ECE276A: Sensing & Estimation in Robotics
Lecture 10: Particle Filter SLAM

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qjfeng@ucsd.edu

Arash Asgharivaskasi: aasghari@eng.ucsd.edu

Ehsan Zobeidi: ezobeidi@ucsd.edu

Rishabh Jangir: rjangir@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, estimate the robot 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 $\mathbf{z}_{0:T}$
- Possible approaches:
  - **MLE**: maximize the data likelihood conditioned on the parameters:

$$\max_{\mathbf{x}_{0:T},\mathbf{m}} \log \; p(\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1} \mid \mathbf{x}_{0:T}, \mathbf{m})$$

  - **MAP**: maximize the posterior pdf of the parameters given the data:

$$\max_{\mathbf{x}_{0:T},\mathbf{m}} \log \; p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$$

  - **BI**: use Bayesian inference to maintain a posterior pdf for the parameters given the data:

$$p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$$

Simultaneous Localization & Mapping (SLAM)

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

$$p(\mathbf{x}_{0:T}, \mathbf{m}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_0(\mathbf{x}_0, \mathbf{m})}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}} \prod_{t=0}^{T-1} \underbrace{p(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{control policy}}$$

- The control policy term is usually not considered
- **MLE**: $\max_{\mathbf{x}_{0:T}, \mathbf{m}} \sum_{t=0}^{T} \log p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m}) + \sum_{t=1}^{T} \log p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$
- **MAP**: equivalent to MLE with the addition of a prior $\log p_0(\mathbf{x}_0, \mathbf{m})$ to the objective function
- **BI**: uses Bayesian smoothing to obtain $p(\mathbf{x}_{0:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1})$

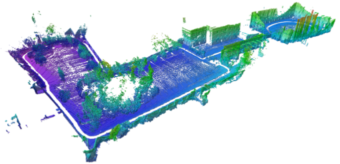
Simultaneous Localization & Mapping (SLAM)

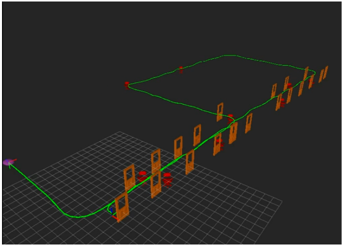
- Earlier SLAM techniques:
  - Bayes filtering to maintain only $p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$
  - Expectation Maximization (EM) treating $\mathbf{x}_t$ as a hidden variable. Given an inital map $\mathbf{m}^{(0)}$, e.g., obtained from the first observation $\mathbf{z}_0$, 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 SLAM approach depends on the particular representation of the robot states $\mathbf{x}_t$, map $\mathbf{m}$, observations $\mathbf{z}_t$, control inputs $\mathbf{u}_t$, observation model p_h, and motion model p_f.

Mapping

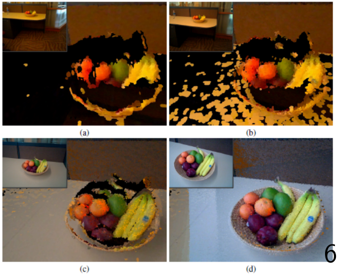
- Given a robot state trajectory $\mathbf{x}_{0:T}$ and a sequence of measurements $\mathbf{z}_{0:T}$, build a map $\mathbf{m}$ of the environment

Sparse Map Representations



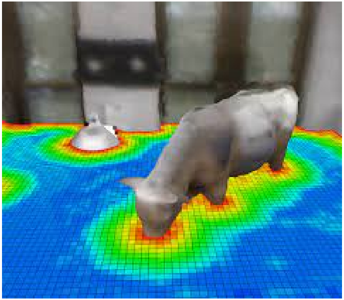


- **Point cloud**: a collection of points, potentially with properties, e.g., color
- **Landmark-based**: objects, each having a semantic class, position, orientation, shape, etc.
- **Surfels**: a collection of oriented discs containing photometric information

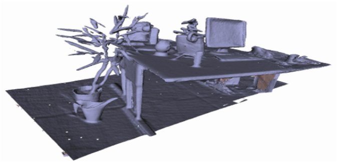


Dense Map Representations

- **Implicit Surface Models**:
  - **Occupancy-based**: assign occupied $(+1)$ or free (-1) labels over the space of the environment
  - **Distance-based**: measure the signed distance (negative inside) to the environment surfaces



- **Explicit Surface Models**:
  - **Polygonal mesh**: a collection of points and connectivity information among them, forming polygons



Popular Sparse 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)

Popular Dense SLAM Algorithms

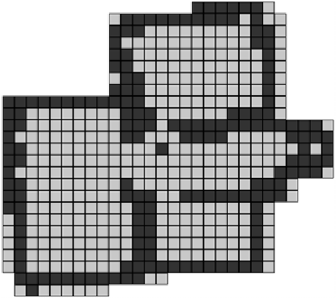
- **Fast SLAM** (Montemerlo et al., AAAI'02)
  - exploits that the occupancy grid cells are independent conditioned on the robot trajectory:

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

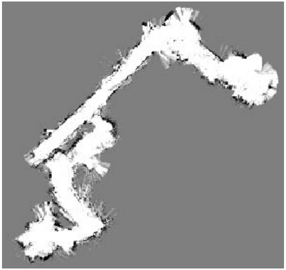
  - uses a particle filter to maintain the robot trajectory pdf and log-odds mapping to maintain a probabilistic map for every particle
- **Kinect Fusion** (Newcombe et al., ISMAR'11)
  - matches consecutive RGBD point clouds using the iterative closest point (ICP) algorithm
  - updates a grid discretization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

Occupancy Grid Map

- One of the simplest and most widely used representations
- The environment is divided into a regular grid with n cells
- **Occupancy grid**: 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



Probabilistic Occupancy Grid Mapping



- The occupancy grid $\mathbf{m}$ is unknown and needs to be estimated given the robot trajectory $\mathbf{x}_{0:t}$ and a sequence of observations $\mathbf{z}_{0:t}$
- Since the map is unknown and the measurements are uncertain, maintain a pmf $p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ over time
- **Independence Assumption**: occupancy grid mapping algorithms usually assume that the cell values are independent conditioned on the robot trajectory:

$$p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \prod_{i=1}^{n} p(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$$

- It is sufficient to track $\gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ for each map cell

Probabilistic Occupancy Grid Mapping

- Model the map cells m_i as independent Bernoulli random variables

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

- How do we update $\gamma_{i,t}$ over time?

- **Bayes Rule**:

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

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

Probabilistic Occupancy Grid Mapping

- **Odds ratio** of the Bernoulli random variable m_i updated via Bayes rule:

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

- Observation model odds ratio: $g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t)$

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

$$g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) = \frac{p_h(\mathbf{z}_t \mid m_i = 1, \mathbf{x}_t)}{p_h(\mathbf{z}_t \mid m_i = -1, \mathbf{x}_t)} = \underbrace{\frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)}}_{\text{inverse observation model odds ratio}} \underbrace{\frac{p(m_i = -1)}{p(m_i = 1)}}_{\text{map prior odds ratio}}$$

Probabilistic Occupancy Grid Mapping

- Observation model odds ratio:

$$g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) = \underbrace{\frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)}}_{\substack{\text{inverse observation model} \\ \text{odds ratio}}} \underbrace{\frac{p(m_i = -1)}{p(m_i = 1)}}_{\substack{\text{map prior} \\ \text{odds ratio}}}$$

- **Inverse observation model**: $p_h(m|\mathbf{z}, \mathbf{x})$

- Assume $\mathbf{z}_t$ indicates whether m_i is occupied or not. Then, the inverse observation model odds ratio specifies how much we trust the observations, i.e., it is the ratio of true positives versus false positives:

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

- The second term $o(m_i) = \frac{p(m_i=1)}{p(m_i=-1)}$ is just a prior occupancy odds ratio and may be chosen as 1 (occupied and free space are equally likely) or < 1 (optimistic about free space)

Probabilistic Occupancy Grid Mapping

- **Odds ratio occupancy grid mapping**:

$$o(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) o(m_i \mid \mathbf{z}_{0:t-1}, \mathbf{x}_{0:t-1})$$

- Observation model odds ratio: $g_h(\mathbf{z}_t \mid m_i, \mathbf{x}_t) = \frac{p(m_i=1|\mathbf{z}_t,\mathbf{x}_t)}{p(m_i=-1|\mathbf{z}_t,\mathbf{x}_t)} \frac{1}{o(m_i)}$

- Take a log to convert the products to sums

- Log-odds of the Bernoulli random variable m_i:

$$\lambda_{i,t} := \lambda(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) := \log o(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$$

- **Log-odds occupancy grid mapping**:

$$\lambda_{i,t} = \underbrace{\log \frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)}}_{\Delta\lambda_{i,t}} - \lambda_{i,0} + \lambda_{i,t-1}$$

Probabilistic Occupancy Grid Mapping

- Estimating the pmf of m_i conditioned on $\mathbf{z}_{0:t}$ and $\mathbf{x}_{0:t}$ is equivalent to accumulating the log-odds ratio $\Delta\lambda_{i,t}$ of the inverse measurement model:

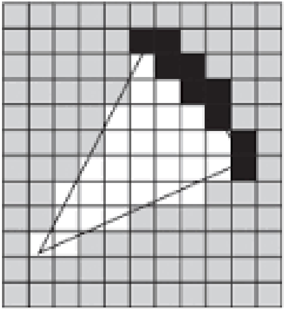
$$\lambda_{i,t} = \lambda_{i,t-1} + (\Delta\lambda_{i,t} - \lambda_{i,0})$$

- If the map prior is uniform, i.e., occupied and free space are equally likely: $\lambda_{i,0} = \log 1 = 0$

- Assuming that $\mathbf{z}_t$ indicates whether m_i is occupied or not, the log-odds ratio $\Delta\lambda_{i,t}$ of the inverse measurement model specifies the measurement "trust", e.g., for an 80% correct sensor:

$$\Delta\lambda_{i,t} = \log \frac{p(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{p(m_i = -1 \mid \mathbf{z}_t, \mathbf{x}_t)} = \begin{cases} +\log 4 & \text{if } \mathbf{z}_t \text{ indicates } m_i \text{ is occupied} \\ -\log 4 & \text{if } \mathbf{z}_t \text{ indicates } m_i \text{ is free} \end{cases}$$

Lidar 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:

$$\lambda_{i,t+1} = \lambda_{i,t} \pm \log 4$$

- 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 $\lambda_{i,t}$ via the logistic sigmoid function:

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

Localization

- Given a map $\mathbf{m}$, a sequence of control inputs $\mathbf{u}_{0:T-1}$, and a sequence of measurements $\mathbf{z}_{0:T}$, infer the robot state trajectory $\mathbf{x}_{0:T}$

Markov Localization in Occupancy Grid Maps

- Use the particle filter to maintain the pdf $p(\mathbf{x}_t | \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}, \mathbf{m})$ of the robot state over time
- Each particle $\boldsymbol{\mu}_{t|t}^{(k)}$ is a hypothesis on the state $\mathbf{x}_t$ with confidence $\alpha_{t|t}^{(k)}$
- The particles specify the pdf of the robot state at time t:

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

- **Prediction step**: use the motion model p_f to obtain the predicted pdf $p_{t+1|t}(\mathbf{x}_{t+1})$
- **Update step**: use the observation model p_h to obtain the updated pdf $p_{t+1|t+1}(\mathbf{x}_{t+1})$

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:

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

  - $f(\mathbf{x}, \mathbf{u})$ is the differential-drive motion model
  - $\mathbf{u}_t = (v_t, \omega_t)$ is the linear and angular velocity input
  - $\boldsymbol{\epsilon}_t \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}\right)$ is a 2-D Gaussian motion noise
- If $\mathbf{u}_t$ is unknown it can be obtained from wheel encoders (linear velocity) and an IMU sensor:
  - The distance traveled during time τ_t for a given encoder count z_t, wheel diameter d, and 360 ticks per revolution is: $\tau_t v_t \approx \frac{\pi d z_t}{360}$
  - ω_t is the yaw rate directly provided by the gyroscope angular velocity measurement

Lidar-based Localization with a Differential-drive Robot

- **Update step**: the particle poses remain unchanged but the weights are scaled by the observation model:

$$\boldsymbol{\mu}^{(k)}_{t+1|t+1} = \boldsymbol{\mu}^{(k)}_{t+1|t} \qquad \alpha^{(k)}_{t+1|t+1} \propto p_h(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}^{(k)}_{t+1|t}, \mathbf{m})\alpha^{(k)}_{t+1|t}$$

- Need to define a lidar observation model: $p_h(\mathbf{z} \mid \mathbf{x}, \mathbf{m})$
- **Laser Correlation Model**: a model for a laser scan $\mathbf{z}$ obtained from sensor pose $\mathbf{x}$ in occupancy grid $\mathbf{m}$ based on correlation between $\mathbf{z}$ and $\mathbf{m}$
- Transform the scan $\mathbf{z}_{t+1}$ to the world frame using $\boldsymbol{\mu}^{(k)}_{t+1|t}$ and find all cells $\mathbf{y}^{(k)}_{t+1}$ in the grid corresponding to the scan
- Update the particle weights using the laser correlation model:

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

Laser Correlation Model

- The laser correlation model sets the likelihood of a laser scan $\mathbf{z}$ proportional to the correlation between the scan's world-frame projection $\mathbf{y} = r(\mathbf{z}, \mathbf{x})$ via the robot pose $\mathbf{x}$ and the occupancy grid $\mathbf{m}$

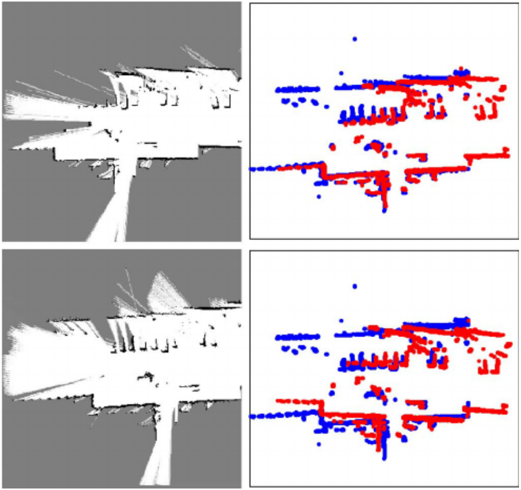
$$p_h(\mathbf{z}|\mathbf{x}, \mathbf{m}) \propto \text{corr}\left(r(\mathbf{z}, \mathbf{x}), \mathbf{m}\right)$$

- Computing **scan-grid correlation**:
  - Transform the scan $\mathbf{z}$ from the laser frame to the world frame using the robot pose $\mathbf{x}$ (transformation from the body frame to the world frame)
  - Find all grid coordinates $\mathbf{y}$ that correspond to the scan, i.e., $\mathbf{y}$ is a vector of grid cell indices i which are visited by the laser scan rays (e.g., using Bresenham's line rasterization algorithm)
  - Let $\mathbf{y} = r(\mathbf{z}, \mathbf{x})$ be the transformation from a lidar scan $\mathbf{z}$ to grid cell indices $\mathbf{y}$. Definite a similarity function $\text{corr}(r(\mathbf{z}, \mathbf{x}), \mathbf{m})$ between the transformed and discretized scan $\mathbf{y}$ and the occupancy grid $\mathbf{m}$:

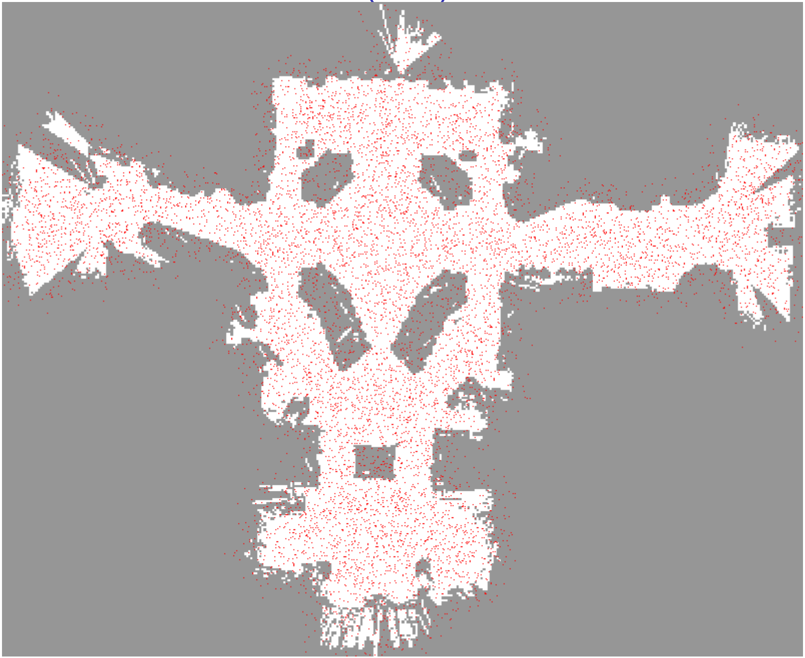
$$\text{corr}(\mathbf{y}, \mathbf{m}) = \sum_i \mathbb{1}\{y_i = m_i\}$$

Laser Correlation Model

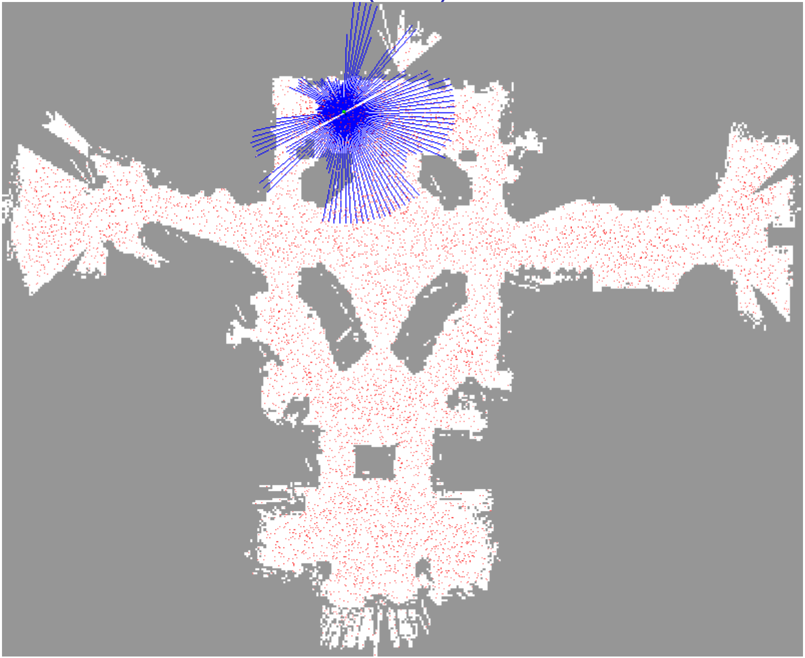
- Transform the scan $\mathbf{z}_{t+1}$ to the world frame using $\boldsymbol{\mu}_{t+1|t}^{(k)}$ and find all cells $\mathbf{y}_{t+1}^{(k)}$ in $\mathbf{m}$ corresponding to the scan
- The correlation $\text{corr}\left(\mathbf{y}_{t+1}^{(k)}, \mathbf{m}\right)$ is large if $\mathbf{y}_{t+1}^{(k)}$ and $\mathbf{m}$ agree



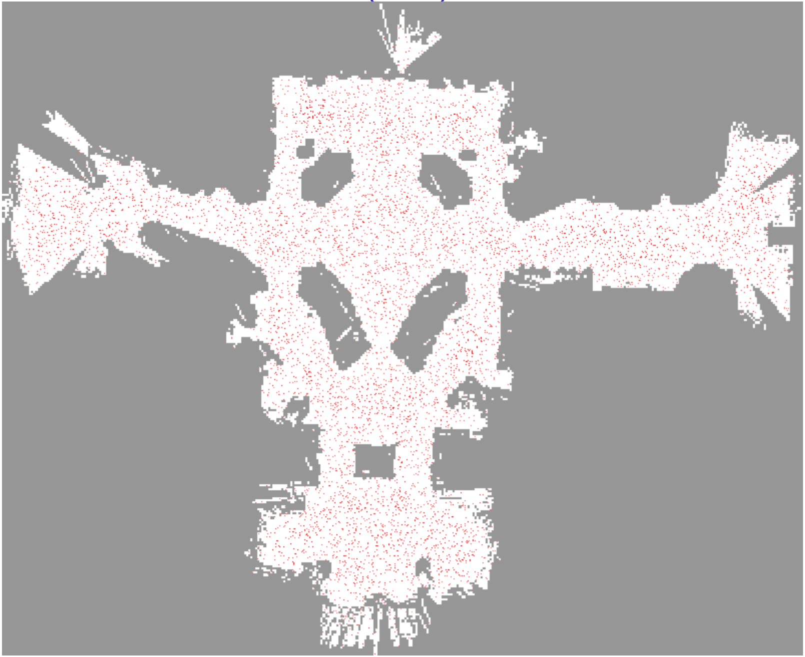
Particle Filter Localization (2-D)



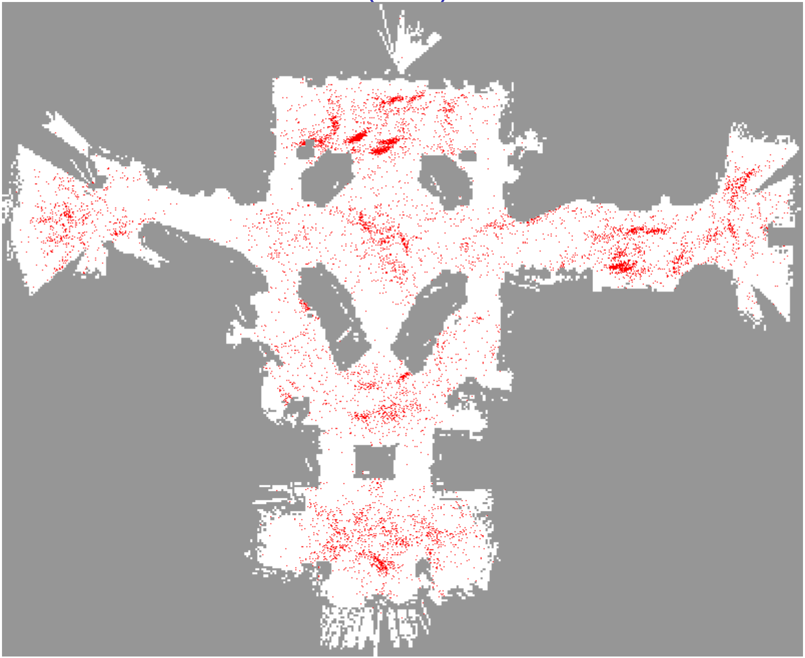
Particle Filter Localization (2-D)



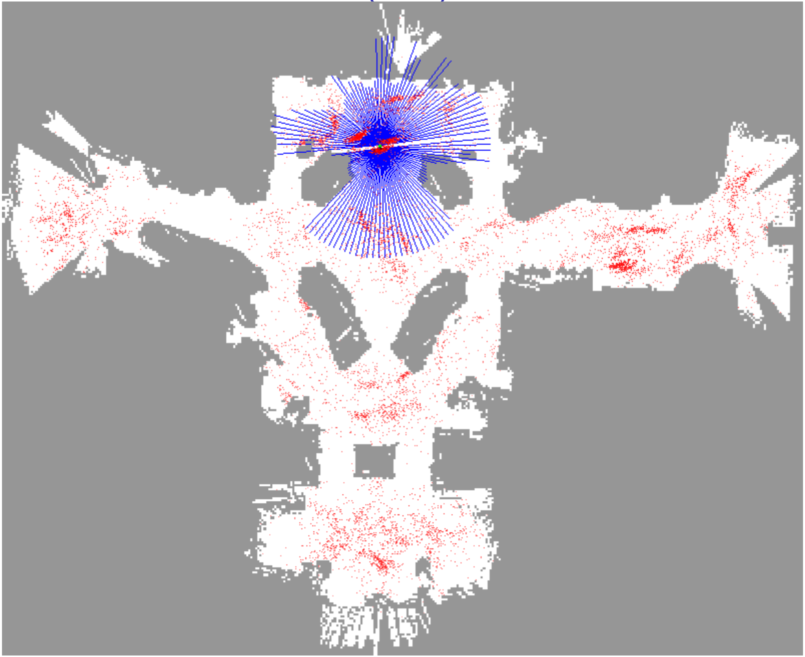
Particle Filter Localization (2-D)



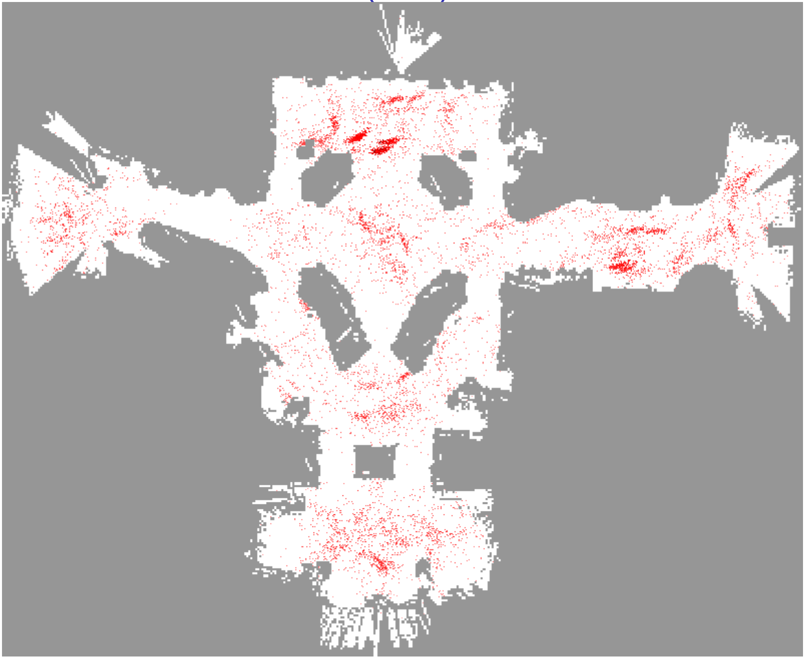
Particle Filter Localization (2-D)



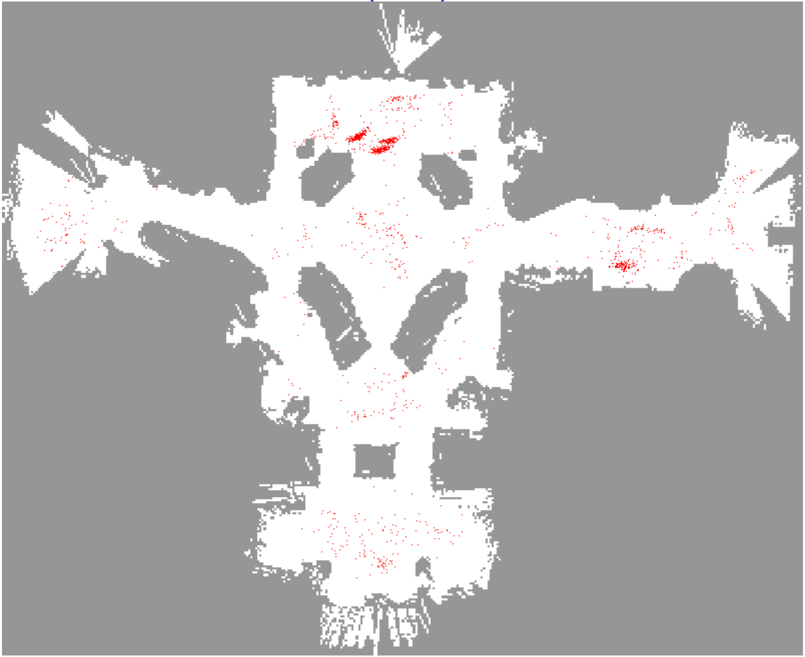
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

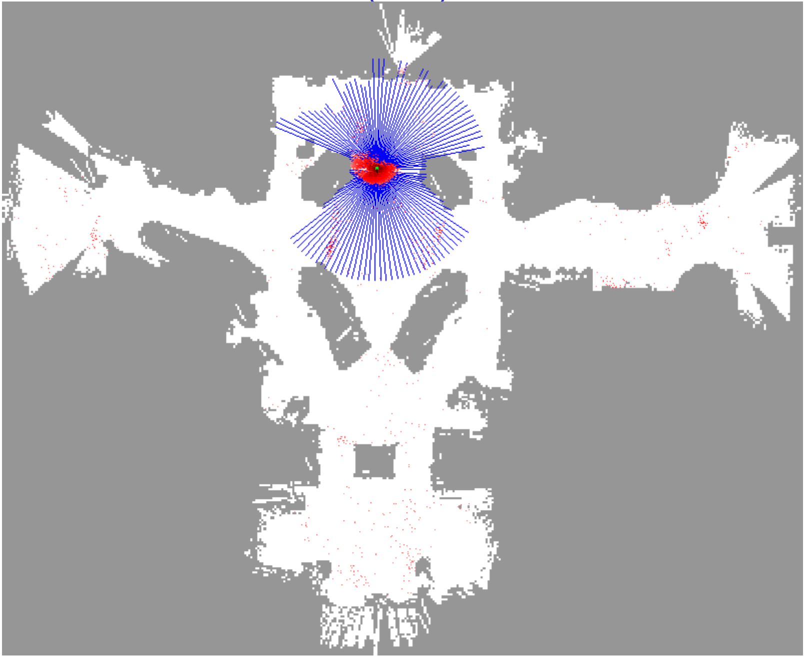


Particle Filter Localization (2-D)



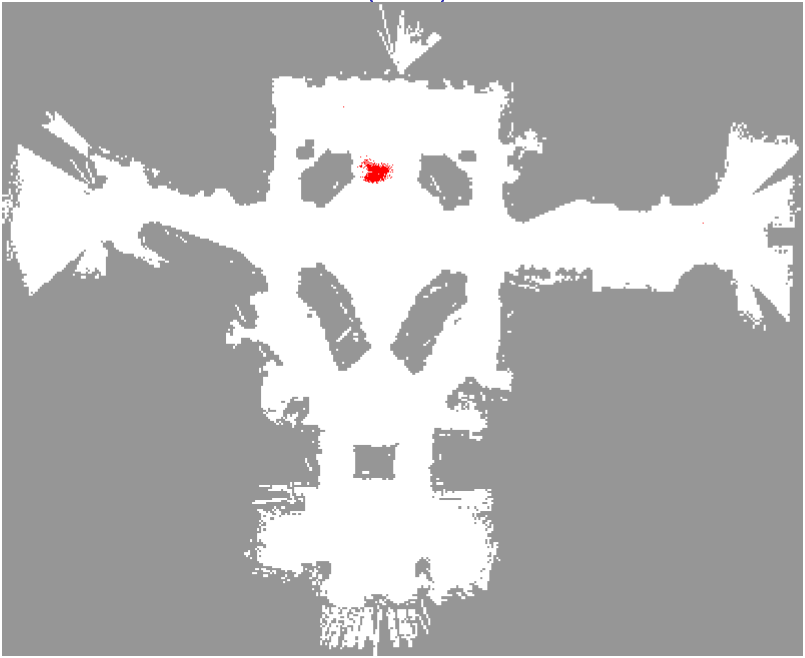
Particle Filter Localization (2-D)

Particle Filter Localization (2-D)

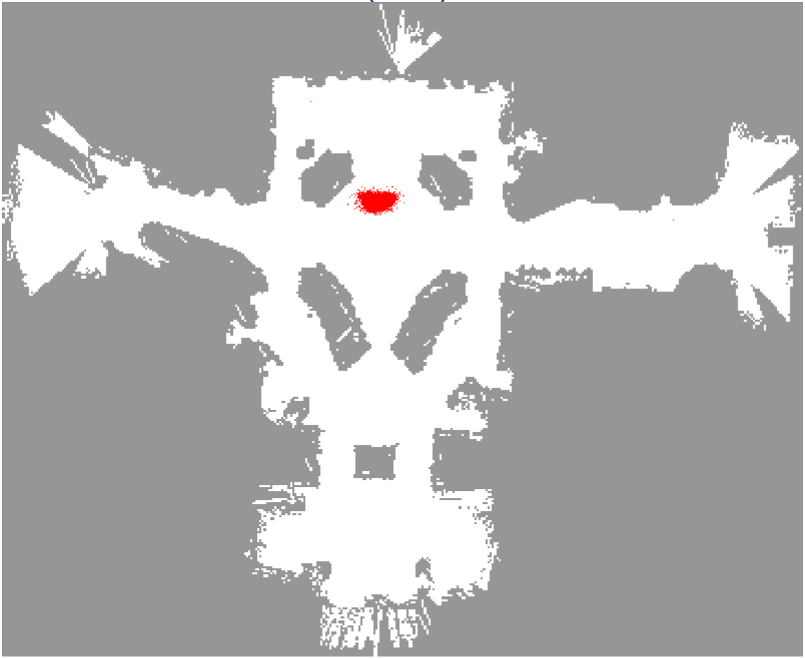


Particle Filter Localization (2-D)

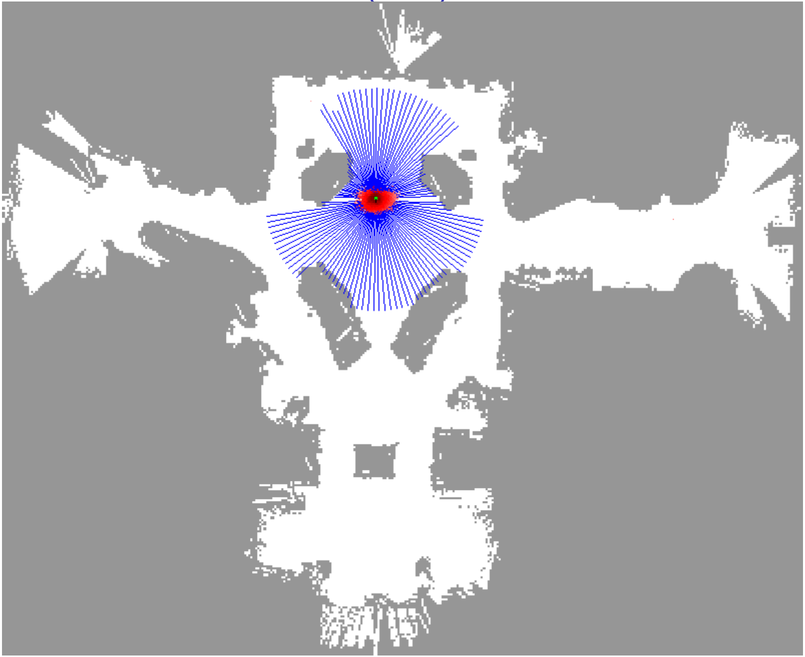
Particle Filter Localization (2-D)



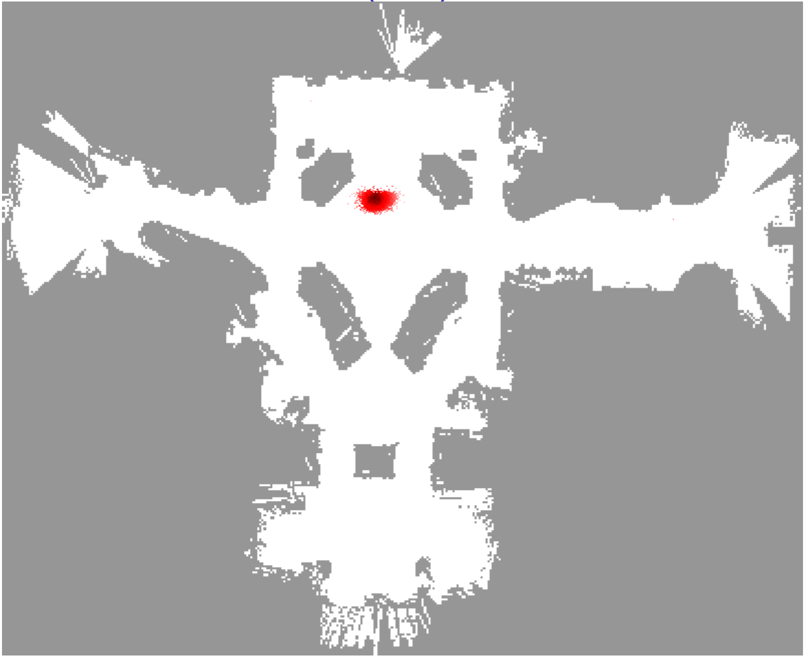
Particle Filter Localization (2-D)



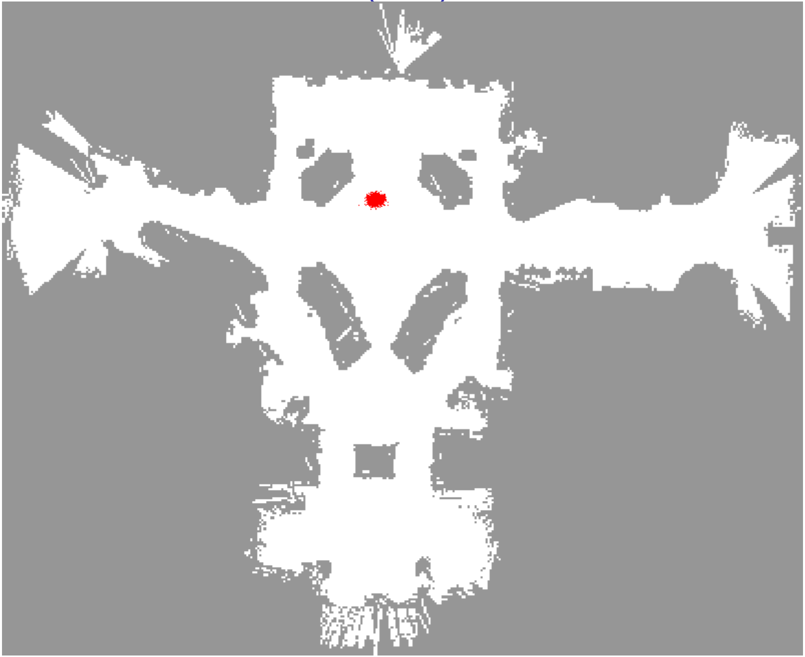
Particle Filter Localization (2-D)



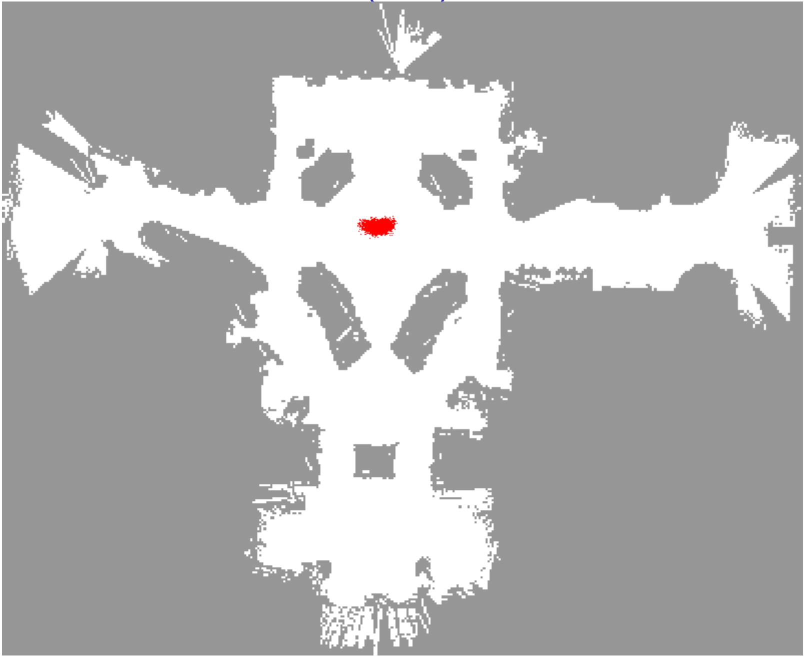
Particle Filter Localization (2-D)



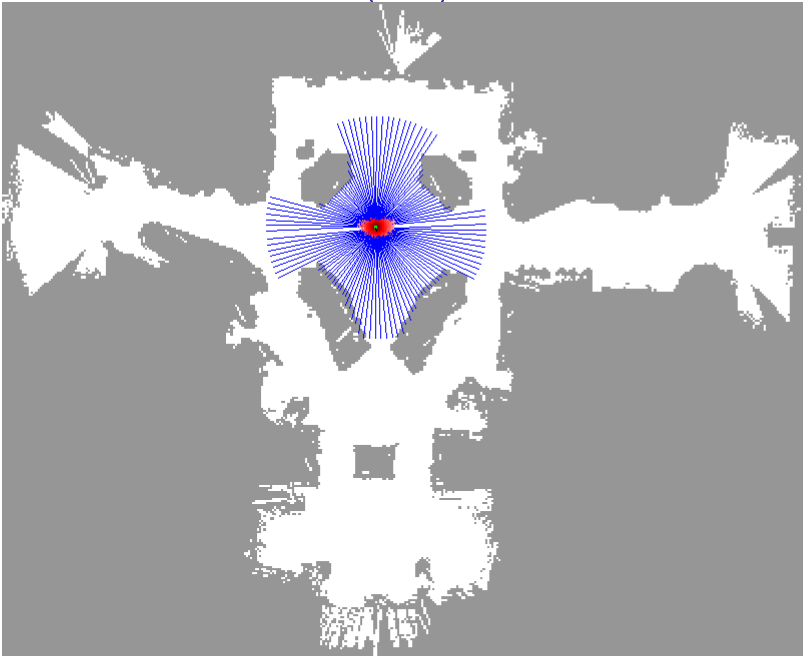
Particle Filter Localization (2-D)



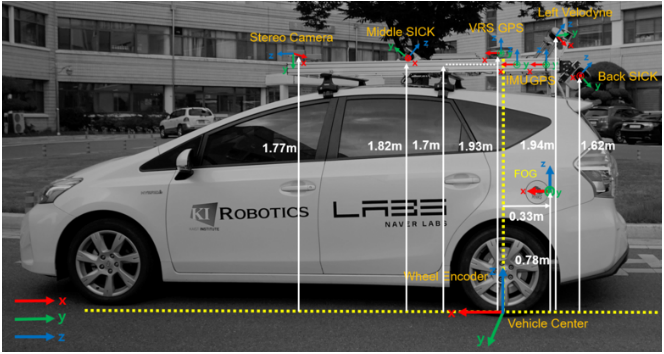
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

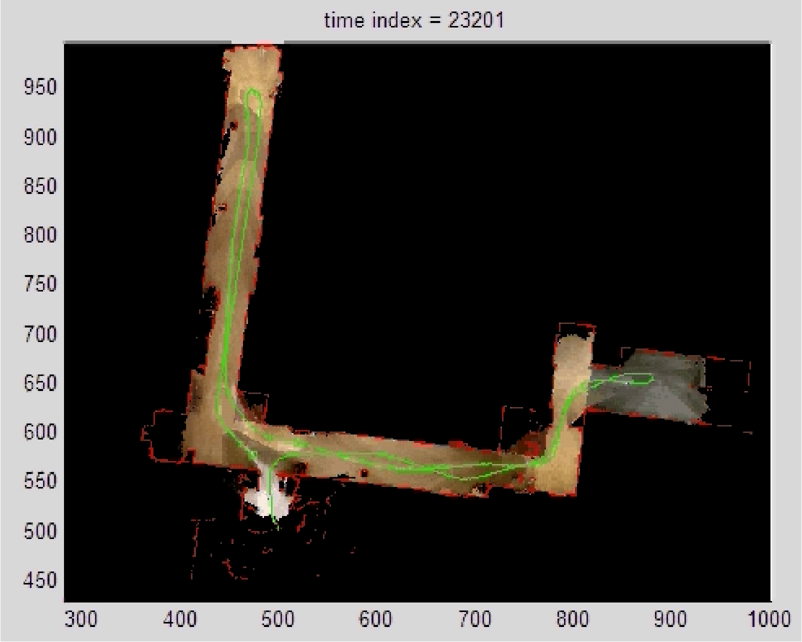


Project 2: Autonomous Car



- Stereo RGB camera
- 2D Lidar
- 2x 3D Lidar
- Encoders, IMU
- Odometry
- Transforms

Project 2: Localization and Texture Map



Project 2: Lidar-based Particle-filter SLAM

- 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. convert the scan to cartesian coordinates
  2. transform the scan from the lidar frame to the body frame and then to the world frame
  3. convert the scan to cells (via **bresenham2D** or **cv2.drawContours**) and update the map log-odds
- **Prediction step**: Use the **differential-drive model** with velocity from the encoders and angular velocity from the gyroscope to predict the motion of each particle and add noise
- **Update step**: combines robot state and map update
  - Use the laser scan from each particle to compute map correlation (via **getMapCorrelation**) and update the particle weights
  - Choose the particle with largest weight $\alpha_{t|t}^{(k)}$, project the laser scan $\mathbf{z}_t$ to the world frame and update the map log-odds
- **Textured map**: use the RGBD images from the largest-weight particle's pose to assign colors to the occupancy grid cells