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

\[ \text{bel}(x_{t-1}) \]

Step 1: Prediction - push belief through dynamics given action

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

Step 2: Correction - apply Bayes rule given measurement

\[ \text{bel}(x_t) = \frac{n P(z_t | x_t) \overline{\text{bel}}(x_t)} {\sum P(z_t | x_t) \overline{\text{bel}}(x_t)} \]
Let’s ground this in the context of the car

\[ P(x_t | u_t, x_{t-1}) \]

\[ P(z_t | x_t) \]
Motion Model

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

\[ P(x_t | u_t, x_{t-1}) \]
A Spectrum of Motion Models

Highest-fidelity models capturing everything we know

(Red Bull F1 Simulator)

VS

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
- \( \begin{bmatrix} \nu \\ \delta \end{bmatrix} \) - SPEED, STEERING ANGLE
\begin{align*}
\dot{x} &= f(x, u) \\
\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 P(x_t | u_t, x_{t-1})
\end{align*}
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
Equations of Motion

\[ \dot{x} = v \cos \theta \]
\[ \dot{y} = v \sin \theta \]
\[ \dot{\theta} = ? \]
Equations of Motion

\[
\begin{align*}
\dot{x} &= v \cos \theta \\
\dot{y} &= v \sin \theta \\
\dot{\theta} &= \omega = \frac{v}{R} = \frac{v \tan \delta}{L} \\
\tan \delta &= \frac{L}{R} \quad \rightarrow \quad R = \frac{L}{\tan \delta}
\end{align*}
\]
\[ \dot{x} = f(x, u) \]

\[ \begin{bmatrix} x_{t-1} + \Delta x \\ y_{t-1} + \Delta y \\ \theta_{t-1} + \Delta \theta \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \]
Integrate the Kinematics Numerically

Assume that steering angle is \textit{piecewise constant} between \( t \) and \( t' \)

\[
\begin{align*}
\dot{x} &= v \cos \theta \\
\dot{y} &= v \sin \theta \\
\dot{\theta} &= \frac{v}{L} \tan \delta
\end{align*}
\]
Integrate the Kinematics Numerically

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

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

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

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

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

Assume that steering angle is piecewise constant between \( t \) and \( t' \)
\[ \theta_t = \theta_{t-1} + \Delta \theta = \theta_{t-1} + \frac{v}{L} \tan \delta \Delta t \]

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

\[ y_t = y_{t-1} + \Delta y = y_{t-1} + \frac{L}{\tan \delta} (\cos \theta_{t-1} - \cos \theta_t) \]
Kinematic Car Model

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

\[
\Rightarrow \quad P(x_t | u_t, x_{t-1})
\]
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

- 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://YOUTUBE.COM/NZKVFCX8S
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

\[ P(z_t | x_t, m) = P(z_1^t, z_2^t, \ldots, z^K_t | x_t, m) = \prod_{k=1}^{K} P(z_t^k | x_t, m) \]
Assumption: Conditional Independence

\[ P(z_t|x_t, m) = P(z_1^t, z_2^t, \ldots, z_K^t|x_t, m) = \prod_{k=1}^{K} P(z_k^t|x_t, m) \]
Single Beam Sensor Model

\[ P(z^k_t \mid x_t, m) \]
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

What the range must have been, given the map

Sensor limit

\[ p(z_t^k \mid x_t, m) \]

\[ p_{\text{hit}}(z_t^k \mid x_t, m) = \begin{cases} \eta \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \leq z_t^k \leq z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \]
Factor 2: Unexpected Objects

What the range must have been, given the map

Sensor limit

\[ p(z_t^k \mid x_t, m) \]

\[ p_{\text{short}}(z_t^k \mid x_t, m) = \begin{cases} \eta \lambda_{\text{short}} e^{-\lambda_{\text{short}}z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases} \]
Factor 2: Unexpected Objects

\[ p(z_t^k \mid x_t, m) \]

\[ p_{\text{short}}(z_t^k \mid x_t, m) = \begin{cases} \eta \lambda_{\text{short}} e^{-\lambda_{\text{short}}z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases} \]
Factor 2: Unexpected Objects (Project)

\[ p(z^k_t \mid x_t, m) \]

What the range must have been, given the map

Sensor limit

\[
p_{\text{short}}(z^k_t \mid x_t, m) = \begin{cases} 
2 \frac{z^k_t - z^k_t}{z^k_t} & \text{if } z^k_t < z^k_t^* \\
0 & \text{otherwise}
\end{cases}
\]
Factor 3: Sensor Failures

What the range must have been, given the map

Sensor limit

\[ p(z_t^k \mid x_t, m) \]

\[ p_{\text{max}}(z_t^k \mid x_t, m) = I(z = z_{\text{max}}) = \begin{cases} 1 & \text{if } z = z_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \]
Factor 4: Random Measurements

$$p(z_t^k \mid x_t, m)$$

What the range must have been, given the map

Sensor limit

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

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

Weights sum to 1
LIDAR Model Algorithm

\[ P(z_t | x_t, m) = \prod_{k=1}^{K} P(z^k_t | x_t, m) \]

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](image1.png)
![Likelihood at various locations](image2.png)
Dealing with Overconfidence

\[
P(z_t | x_t, m) = \prod_{k=1}^{K} P(z^k_t | x_t, m)
\]

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

\[
P(z^k_t | x_t, m) \to P(z^k_t | x_t, m)^\alpha, \alpha < 1
\]
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)**
  \[ Bel(x_t) = \int P(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \]

- **Measurement (Correction)**
  \[ Bel(x_t) = \eta P(z_t|x_t) Bel(x_t) \]

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

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

with a measurement

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

Linear Gaussian
### Components of a Kalman Filter

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>Matrix $(n \times n)$ that describes how the state evolves from $t-1$ to $t$ without controls or noise.</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>Matrix $(n \times l)$ that describes how the control $u_{t-1}$ changes the state from $t-1$ to $t$.</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Matrix $(k \times n)$ that describes how to map the state $x_t$ to an observation $z_t$.</td>
</tr>
<tr>
<td>$\epsilon_t$</td>
<td>Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance $R$ and $Q$ respectively.</td>
</tr>
</tbody>
</table>
Goal of the Kalman Filter

2 step process:

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

Belief

\[ p(x_t | z_0:t, u_0:t-1) \]

Idea: recursive update

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