ECE276A: Sensing & Estimation in Robotics
Lecture 8: Particle Filter

Nikolay Atanasov

natanasov@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Outline

Particle Filter

Particle Filter SLAM

Monte Carlo Sampling

Particle Filter

- **Particle filter**: Bayes filter in which $p_{t+1|t}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t})$ and $p_{t+1|t+1}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t+1}, \mathbf{u}_{0:t})$ are discrete distributions with N possible values called particles

- A probability mass function $\alpha[1], \ldots, \alpha[N]$ over N values $\boldsymbol{\mu}[1], \ldots, \boldsymbol{\mu}[N]$ can be viewed as a continuous-space probability density function:

$$p(\mathbf{x}) = \sum_{k=1}^{N} \alpha[k]\delta(\mathbf{x} - \boldsymbol{\mu}[k])$$

where δ is the Dirac delta function:

$$\delta(x) := \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases} \qquad \int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0) \qquad \int_{-\infty}^{\infty} \delta(x)dx = 1$$

Particle Filter

- **Particle**: a hypothesis that the value of $\mathbf{x}$ is $\boldsymbol{\mu}[k]$ with probability $\alpha[k]$
- The particle filter uses particles with locations $\boldsymbol{\mu}[k]$ and weights $\alpha[k]$ for $k = 1, \ldots, N$ to represent the pdfs $p_{t|t}$ and $p_{t+1|t}$:

$$p_{t|t}(\mathbf{x}_t) = \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_t - \boldsymbol{\mu}_{t|t}[k]\right)$$
$$p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{k=1}^{N} \alpha_{t+1|t}[k]\delta\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right)$$

- To derive the particle filter, substitute these pdfs in the Bayes filter prediction and update steps
- The prediction and update steps should maintain the form of the pdfs as a mixture of delta functions

Particle Filter Prediction Step

- Plug $p_{t|t}(\mathbf{x}_t) = \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_t - \boldsymbol{\mu}_{t|t}[k]\right)$ in the Bayes filter prediction step:

$$
\begin{aligned}
p_{t+1|t}(\mathbf{x}_{t+1}) &= \int p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_t - \boldsymbol{\mu}_{t|t}[k]\right) d\mathbf{x}_t \\
&= \sum_{k=1}^{N} \alpha_{t|t}[k] p_f(\mathbf{x}_{t+1} \mid \boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t)
\end{aligned}
$$

- Since the predicted pdf is not a mixture of delta functions we need to approximate it

- Apply the motion model to each particle $\boldsymbol{\mu}_{t|t}[k]$ to obtain $\boldsymbol{\mu}_{t+1|t}[k] \sim p_f(\cdot \mid \boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t)$ and approximate:

$$
p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{k=1}^{N} \alpha_{t|t}[k] p_f(\mathbf{x}_{t+1} \mid \boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t) \approx \sum_{k=1}^{N} \alpha_{t|t}[k]\delta(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k])
$$

- The prediction step changes only the particle positions but not their weights

Particle Filter Update Step

- Plug $p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right)$ in the Bayes filter update step:

$$p_{t+1|t+1}(\mathbf{x}_{t+1}) = \frac{p_h\left(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}\right) \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right)}{\int p_h\left(\mathbf{z}_{t+1} \mid \mathbf{s}\right) \sum_{j=1}^{N} \alpha_{t|t}[j]\delta\left(\mathbf{s} - \boldsymbol{\mu}_{t+1|t}[j]\right) d\mathbf{s}}$$
$$= \sum_{k=1}^{N} \underbrace{\left[\frac{\alpha_{t+1|t}[k] p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[k]\right)}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[j]\right)}\right]}_{\alpha_{t+1|t+1}[k]} \delta(\mathbf{x} - \underbrace{\boldsymbol{\mu}_{t+1|t}[k]}_{\boldsymbol{\mu}_{t+1|t+1}[k]})$$

- The updated pdf turns out to be a mixture of delta functions so no approximation is necessary

- The update step changes only the particle weights but not their positions

Particle Resampling

- **Particle depletion**: most updated particle weights become close to zero because a finite number of particles is not enough to represent the state pdf, e.g., the observation likelihoods $p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[k]\right)$ may be small at all $k = 1, \ldots, N$

- Resampling tries to avoid particle depletion by adding new particles at locations with high weights and reducing the particles at locations with low weights. It focuses the representation power of the particles to likely regions.

- Given particle set $\left\{\boldsymbol{\mu}_{t|t}[k], \alpha_{t|t}[k]\right\}$, resampling is applied if the **effective number of particles**: $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t|t}[k]\right)^2}$ is less than a threshold

- **Resampling**
  - Draw $j \in \{1, \ldots, N\}$ independently with replacement with probability $\alpha_{t|t}[j]$
  - Add $\boldsymbol{\mu}_{t|t}[j]$ with weight $\frac{1}{N}$ to the new particle set
  - Repeat N times

Particle Filter Summary

- **Prior**: $\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim p_{t|t}(\mathbf{x}_t) := \sum_{k=1}^{N} \alpha_{t|t}[k]\delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}[k]\right)$
- **Prediction**: let $\boldsymbol{\mu}_{t+1|t}[k] \sim p_f(\cdot \mid \boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t)$ and $\alpha_{t+1|t}[k] = \alpha_{t|t}[k]$ so that:

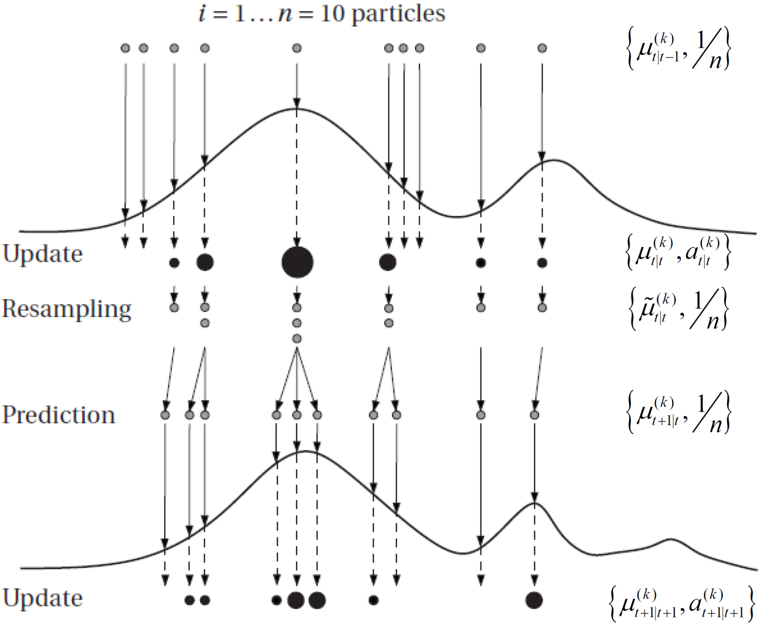
$$p_{t+1|t}(\mathbf{x}_{t+1}) \approx \sum_{k=1}^{N} \alpha_{t+1|t}[k]\delta\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right)$$

- **Update**: rescale the particle weights based on the observation likelihood:

$$p_{t+1|t+1}(\mathbf{x}_{t+1}) = \sum_{k=1}^{N} \left[\frac{\alpha_{t+1|t}[k]p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[k]\right)}{\sum_{j=1}^{N} \alpha_{t+1|t}[j]p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}[j]\right)}\right] \delta\left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{t+1|t}[k]\right)$$

- **Resampling**: If $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t+1|t+1}[k]\right)^2} \leq N/10$, resample the particle set $\left\{\boldsymbol{\mu}_{t+1|t+1}[k], \alpha_{t+1|t+1}[k]\right\}$

Particle Filter Summary



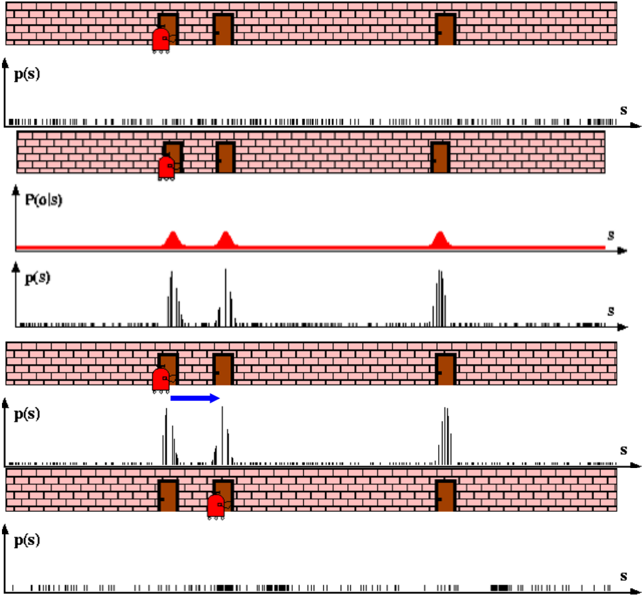
Particle Filter Localization (1-D)

Prior:

Update:

Predict:

Resample:



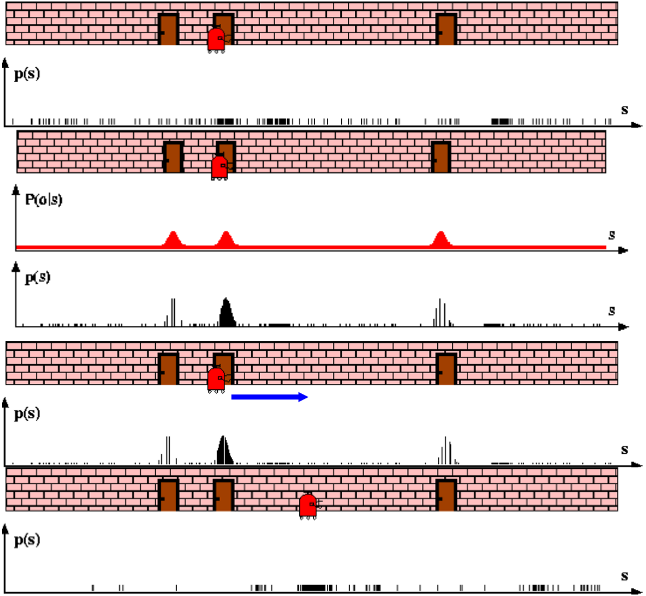
Particle Filter Localization (1-D)

Prior:

Update:

Predict:

Resample:



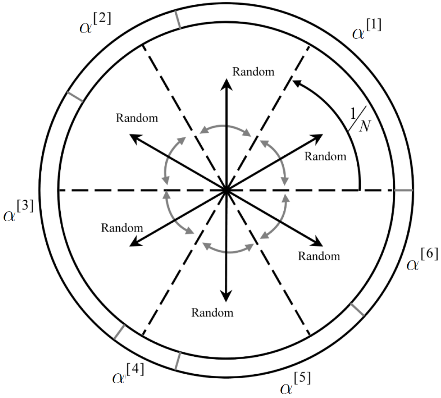
Stratified Resampling

- Sampling the particle set $\{\boldsymbol{\mu}[k], \alpha[k]\}$ independently results in high variance, i.e., sometimes samples with large weights might not be selected, while samples with very small weights may be selected multiple times
- **Stratified resampling**: guarantees that samples with large weights appear at least once and those with small weights – at most once
  - Add the particle weights along the circumference of a circle
  - Divide the circle into N equal pieces and sample a uniform distribution in each piece
  - Select the particles corresponding to the uniform distribution samples
- Stratified resampling is optimal in terms of variance (Thrun et al. 2005)

Stratified Resampling

Stratified Resampling

Input: particle set $\{\boldsymbol{\mu}[k], \alpha[k]\}_{k=1}^{N}$
Output: resampled particle set
$j \leftarrow 1,\ c \leftarrow \alpha[1]$
for $k = 1, \ldots, N$ **do**
  $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$
  $\beta = u + \frac{k-1}{N}$
  while $\beta > c$ **do**
    $j = j + 1,\ c = c + \alpha[j]$
  add $\left(\boldsymbol{\mu}[j], \frac{1}{N}\right)$ to the new set

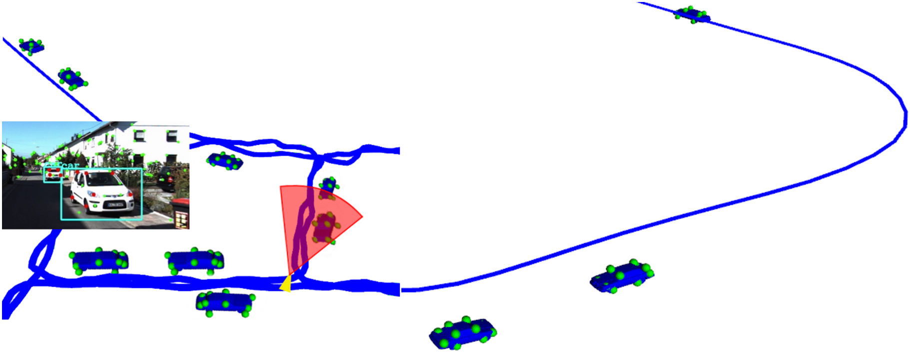


- **Systematic resampling**: the same as stratified resampling except that the **same** uniform is used for each piece, i.e., $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$ is sampled only once before the for loop above.

Outline

SLAM Overview

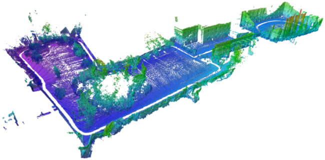
- SLAM problem: given sensor measurements $\mathbf{z}_{0:T}$ (e.g., LiDAR scans) and control inputs $\mathbf{u}_{0:T-1}$ (e.g., velocity), estimate the robot state trajectory $\mathbf{x}_{0:T}$ (e.g., pose) and build a map $\mathbf{m}$ of the environment

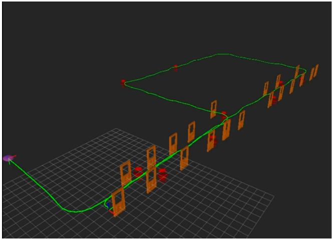


Mapping

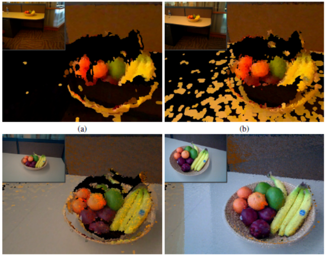
- Given a robot state trajectory $\mathbf{x}_{0:T}$ and a sequence of measurements $\mathbf{z}_{0:T}$, build a map $\mathbf{m}$ of the environment

Sparse Map Representations





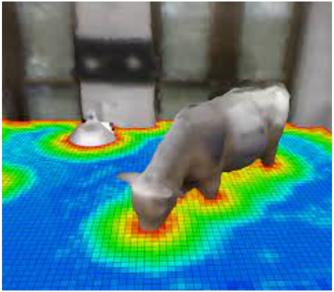
- **Point cloud**: a collection of points, potentially with properties, e.g., color
- **Landmarks**: a collection of objects, each having a category, position, orientation, shape, etc.
- **Surfels**: a collection of oriented discs containing photometric information

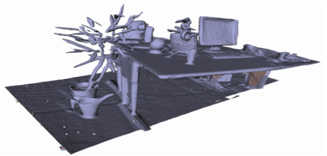


(a) (b) (c) (d)

Dense Map Representations

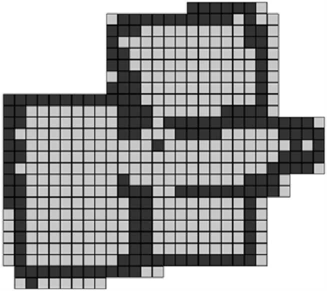
- **Implicit Surface Models**:
  - **Occupancy-based**: assign occupied $(+1)$ or free (-1) labels over the space of the environment
  - **Distance-based**: measure the signed distance (negative inside) to the environment surfaces
- **Explicit Surface Models**:
  - **Polygonal mesh**: a collection of points and connectivity information among them, forming polygons



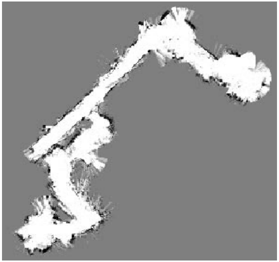


Occupancy Grid Map

- 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



- **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, we maintain a probability mass function $p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ over time
- **Independence Assumption**: most occupancy grid mapping algorithms 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 the probability of being occupied, $\gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$, for each map cell $i = 1, \ldots, n$

Probabilistic Occupancy Grid Mapping

- Model the map cells m_i as independent Bernoulli random variables

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

- 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:

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

Probabilistic Occupancy Grid Mapping

- Observation model odds ratio:

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

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

- **Log-odds occupancy grid mapping**: estimating the probability mass function 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:

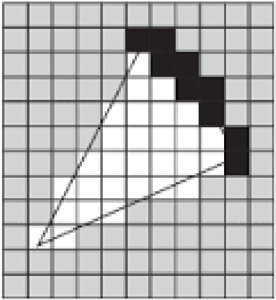
$$\lambda_{i,t} = \lambda_{i,t-1} + (\Delta\lambda_{i,t} - \lambda_{i,0})$$

- 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 grid of map log-odds $\lambda_{i,t}$ for $i = 1, \ldots, n$
- Given a new LiDAR 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 a 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 $\mathbf{x}_t$ 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 input $\mathbf{u}_t$ and motion model p_f to obtain the predicted pdf $p_{t+1|t}(\mathbf{x}_{t+1})$
- **Update step**: use the observation $\mathbf{z}_{t+1}$ and observation model p_h to obtain the updated pdf $p_{t+1|t+1}(\mathbf{x}_{t+1})$

Prediction Step with Differential-drive Robot Model

- Each particle $\boldsymbol{\mu}_{t|t}[k] \in \mathbb{R}^3$ represents a possible 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 2-D Gaussian motion noise

- If $\mathbf{u}_t$ is unknown it can be obtained from wheel encoders (linear velocity v_t) and an IMU sensor (angular velocity ω_t):
  - The distance traveled during time τ_t for a given encoder count z_t, wheel diameter d, and 360 ticks per revolution is: $\tau_t v_t \approx \frac{\pi d z_t}{360}$
  - The angular velocity ω_t is provided by the gyroscope yaw rate measurement directly

Update Step with LiDAR Correlation Model

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

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

- Need to define a LiDAR observation model: $p_h(\mathbf{z} \mid \mathbf{x}, \mathbf{m})$

- **LiDAR correlation model**: likelihood model $p_h(\mathbf{z}|\mathbf{x}, \mathbf{m})$ for LiDAR scan $\mathbf{z}$ obtained from sensor pose $\mathbf{x}$ in occupancy grid $\mathbf{m}$. Set the LiDAR scan likelihood 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 $\mathbf{m}$:

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

- Transform the scan $\mathbf{z}_{t+1}$ to the world frame using $\boldsymbol{\mu}_{t+1|t}[k]$, find all cells $\mathbf{y}_{t+1}[k]$ in the grid corresponding to the scan, and update the particle weights using the scan-map correlation:

$$\alpha_{t+1|t+1}[k] \propto \text{corr}\left(\mathbf{y}_{t+1}[k], \mathbf{m}\right)\alpha_{t+1|t}[k]$$

Update Step with LiDAR Correlation Model

- Computing correlation between LiDAR scan $\mathbf{z}$ obtained from pose $\mathbf{x}$ and occupancy grid map $\mathbf{m}$:
  - Transform the scan $\mathbf{z}$ from the LiDAR 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 LiDAR scan rays, e.g., obtained 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 the correlation $\text{corr}(r(\mathbf{z}, \mathbf{x}), \mathbf{m})$ between the transformed and discretized scan $\mathbf{y}$ and the occupancy grid $\mathbf{m}$ as:

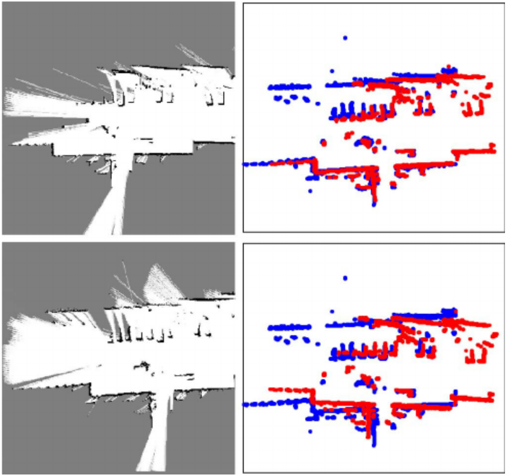
$$\text{corr}(\mathbf{y}, \mathbf{m}) = \sum_i \mathbb{1}\{y_i = m_i\}$$

where:

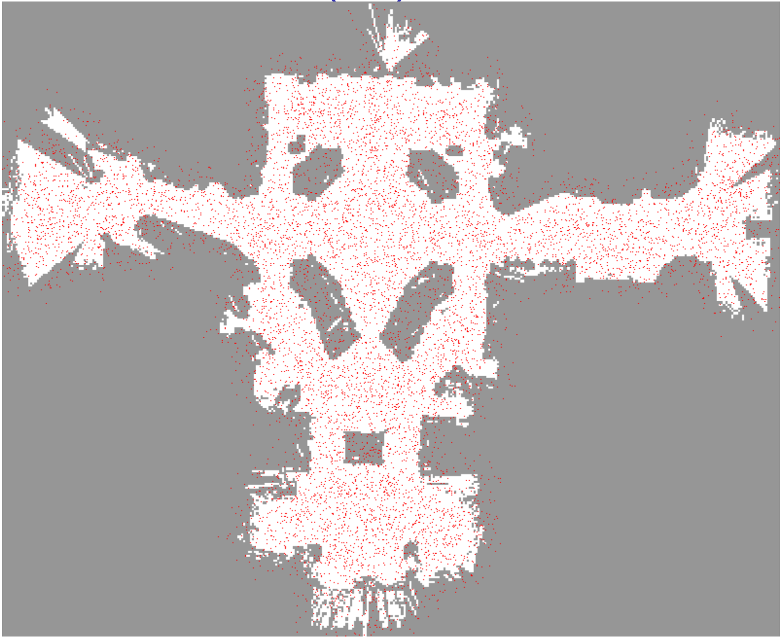
$$\mathbb{1}\{y_i = m_i\} = \begin{cases} 1, & \text{if } y_i = m_i, \\ 0, & \text{else.} \end{cases}$$

Update Step with LiDAR Correlation Model

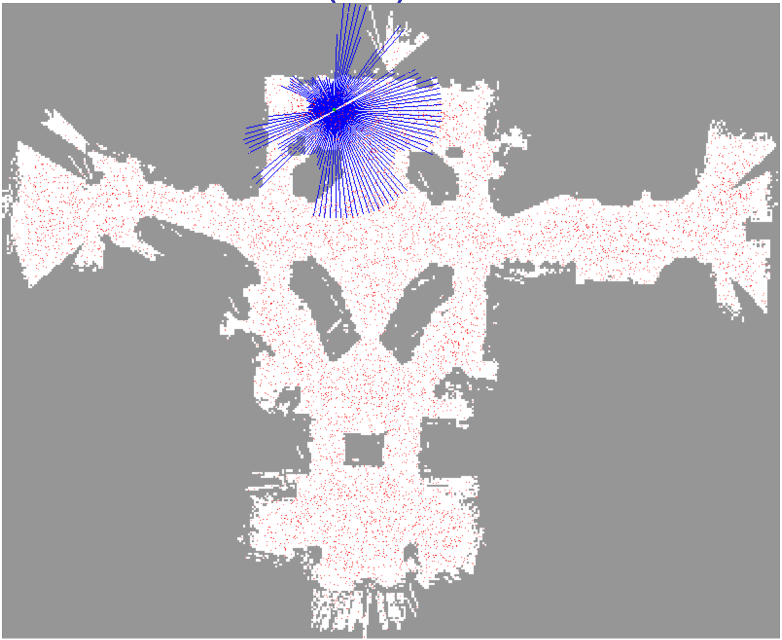
- Transform the scan $\mathbf{z}_{t+1}$ to the world frame using $\boldsymbol{\mu}_{t+1|t}[k]$ and find all cells $\mathbf{y}_{t+1}[k]$ in $\mathbf{m}$ corresponding to the scan
- The correlation $\text{corr}(\mathbf{y}_{t+1}[k], \mathbf{m})$ is large if $\mathbf{y}_{t+1}[k]$ and $\mathbf{m}$ agree



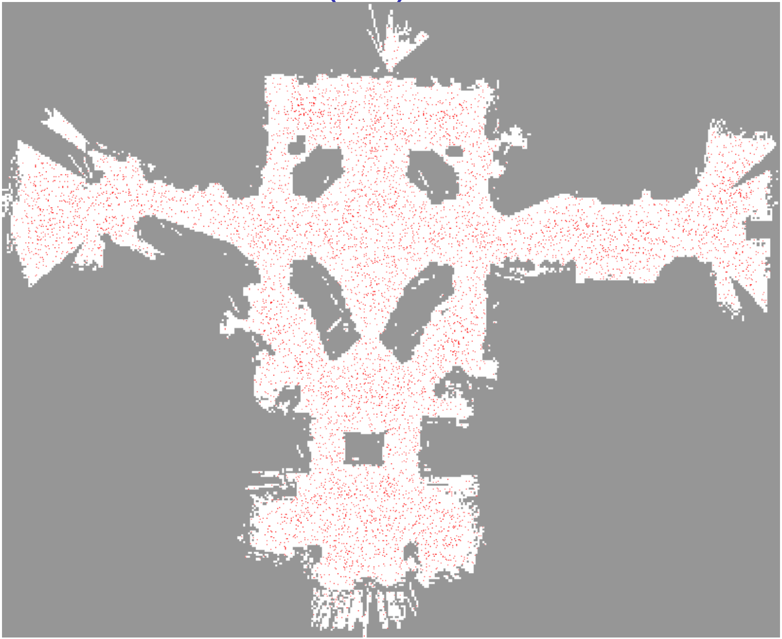
Particle Filter Localization (2-D)



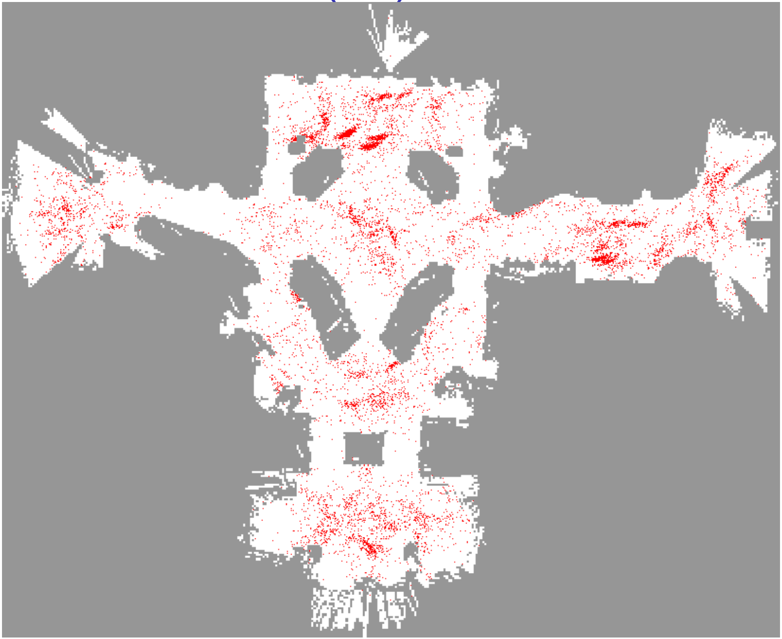
Particle Filter Localization (2-D)



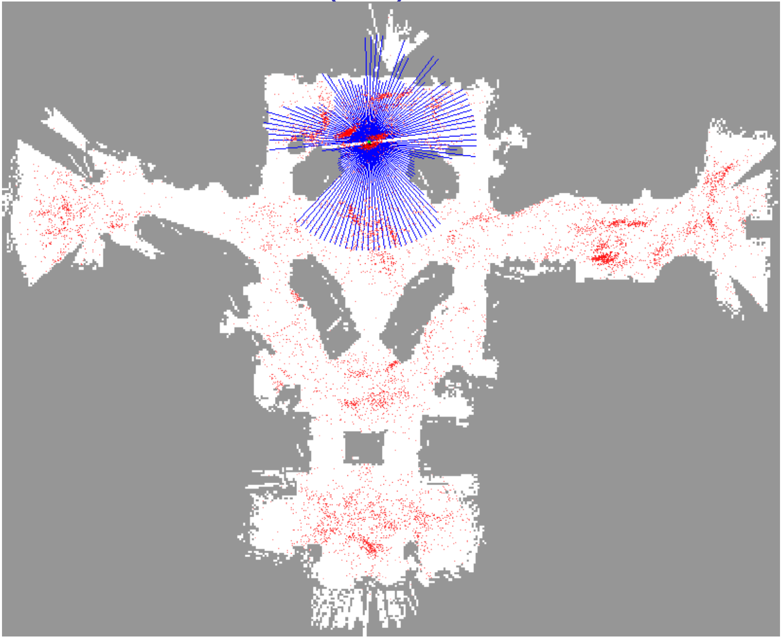
Particle Filter Localization (2-D)



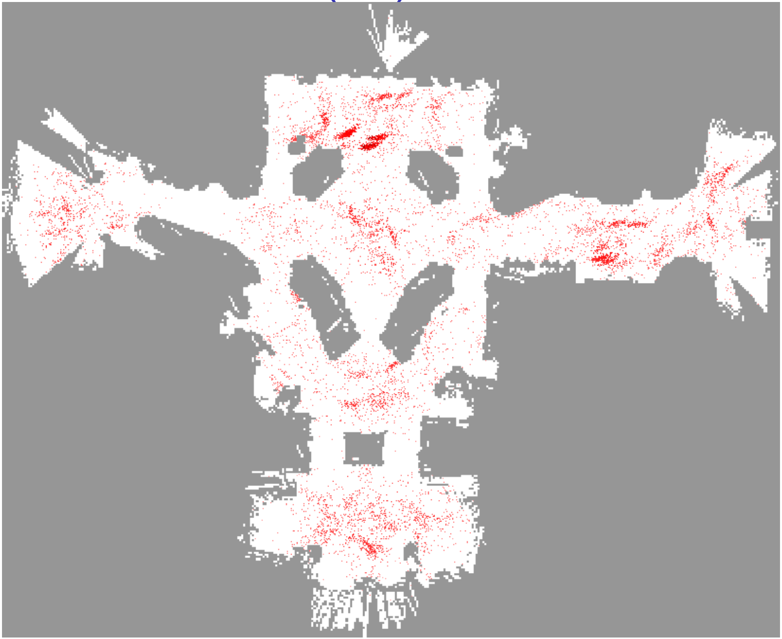
Particle Filter Localization (2-D)



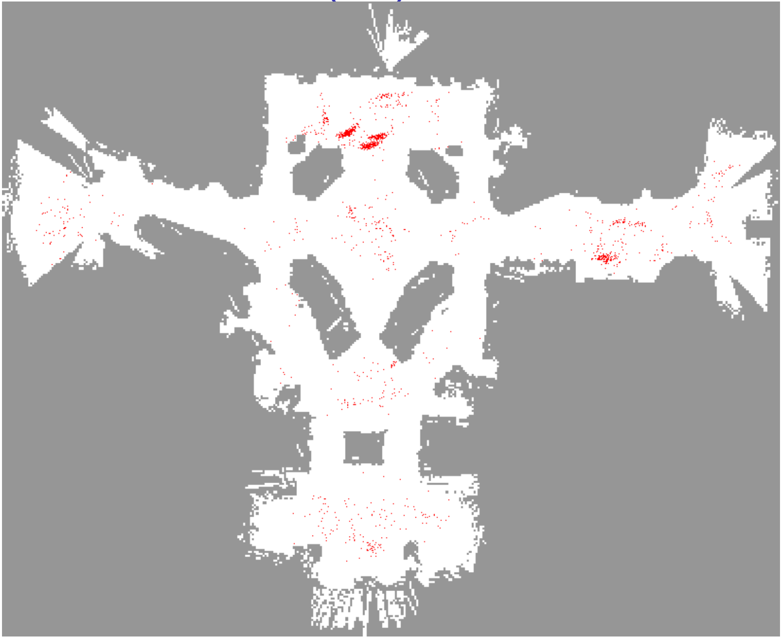
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

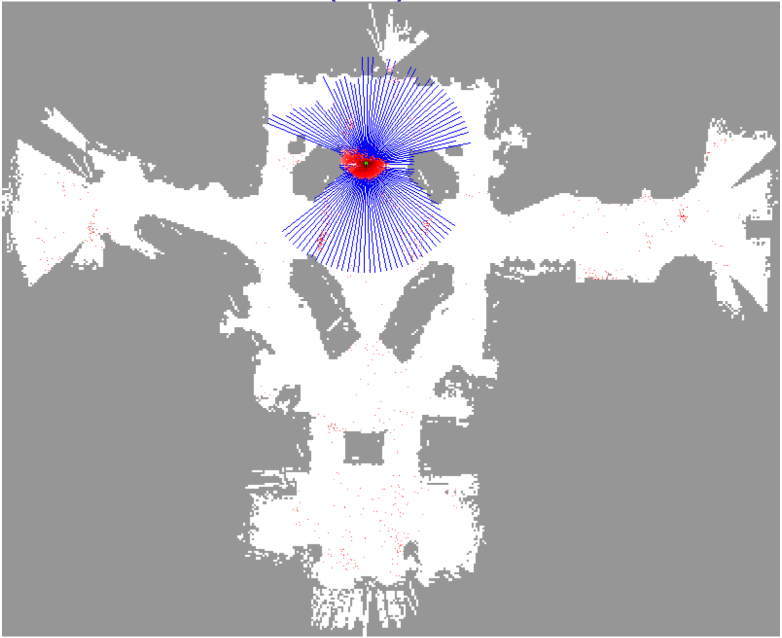


Particle Filter Localization (2-D)



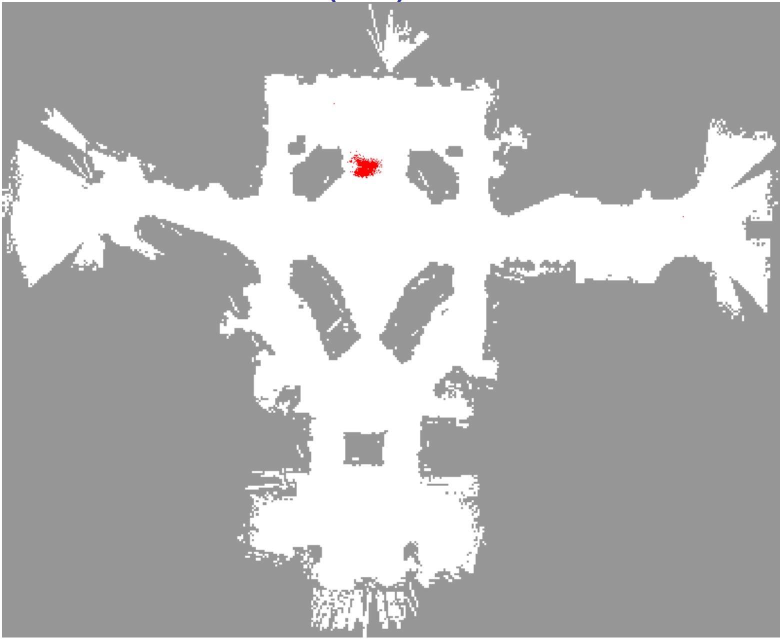
Particle Filter Localization (2-D)

Particle Filter Localization (2-D)

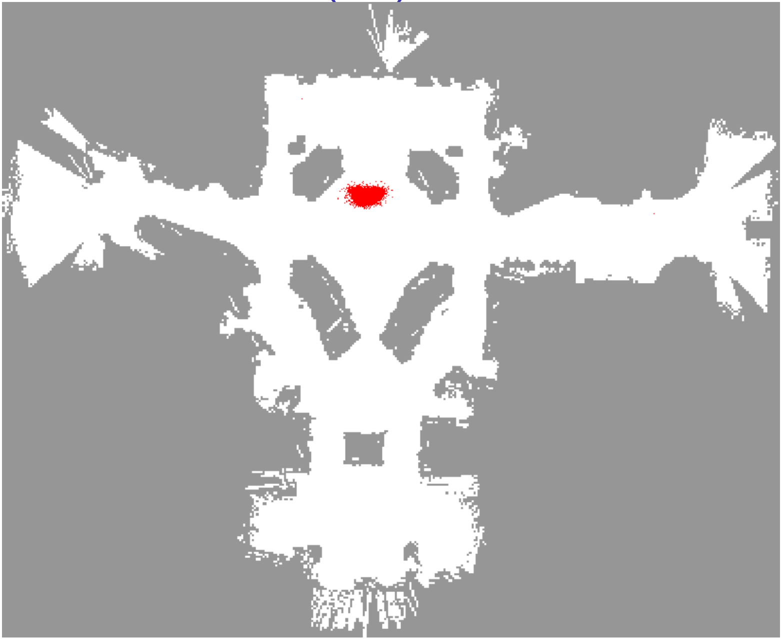


Particle Filter Localization (2-D)

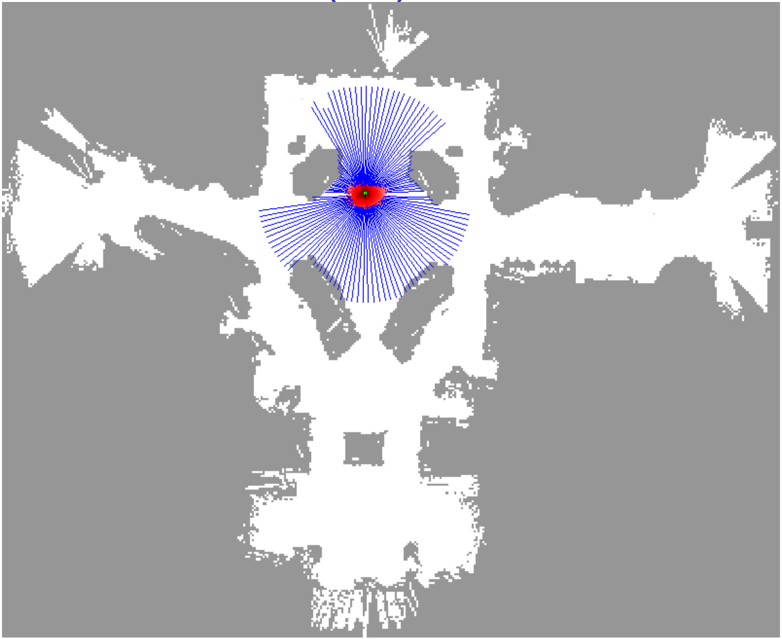
Particle Filter Localization (2-D)



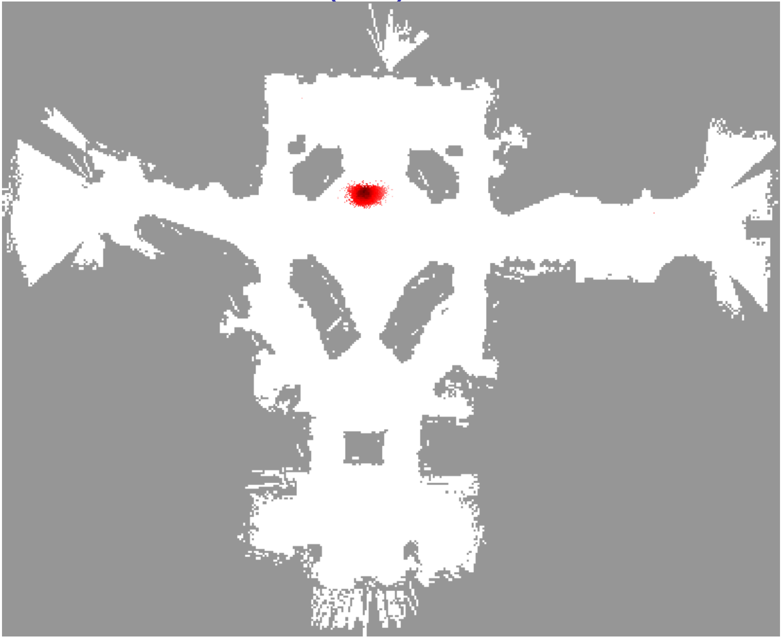
Particle Filter Localization (2-D)



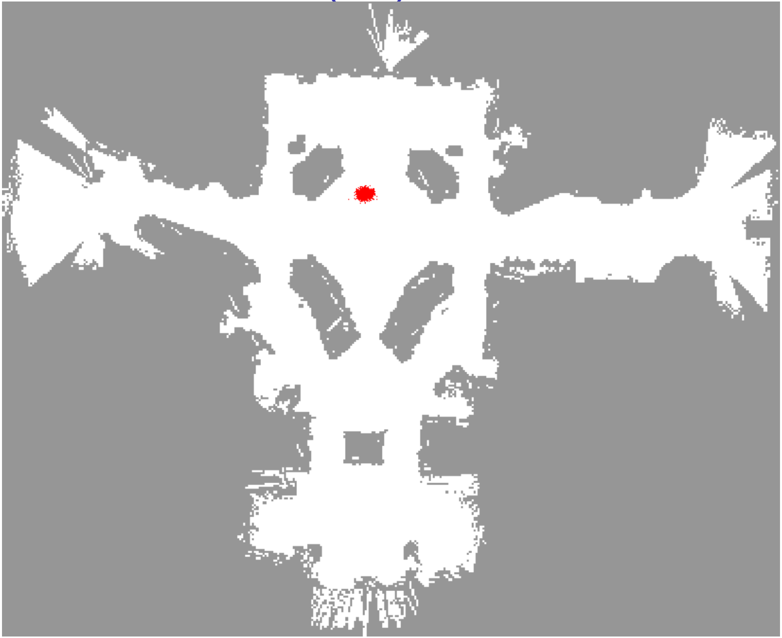
Particle Filter Localization (2-D)



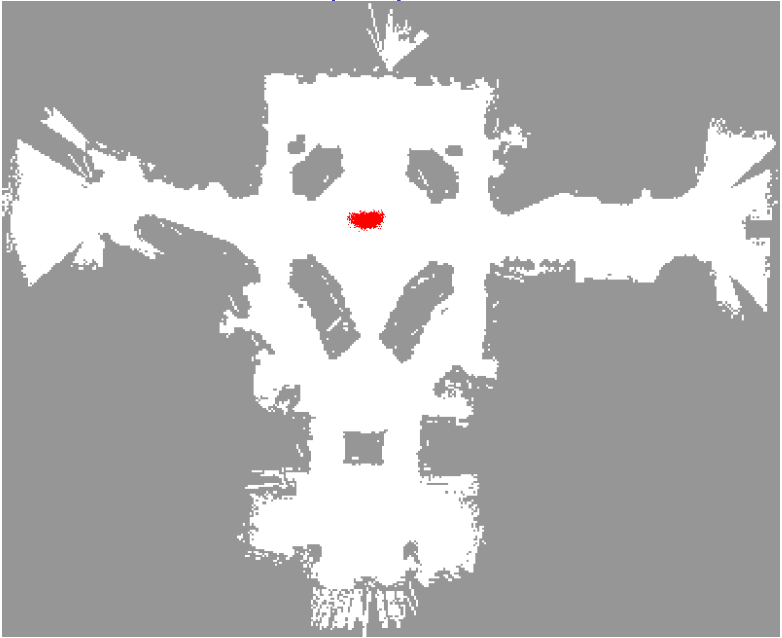
Particle Filter Localization (2-D)



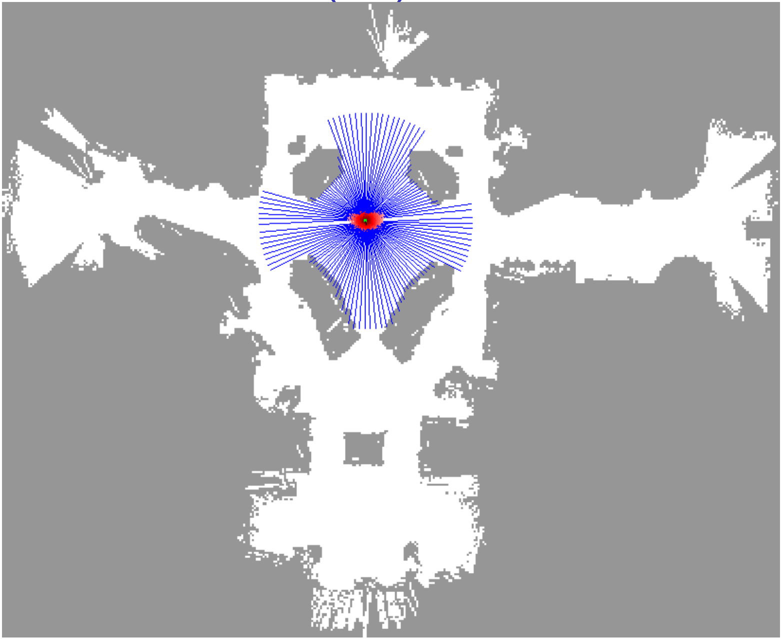
Particle Filter Localization (2-D)



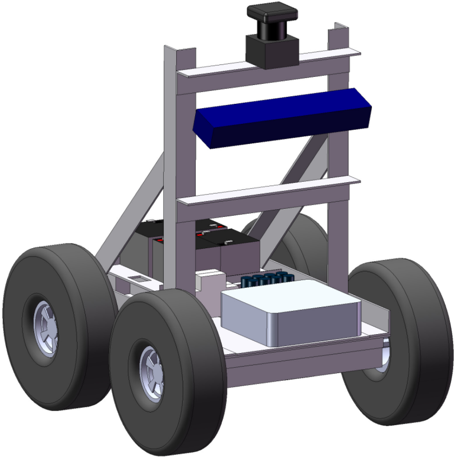
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

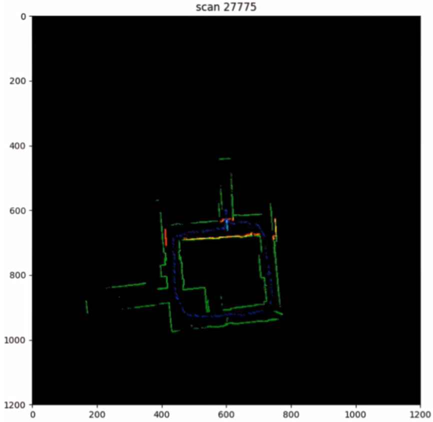


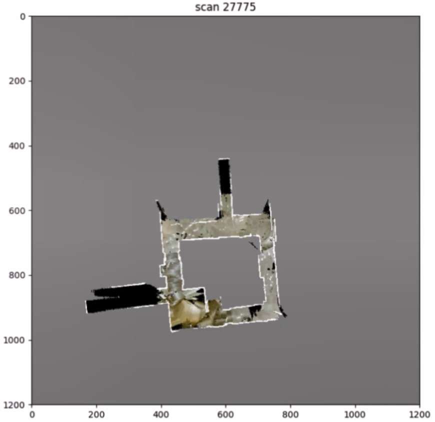
Project 2: Magic Differential-Drive Robot



- Wheel encoders
- IMU
- 2D Lidar
- RGBD camera

Project 2: Localization and Texture Mapping





Project 2: LiDAR-based Particle-filter SLAM

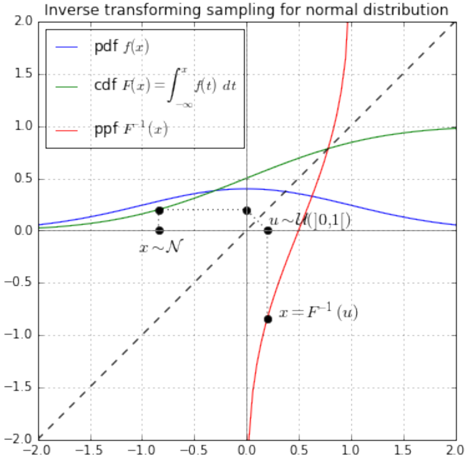
- Initial particle set $\boldsymbol{\mu}_{0|0}[k] = (0,0,0)^\top$ with weights $\alpha_{0|0}[k] = \frac{1}{N}$
- Use the first LiDAR scan to initialize the map:
  1. convert the scan to Cartesian coordinates
  2. transform the scan from the lidar frame to the body frame and then to the world frame
  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 v_t from the encoders and angular velocity ω_t from the gyroscope plus noise to predict the motion of each particle
- **Update step**: combines robot state update and map update:
  - convert the LiDAR scan $\mathbf{z}_{t+1}$ to the world frame from each particle's pose $\boldsymbol{\mu}_{t+1|t+1}[k]$ to compute map correlation and update the particle weights
  - choose the particle with highest weight $\alpha_{t+1|t+1}[k]$, project the LiDAR scan $\mathbf{z}_{t+1}$ to the world frame and update the map log-odds
- **Textured map**: use the RGBD images from the highest-weight particle's pose to assign colors to the occupancy grid cells

Outline

Inverse Transform Sampling

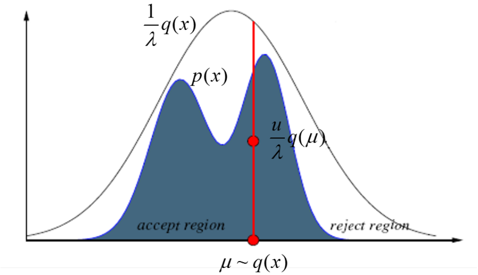
- How do we sample from a **target distribution** with pdf $p(x)$ and CDF $F(x) = \int_{-\infty}^{x} p(s)ds$?

- **Proposal distribution**: $\mathcal{U}(0,1)$
- **Inverse transform sampling**:
  1. Draw $u \sim \mathcal{U}(0,1)$
  2. Return inverse CDF value:
  $$\mu = F^{-1}(u)$$
  3. The CDF of $F^{-1}(u)$ is:
  $$\begin{aligned}\mathbb{P}(F^{-1}(u) \leq x) &= \mathbb{P}(u \leq F(x)) \\ &= F(x)\end{aligned}$$



Rejection Sampling

- Can we sample from a **target distribution** with pdf $p(x)$ without using its CDF $F(x)$?
- **Proposal distribution**: easy-to-sample pdf $q(x)$, e.g., Uniform or Gaussian, that satisfies $p(x) \leq \frac{1}{\lambda} q(x)$ for some $\lambda \in (0, 1)$
- **Rejection sampling**:
  1. Draw $\mu \sim q(\cdot)$ and $u \sim \mathcal{U}(0, 1)$
  2. Return μ only if $\frac{u}{\lambda} q(\mu) \leq p(\mu)$



- If λ is small, many rejections are necessary. Good $q(x)$ and λ are difficult to choose.

Sample Importance Resampling

- Can we sample from a **target distribution** with pdf $p(x)$ without scaling by $\lambda \in (0, 1)$?
- **Proposal distribution**: pdf $q(x)$
- **Sample importance resampling**
  1. Draw $\mu[1], \ldots, \mu[N]$ from $q(\cdot)$
  2. Compute importance weights $\alpha[k] = \frac{p(\mu[k])}{q(\mu[k])}$ and normalize $\alpha[k] = \frac{\alpha[k]}{\sum_j \alpha[j]}$
  3. Draw $\mu[k]$ independently with replacement from $\{\mu[1], \ldots, \mu[N]\}$ with probability $\alpha[k]$
- If $q(x)$ is a poor approximation of $p(x)$, then even the best samples from $q(x)$ may not be good samples for resampling

Particle Filter

- Particle filter: Monte-Carlo approximation of pdf $p_{t|t}(\mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$ using a finite weighted set of particles $\left\{ \boldsymbol{\mu}_{t|t}[k], \alpha_{t|t}[k] \right\}$ updated over time t

- Particles $\left\{ \boldsymbol{\mu}_{t|t}[k], \alpha_{t|t}[k] \right\}$ approximate $p_{t|t}(\mathbf{x}_t)$ in the sense that the weighted sum of any function g evaluated over the particle set converges to the expectation with respect to $p_{t|t}(\mathbf{x}_t)$:

$$\sum_{k=1}^{N} \alpha_{t|t}[k] g(\boldsymbol{\mu}_{t|t}[k]) \rightarrow \int g(\mathbf{x}_t) p_{t|t}(\mathbf{x}_t) d\mathbf{x}_t \quad \text{as } N \rightarrow \infty$$

- **Idea**: apply sample importance resampling to **target distribution**:

$$p(\mathbf{x}_{0:t+1} | \mathbf{z}_{0:t+1}, \mathbf{u}_{0:t}) = p(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t) p(\mathbf{x}_{0:t} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$

- **Proposal distribution**:

$$q(\mathbf{x}_{0:t+1} | \mathbf{z}_{0:t+1}, \mathbf{u}_{0:t}) = q(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t) q(\mathbf{x}_{0:t} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$

Sample Importance Resampling in the Particle Filter

1. **Sample** $\boldsymbol{\mu}_{t+1|t+1}[1], \ldots, \boldsymbol{\mu}_{t+1|t+1}[N]$ from $q(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t)q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$
   - Since $\boldsymbol{\mu}_{t|t}[1], \ldots, \boldsymbol{\mu}_{t|t}[N]$ from $q(\mathbf{x}_{0:t}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$ are already available from the prior, we only need to sample:
   $$\boldsymbol{\mu}_{t+1|t+1}[k] \sim q(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t = \boldsymbol{\mu}_{t|t}[k]) \qquad \forall k = 1, \ldots, N$$
   - The performance depends on the choice of proposal $q(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t)$
   - Common proposal choice: **motion model** $q(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{u}_t, \mathbf{x}_t) = p_f(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$; easy to sample from but may be suboptimal because $\mathbf{z}_{t+1}$ is not considered
2. **Compute and normalize importance weights**:
$$\begin{aligned}
\alpha_{t+1|t+1}[k] &\propto \frac{p(\boldsymbol{\mu}_{t+1|t+1}[k]|\mathbf{z}_{t+1}, \mathbf{u}_t, \boldsymbol{\mu}_{t|t}[k])p(\boldsymbol{\mu}_{t|t}[k]|\mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})}{q(\boldsymbol{\mu}_{t+1|t+1}[k]|\mathbf{z}_{t+1}, \mathbf{u}_t, \boldsymbol{\mu}_{t|t}[k])q(\boldsymbol{\mu}_{t|t}[k]|\mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})}\\
&= \frac{p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t+1}[k])p_f(\boldsymbol{\mu}_{t+1|t+1}[k]|\boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t)}{p_f(\boldsymbol{\mu}_{t+1|t+1}[k]|\boldsymbol{\mu}_{t|t}[k], \mathbf{u}_t)}\alpha_{t|t}[k]\\
&= p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t+1}[k])\alpha_{t|t}[k]
\end{aligned}$$
3. **Resample**: if N_{eff} is small, draw $\mu_{t+1|t+1}[k]$ independently with replacement from $\left\{\mu_{t+1|t+1}[1], \ldots, \mu_{t+1|t+1}[N]\right\}$ with probability $\alpha_{t+1|t+1}[k]$ and reset the weights to $1/N$