F1TENTH Autonomous Racing

Localization and Mapping
Particle Filters

Rahul Mangharam
University of Pennsylvania
rahulm@seas.upenn.edu

F1 TENTH

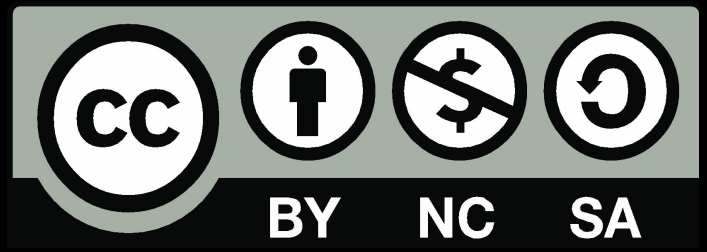
Safe Autonomy Lab
University of Pennsylvania

Penn Engineering

Acknowledgements

This course is a collaborative development with significant contributions from:
Hongrui Zheng (lead), Matthew O'Kelly (lead), Johannes Betz (lead), Madhur Behl, Joseph Auckley, Luca Carlone, Jack Harkins, Kuk Jang, Paril Jain, Sertac Karaman, Dhruv Karthik, Nischal KN, Thejas Kesari, Venkat Krovi, Matthew Lebermann, Kim Luong, Yash Pant, Varundev Shukla, Nitesh Singh, Siddharth Singh, Cyrill Stachniss, Rosa Zheng, Xiyuan Zhu and many others.

We are grateful for learning from each other



Lesson Plan

1. Map making with Hector SLAM
2. Particle Filter Localization
3. Adaptive Monte Carlo Localization (AMCL)
4. Recursive Bayes Filtering used in AMCL
5. Tuning Particle Filters in ROS

Intuition

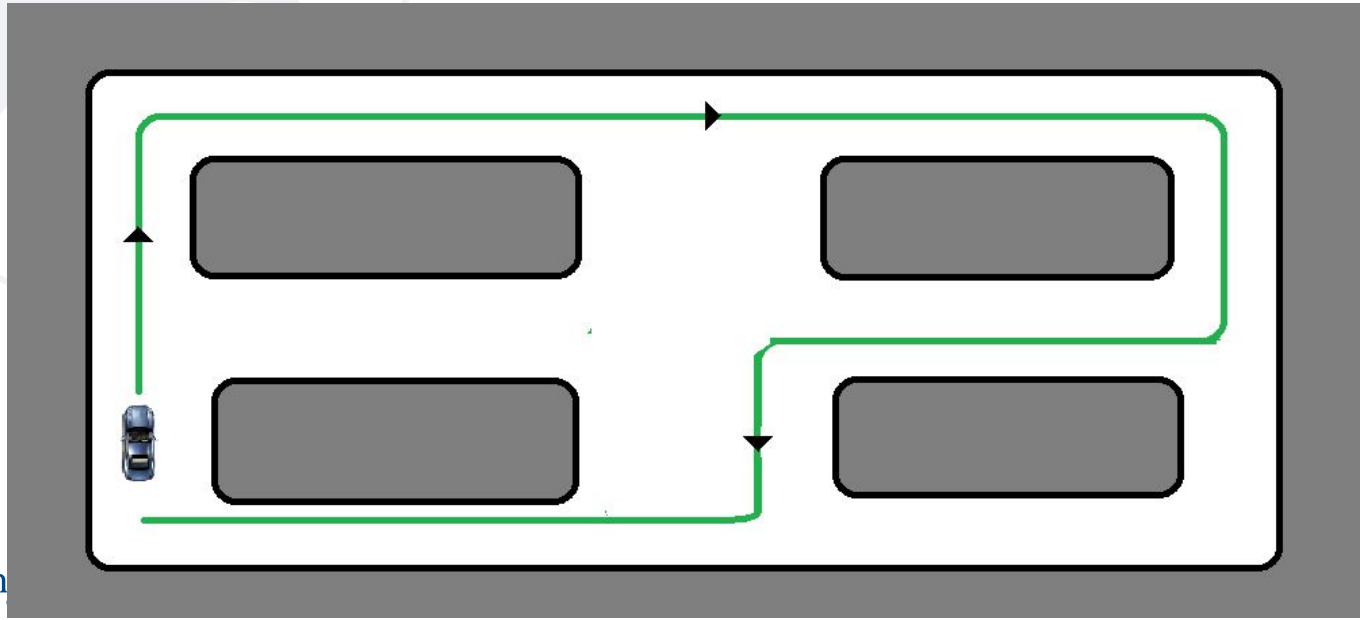
Analysis

Supplement lecture: Hands-on Tutorial

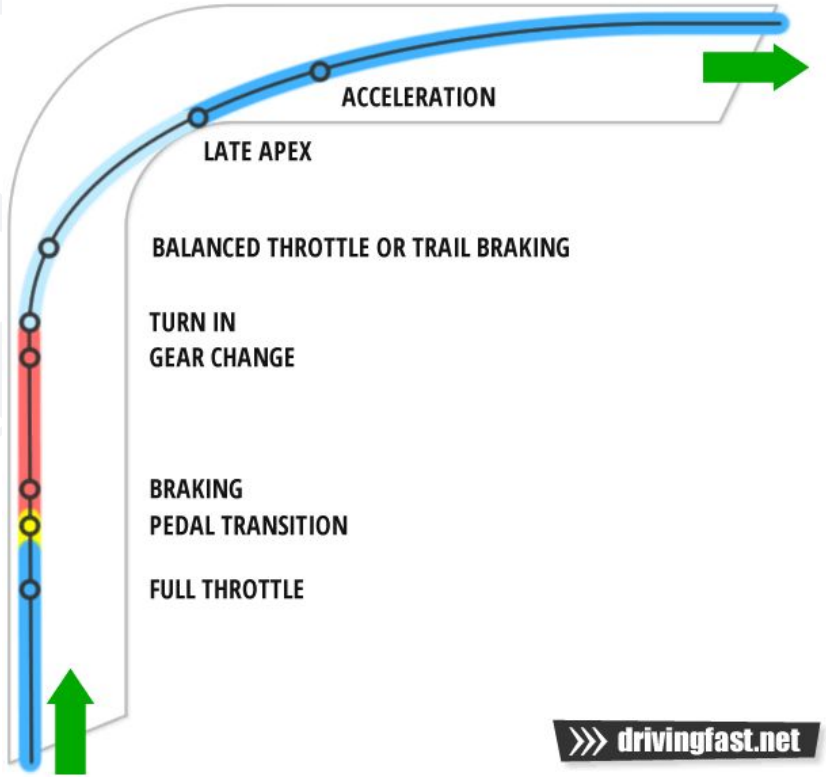
Limitations : Basic Path Planning

High Level Path Assignments

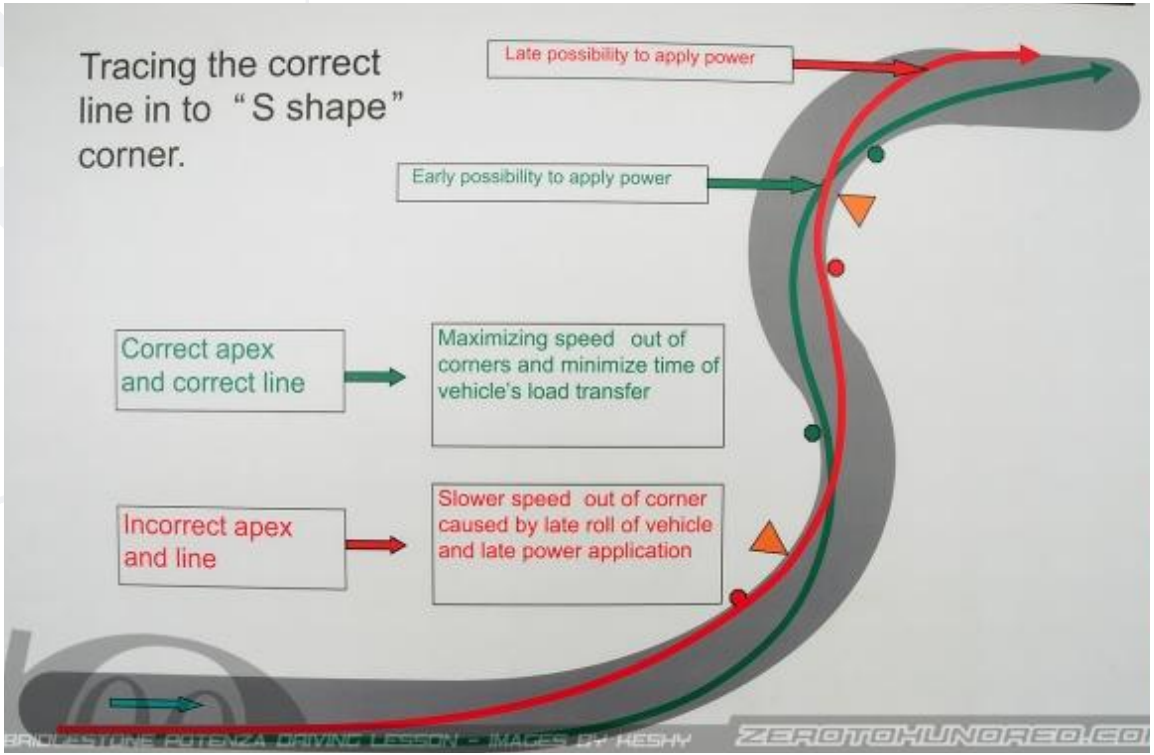
→ 2nd right, 2nd right, 1st right, 1st left, 1st right



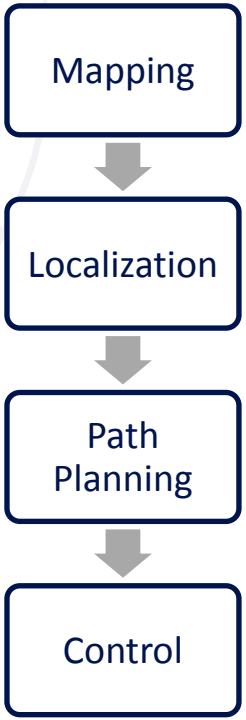
Race Lines


ACCELERATION
LATE APEX
BALANCED THROTTLE OR TRAIL BRAKING
TURN IN
GEAR CHANGE
BRAKING
PEDAL TRANSITION
FULL THROTTLE
drivingfast.net

Limitations : No Future Information

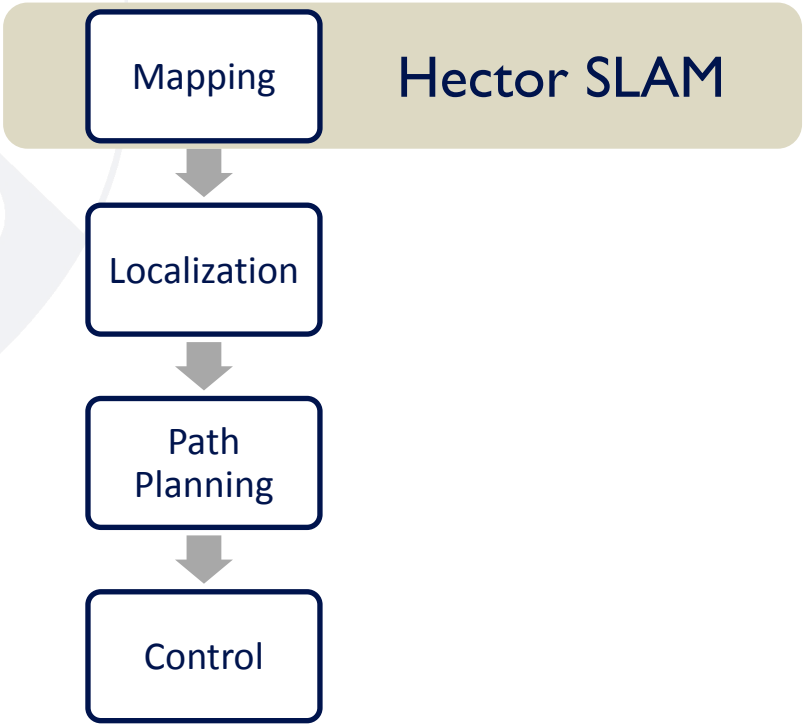


System Overview



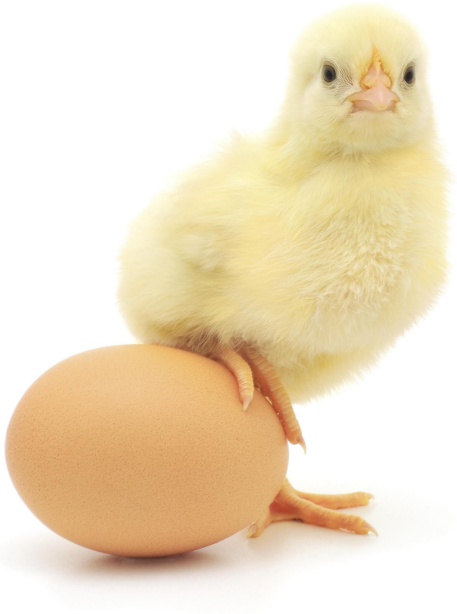
Penn Engineering

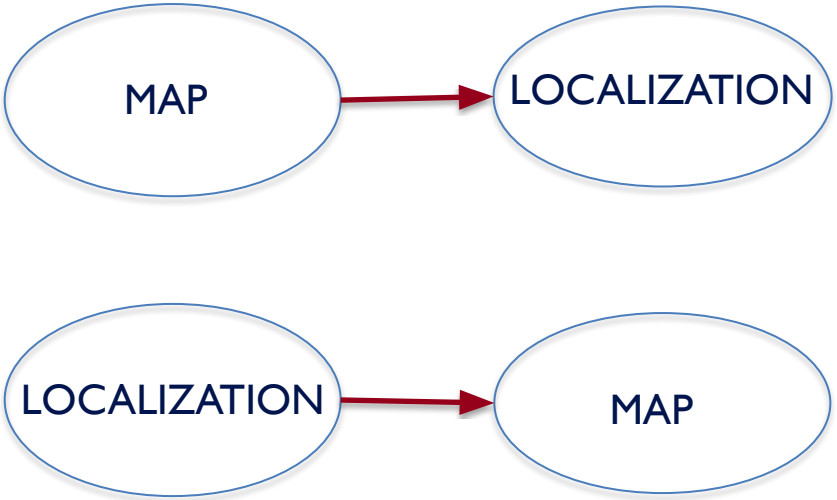
System Overview



Penn Engineering

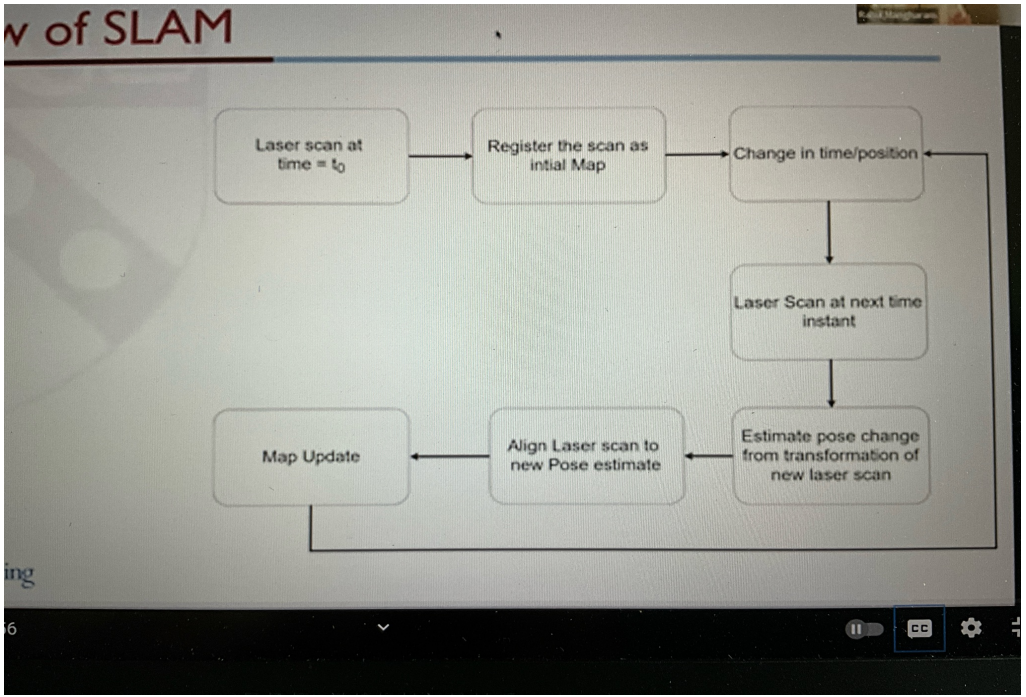
SLAM :A Chicken-Egg problem

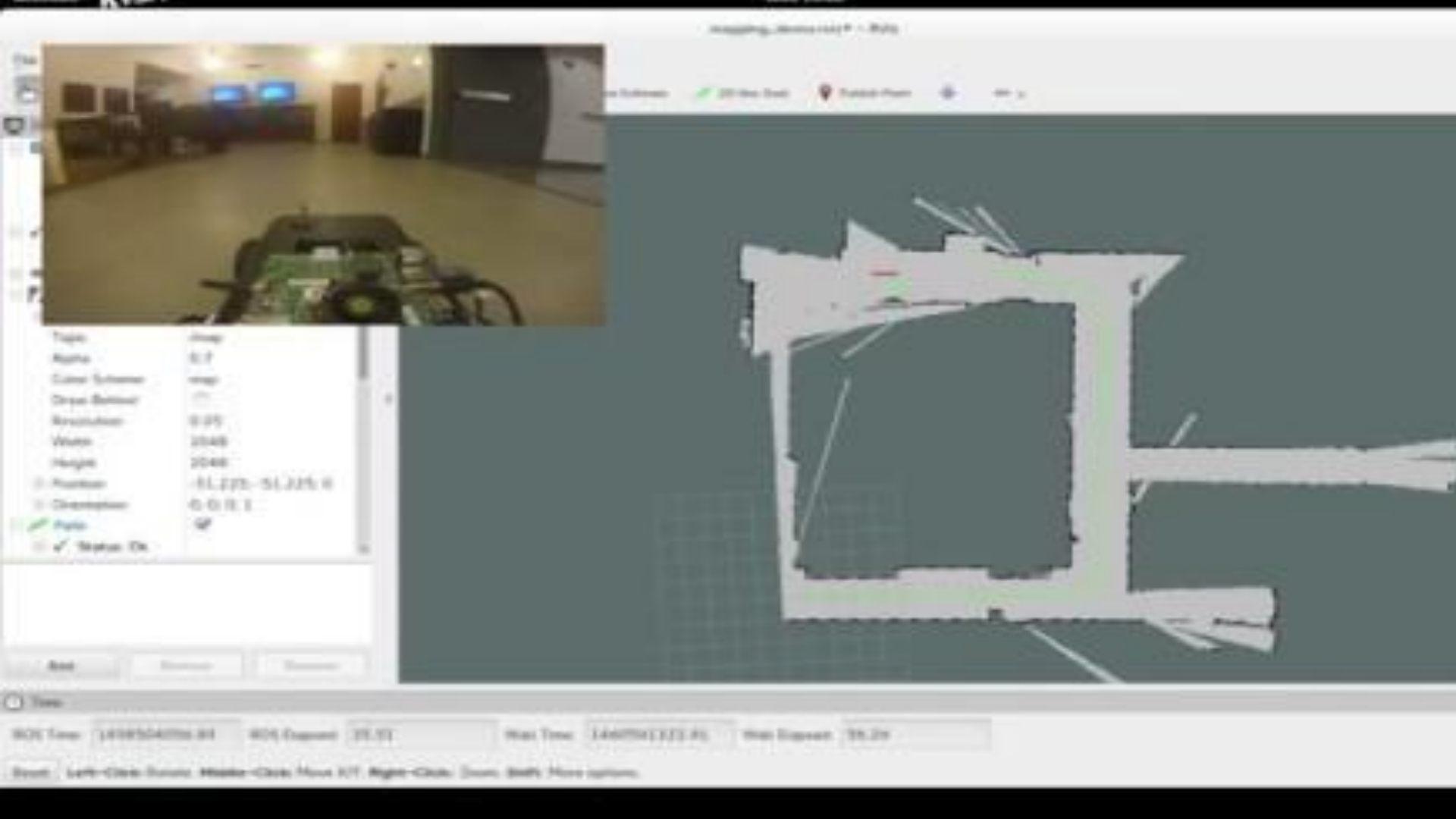




At first you need a map to localize the robot and the localization information is needed to make the map, chicken egg problem: who came first

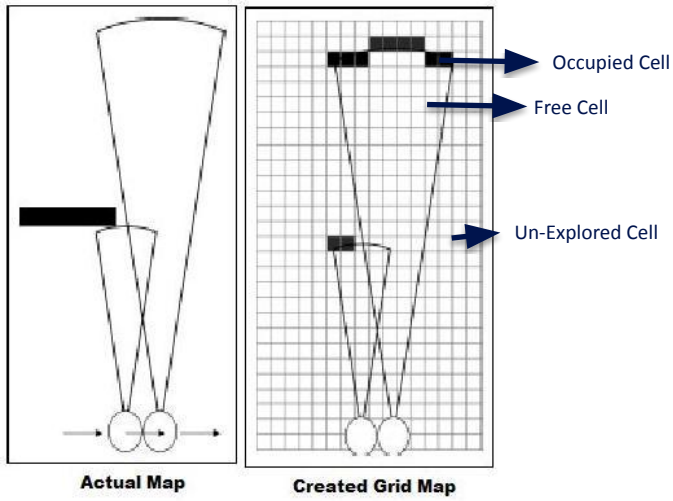
Overview of SLAM





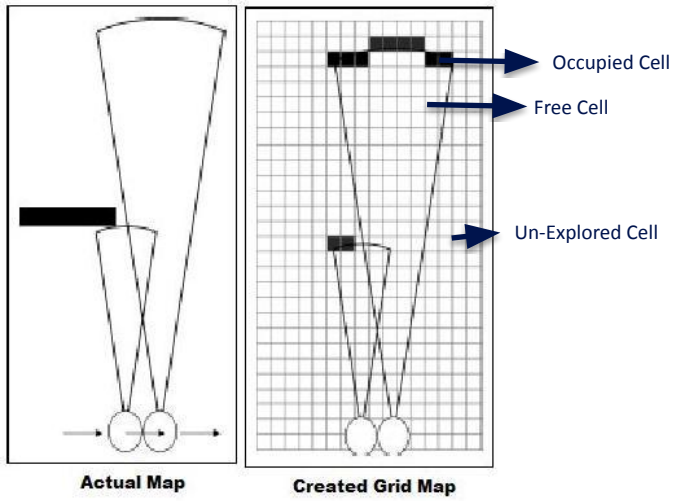
Occupancy Grid Mapping

Measurement Model



Penn Engineering

Occupancy Grid Mapping



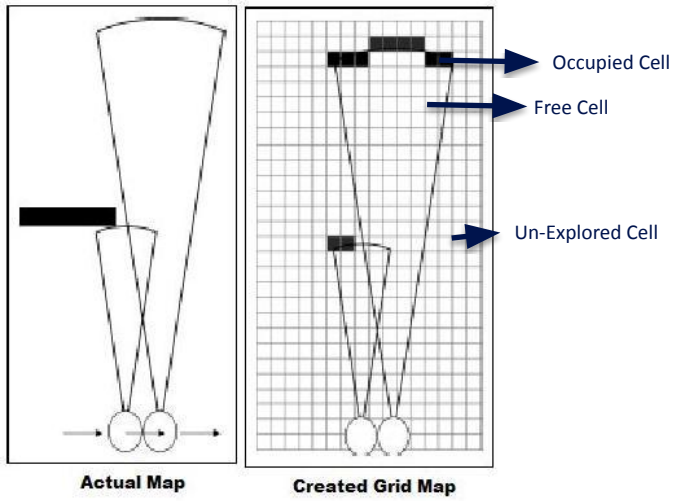
Measurement Model

- Measurement :

  $m_{x,y} = 1$ LiDAR hit

  $m_{x,y} = 0$ No occlusion

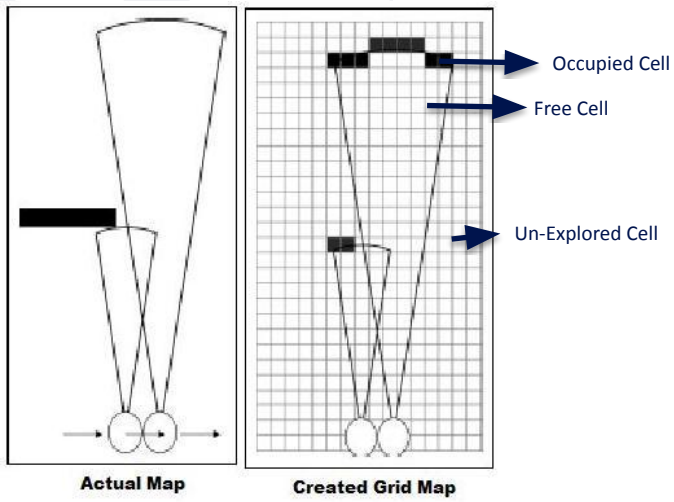
Occupancy Grid Mapping



Measurement Model

- Measurement :
  - $m_{x,y} = 1$ LiDAR hit
  - $m_{x,y} = 0$ No occlusion
- Map Cell:
  - Z = 1 Occupied
  - Z = 0 UnExplored
  - Z = -1 Free

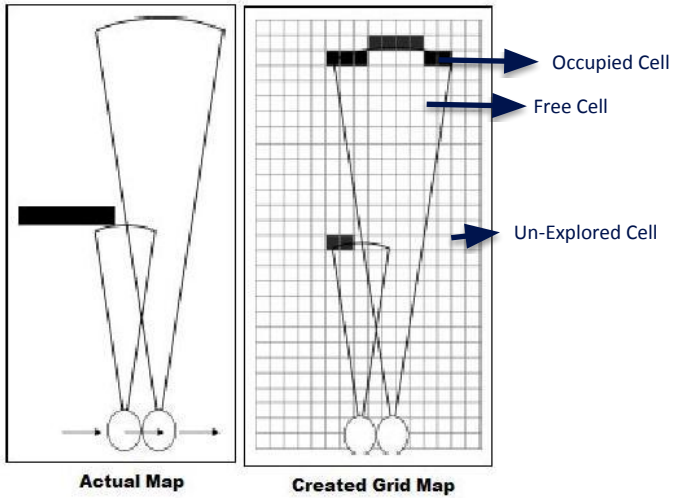
Occupancy Grid Mapping



Measurement Model

- Measurement :
  - $m_{x,y} = 1$ LiDAR hit
  - $m_{x,y} = 0$ No occlusion
- Map Cell:
  - Z = 1 Occupied
  - Z = 0 UnExplored
  - Z = -1 Free
- Measurement Model : $p(z|m_{x,y})$

Occupancy Grid Mapping



$$\log odd\_occ := \log \frac{p(z=1|m_{x,y}=1)}{p(z=1|m_{x,y}=0)}$$

Log Probability for occupied cells

$$\log odd\_free := \log \frac{p(z=-1|m_{x,y}=0)}{p(z=-1|m_{x,y}=1)}$$

Log Probability for free cells

Penn Engineering

Occupancy Grid Mapping

Map Update :

- Cells with z = 1:
  - log `odd` = log `odd` + log `odd_occ`
- Cells with z = -1:
  - log `odd` = log `odd` - log `odd_free`

Occupancy Grid Mapping

Map Update :

- Cells with z = 1:
  - log `odd` = log `odd` + log `odd_occ`
- Cells with z = -1:
  - log `odd` = log `odd` - log `odd_free`

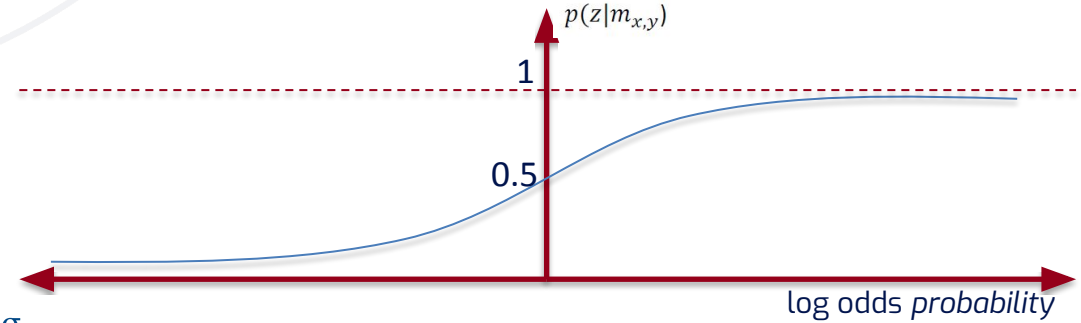
- Threshold the cell values by upper/lower limit to avoid being completely certain

Occupancy Grid Mapping

Map Update :

- Cells with z = 1:
  - log `odd` = log `odd` + log `odd_occ`
- Cells with z = -1:
  - log `odd` = log `odd` - log `odd_free`

- Threshold the cell values by upper/lower limit to avoid being completely certain

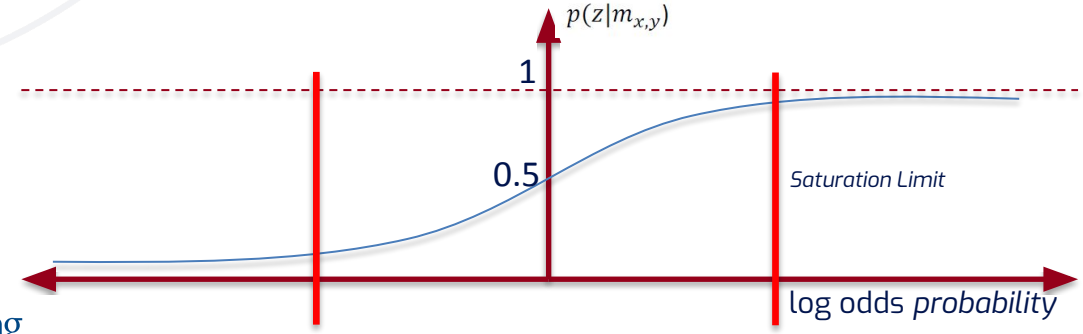


Occupancy Grid Mapping

Map Update :

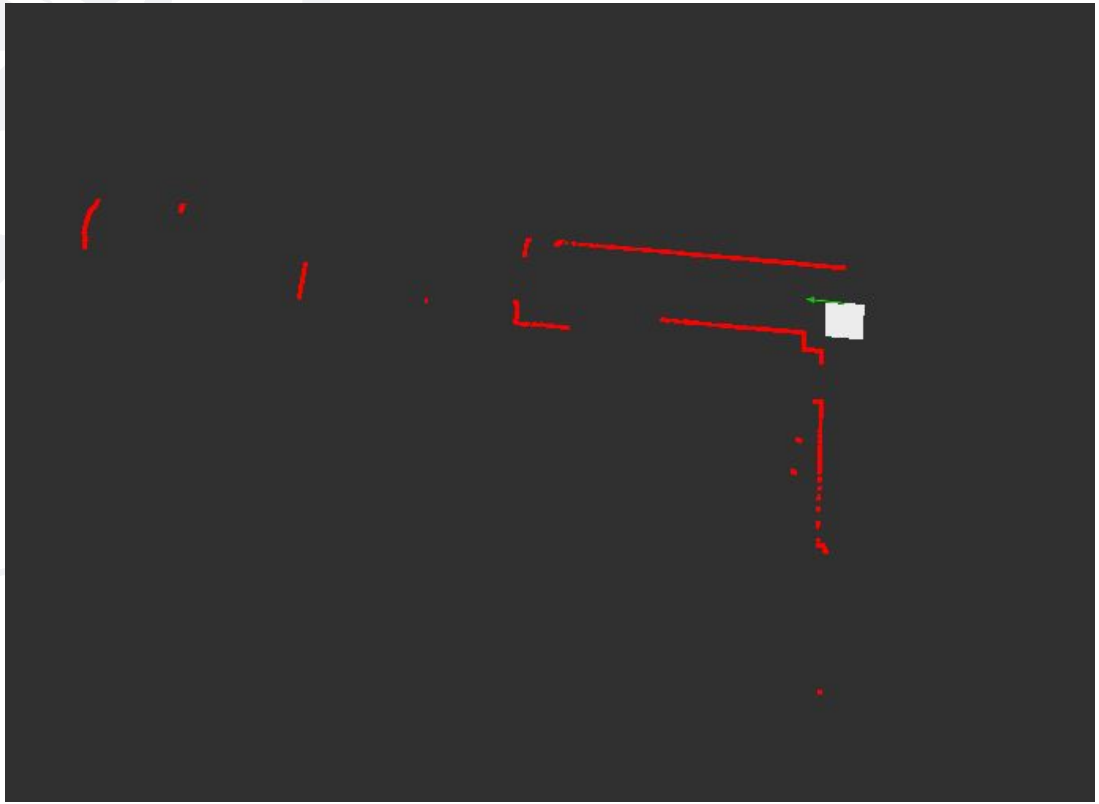
- Cells with z = 1:
  - log `odd` = log `odd` + log `odd_occ`
- Cells with z = -1:
  - log `odd` = log `odd` - log `odd_free`

- Threshold the cell values by upper/lower limit to avoid being completely certain



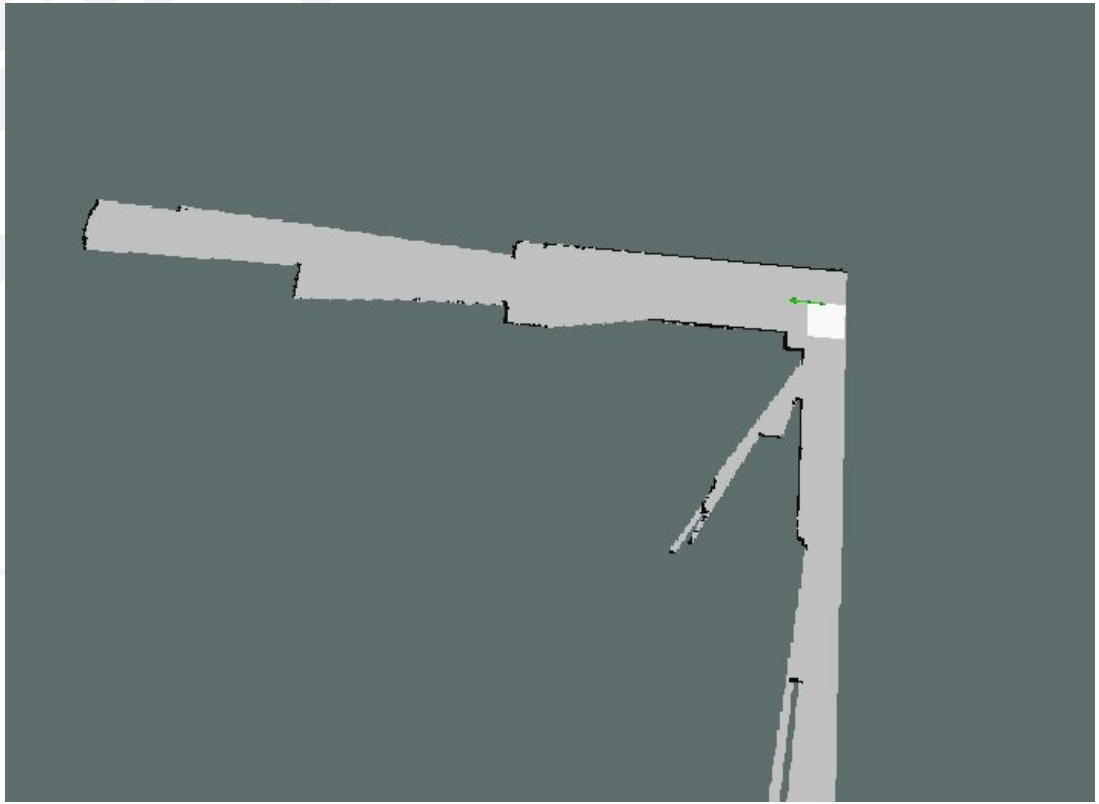
Penn Engineering

Registering the first Scan



Penn Engineering

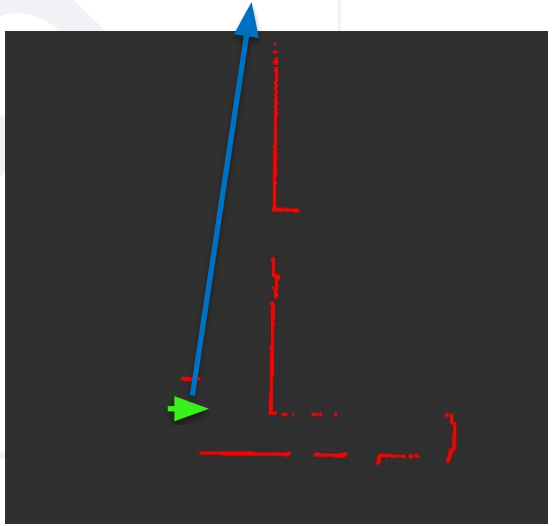
Registering the first Scan



Penn Engineering

Scan Matching

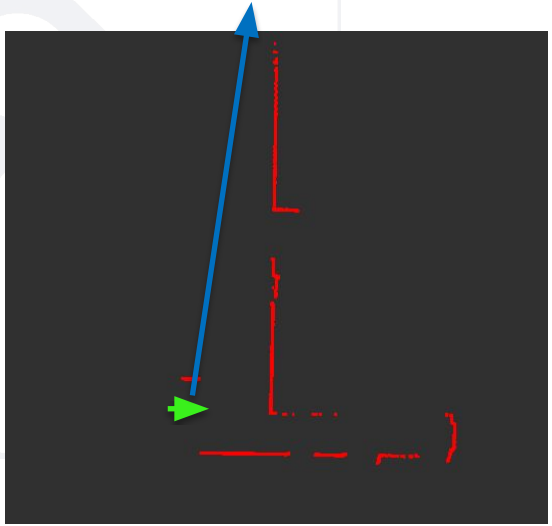
Pose of the Car at t = t1



Laser Scans w.r.t car at Time t = t_1

Scan Matching

Pose of the Car at t = t1



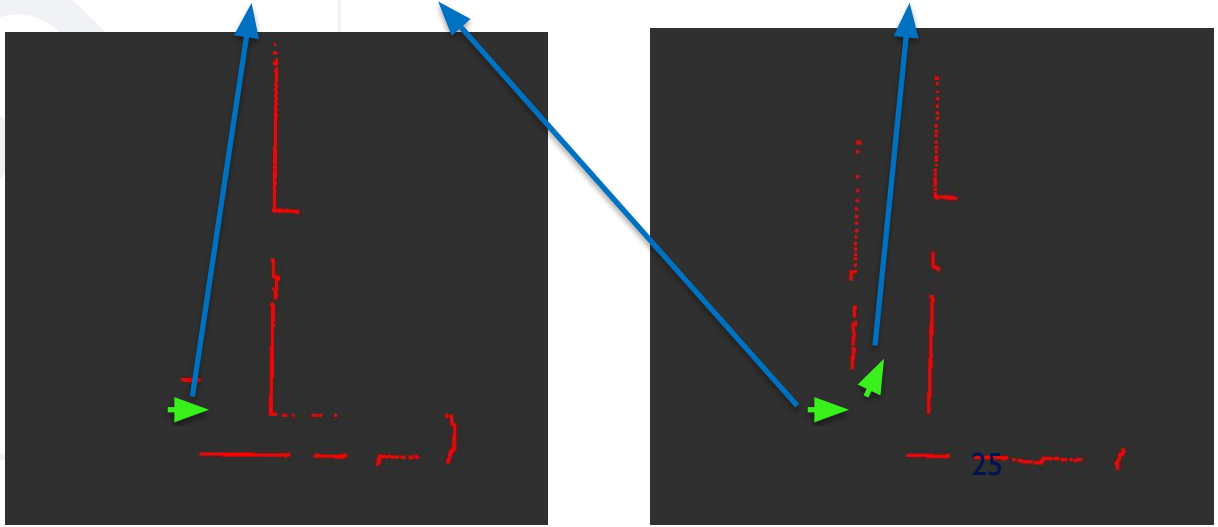
Laser Scans w.r.t car at Time t = t_1

Laser Scans w.r.t car at Time t = t_2

Scan Matching

Pose of the Car at t = t1

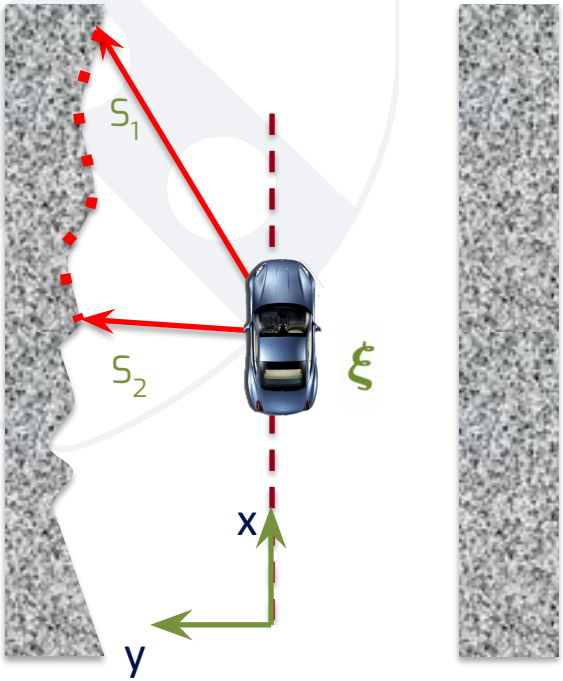
Pose of the Car at t=t2



Laser Scans w.r.t car at Time t = t_1

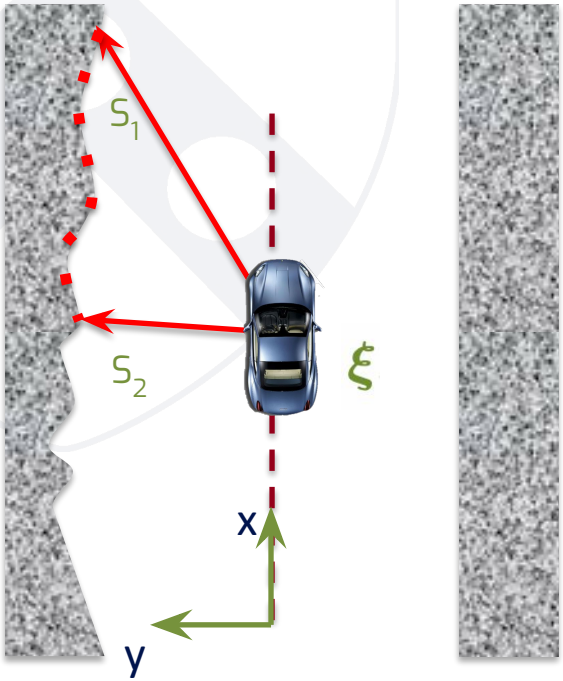
Laser Scans w.r.t car at Time t = t_2

Scan matching: Hector Slam



Penn Engineering

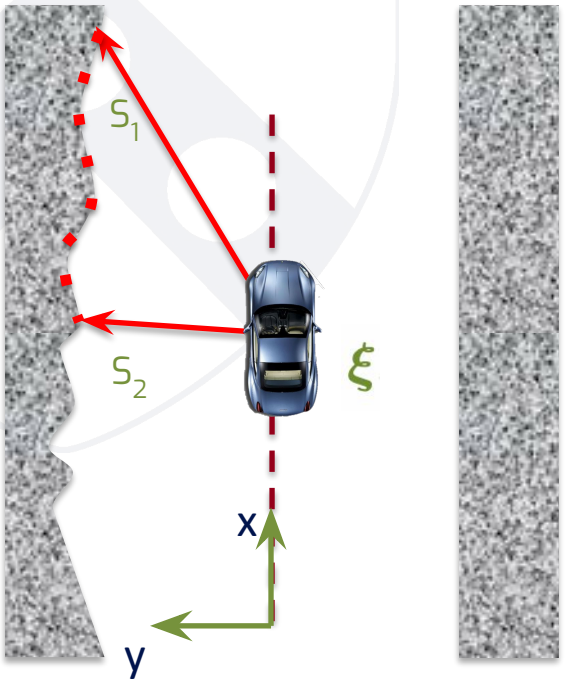
Scan matching: Hector Slam



Robot Pose $\boldsymbol{\xi} = (p_x, p_y, \psi)^{\mathrm{T}}$

Penn Engineering

Scan matching: Hector Slam



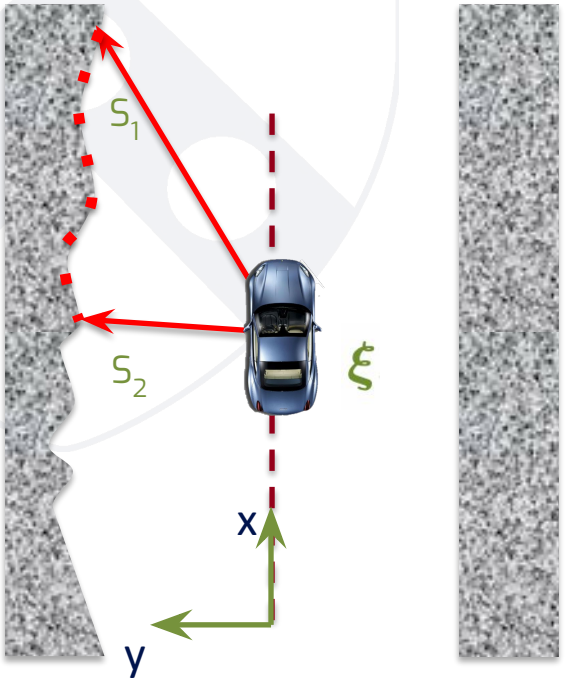
Robot Pose $\boldsymbol{\xi} = (p_x, p_y, \psi)^{\mathrm{T}}$

Impact coordinates of i[th] scan in world frame

Total of n scans

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

Scan matching: Hector Slam



Robot Pose $\boldsymbol{\xi} = (p_x, p_y, \psi)^{\mathrm{T}}$

Impact coordinates of i[th] scan in world frame

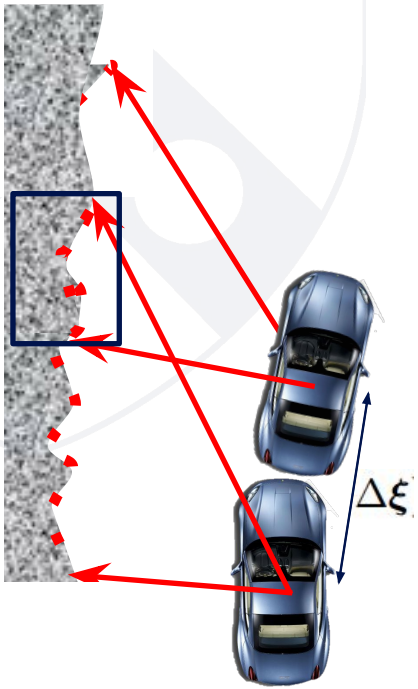
Total of n scans

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

Map Value at coordinates given by S_i

Penn Engineering

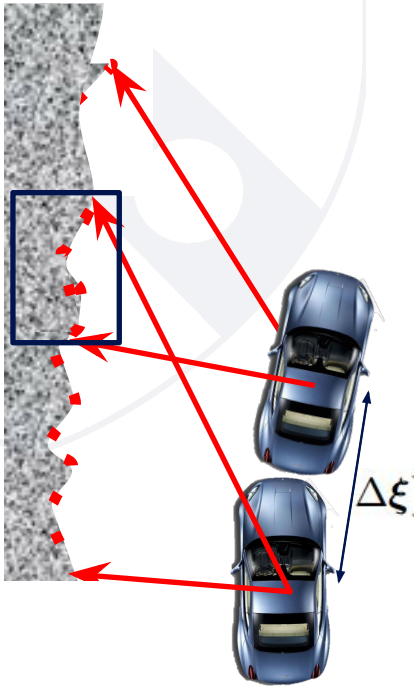
Scan matching: Hector Slam



$$\sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}))]^2 \rightarrow 0\,.$$

Penn Engineering

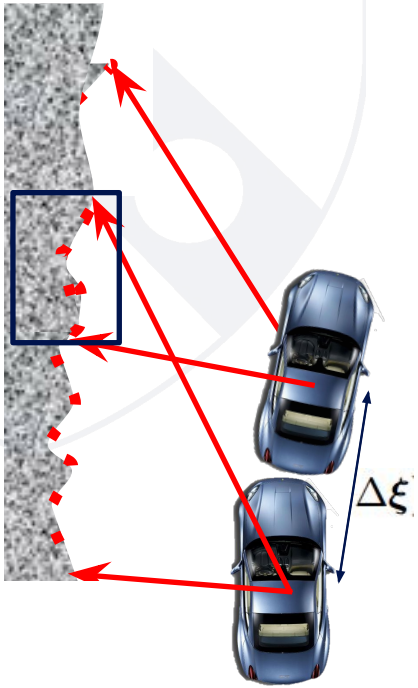
Scan matching: Hector Slam



$$\sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}))]^2 \rightarrow 0\,.$$

Taylor Expansion of Function M

Scan matching: Hector Slam

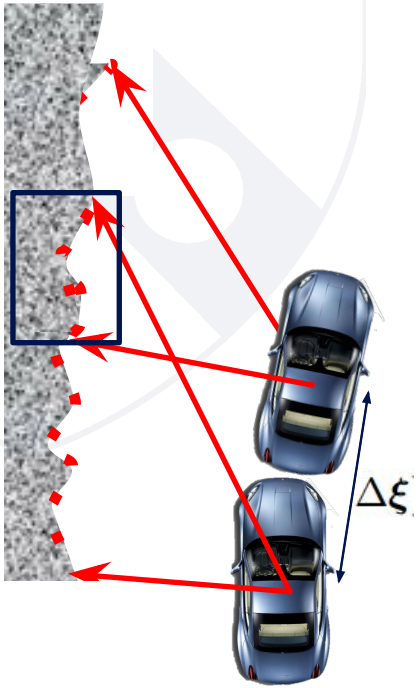


$$\sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}))]^2 \to 0\,.$$

Taylor Expansion of Function M

$$\sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi})) - \nabla M(\mathbf{S}_i(\boldsymbol{\xi}))\frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\Delta\boldsymbol{\xi})\right]^2 \to 0$$

Scan matching: Hector Slam



$$\sum_{i=1}^{n} [1 - M(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi}))]^2 \to 0\,.$$

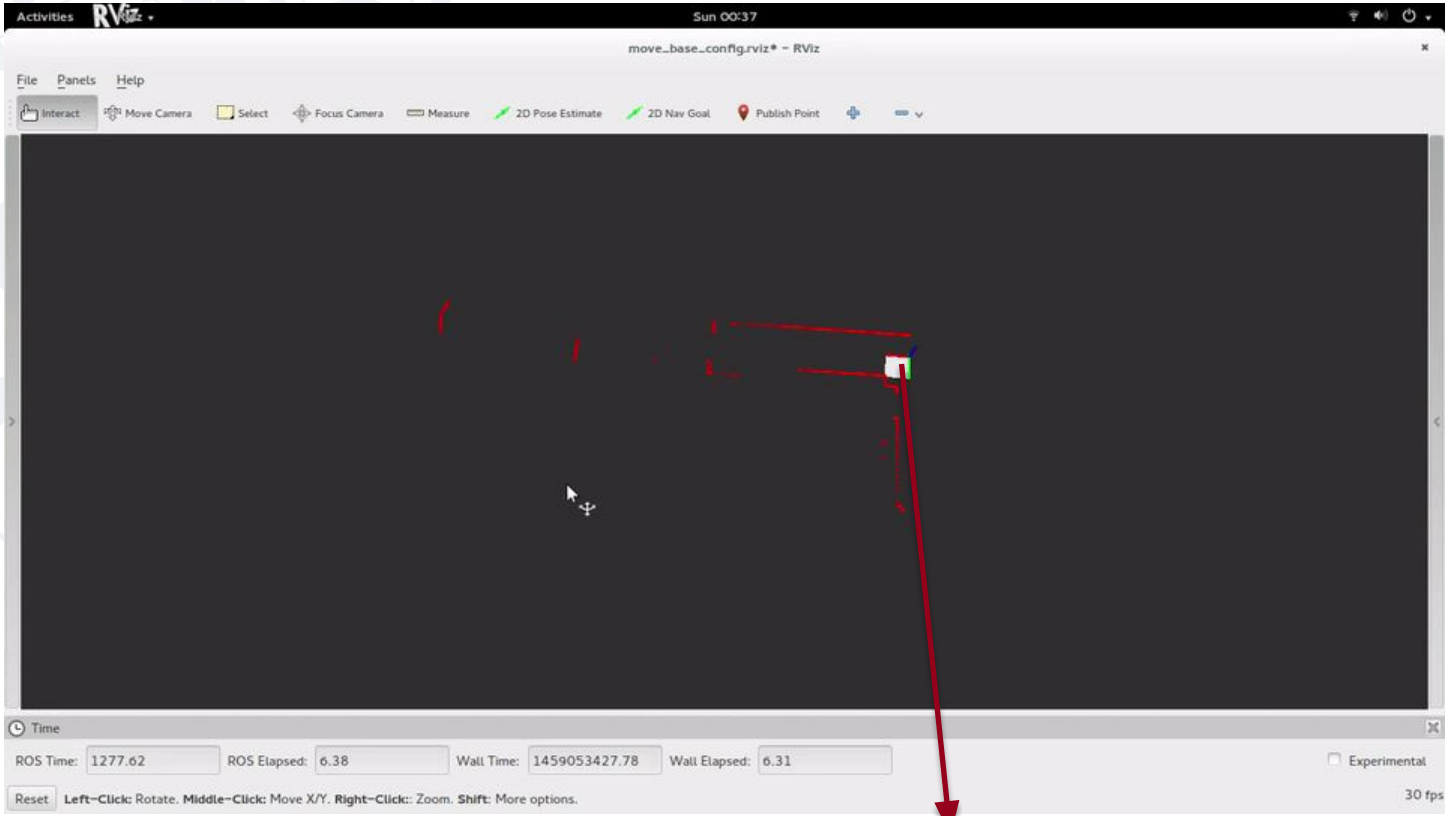
Taylor Expansion of Function M

$$\sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi})) - \nabla M(\mathbf{S}_i(\boldsymbol{\xi}))\frac{\partial \mathbf{S}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\Delta\boldsymbol{\xi})\right]^2 \to 0$$

Solving for $\Delta\xi$ yields Gauss-Newton Equation

Evaluation of Gauss-Newton equation gives a step $\Delta\xi$ that minimizes the objective function

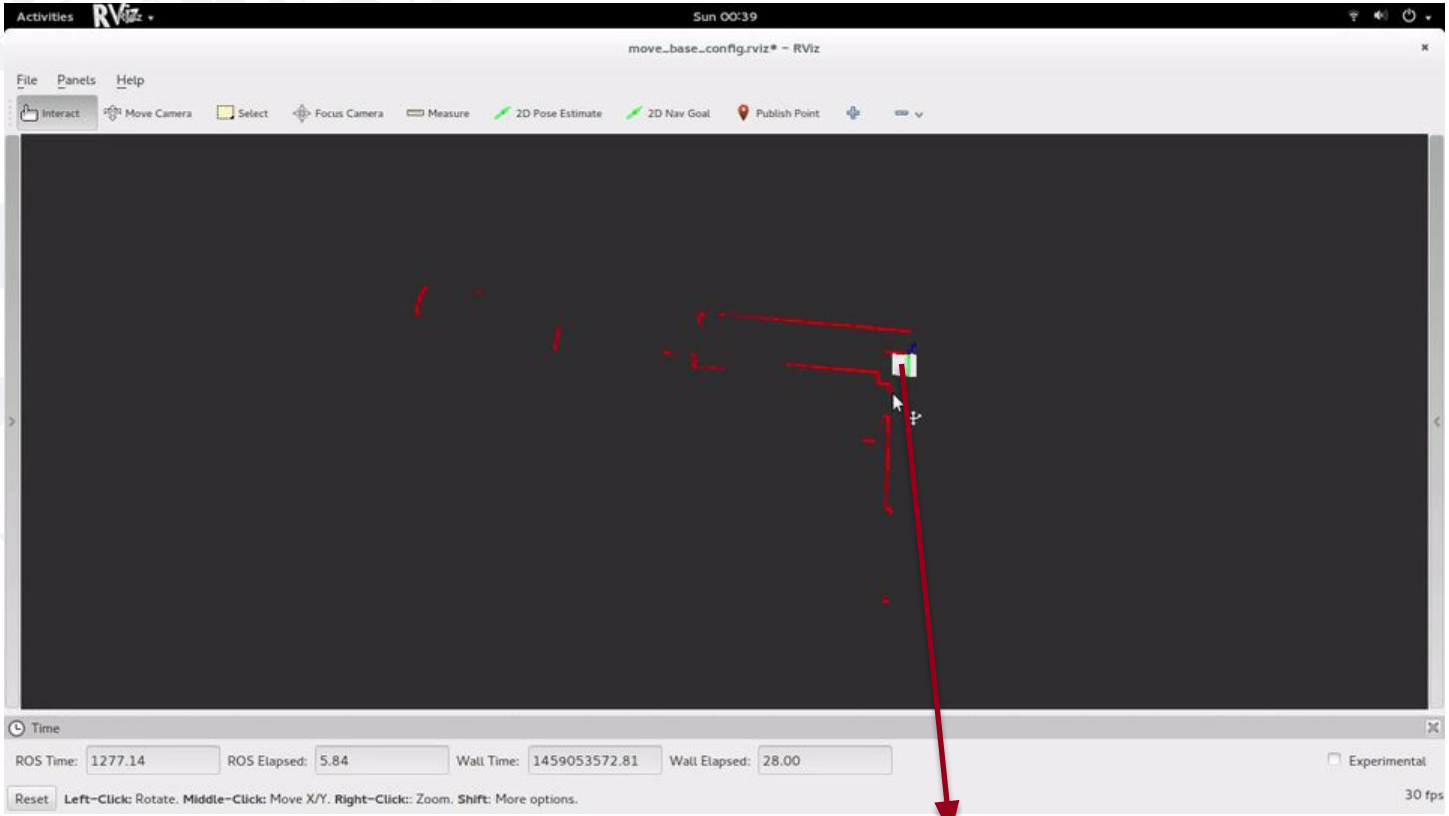
Raw LiDAR Scans



Baseframe Axes

Penn Engineering

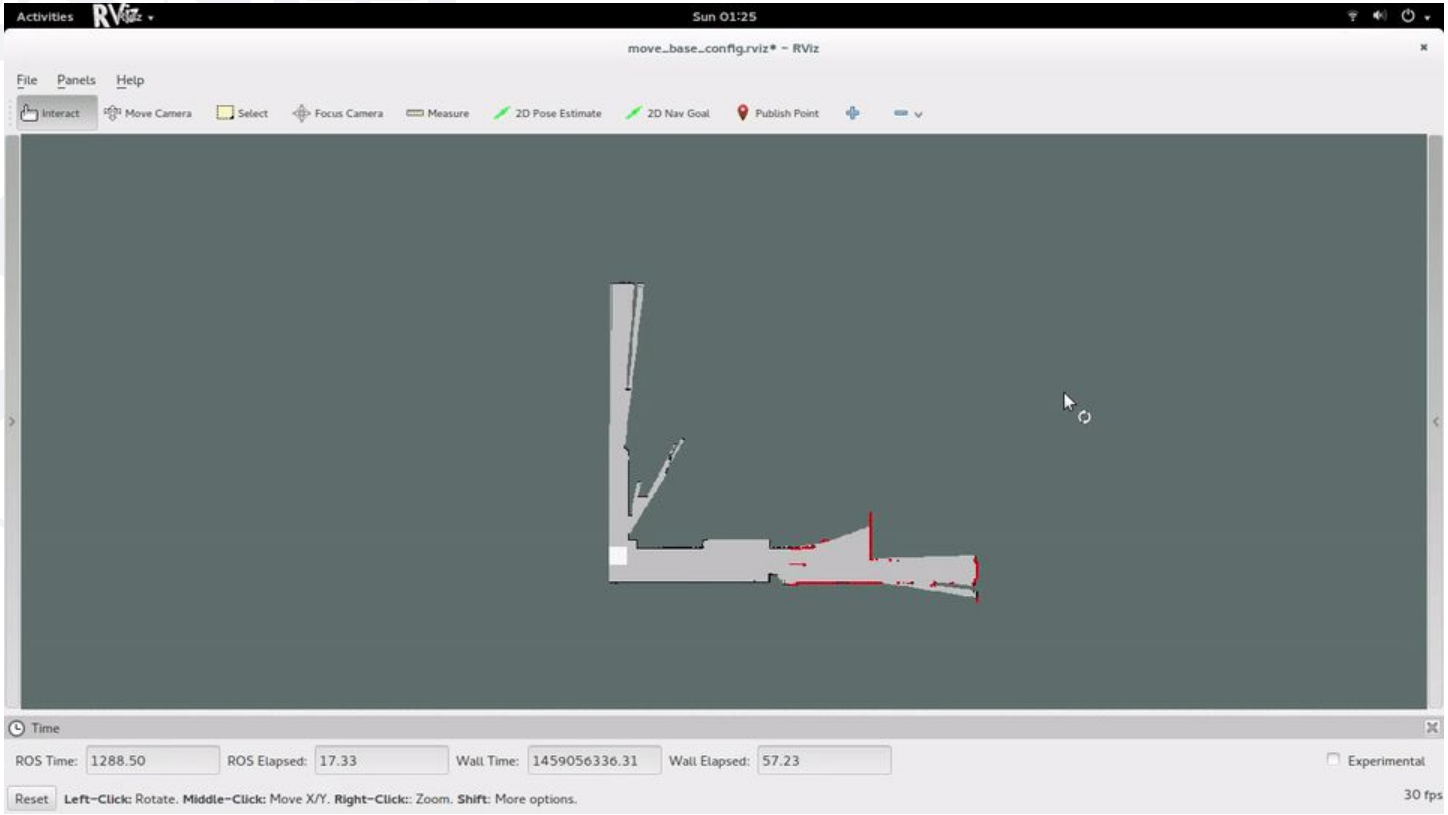
Scans after transforming by $\Delta\xi$ at each stage



Mapframe Axes

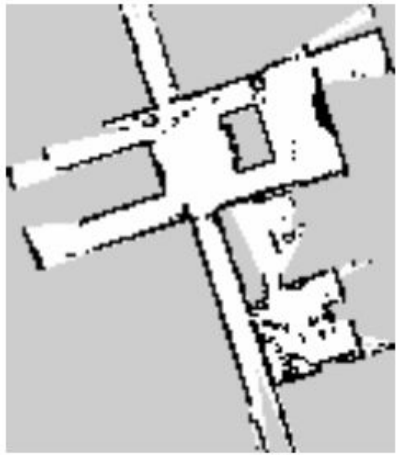
Penn Engineering

Map Update

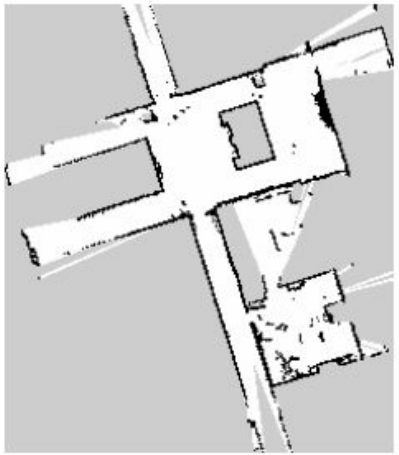


Penn Engineering

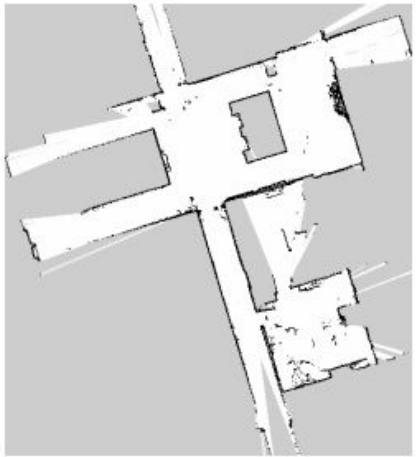
Multi-Resolution Map Representation



20 cm Grid Cell

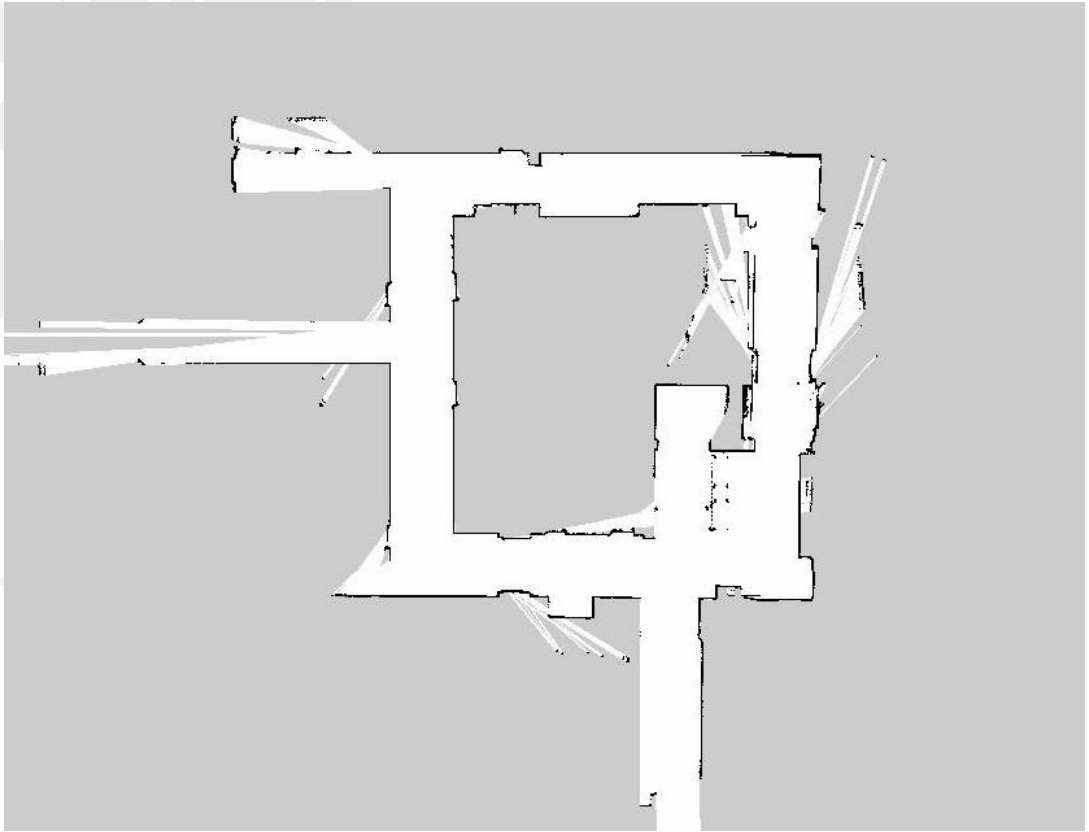


10 cm Grid Cell



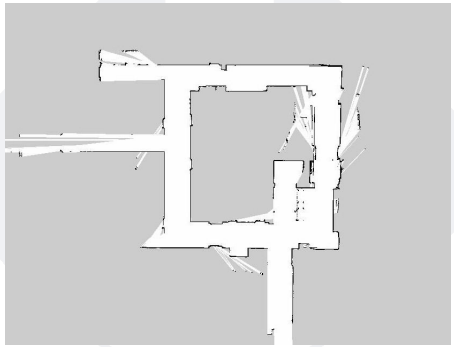
5 cm Grid Cell

Saving the map



Penn Engineering

Saving the map



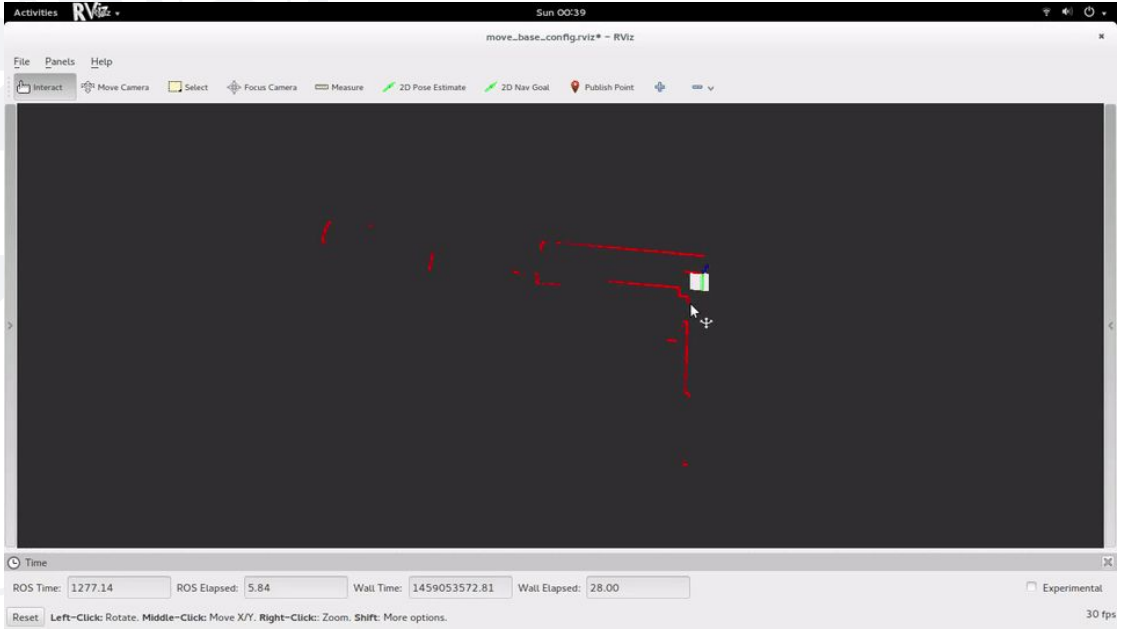
- ROS Package called **MAP Server**
- Allows saving a map currently being published over/map topic
- Save the map:

```
rosrun map_server map_saver [-f mapname]
```

- Load the map:

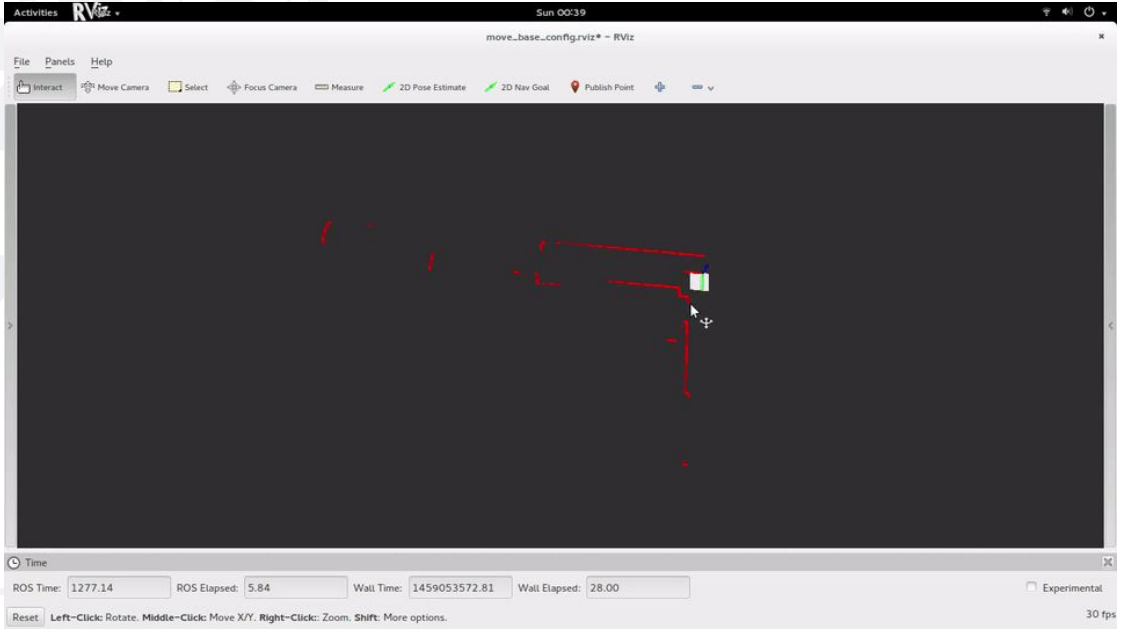
```
rosrun map_server map_server <name.yaml>
```

Odometry using Hector Mapping



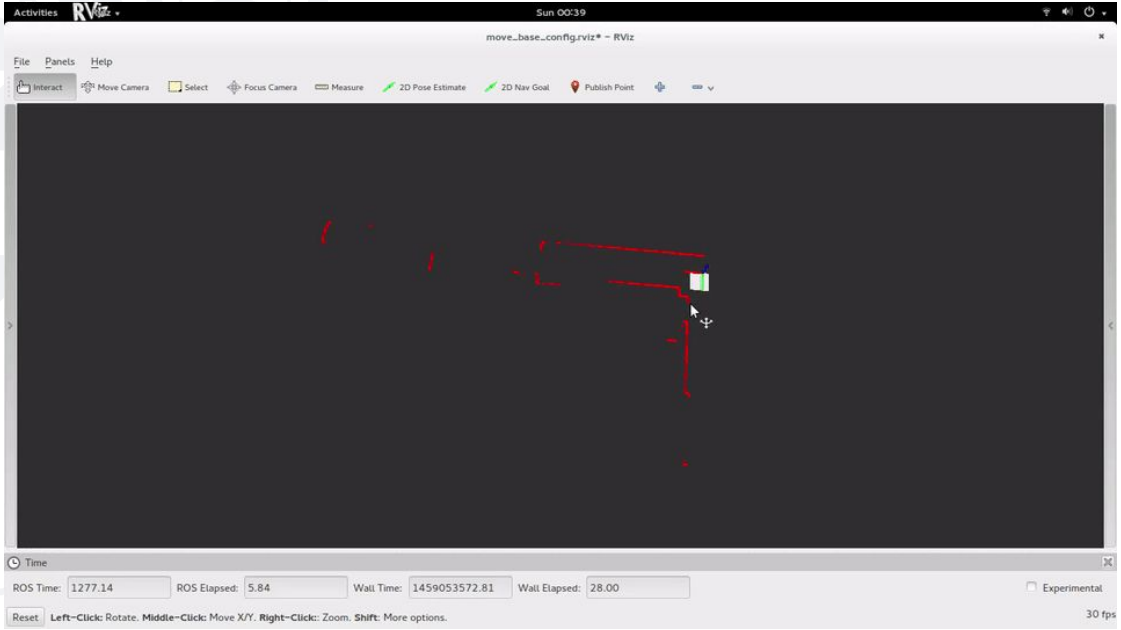
Penn Engineering

Odometry using Hector Mapping



- Using Hector Slam for measuring $\Delta\xi$, while discarding the map

Odometry using Hector Mapping

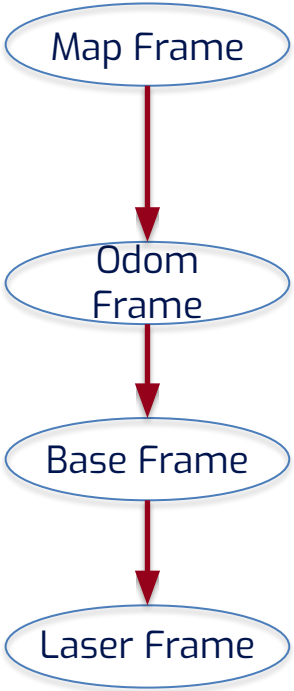


- Using Hector Slam for measuring $\Delta\xi$, while discarding the map

- Optional Approach : CSM (Canonical Scan Matcher) by Andrea Censi
  - Scan matching between 2 scans

Penn Engineering

System Tf tree



Tf Provided by Hector odometry

Tf required by Hector package

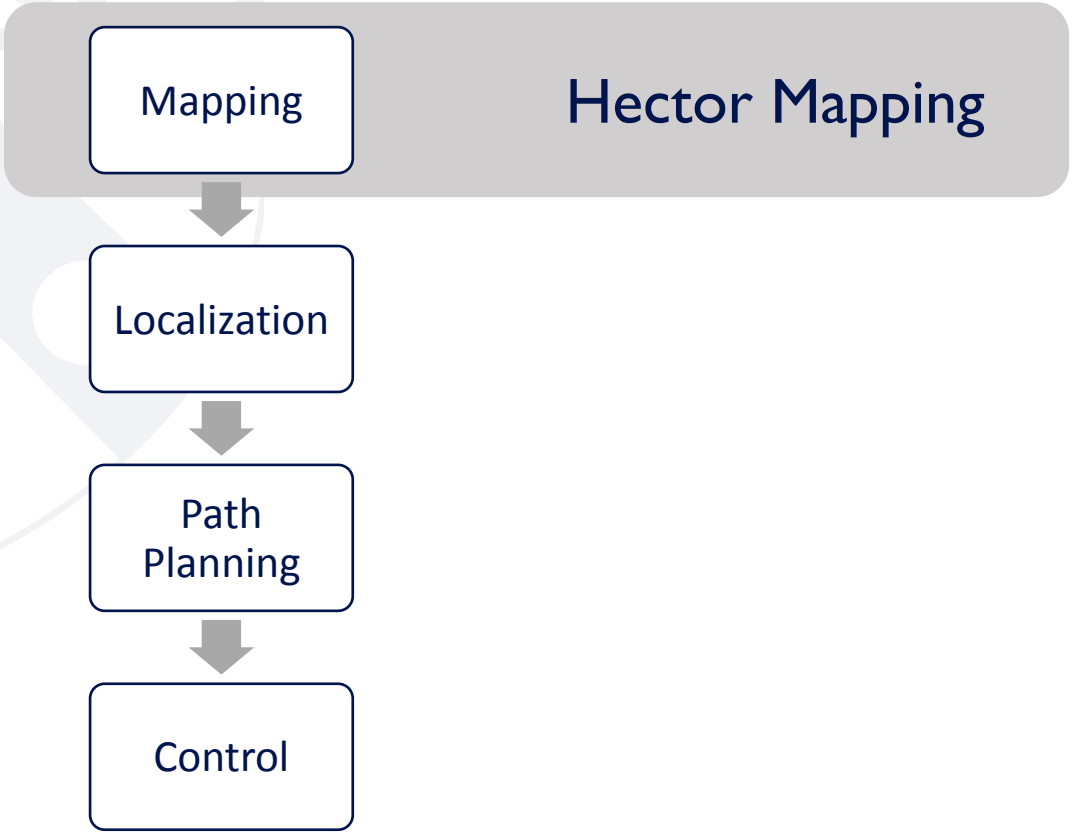
Parameters for Hector SLAM : ROS

- map resolution - Grid resolution
- map_update_distance_thresh - minimum distance to be travelled before having a map update
- map_update_angle_thresh - minimum angle to be travelled before a map update
- laser_max_dist - Laser sensor specification
- update_factor_free - Log odds probability for occupied cells
- update_factor_occupied - Log odds probability for free cells

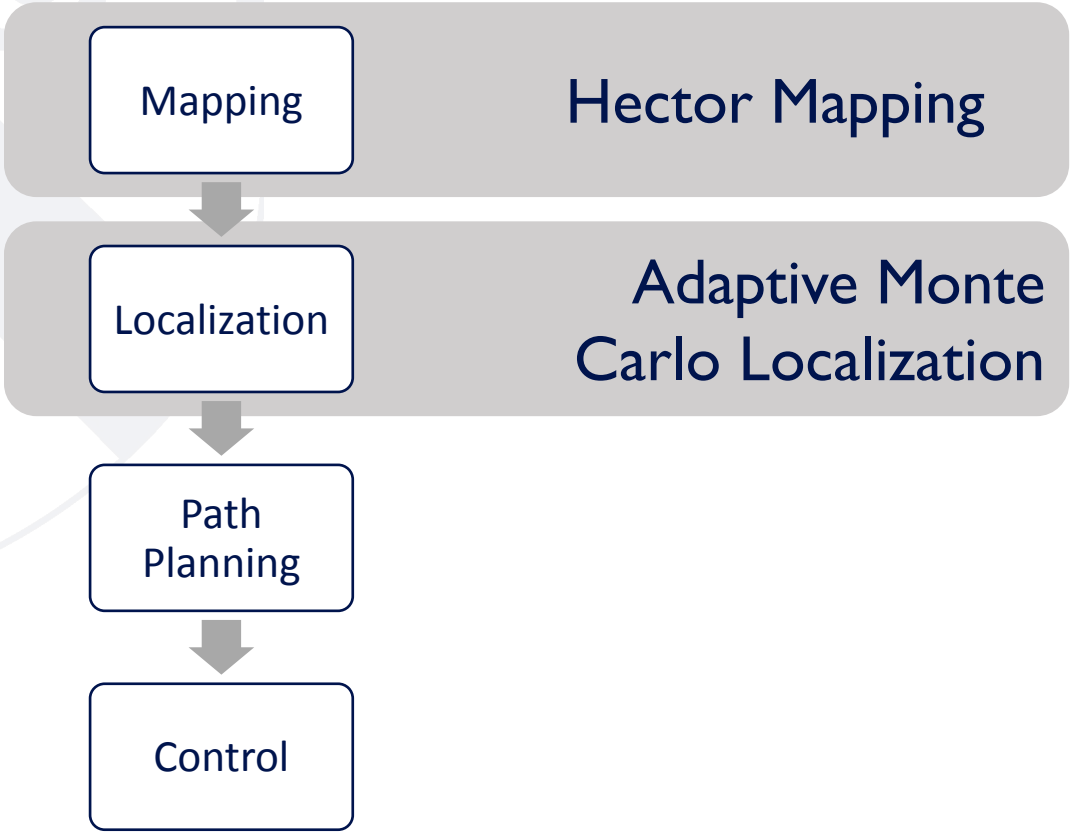
Next up!

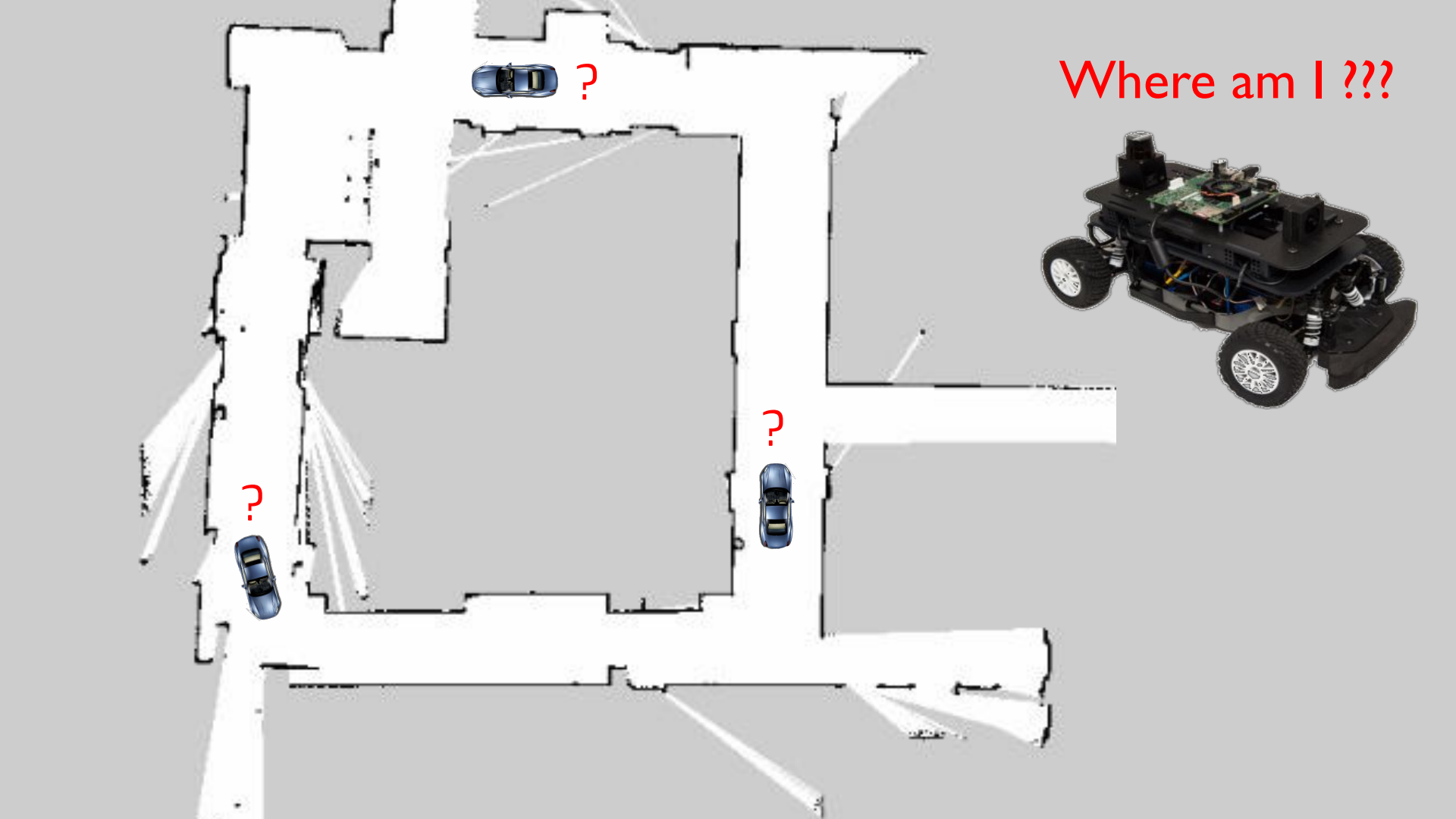
1. Using the map generated today
2. Localizing using Adaptive Monte Carlo localization (AMCL)
3. Integrating Hector odometry and AMCL

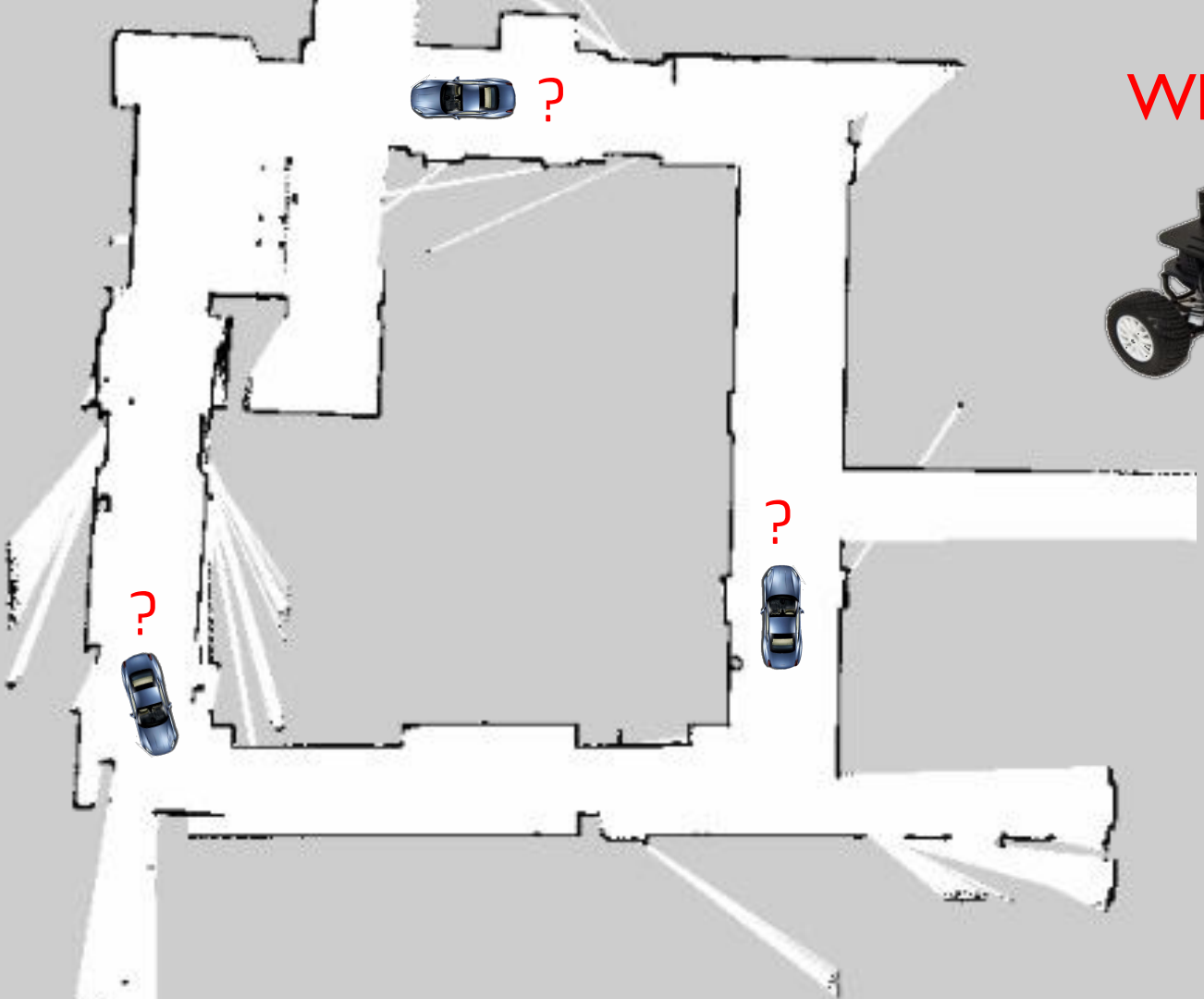
System Overview



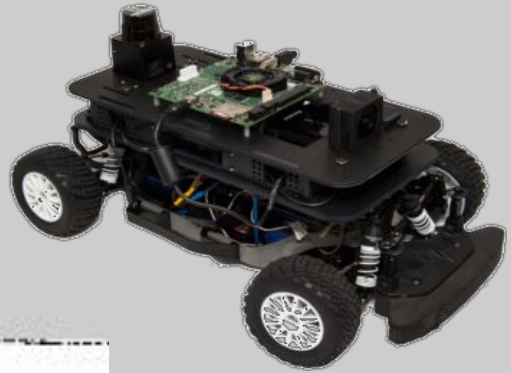
System Overview




Where am I ???
?
?
?



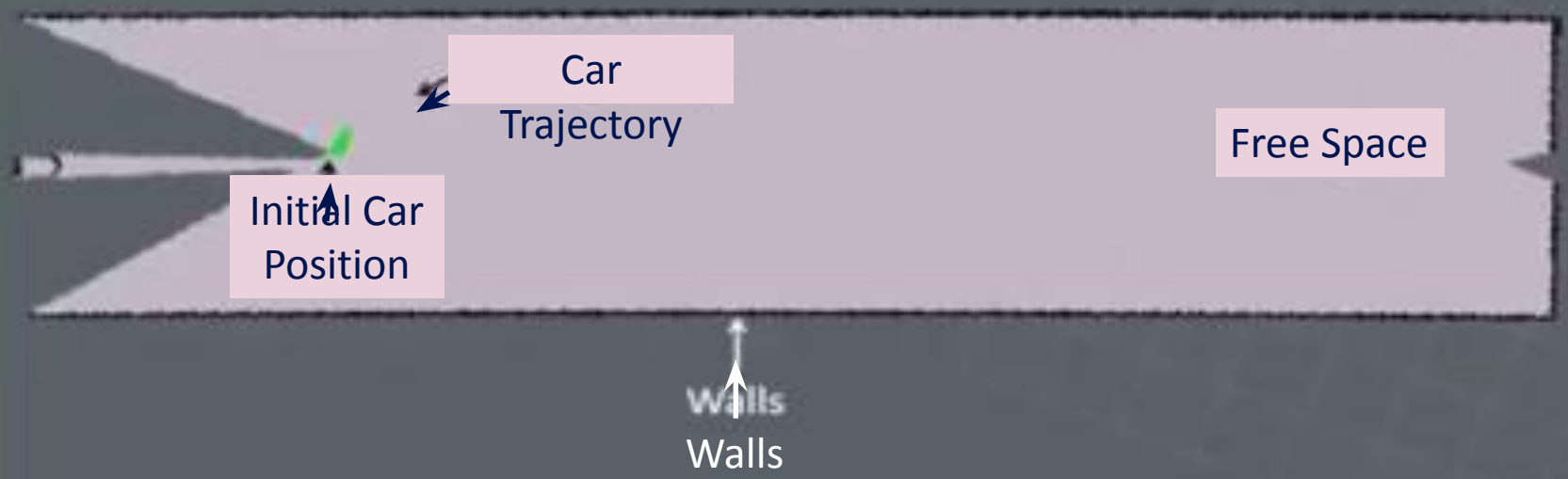
Where am I ???



Position &
Orientation

Localization using Odometry

Localization using Odometry

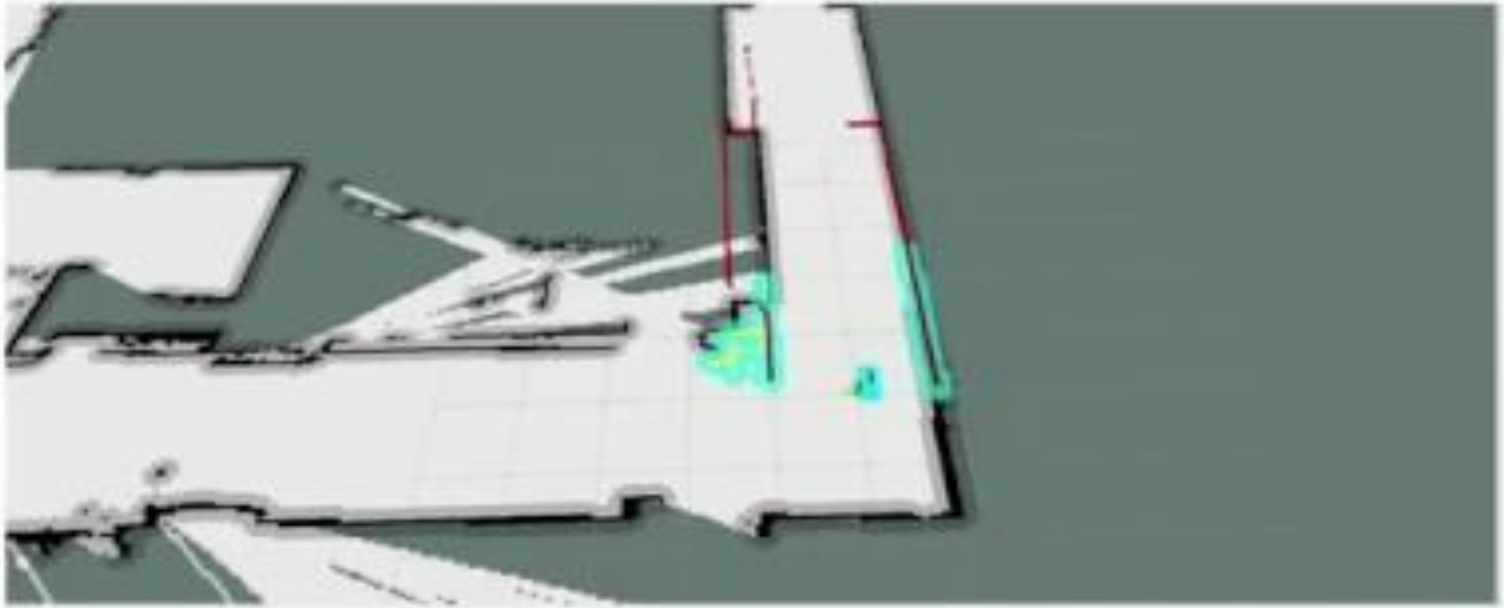


Drawbacks of Localization using Wheel Odometry

Wheel spin due to lack of traction

Drawbacks of Localization using Hector odometry

Failed scan matching due to lack of features



Issue

- A mechanism to compensate the mistakes committed by odometry
- A solution robust to compensate for lack of information on initial position

Issue

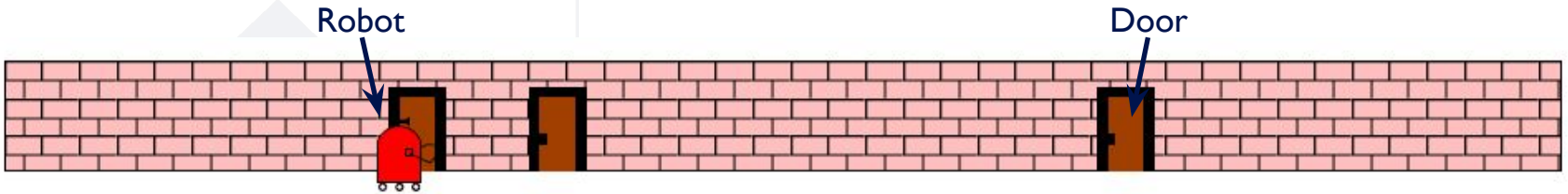
- A mechanism to compensate the mistakes committed by odometry
- A solution robust to compensate for lack of information on initial position

Solution: Monte Carlo Localization

Alternate Solutions: Kalman Filter, Topological Markov Localization

Particle Filter: Intuition

A Example in 1 Dimension

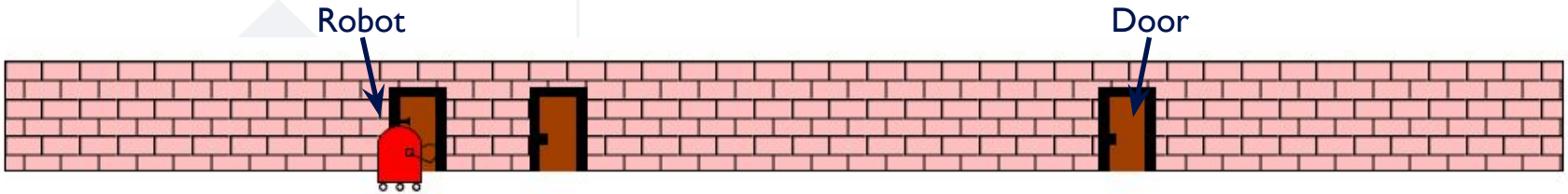


Penn Engineering

Courtesy: Thrun, Burgard, Fox

Particle Filter

A Example in 1 Dimension

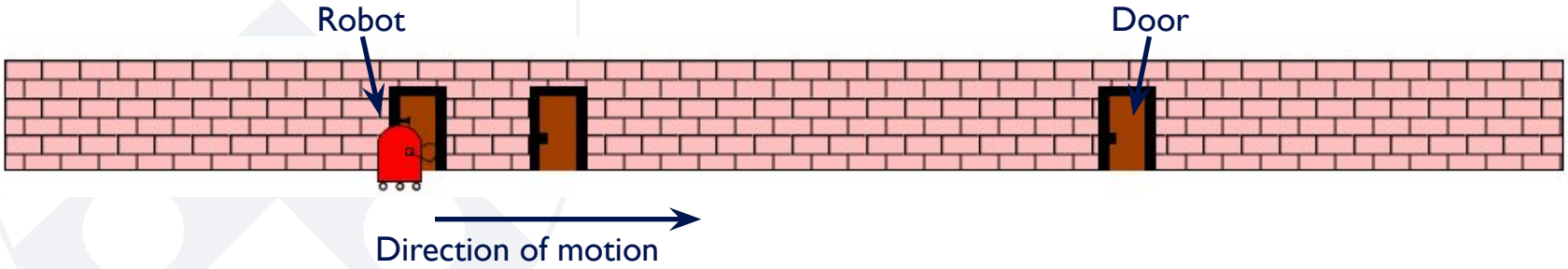


Belief State

Particle Filter

At time t = 1

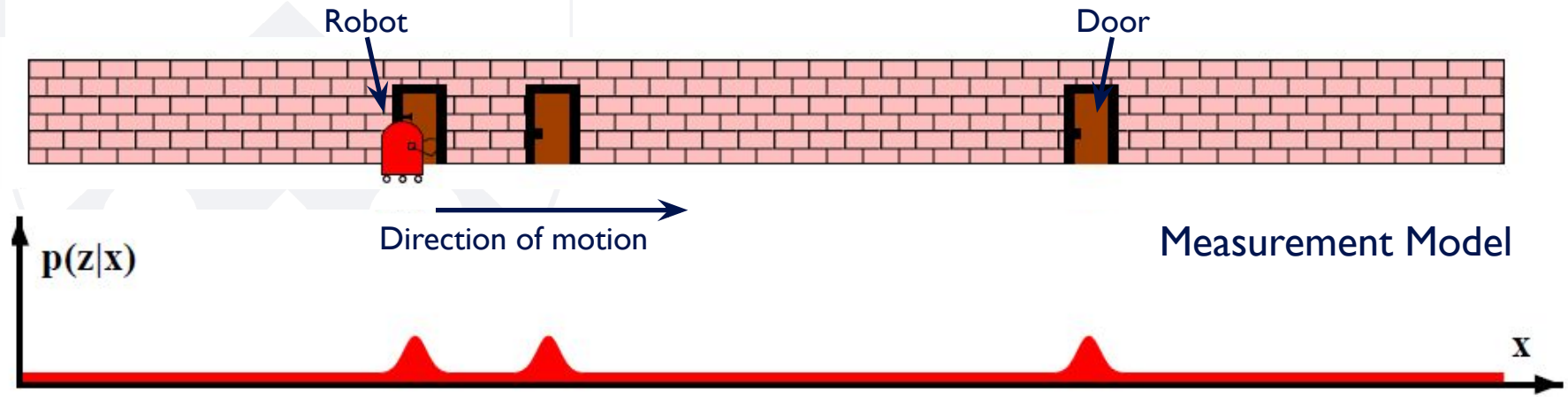
A Example in 1 Dimension



Particle Filter

At time t = 1

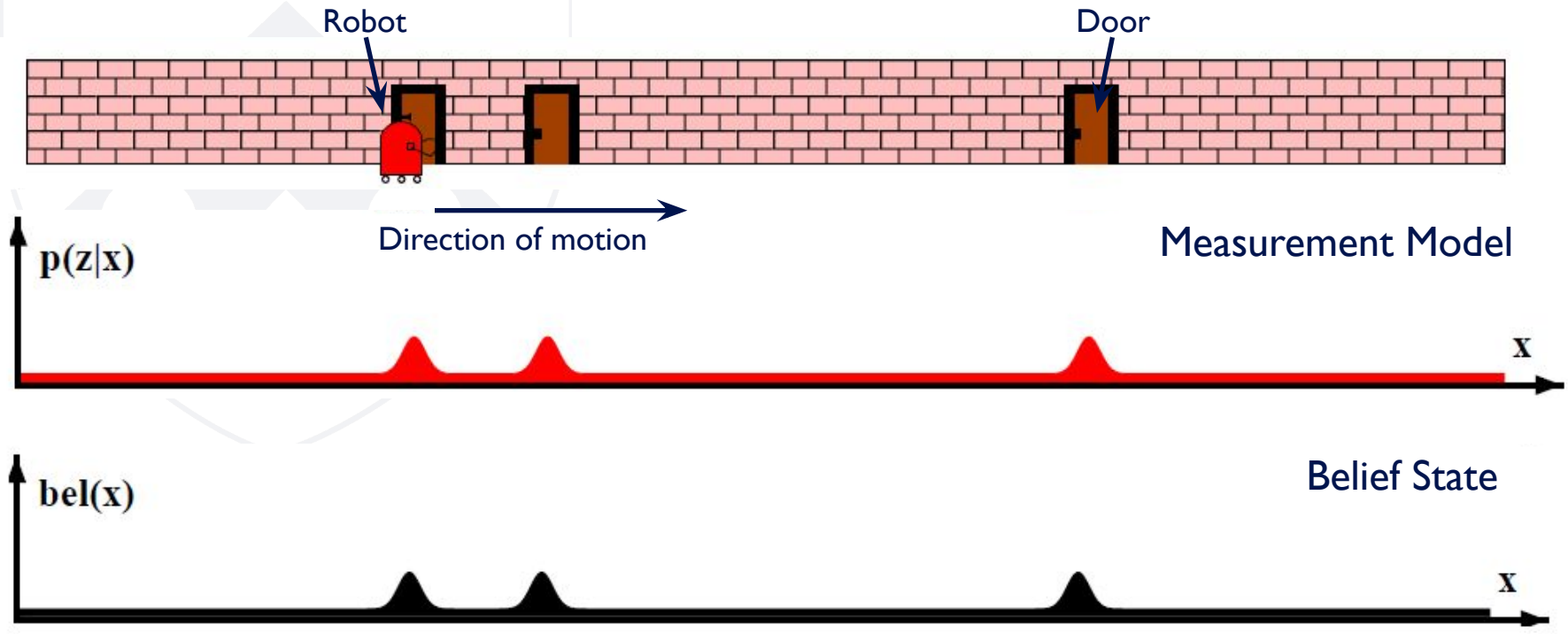
A Example in 1 Dimension



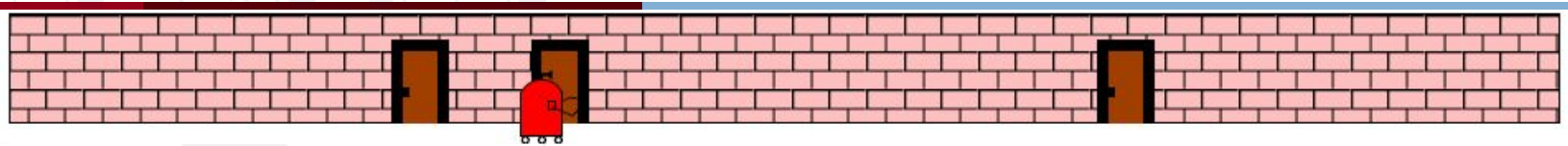
Particle Filter

At time t = 1

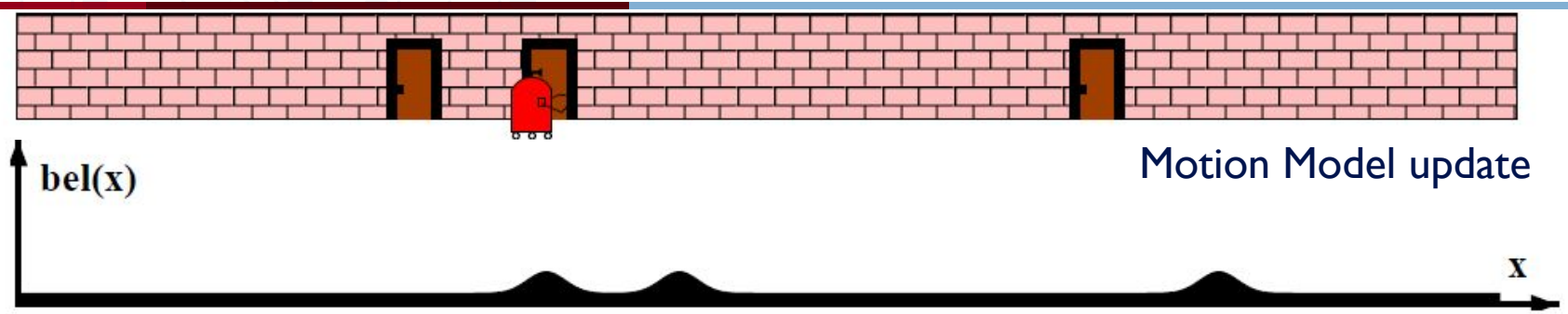
A Example in 1 Dimension



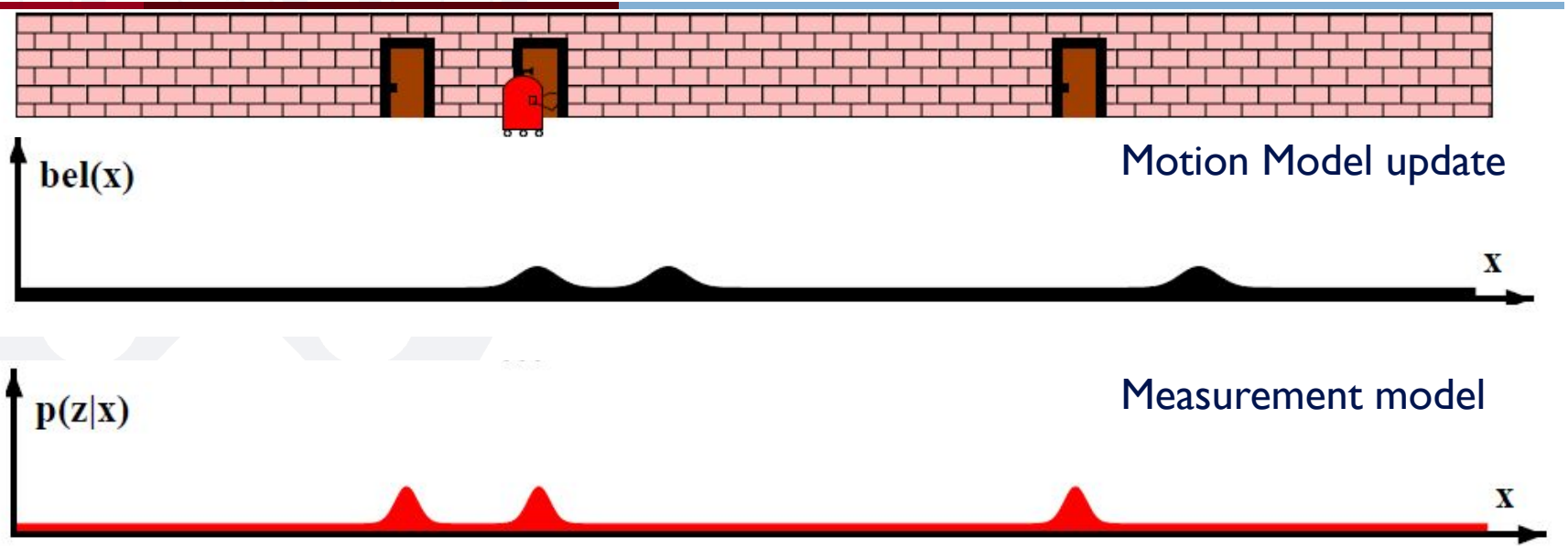
At time t = 2, robot moves forward a certain distance



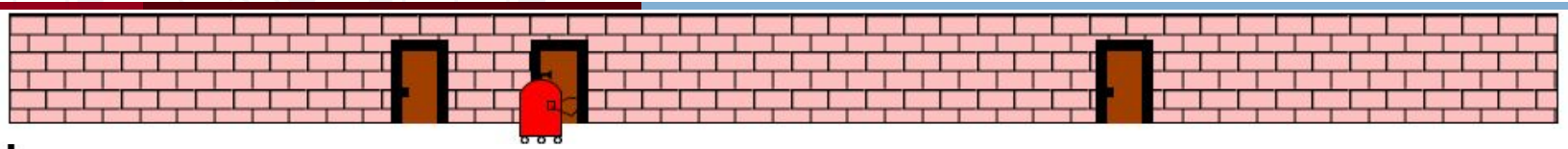
At time t = 2, robot moves forward a certain distance



At time t = 2, robot moves forward a certain distance



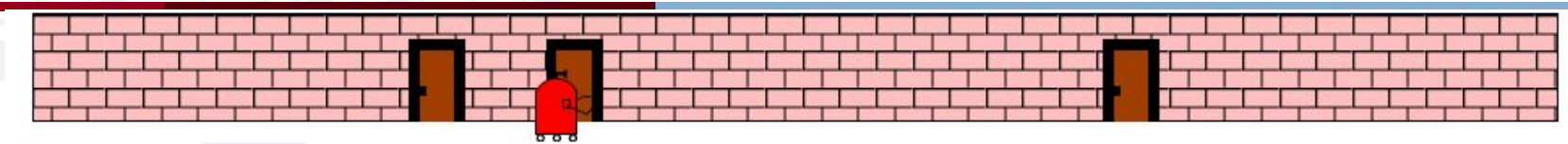
At time t = 2, robot moves forward a certain distance



Motion Model update

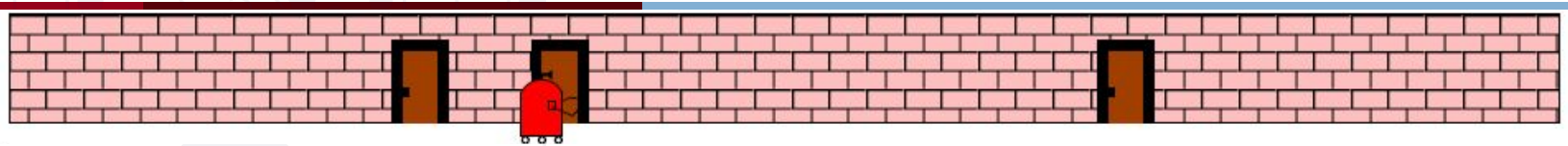
Measurement model

Belief State



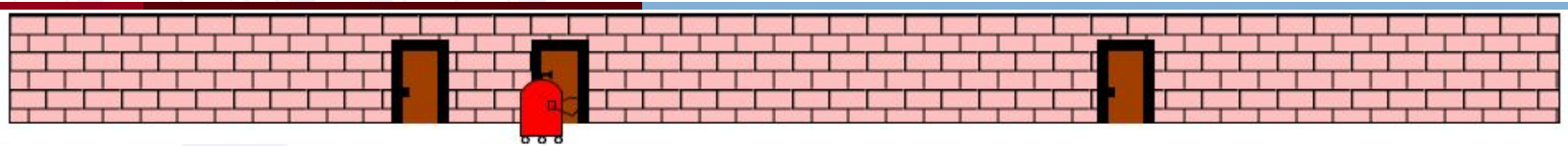
Continuous State

After matching the probability in two states the probability there is a door in the second bump has increased and the probability that the robot is at the door is increased



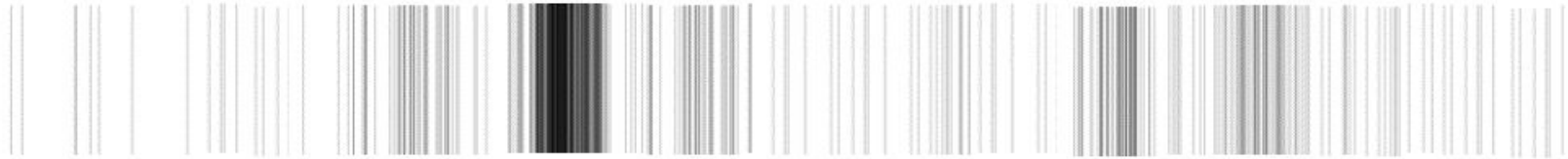
Continuous State

Discrete State

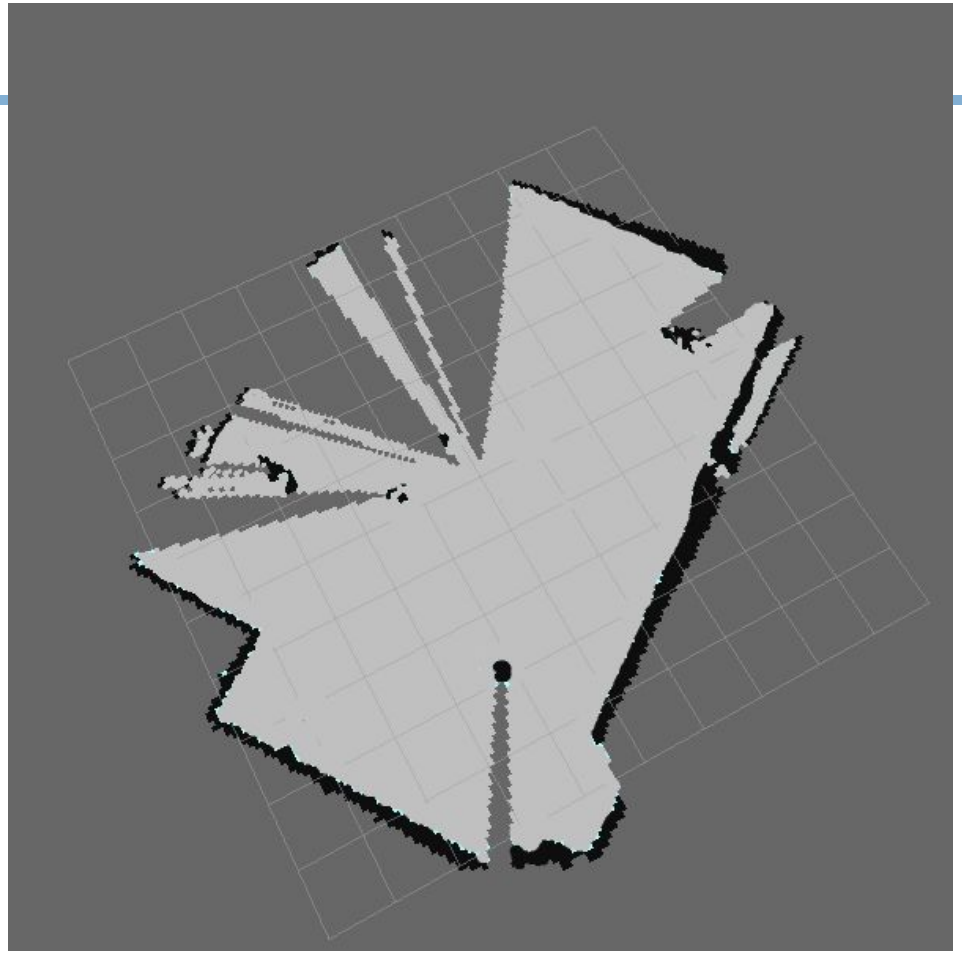


Continuous State

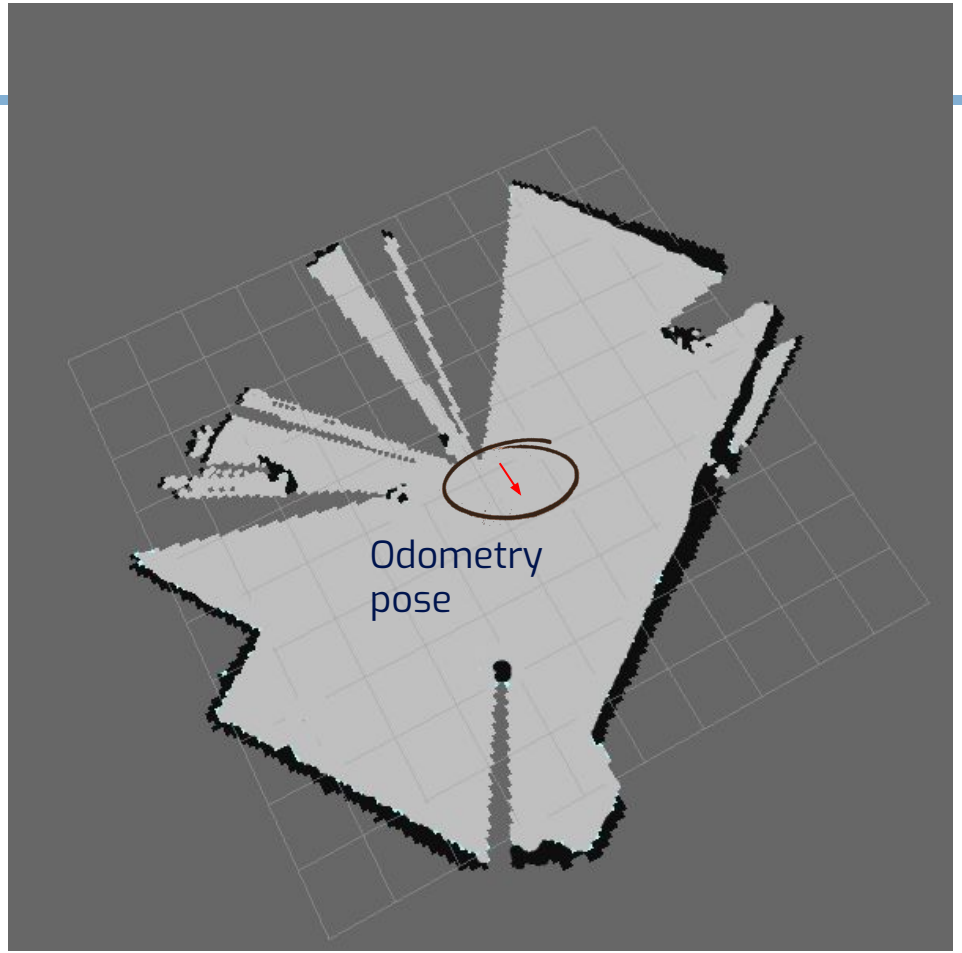
Discrete State



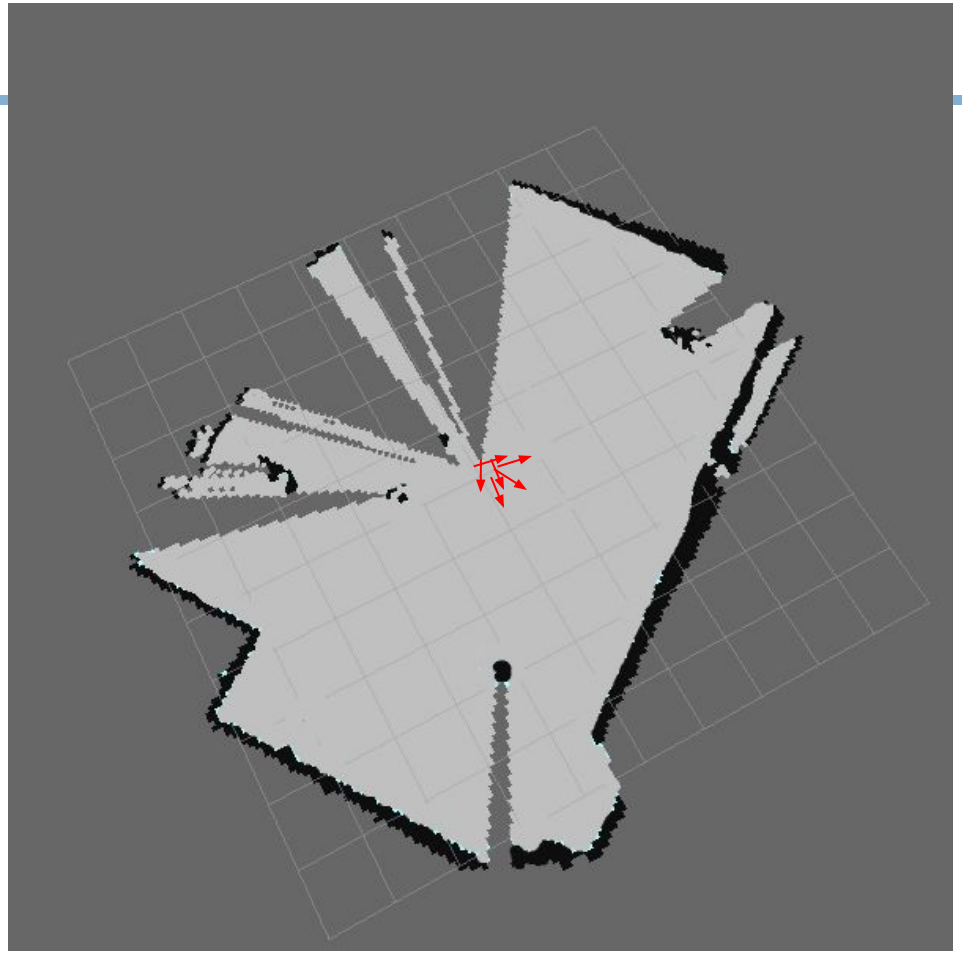
Particle Filter in 2D



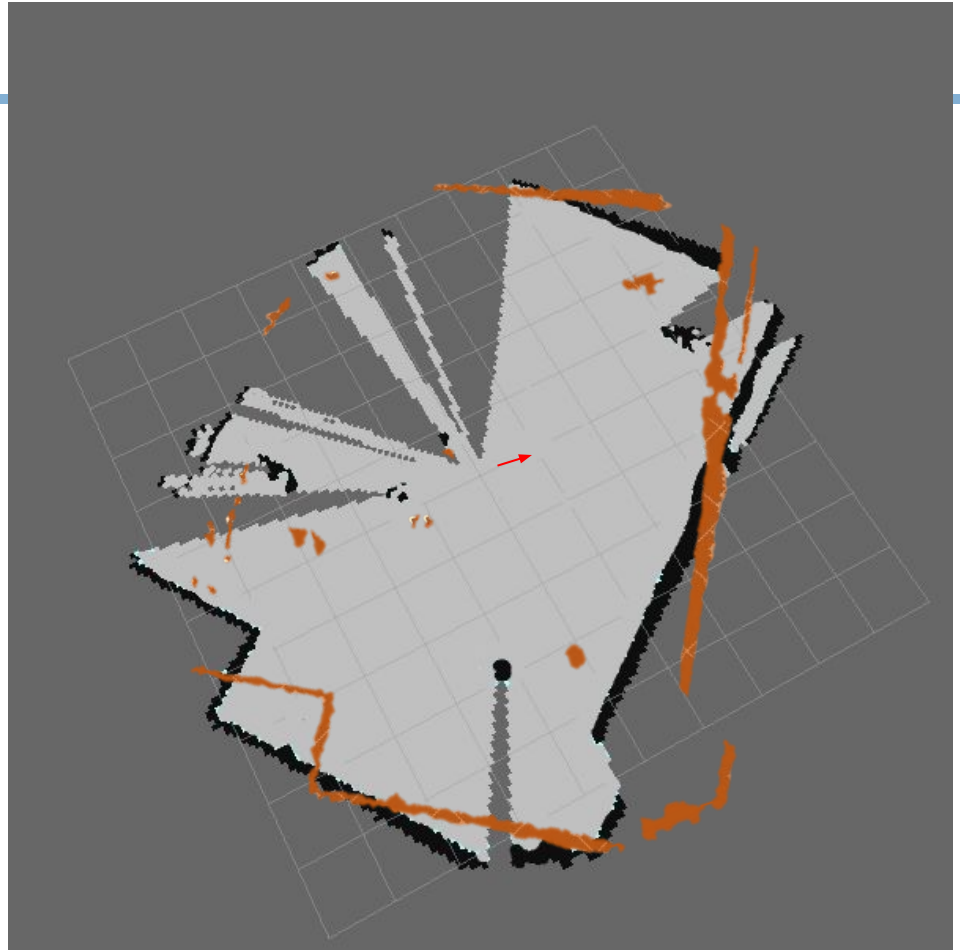
Particle Filter in 2D



Particle Filter in 2D

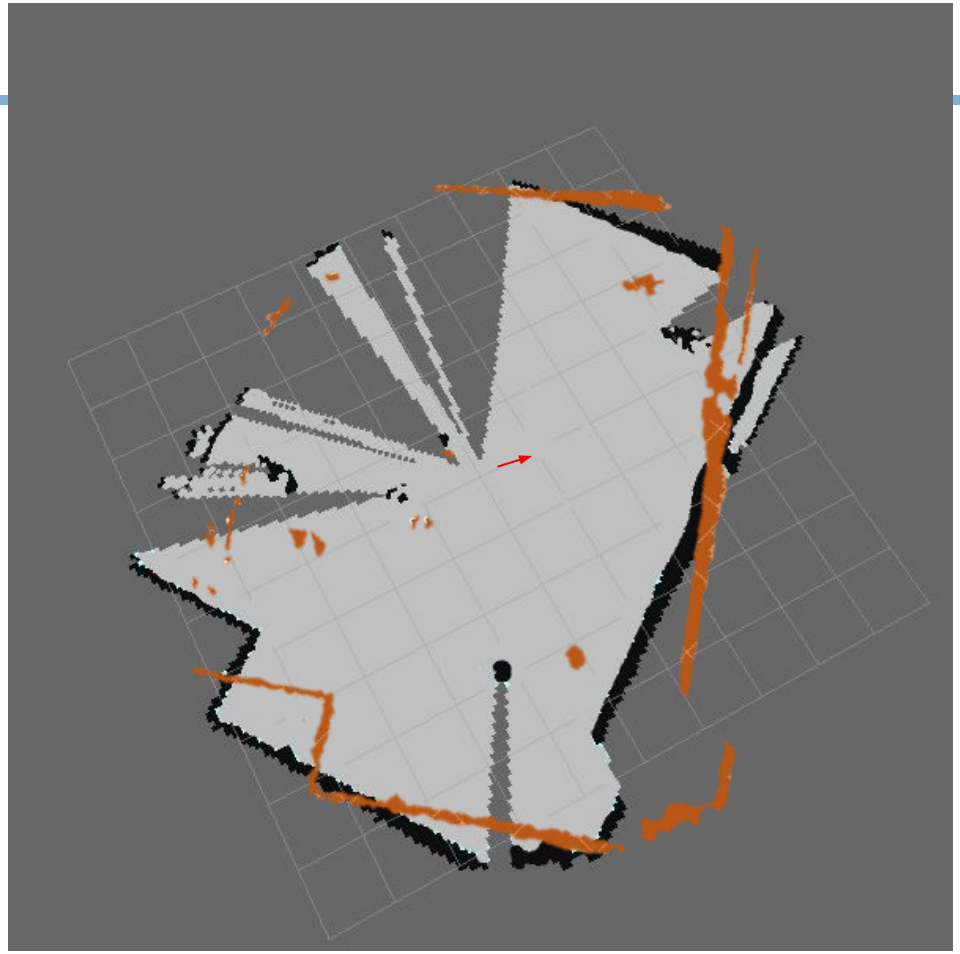


Scan Correlation



Scan Correlation

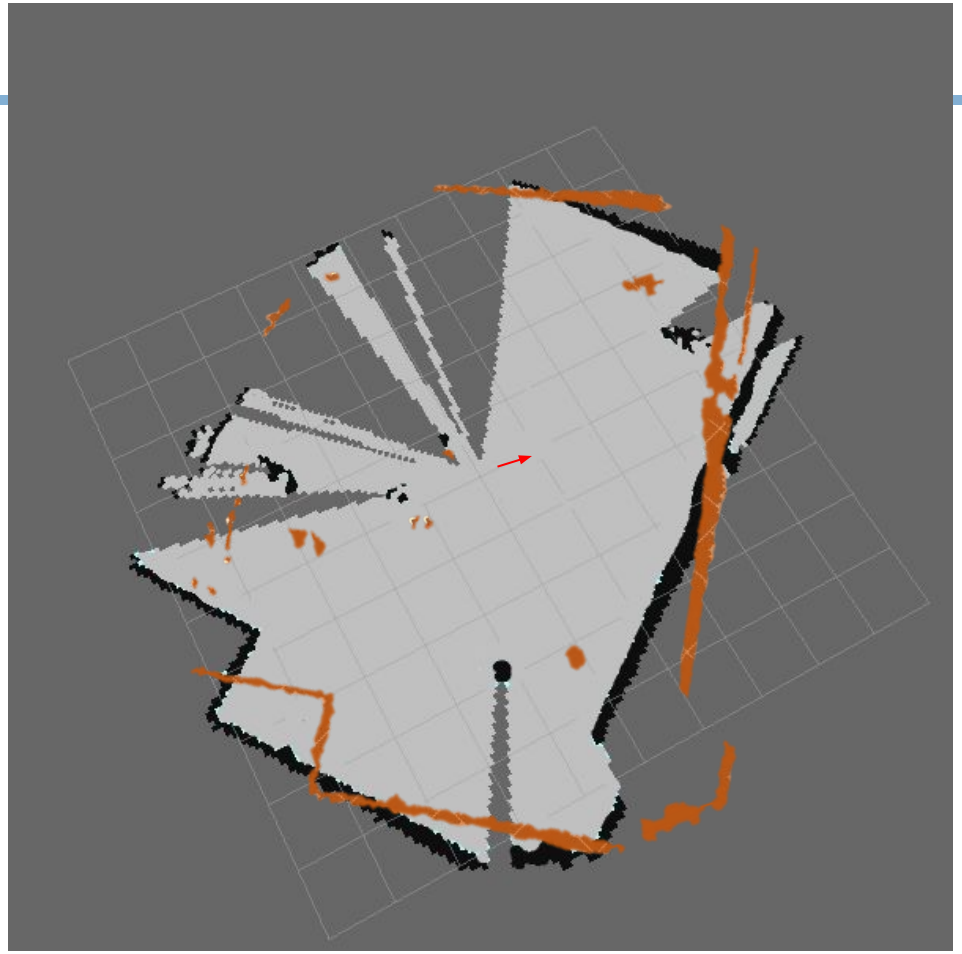
$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$



Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

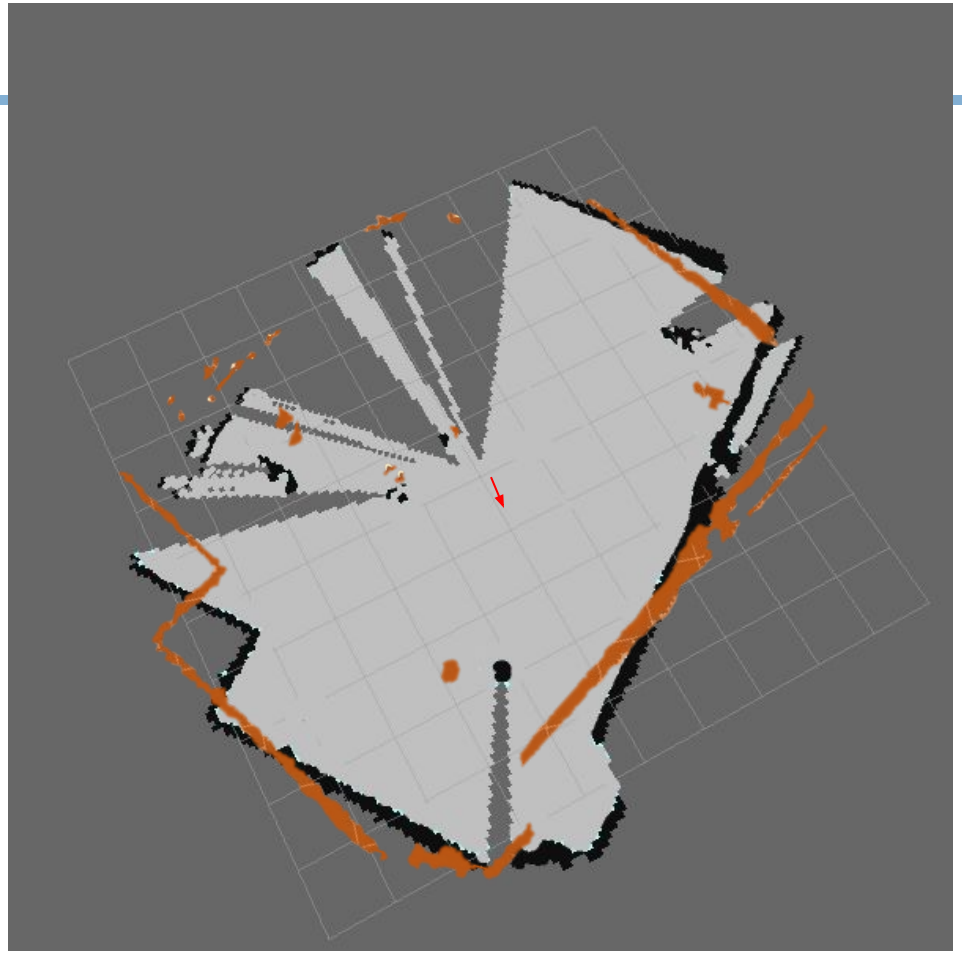
Particle	Weight
Particle 1	S_1



Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

Particle	Weight
Particle 1	S_1
Particle 2	S_2

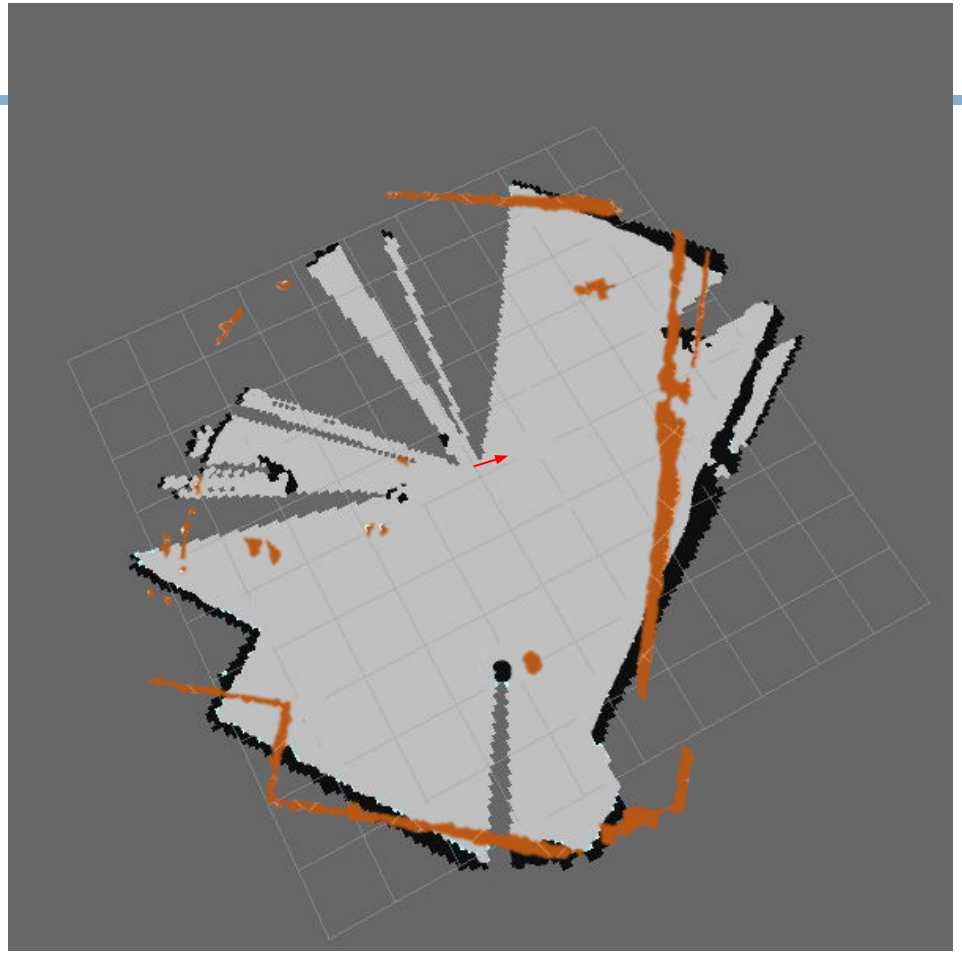


Penn Engineering

Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

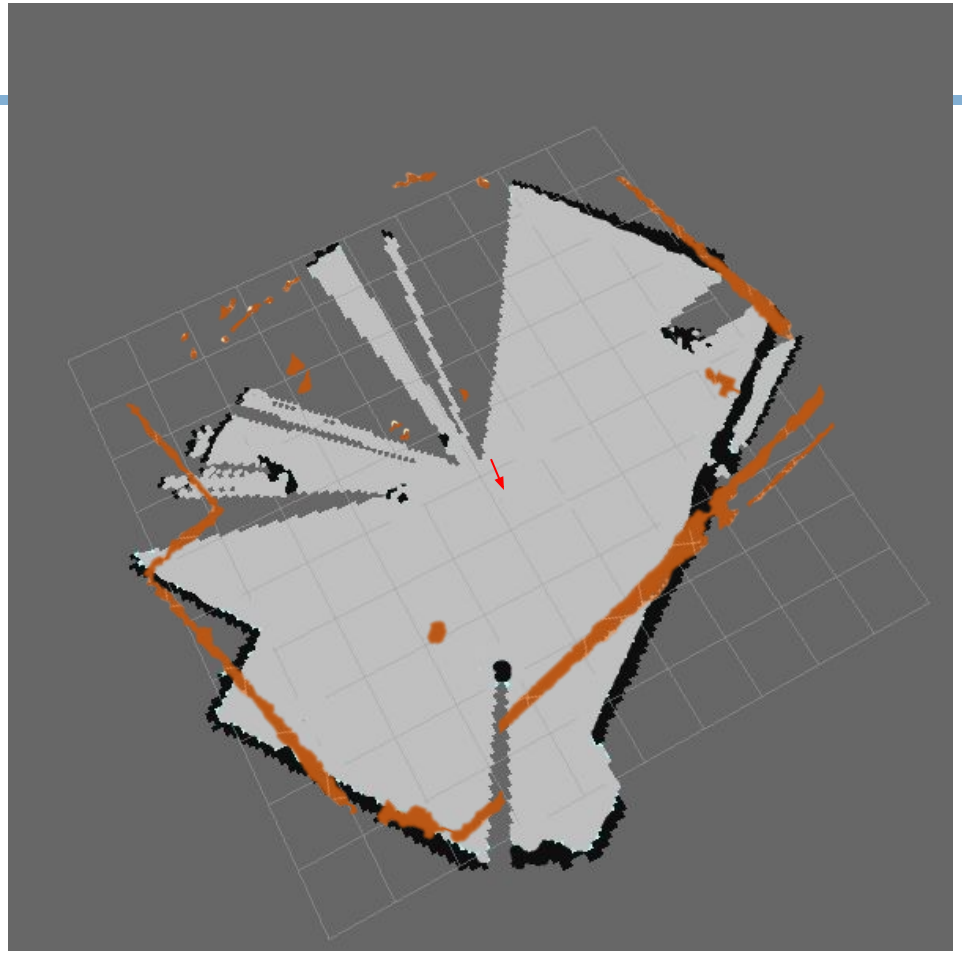
Particle	Weight
Particle 1	S_1
Particle 2	S_2
Particle 3	S_3



Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

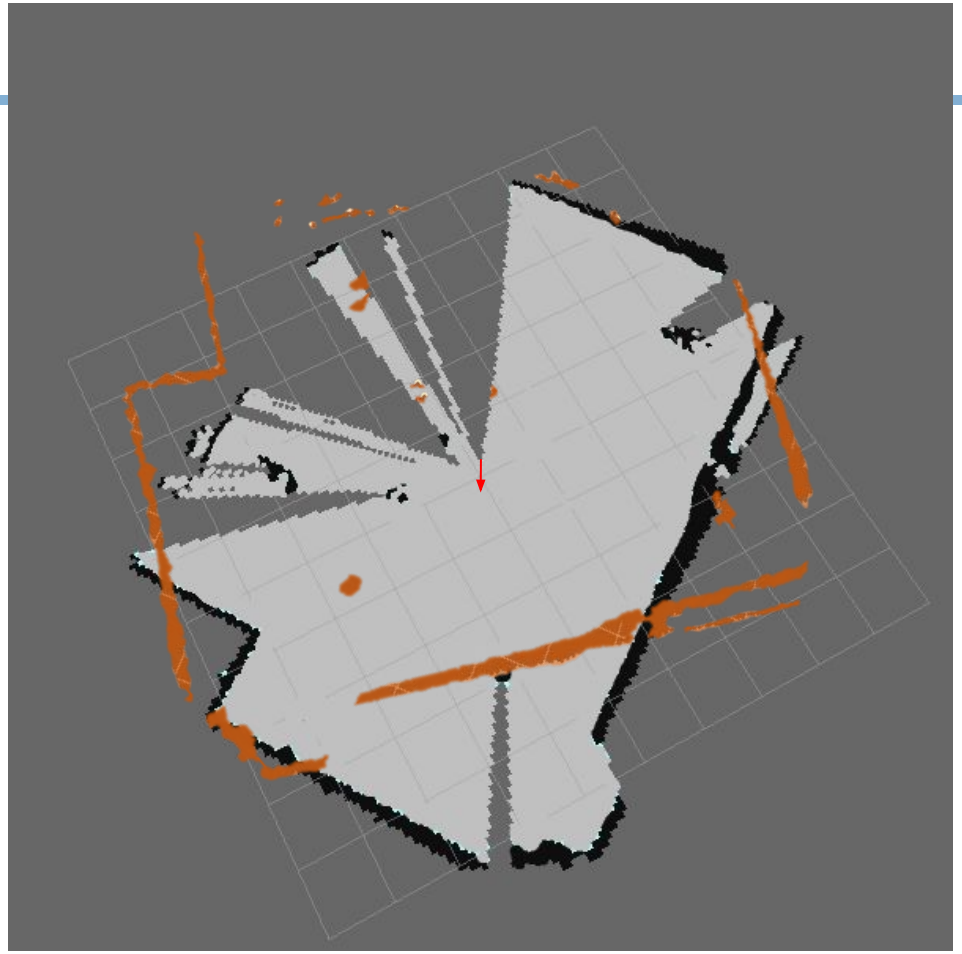
Particle	Weight
Particle 1	S_1
Particle 2	S_2
Particle 3	S_3
Particle 4	S_4



Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

Particle	Weight
Particle 1	S_1
Particle 2	S_2
Particle 3	S_3
Particle 4	S_4
Particle 5	S_5

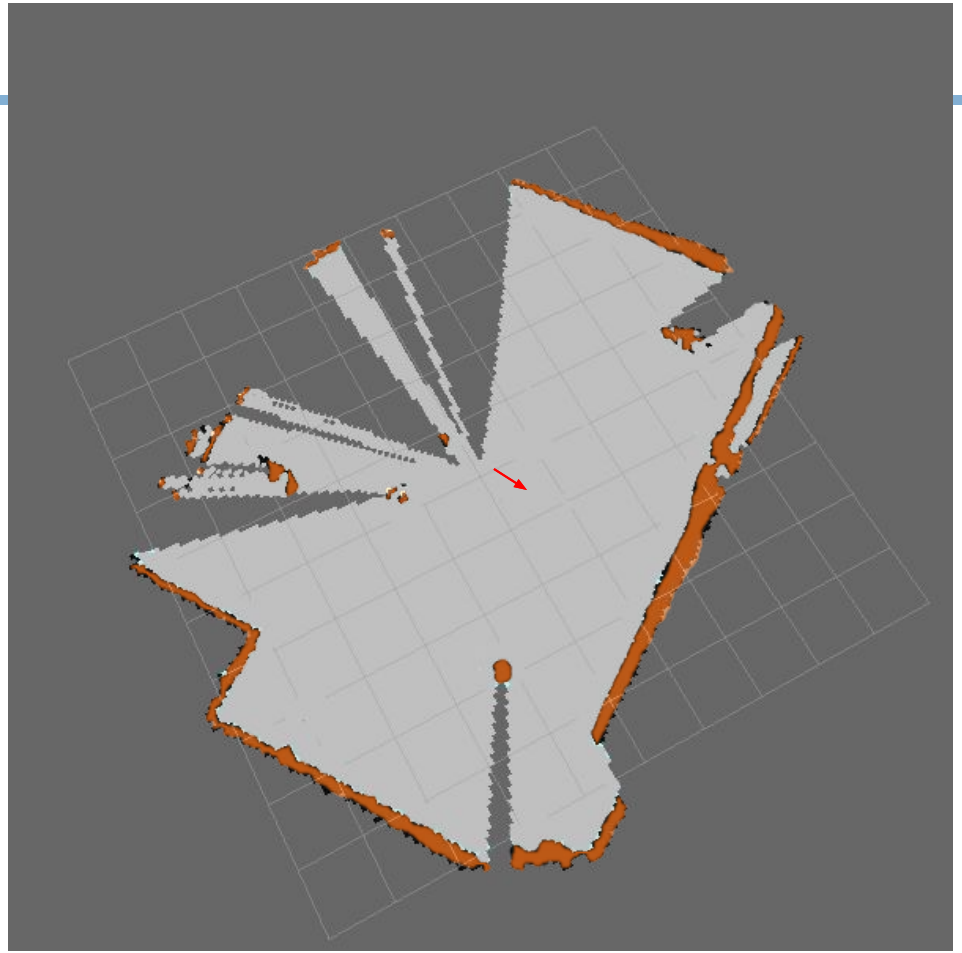


Penn Engineering

Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

Particle	Weight
Particle 1	S_1
Particle 2	S_2
Particle 3	S_3
Particle 4	S_4
Particle 5	S_5
Particle 6	S_6

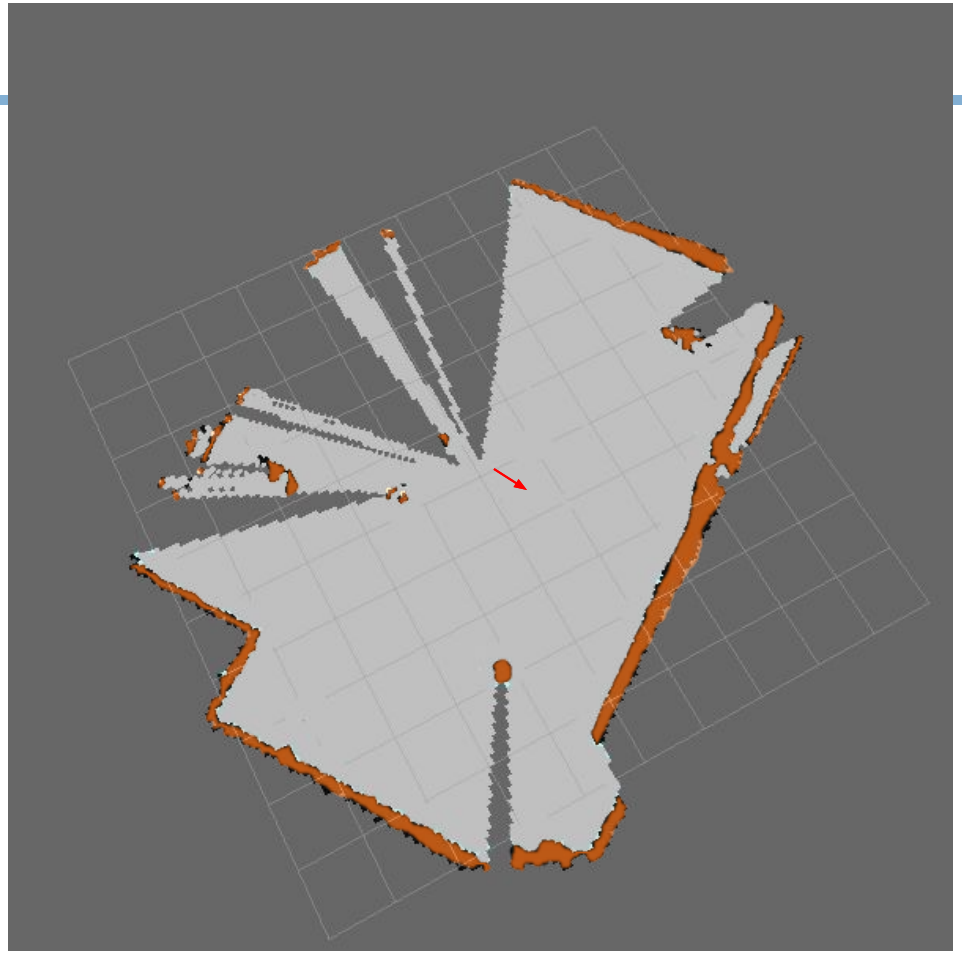


Penn Engineering

Scan Correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

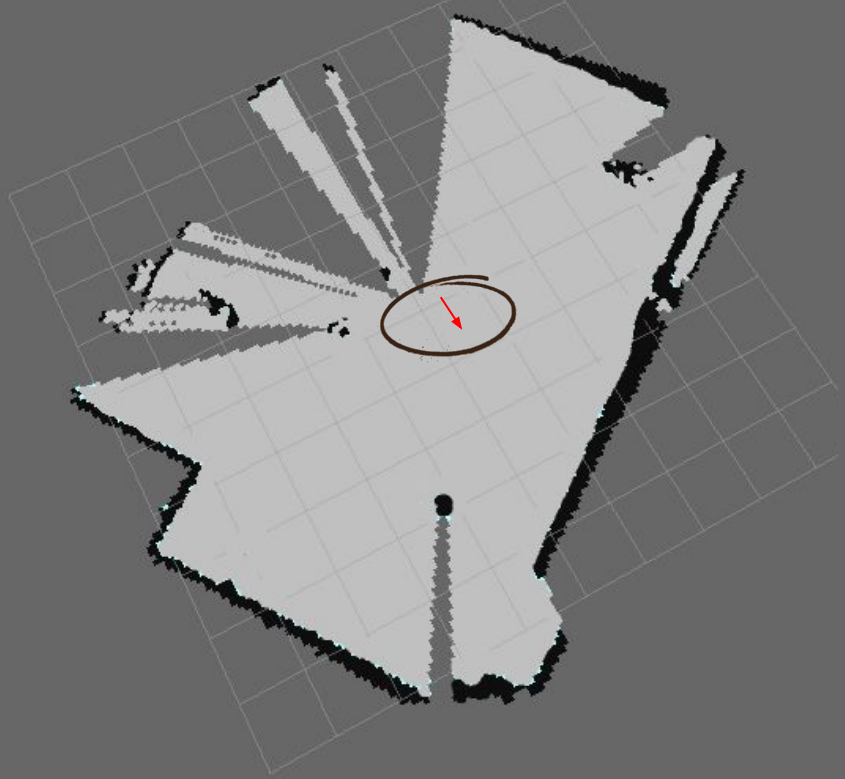
Particle	Weight
Particle 1	S_1
Particle 2	S_2
Particle 3	S_3
Particle 4	S_4
Particle 5	S_5
Particle 6	S_6



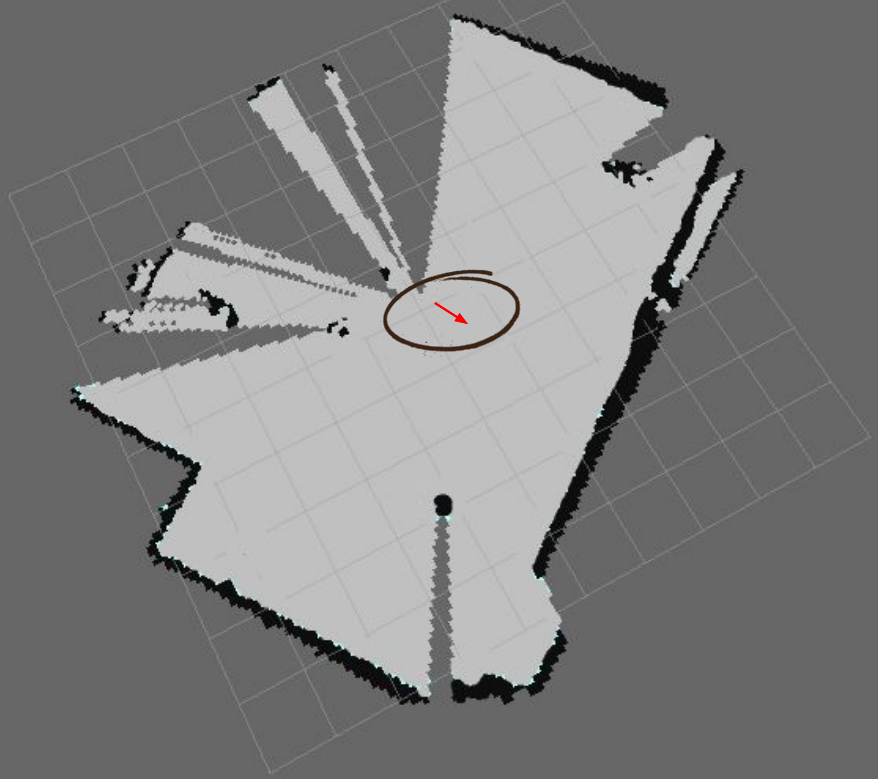
Penn Engineering

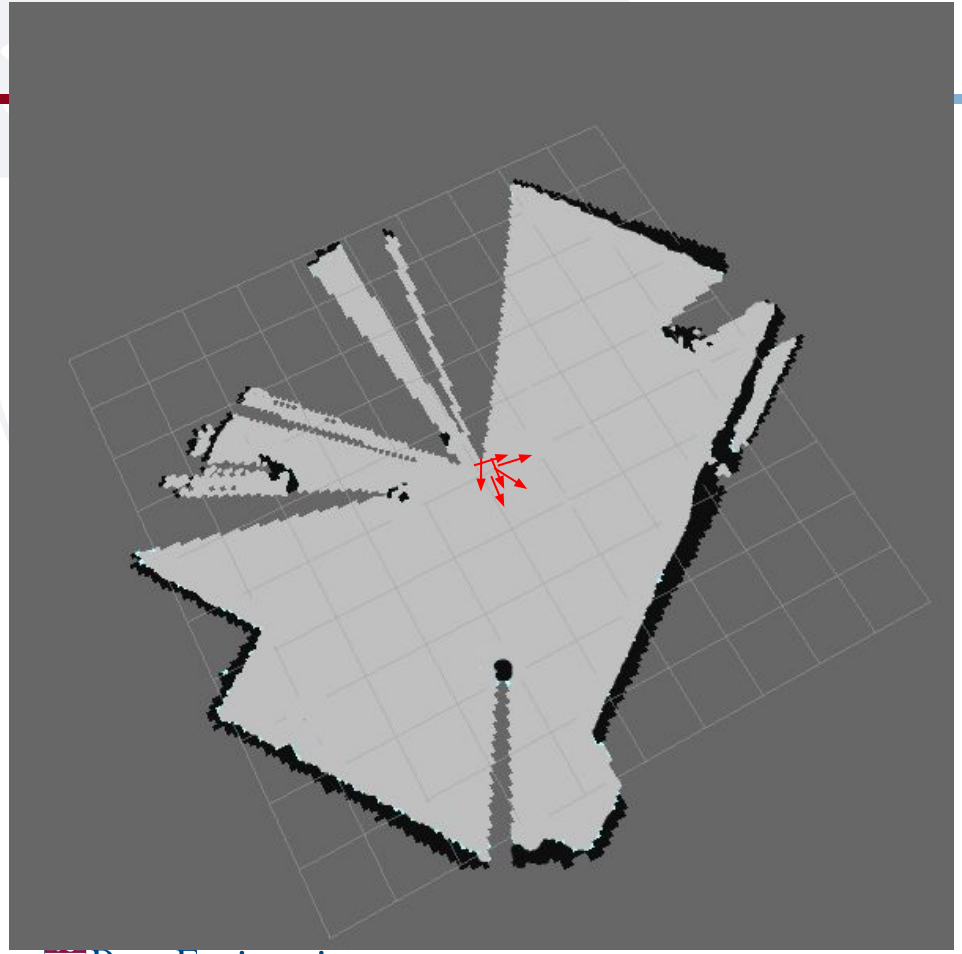
Localization using

Odometry



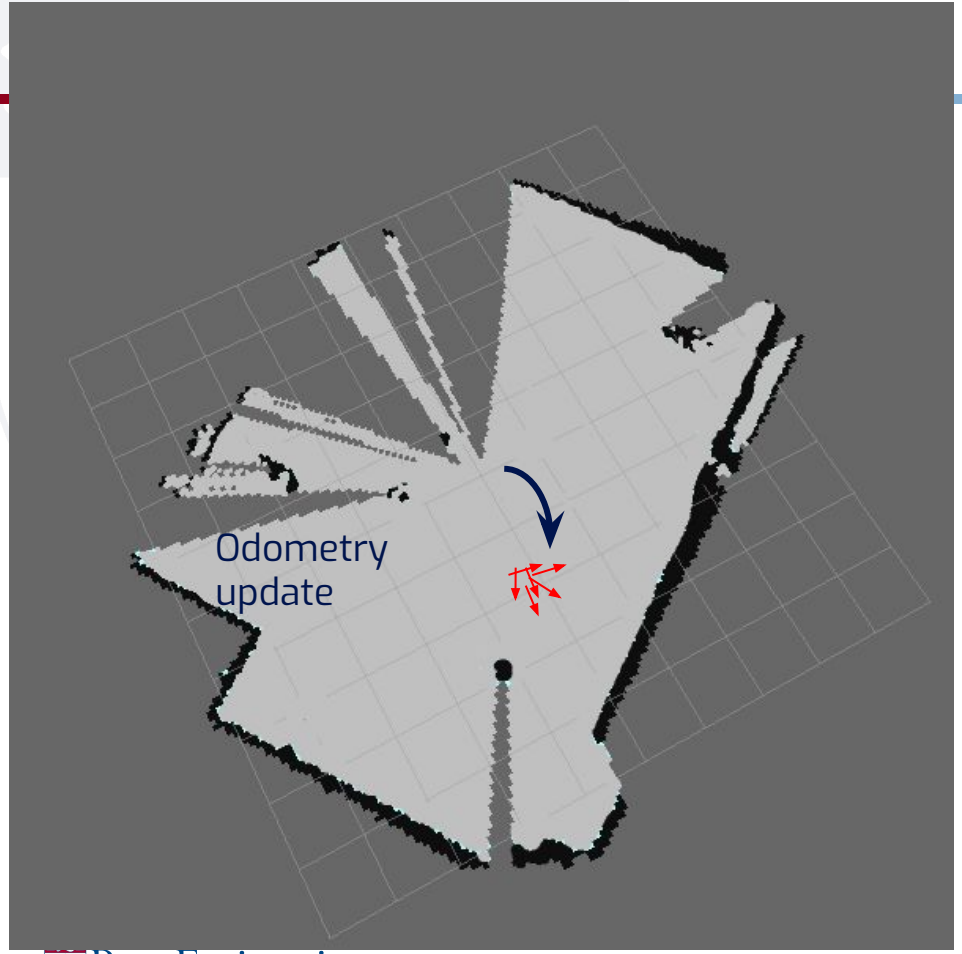
Particle Filter





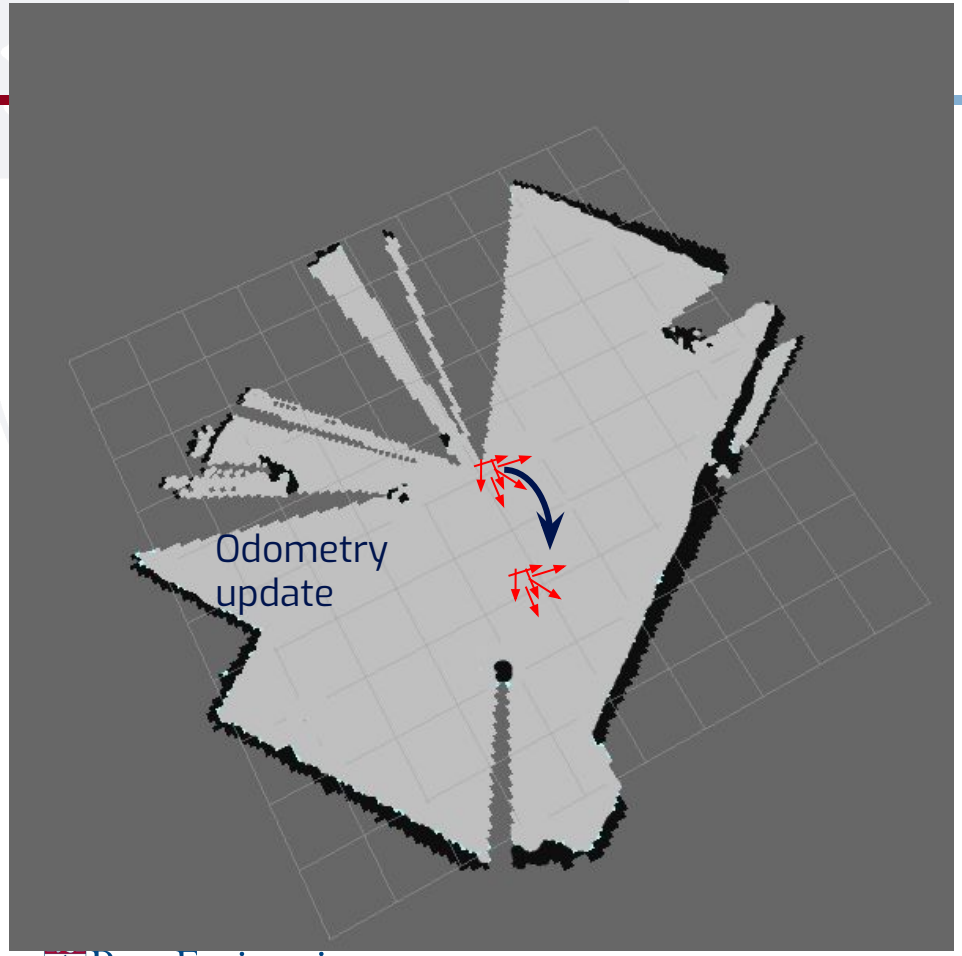
Update Step

- Update the particle cloud with the update in position from the odometry
- Repeat Scan matching process for each particle and determine the weights.



Update Step

- Update the particle cloud with the update in position from the odometry
- Repeat Scan matching process for each particle and determine the weights.



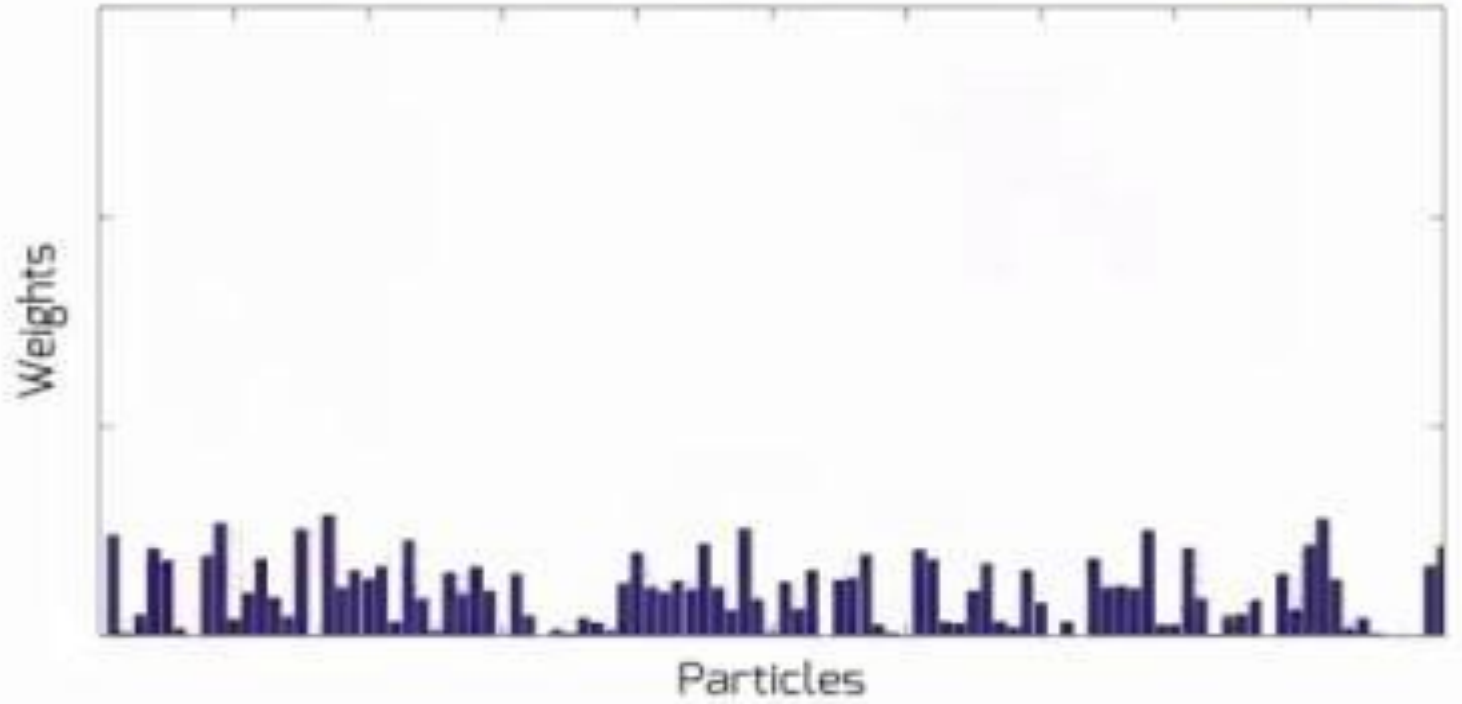
Update Step

- Update the particle cloud with the update in position from the odometry
- Repeat Scan matching process for each particle and determine the weights.

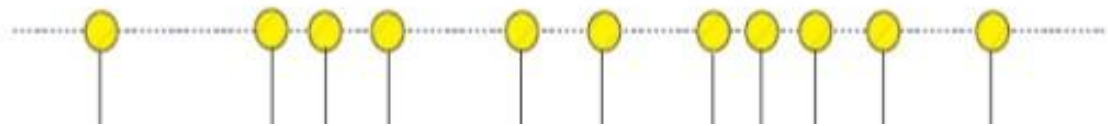
Particle Weights

$$W_t \leftarrow W_{t-1} \times S$$

Particle Filter without Resampling

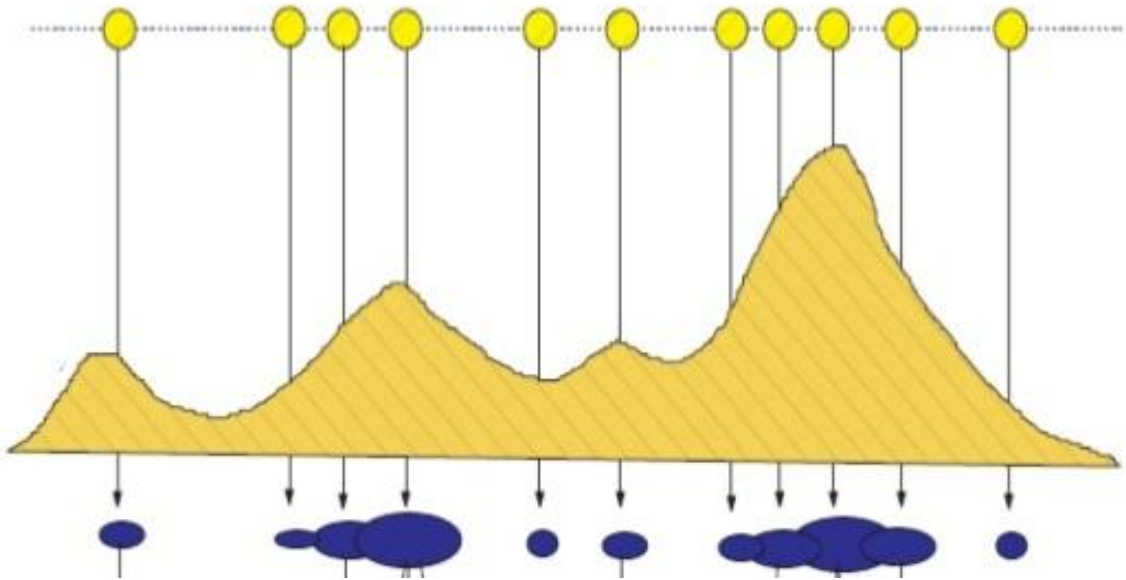


Resampling



Original Particles

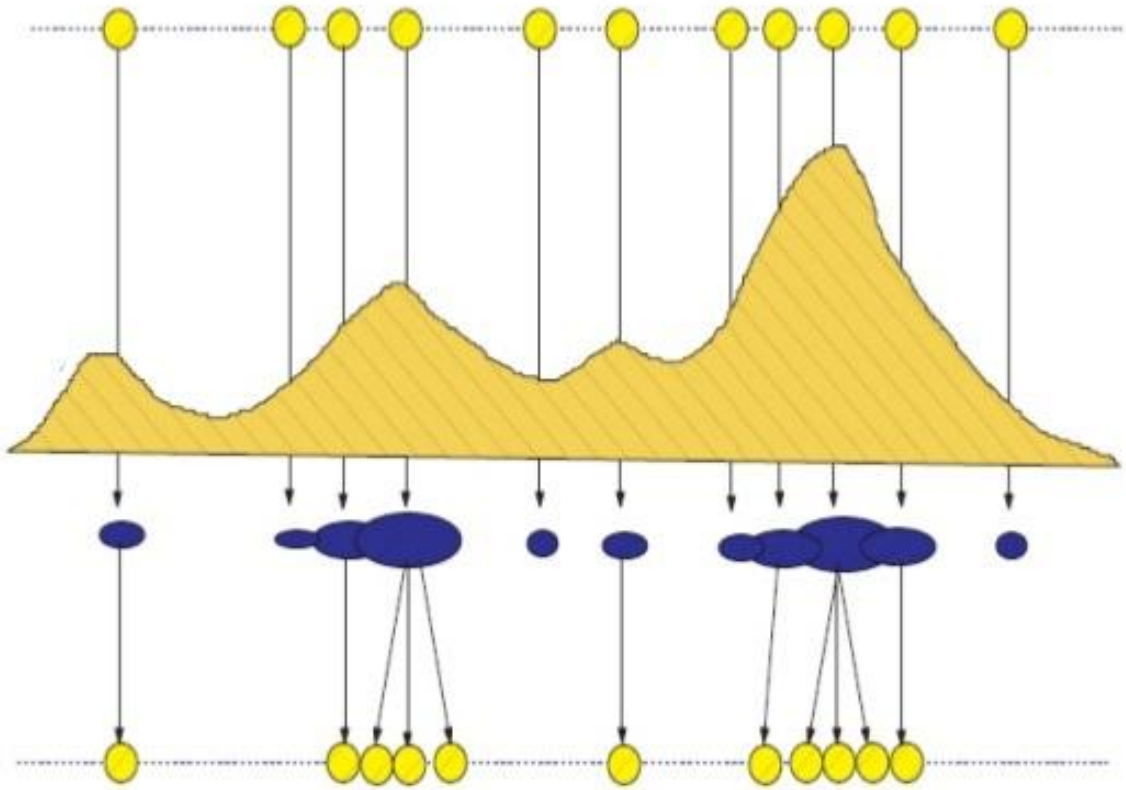
Resampling



Original Particles

After N iterations

Resampling



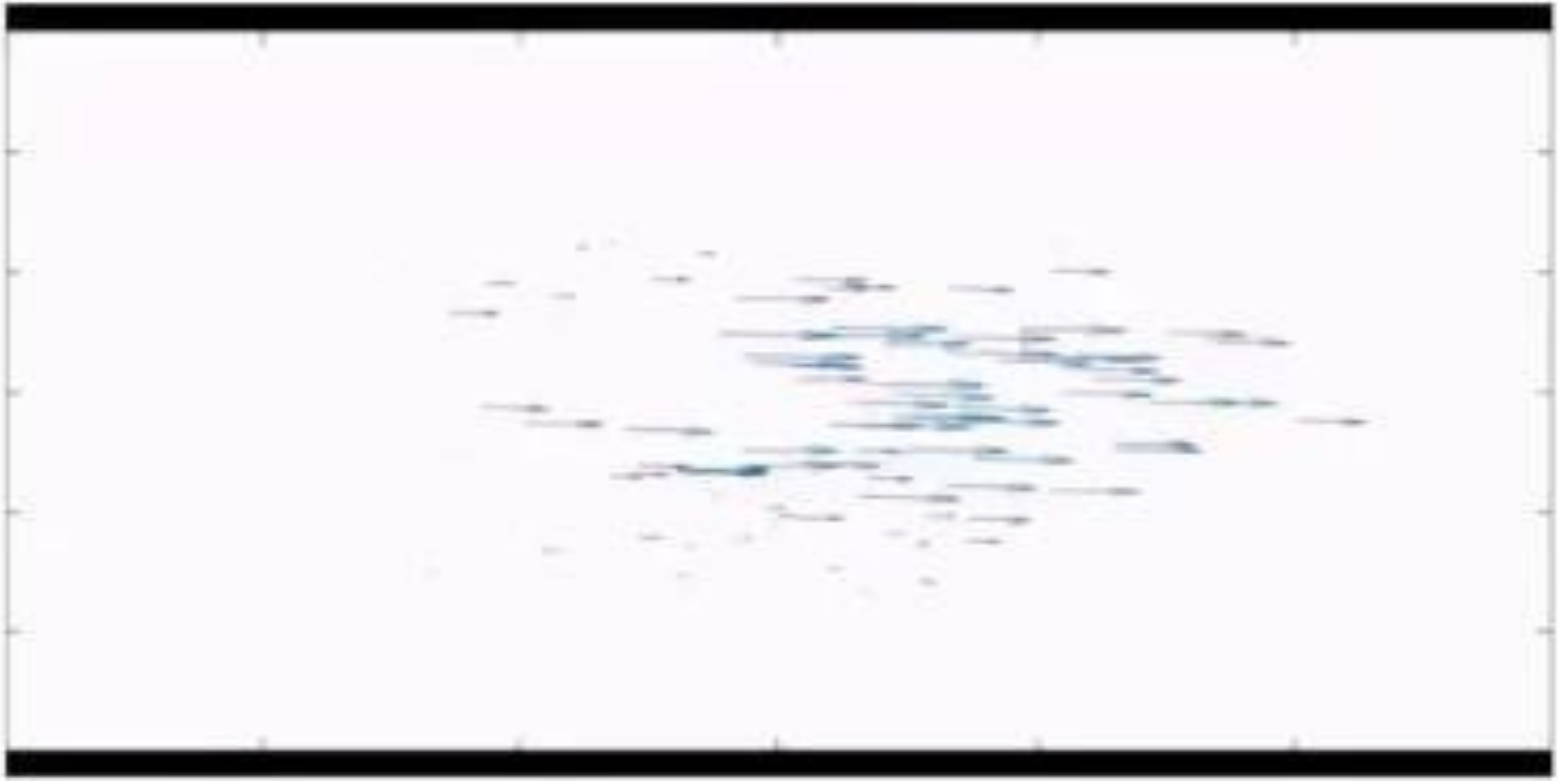
Original Particles

After N iterations

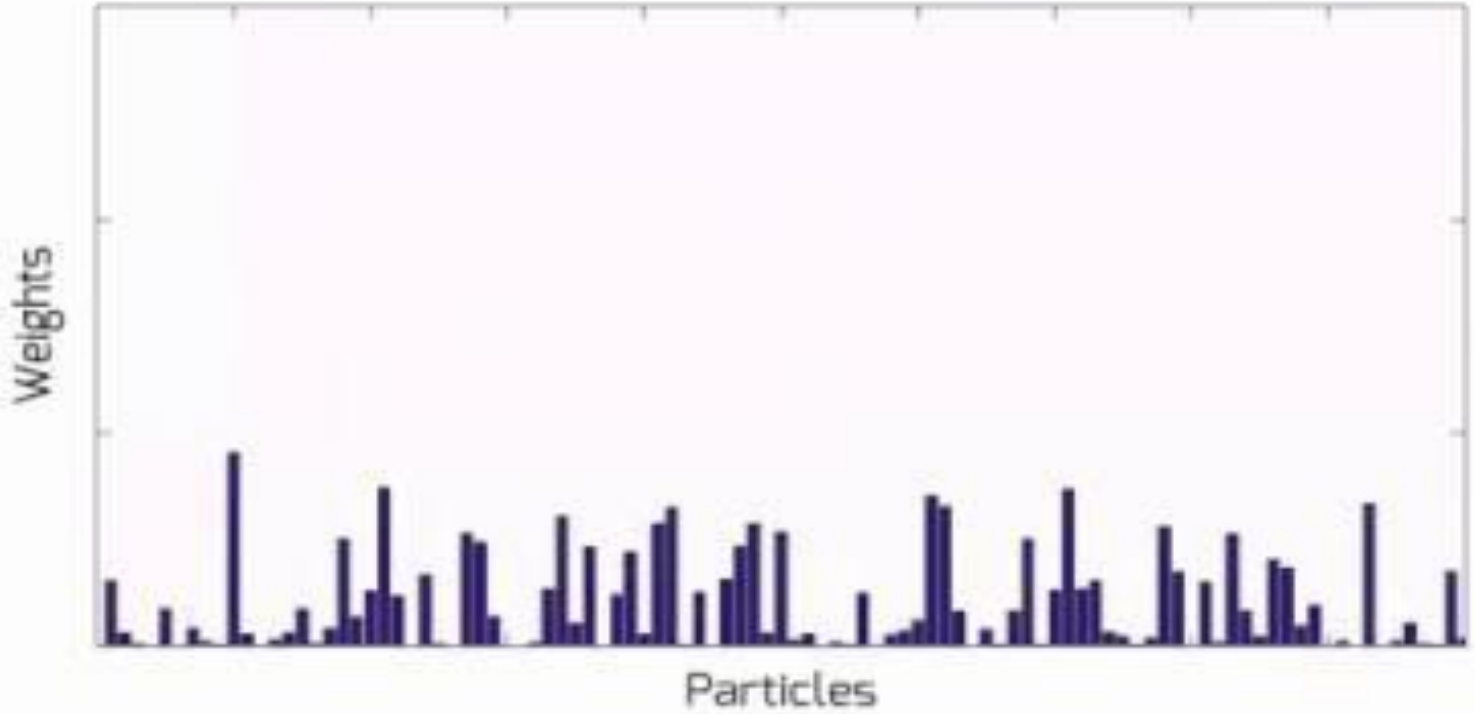
Resampling

Courtesy: Thrun, Burgard, Fox

Particles



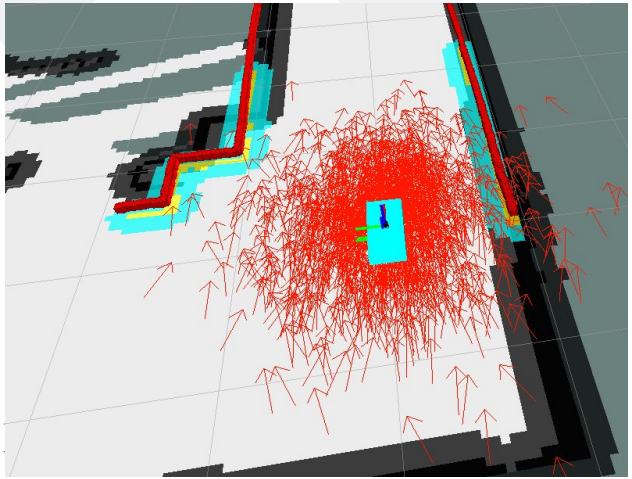
Particle filter with Resampling

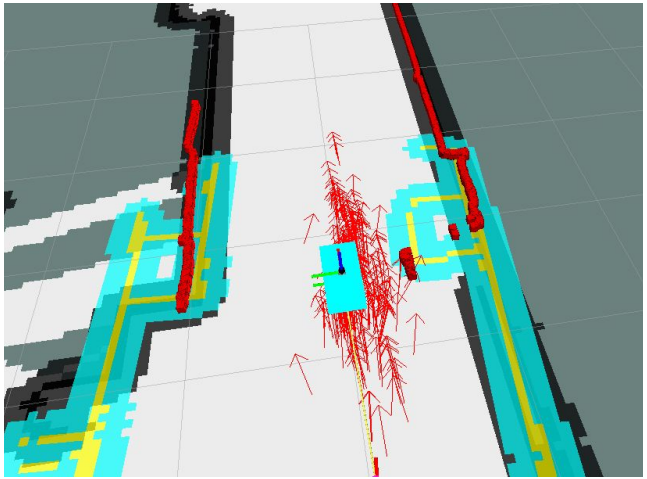


Kullback–Leibler divergence (KLD Sampling)

- Variable Particle size
- Sample size is proportional to error between odometry position and sample based approximation

  i.e smaller sample size when particles have converged

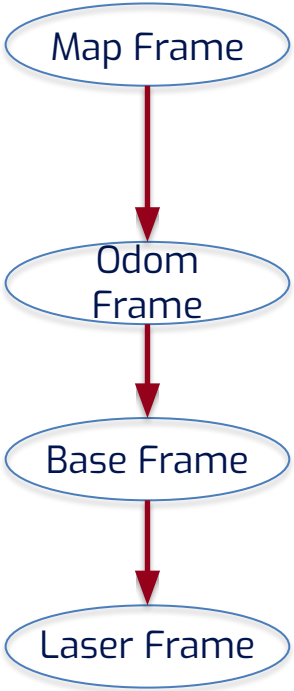




Particle Filters in ROS

- Adaptive Monte Carlo Localization Package
- Localization for a robot moving in a 2D space
- Localizes against a pre-existing map

System Tf tree



Tf Provided by Hector dometry

Tf required by Hector package

Penn Engineering

Input and Output Parameters

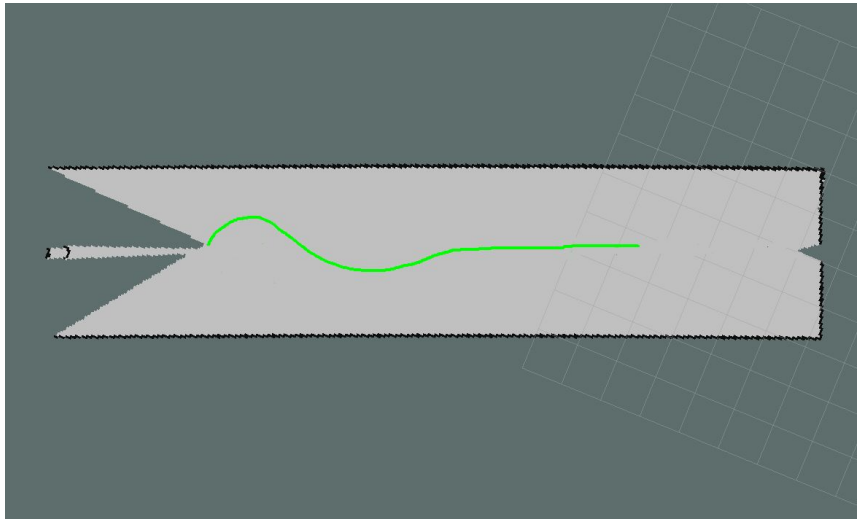
Input Parameters:

1. Laser Scan

Input and Output Parameters

Input Parameters:

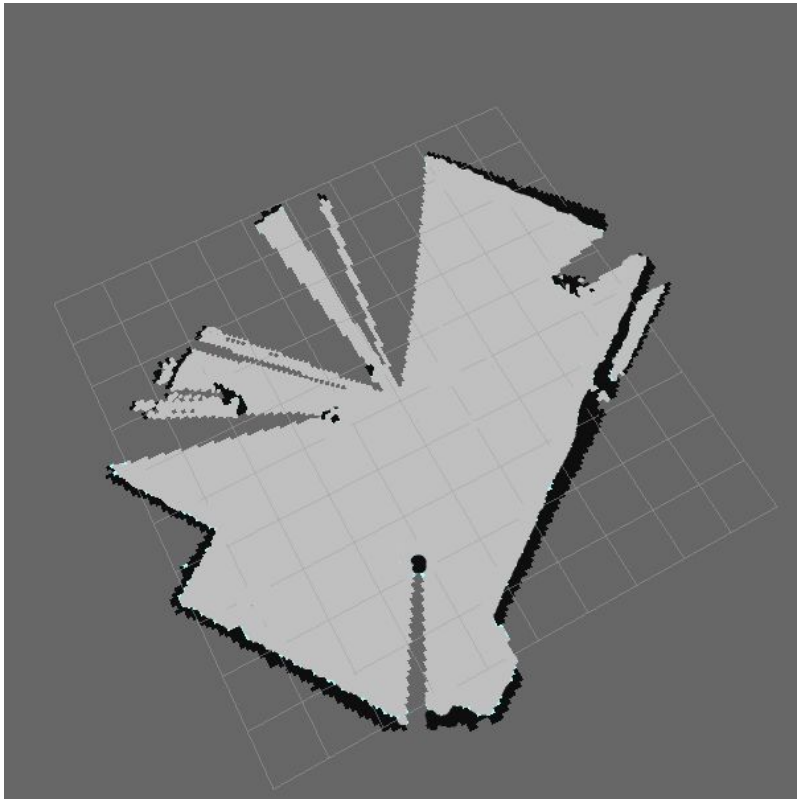
1. Laser Scan
2. Dead Reckoning/Odometry



Input and Output Parameters

Input Parameters:

1. Laser Scan
2. Dead Reckoning/Odometry
3. Map



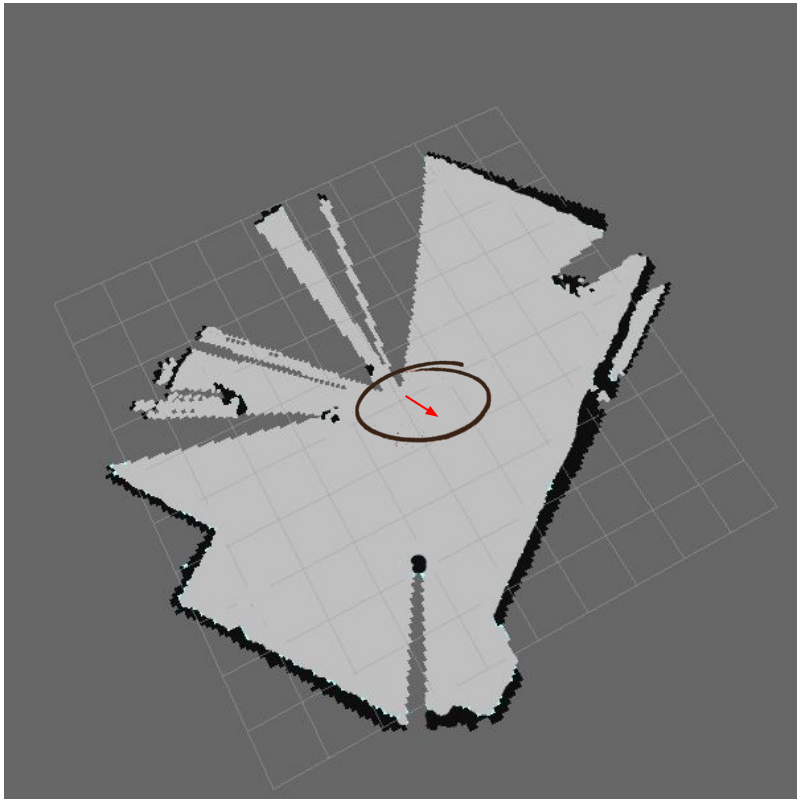
Input and Output Parameters

Input Parameters:

1. Laser Scan
2. Dead Reckoning/Odometry
3. Map

Output Parameters:

4. AMCL pose



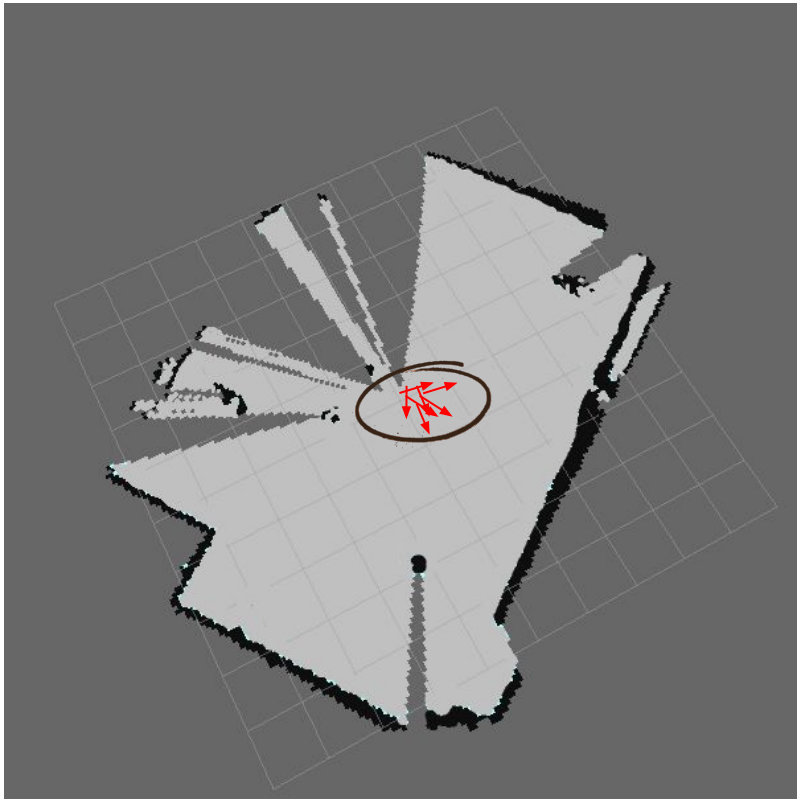
Input and Output Parameters

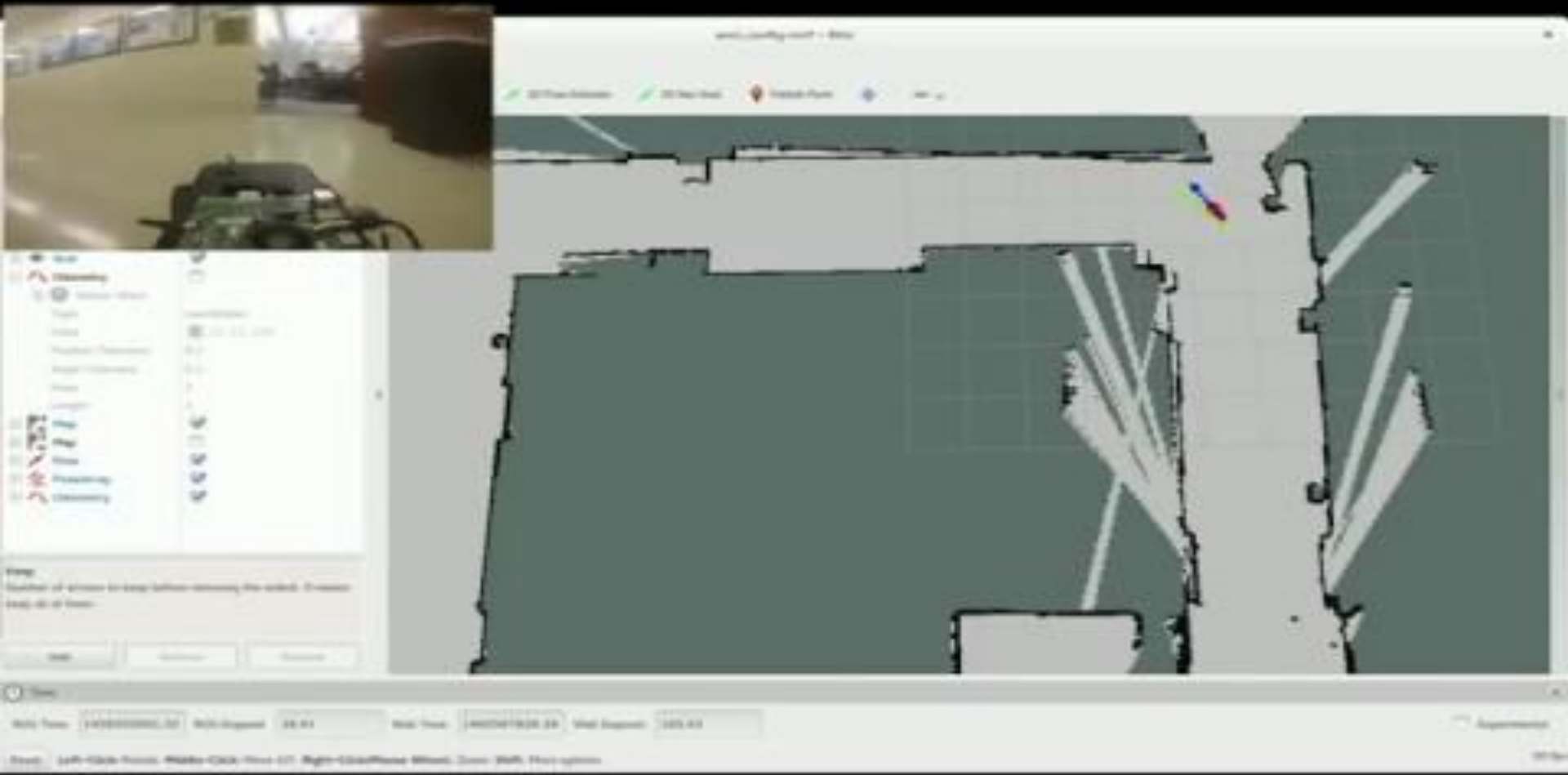
Input Parameters:

1. Laser Scan
2. Dead Reckoning/Odometry
3. Map

Output Parameters:

4. AMCL pose
5. Particle cloud





Particle Filters: Analysis

Particle Filters: Problem Definition

- Estimating the state of a dynamical system is fundamental problem
- The recursive Bayes Filter is an effective approach to estimate the belief about the state of a dynamical system
  - How to represent this belief?
  - How to maximize it?

Particle Filters: Problem Definition

- Estimating the state of a dynamical system is fundamental problem
- The recursive Bayes Filter is an effective approach to estimate the belief about the state of a dynamical system
  - How to represent this belief?
  - How to maximize it?
- Particle filters are a way to **efficiently** represent an arbitrary **(non-Gaussian) distribution**
- Basic Principle
  - Set of state hypothesis ('particles')
  - Survival of the fittest

Let's start with the Bayes Filter

The key idea of Bayes filtering is to estimate a probability density over the state-space conditioned on the data.

This posterior is typically called the belief and is denoted:

$$Bel(x_t) = p(x_t \mid d_{0\ldots t})$$

Belief or posterior

Robot state

At time t

The data from time 0 to t.

Let's start with the Bayes Filter

For mobile robots, we distinguish two types of data: perceptual data such as laser range measurements, and odometry data, which carries information about robot motion. Denoting the former by o (for observation) and the latter by a (for action), we have:

$$Bel(x_t) = p(x_t \mid o_t, a_{t-1}, o_{t-1}, \dots)$$

Belief or posterior

Robot state

Perceptual Observation

Odometry

History...

Bayes Filter

z = observation
u = action
x = state

$$Bel(x_t) = P(x_t \mid u_1, z_1, \ldots, u_t, z_t)$$

Bayes
$$= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

Markov
$$= \eta \; P(z_t \mid x_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

Total prob.
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \, dx_{t-1}$$

Markov
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \, dx_{t-1}$$

Markov
$$= \eta P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) P(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1}) \, dx_{t-1}$$

$$= \eta P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \, dx_{t-1}$$

Observation Update

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)}$$

Use Bayes rule to rewrite the the belief about the robots pose given in the previous slide.

$$Bel(x_t) = \frac{p(o_t \mid x_t, a_{t-1}, \ldots, o_0)p(x_t \mid a_{t-1}, \ldots, o_0)}{p(o_t \mid a_{t-1}, \ldots, o_0)}$$

$$Bel(x_t) = \eta p(o_t \mid x_t, a_{t-1}, \ldots, o_0)p(x_t \mid a_{t-1}, \ldots, o_0)$$

Markov Assumption

Bayes filters rest on the assumption that future data is independent of past data given knowledge of the current state. This is typically referred to as the Markov assumption. Mathematically, the Markov assumption implies:

$$p(o_t \mid x_t, a_{t-1}, \ldots, o_0) = p(o_t \mid x_t)$$

Simplifying the Update...

Using the Markov assumption we can rewrite the belief based on the last perceptual measurement as:

$$Bel(x_t) = \eta p(o_t \mid x_t) p(x_t \mid a_{t-1}, \ldots, o_0)$$

Odometry Update

What if we just received an odometry measurement instead?

$$Bel(x_t) = p(x_t \mid x_{t-1}, a_{t-1}, \ldots, o_0)$$

Rewrite via the Law of Total Probability...

$$Bel(x_t) = \int p(x_t \mid x_{t-1}, a_{t-1}, \ldots, o_0) p(x_{t-1} \mid a_{t-1}, \ldots, o_0) dx_{t-1}$$

Apply the Markov assumption...

$$Bel(x_t) = \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Bayes Recursive Filter

After some simplification we have:

Integrate out over previous beliefs. In practice finite approximation

$$Bel(x_t) = \eta \; p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Normalization Constant
Make sure everything adds up to 1!

Observation Likelihood
Sensor Model
Compute how likely your measurements were given updated particles.

Transition Model
Motion Model
Simulate noisy dynamics of particles based on control input.

Previous Belief
Draw your particles with replacement according to importance weight.

Read right to left

In practice we represent the distributions non-parametrically using particles (a finite set of samples)

“Where Am I?”

Penn Engineering

Courtesy: Cyrill Stachniss

“Where Am I?”

Penn Engineering

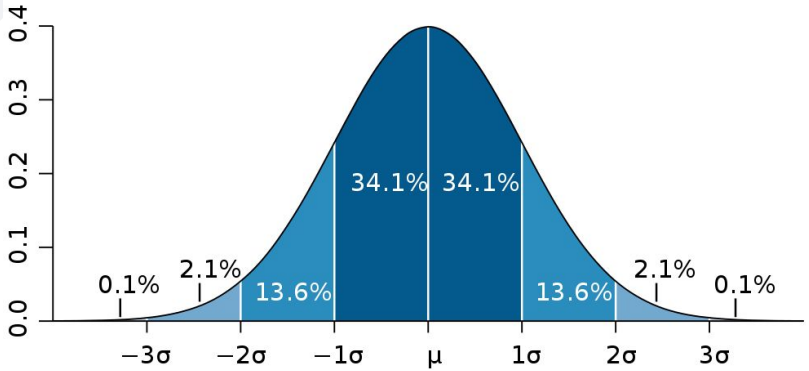
Courtesy: Cyrill Stachniss

Gaussian Filters

The Kalman filter and its variants can only model

Gaussian distributions

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \right)$$

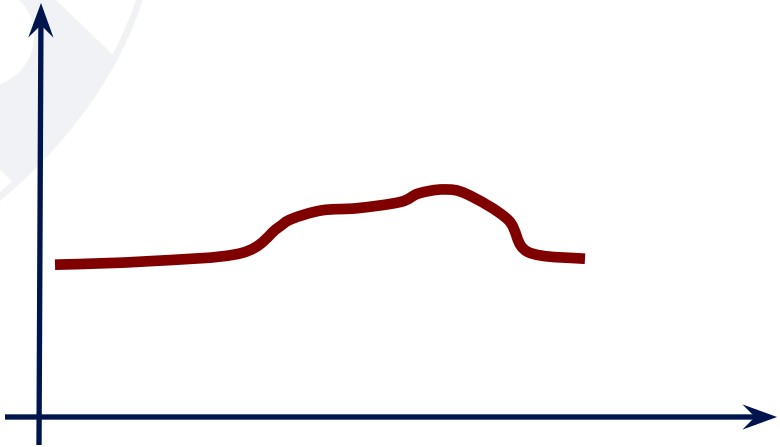


Penn Engineering

Image courtesy: K. Arras

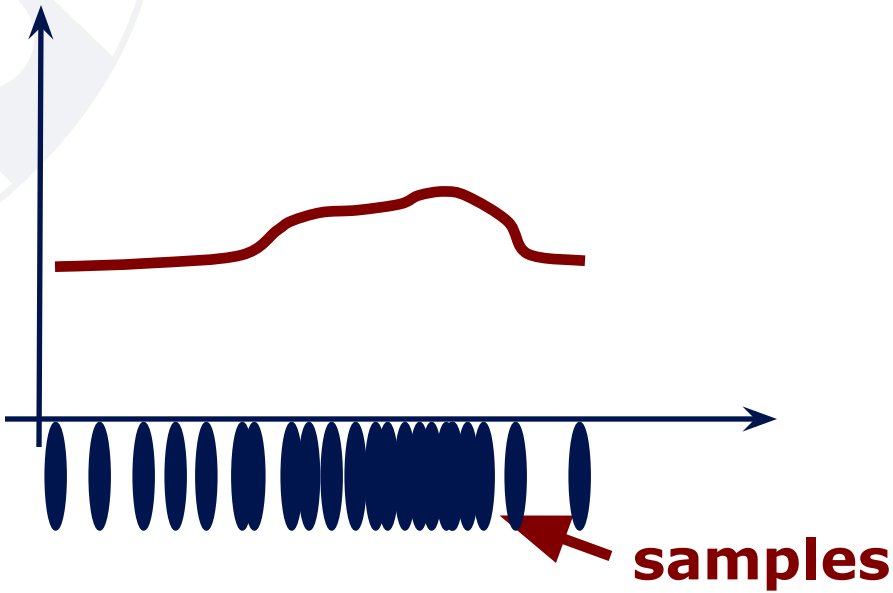
Flexible Function Approximation

Goal: approach for dealing with **arbitrary distributions**



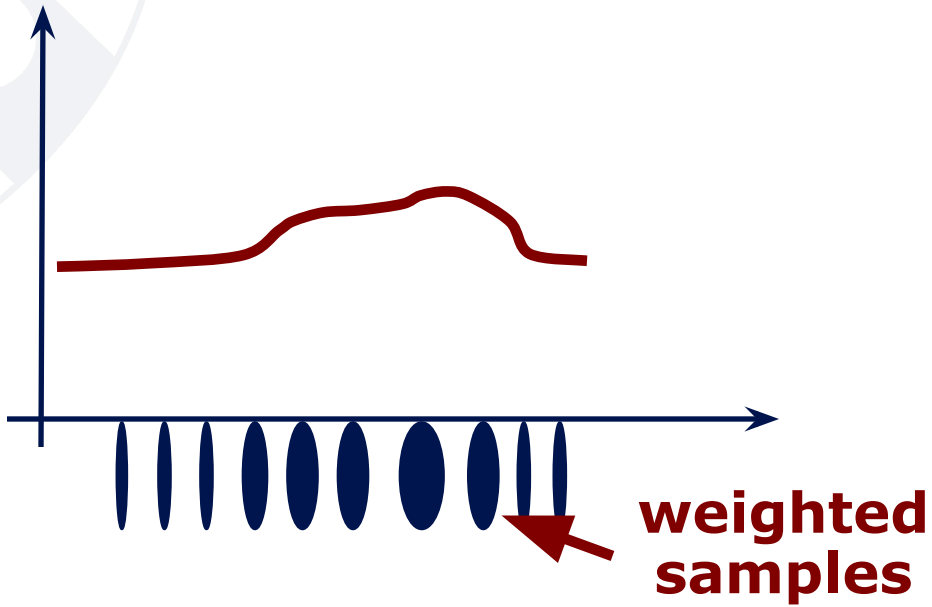
Key Idea: Samples

Multiple samples to represent arbitrary distributions



Key Idea: Weighted Samples

Multiple **weighted samples** to represent arbitrary distributions



Particle Set

- Set of weighted samples

$$\mathcal{X} = \left\{ \left\langle x^{[j]}, w^{[j]} \right\rangle \right\}_{j=1,\ldots,J}$$

state hypothesis

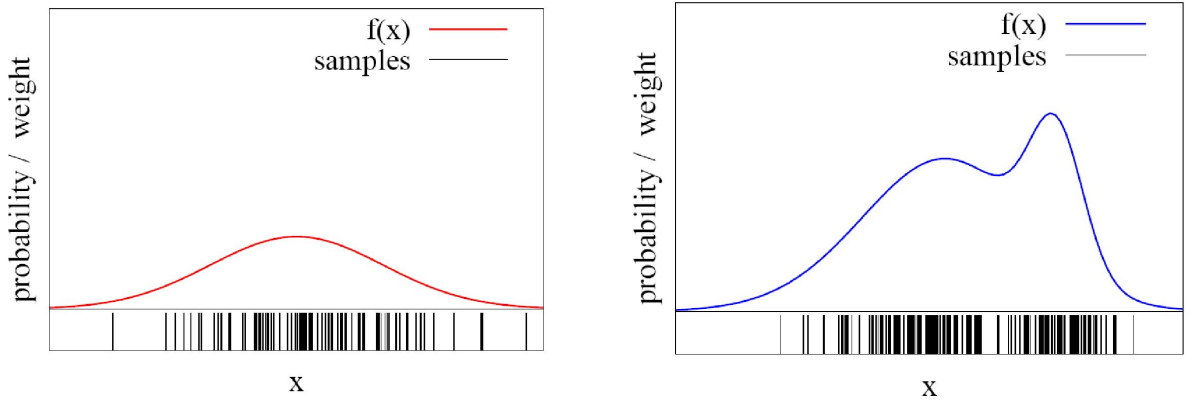
importance weight

- The samples represent the posterior

$$p(x) = \sum_{j=1}^{J} w^{[j]} \, \delta_{x^{[j]}}(x)$$

Particles for Approximation

- Particles for function approximation

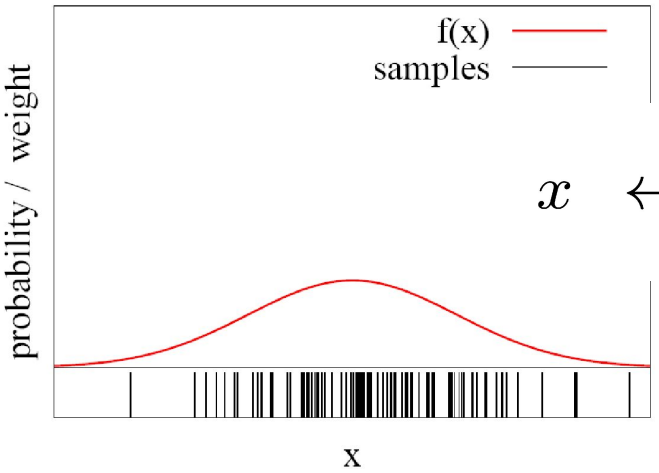


- The more particles fall into a region, the higher the probability of the region

How to obtain such samples?

Closed-form sampling is only possible for a few distributions

- Example: Gaussian



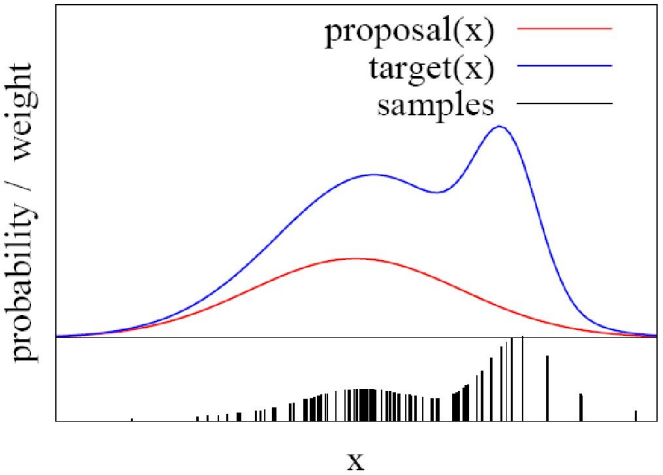
$$x \leftarrow \frac{1}{2}\sum_{i=1}^{12} \text{rand}(-\sigma, \sigma)$$

How to sample from **other** distributions?

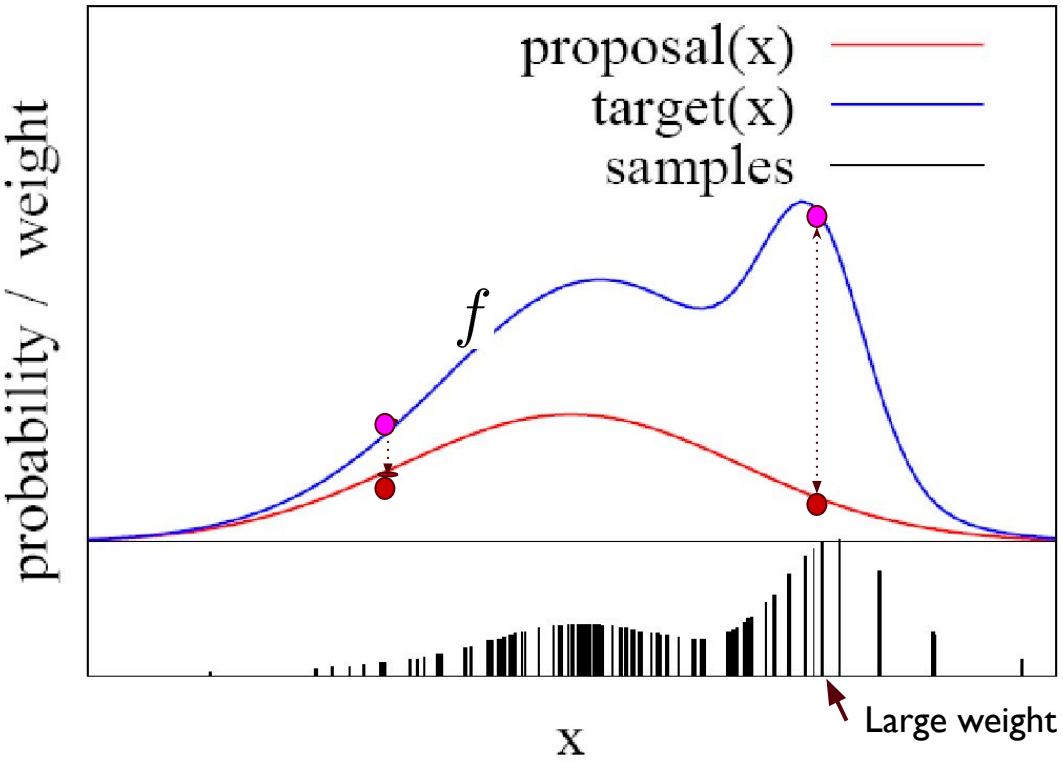
Importance Sampling Principle

- We can use a different distribution π to generate samples from f
- Account for the "differences between π and f" using a weight $\omega = f(x)/\pi(x)$
- Target f
- Proposal π
- Pre-condition:

$$f(x) > 0 \rightarrow \pi(x) > 0$$



Importance Sampling Principle



$$\omega = f(x)/\pi(x)$$

Particle Filter for Dynamic State Estimation Problems

- Recursive Bayes filter
- Non-parametric approach
- Models the distribution by samples
- **Prediction:** draw from the proposal
- **Correction:** weighting by the ratio of target and proposal

The more samples we use,
the better is the estimate!

Particle Filter Algorithm

1. Sample the particles using the proposal distribution

$$x_t^{[j]} \sim proposal(x_t \mid \ldots)$$

2. Compute the importance weights

$$w_t^{[j]} = \frac{target(x_t^{[j]})}{proposal(x_t^{[j]})}$$

3. Resampling: Draw sample i with probability $w_t^{[i]}$ and repeat J times

Particle Filter Algorithm

Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$: ← Input: Old sample set, controls, observations

1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ ← Predicted set and Correct set

2: *for* $j = 1$ *to* J *do*

3: *sample* $x_t^{[j]} \sim \pi(x_t)$ **Prediction Step**

4: $w_t^{[j]} = \frac{p(x_t^{[j]})}{\pi(x_t^{[j]})}$

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* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$

9: *add* $x_t^{[i]}$ *to* $\mathcal{X}_t$

10: *endfor*

11: *return* $\mathcal{X}_t$

Particle Filter Algorithm

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* $x_t^{[j]} \sim \pi(x_t)$
4: $w_t^{[j]} = \frac{p(x_t^{[j]})}{\pi(x_t^{[j]})}$ ← **Correction Step**
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* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$
9: *add* $x_t^{[i]}$ *to* $\mathcal{X}_t$
10: *endfor*
11: *return* $\mathcal{X}_t$

Particle Filter Algorithm

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* $x_t^{[j]} \sim \pi(x_t)$
4: $w_t^{[j]} = \frac{p(x_t^{[j]})}{\pi(x_t^{[j]})}$
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* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$
9: *add* $x_t^{[i]}$ *to* $\mathcal{X}_t$
10: *endfor*
11: *return* $\mathcal{X}_t$

Re-sampling step

Monte Carlo Localization

- Each particle is a pose hypothesis
- Proposal is the motion model

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

- Correction via the observation model

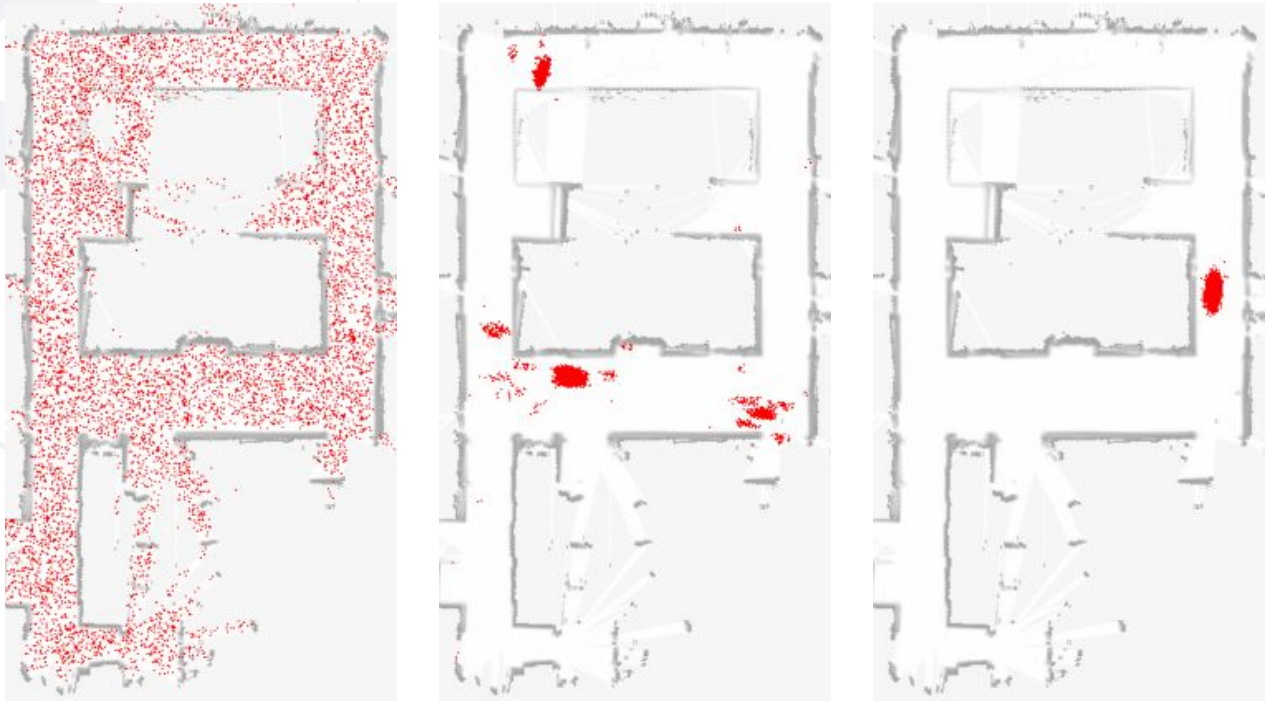
$$w_t^{[j]} = \frac{target}{proposal} \propto p(z_t \mid x_t, m)$$

Particle Filter for Localization

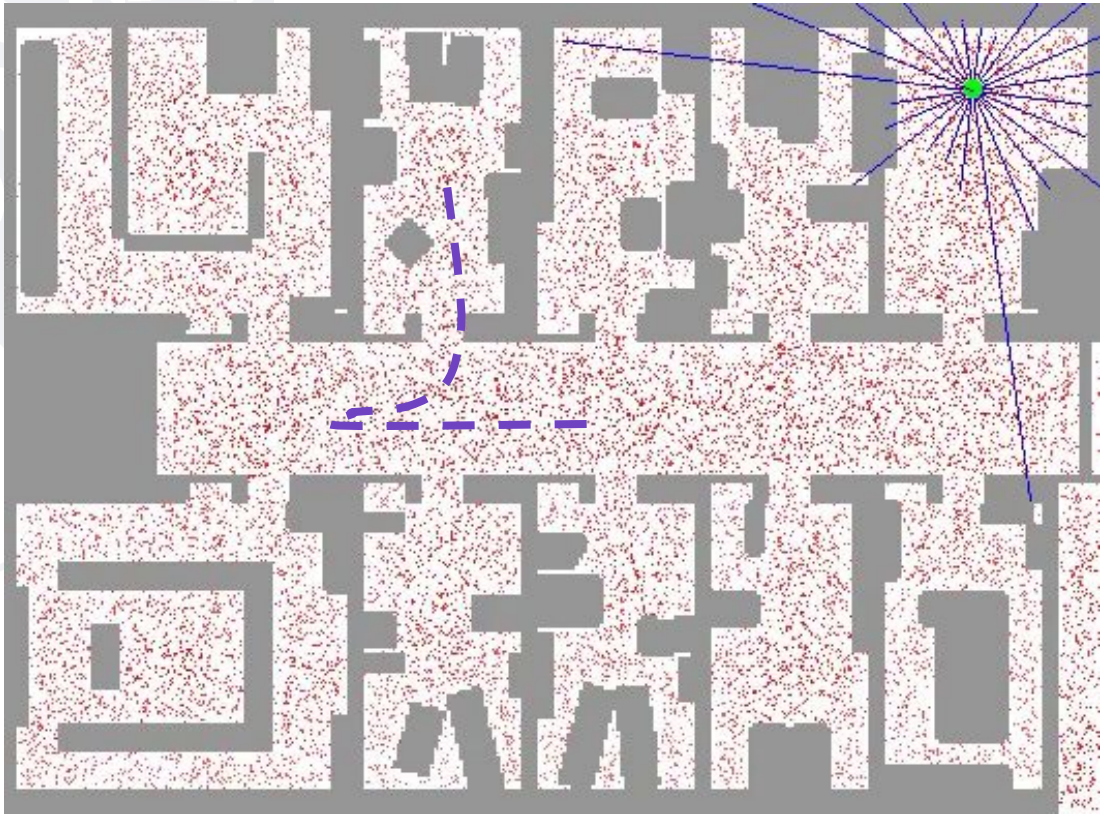
Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: *for* $j = 1$ *to* J *do*
3: *sample* $x_t^{[j]} \sim p(x_t \mid u_t, x_{t-1}^{[j]})$
4: $w_t^{[j]} = p(z_t \mid x_t^{[j]})$
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* $i \in 1, \ldots, J$ *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:
Solve "Where Am I?" Using Particles



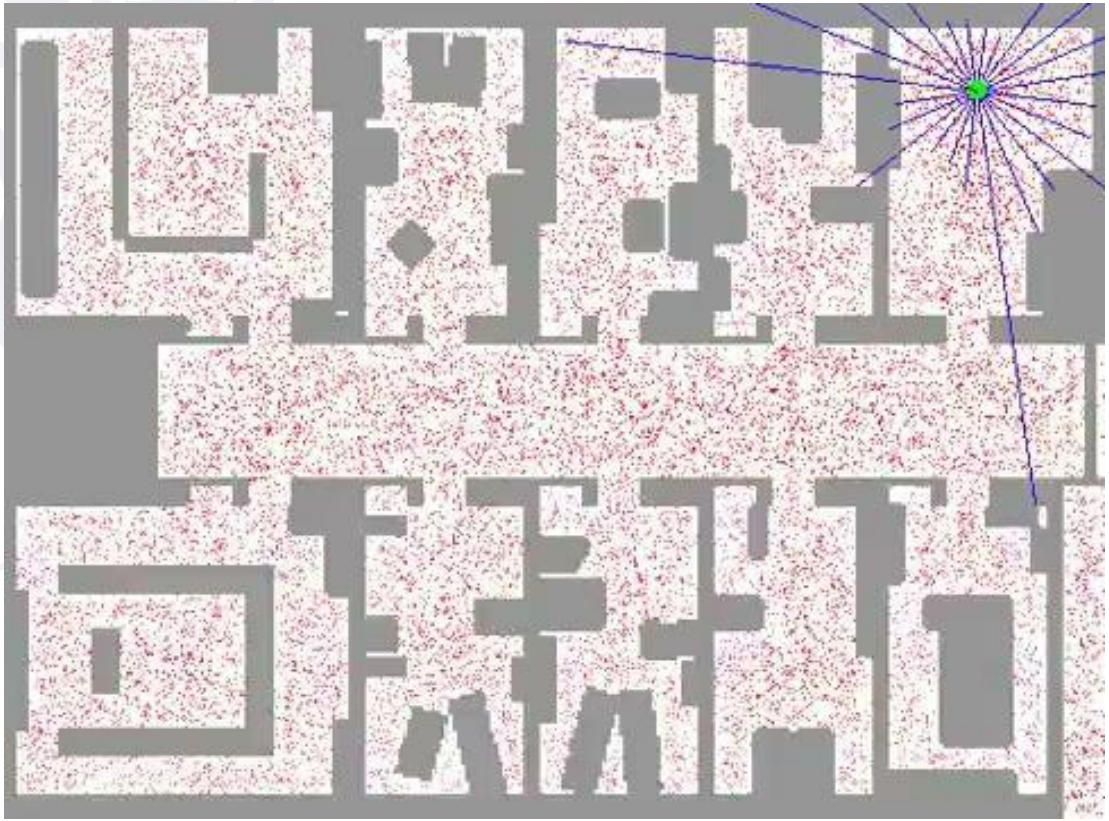
Particle Filter for Localization in a known Map



Penn Engineering

Courtesy: D. Fox

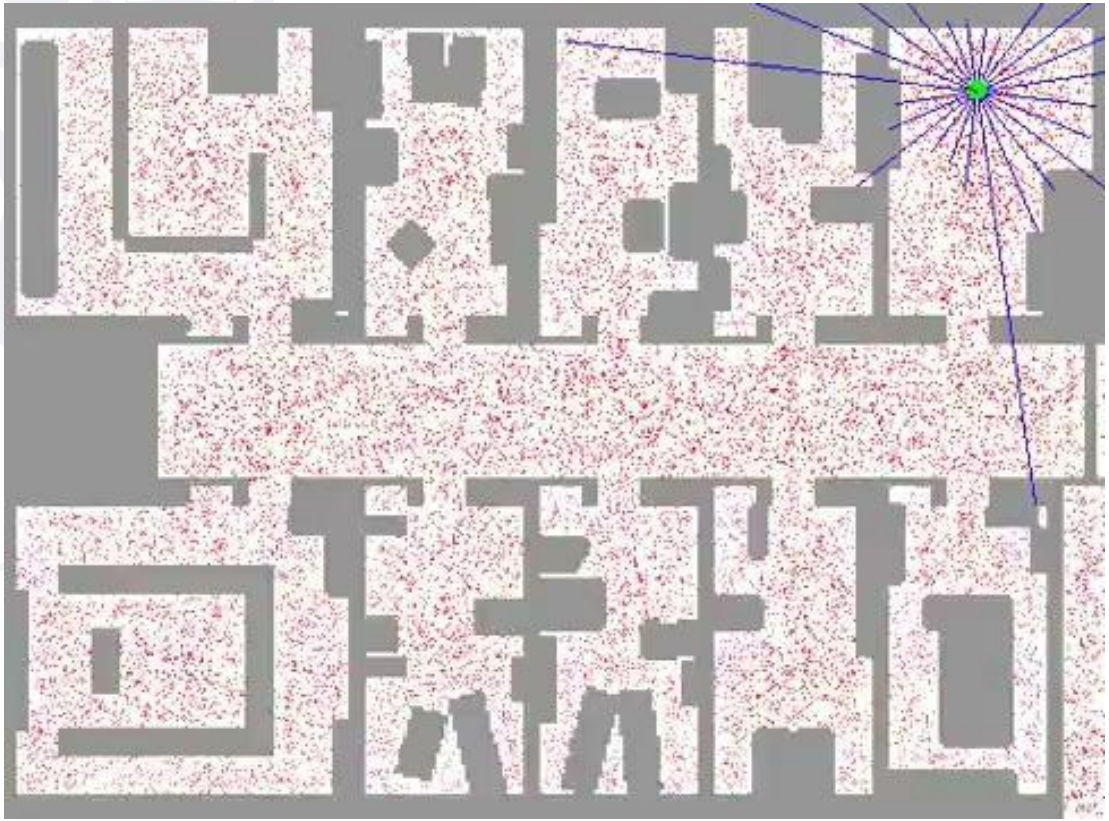
Particle Filter for Localization in a known Map



Penn Engineering

Courtesy: D. Fox

Particle Filter for Localization in a known Map

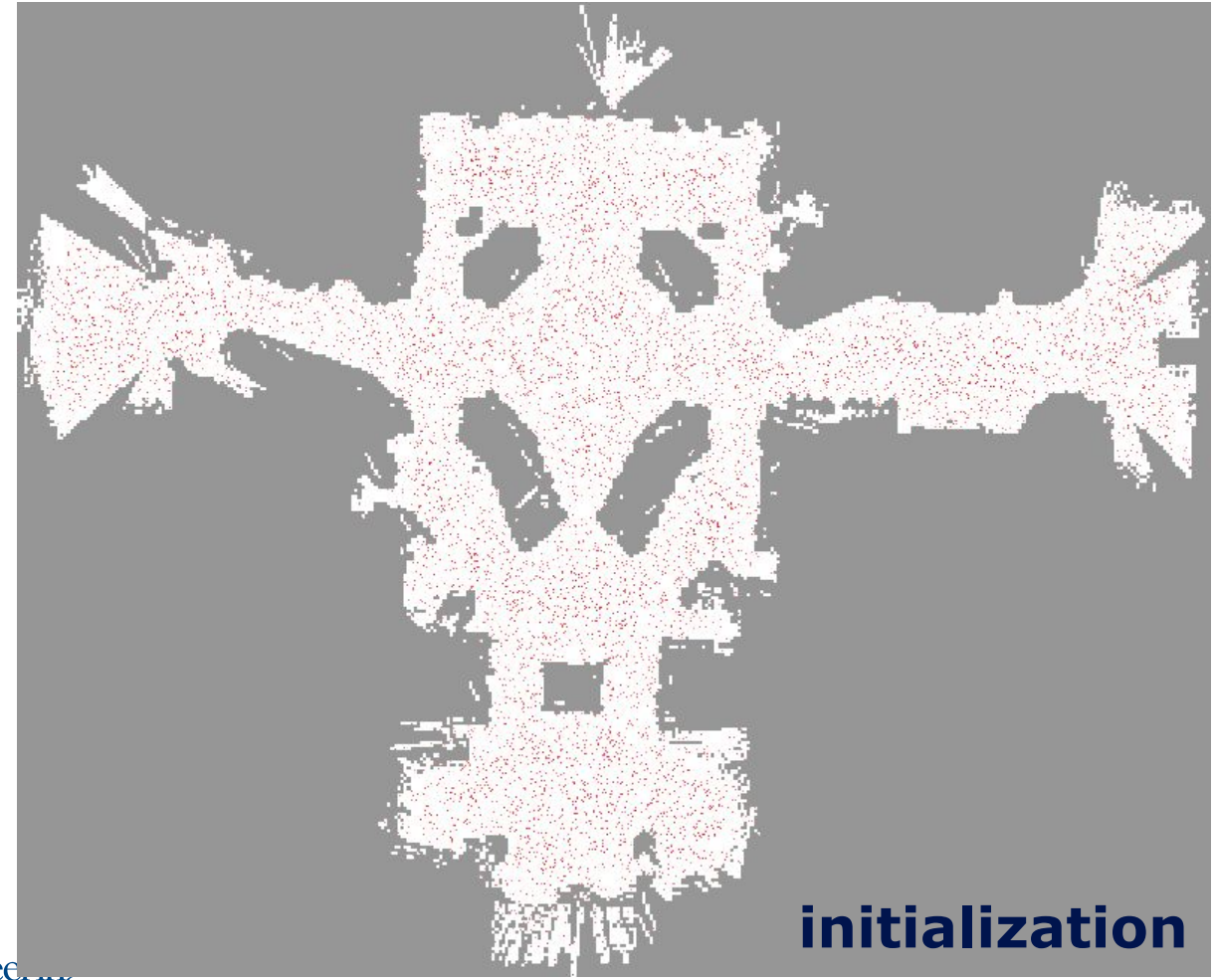


Penn Engineering

Courtesy: D. Fox

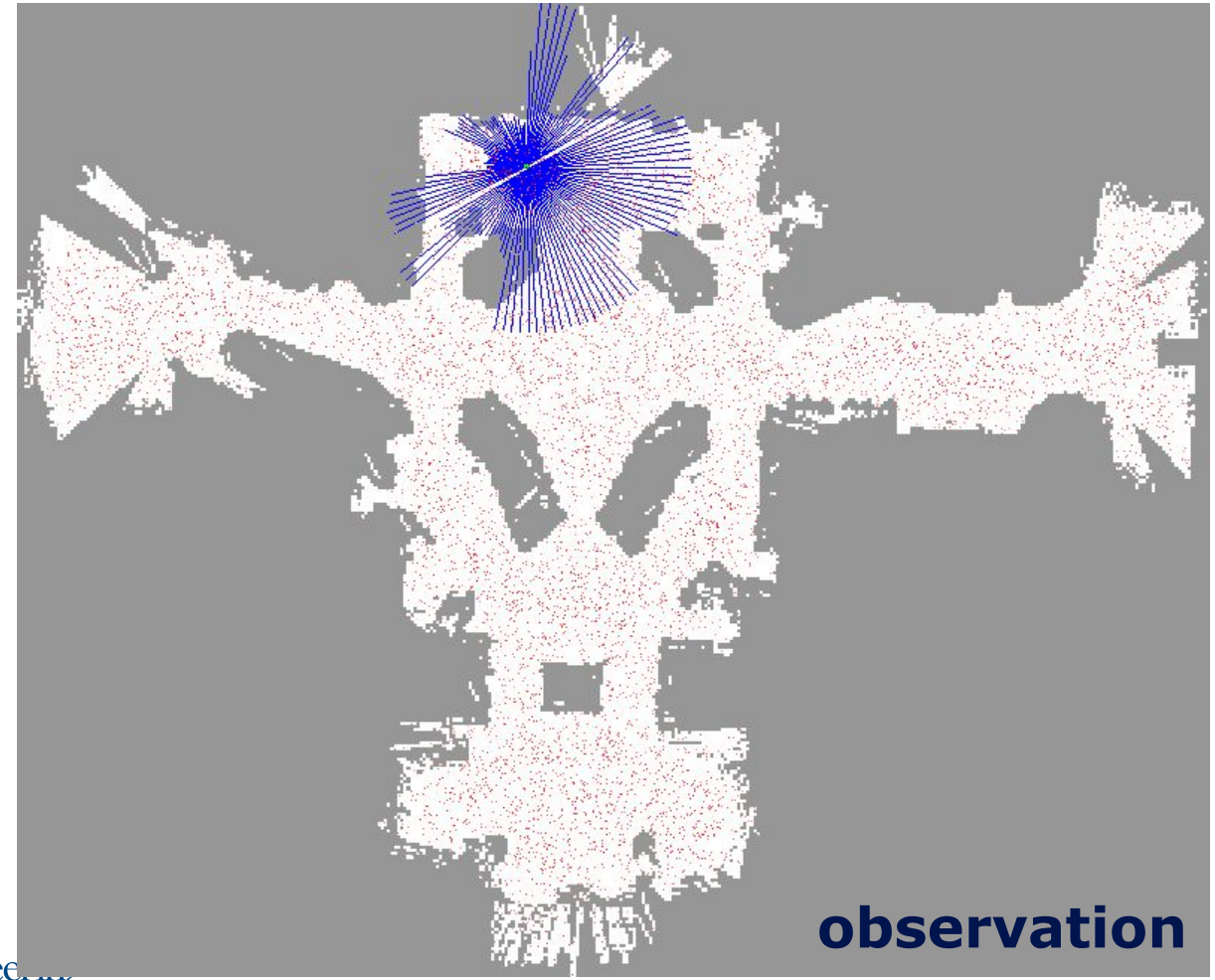
Resampling

- Draw sample i with probability $w_t^{[i]}$
- Repeat J times
- Informally: "Replace unlikely samples by more likely ones"
- Survival of the fittest
- "Trick" to avoid that many samples cover unlikely states
- Needed as we have a limited number of samples



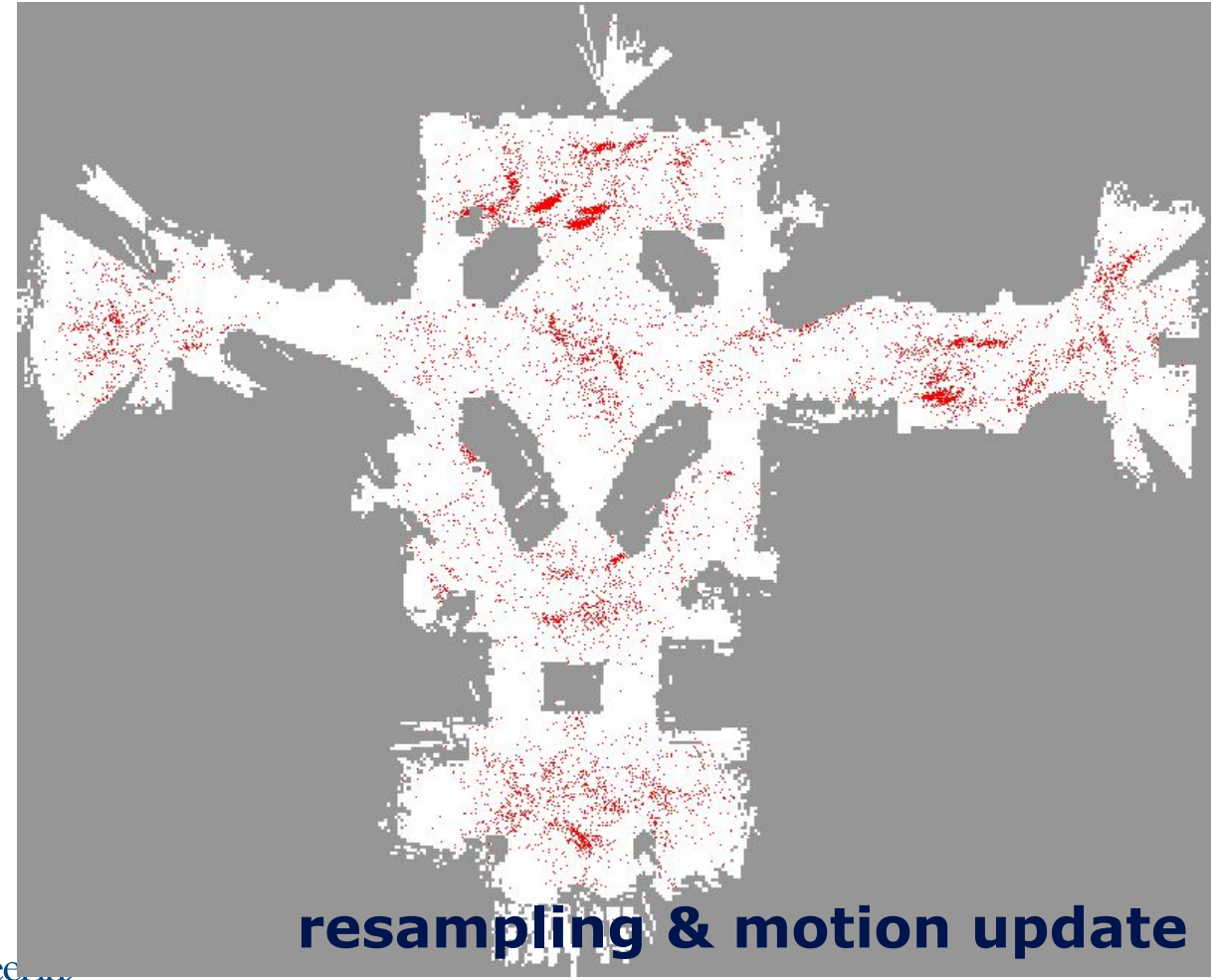
Penn Engineering

Courtesy: Thrun, Burgard, Fox



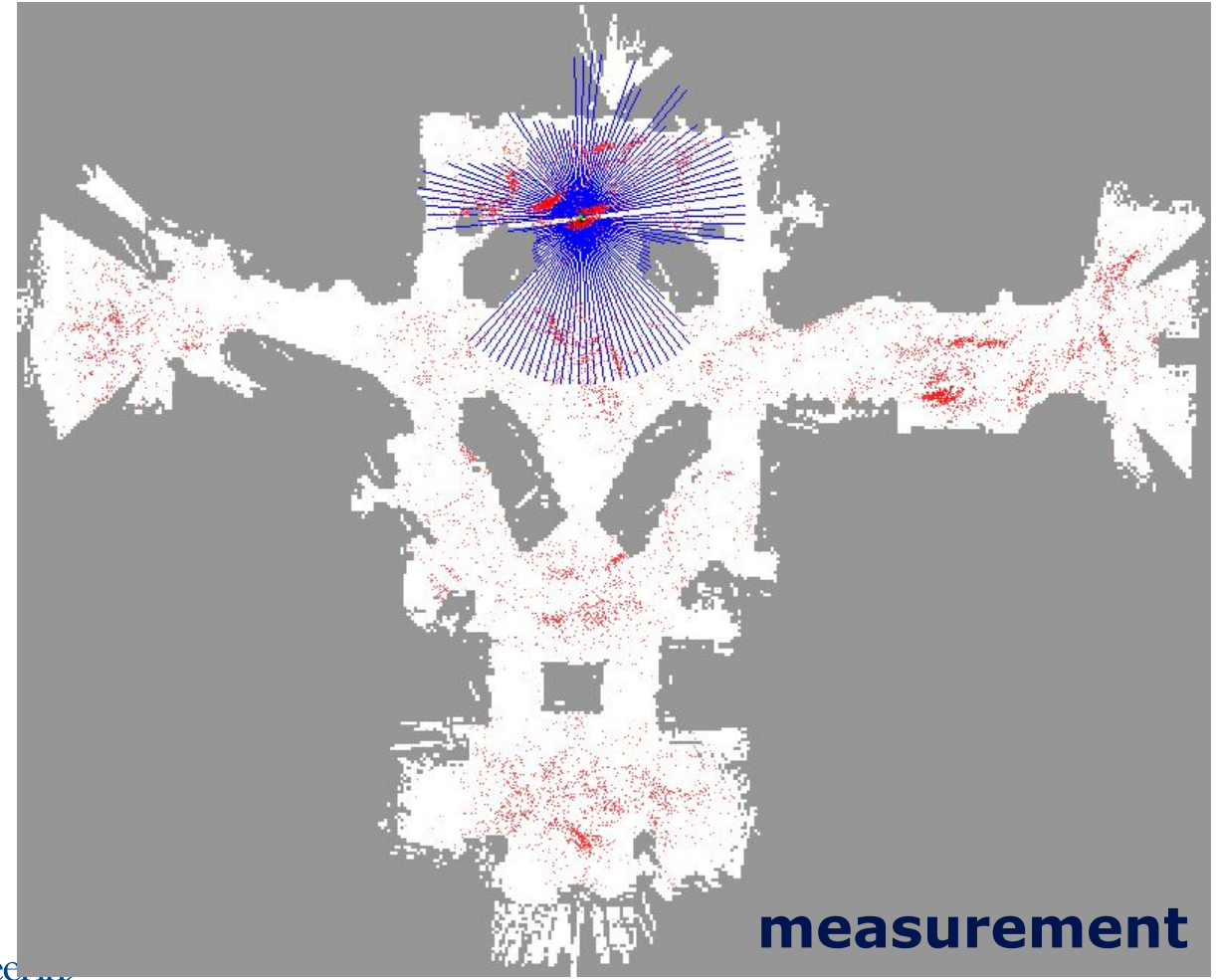
Penn Engineering

Courtesy: Thrun, Burgard, Fox



Penn Engineering

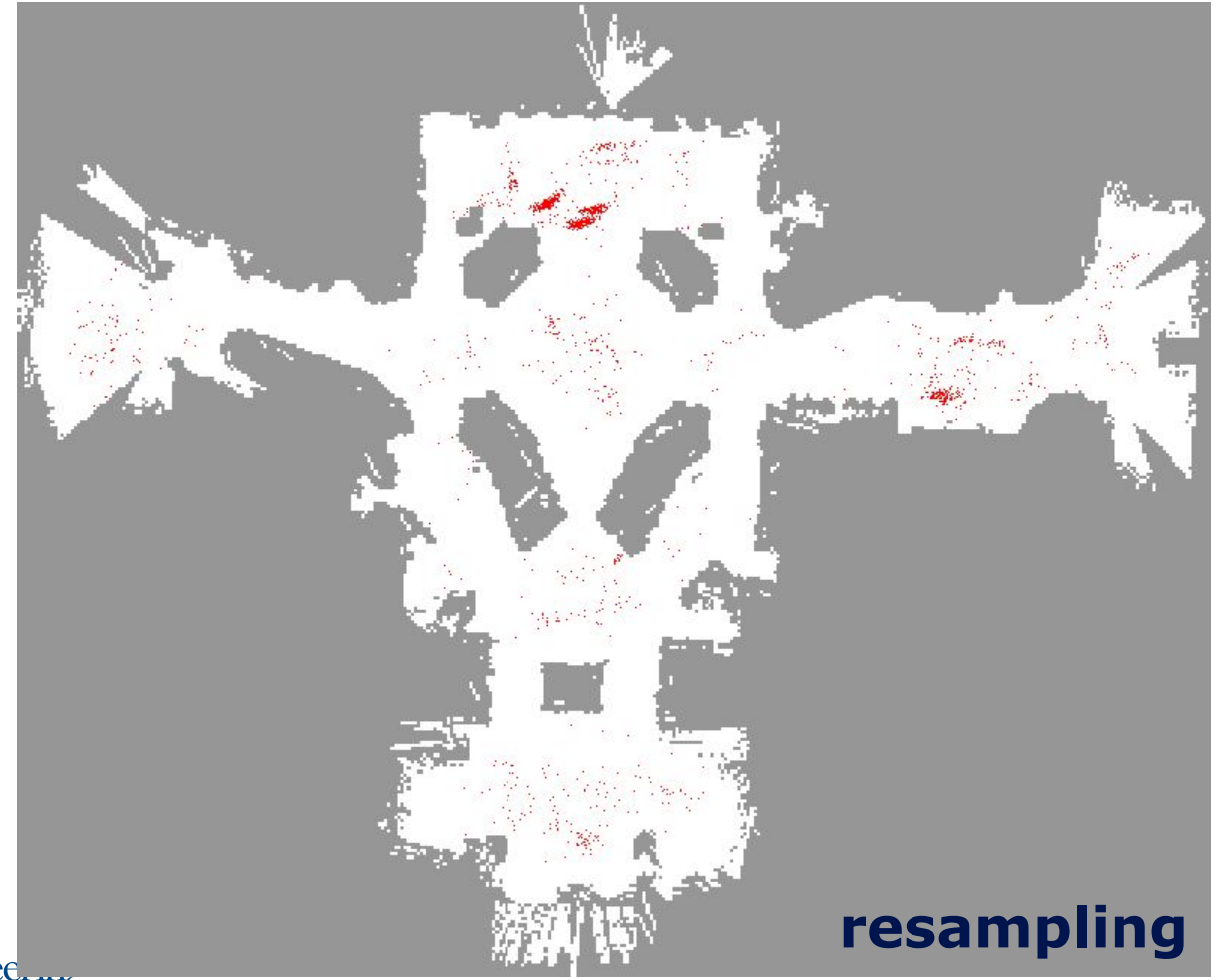
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

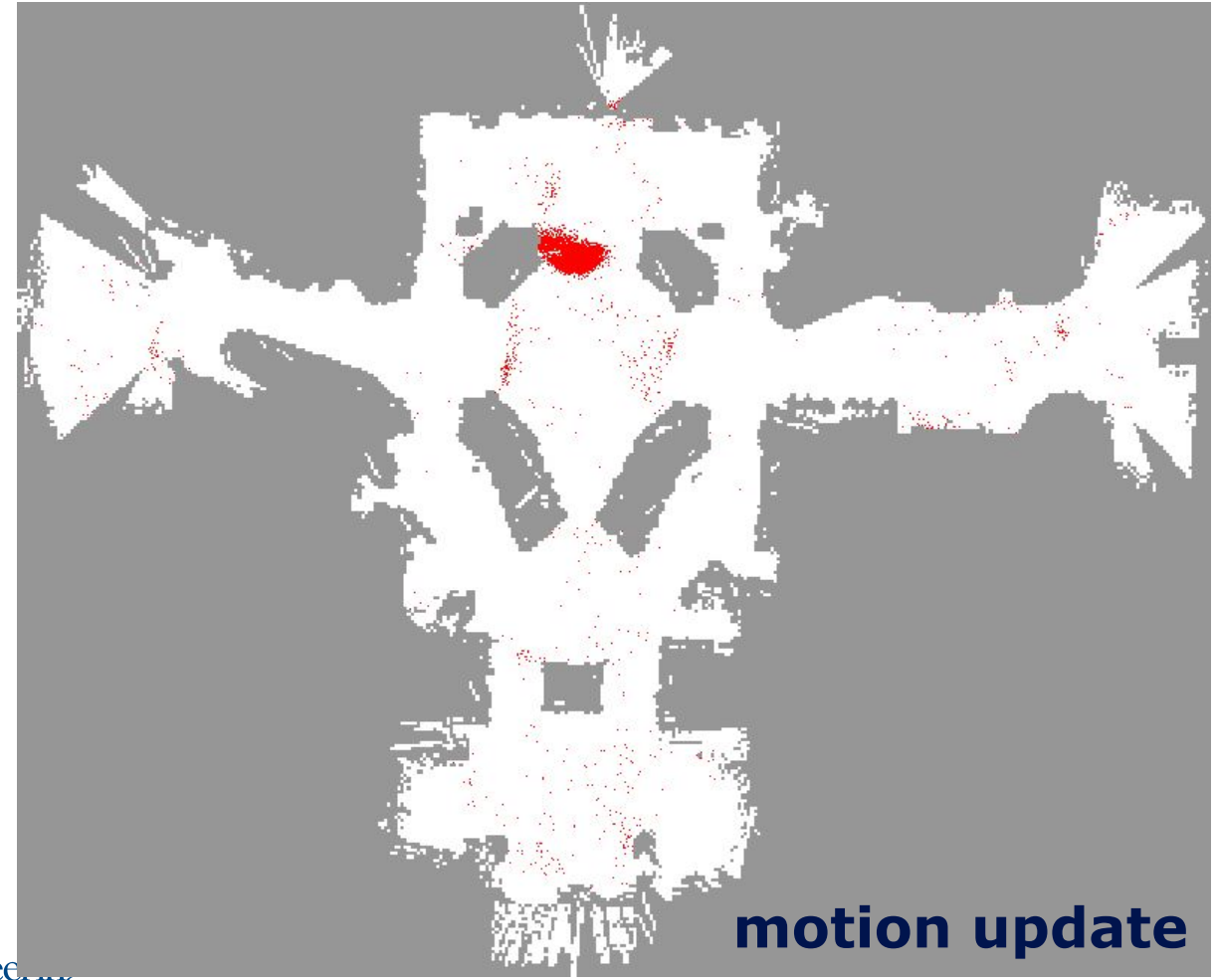
Penn Engineering

Courtesy: Thrun, Burgard, Fox



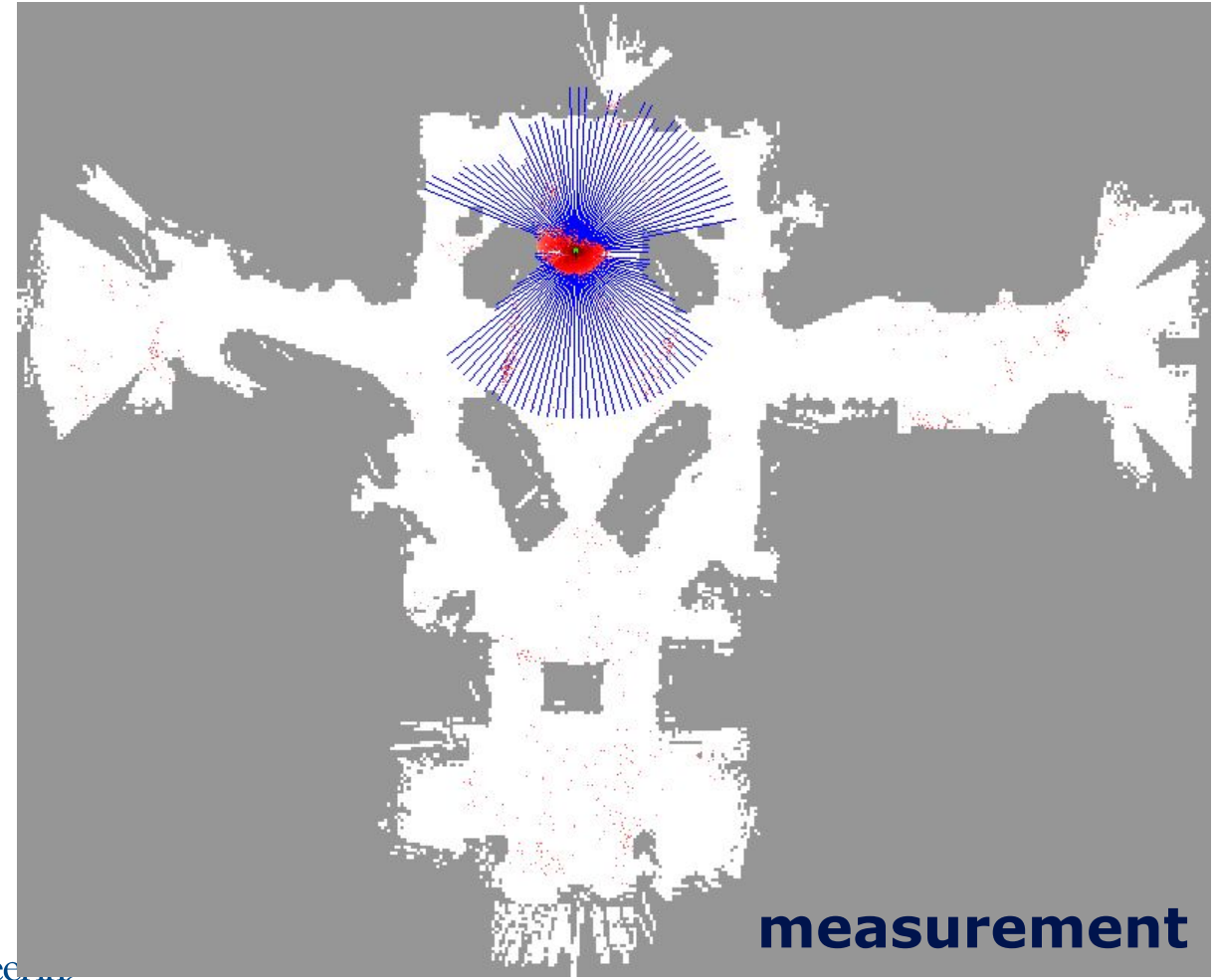
Penn Engineering

Courtesy: Thrun, Burgard, Fox



Penn Engineering

Courtesy: Thrun, Burgard, Fox

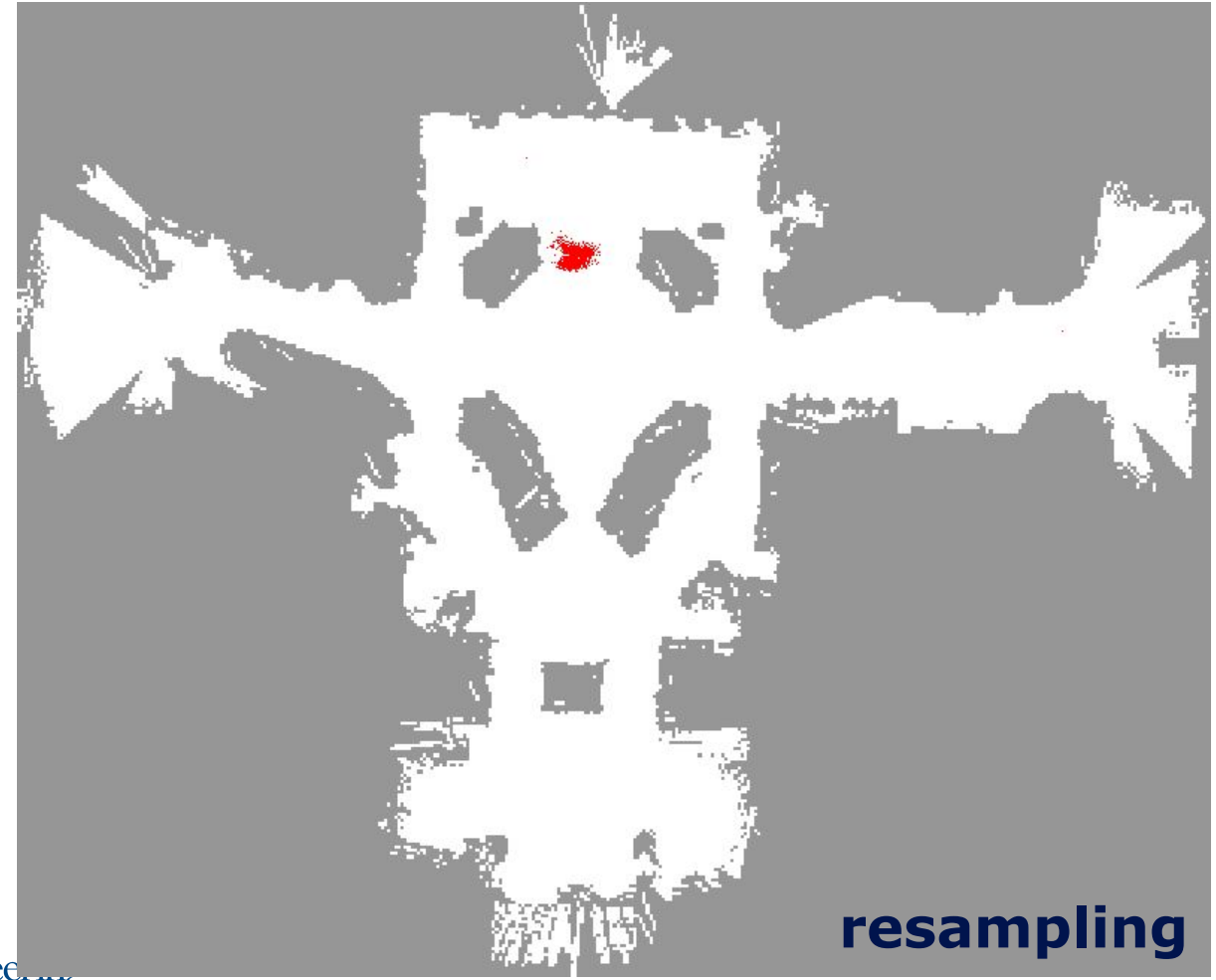


Penn Engineering

Courtesy: Thrun, Burgard, Fox

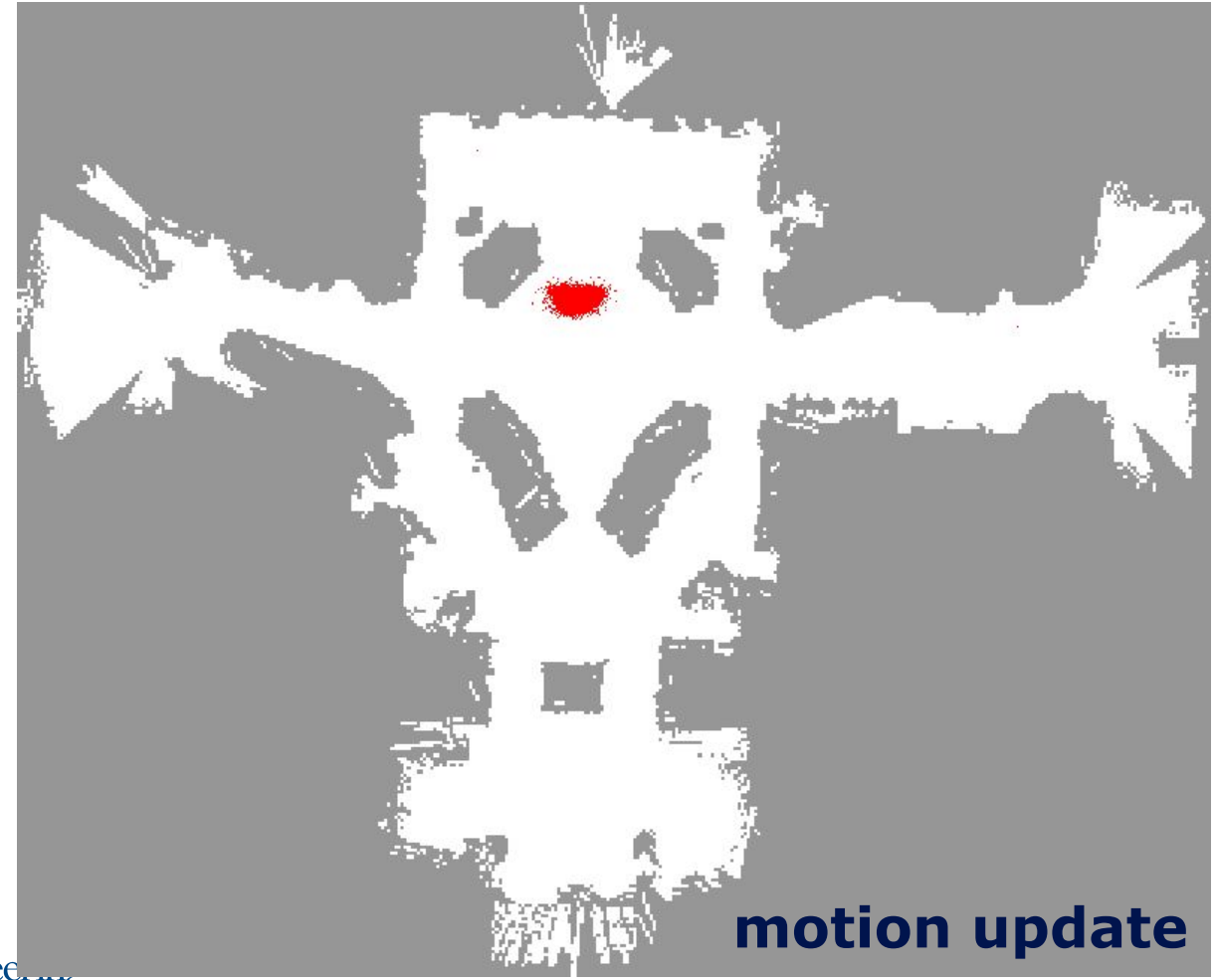
Penn Engineering

Courtesy: Thrun, Burgard, Fox



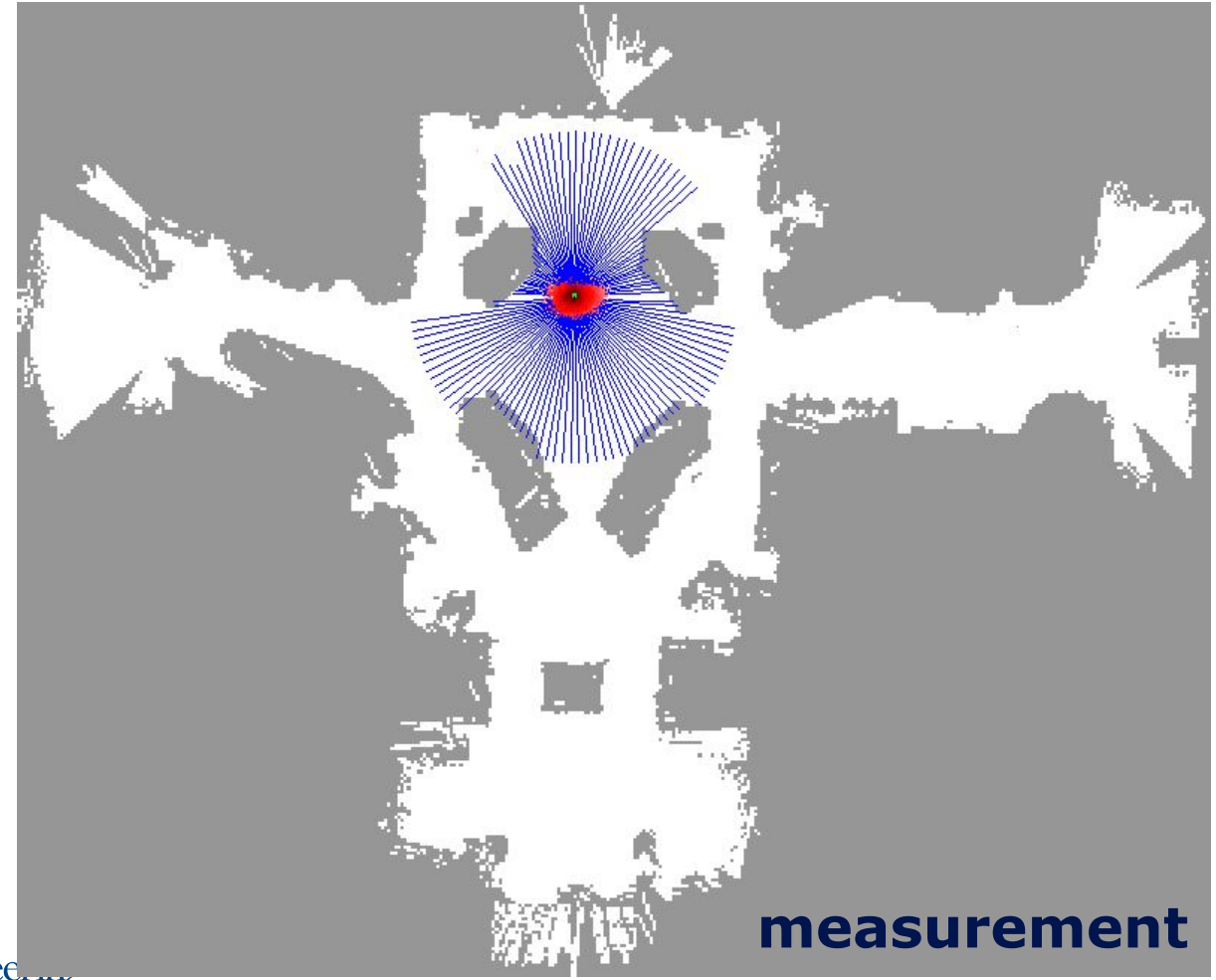
Penn Engineering

Courtesy: Thrun, Burgard, Fox



Penn Engineering

Courtesy: Thrun, Burgard, Fox

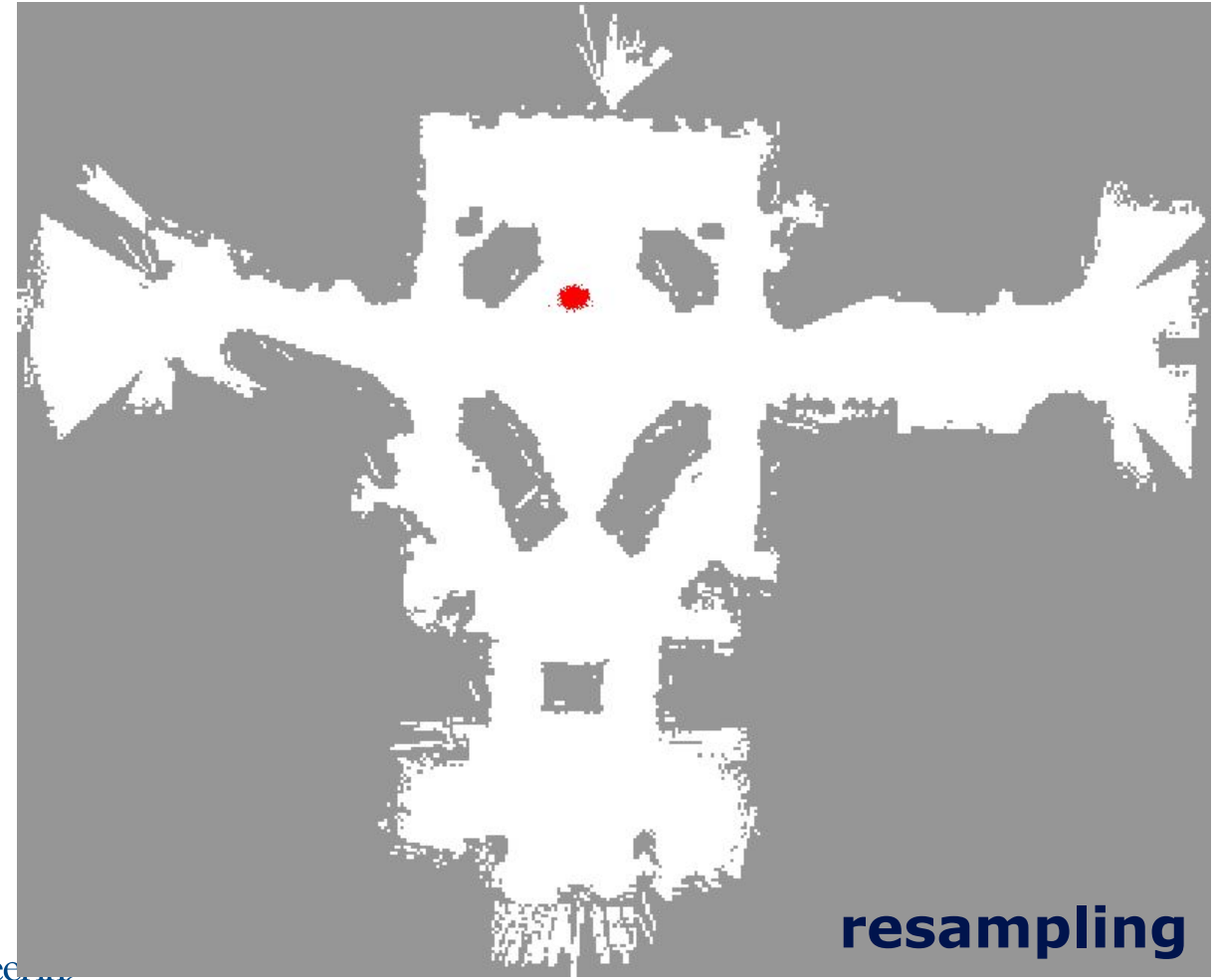


Penn Engineering

Courtesy: Thrun, Burgard, Fox

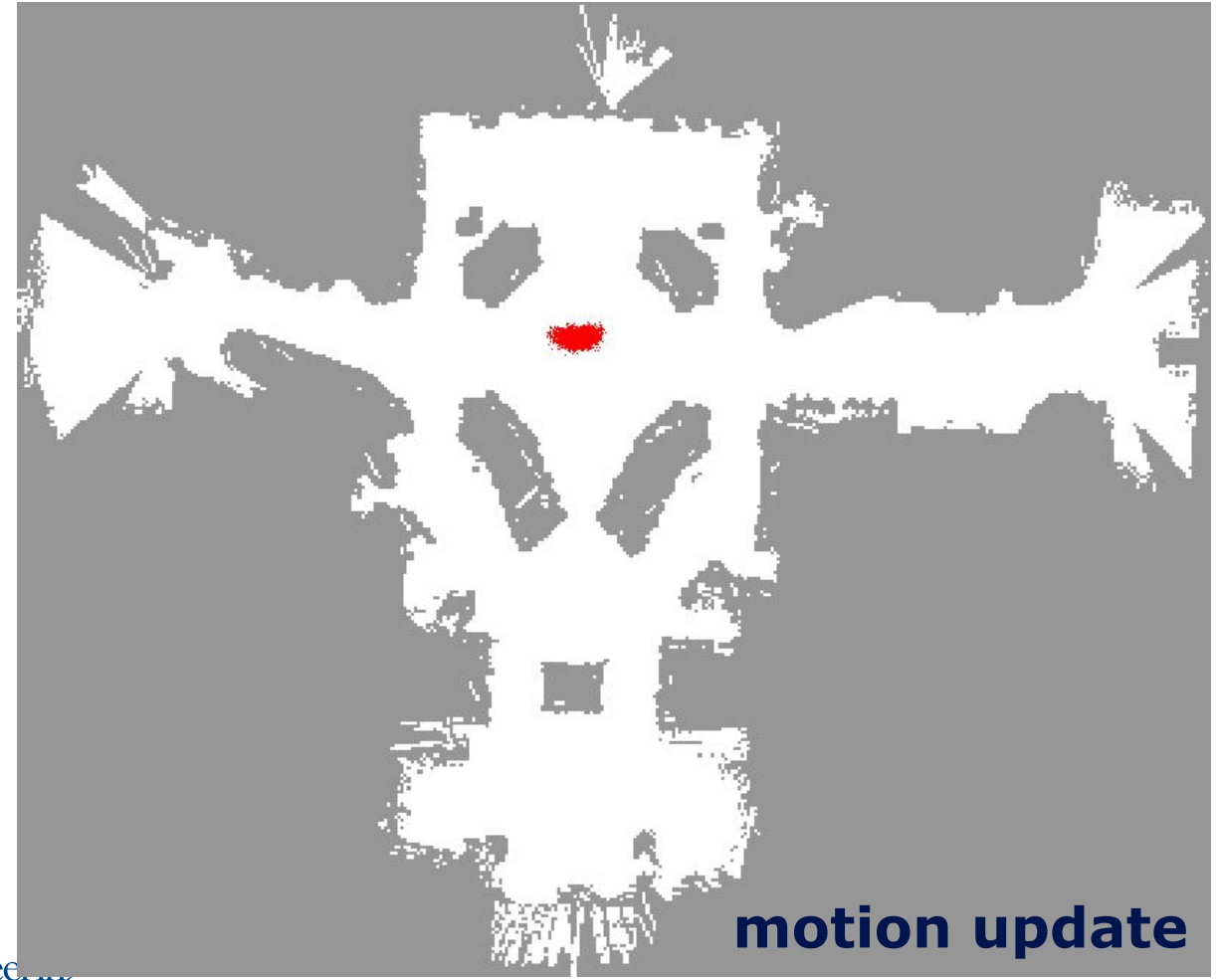
Penn Engineering

Courtesy: Thrun, Burgard, Fox



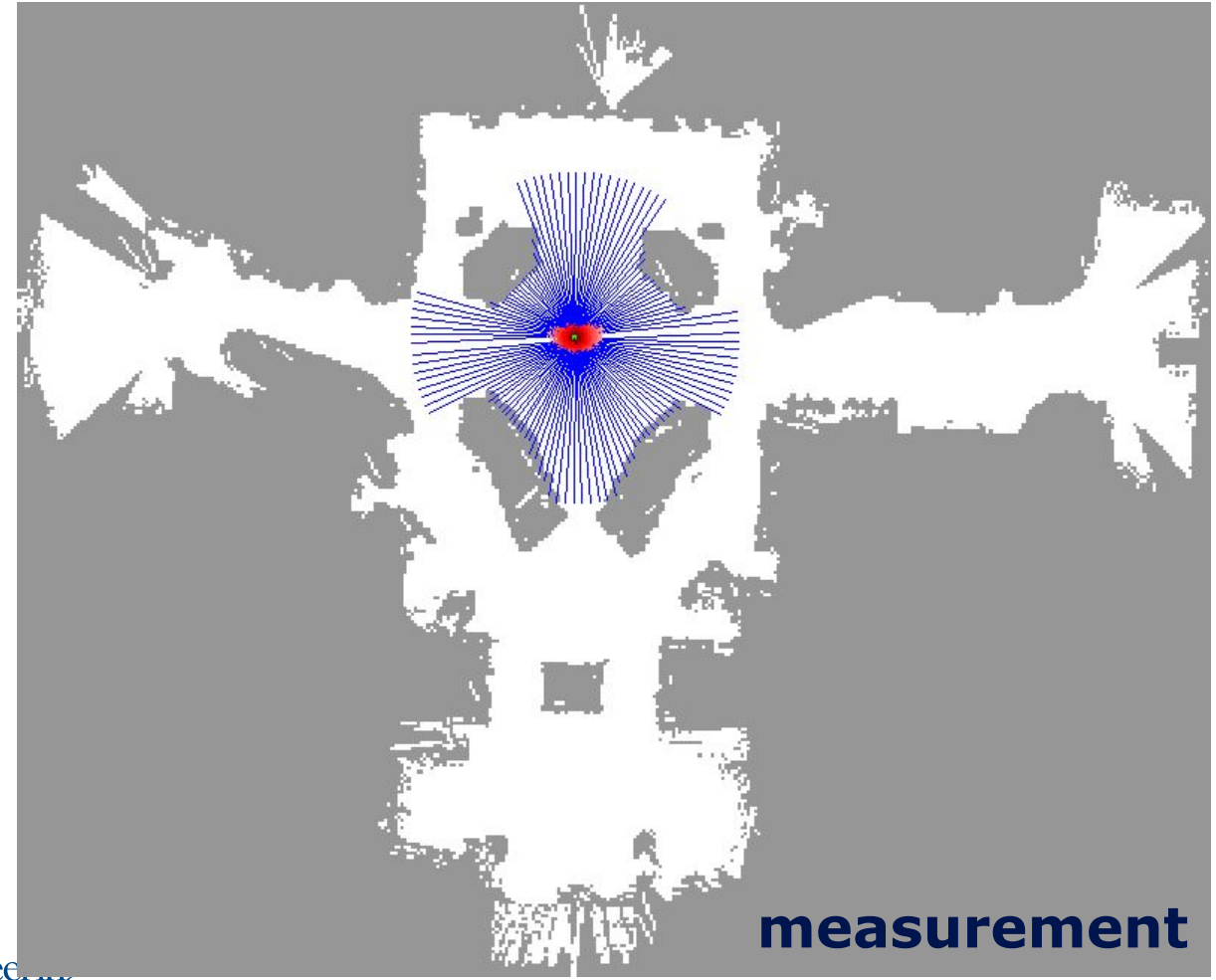
Penn Engineering

Courtesy: Thrun, Burgard, Fox



Penn Engineering

Courtesy: Thrun, Burgard, Fox



Penn Engineering

Courtesy: Thrun, Burgard, Fox

Particle Filters: Tuning

Tuning Parameters

Find in `localize.launch`

- Number of Particles: 250-10000
- Angle Step: Downsamples the range measurements
- Squash factor: Smooths particle weights
- Max range: 10-30 meters, good for getting rid of useless measurements with no returns, floor etc.
- Method: Use RM-GPU

ROS: Map Server

Edit `map_server.launch`

```
<launch>
  <arg name="map" default="$(find particle_filter)/maps/your_map.yaml"/>
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map)" />
</launch>
```

Then add `your_map.pgm` and `your_map.yaml` to `~/path/to/your_workspace/src/particle_filter/maps/`

AMCL Parameters

`min_particles`

Default: 100

The minimum number of particles to be used for calculating correlation

`max_particles`

Default: 500

The maximum number of particles to be used for calculating correlation

AMCL Parameters

`update_min_d`

Default: 0.2m

The minimum translation movement required by the vehicle before an pose update is published

`update_min_a`

Default: $\pi/6$ radians

The minimum angular movement required by the vehicle before an pose update is published

AMCL Parameters

`initial_pose_x`	Default: 0
`initial_pose_y`	Default: 0
`initial_pose_a`	Default: 0

The initial mean position of the particles to initialize the particle filter

AMCL Parameters

`initial_cov_xx`	Default: 0
`initial_cov_yy`	Default: 0
`initial_cov_aa`	Default: 0

The covariance of particles distributed around the mean

Summary – Particle Filters

- Particle filters are non-parametric, recursive Bayes filters
- Posterior is represented by a set of weighted samples
- Proposal to draw the samples for t+1
- Weight to account for the differences between the proposal and the target

The art is to design appropriate motion and sensor models

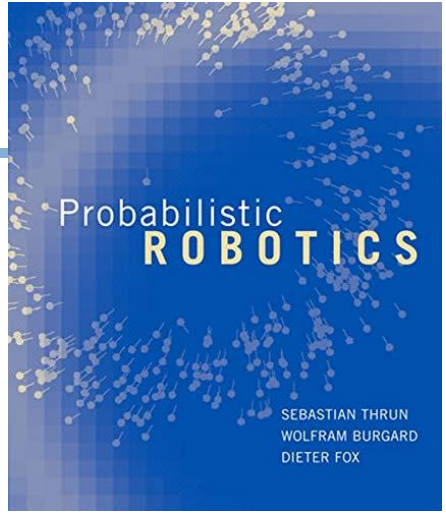
Summary: Particle Filter Localization

- Particles are propagated according to the motion model
- They are weighted according to the likelihood of the observation
- Called: Monte-Carlo localization (MCL)
- MCL is the gold standard for indoor mobile robot localization today

Additional Resources & Hands-on Tutorial

References

Awesome free book →



S. Thrun, W. Burgard. "Probabilistic Robotics." Chapter 4 and Chapter 8.
http://www.probabilistic-robotics.org/

S. Thrun. "Artificial Intelligence for Robotics, Lesson 3." Udacity.
https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

S. Thrun, D. Fox, W. Burgard and F. Dellaert. "Robust Monte Carlo Localization for Mobile Robots." Artificial Intelligence Journal. 2001.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.6016&rep=rep1&type=pdf

D. Fox, W. Burgard, and S. Thrun. "Markov localization for mobile robots in dynamic environments," Journal of Artificial Intelligence Research , vol. 11, pp. 391427, 1999.
http://www.jair.org/media/616/live-616-1819-jair.pdf

D. Fox. "KLD-sampling: Adaptive particle filters," Advances in Neural Information Processing Systems 14 (NIPS), Cambridge, MA, 2002. MIT Press.
https://papers.nips.cc/paper/1998-kld-sampling-adaptive-particle-filters.pdf

D. Bagnell "Particle Filters: The Good, The Bad, The Ugly"
http://www.cs.cmu.edu/~16831-f12/notes/F14/16831_lecture05_gseyfarth_zbatts.pdf

C. Walsh, S. Karaman. "CDDT: Fast Approximate 2D Ray Casting for Accelerated Localization." Arxiv, 2017.
http://arxiv.org/abs/1705.01167