Autonomous Robotics

Winter 2026

Abhishek Gupta, Siddhartha Srinivasa

TAs: Carolina Higuera, Entong Su, Rishabh Jain

Class Outline

State Estimation
- Robotic System Design
- Filtering
- Localization
- SLAM

Control
- Feedback Control
- PID Control
- MPC
- LQR

Planning
- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning
- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL

Recap

So what do we need to define to instantiate this?

Key Idea: Apply Markov to get a recursive update!

Step 0. Start with the belief at time step t-1

$$bel(x_{t-1})$$

Step 1: Prediction - push belief through dynamics given action

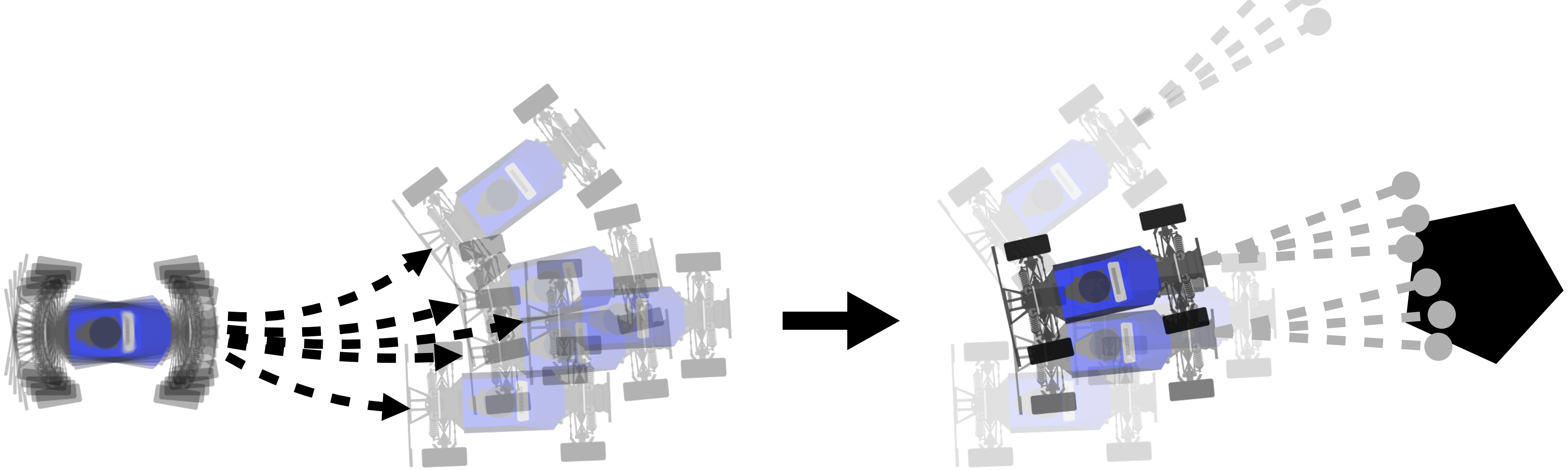
$$\overline{bel}(x_t) = \sum P(x_t | u_t, x_{t-1}) bel(x_{t-1})$$

Step 2: Correction - apply Bayes rule given measurement

$$bel(x_t) = \eta P(z_t | x_t) \overline{bel}(x_t)$$

Let's ground this in the context of the car

PREDICTION **CORRECTION**



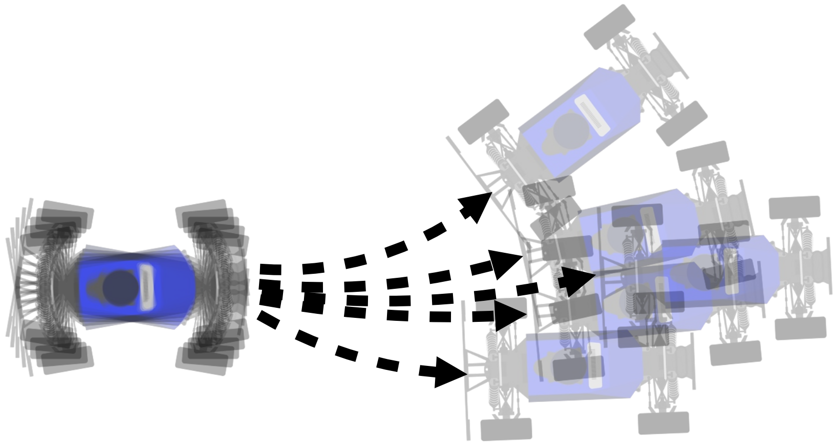
PREDICTION

$$P(x_t|u_t, x_{t-1})$$

CORRECTION

$$P(z_t|x_t)$$

Motion Model



How do we know this?
→ it's just physics!

$$P(x_t | u_t, x_{t-1})$$

Kinematic Car Model

$$\dot{x} = f(x, u)$$

$$x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

- x — X-COORDINATE
- y — Y-COORDINATE
- θ — HEADING

$$u = \begin{bmatrix} v \\ \delta \end{bmatrix}$$

- v — SPEED
- δ — STEERING ANGLE

Kinematic Car Model

$$\dot{x} = f(x, u)$$

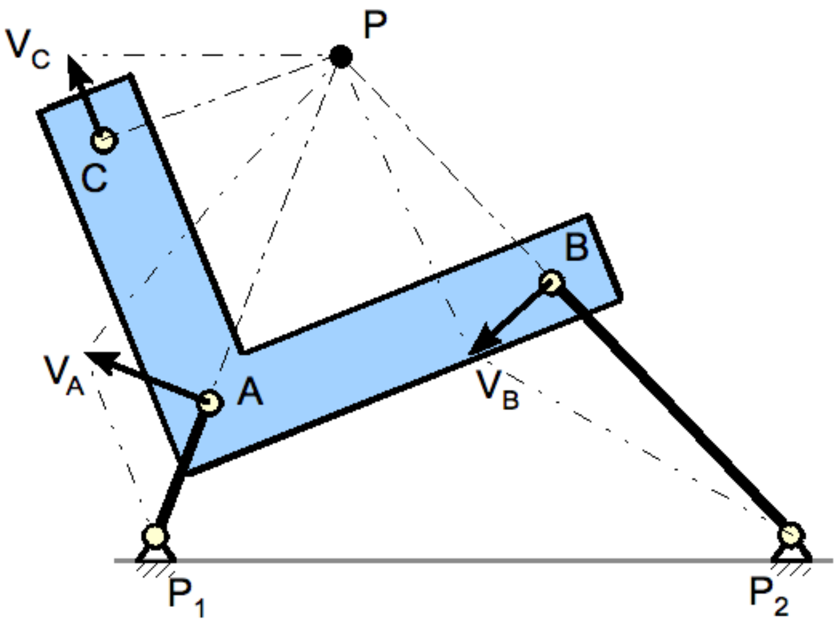
→ INTEGRATE

$$\begin{bmatrix} x_{t-1} + \Delta x \\ y_{t-1} + \Delta y \\ \theta_{t-1} + \Delta \theta \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$$

→ ADD NOISE

$$P(x_t | u_t, x_{t-1})$$

Definition: Instant Center of Rotation (CoR)



A planar **rigid body** undergoing a **rigid transformation** can be viewed as undergoing a **pure rotation** about an instant center of rotation.

rigid body: a non-deformable object

rigid transformation: a combined rotation and translation

HTTPS://EN.WIKIPEDIA.ORG/WIKI/INSTANT_CENTRE_OF_ROTATION

Lecture Outline

Recap

↓

Motion Models

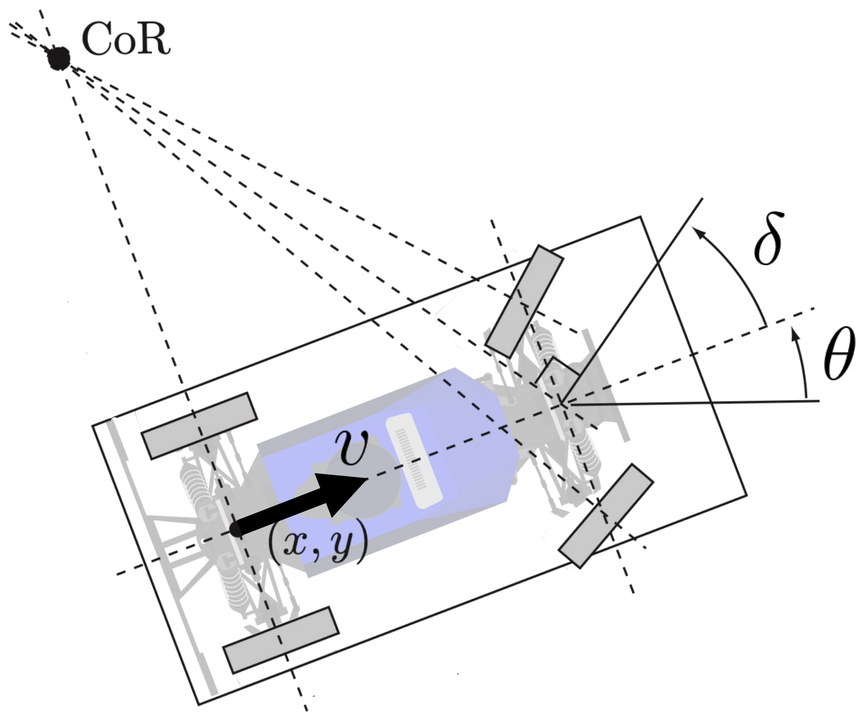
↓

Observation Models

↓

Particle Filters

Equations of Motion

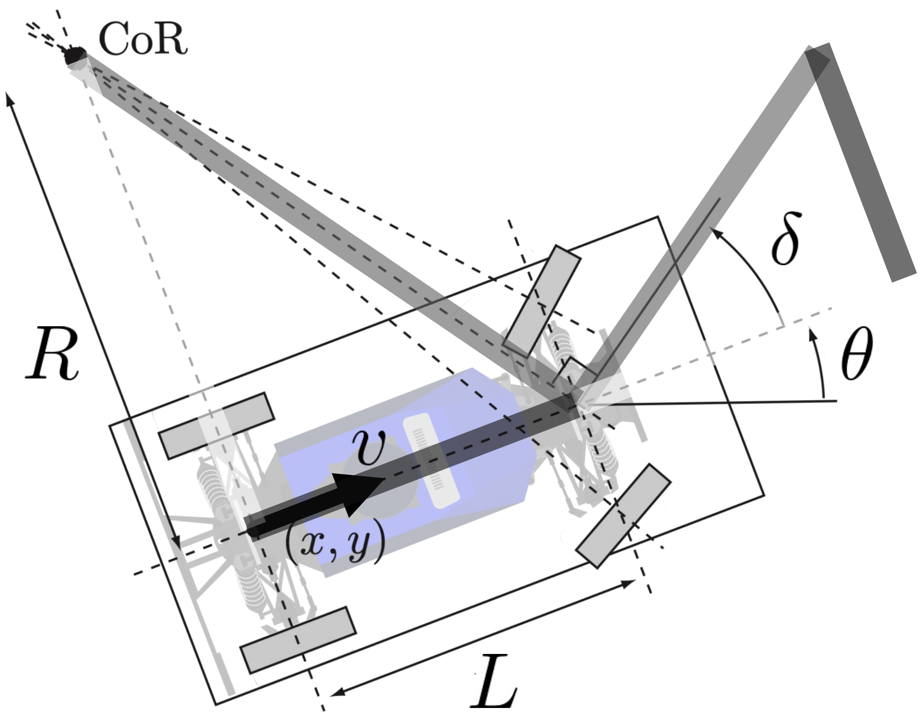


$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \textbf{?}$$

Equations of Motion



$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

$$\tan\delta = \frac{L}{R} \rightarrow R = \frac{L}{\tan\delta}$$

Kinematic Car Model

$$\dot{x} = f(x, u) \quad \xrightarrow{\text{INTEGRATE}} \quad \begin{bmatrix} x_{t-1} + \Delta x \\ y_{t-1} + \Delta y \\ \theta_{t-1} + \Delta\theta \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$$

Integrate the Kinematics Numerically

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \frac{v}{L}\tan\delta$$

Assume that steering angle is **piecewise constant** between t and t’

Integrate the Kinematics Numerically

$$\Delta x = \int_t^{t'} v \cos\theta(t) dt = \int_t^{t'} \frac{v\cos\theta}{\dot\theta}\frac{d\theta}{dt} dt = \frac{v}{\dot\theta}\int_\theta^{\theta'} \cos\theta d\theta$$

$$= \frac{L}{\tan\delta}(\sin\theta' - \sin\theta)$$

$$\Delta y = \frac{L}{\tan\delta}(\cos\theta - \cos\theta')$$

$$\Delta\theta = \int_t^{t'} \dot\theta dt = \frac{v}{L}\tan\delta\Delta t$$

$$\dot x = v\cos\theta$$
$$\dot y = v\sin\theta$$
$$\dot\theta = \frac{v}{L}\tan\delta$$

Assume that steering angle is **piecewise constant** between t and t’

Kinematic Car Update

$$\theta_t = \theta_{t-1} + \Delta\theta = \theta_{t-1} + \frac{v}{L} \tan\delta \Delta t$$

$$x_t = x_{t-1} + \Delta x = x_{t-1} + \frac{L}{\tan\delta}(\sin\theta_t - \sin\theta_{t-1})$$

$$y_t = y_{t-1} + \Delta y = y_{t-1} + \frac{L}{\tan\delta}(\cos\theta_{t-1} - \cos\theta_t)$$

Kinematic Car Model

$$\dot{x} = f(x, u)$$

→ INTEGRATE

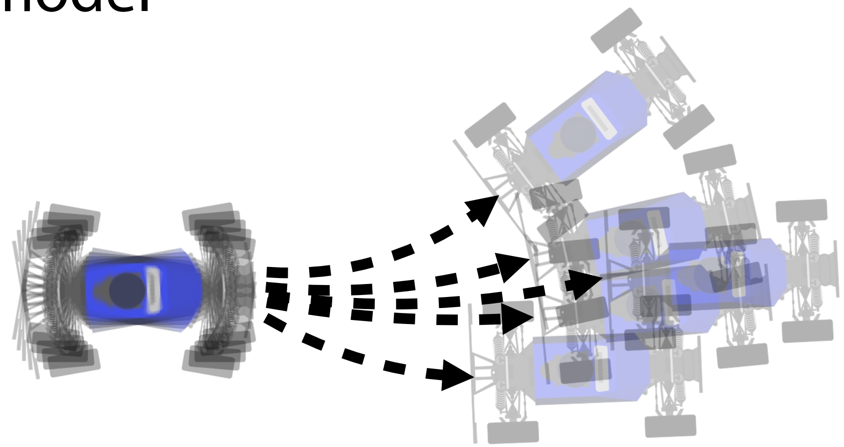
$$\begin{bmatrix} x_{t-1} + \Delta x \\ y_{t-1} + \Delta y \\ \theta_{t-1} + \Delta \theta \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$$

→ ADD NOISE

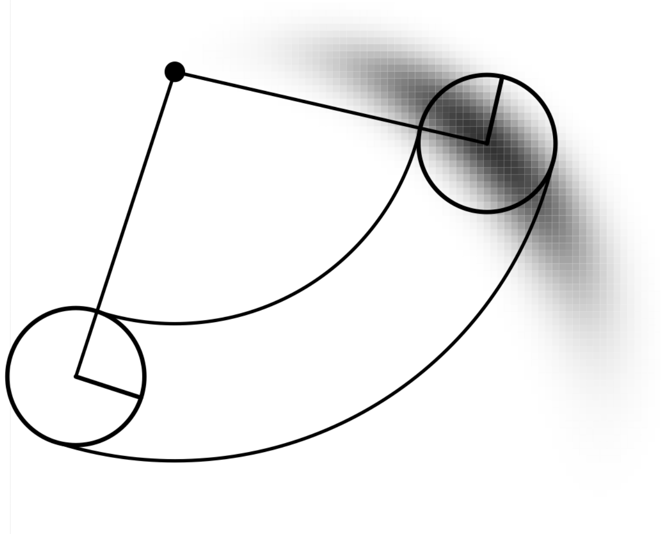
$$P(x_t | u_t, x_{t-1})$$

Why is the kinematic car model probabilistic?

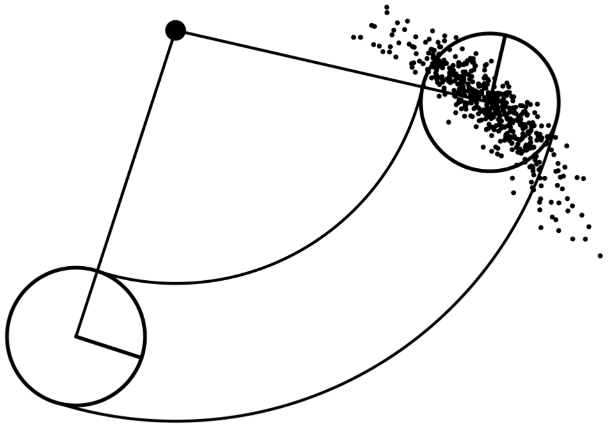
- Control signal error: voltage discretization, communication lag
- Unmodeled physics parameters: friction of carpet, tire pressure
- Incorrect physics: ignoring tire deformation, ignoring wheel slippage
- Our probabilistic motion model
  - Add noise to control before propagating through model
  - Add noise to state after propagating through model



Motion Model Summary



MOTION MODEL
PROB. DENSITY FUNCTION



MOTION MODEL
SAMPLES

- Write down the deterministic equations of motion (kinematic car model)
- Introduce stochasticity to account against various factors

Lecture Outline

Recap

↓

Motion Models

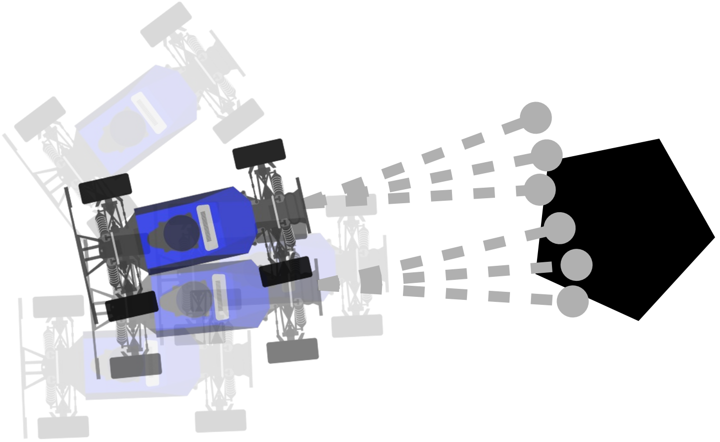
↓

Observation Models

↓

Particle Filters

Sensor Model

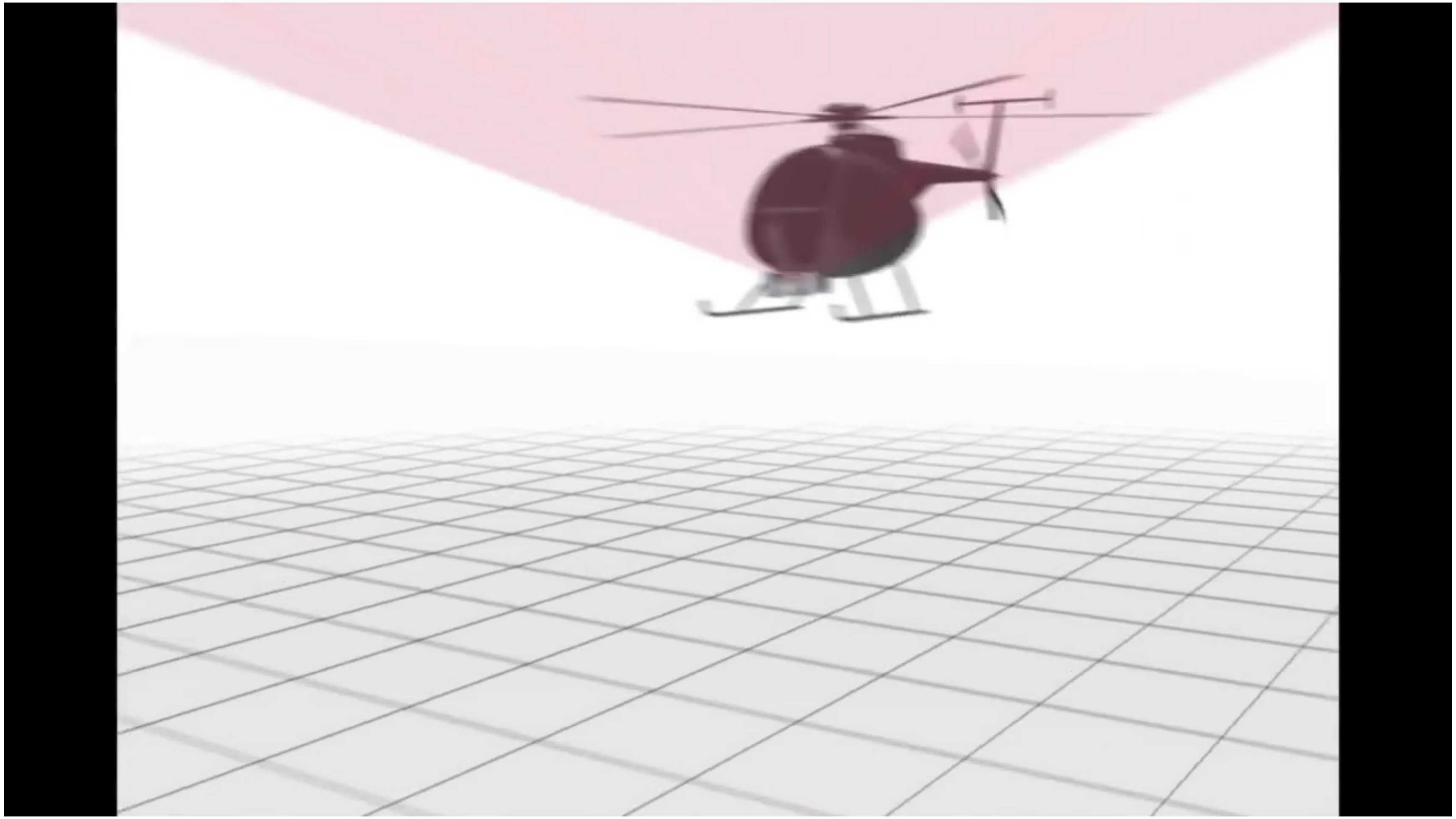


$$P(z_t|x_t)$$

How Does LIDAR Work?

HTTPS://YOUTU.BE/NZKVF1CXE8S

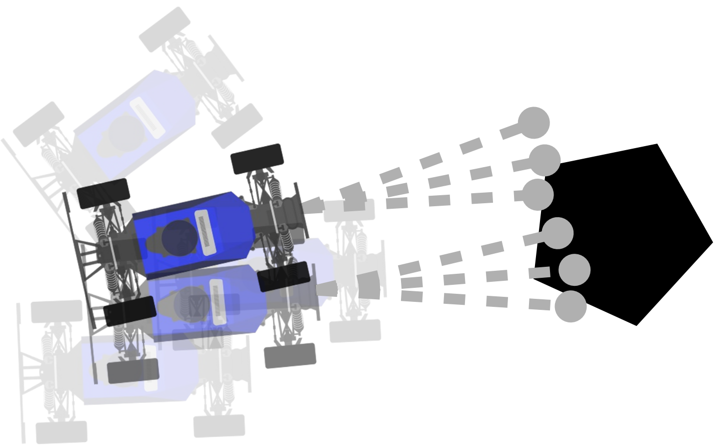
LIDAR in the Real World



HTTPS://YOUTU.BE/I8YV5D8CPOC

Why is the sensor model probabilistic?

- Incomplete/incorrect map: pedestrians, objects moving around
- Unmodeled physics: lasers go through glass
- Sensing assumptions: light interference from other sensors, multiple laser returns (bouncing off multiple objects)



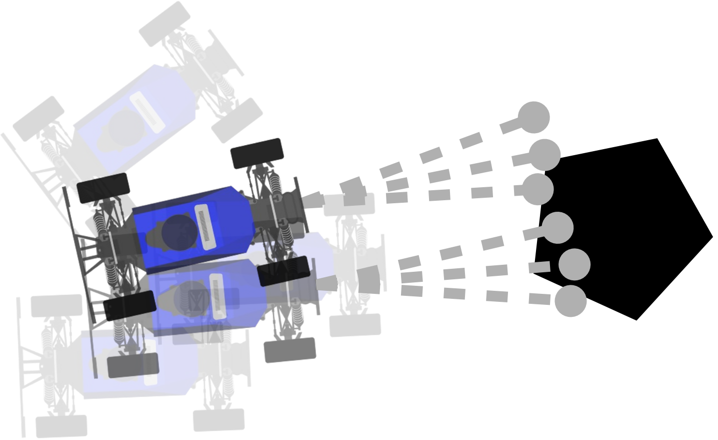
What defines a good sensor model?

- Overconfidence can be catastrophic for Bayes filter
- LIDAR is very precise, but has distinct modes of failure
  - Anticipate specific types of failures, and add stochasticity accordingly

What sensor model should I use for MuSHR?

$$P(z_t|x_t) \rightarrow P(z_t|x_t, m)$$

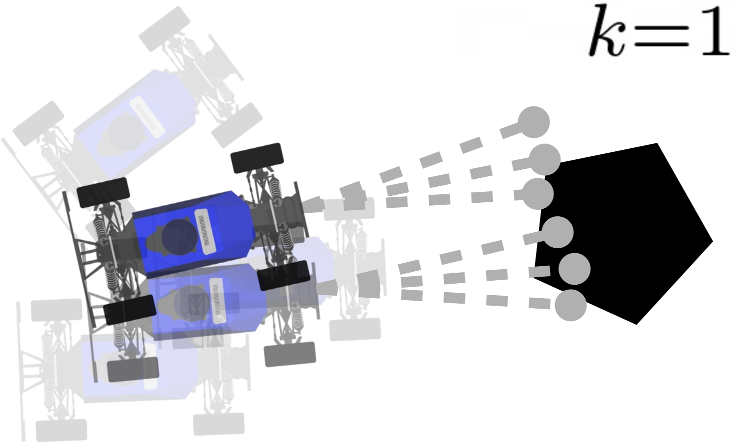
LASER SCAN STATE MAP



Assumption: Conditional Independence

$$P(z_t|x_t, m) = P(z_t^1, z_t^2, \cdots, z_t^K|x_t, m)$$

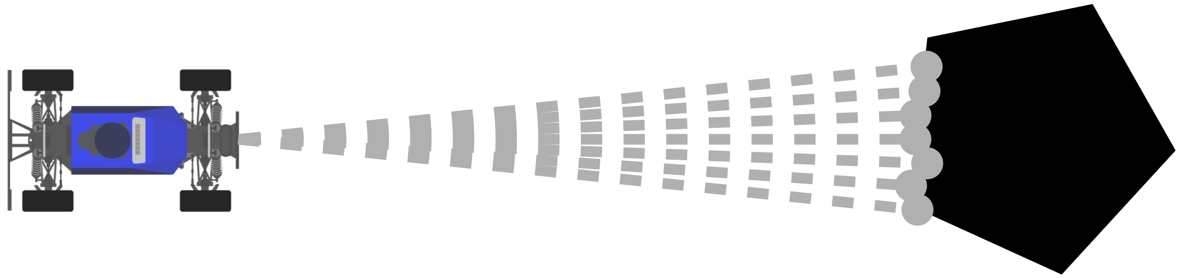
$$= \prod_{k=1}^{K} P(z_t^k|x_t, m)$$



Assumption: Conditional Independence

$$P(z_t|x_t, m) = P(z_t^1, z_t^2, \cdots, z_t^K|x_t, m)$$

$$= \prod_{k=1}^{K} P(z_t^k|x_t, m)$$



Single Beam Sensor Model

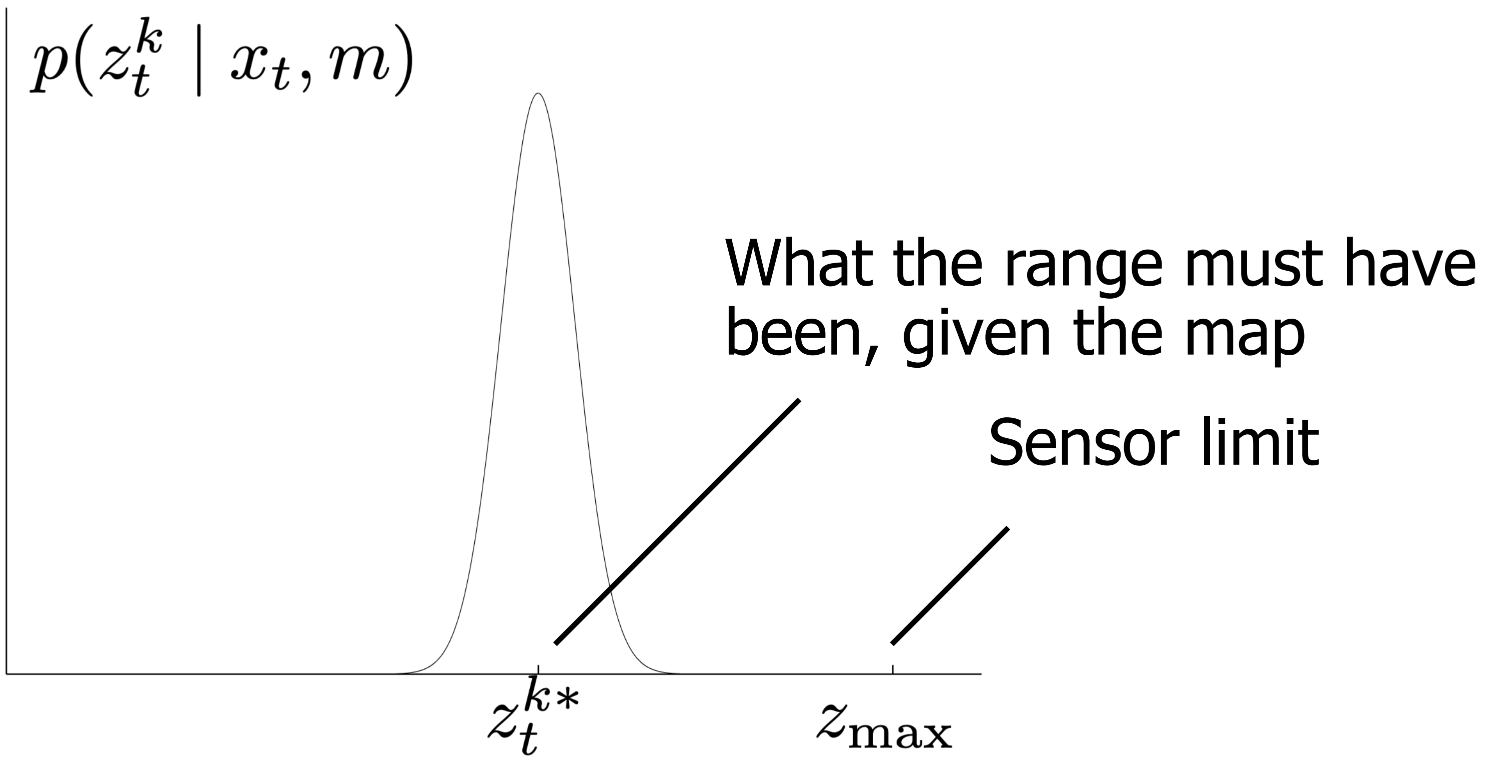
$$P(z_t^k | x_t, m)$$

DISTANCE

Typical Sources of Stochasticity

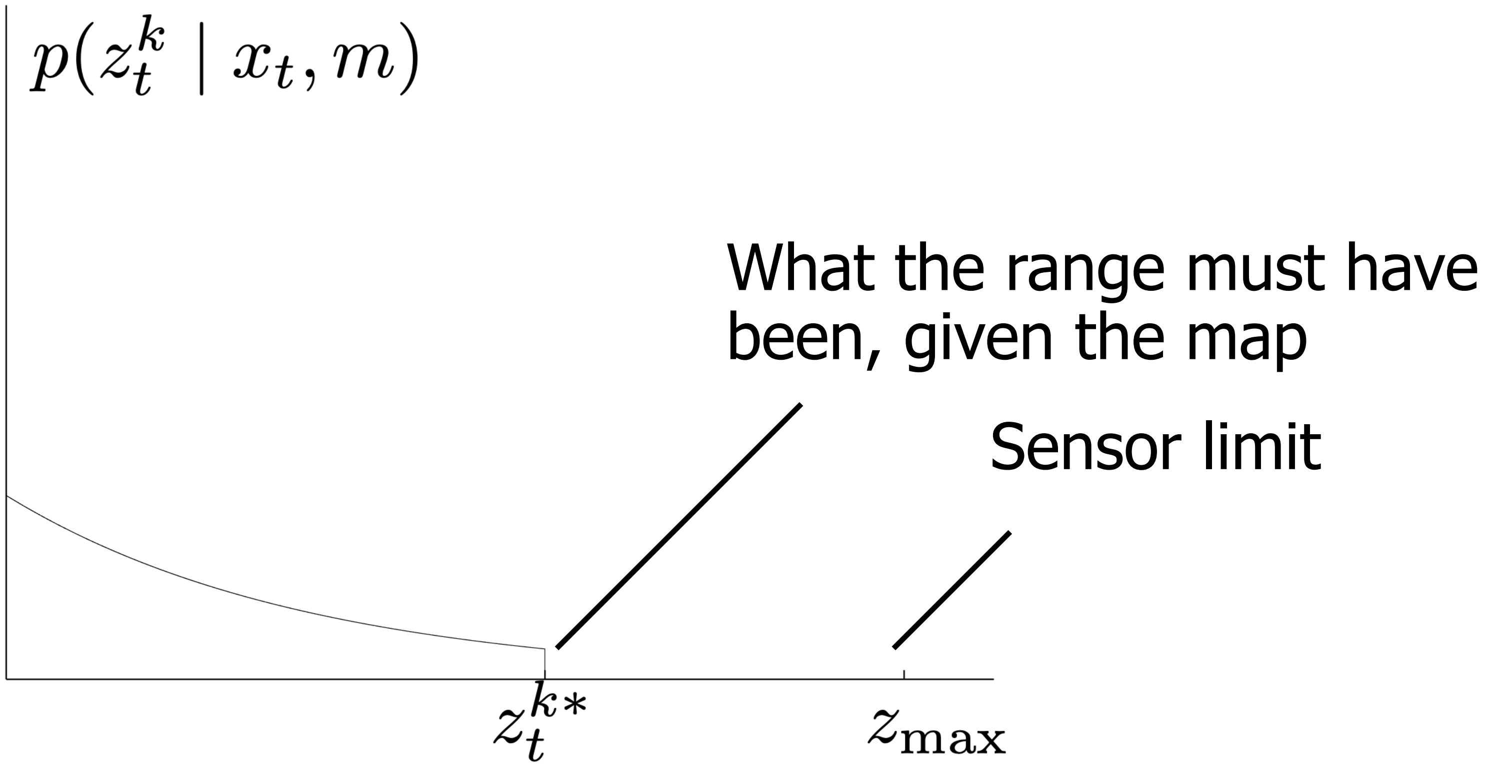
1. Correct range (distance) with local measurement noise
2. Unexpected objects
3. Sensor failures
4. Random measurements

Factor 1: Local Measurement Noise



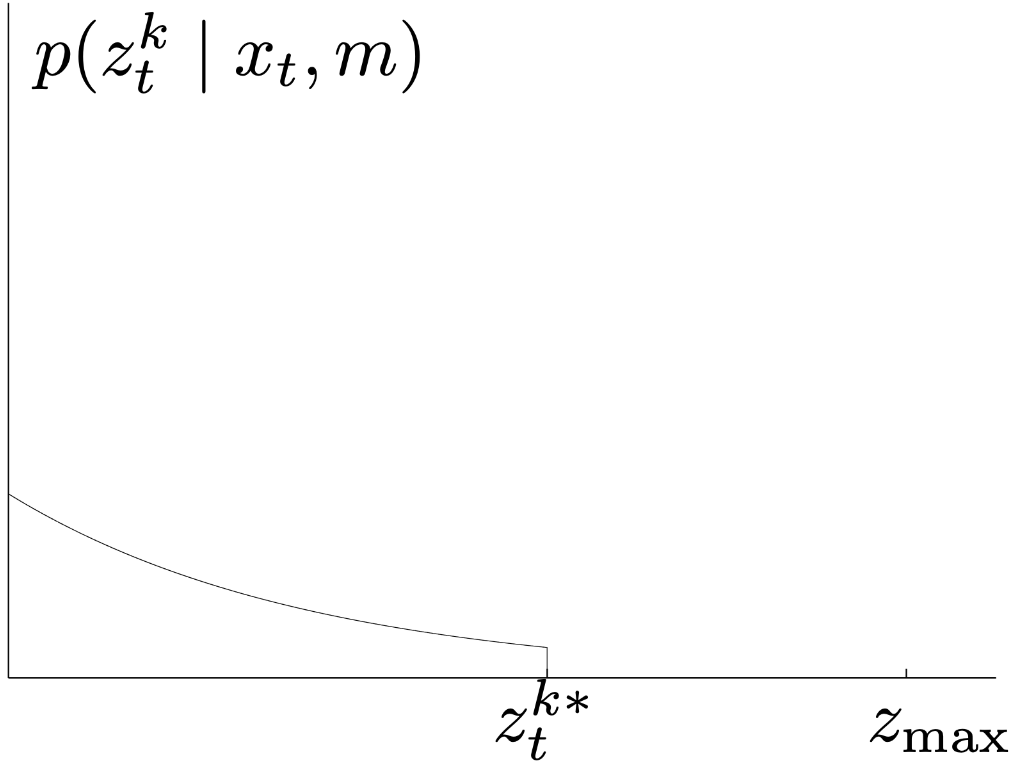
$$p_{\text{hit}}(z_t^k \mid x_t, m) \quad = \quad \begin{cases} \eta\, \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \le z_t^k \le z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Factor 2: Unexpected Objects



$$p_{\text{short}}(z_t^k \mid x_t, m) = \begin{cases} \eta \, \lambda_{\text{short}} \, e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases}$$

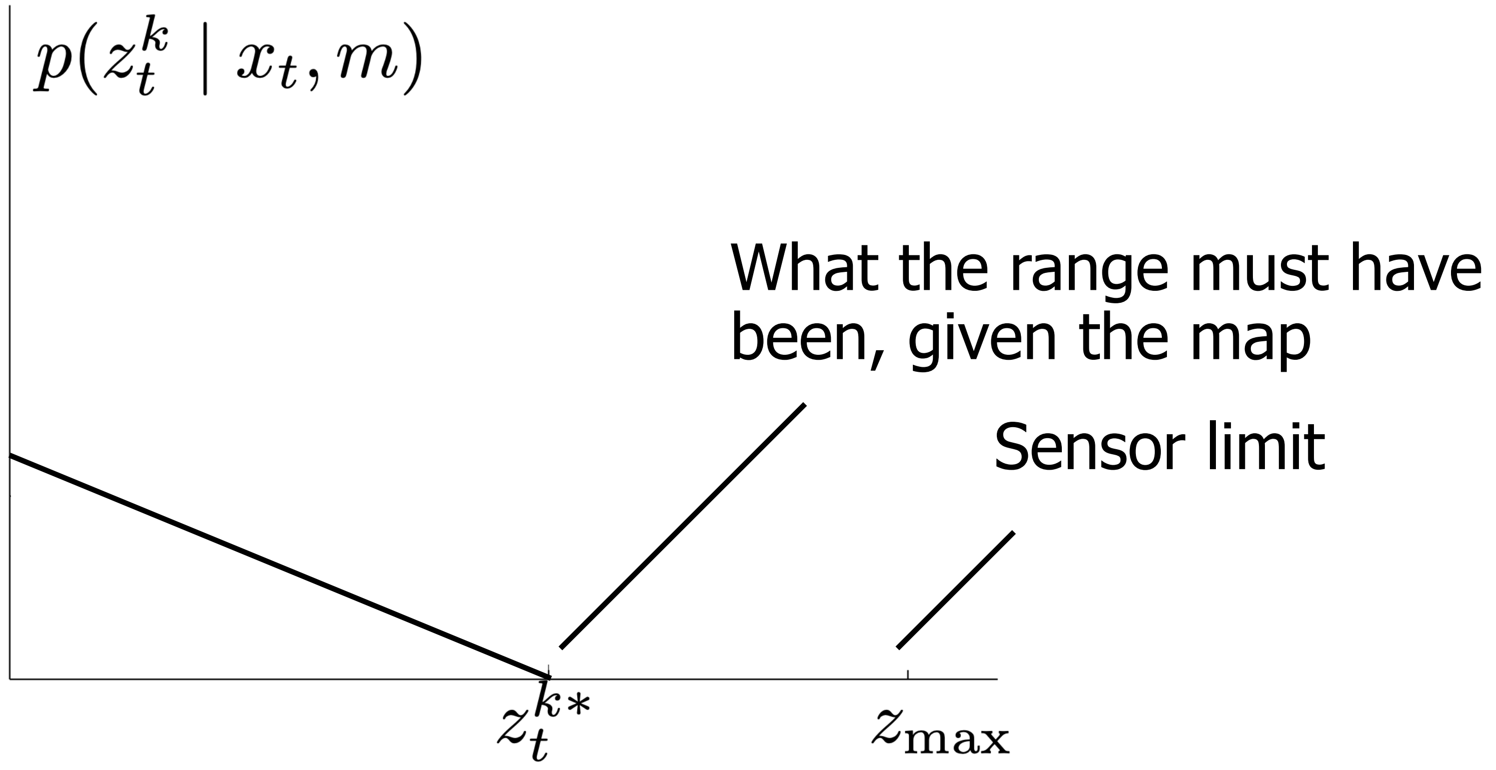
Factor 2: Unexpected Objects



1									128
0	1								64
0	0	1							32
0	0	0	1						16
0	0	0	0	1					8
0	0	0	0	0	1				4
0	0	0	0	0	0	1			2
0	0	0	0	0	0	0	1		1

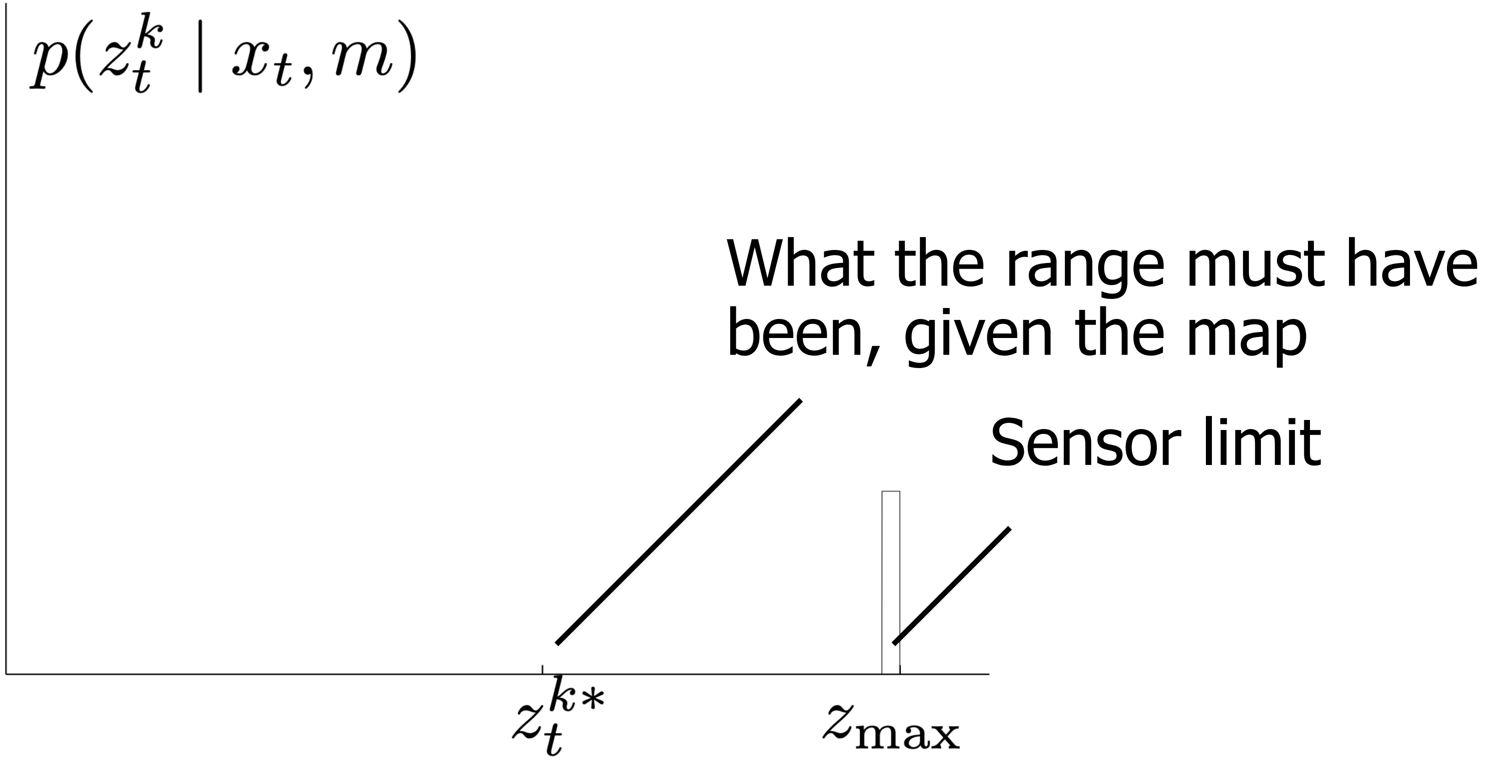
$$p_{\text{short}}(z_t^k \mid x_t, m) = \begin{cases} \eta\, \lambda_{\text{short}}\, e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases}$$

Factor 2: Unexpected Objects (Project)



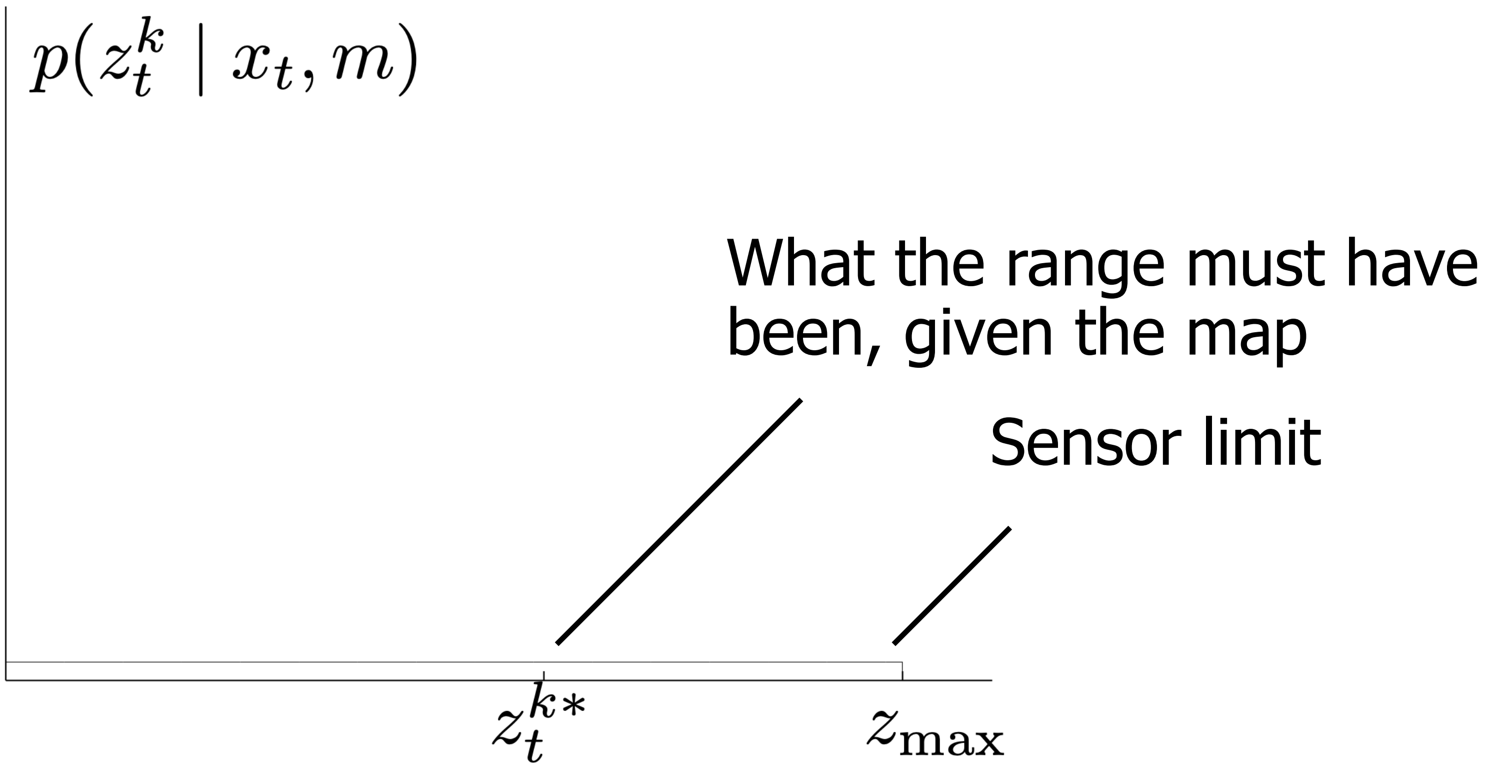
$$p_{\text{short}}(z_t^k | x_t, m) = \begin{cases} 2\frac{z_t^{k^*} - z_t^k}{z_t^{k^*}} & \text{if } z_t^k < z_t^{k^*} \\ 0 & \text{otherwise} \end{cases}$$

Factor 3: Sensor Failures



$$p_{\max}(z_t^k \mid x_t, m) = I(z = z_{\max}) = \begin{cases} 1 & \text{if } z = z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Factor 4: Random Measurements

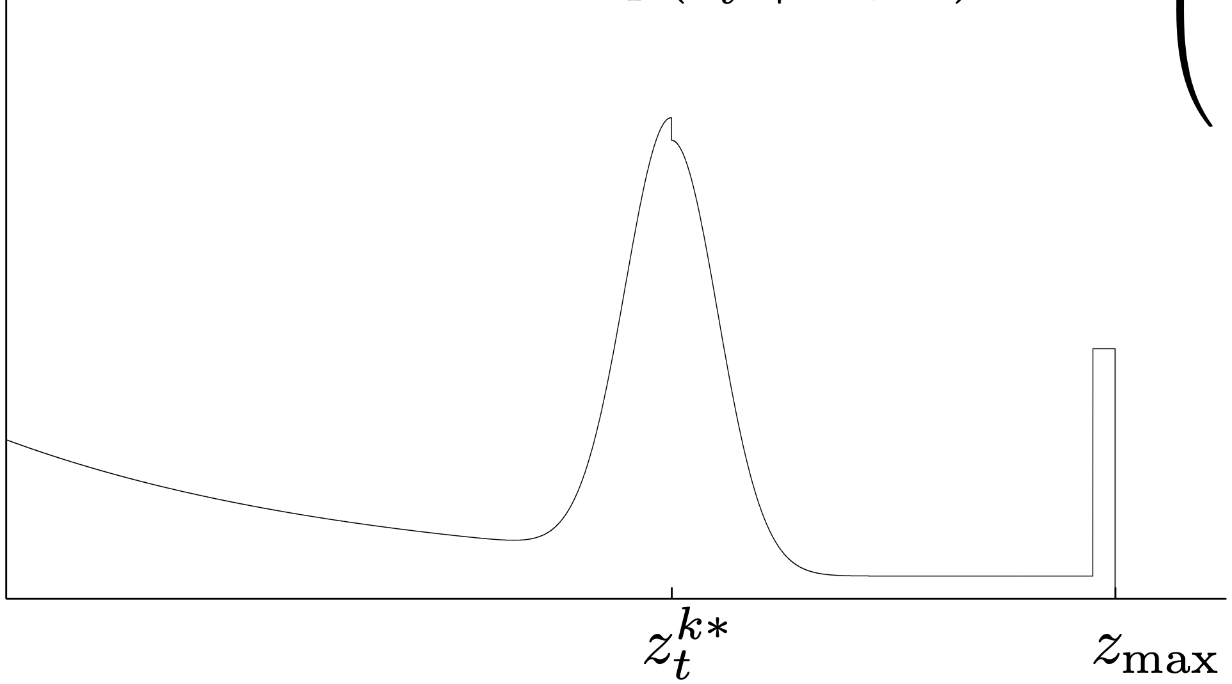


$$p_{\text{rand}}(z_t^k \mid x_t, m) = \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z_t^k < z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Putting It All Together

$$p(z_t^k \mid x_t, m) = \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\text{max}} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k \mid x_t, m) \\ p_{\text{short}}(z_t^k \mid x_t, m) \\ p_{\text{max}}(z_t^k \mid x_t, m) \\ p_{\text{rand}}(z_t^k \mid x_t, m) \end{pmatrix}$$

Weights sum to 1



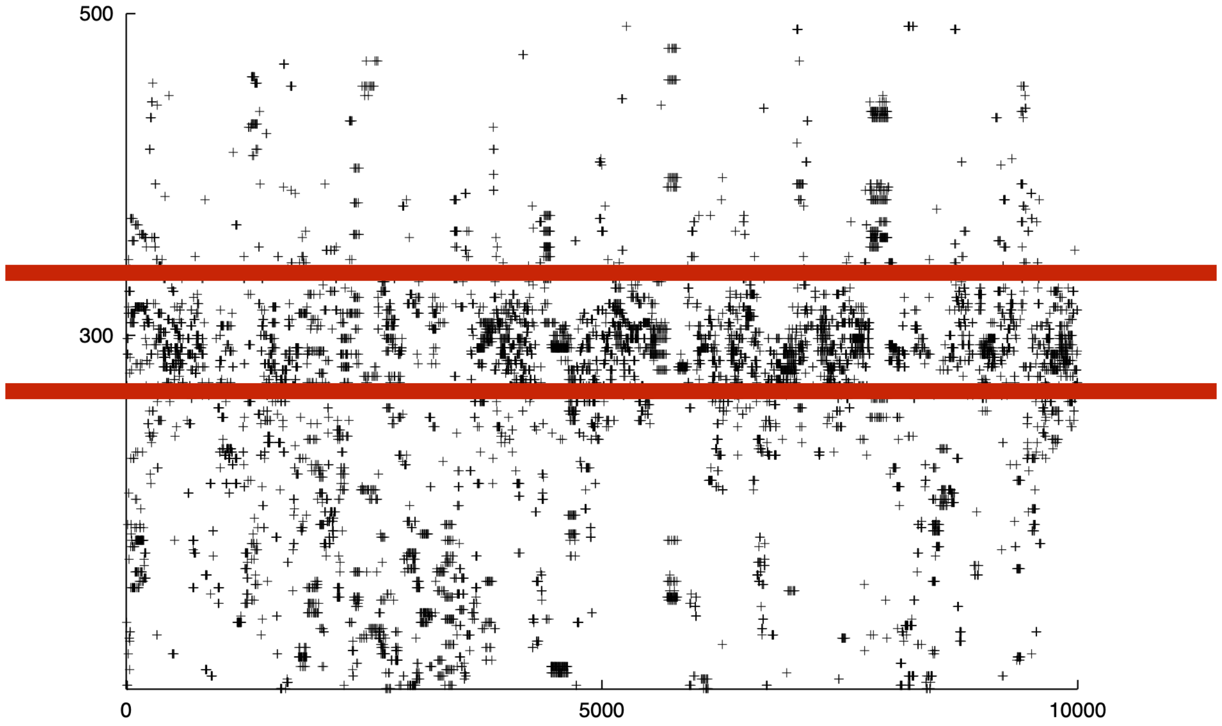
LIDAR Model Algorithm

$$P(z_t|x_t, m) = \prod_{k=1}^{K} P(z_t^k|x_t, m)$$

1. Use robot **state** to compute the sensor's pose on the **map**
2. Ray-cast from the sensor to compute a simulated laser scan
3. For each beam, compare ray-casted distance to **real laser scan distance**
4. Multiply all probabilities to compute the likelihood of that real laser scan

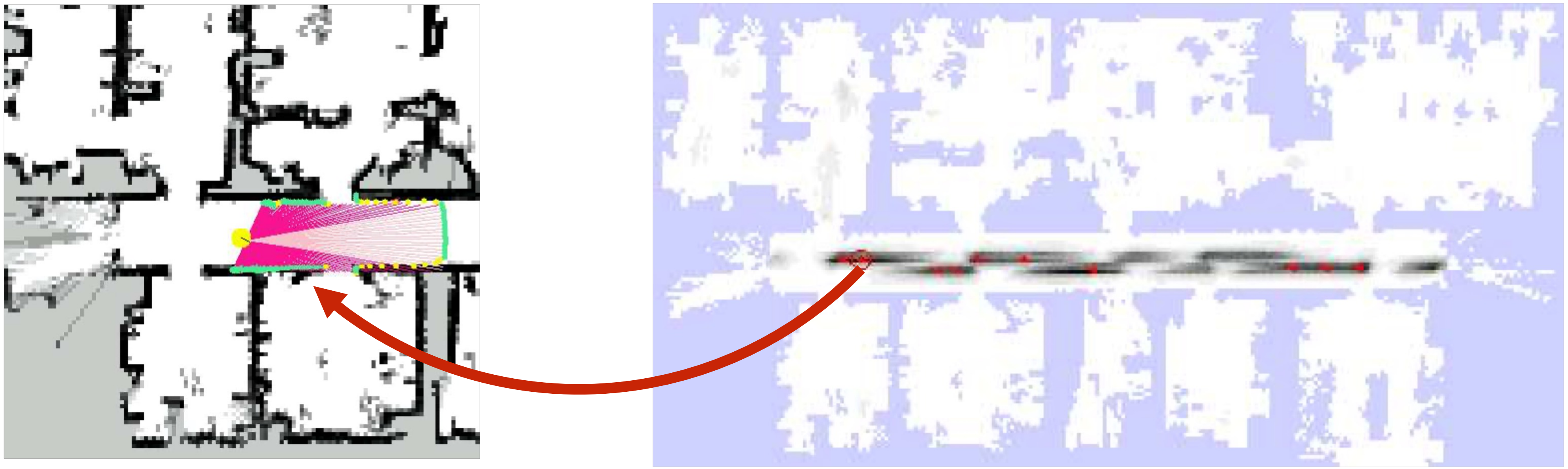
Tuning Single Beam Parameters

- Offline: collect lots of data and optimize parameters



Tuning Single Beam Parameters

- Online: simulate a scan and plot the likelihood from different positions



Actual scan

Likelihood at various locations

Dealing with Overconfidence

$$P(z_t|x_t, m) = \prod_{k=1}^{K} P(z_t^k|x_t, m)$$

- Subsample laser scans: convert 180 beams to 18 beams
- Force the single beam model to be less confident

$$P(z_t^k|x_t, m) \rightarrow P(z_t^k|x_t, m)^\alpha, \alpha < 1$$

MuSHR Localization Project

- Implement kinematic car motion model
- Implement different factors of single-beam sensor model
- Combine motion and sensor model with the Particle Filter algorithm

Lecture Outline

Recap

↓

Motion Models

↓

Observation Models

↓

Particle Filtering

Why is the Bayes filter challenging to implement?

Key Idea: Apply Markov to get a recursive update!

Step 0. Start with the belief at time step t-1

$$bel(x_{t-1})$$

Step 1: Prediction - push belief through dynamics given action

$$\overline{bel}(x_t) = \sum P(x_t|u_t, x_{t-1})bel(x_{t-1})$$

Step 2: Correction - apply Bayes rule given measurement

$$bel(x_t) = \eta P(z_t|x_t)\overline{bel}(x_t)$$

Intractable due to discretization

How does discretization work for Bayesian filters?

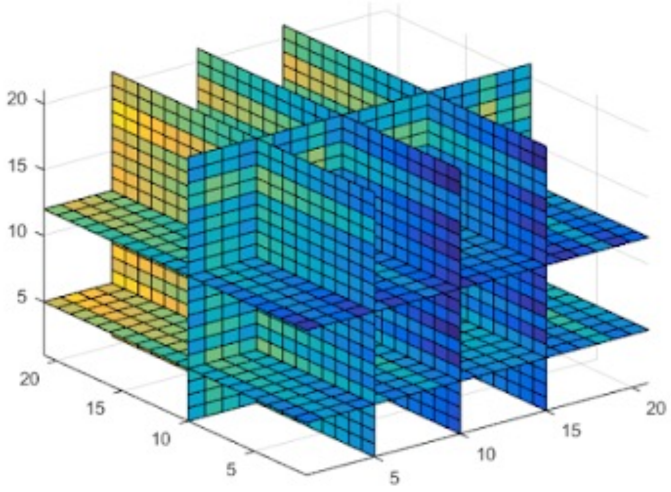
$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

X-COORDINATE - Discretize into K bins

Y-COORDINATE - Discretize into K bins

HEADING - Discretize into K bins

Overall K^3 bins

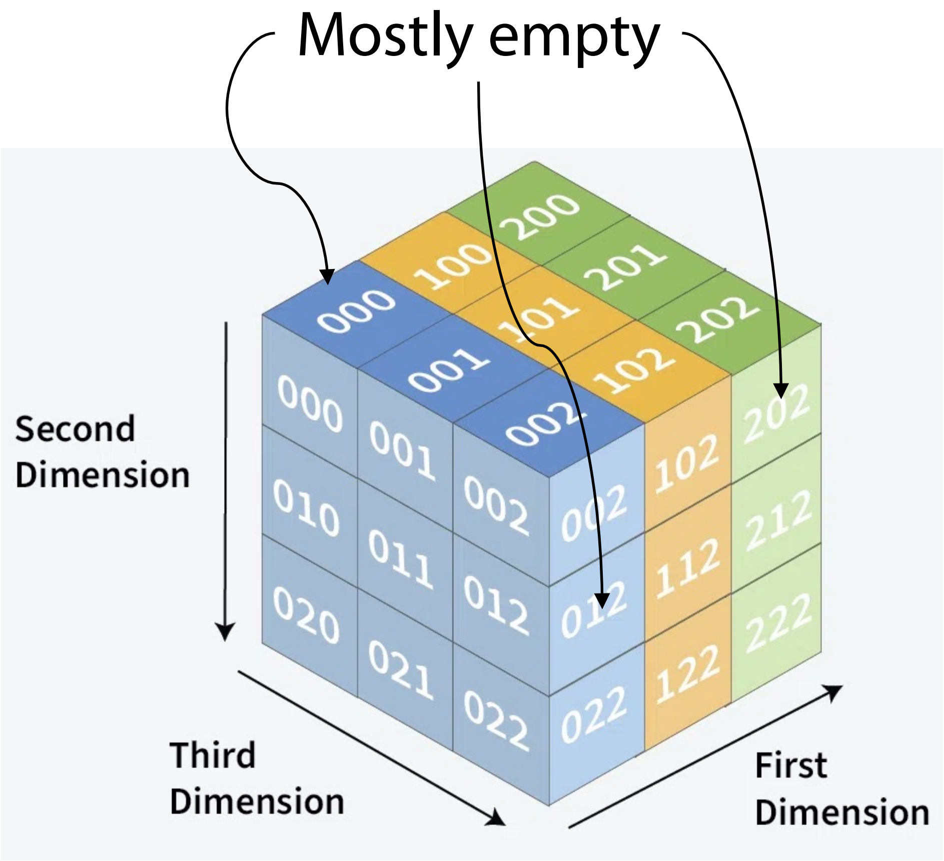


Exponentially expensive with dimension for each summation

Many of these bins will be empty!

How can we do better?

Let's change our way of thinking



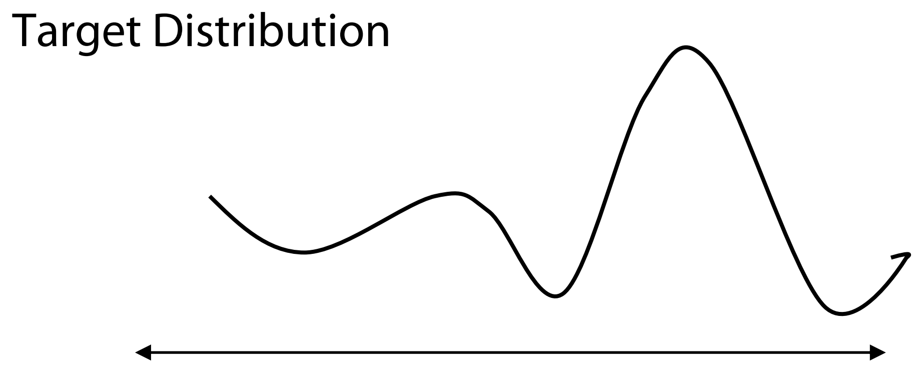
$[s_1, s_1, s_2, s_{10}, s_{40}, s_{40}, s_{40}, s_{55}, s_{55}]$

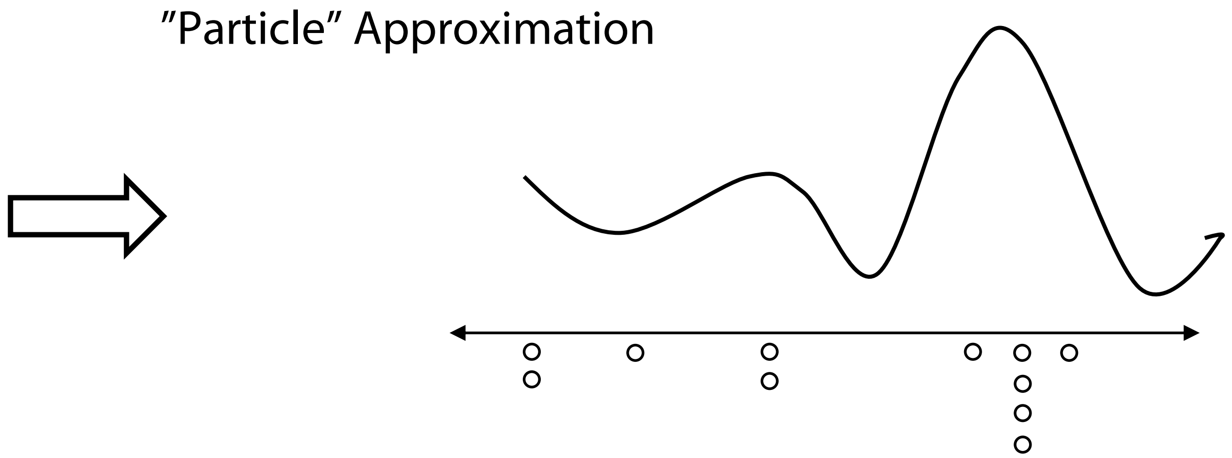
Keep a list of only the states with likelihood, with number of repeat instances proportional to probability

No discretization per dimension!

Is this even a useful/valid representation of belief?

Let's change our way of thinking





Is this even a useful/valid representation of belief?

⇩

Depends what we want to do with the probability distribution!

→ Typically we want to compute averages (expectations)

Downstream Usage of Estimated Probability Distributions

What do we actually intend to do with the belief $bel(x_{t+1})$?

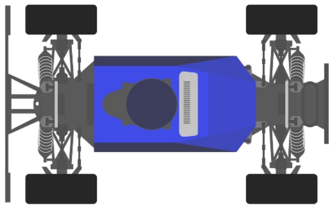
→ Often times we will be evaluating the expected value

$$\mathbb{E}[f] = \int_x f(x) bel(x) dx$$

Mean position: $f(x) \equiv x$

Probability of collision: $f(x) \equiv \mathbb{I}(x \in \mathcal{O})$

Mean value / cost-to-go: $f(x) \equiv V(x)$



Computing Expectations without Closed Form Likelihoods

Monte-Carlo Simulation

$$\mathbb{E}_{x \sim Bel(x_t)}[f(x)] = \int_x f(x) Bel(x) dx \approx \sum_x f(x) Bel(x)$$

Sample from the belief: $x_1, \cdots, x_N \sim Bel(x_t)$

$$\mathbb{E}_{x \sim Bel(x_t)}[f(x)] \approx \frac{1}{N} \sum_i^N f(x^{(i)})$$

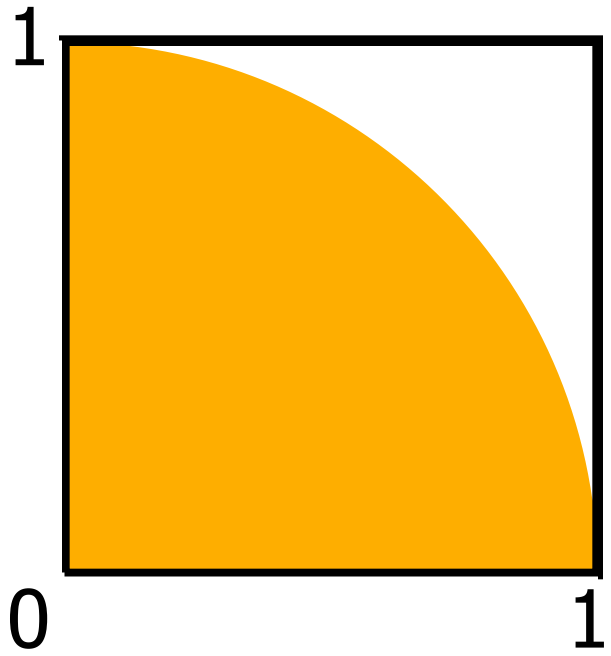
Don't require closed form distributions (Gaussian/Beta, etc), just samples (particles)!

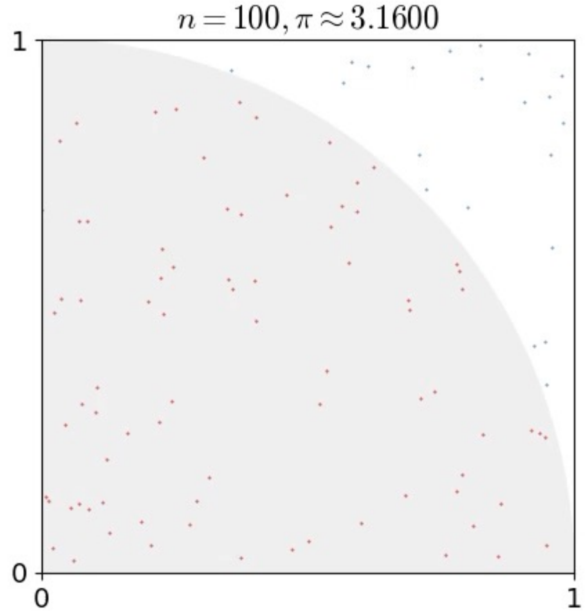
→ Replace fancy math by brute force simulation!!

Examples of Monte Carlo Estimation

$$\mathbb{E}[\mathbb{I}(x \in \mathcal{O})] = P(x \in \mathcal{O}) = \frac{\pi}{4} \approx \frac{1}{N} \sum \mathbb{I}(x^{(i)} \in \mathcal{O})$$

1. Sample points uniformly from unit square
2. Count number in quarter-circle (i.e.$\|x_i\| \leq 1$)
3. Divide by N, multiply by 4





→ Exercise: What are other practical problems where this is useful?

ADAPTED FROM WIKIPEDIA

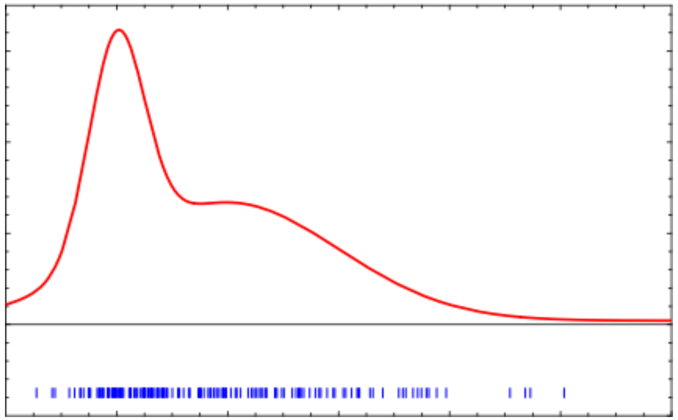
Bringing this Back to Estimation – Belief Distribution

Let's consider the Bayesian filtering update

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

Represent the belief with a set of particles! Each is a hypothesis of what the state might be.

Higher likelihood regions have more particles



How do we “propagate” belief across timesteps with particles?

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

Dynamics Update
$$\overline{Bel}(x_t) = \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Measurement Correction
$$Bel(x_t) = \eta P(z_t|x_t) \overline{Bel}(x_t)$$

How do we sample from the product of two distributions?

How do we compute conditioning/normalization with particles?

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL