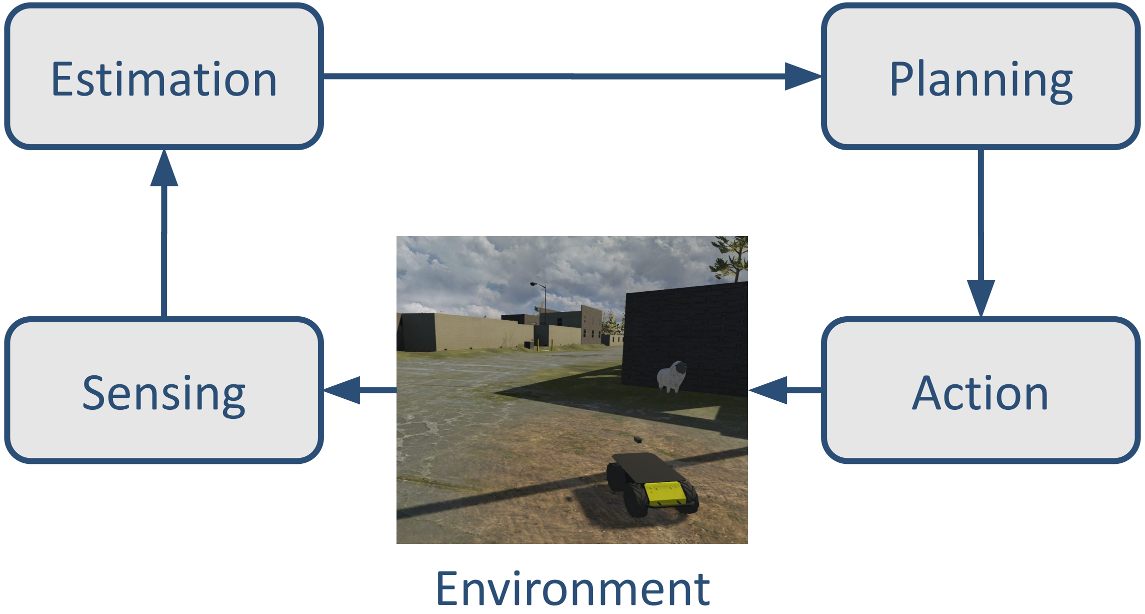
Localization and Mapping

Arash Asgharivaskasi Nikolay Atanasov
Existential Robotics Laboratory
University of California, San Diego

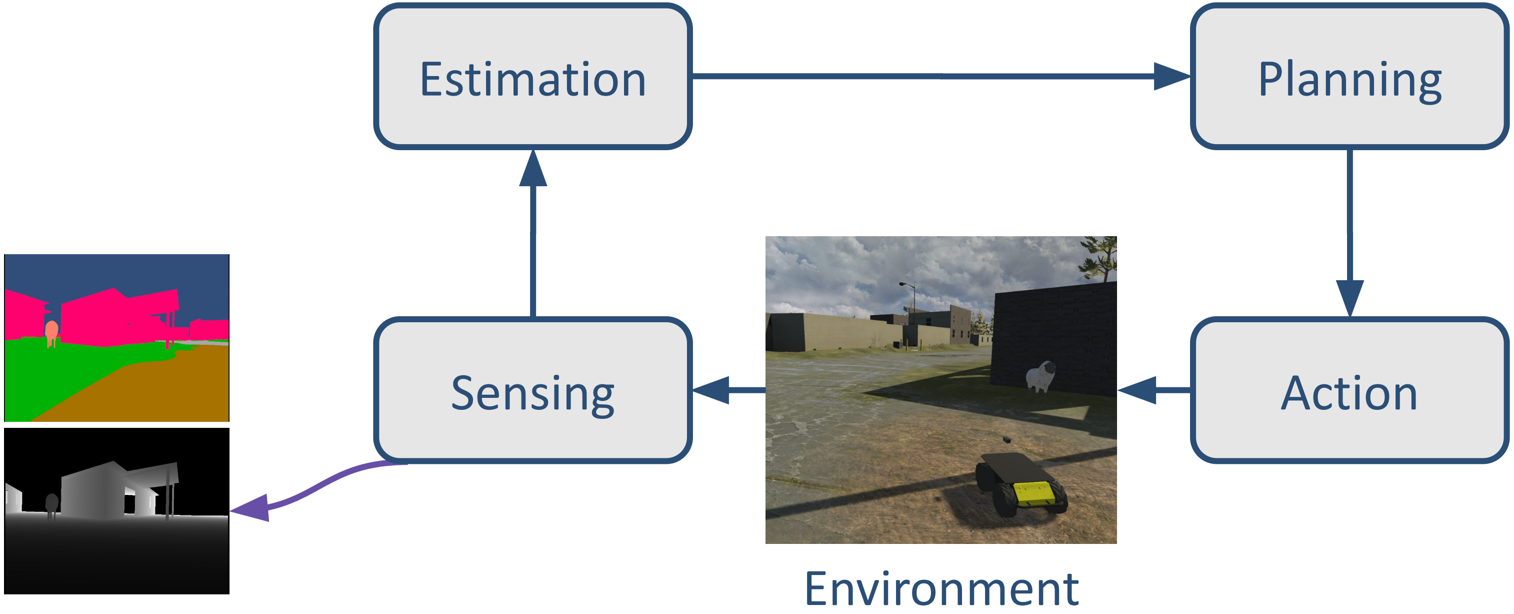
CONTEXTUAL ROBOTICS INSTITUTE

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

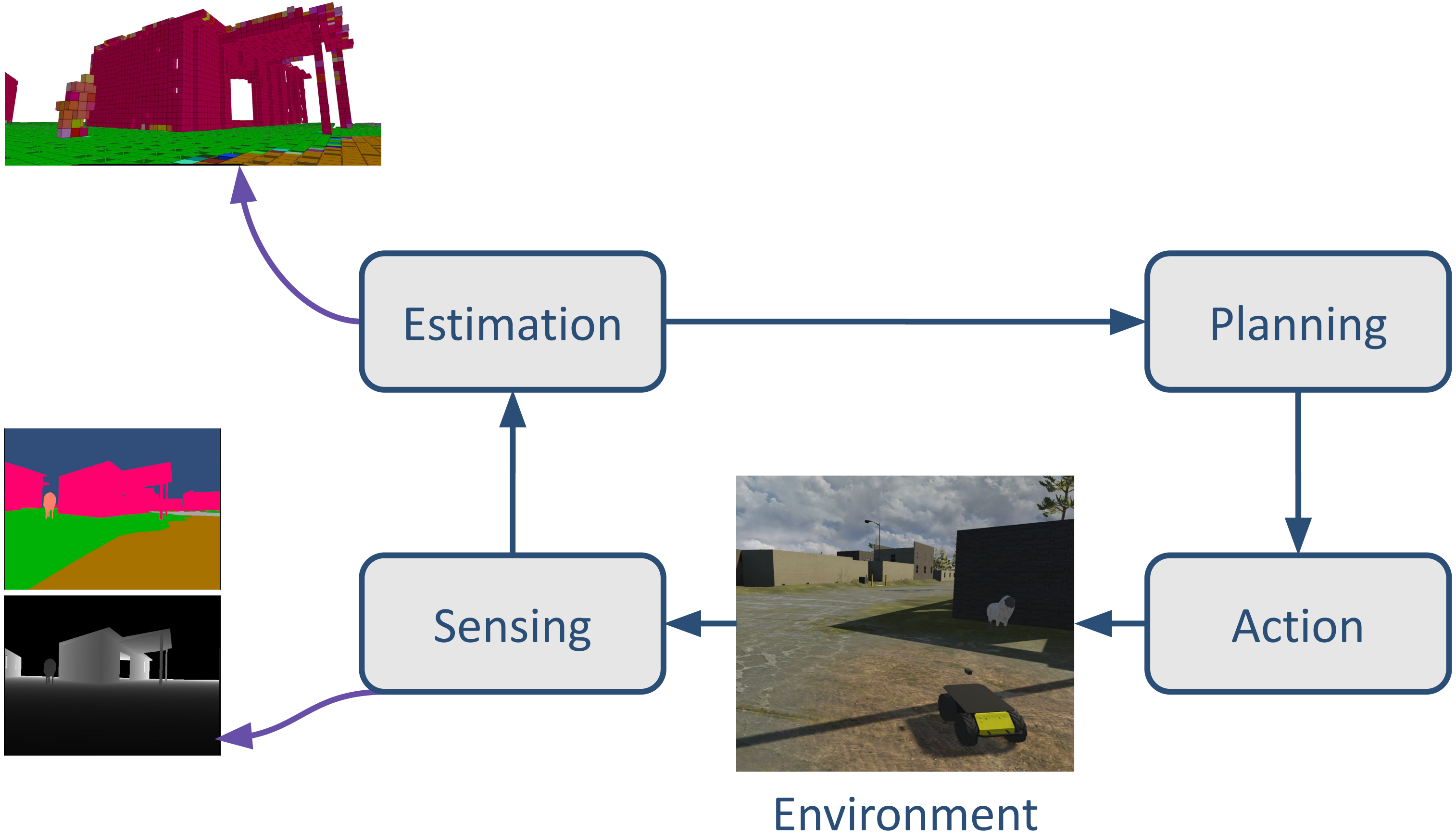
Robot Autonomy



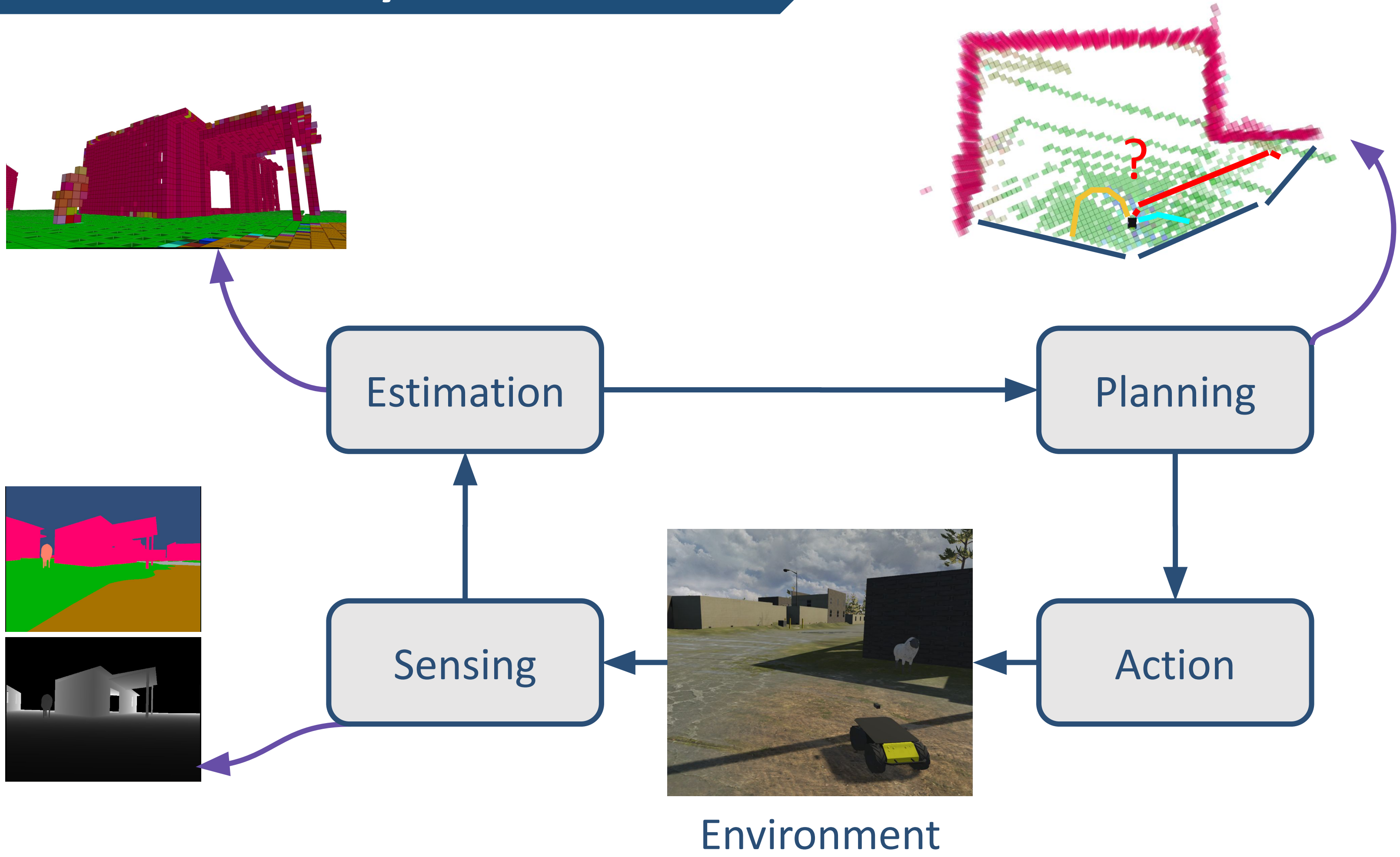
Robot Autonomy



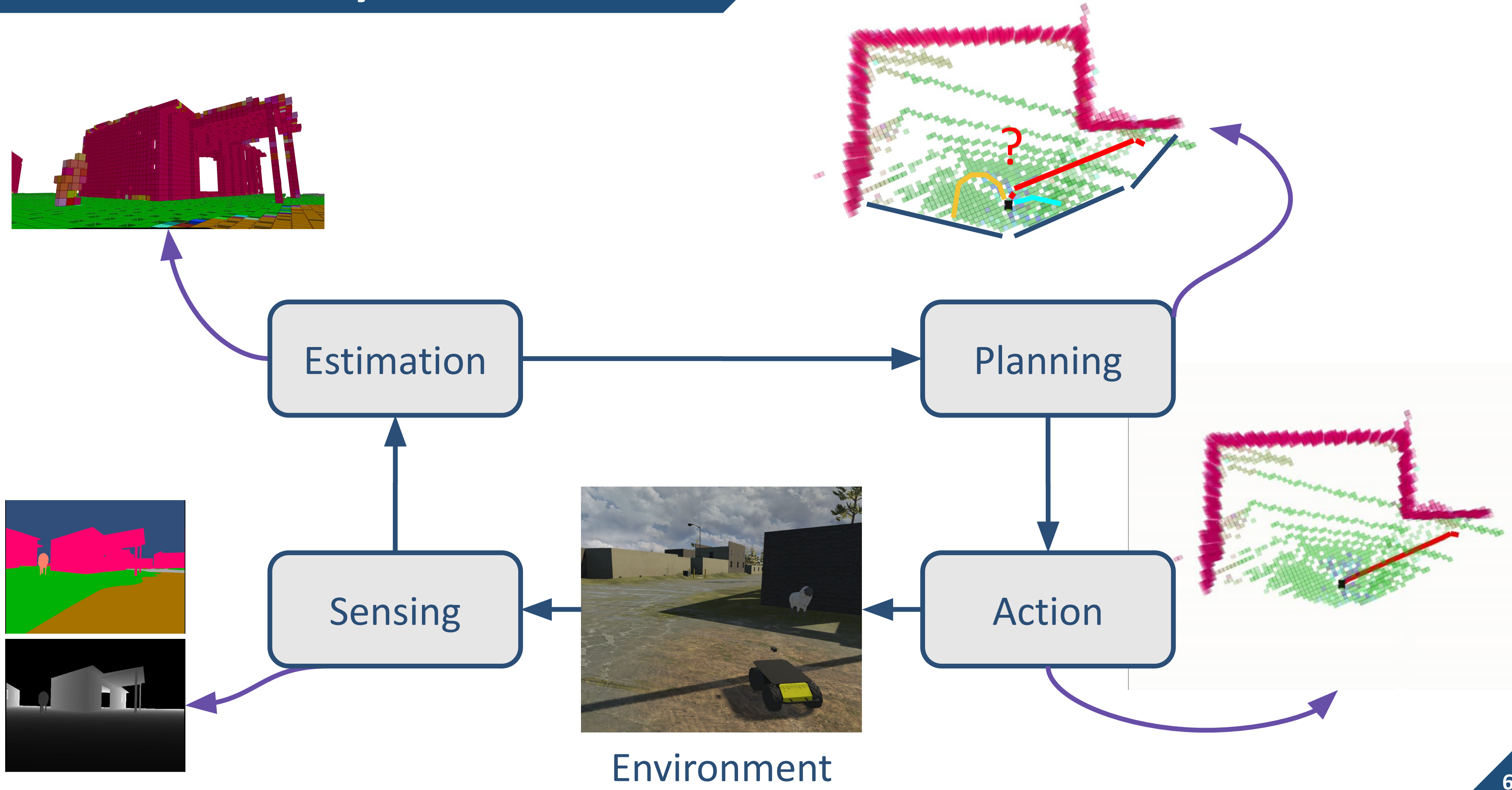
Robot Autonomy



Robot Autonomy

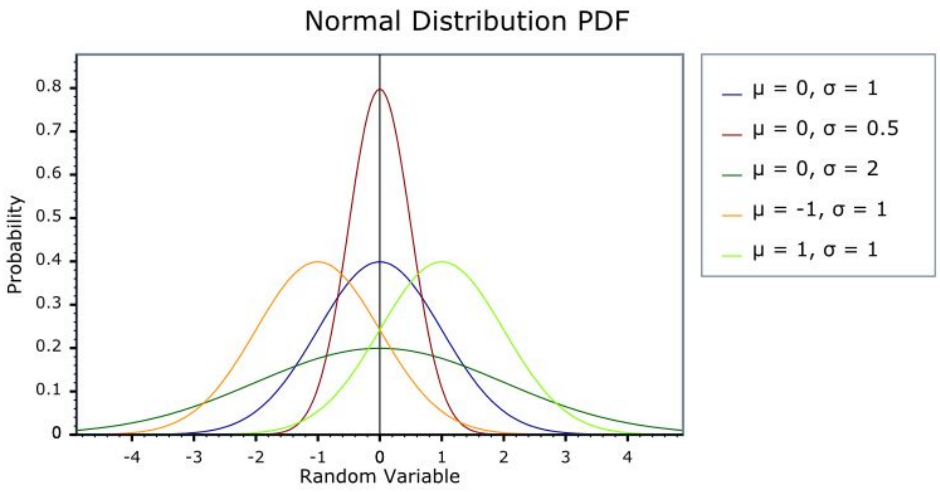


Robot Autonomy



Robot Autonomy

Probability Review



Probability Axioms

- For an event A in sample space □, we assume that the following is always true:

1. $P(A) \geq 0$
2. $P(\Omega) = 1$
3. If $A_i \cap A_j = \varnothing,\ \forall i \neq j$, then $P\left(\bigcup_i A_i\right) = \sum_i P(A_i)$

- Corollary:

1. $P(\varnothing) = 0$
2. $\max\{P(A), P(B)\} \leq P(A \cup B) = P(A) + P(B) - P(A \cap B) \leq P(A) + P(B)$
3. $A \subseteq B \Rightarrow P(A) \leq P(B)$

Probability Formulas

- Total Probability:

$$P(B) = \sum_{i=1}^{n} P\left(B \cap A_i\right),\ \text{if } \Omega = \bigcup_i A_i \text{ and } A_i \cap A_j = \varnothing,\ \forall i \neq j$$

- Conditional Probability:

$$P(A \cap B) = P(A|B)P(B)$$

- Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},\ P(B) \neq 0$$

- Corollary:

$$P\left(A_i|B\right) = \frac{P\left(B|A_i\right)P\left(A_i\right)}{\sum_{i=1}^{n} P\left(B|A_i\right)P\left(A_i\right)},\ \text{if } \Omega = \bigcup_i A_i \text{ and } A_i \cap A_j = \varnothing,\ \forall i \neq j$$

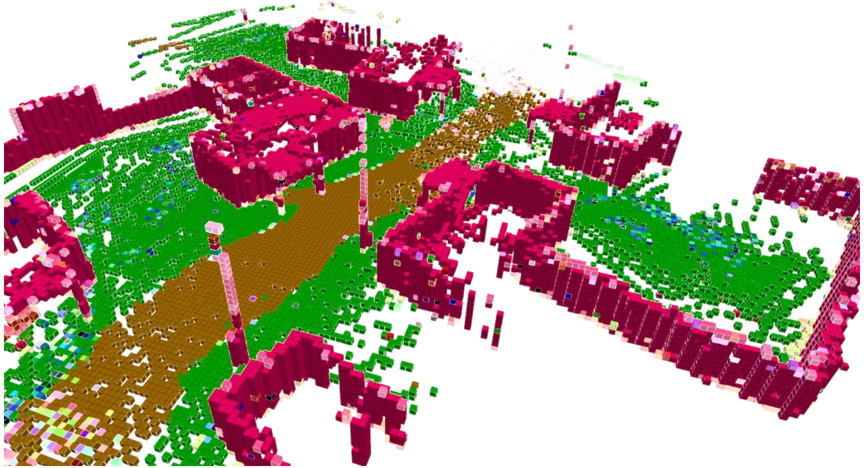
Independent Events

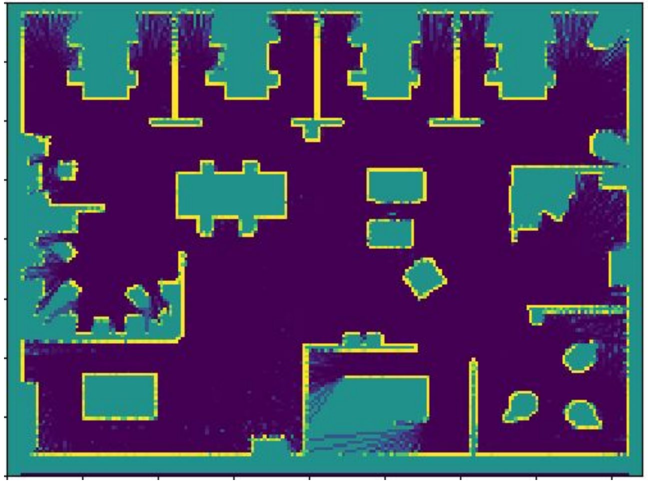
- Events A and B are independent if and only if:

$$\mathrm{P}(\mathrm{A}|\mathrm{B}) = \mathrm{P}(\mathrm{A}) \Leftrightarrow \mathrm{P}(\mathrm{A} \cap \mathrm{B}) = \mathrm{P}(\mathrm{A})\mathrm{P}(\mathrm{B})$$

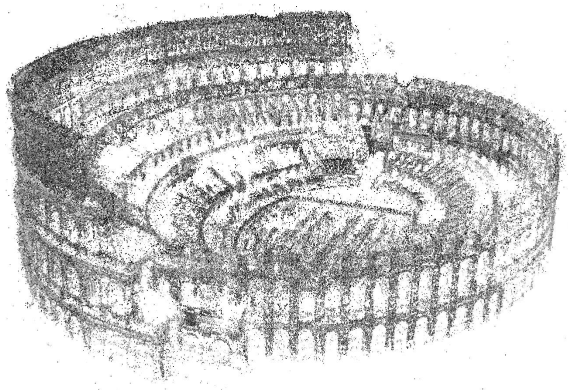
- Observing one does not give any information about another
- Disjoint events are always dependent (why?)

Mapping

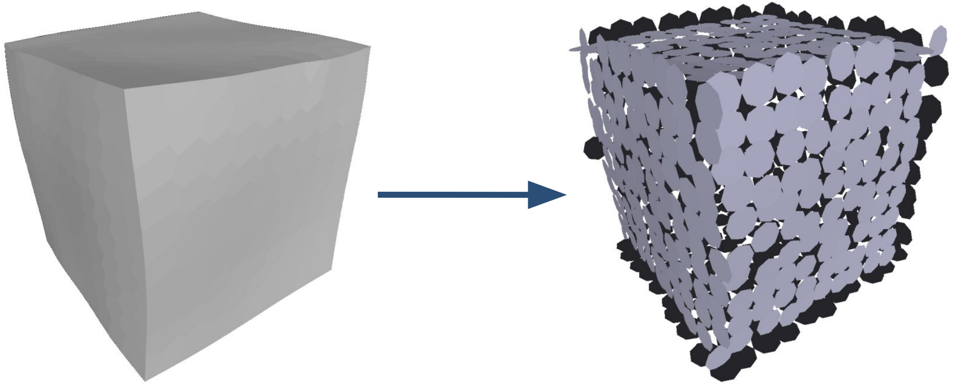




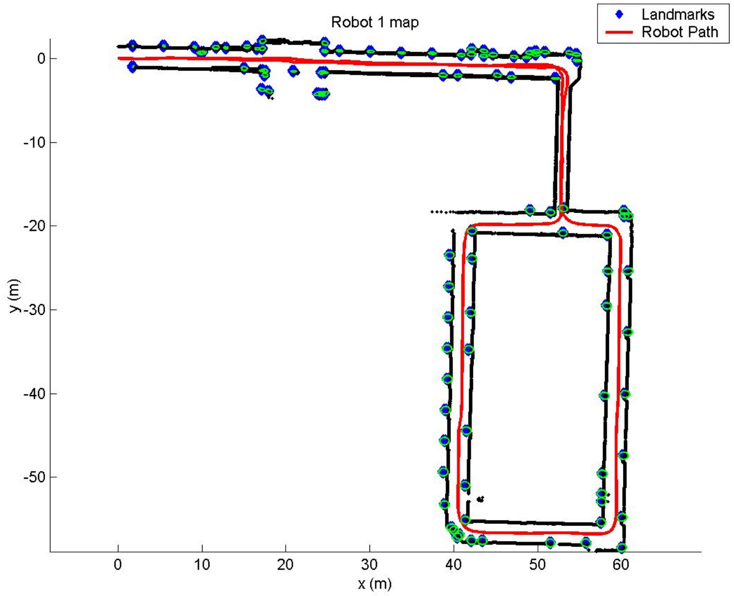
Map Representations (Sparse)



Point Cloud

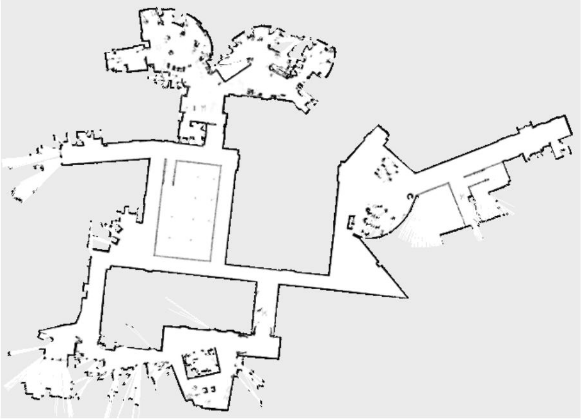


Surfels

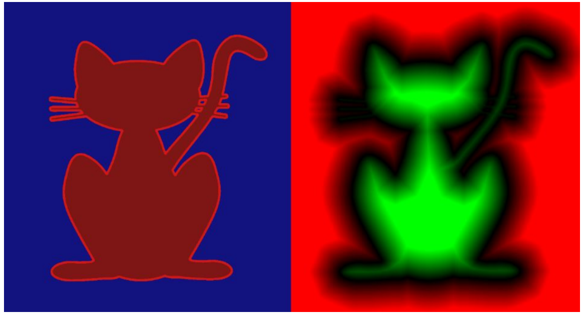


Landmark-based

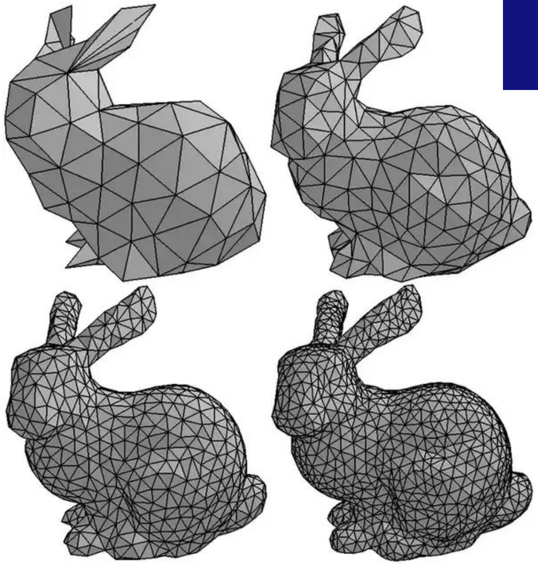
Map Representations (Dense)



Occupancy Grid Map

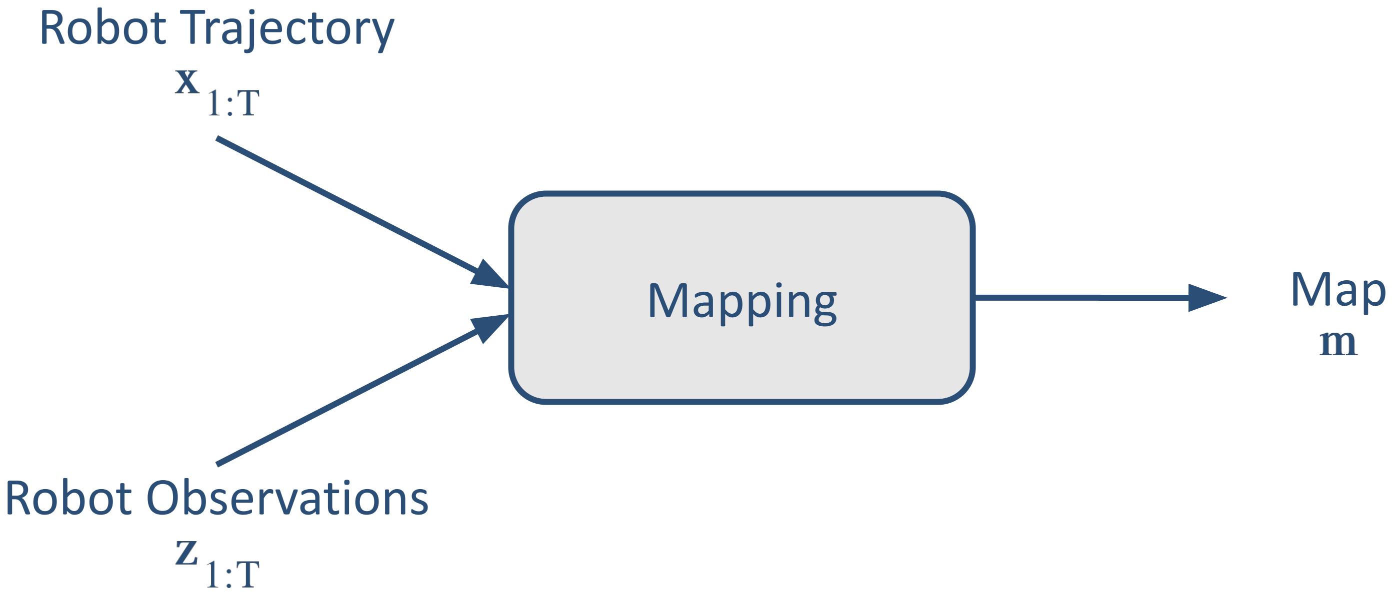


Signed Distance Field

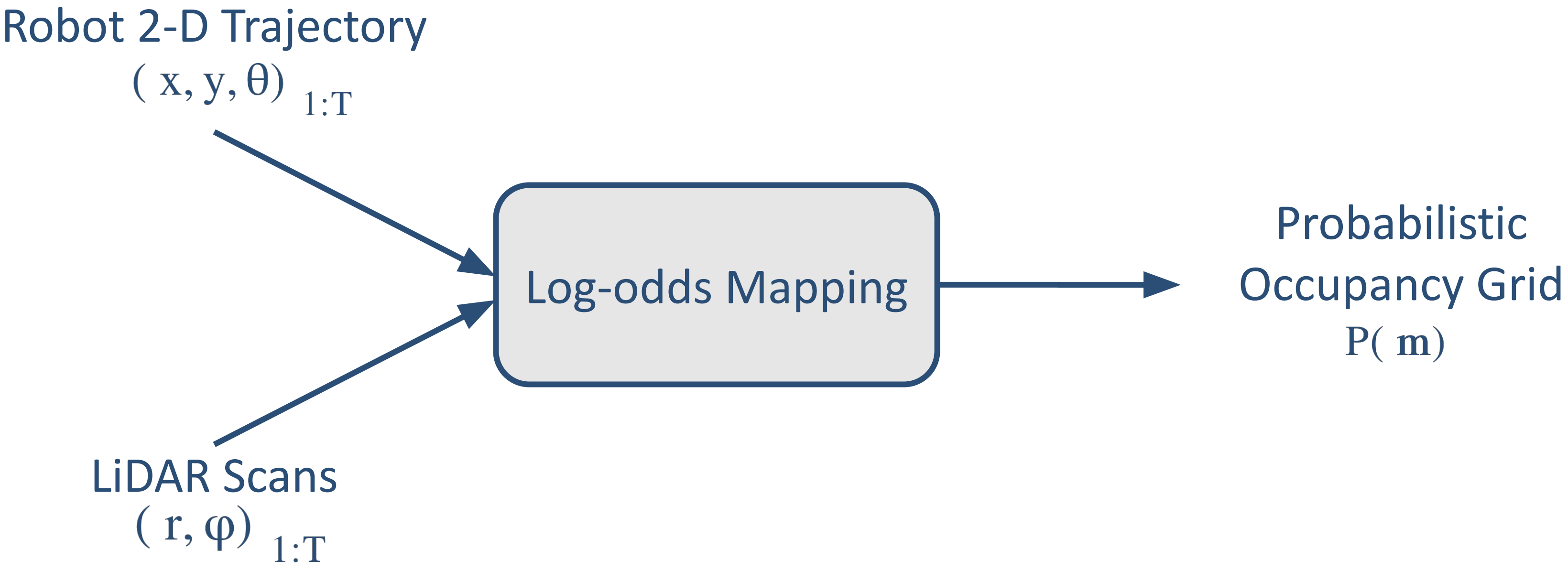


Polygon Mesh

Problem Formulation



Problem Formulation

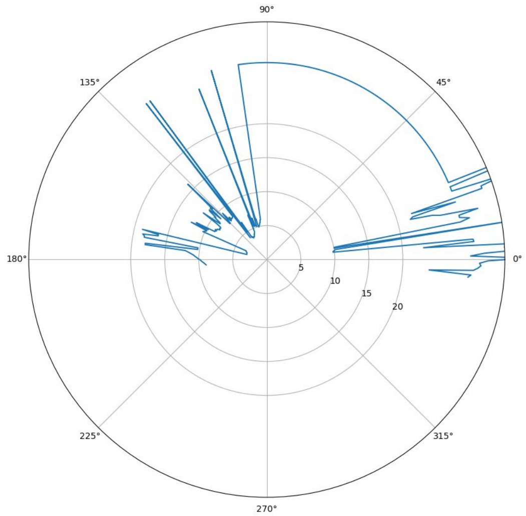


LiDAR Scanner

- LiDAR: Light Detection And Ranging

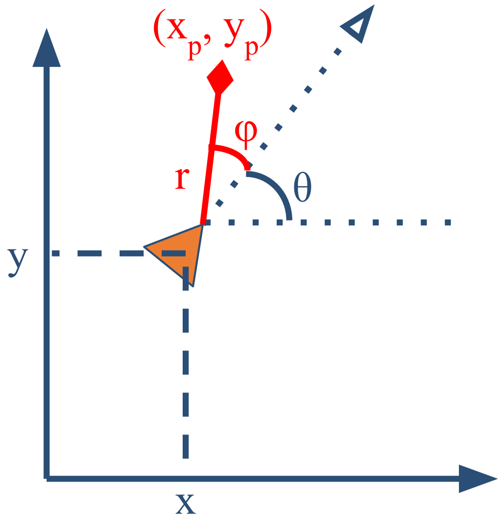
- Illuminates the scene with pulsed laser light and measures the return times and wavelengths of the reflected pulses

- Each LiDAR scan provides a collection of range measurements $\{r_1, \cdots, r_B\}$ over a set of known angles $\{\varphi_1, \cdots, \varphi_B\}$ where B is the number of rays



Point Cloud Position

- Given robot pose (x, y, θ) and a range measurement (r, φ), the location (x_p, y_p) of the observed point can be found by polar to Cartesian transformation:



$$x_p = x + r\cos(\phi + \theta)$$

$$y_p = y + r\sin(\phi + \theta)$$

Demo

1. Visit the wiki page of the workshop
2. Navigate to the Localization and Mapping page
3. Run Demo 1

Point Cloud Representation

- Pros:
  - Does not require ray-tracing
  - Light-weight computation

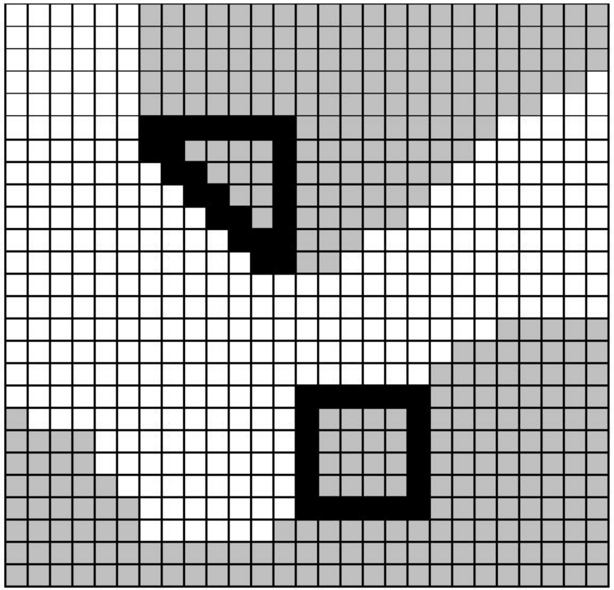
- Cons:
  - Memory inefficient: map size grows with the number of observations
  - Not suitable for collision avoidance

Occupancy Grid Mapping

- One of the simplest yet widely used map representations

- The environment is divided into a regular grid of N cells

- The value of each cell indicates its occupancy (1=occupied and 0=free)



Log-odds Mapping

- We represent the map $\mathbf{m}$ as a vector where an element m_i is a cell

- Due to the measurement noise and the error in robot location, we need to maintain a probability distribution over the map as $P(\mathbf{m}|\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$

- For simplicity, we assume that map cells are independent:

$$P\left(\mathbf{m}|\mathbf{x}_{1:T}, \mathbf{z}_{1:T}\right) = \prod_{i=1}^{N} P\left(m_i|\mathbf{x}_{1:T}, \mathbf{z}_{1:T}\right)$$

- Therefore, we can compute occupancy probability for each cell m_i separately

Log-odds Mapping

- Given robot trajectory $\mathbf{x}_{1:T}$ and observations $\mathbf{z}_{1:T}$, let $\boldsymbol{\gamma}_{i,T}$ be the probability that m_i is occupied, i.e.:

$$\gamma_{i,T} = P\left(m_i = 1 | \mathbf{x}_{1:T}, \mathbf{z}_{1:T}\right)$$

$$\overset{\text{Bayes Rule}}{=} \frac{1}{\eta_T} P\left(\mathbf{z}_T | m_i = 1, \mathbf{x}_T\right) P\left(m_i = 1 | \mathbf{x}_{1:T-1}, \mathbf{z}_{1:T-1}\right) = \frac{1}{\eta_T} P\left(\mathbf{z}_T | m_i = 1, \mathbf{x}_T\right) \gamma_{i,T-1}$$

$$1 - \gamma_{i,T} = P\left(m_i = 0 | \mathbf{x}_{1:T}, \mathbf{z}_{1:T}\right)$$

$$\overset{\text{Bayes Rule}}{=} \frac{1}{\eta_T} P\left(\mathbf{z}_T | m_i = 0, \mathbf{x}_T\right) \left(1 - \gamma_{i,T-1}\right)$$

$$\Rightarrow \boxed{\frac{\gamma_{i,T}}{1 - \gamma_{i,T}} = \frac{P\left(\mathbf{z}_T | m_i = 1, \mathbf{x}_T\right)}{P\left(\mathbf{z}_T | m_i = 0, \mathbf{x}_T\right)} \frac{\gamma_{i,T-1}}{1 - \gamma_{i,T-1}}}$$

Odds Ratio

Inverse Observation Model

- Using Bayes rule once more, we have:

$$\frac{P(\mathbf{z}_T | m_i = 1, \mathbf{x}_T)}{P(\mathbf{z}_T | m_i = 0, \mathbf{x}_T)} = \underbrace{\frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)}}_{\text{inverse observation model}} \underbrace{\frac{P(m_i = 0)}{P(m_i = 1)}}_{\text{map prior}}$$

- The inverse observation model simply encodes our trust in the sensors:

$$\frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)} = \begin{cases} TP/FP & \mathbf{z}_T \text{ indicates occupied} \\ FN/TN & \mathbf{z}_T \text{ indicates free} \end{cases}$$

Inverse Observation Model

- A "good" sensor has high TP and TN, and small FP and FN

$$\frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)} = \begin{cases} TP/FP & \mathbf{z}_T \text{ indicates occupied} \\ FN/TN & \mathbf{z}_T \text{ indicates free} \end{cases}$$

$$\frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)} = \boxed{\begin{cases} 10 \\ 0.1 \end{cases}} \quad \boxed{\begin{cases} 0.4 \\ 0.4 \end{cases}} \quad \boxed{\begin{cases} 1 \\ 1 \end{cases}} \quad \boxed{\begin{cases} 10 \\ 10 \end{cases}} \quad \begin{matrix} \mathbf{z}_T \text{ indicates occupied} \\ \mathbf{z}_T \text{ indicates free} \end{matrix}$$

Good Sensor | Bad Sensor | Bad Sensor | Bad Sensor

Log-odds Mapping

- Going back to the odds ratio formula:

$$\frac{\gamma_{i,T}}{1-\gamma_{i,T}} = \frac{P(\mathbf{z}_T | m_i = 1, \mathbf{x}_T)}{P(\mathbf{z}_T | m_i = 0, \mathbf{x}_T)} \frac{\gamma_{i,T-1}}{1-\gamma_{i,T-1}} = \frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)} \frac{P(m_i = 0)}{P(m_i = 1)} \frac{\gamma_{i,T-1}}{1-\gamma_{i,T-1}}$$

- Taking log of both sides yields:

$$\underbrace{\log\left(\frac{\gamma_{i,T}}{1-\gamma_{i,T}}\right)}_{\lambda_{i,T}} = \underbrace{\log\left(\frac{P(m_i = 1 | \mathbf{z}_T, \mathbf{x}_T)}{P(m_i = 0 | \mathbf{z}_T, \mathbf{x}_T)}\right)}_{\Delta\lambda_{i,T}(\mathbf{z}_T)} + \underbrace{\log\left(\frac{\gamma_{i,T-1}}{1-\gamma_{i,T-1}}\right)}_{\lambda_{i,T-1}} - \underbrace{\log\left(\frac{P(m_i = 1)}{P(m_i = 0)}\right)}_{\lambda_{i,0}}$$

- Therefore, the map update can be expressed as accumulating log of odds ratios over time; hence the name log-odds mapping:

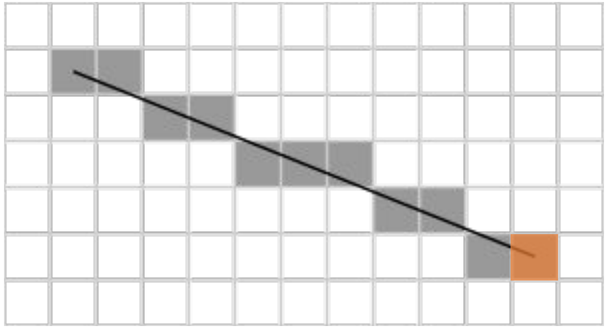
$$\boxed{\lambda_{i,T} = \Delta\lambda_{i,T}(\mathbf{z}_T) + \lambda_{i,T-1} - \lambda_{i,0}}$$

Log-odds Mapping

- In order to recover cell occupancy probability from log-odds value, we use the logistic sigmoid function:

$$P\left(m_i = 1 \mid \mathbf{x}_{1:T}, \mathbf{z}_{1:T}\right) = \gamma_{i,T} = \sigma\left(\lambda_{i,T}\right) = \frac{\exp\left(\lambda_{i,T}\right)}{1 + \exp\left(\lambda_{i,T}\right)}$$

- For occupancy grid mapping using LiDAR, we need to determine the cells visited by each ray



- This process is called line rasterization

Log-odds Mapping (Summary)

- Log-odds occupancy grid mapping is equivalent to maintaining a grid of log-odds values $\lambda_{i,T}$ for each cell

- There is a one-to-one relationship between log-odds value $\lambda_{i,T}$ and occupancy probability $\boldsymbol{\gamma}_{i,T}$, namely the logistic sigmoid function

- Upon each LiDAR measurement $\mathbf{z}_T$, follow the below steps:
  - For each ray, find the set of visited cells using line rasterization
  - Use the log-odds formula in order to update the map:
    - If cell i is observed free:

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

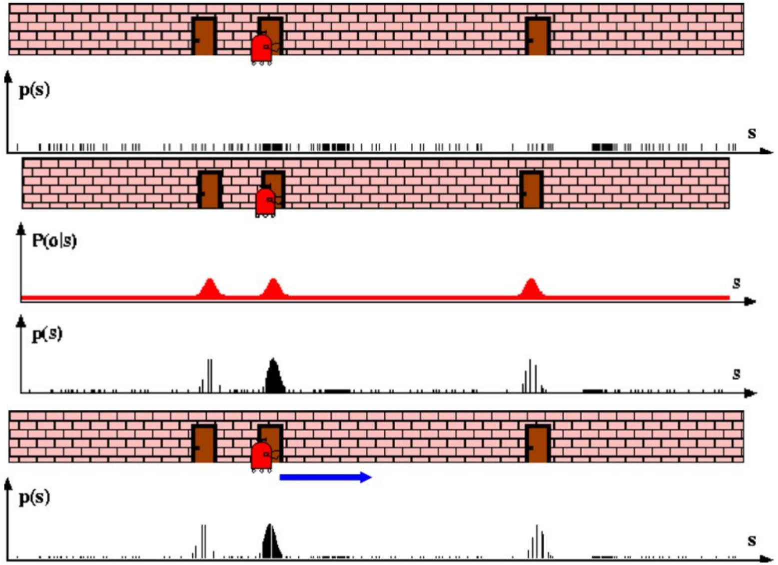
    - If cell i is observed occupied:

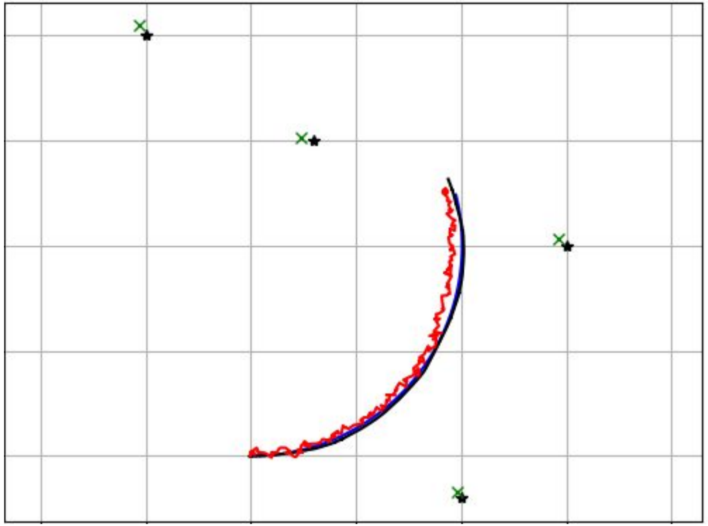
$$\lambda_{i,T} = \Delta\lambda_{i,T}(occupied) + \lambda_{i,T-1} - \lambda_{i,0}$$

Demo

1. Visit the wiki page of the workshop
2. Navigate to the Localization and Mapping page
3. Run Demo 2

Localization



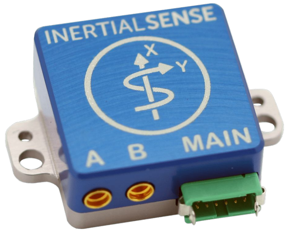


Robot Localization

- Odometry: The use of motion sensor data to estimate change of robot pose

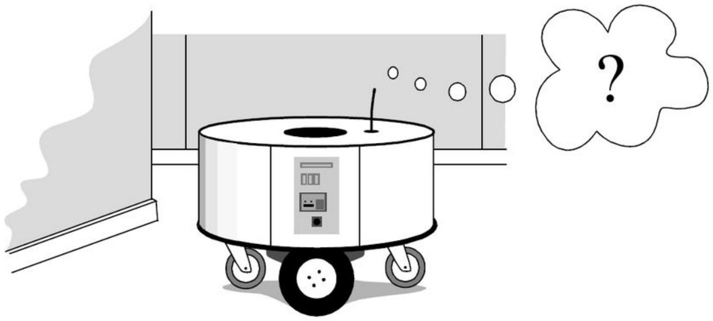
- Motion sensors usually have small measurement noise:
  - Accurate between t and t+1
  - Error accumulates during long time horizon

- Therefore, simply relying on odometry leads to large error in robot pose estimation



IMU

Wheel Encoder

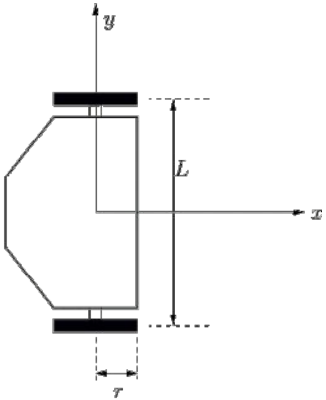


Differential-drive Model

- Consider the differential-drive motion model

- The velocity of left and right wheels control the motion of the robot:
  - Sum of wheels velocities: Robot linear velocity
  - Difference of wheels velocity: Robot angular velocity

- Therefore, the control inputs are linear velocity v and angular velocity ω

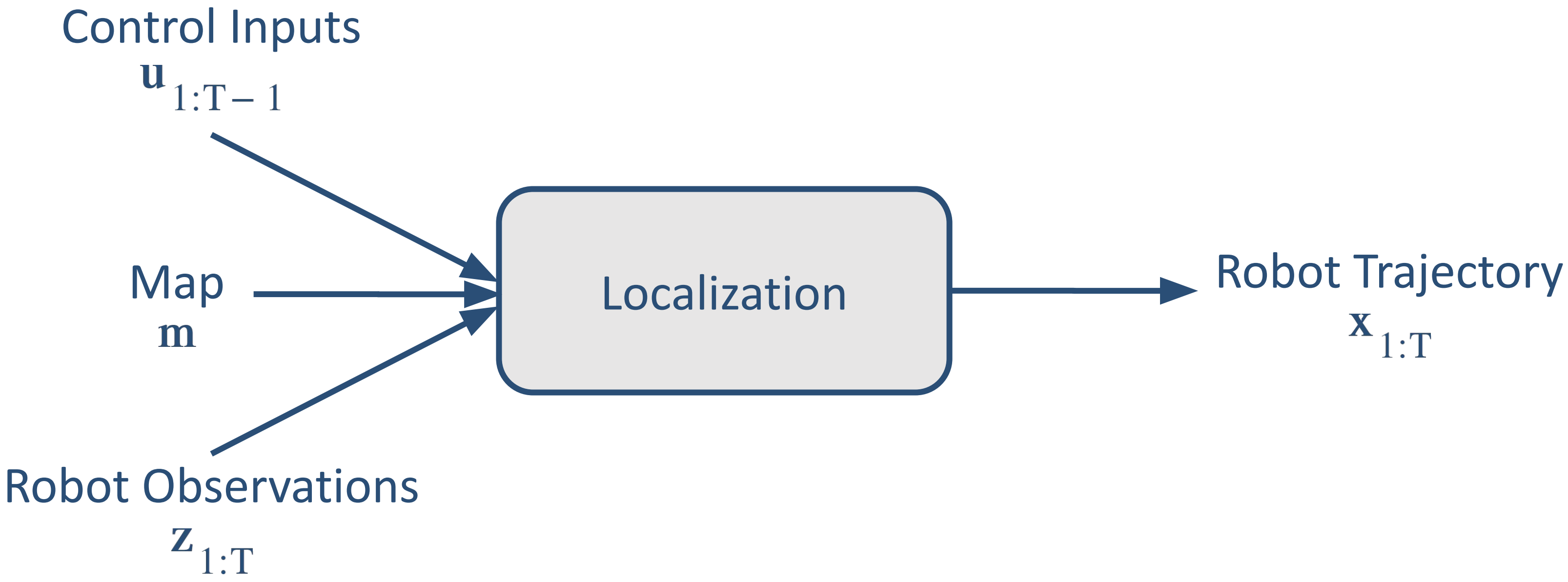


$$x_{t+1} = x_t + \tau\, v_t \cos(\theta_t)$$
$$y_{t+1} = y_t + \tau\, v_t \sin(\theta_t)$$
$$\theta_{t+1} = \theta_t + \tau\, \omega_t$$

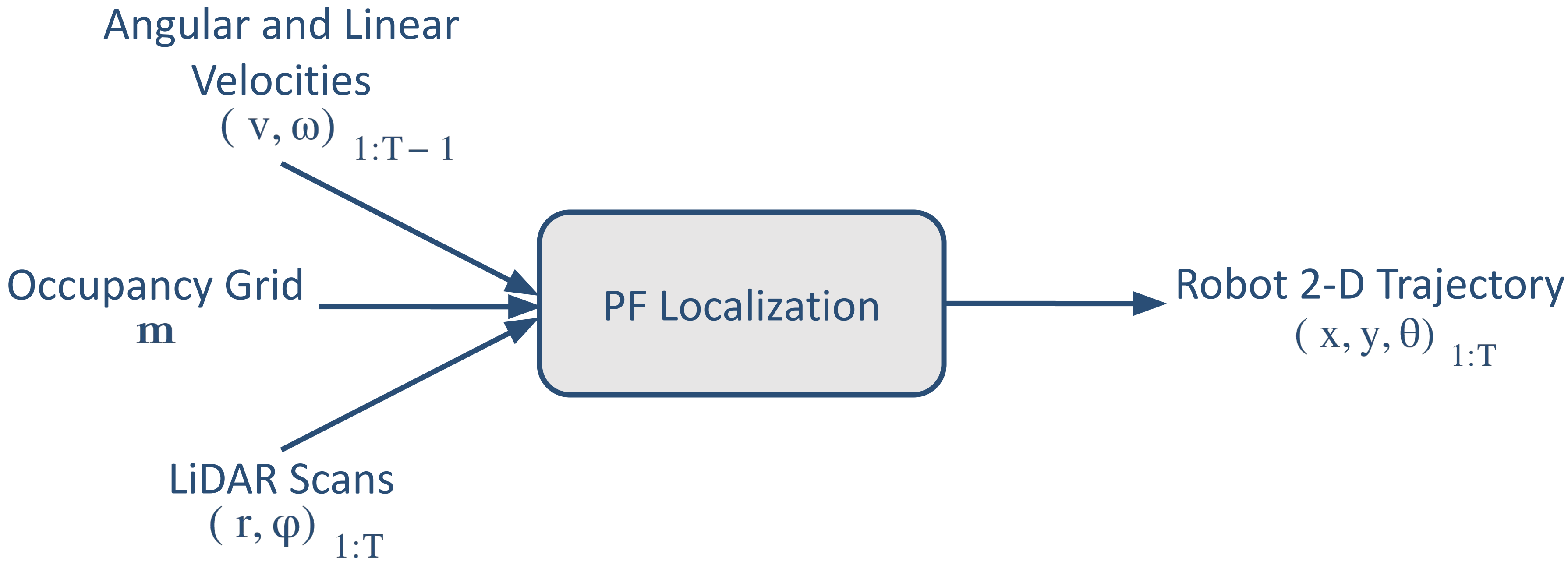
Demo

1. Visit the wiki page of the workshop
2. Navigate to the Localization and Mapping page
3. Run Demo 3

Problem Formulation



Problem Formulation



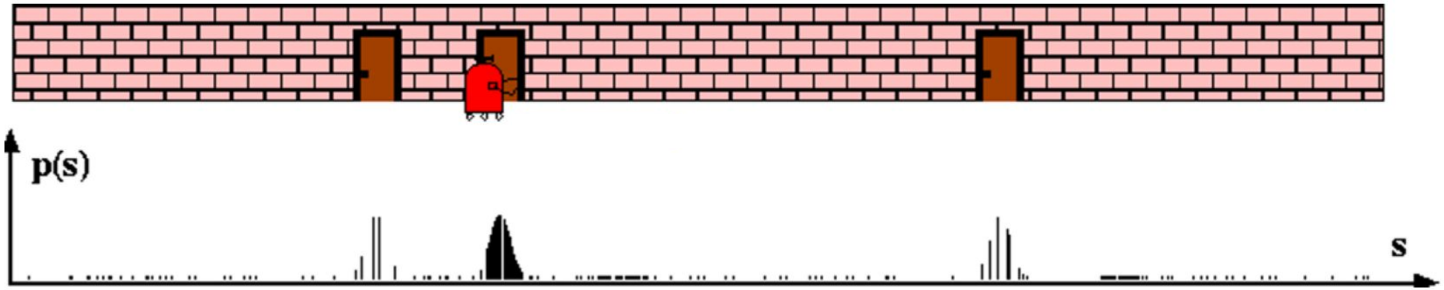
Particle Filtering

- We maintain a probability distribution over robot pose via a set of hypotheses (particles) with pose $\{\boldsymbol{\mu}^{(k)}\}_k$ and their respective weight $\{\alpha^{(k)}\}_k$:

$$P\left(\mathbf{x}_T \middle| \mathbf{z}_{1:T}, \mathbf{u}_{1:T-1}\right) = \begin{cases} \alpha^{(k)} & \mathbf{x}_T = \boldsymbol{\mu}^{(k)} \\ 0 & \text{otherwise} \end{cases}$$

- The weights $\alpha^{(k)}$ represent the probability of each particle, hence:

$$\sum_{k=1}^{N} \alpha^{(k)} = 1$$



Prediction Step

- Given a particle set $\{(\boldsymbol{\mu}_{T|T}^{(k)}, \alpha_{T|T}^{(k)})\}_k$ at time T, we are interested to obtain the predicted particle set $\{(\boldsymbol{\mu}_{T+1|T}^{(k)}, \alpha_{T+1|T}^{(k)})\}_k$ after applying the control $\mathbf{u}_T$
- We can simply plug each $\boldsymbol{\mu}_{T|T}^{(k)}$ to the motion model
- However, we also want to model the motion sensor noise
- In other words, we want our particles be diverse, since each of them represent a hypothesis over robot pose
- Therefore, we should add a small noise to our motion model when we apply the motion model to each particle

Prediction Step

- For a differential-drive motion model, each particle represents a 2-D robot pose:

$$\boldsymbol{\mu}_{T|T}^{(k)} = \left(\mu_{x,\,T|T}^{(k)},\ \mu_{y,\,T|T}^{(k)},\ \mu_{\theta,\,T|T}^{(k)}\right)$$

- Prediction step:

$$\begin{aligned}
\mu_{x,\,T+1|T}^{(k)} &= \mu_{x,\,T|T}^{(k)} + \tau\left(v_T + n_v^{(k)}\right)\cos\left(\mu_{\theta,\,T|T}^{(k)}\right)\\
\mu_{y,\,T+1|T}^{(k)} &= \mu_{y,\,T|T}^{(k)} + \tau\left(v_T + n_v^{(k)}\right)\sin\left(\mu_{\theta,\,T|T}^{(k)}\right)\\
\mu_{\theta,\,T+1|T}^{(k)} &= \mu_{\theta,\,T|T}^{(k)} + \tau\left(\omega_T + n_\omega^{(k)}\right)
\end{aligned}$$

- Note that the particle weights $\{\alpha^{(k)}\}_k$ do not change during the prediction step

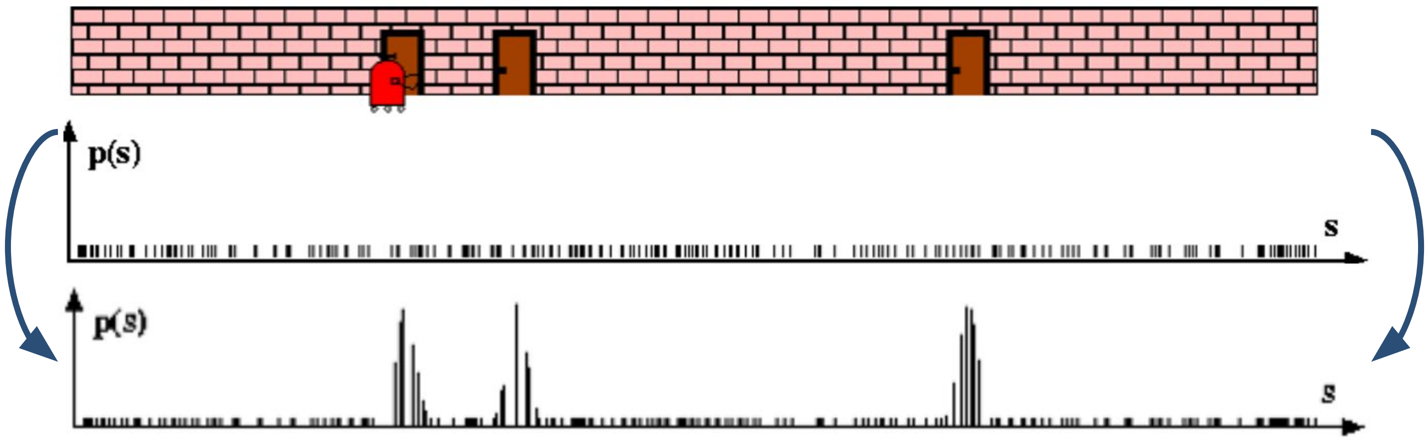
- We denote the predicted particle set as $\{(\boldsymbol{\mu}_{T+1|T}{}^{(k)}, \alpha_{T+1|T}{}^{(k)})\}_k$

Update Step

- Given a predicted particle set $\{(\boldsymbol{\mu}_{T+1|T}{}^{(k)}, \alpha_{T+1|T}{}^{(k)})\}_k$, we want to update the weight of each particle according to the sensor observation $\mathbf{z}_{T+1}$

- Here, we are interested to know if a hypothesis $\boldsymbol{\mu}_{T+1|T}{}^{(k)}$ agrees with an observation $\mathbf{z}_{T+1}$

- If it agrees (disagrees), increase (decrease) the particle's weight $\alpha_{T+1|T}{}^{(k)}$



Update Step

- In order to incorporate a new observation $\mathbf{z}_{T+1}$ to the particle set, we need to use the Bayes rule:

$$
\begin{aligned}
& P\left(\mathbf{x}_{T+1}=\boldsymbol{\mu}_{T+1|T}^{(k)} \middle| \mathbf{z}_{1:T+1}, \mathbf{u}_{1:T}\right)=\alpha_{T+1|T+1}^{(k)} \\
& \overset{\text{Bayes Rule}}{=} \frac{1}{\eta_{T+1}} P\left(\mathbf{z}_{T+1} \middle| \mathbf{x}_{T+1}=\boldsymbol{\mu}_{T+1|T}^{(k)}\right) P\left(\mathbf{x}_{T+1}=\boldsymbol{\mu}_{T+1|T}^{(k)} \middle| \mathbf{z}_{1:T}, \mathbf{u}_{1:T}\right) \\
& =\frac{1}{\eta_{T+1}} \underbrace{P\left(\mathbf{z}_{T+1} \middle| \mathbf{x}_{T+1}=\boldsymbol{\mu}_{T+1|T}^{(k)}\right)}_{\text{forward observation model}} \alpha_{T+1|T}^{(k)}
\end{aligned}
$$

- We still need expressions for the forward observation model and the denominator η_{T+1}

Laser Correlation Model

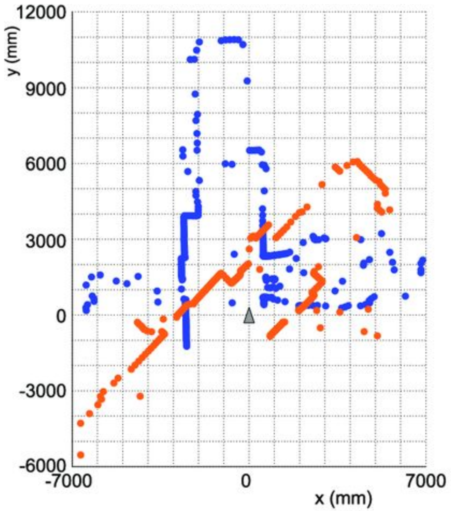
- We define the forward observation model as a function of the correlation between map $\mathbf{m}$ and LiDAR scan $\mathbf{z}$ obtained from robot pose $\mathbf{x}$:

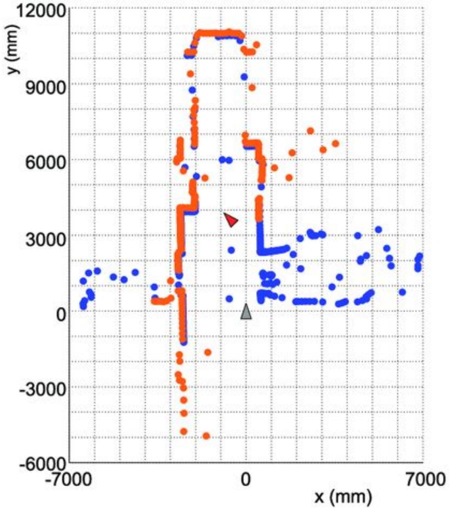
$$P(\mathbf{z}|\mathbf{x}) \propto \exp(\mathrm{corr}(\mathrm{r}(\mathbf{z}, \mathbf{x}), \mathbf{m}))$$

  where $\mathrm{r}(\mathbf{z}, \mathbf{x})$ is the LiDAR point cloud from pose $\mathbf{x}$ and:

$$\mathrm{corr}(\mathbf{r}, \mathbf{m}) = \sum_{i} \mathbb{1}\left\{\mathrm{r}_i = \mathrm{m}_i\right\}$$

- The correlation is large if $\mathrm{r}(\mathbf{z}, \mathbf{x})$ and the map $\mathbf{m}$ agree





Update Step

- Going back to the weight update:

$$\alpha^{(k)}_{T+1|T+1} = \frac{1}{\eta_{T+1}} \exp\left(\mathrm{corr}\left(\mathrm{r}\left(\mathbf{z}_{T+1}, \boldsymbol{\mu}^{(k)}_{T+1|T}\right), \mathbf{m}\right)\right) \alpha^{(k)}_{T+1|T}$$

- Furthermore, the denominator η_{T+1} is constant among all particles

- Therefore, we can eliminate it by normalization:

$$\text{for } k = 1 \ldots N:$$

$$\alpha^{(k)}_{T+1|T+1} \leftarrow \exp\left(\mathrm{corr}\left(\mathrm{r}\left(\mathbf{z}_{T+1}, \boldsymbol{\mu}^{(k)}_{T+1|T}\right), \mathbf{m}\right)\right) \alpha^{(k)}_{T+1|T}$$

$$\eta_{T+1} = \sum_{k=1}^{N} \alpha^{(k)}_{T+1|T+1}$$

$$\text{for } k = 1 \ldots N:$$

$$\alpha^{(k)}_{T+1|T+1} \leftarrow \frac{\alpha^{(k)}_{T+1|T+1}}{\eta_{T+1}}$$

Resampling

- Particle depletion: a situation in which most of the particle weights become close to zero

- This means that only a minority of the particles can be good candidates for the true robot pose while the rest are irrelevant

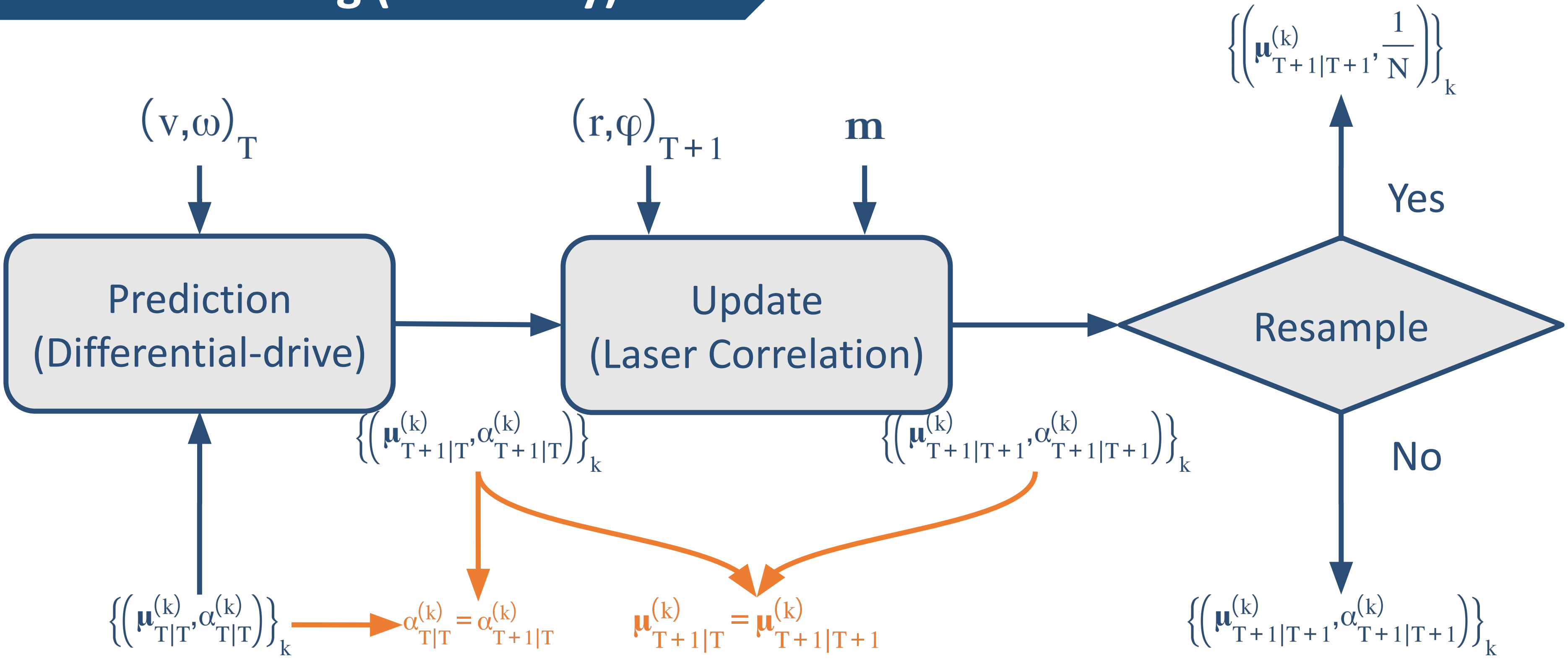
- Resampling is a procedure to avoid particle depletion

- Given a particle set $\{(\boldsymbol{\mu}_{T|T}^{(k)}, \alpha_{T|T}^{(k)})\}_k$, create a new particle set with equal weights $\alpha_{T|T}^{(k)} = 1/N$ by adding many particles to the locations with high weights and few particles to the locations with small weights

- Resampling can be triggered whenever the effective number of particles N_{eff} is less than a threshold:

$$N_{eff} = \frac{1}{\sum_k \left(\alpha_{T|T}^{(k)}\right)^2} < N_{thr}$$

Particle Filtering (Summary)



Demo

1. Visit the wiki page of the workshop
2. Navigate to the Localization and Mapping page
3. Run Demo 4

Thank you!

Scan to visit our website!

Existential Robotics Laboratory

Arash Asgharivaskasi
aasghari@ucsd.edu

CONTEXTUAL ROBOTICS INSTITUTE

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering