Autonomous Robotics
Winter 2024

Abhishek Gupta
TAs: Karthikeya Vemuri, Arnav Thareja
Marius Memmel, Yunchu Zhang

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

Bayes filter in a nutshell

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

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

Lecture Outline

Instantiating Motion Models

↓

Instantiating Sensor Models

↓

Putting together for the Car

↓

Kalman Filtering

Bayes filter in a nutshell

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

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

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

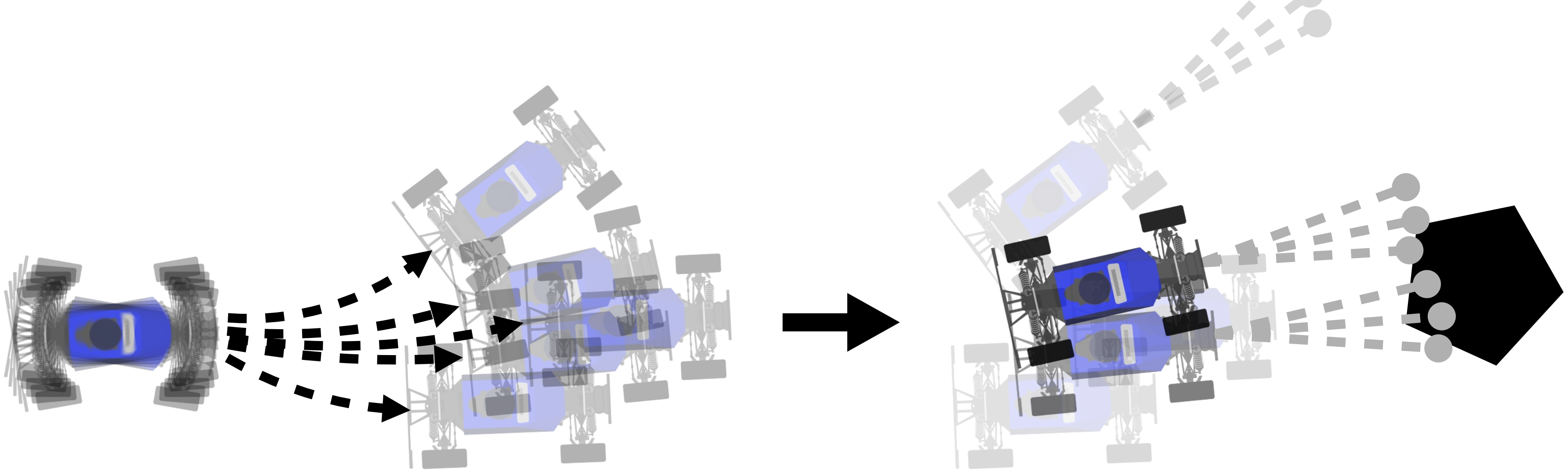
$$\overline{bel}(x_t) = \sum \boxed{P(x_t|u_t, x_{t-1})} bel(x_{t-1})$$

Step 2: Correction - apply Bayes rule given measurement

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

Let’s ground this in the context of the car

PREDICTION **CORRECTION**

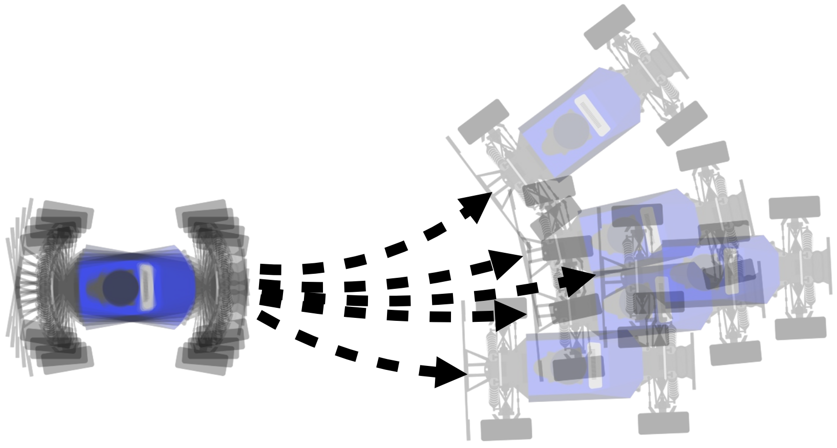


PREDICTION **CORRECTION**

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

$$P(z_t|x_t)$$

Motion Model

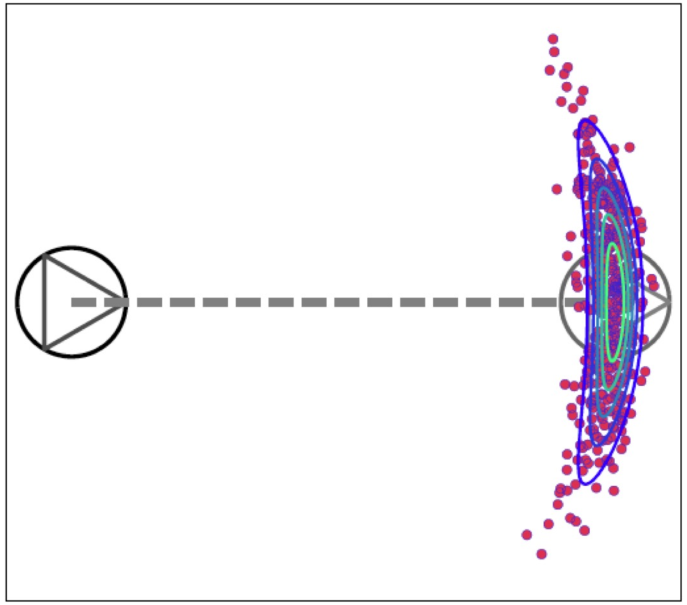


How do we know this?
→ it's just physics!

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

A Spectrum of Motion Models

VS



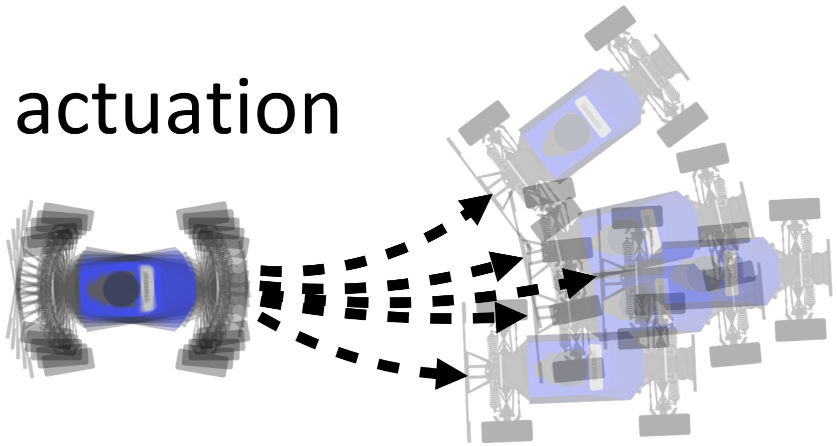
Highest-fidelity models capturing everything we know

(Red Bull F1 Simulator)

Simple model with lots of noise

Why is the motion model probabilistic?

- If we know how to write out equations of motion, shouldn't we be able to predict exactly where an object ends up?
- "All models are wrong, but some are useful" — George Box
  - Examples: ideal gas law, Coulomb friction
- Stochasticity is a catch-all for model error, actuation error, …



What defines a good motion model?

- In theory: try to accurately model the uncertainty (e.g., actuation errors)
- In practice…
  - We need just enough stochasticity to **explain any measurements** we'll see (Bayes filter uses measurements to hone in on the right state)
  - We need a model that can deal with **unknown unknowns** (No matter the model, we need to overestimate uncertainty)
  - We would like a model that is **computationally cheap** (Bayes filter repeatedly invokes this model to predict state after actions)
- Key idea: simple model + stochasticity

What motion model should I use for MuSHR?

- A **kinematic model** governs how wheel speeds map to robot velocities
- A **dynamic model** governs how wheel torques map to robot accelerations
- For MuSHR, we'll ignore dynamics and focus on kinematics (assuming the wheel actuators can set speed directly)
- Other assumptions: wheels roll on hard, flat, horizontal ground without slipping

Kinematic Car Model

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

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

X-COORDINATE

Y-COORDINATE

HEADING

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

SPEED

STEERING ANGLE

Kinematic Car Model

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

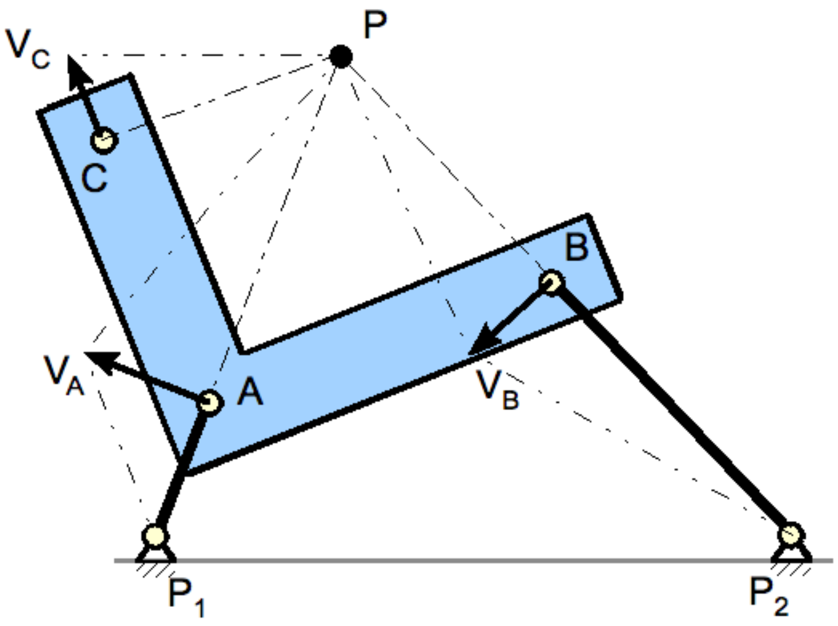
$\rightarrow$ INTEGRATE

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

$\rightarrow$ ADD NOISE

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

Definition: Instant Center of Rotation (CoR)



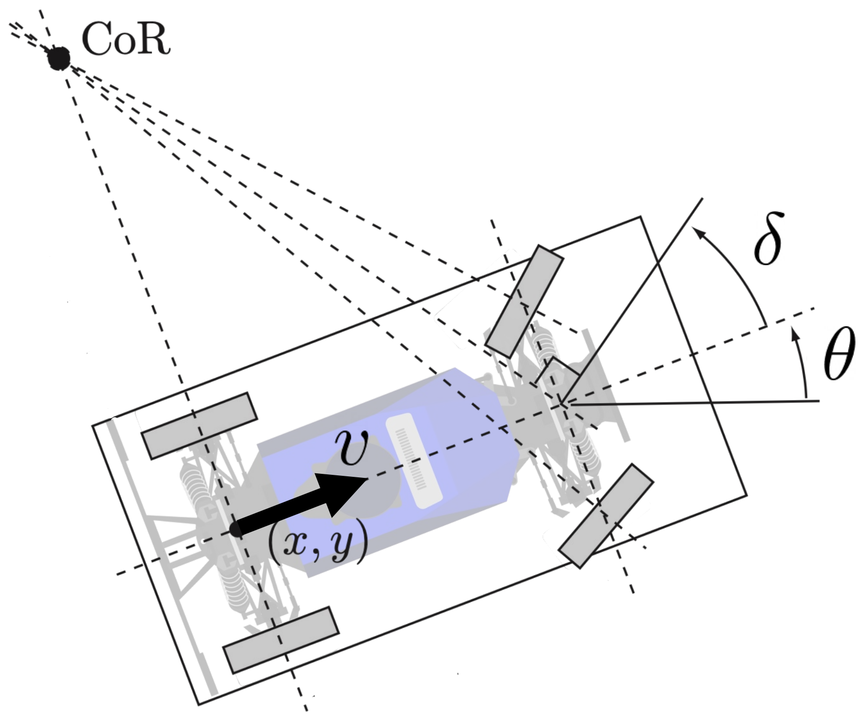
A planar **rigid body** undergoing a **rigid transformation** can be viewed as undergoing a **pure rotation** about an instant center of rotation.

rigid body: a non-deformable object

rigid transformation: a combined rotation and translation

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

Equations of Motion

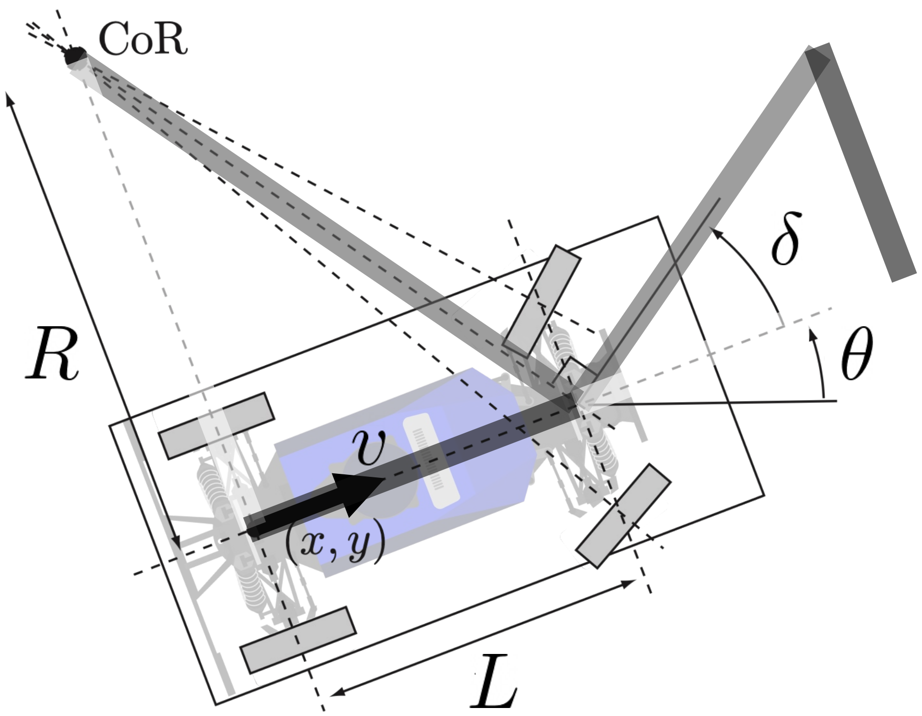


$$\dot{x} = v \cos \theta$$

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

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

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

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

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

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

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

$$\boxed{\begin{aligned}\dot x &= v\cos\theta\\ \dot y &= v\sin\theta\\ \dot\theta &= \frac{v}{L}\tan\delta\end{aligned}}$$

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

Kinematic Car Update

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

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

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

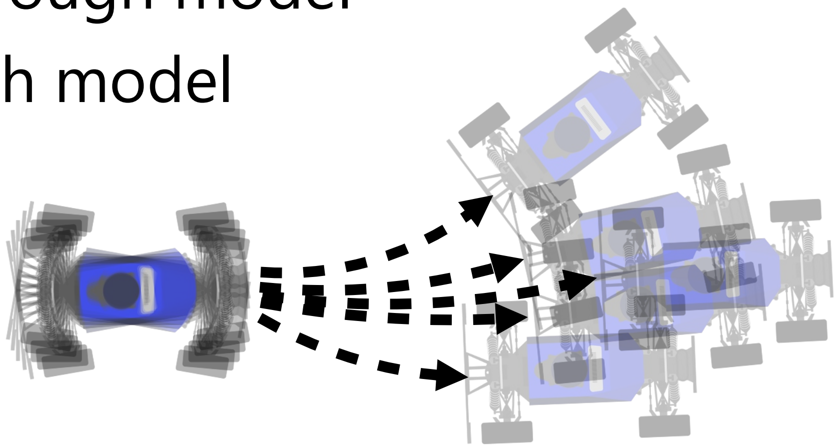
Kinematic Car Model

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

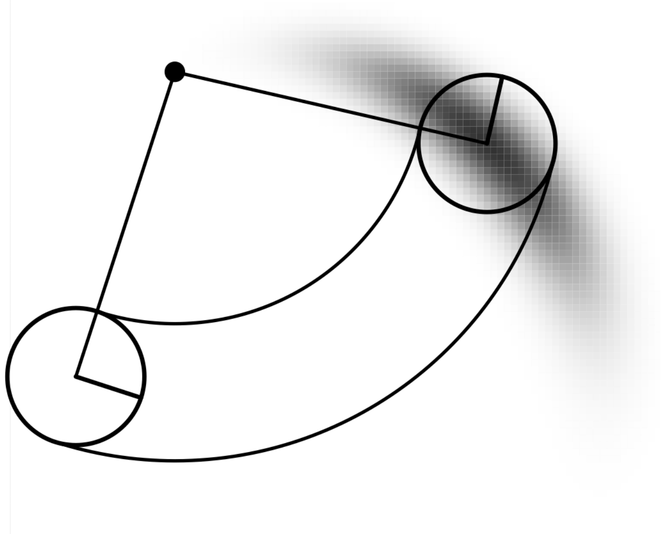
$$\xrightarrow[\text{ADD NOISE}]{} P(x_t | u_t, x_{t-1})$$

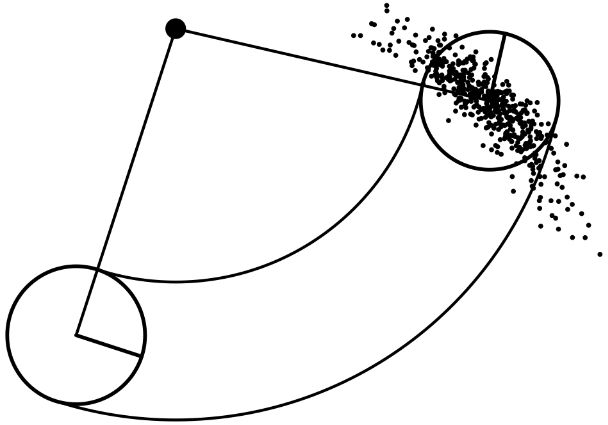
Why is the kinematic car model probabilistic?

- 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

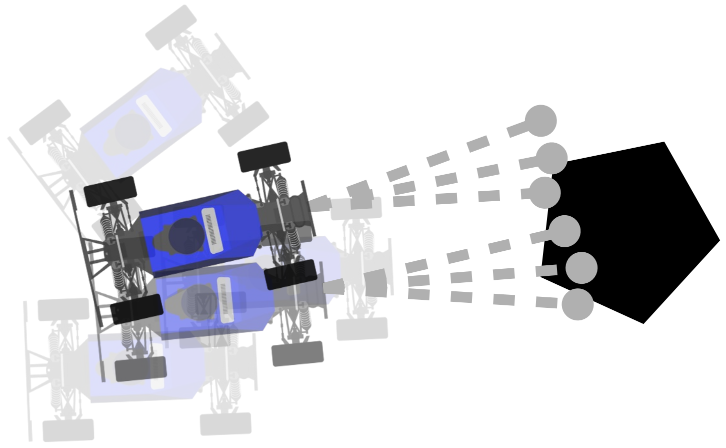
Instantiating Motion Models

Instantiating Sensor Models

Putting together for the Car

Kalman Filtering

Sensor Model

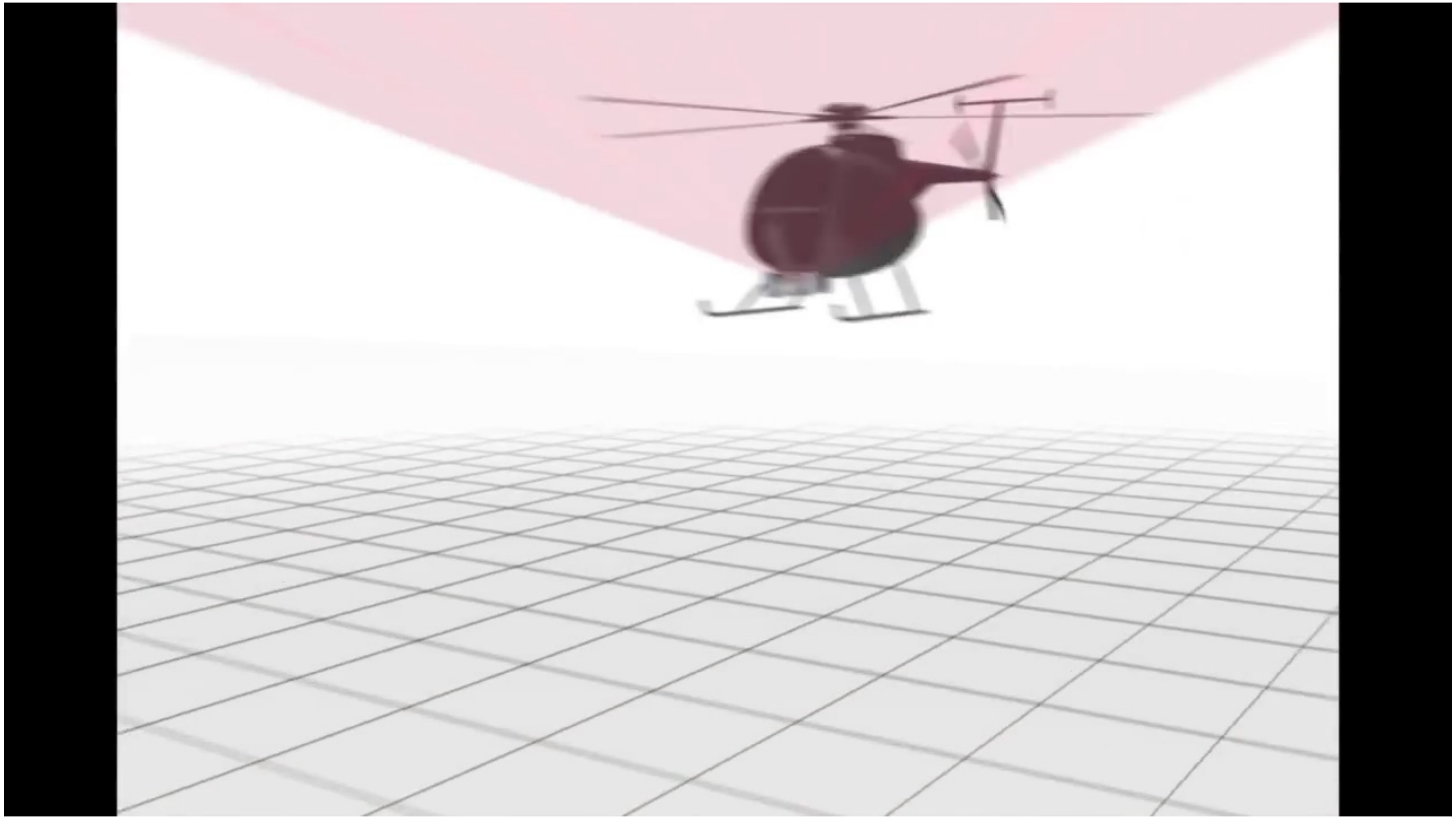


$$P(z_t|x_t)$$

How Does LIDAR Work?

HTTPS://YOUTU.BE/NZKVF1CXE8S

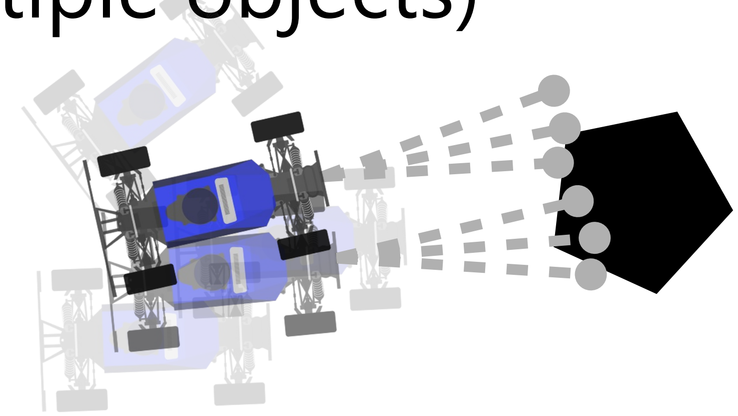
LIDAR in the Real World



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

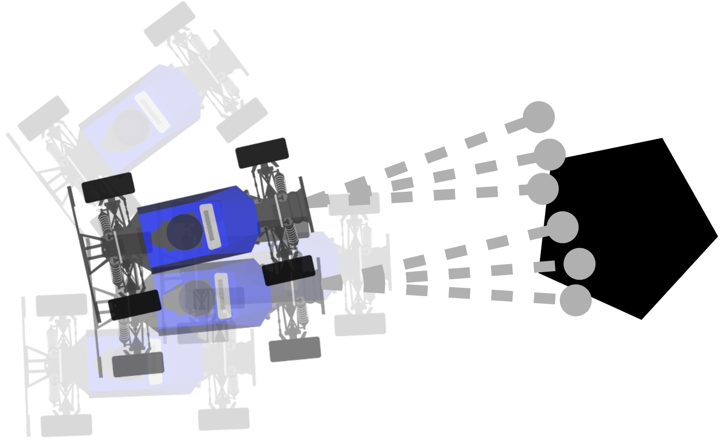
What sensor model should I use for MuSHR?

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

LASER SCAN

STATE

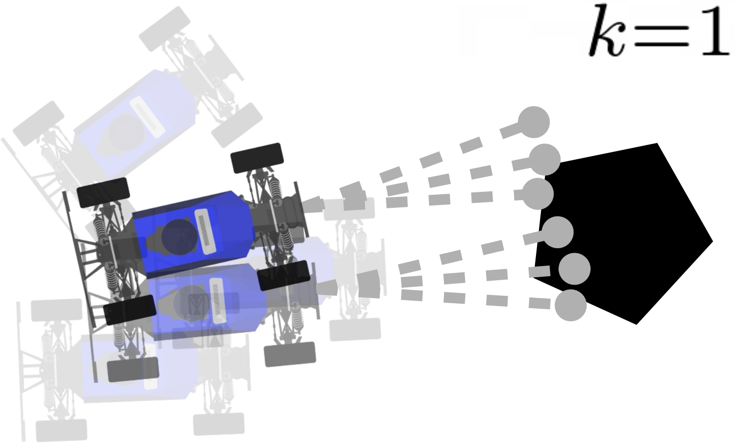
MAP



Assumption: Conditional Independence

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

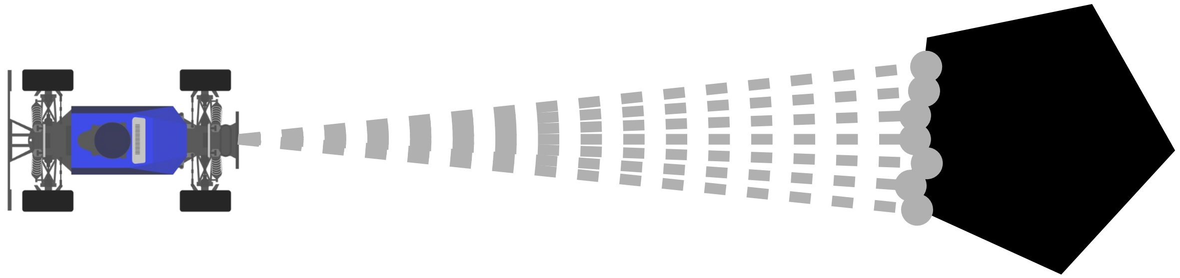
$$= \prod_{k=1}^{K} P(z_t^k|x_t, m)$$



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

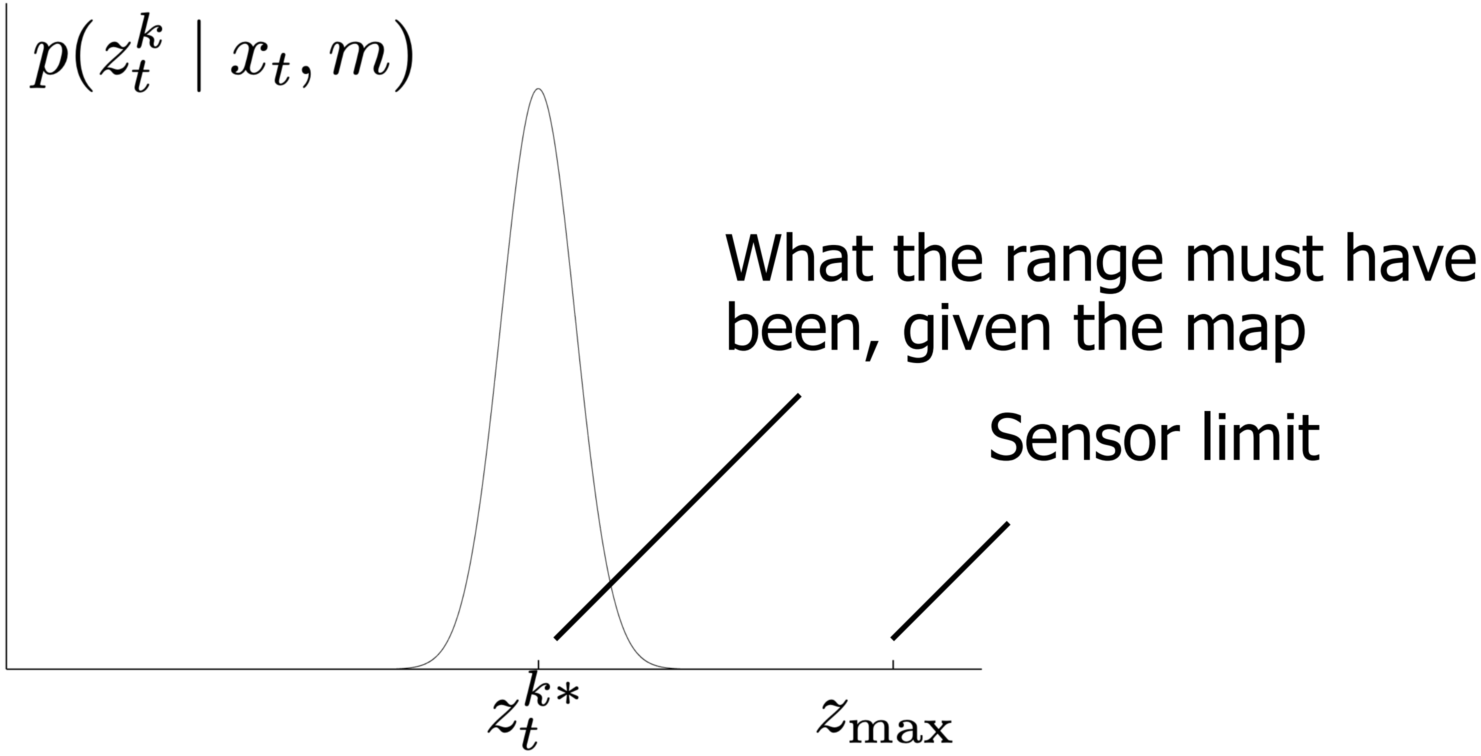
$$P(z_t^k | x_t, m)$$

DISTANCE

Typical Sources of Stochasticity

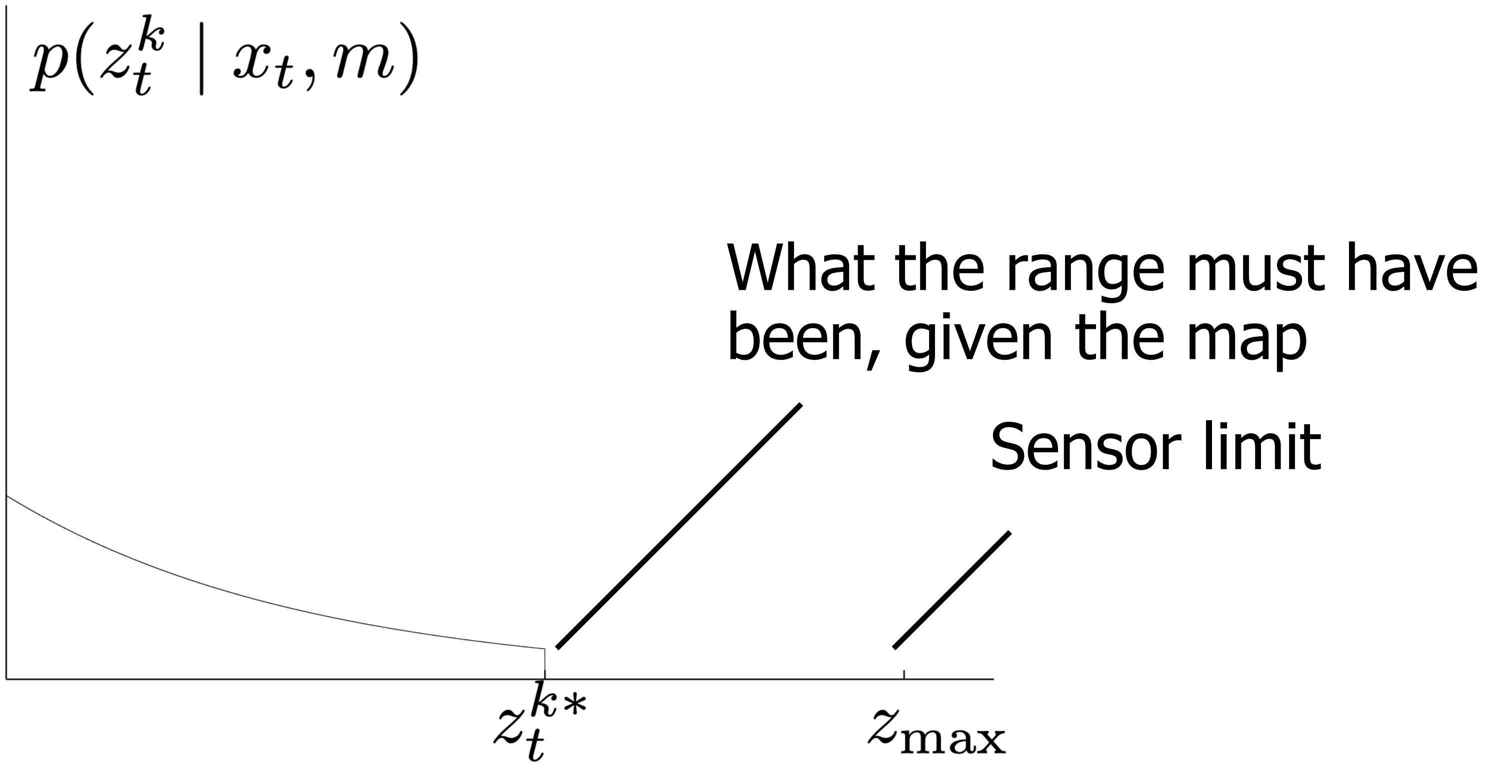
1. Correct range (distance) with local measurement noise
2. Unexpected objects
3. Sensor failures
4. Random measurements

Factor 1: Local Measurement Noise



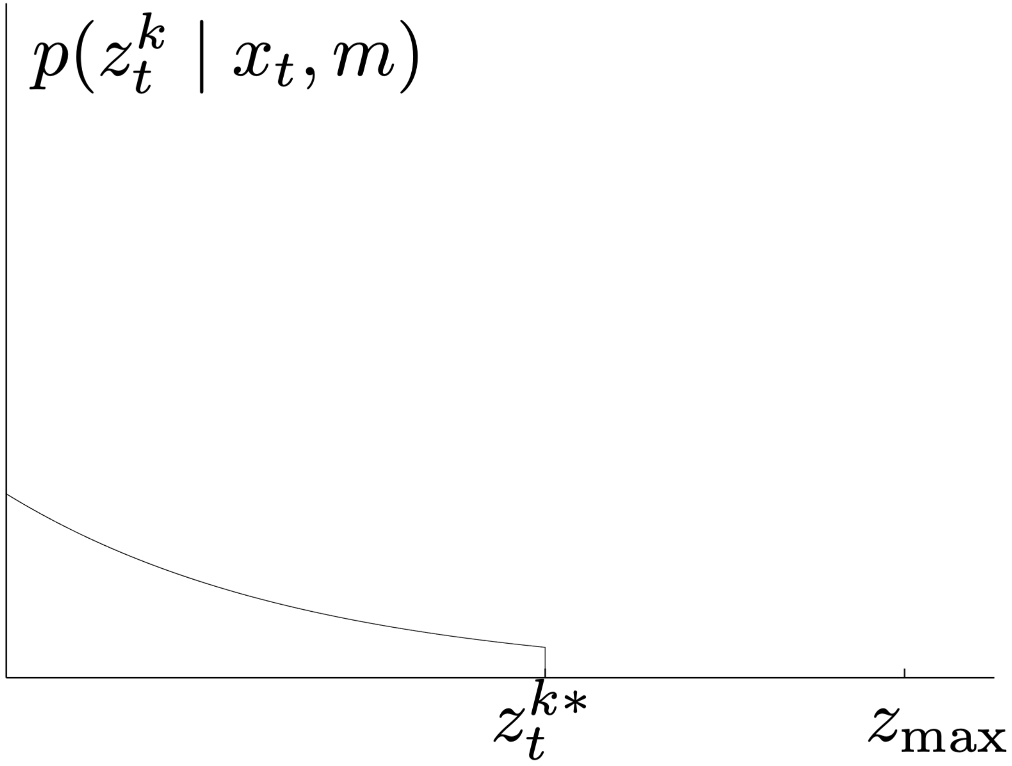
$$p_{\text{hit}}(z_t^k \mid x_t, m) \quad = \quad \begin{cases} \eta\,\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \le z_t^k \le z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Factor 2: Unexpected Objects



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

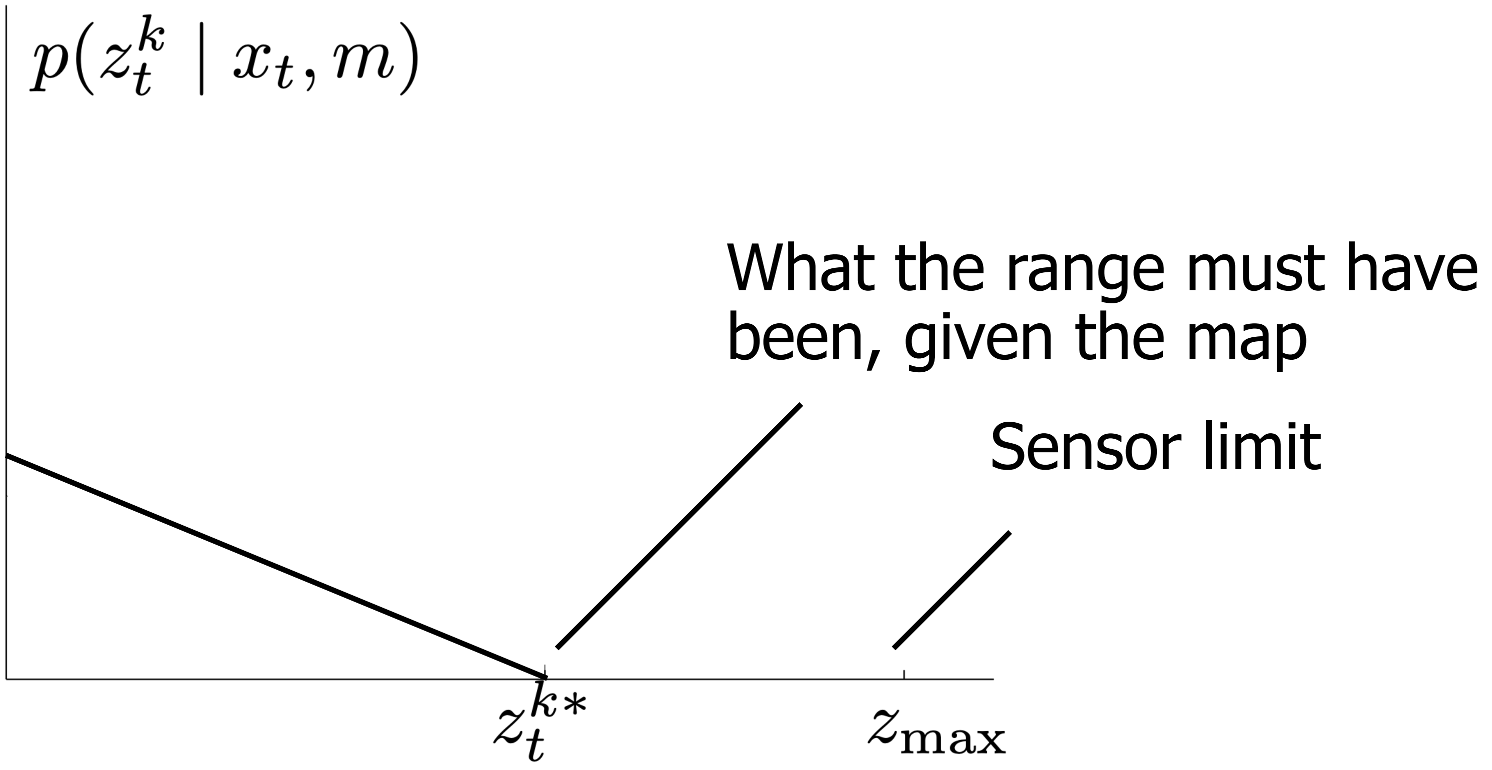
Factor 2: Unexpected Objects



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

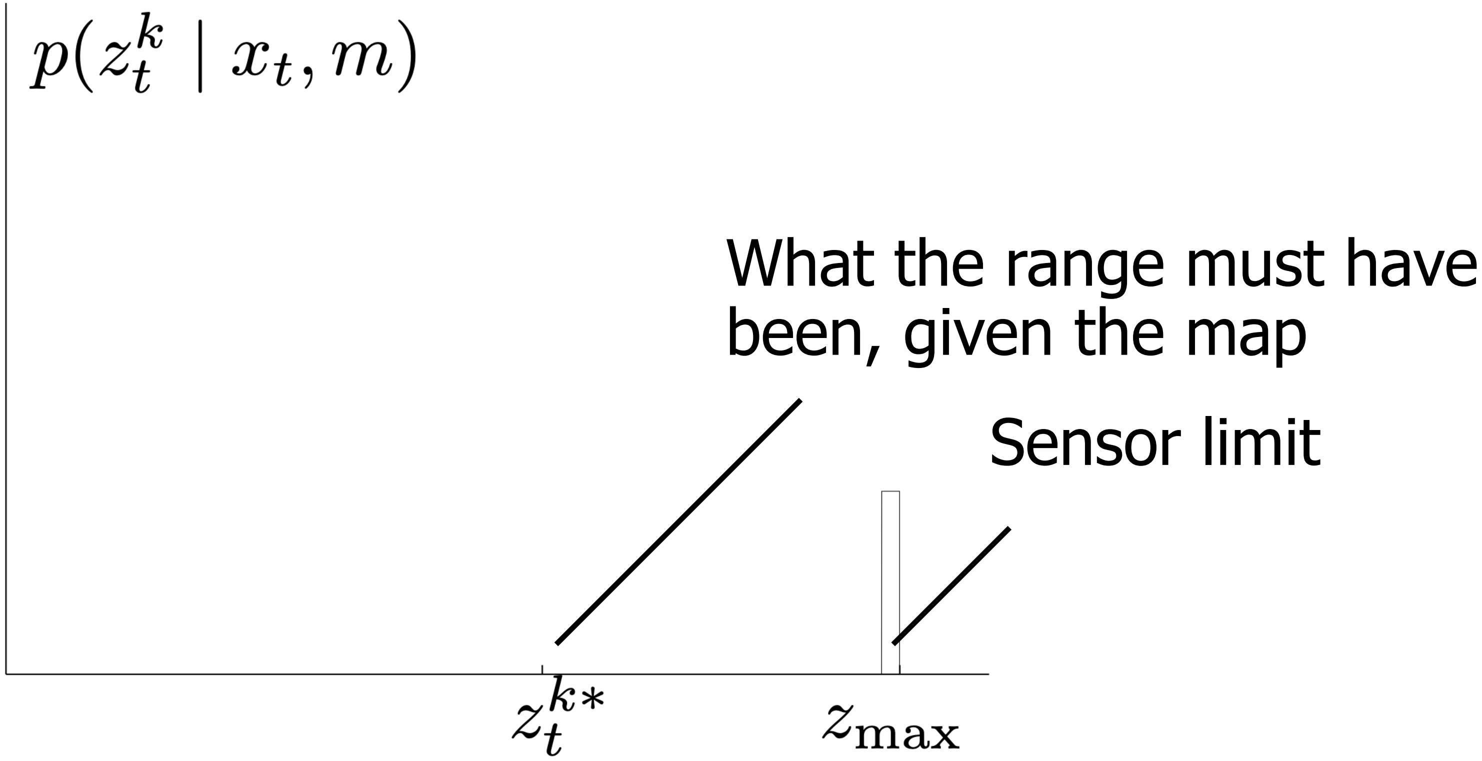
$$p_{\text{short}}(z_t^k \mid x_t, m) \quad = \quad \begin{cases} \eta \, \lambda_{\text{short}} \; e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases}$$

Factor 2: Unexpected Objects (Project)



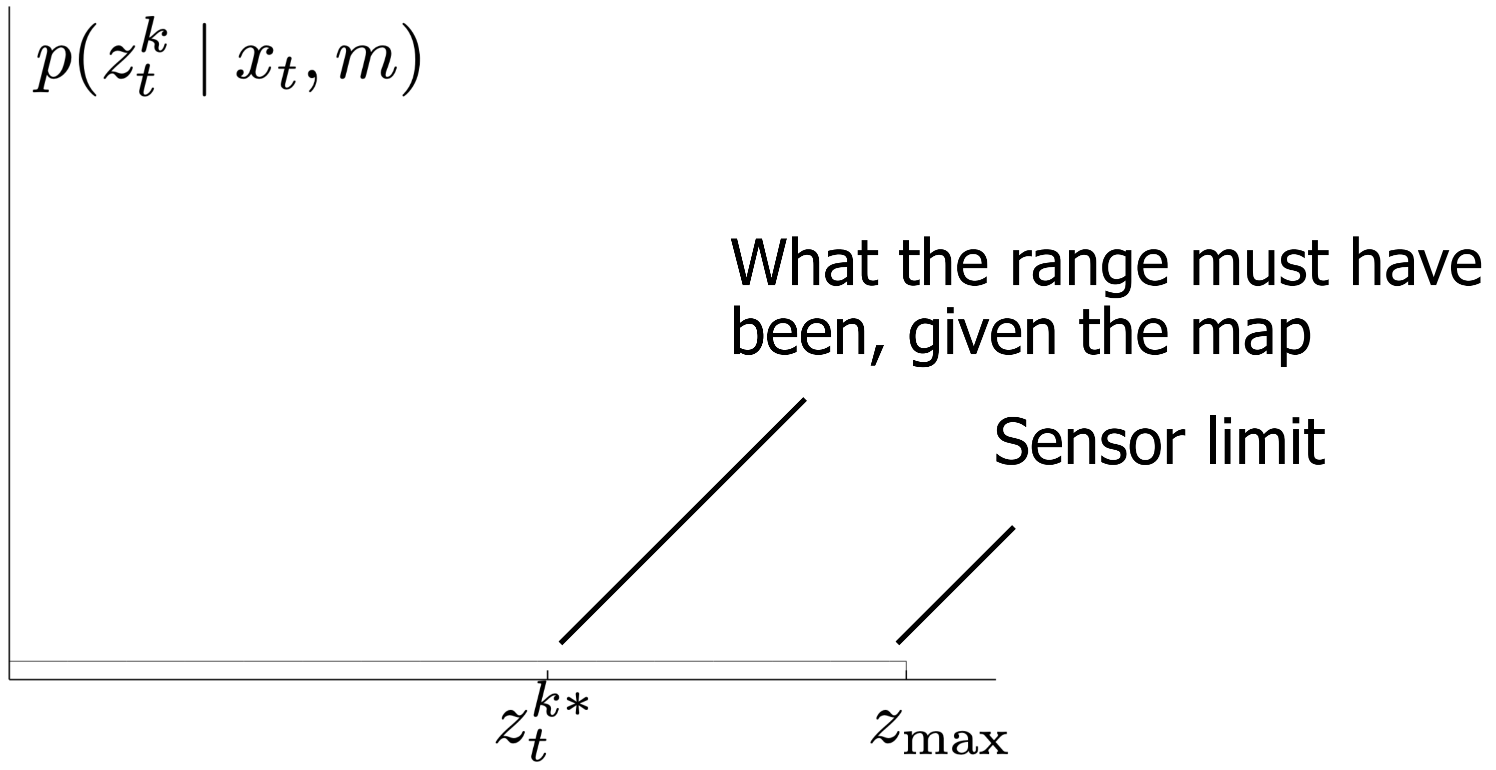
$$p_{\text{short}}(z_t^k | x_t, m) = \begin{cases} 2\frac{z_t^{k^*} - z_t^k}{z_t^{k^*}} & \text{if } z_t^k < z_t^{k^*} \\ 0 & \text{otherwise} \end{cases}$$

Factor 3: Sensor Failures



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

Factor 4: Random Measurements

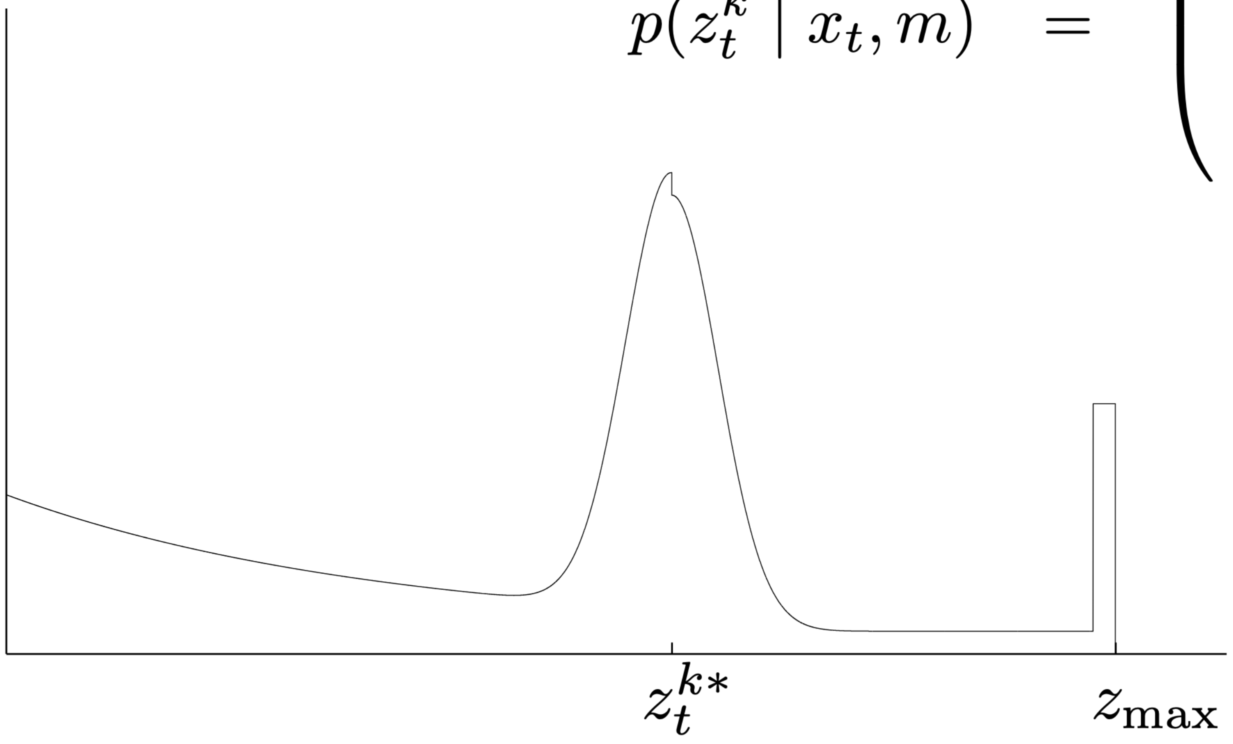


$$p_{\text{rand}}(z_t^k \mid x_t, m) \quad = \quad \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z_t^k < z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

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

Lecture Outline

Instantiating Motion Models

↓

Instantiating Sensor Models

↓

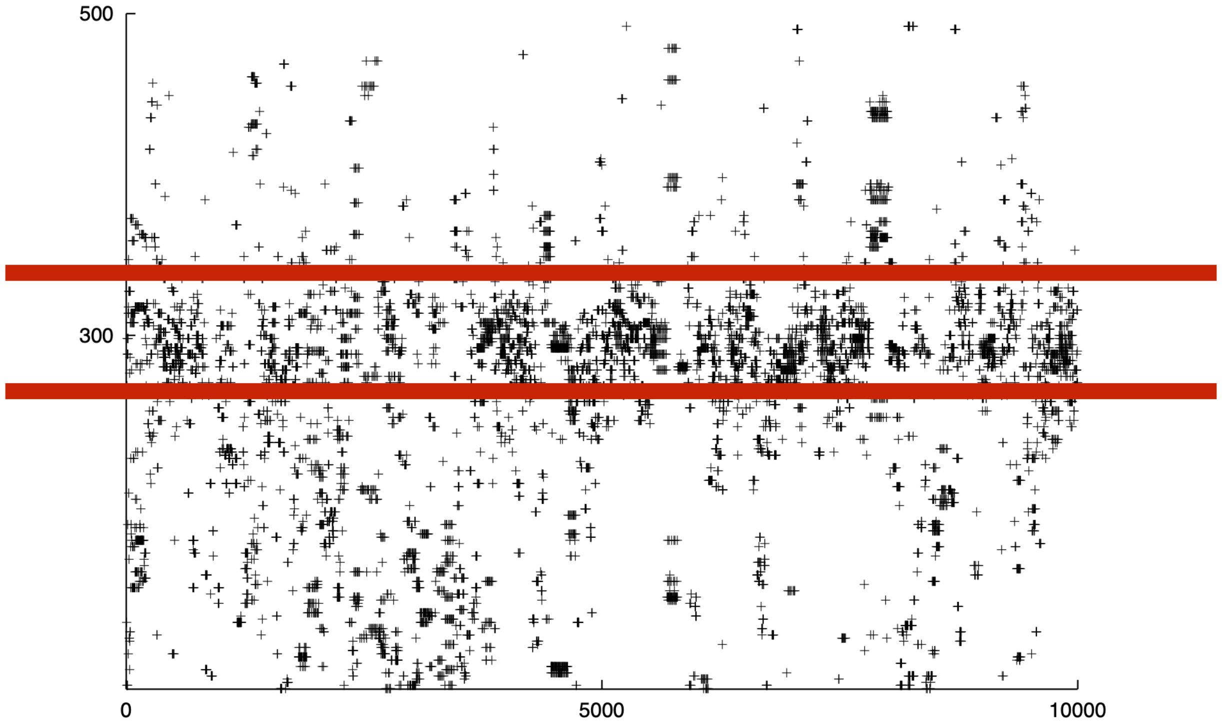
Putting together for the Car

↓

Kalman Filtering

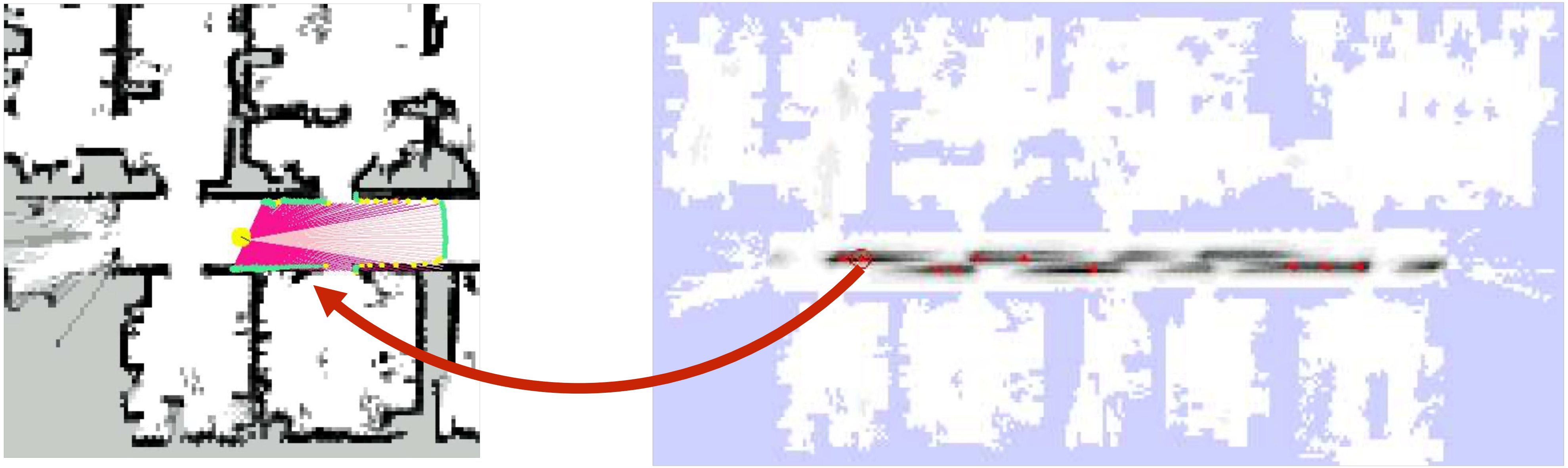
Tuning Single Beam Parameters

- 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

Instantiating Motion Models

↓

Instantiating Sensor Models

↓

Putting together for the Car

↓

Kalman Filtering

What makes this challenging?

Need to choose form of probability distributions

- Dynamics (Prediction)

$$\overline{Bel}(x_t) = \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

- Measurement (Correction)

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

Tractable computation of Bayesian posteriors

What makes this challenging?

- Dynamics (Prediction)

$$\overline{Bel}(x_t) = \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

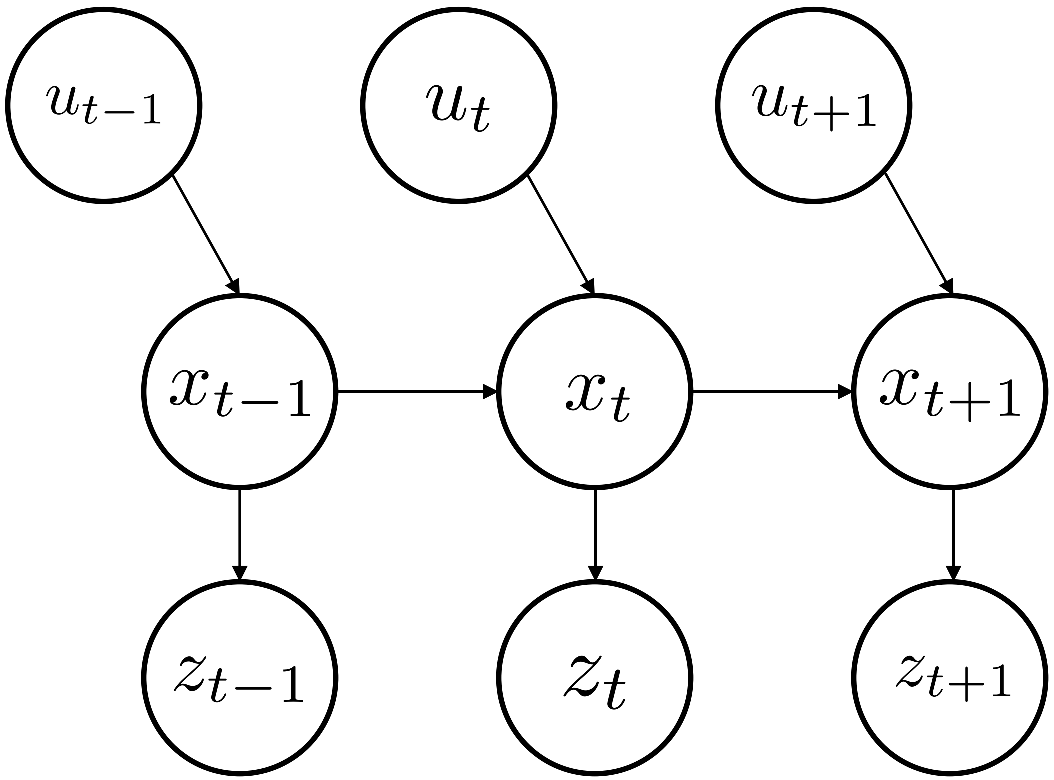
- Measurement (Correction)

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

Model as Linear Gaussian

Discrete Kalman Filter

Kalman filter = Bayes filter with Linear Gaussian dynamics and sensor models



Discrete Kalman Filter

Estimates the state *x* of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_{t+1} = Ax_t + Bu_t + \epsilon_t$$
$$\epsilon_t \sim \mathcal{N}(0, Q)$$

Linear Gaussian

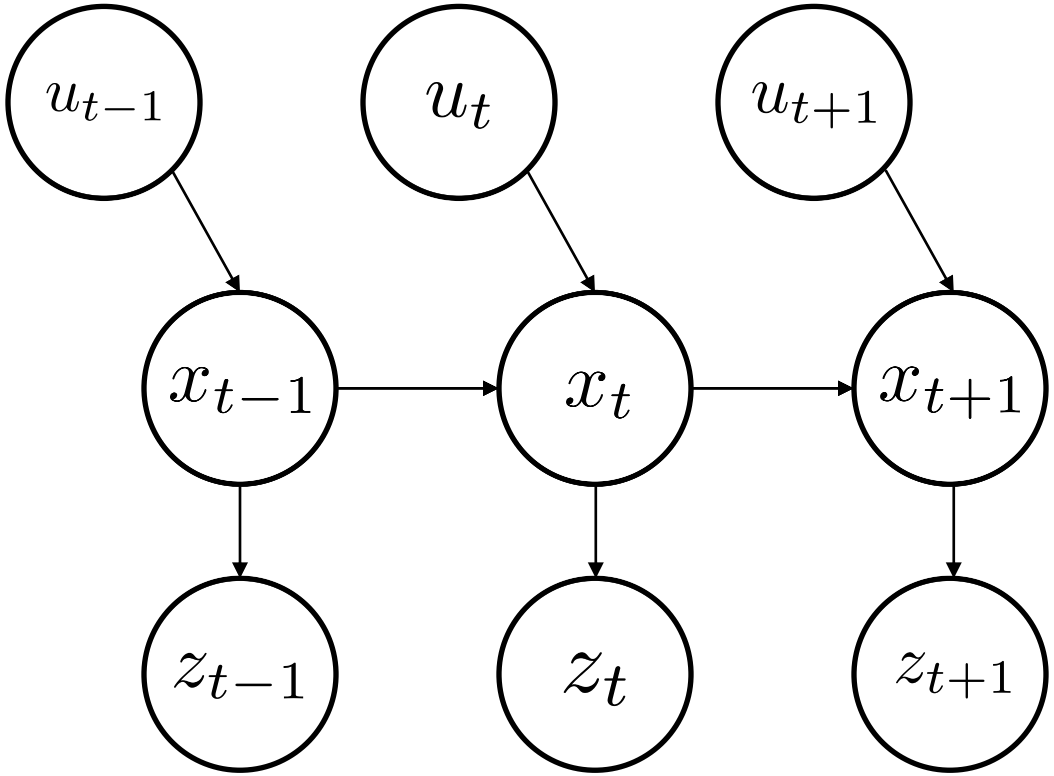
with a measurement

$$z_{t+1} = Cx_{t+1} + \delta_t$$
$$\delta_t \sim \mathcal{N}(0, R)$$

Components of a Kalman Filter

A	Matrix (n x n) that describes how the state evolves from *t-1* to *t* without controls or noise.
B	Matrix (n x l) that describes how the control u_{t-1} changes the state from *t-1* to *t*
C	Matrix (k x n) that describes how to map the state x_t to an observation z_t.
ϵ_t δ_t	Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance *R* and *Q* respectively.

Goal of the Kalman Filter



Belief

$$p(x_t | z_{0:t}, u_{0:t-1})$$

Idea: recursive update

$$\propto p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}) p(x_{t-1}|z_{0:t-1}, u_{0:t-2}) dx_{t-1}$$

Measurement

Dynamics

Recursive Belief

2 step process:

- Dynamics update (incorporate action)
- Measurement update (incorporate sensor reading)

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