ECE276A: Sensing & Estimation in Robotics
Lecture 9: Particle Filter SLAM

Instructor:
Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:
Qiaojun Feng: qif007@eng.ucsd.edu
Arash Asgharivaskasi: aasghari@eng.ucsd.edu
Thai Duong: tduong@eng.ucsd.edu
Yiran Xu: y5xu@eng.ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Simultaneous Localization & Mapping (SLAM)

- Chicken-and-egg problem:
  - **Mapping**: given the robot state trajectory $\mathbf{x}_{0:T}$, build a map $\mathbf{m}$ of the environment
  - **Localization**: given a map $\mathbf{m}$ of the environment, localize the robot and estimate its trajectory $\mathbf{x}_{0:T}$
- SLAM is a parameter estimation problem for $\mathbf{x}_{0:T}$ and $\mathbf{m}$
- Given a dataset of the robot inputs $\mathbf{u}_{0:T-1}$ and observations $\mathbf{z}_{0:T}$, maximize the data likelihood conditioned on the parameters (MLE) or the posterior likelihood of the parameters given the data (MAP) or use Bayesian Inference to maintain the posterior likelihood of the parameters given the data:
  - **MLE**: $\max_{\mathbf{x}_{0:T},\mathbf{m}} \log\ p(\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1} \mid \mathbf{x}_{0:T}, \mathbf{m})$
  - **MAP**: $\max_{\mathbf{x}_{0:T},\mathbf{m}} \log\ p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$
  - **BI**: maintain $p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$

Simultaneous Localization & Mapping (SLAM)

- Solutions to the SLAM problem exploit the decomposition of the joint pdf due to the Markov assumptions:

$$p(\mathbf{x}_{0:T}, \mathbf{m}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_{0|0}(\mathbf{x}_0, \mathbf{m})}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}}$$

- The MLE formulation becomes:

$$\max_{\mathbf{x}_{0:T}, \mathbf{m}} \sum_{t=0}^{T} \log p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m}) + \sum_{t=1}^{T} \log p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

- The MAP formulation is equivalent with the addition of a prior $\log p_{0|0}(\mathbf{x}_0, \mathbf{m})$ to the objective function

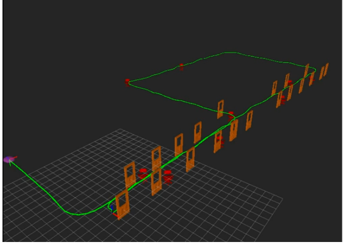
- The BI formulation uses Bayesian smoothing to maintain $p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:t-1})$

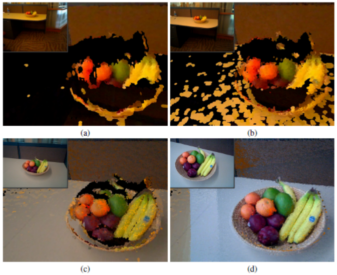
Simultaneous Localization & Mapping (SLAM)

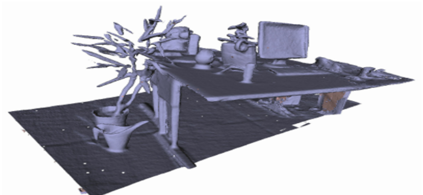
- Early SLAM approaches were based on simplified versions of the MLE/MAP/BI formulations:
  - Bayes filtering to maintain only $p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$
  - EM treating $\mathbf{x}_t$ as a hidden variable. Given an inital map $\mathbf{m}^{(0)}$, e.g., obtained from the first observation, iterate:
    - E: Estimate the distribution of $\mathbf{x}_t$ given $\mathbf{m}^{(i)}$
    - M: Update $\mathbf{m}^{(i+1)}$ by maximizing (over $\mathbf{m}$) the log-likelihood of the measurements conditioned on $\mathbf{x}_t$ and $\mathbf{m}$
- The implementation of any of the SLAM approaches depends on the particular representations of the robot states $\mathbf{x}_t$, map $\mathbf{m}$, observations $\mathbf{z}_t$, and control inputs $\mathbf{u}_t$

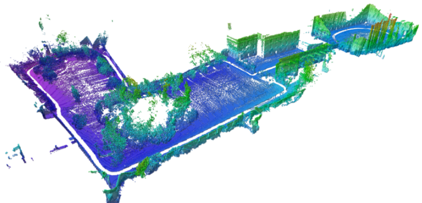
Map Representations

- **Landmark-based**: a collection of objects, each having a position, orientation, and object class
- **Occupancy grid**: a discretization of space into cells with a binary occupancy model
- **Surfels**: a collection of oriented discs containing photometric information
- **Polygonal mesh**: a collection of points and connectivity information among them, forming polygons







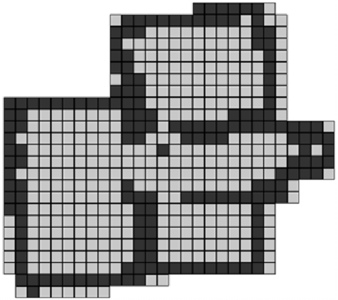


Occupancy Grid Map

- The environment is divided into a regular grid with n cells

- **Occupancy grid map**: a vector $\mathbf{m} \in \mathbb{R}^n$, whose i-th entry indicates whether the i-th cell is free ($m_i = -1$) or occupied ($m_i = 1$)

- The cells are called pixels (pictures (pics) elements) in 2D and voxels (volumes elements) in 3D



- If the map $\mathbf{m}$ is unknown and needs to be estimated from observations $\mathbf{z}_{0:t}$ obtained from robot states $\mathbf{x}_{0:t}$, a probability mass function $p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ is maintained over time

- Occupancy grid mapping algorithms usually assume that the occupancy grid cells are independent conditioned on the robot trajectory and model the distribution $p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ of each cell m_i

Popular Occupancy Grid SLAM Algorithms

- **Fast SLAM** (Montemerlo et al., AAAI'02)
  - exploits that the occupancy grid cells are independent conditioned on the robot trajectory:

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}) = p(\mathbf{x}_{0:t} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}) \prod_i p(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$$

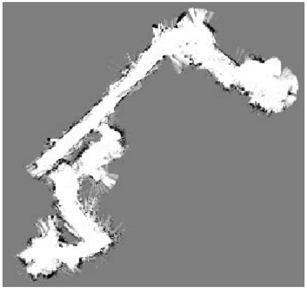
  - uses a particle filter to maintain the robot trajectory pdf and log-odds mapping to maintain a probabilistic map for every particle
- **Kinect Fusion** (Newcombe et al., ISMAR'11)
  - matches consecutive RGBD point clouds using the iterative closest point (ICP) algorithm
  - updates a grid discretization of the truncated signed distance function (TSDF) representing the scene surface via weighted averaging

Popular Landmark-based SLAM Algorithms

- **Rao-Blackwellized Particle Filter** uses particles for $\mathbf{x}_{0:t}$ and Gaussian distributions for the landmark positions $\mathbf{m}$
- **Kalman Filter** uses Gaussian distributions both for the robot poses $\mathbf{x}_{0:t}$ and the landmark positions $\mathbf{m}$
- **Factor Graphs SLAM**
  - Estimates the whole robot trajectory $\mathbf{x}_{0:t}$ using the MAP formulation
  - The log observation and motion models are modelled as nonlinear functions subject to additive Gaussian noise
  - The motion and observation log-likelihoods are proportional to the Mahalonobis distance
  - This leads to a **sparse** (due to the Markov assumptions), **nonlinear** (due to the motion and observation models) **least-squares** (due to the Mahalonobis distance) optimization problem
  - The problem can be solved using the Gauss-Newton descent algorithm (an approximation to Newton's method that avoids computing the Hessian)

Occupancy Grid Mapping

- Given the robot trajectory $\mathbf{x}_{0:t}$ and a sequence of observations $\mathbf{z}_{0:t}$, build an occupancy grid map $\mathbf{m}$ of the environment



- Since the map is unknown and the measurements are uncertain, we need to maintain a pdf $p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$
- Model the map cells m_i as **independent** Bernoulli random variables

$$m_i = \begin{cases} \text{Occupied (1)} & \text{with prob. } \gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \\ \text{Free (-1)} & \text{with prob. } 1 - \gamma_{i,t} \end{cases}$$

- To have a probabilistic map representation, we just need to keep a vector of occupancy probabilities $\boldsymbol{\gamma}_t$ over time

Occupancy Grid Mapping

- How do we update the map distribution $\gamma_{i,t}$ over time?
- **Bayes Rule**:

$$\begin{aligned}
\gamma_{i,t} &= p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \frac{1}{\eta_t} p_h(\mathbf{z}_t \mid m_i = 1, \mathbf{x}_t) p(m_i = 1 \mid \mathbf{z}_{0:t-1}, \mathbf{x}_{0:t-1}) \\
&= \frac{1}{\eta_t} p_h(\mathbf{z}_t \mid m_i = 1, \mathbf{x}_t) \gamma_{i,t-1} \\
(1 - \gamma_{i,t}) &= p(m_i = -1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \frac{1}{\eta_t} p_h(\mathbf{z}_t \mid m_i = -1, \mathbf{x}_t)(1 - \gamma_{i,t-1})
\end{aligned}$$

- The **odds ratio** of a binary random variable m_i updated over time via Bayes rule and measurements $\mathbf{z}_{0:t}$ is:

$$\begin{aligned}
o(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) &:= \frac{p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})}{p(m_i = -1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})} = \frac{\gamma_{i,t}}{1 - \gamma_{i,t}} \\
&= \underbrace{\frac{p_h(\mathbf{z}_t \mid m_i = 1, \mathbf{x}_t)}{p_h(\mathbf{z}_t \mid m_i = -1, \mathbf{x}_t)}}_{g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t)} \underbrace{\frac{\gamma_{i,t-1}}{1 - \gamma_{i,t-1}}}_{o(m_i \mid \mathbf{z}_{0:t-1}, \mathbf{x}_{0:t-1})}
\end{aligned}$$

Occupancy Grid Mapping

- Estimating the pdf of m_i conditioned on $\mathbf{z}_{0:t}$ is equivalent to accumulating the **log-odds ratio**:

$$\begin{aligned}\lambda(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) :&= \log o(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \log \left(g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) o(m_i \mid \mathbf{z}_{0:t-1}, \mathbf{x}_{0:t-1})\right)\\ &= \lambda(m_i \mid \mathbf{z}_{0:t-1}, \mathbf{x}_{0:t-1}) + \log g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t)\\ &= \lambda(m_i) + \sum_{s=0}^{t} \log g_h(\mathbf{z}_s \mid m_i, \mathbf{x}_s)\end{aligned}$$

- Probabilistic occupancy grid mapping reduces to keeping track of the cell log-odds $\lambda_{i,t} := \lambda(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$:

$$\lambda_{i,t} = \lambda_{i,t-1} + \Delta\lambda_{i,t-1} \quad \leftarrow \text{Measurement ``trust''}$$

- To perform the update above, we just need to specify the observation model log-odds ratio $\Delta\lambda_{i,t} := \log g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t)$

Inverse Observation Model

- Using Bayes rule again, we can simplify the observation log-odds ratio:

$$g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) = \frac{p_h(\mathbf{z}_t \mid m_i = 1, \mathbf{x}_t)}{p_h(\mathbf{z}_t \mid m_i = -1, \mathbf{x}_t)} = \frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t) p(m_i = -1)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t) p(m_i = 1)}$$

$$\Delta\lambda_{i,t} = \log g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) = \log \frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)} - \lambda(m_i)$$

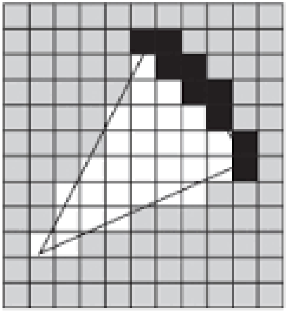
- The first term is an **inverse observation model**

- It specifies how much we trust our observation $\mathbf{z}_t$, e.g., if $\mathbf{z}_t$ indicates that m_i is occupied, what is the log ratio of true positive vs false positive:

$$\frac{p(m_i = 1 \mid m_i \text{ is observed occupied at time } t)}{p(m_i = -1 \mid m_i \text{ is observed occupied at time } t)} = \frac{80\%}{20\%} = 4$$

- The second term $\lambda(m_i) = \log \frac{p(m_i=1)}{p(m_i=-1)}$ is just a prior occupancy log-odds ratio and may be chosen as 0 (unknown) or > 0 (optimistic about free space)

Lidar-based Occupancy Grid Mapping



- Maintain a grid of the map log-odds $\lambda_{i,t}$
- Given a new laser scan $\mathbf{z}_{t+1}$, transform it to the world frame using the robot pose $\mathbf{x}_{t+1}$
- Determine the cells that the lidar beams pass through (e.g., using Bresenham's line rasterization algorithm)
- For each observed cell i, decrease the log-odds if it was observed free or increase the log-odds if the cell was observed occupied:

$$\lambda_{i,t+1} = \lambda_{i,t} + \log g_h(\mathbf{z}_{t+1} \mid m_i, \mathbf{x}_{t+1})$$

- Constrain $\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MAX}$ to avoid overconfident estimation
- May introduce a decay on $\lambda_{i,t}$ to handle changing maps
- The map pmf $\gamma_{i,t}$ can be recovered from the log-odds $\lambda_{i,t}$:

$$\gamma_{i,t} = p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = 1 - \frac{1}{1 + \exp\left(\lambda_{i,t}\right)}$$

Markov Localization in Occupancy Grid Maps

- Given an occupancy grid map $\mathbf{m}$, a sequence of inputs $\mathbf{u}_{0:t-1}$, and a sequence of observations $\mathbf{z}_{0:t}$, infer the robot state $\mathbf{x}_t$
- **Approach**:
  - Use a delta-mixture to represent the pdf of the robot state at time t:

$$p_{t|t}(\mathbf{x}_t) := p(\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}) \approx \sum_{k=1}^{N} \alpha_{t|t}^{(k)} \delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)}\right)$$

  - Each $\boldsymbol{\mu}_{t|t}^{(k)}$ is a hypothesis on the robot state $\mathbf{x}_t$ with confidence $\alpha_{t|t}^{(k)}$
  - Use the particle filter to propagate the pdf over time
  - **Prediction step**: use the motion model
  - **Update step**: use the observation model

Lidar-based Localization with a Differential-drive Robot

- Each particle $\boldsymbol{\mu}_{t|t}^{(k)} \in \mathbb{R}^3$ represents a possible robot 2-D position (x, y) and orientation θ
- **Prediction step**: for every particle $\boldsymbol{\mu}_{t|t}^{(k)}$, $k = 1, \ldots, N$ compute:

$$\boldsymbol{\mu}_{t+1|t}^{(k)} = f\left(\boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t + \boldsymbol{\epsilon}_t\right)$$

  where f is the differential-drive motion model, $\mathbf{u}_t = (v_t, \omega_t)$ is the linear and angular velocity input (either known or obtained from the Encoders and IMU), and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathcal{E})$ is a 2-D Gaussian motion noise

- **Update step**:
  - Transform the scan $\mathbf{z}_{t+1}$ to the world frame using $\boldsymbol{\mu}_{t+1|t}^{(k)}$ for $k = 1, \ldots, N$ and find all cells $\mathbf{y}_{t+1}^{(k)}$ in the grid corresponding to the scan
  - Update the particle weights using the laser correlation model:

$$p_h(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{m}) \propto \exp\left(\mathbf{corr}\left(\mathbf{y}_{t+1}^{(k)}, \mathbf{m}\right)\right)$$

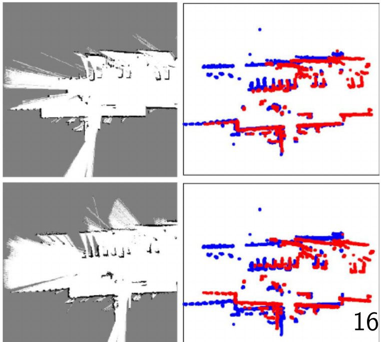
Laser Correlation Model

- A model for a laser scan $\mathbf{z}$ obtained from sensor pose $\mathbf{x}$ in an occupancy map $\mathbf{m}$ obtained by modeling the correlation between $\mathbf{z}$ and $\mathbf{m}$
- **Occupancy grid map**: a grid with free $(m_i = -1)$ and occupied $(m_i = 1)$ cells
- **Laser Correlation Model**:
  1. Transform the scan $\mathbf{z}$ to the world frame using $\mathbf{x}$ and find all points $\mathbf{y}$ in the grid that correspond to the scan
  2. Let the observation model be proportional to the similarty $\mathbf{corr}(\mathbf{y}, \mathbf{m})$ between the transformed scan $\mathbf{y}$ and the grid $\mathbf{m}$
- The correlation is large if $\mathbf{y}$ and $\mathbf{m}$ agree:

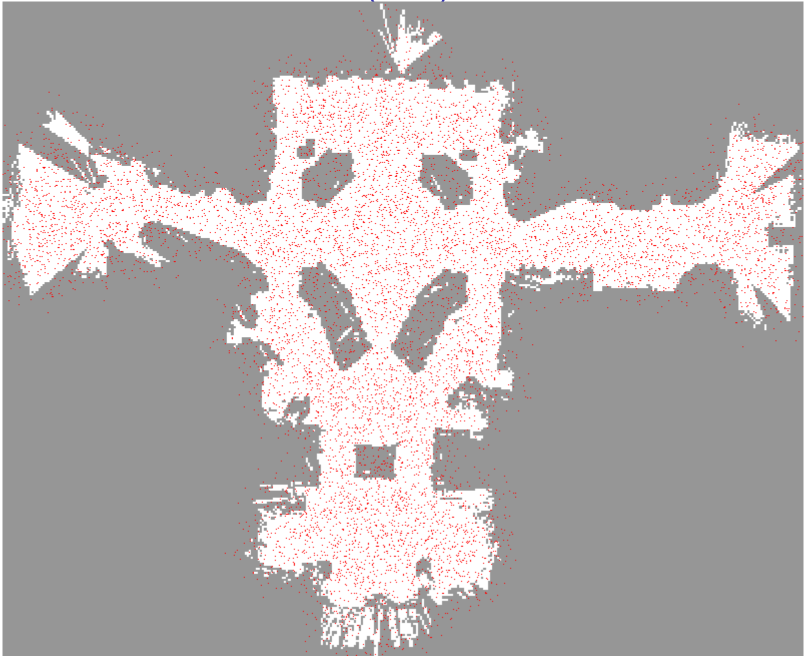
$$\mathbf{corr}(y, m) := \sum_i \mathbb{1}\{m_i = y_i\}$$

- The weights can be converted to probabilities via the **softmax** function:

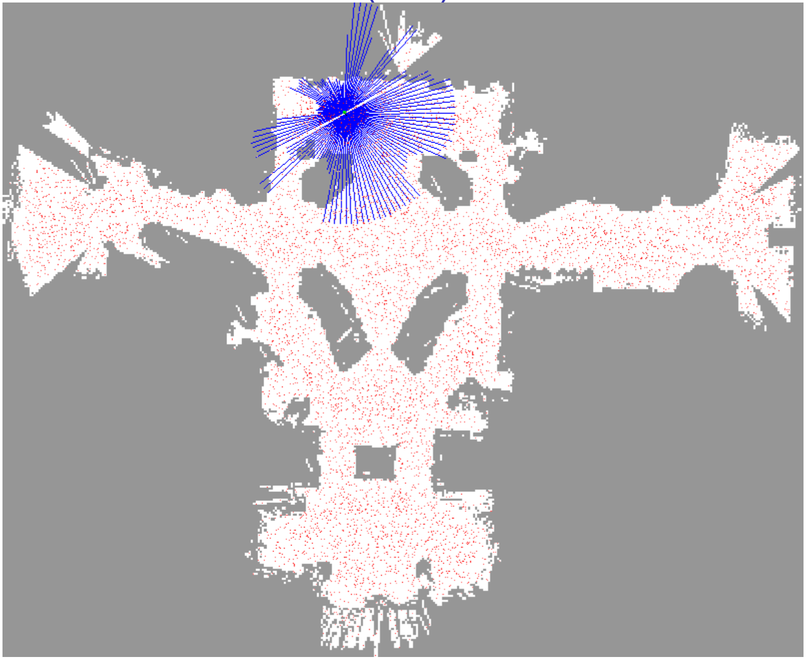
$$p_h(\mathbf{z} \mid \mathbf{x}, \mathbf{m}) = \frac{e^{\mathbf{corr}(\mathbf{y},\mathbf{m})}}{\sum_{\mathbf{v}} e^{\mathbf{corr}(\mathbf{v},\mathbf{m})}} \propto e^{\mathbf{corr}(\mathbf{y},\mathbf{m})}$$



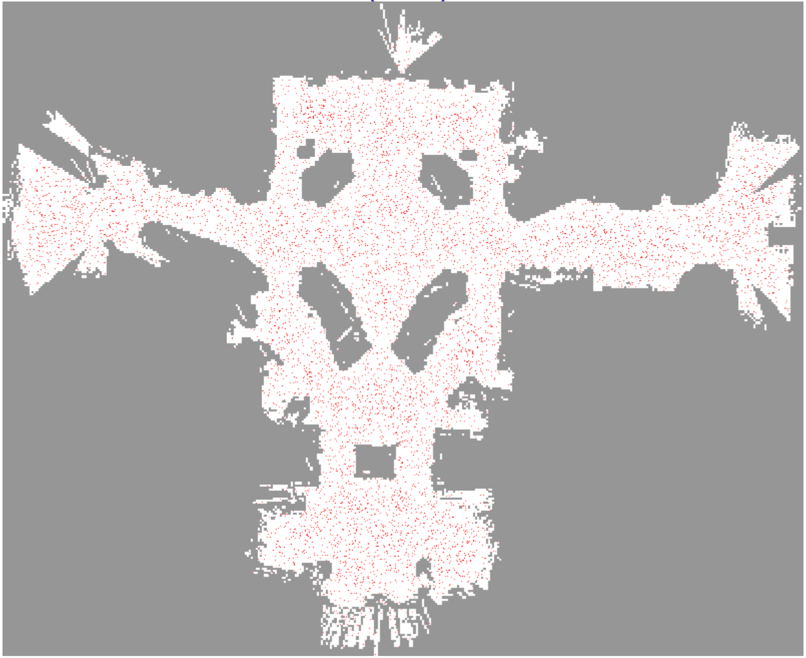
Particle Filter Localization (2-D)



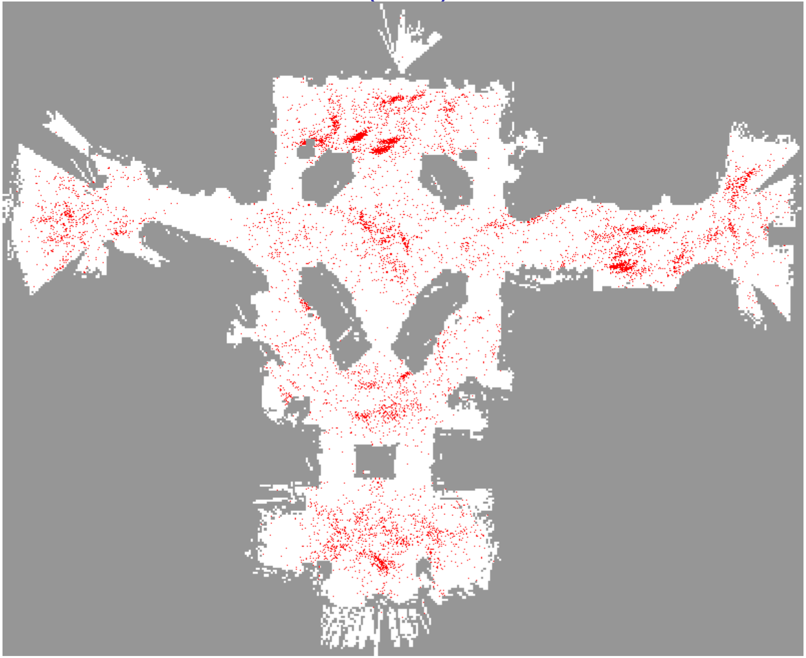
Particle Filter Localization (2-D)



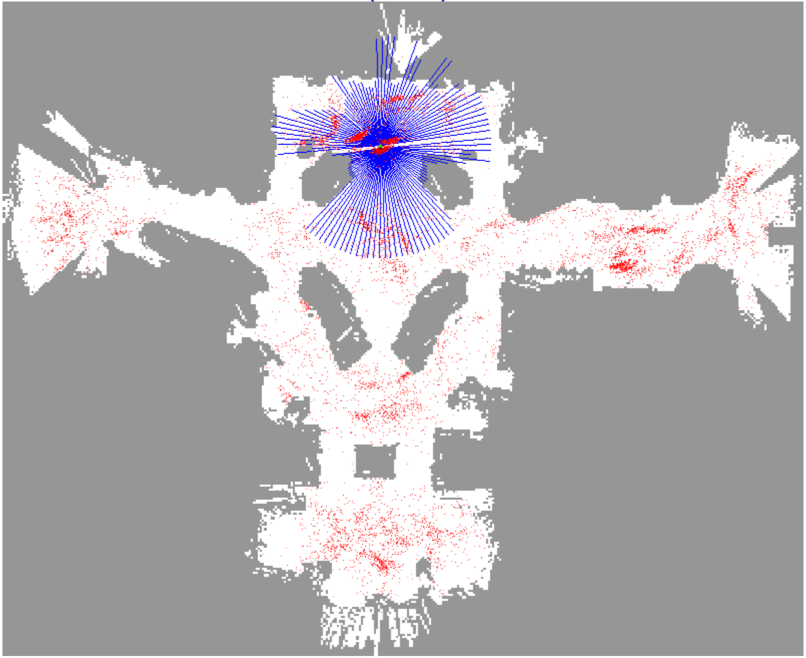
Particle Filter Localization (2-D)



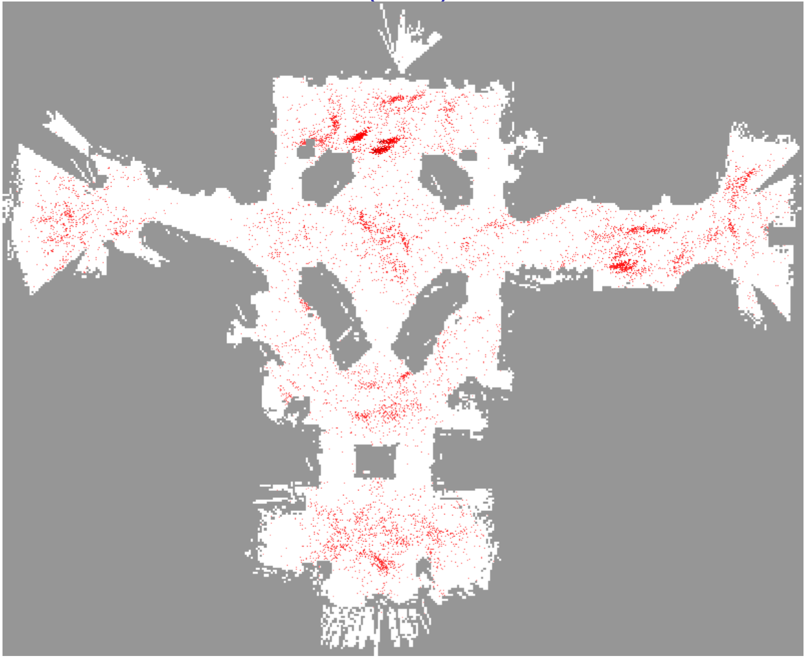
Particle Filter Localization (2-D)



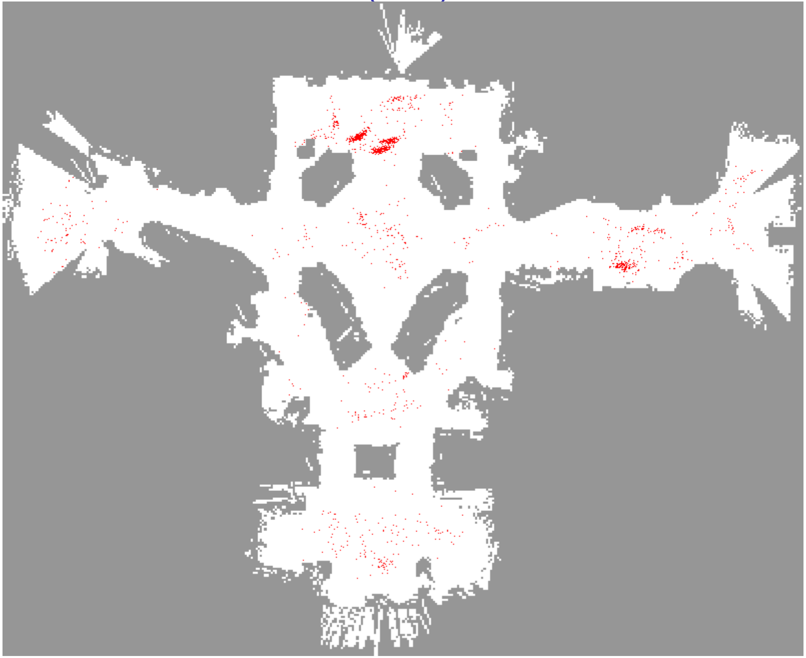
Particle Filter Localization (2-D)



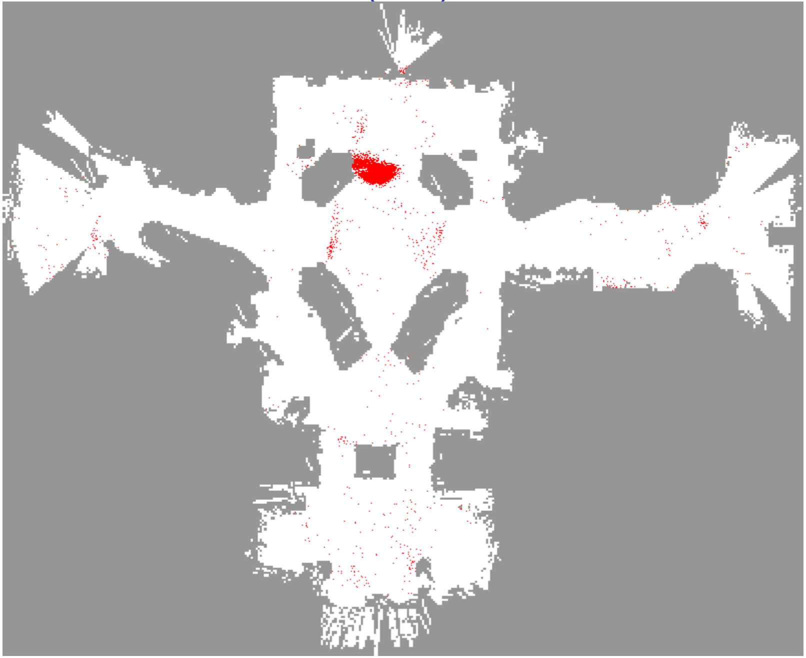
Particle Filter Localization (2-D)



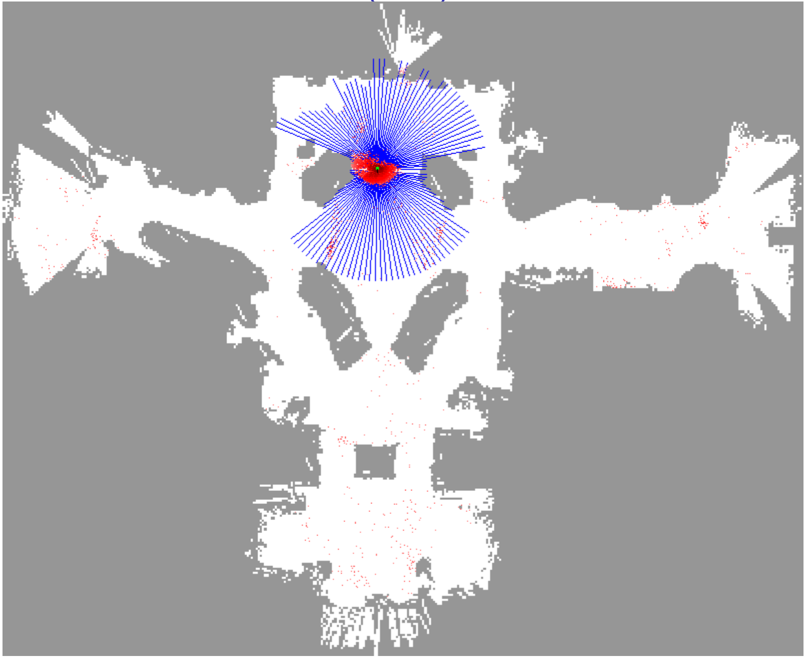
Particle Filter Localization (2-D)



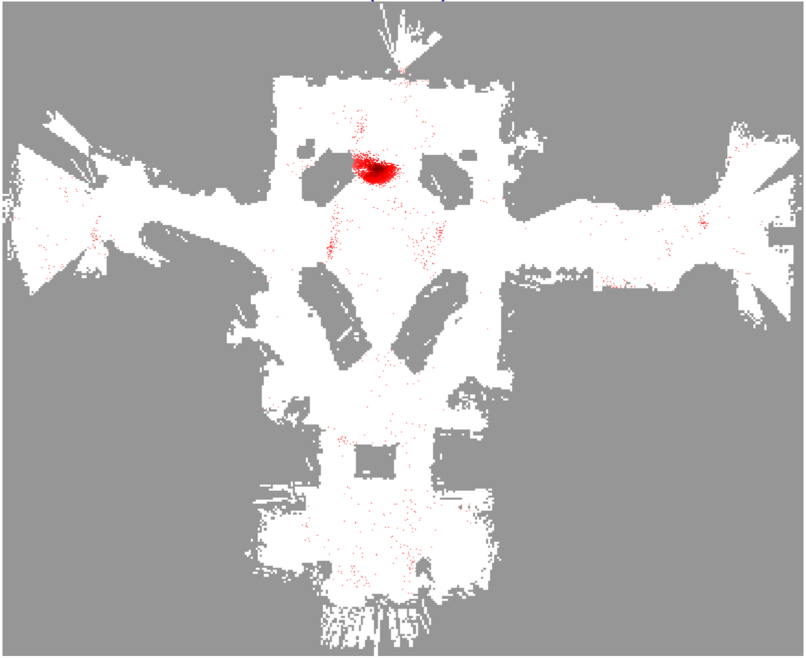
Particle Filter Localization (2-D)



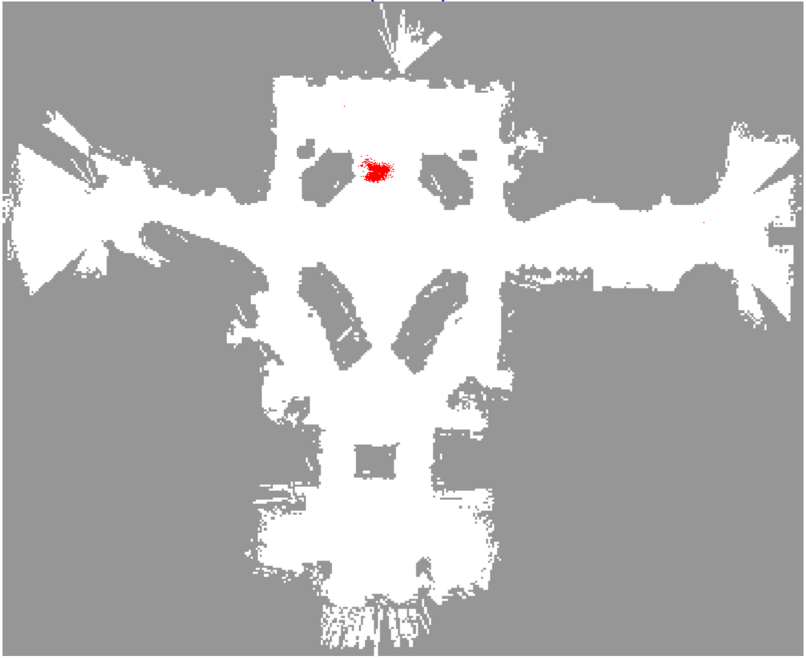
Particle Filter Localization (2-D)



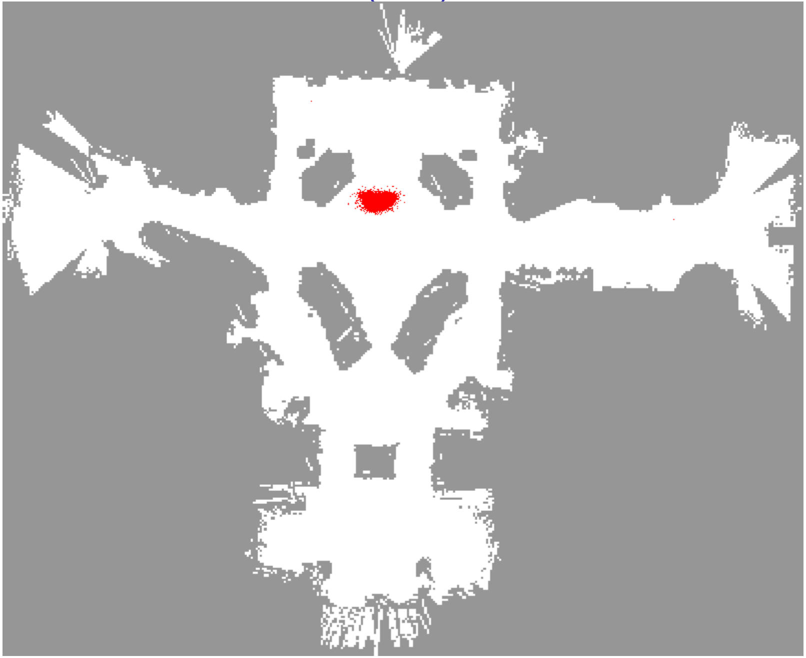
Particle Filter Localization (2-D)



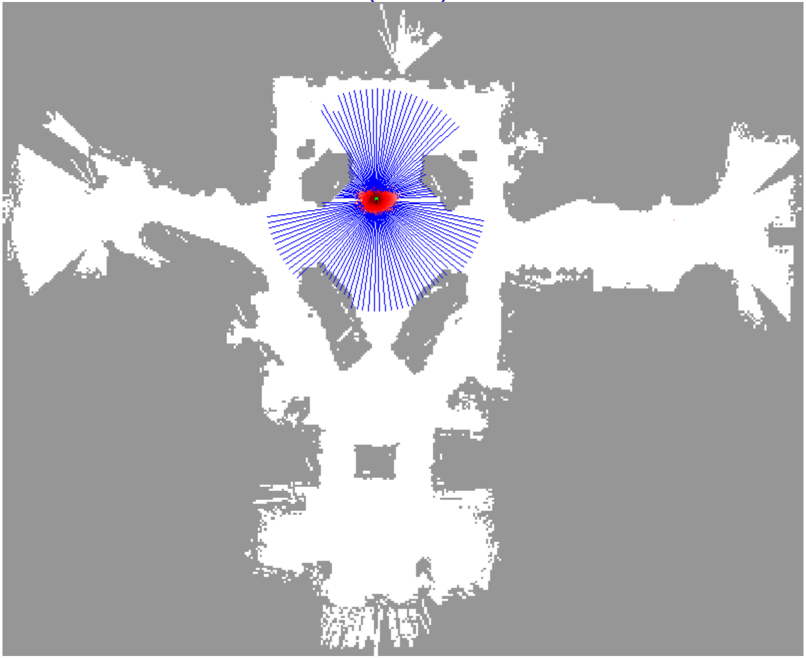
Particle Filter Localization (2-D)



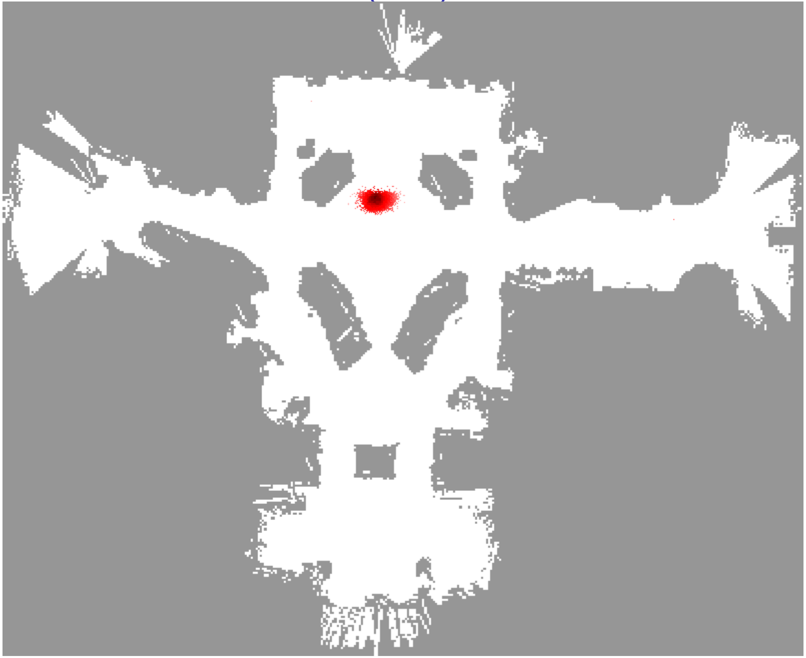
Particle Filter Localization (2-D)



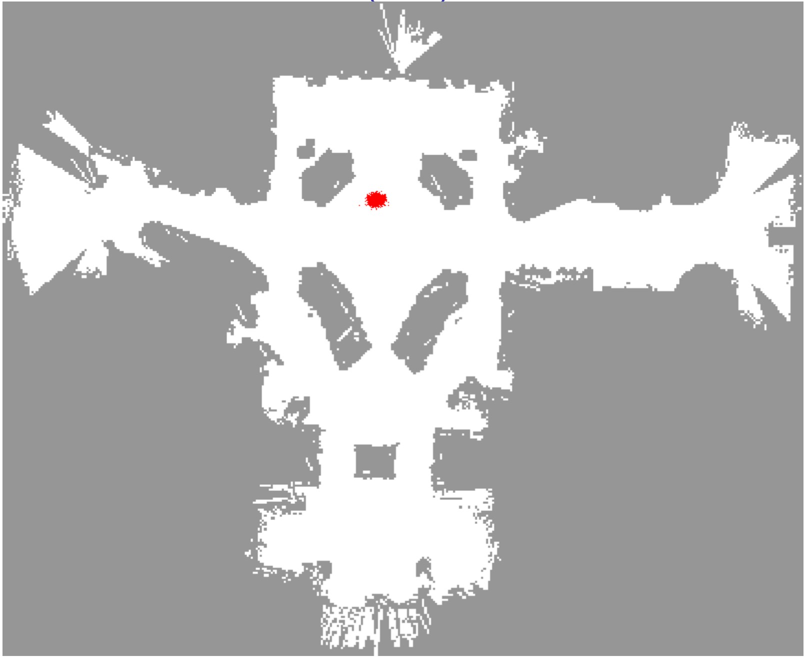
Particle Filter Localization (2-D)



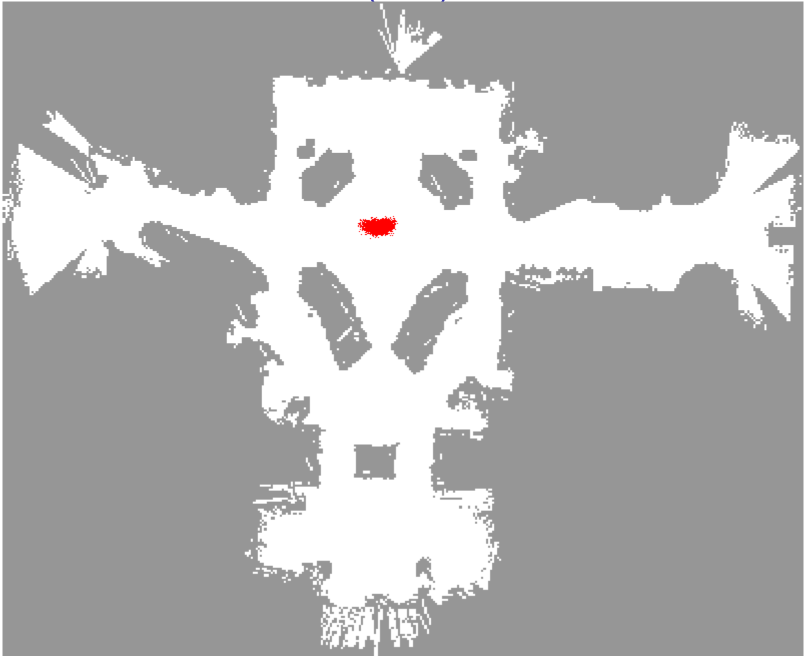
Particle Filter Localization (2-D)



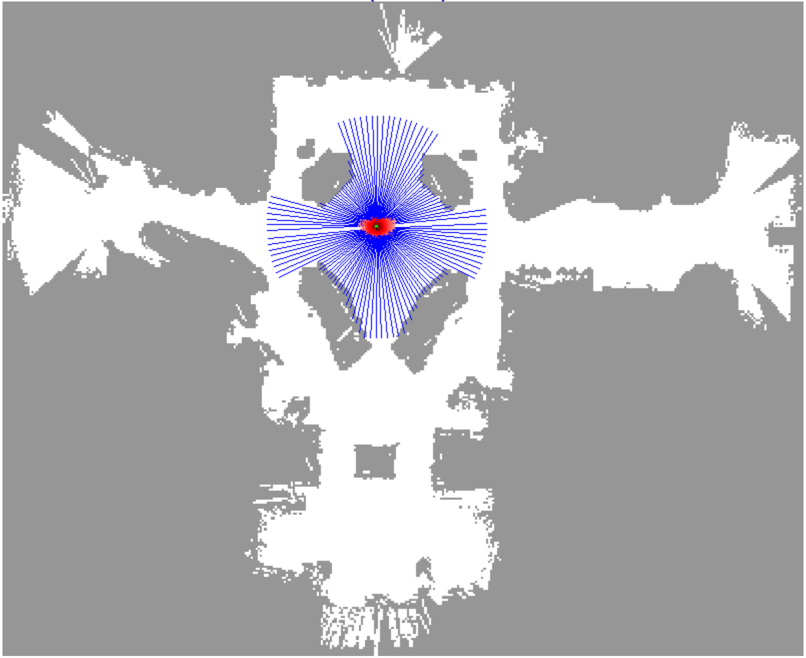
Particle Filter Localization (2-D)



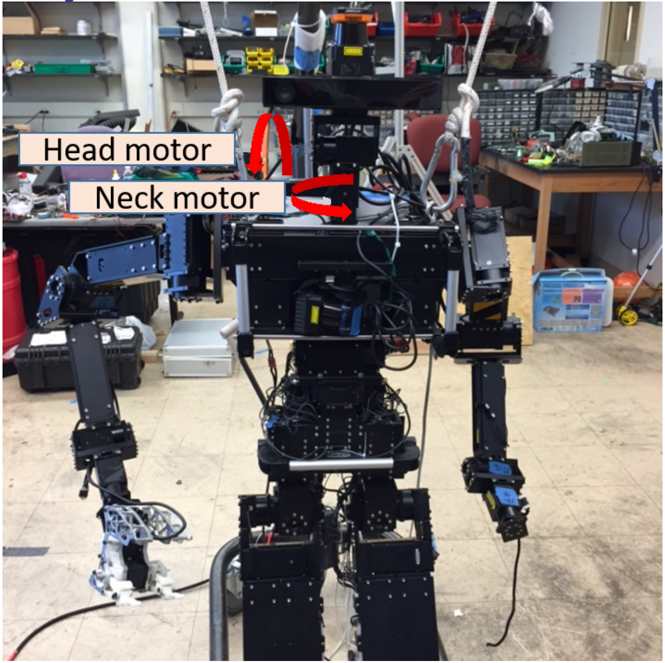
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

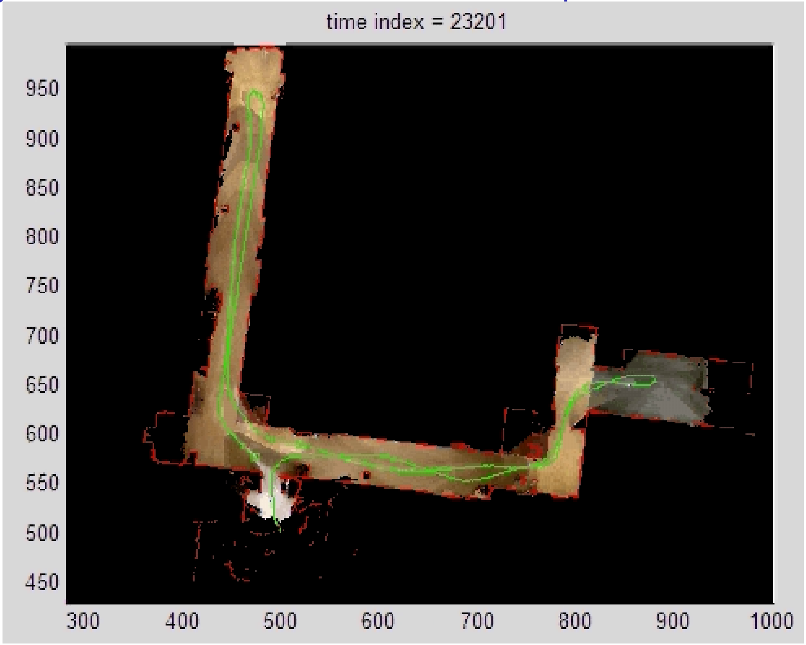


Project 2: Humanoid THOR



- RGBD camera
- 2D Lidar
- IMU
- Odometry
- Transforms

Project 2: Localization and Texture Map



Project 2: Lidar-based Localization & Mapping

- Initial particle set $\boldsymbol{\mu}_{0|0}^{(k)} = (0, 0, 0)^\top$ with weights $\alpha_{0|0}^{(k)} = \frac{1}{N}$
- Use the first laser scan to initialize the map:
  1. use the head angle to remove the ground plane from the scan
  2. convert the scan to Cartesian coordinates
  3. transform the scan from the lider frame to the body frame and then to the world frame
  4. convert the scan to cells (via **bresenham2D** or **cv2.drawContours**) and update the map log-odds
- Use an **odometry motion model** to predict motion for each particle
- Use the laser scan from each particle to compute map correlation (via **getMapCorrelation**) and update the particle weights
- Choose the best particle, project the laser scan, and update the map log-odds (in general, each particle should maintain its own map)
- **Textured map**: use the RGBD images from the best particle's pose to assign colors to the occupancy grid cells