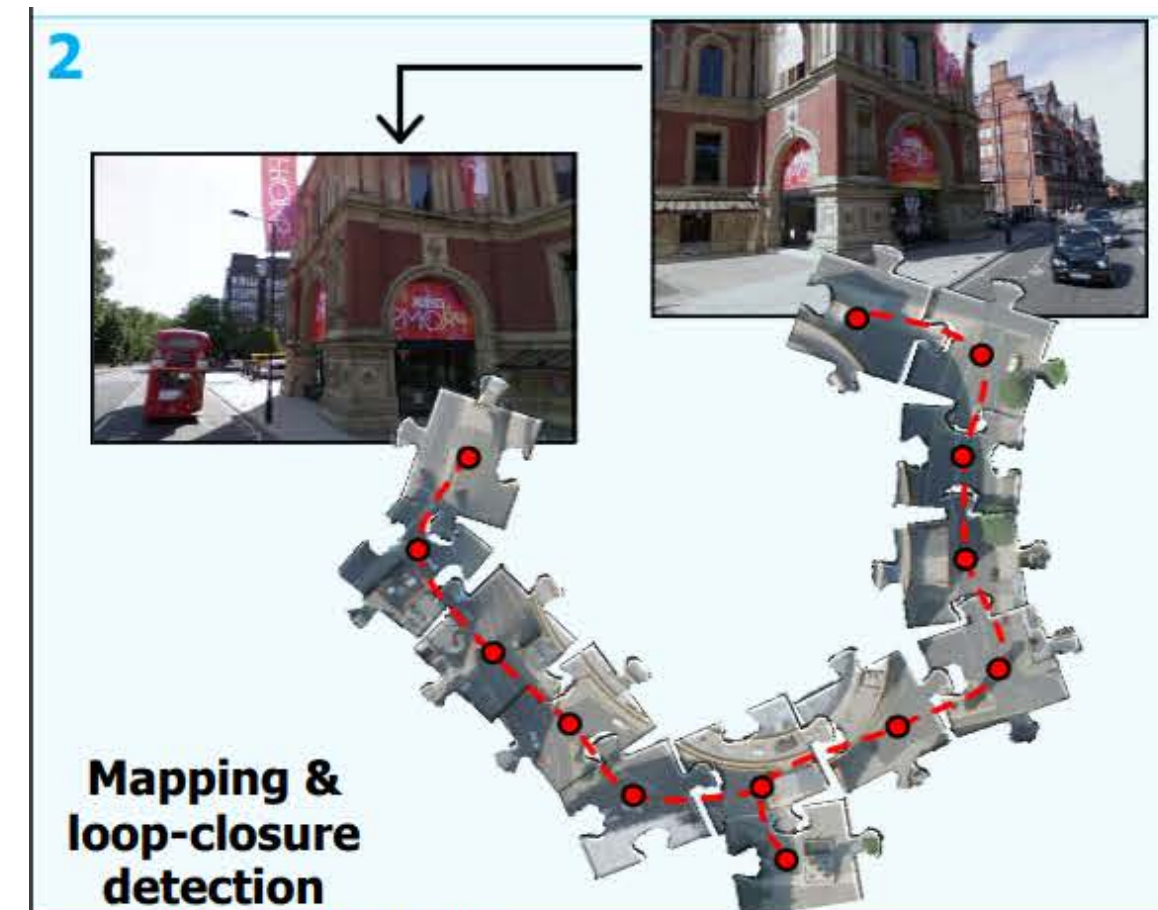
- Robot observations of the **relative landmark locations can be considered nearly independent**, because the relative landmark locations are independent from the robot's coordinate frame
- Robot observations of the **absolute landmark locations are less certain**, because the absolute landmark location 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

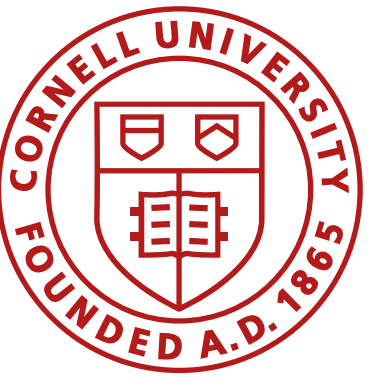


Simultaneous Localization and Mapping

Why is it hard?

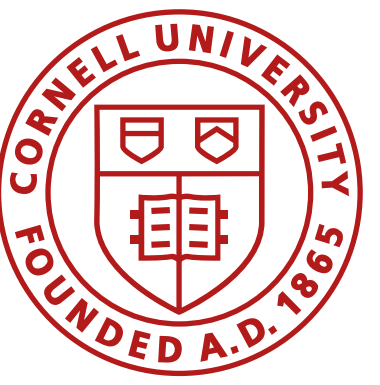
- Map size
 - The larger the environment relative to the robot's perceptual range, the more difficult it is to acquire the map
- Perceptual ambiguity
 - The more different places look alike, the more difficult it is to establish correspondence between different locations traversed at different points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



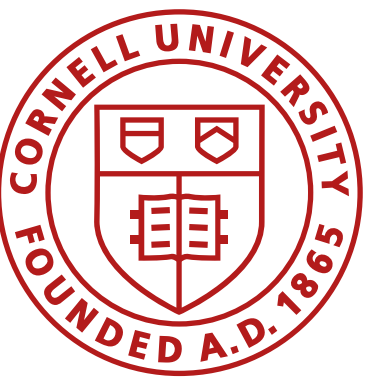


SLAM solutions

- 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 | Probability distribution over landmarks and the most recent pose (online SLAM)
 - Fast SLAM

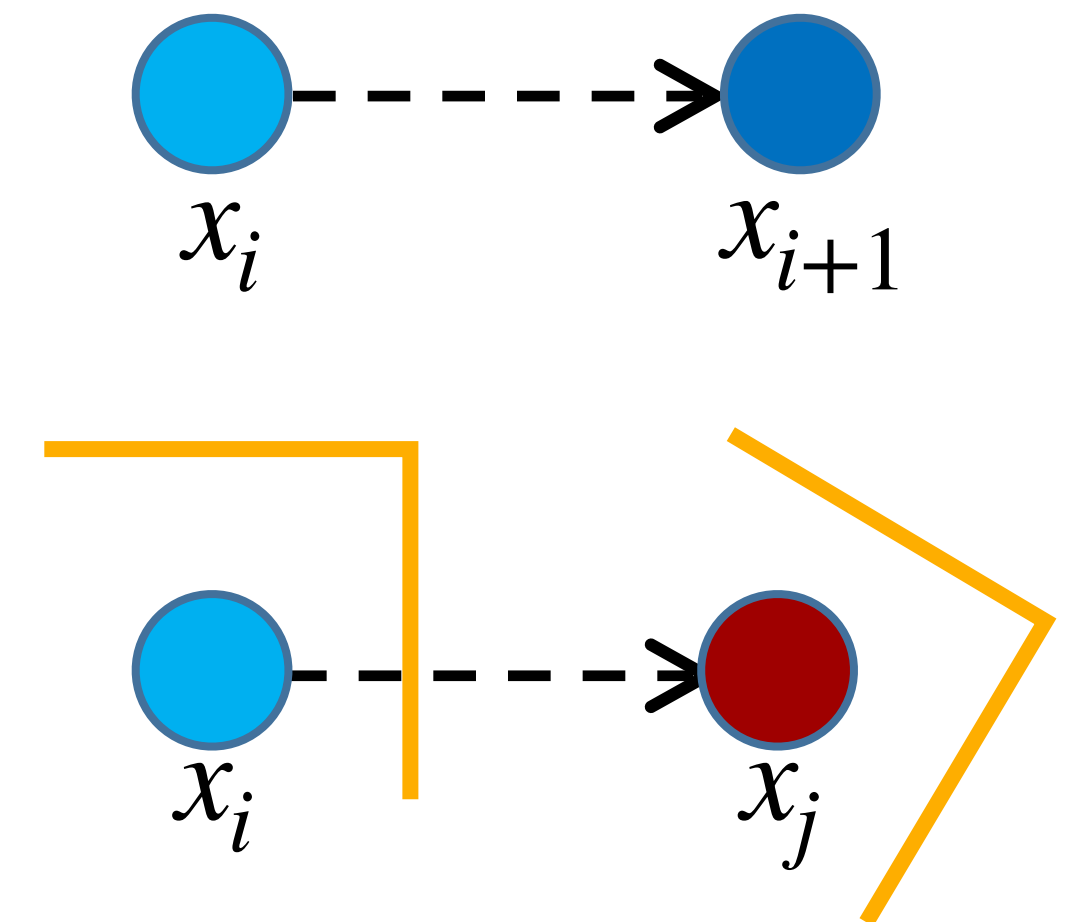


Graph SLAM



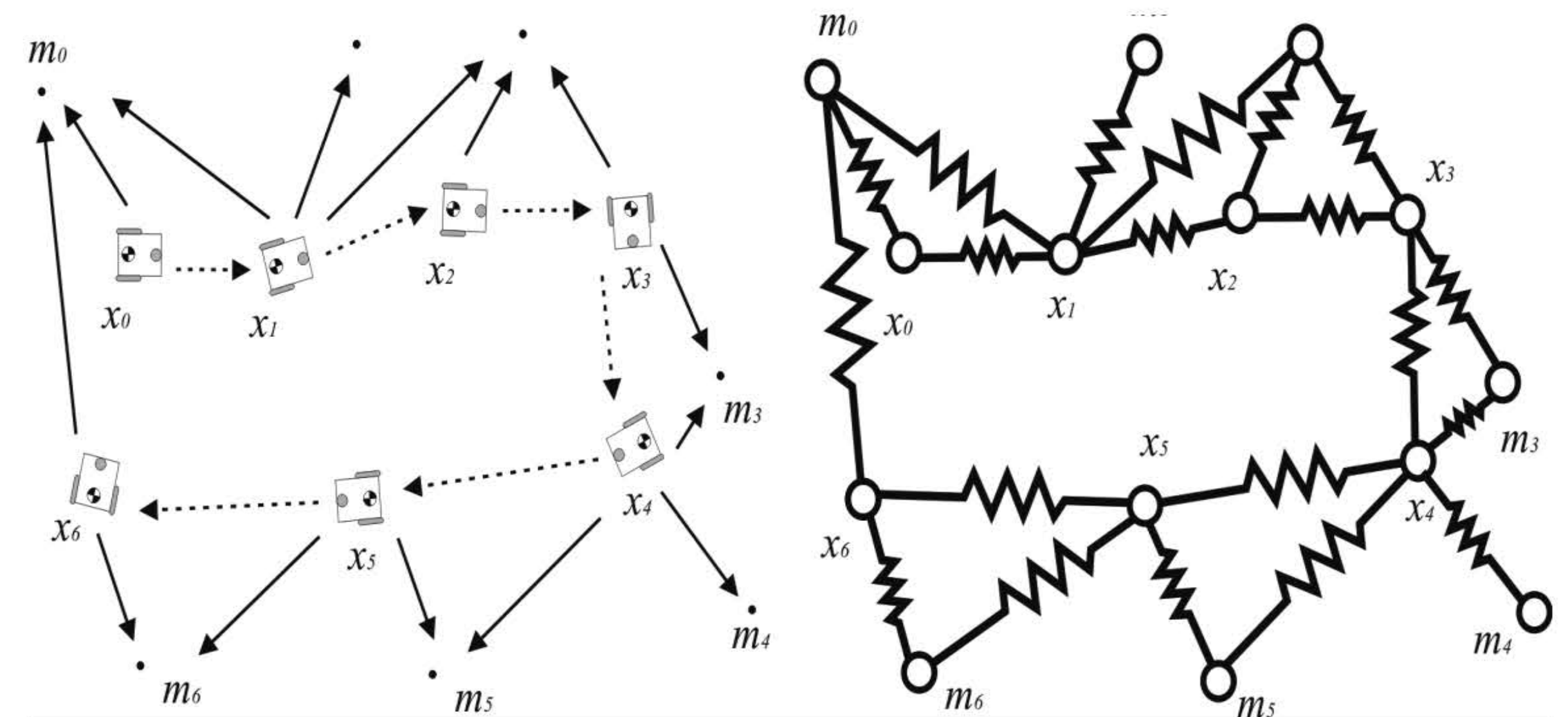
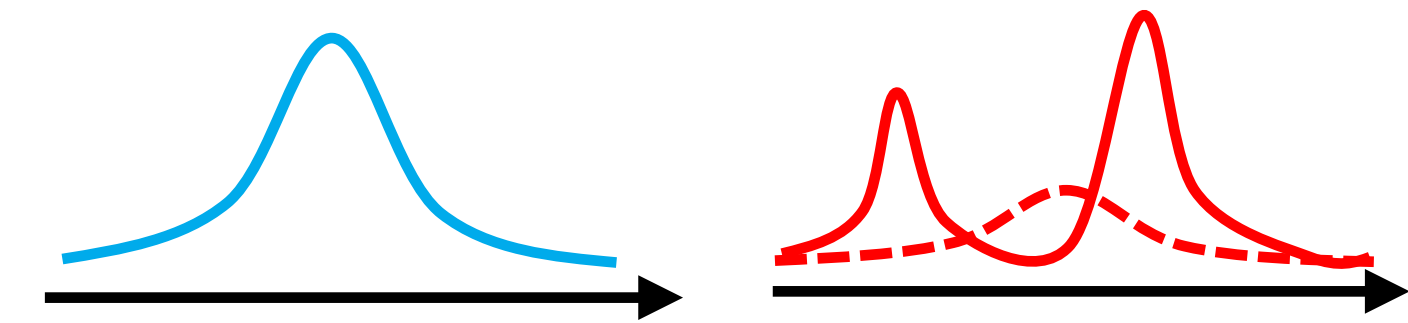
Graph SLAM

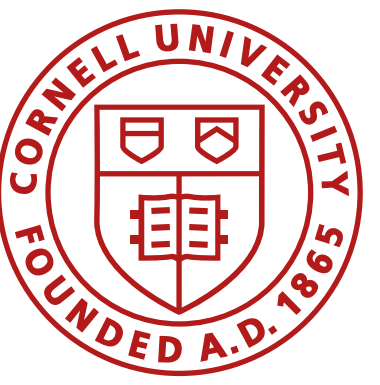
- Graph represents a set of objects where pairs of objects are connected by links encoding relations between them
- Create an edge if...
 - ... the robot moves from x_i to x_{i+1}
 - edge corresponds to odometry measurement
 - ... the robot observes the same part of the environment from x_i and from 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



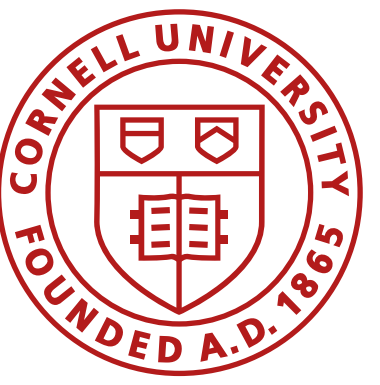
Graph SLAM

- 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



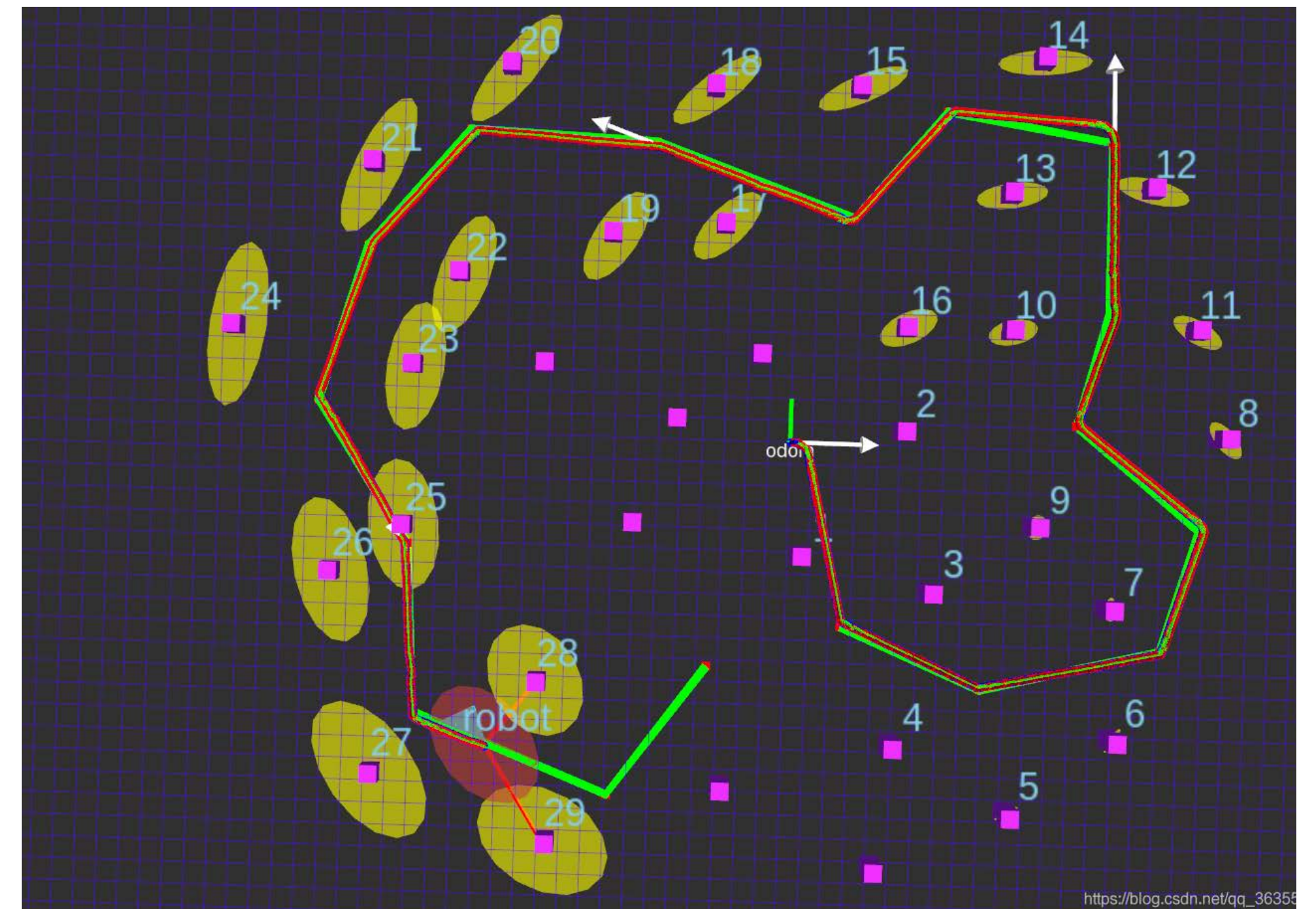


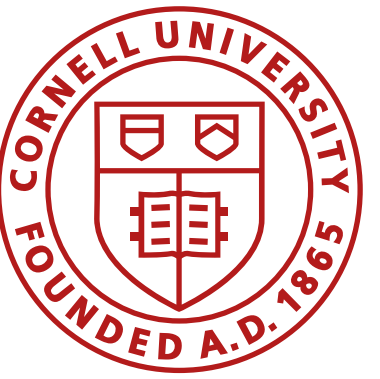
EKF SLAM



EKF SLAM

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





Kalman Filter

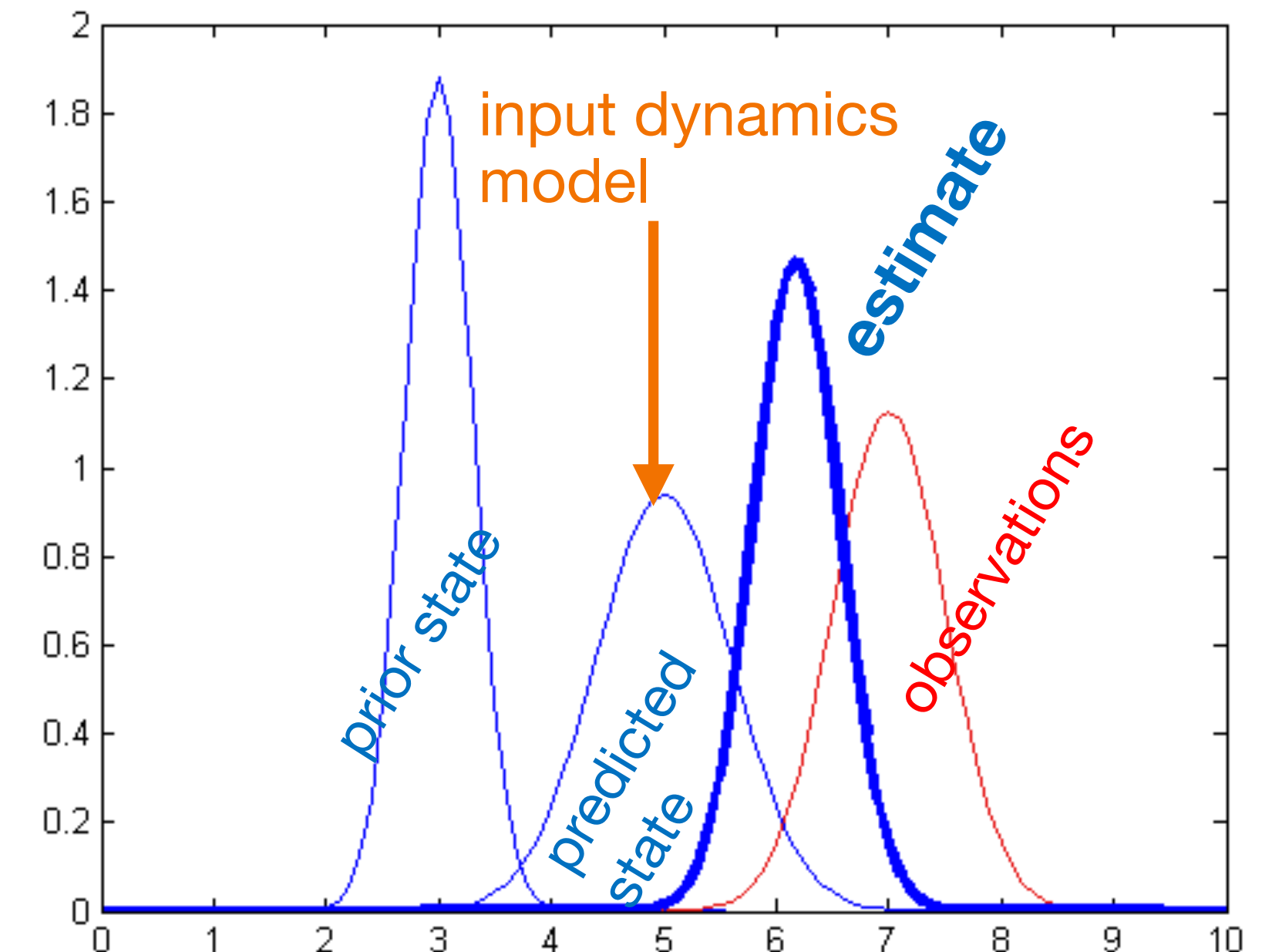
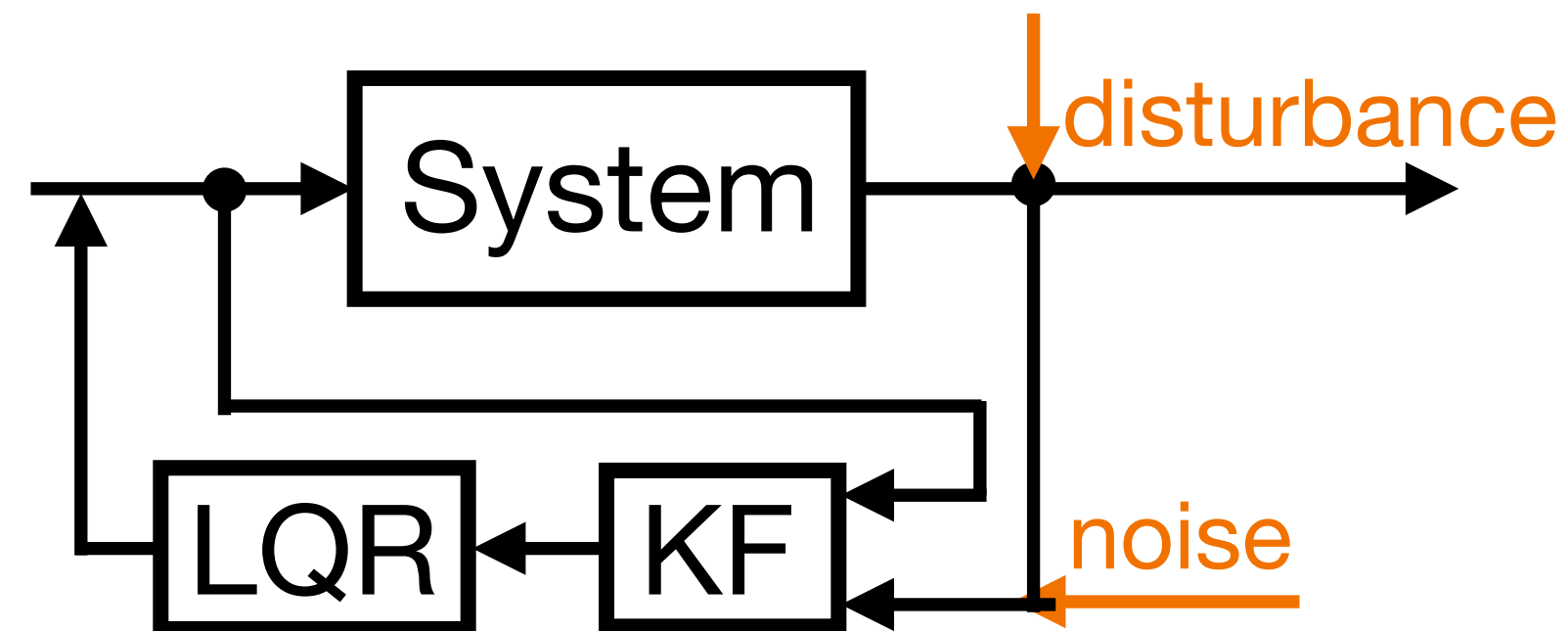
Kalman Filter $(\mu(t-1), \Sigma(t-1), u(t), z(t))$

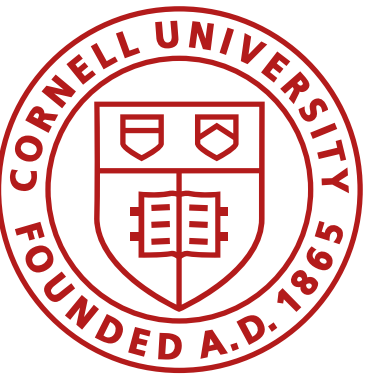
1. $\mu_p(t) = A\mu(t-1) + Bu(t)$
2. $\Sigma_p(t) = A\Sigma(t-1)A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)C^T(C\Sigma_p(t)C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}(z(t) - C\mu_p(t))$
5. $\Sigma(t) = (I - K_{KF}C)\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

prediction

update

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



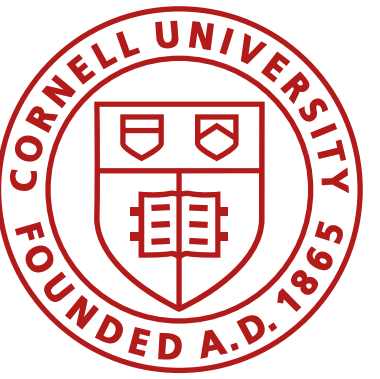


EKF SLAM

- Goal: Estimate $p(x_k, m \mid u_{1:k}, z_{1:N})$
- Assume all noise is Gaussian
- Track a Gaussian belief of the current state and landmarks
- Linearize around every state and run the Kalman Filter

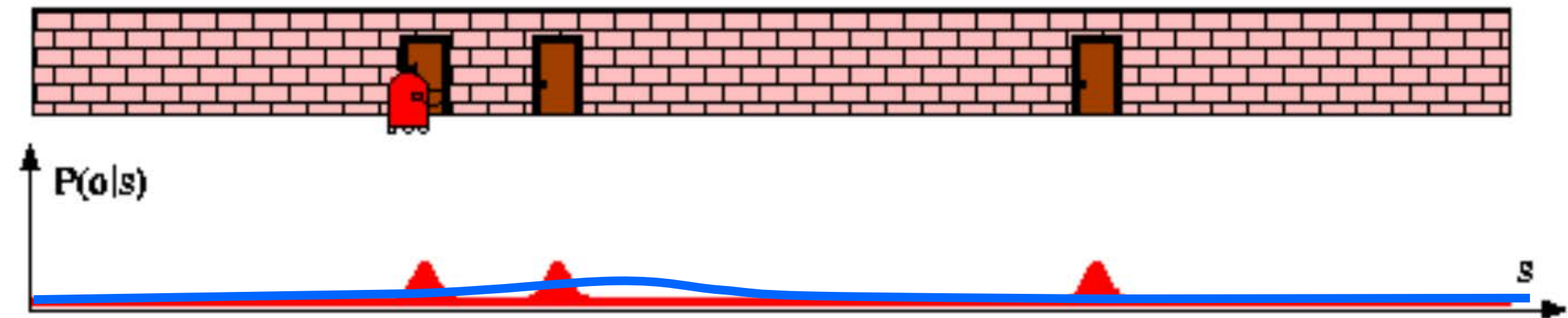
$$x = \begin{bmatrix} \bar{\varphi} \\ \bar{\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \varphi \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix} \quad P = \begin{bmatrix} P_{\varphi\varphi} & P_{\varphi\mathcal{M}} \\ P_{\mathcal{M}\varphi} & P_{\mathcal{M}\mathcal{M}} \end{bmatrix} = \begin{bmatrix} P_{\varphi\varphi} & P_{\varphi\mathcal{L}_1} & \dots & P_{\varphi\mathcal{L}_n} \\ P_{\mathcal{L}_1\varphi} & P_{\mathcal{L}_1\mathcal{L}_1} & \dots & P_{\mathcal{L}_1\mathcal{L}_n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{\mathcal{L}_n\varphi} & P_{\mathcal{L}_n\mathcal{L}_1} & \dots & P_{\mathcal{L}_n\mathcal{L}_n} \end{bmatrix}$$

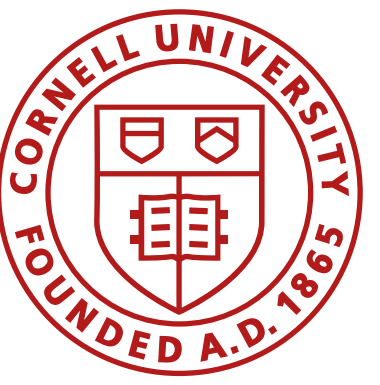
- Landmark matrix grows, making the inversion step costly!
- In Full SLAM the trajectory matrix grows even faster



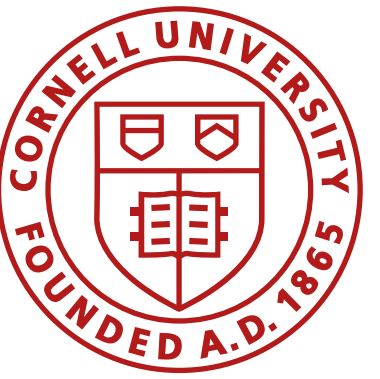
EKF SLAM

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





Fast SLAM

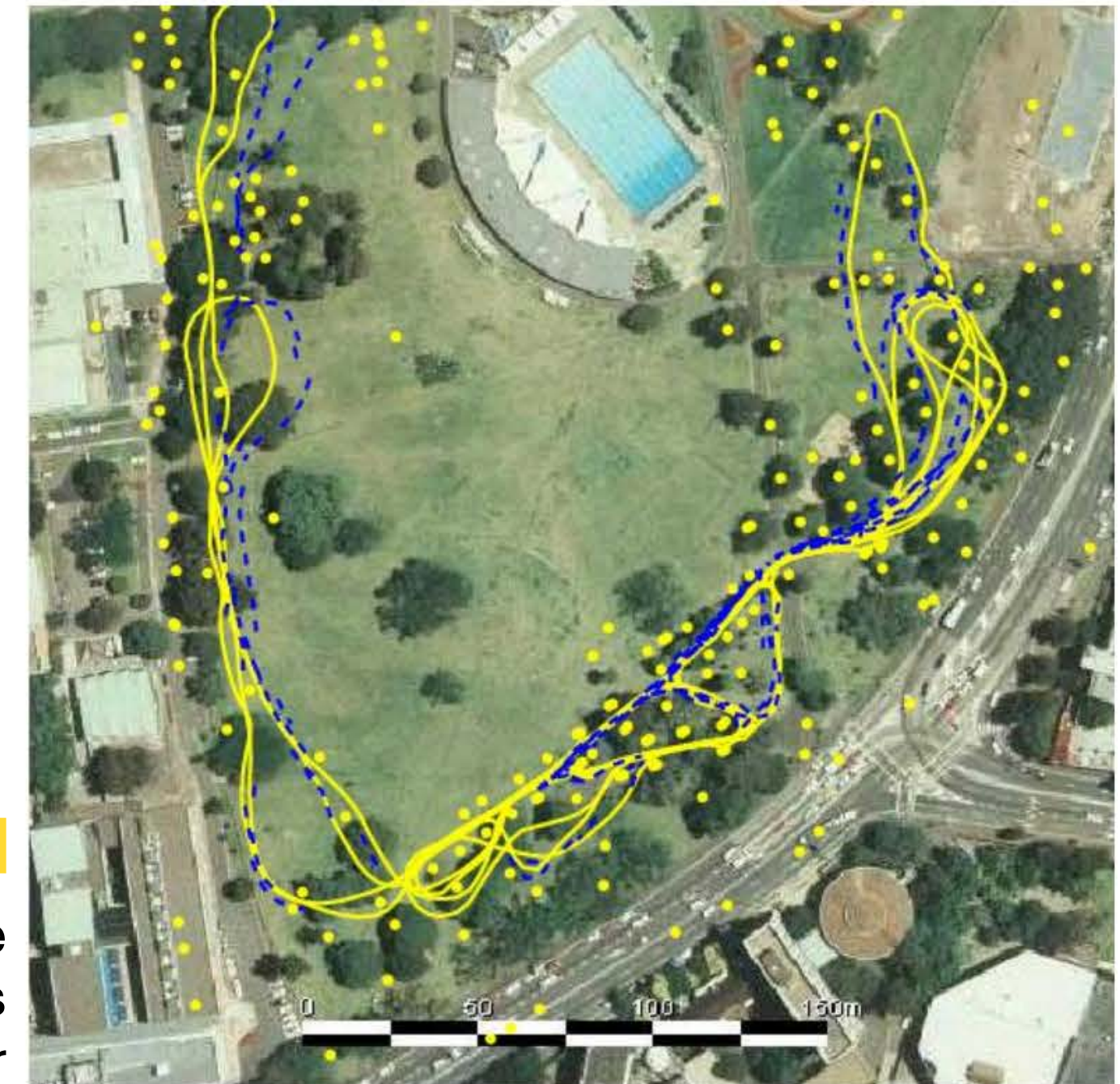


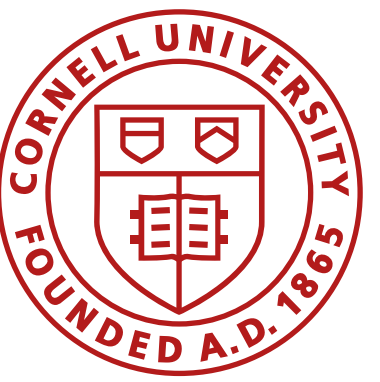
Fast SLAM

- 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-based map
 - 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



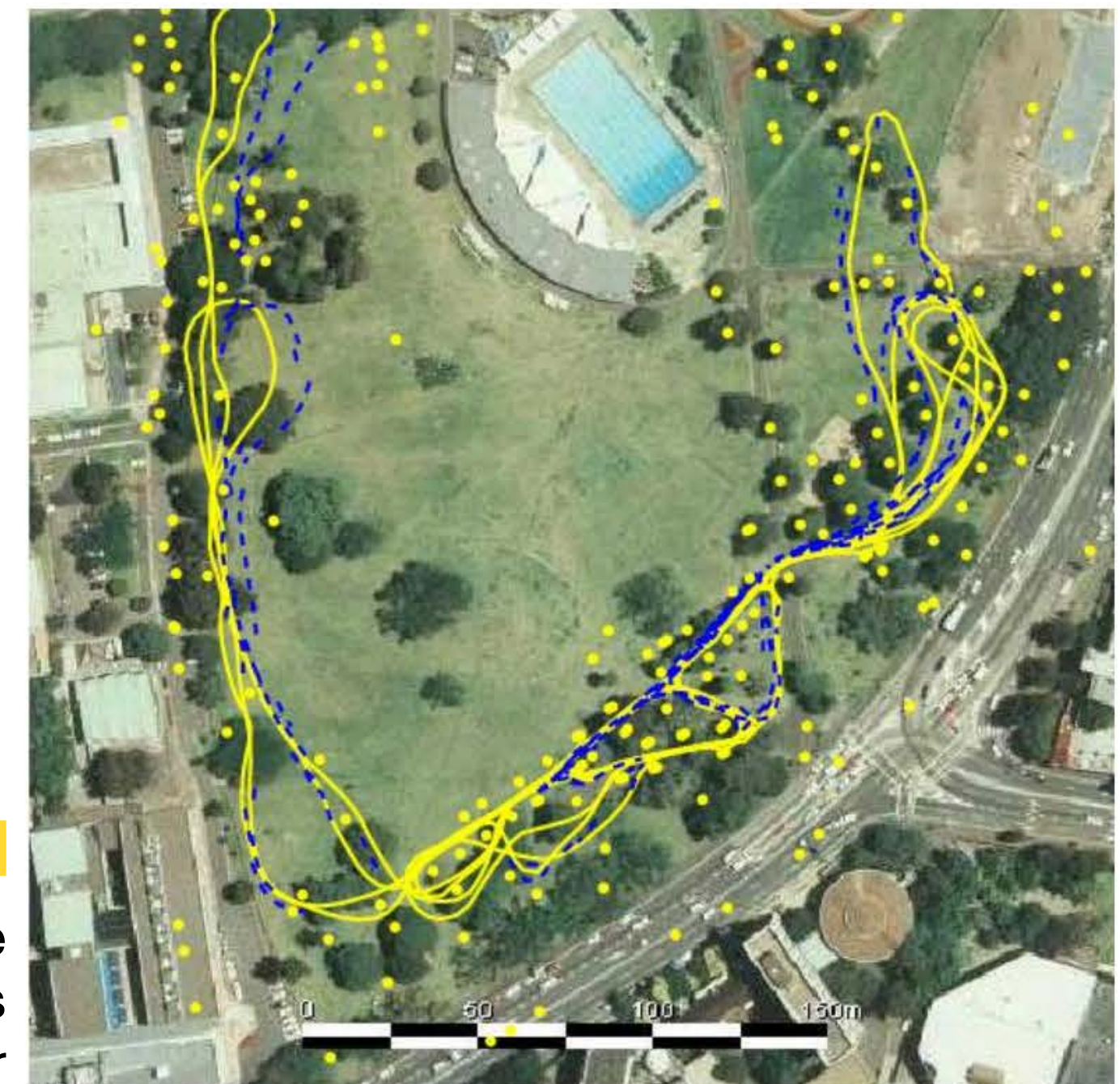


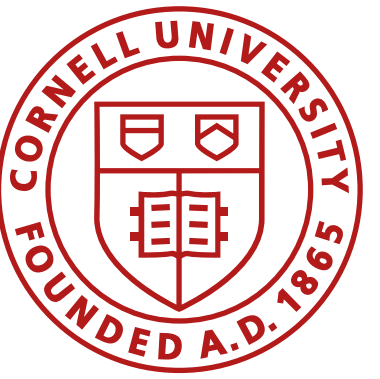
Fast SLAM

- 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, approx 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

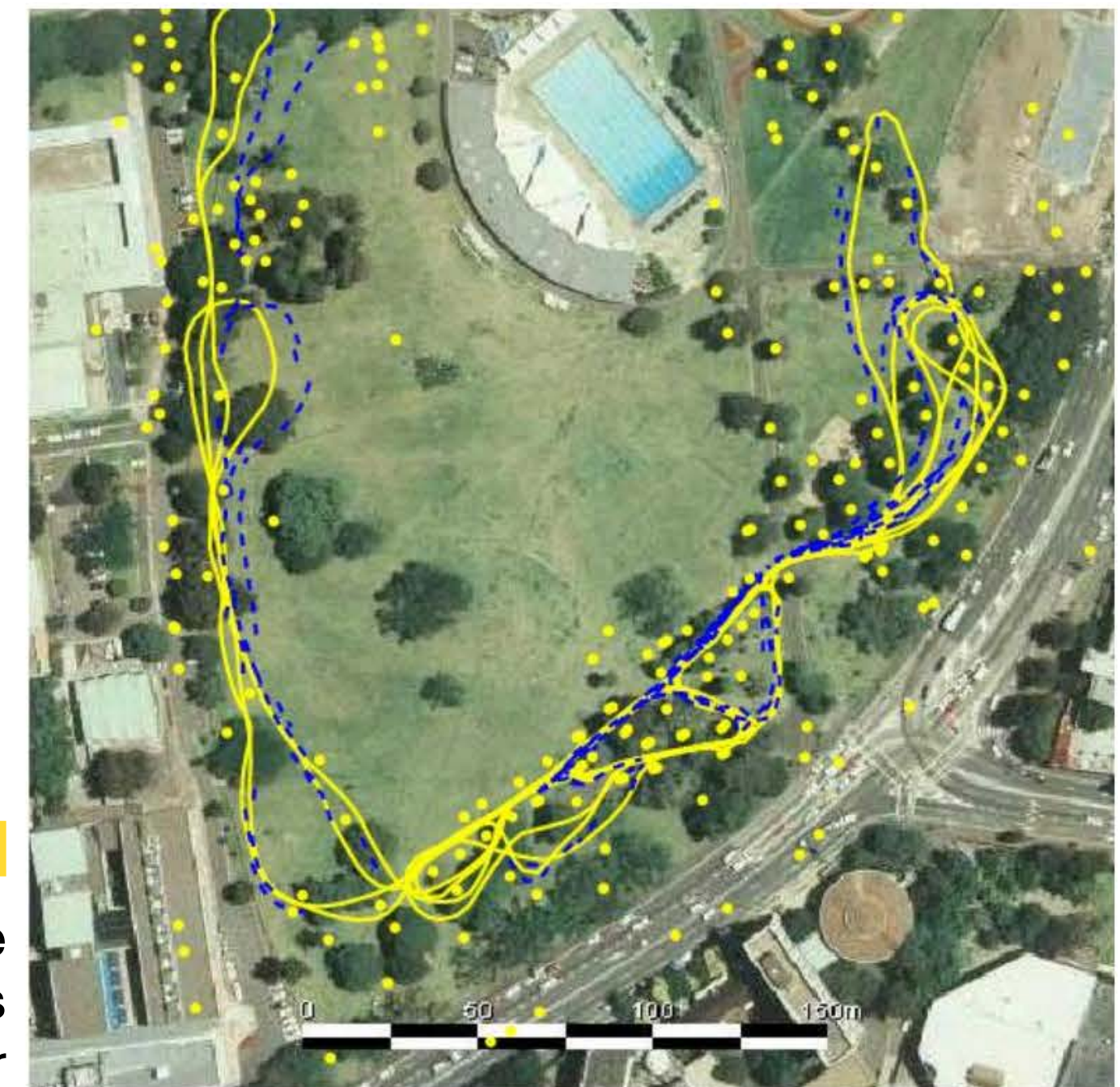
- Distribution is estimated by a fixed number of particles
 - Each particle, k , contains an estimate of robot path and the mean and covariance of each of the n features

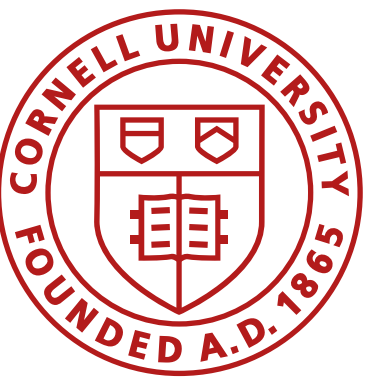
- $P^{[k]}(x_t^{[k]}; \mu^{[k]}, \Sigma_1^{[k]}, \dots, \mu^{[k]}, \Sigma_n^{[k]})$

- **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 update associated landmarks
- **Step 3:** Update weights based on $p(z_t | x_t^{[k]}, m^{[k]})$
- **Step 4:** Resample distribution

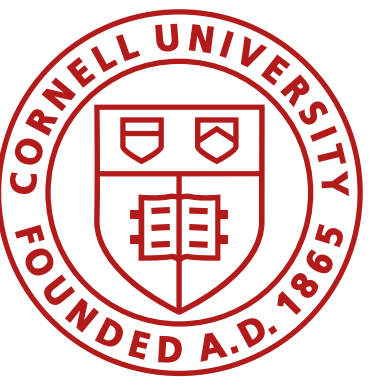
Victoria Park dataset
University of Sydney

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





SLAM State of the Art

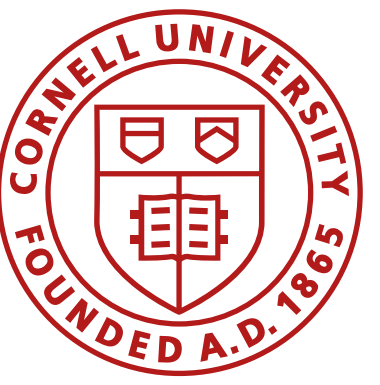


State of the Art in SLAM

The screenshot displays a 2D SLAM software interface. On the left, a 2D grid map shows a robot's path as a white line. The right side shows a live camera feed of a room with desks and a robot. A terminal window at the bottom right shows the following output:

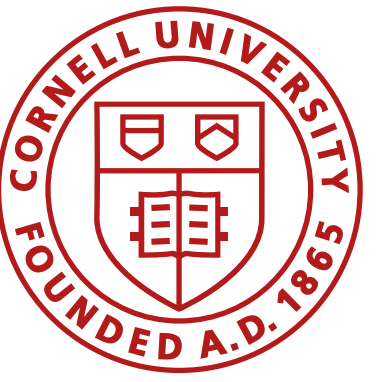
```
[RUNNING] Bag Time: 1426547943.786641 Durat
[RUNNING] Bag Time: 1426547943.787251 Durat
[RUNNING] Bag Time: 1426547943.816881 Durat
[RUNNING] Bag Time: 1426547943.826363 Durat
[RUNNING] Bag Time: 1426547943.821308 Durat
[RUNNING] Bag Time: 1426547943.847086 Durat
[RUNNING] Bag Time: 1426547943.858950 Durat
[RUNNING] Bag Time: 1426547943.867236 Durat
[RUNNING] Bag Time: 1426547943.877302 Durat
[RUNNING] Bag Time: 1426547943.887371 Durat
[RUNNING] Bag Time: 1426547943.907523 Durat
Upn: 19.179297 / 1880.828576
```

At the bottom of the interface, there are time-related fields: ROS Time: 1429159475.18, ROS Elapsed: 228.56, Wall Time: 1429159475.22, Wall Elapsed: 228.56, and an Experimental checkbox.



State of the Art in SLAM

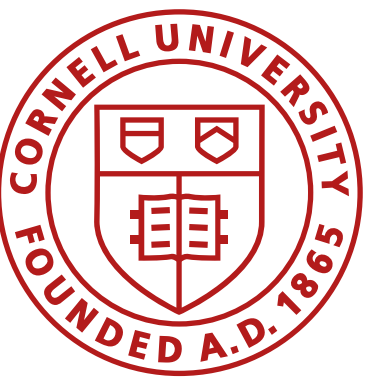




State of the Art in SLAM

- 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

