Temporal inference and SLAM
Homework 6 has been posted and is due on April 24th.
- Any questions about the homework?
- How many of you have looked at/started/finished homework 6?

Homework 7 will be posted on Wednesday and will be due on Friday 3rd.
- It will be shorter so that it can fit in the 1.5 week you will have for it.
- Do you prefer the deadline to be on May 5th (with the same content)?
- You can use all of your remaining late days for it.

Changes to office hours this Wednesday:
- Yannis, 3 – 5 pm.
- Neeraj, 5 – 7 pm.
- All four hours will take place at Smith Hall 225 / graphics lounge.
Any topics you want covered in the last two lectures?

Current plan is two of the following:

- Color.
- Faces.
- Segmentation.
- Graph-based techniques.
<table>
<thead>
<tr>
<th>Section</th>
<th>Lectures</th>
<th>See also</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Image processing.</td>
<td>1 – 7</td>
<td>18-793: Image and Video Processing</td>
</tr>
<tr>
<td>4. Semantic vision.</td>
<td>17 – 20</td>
<td>16-824: Vision Learning and Recognition</td>
</tr>
<tr>
<td>5. Dealing with motion.</td>
<td>21 – 24</td>
<td>16-831: Statistical Techniques in Robotics</td>
</tr>
</tbody>
</table>

Registration week
15-463/15-663/15-862
Computational Photography

Learn about scientific and unconventional cameras – and build your own!

cameras that see around corners

cameras that capture video at the speed of light

cameras that measure depth in real time

cameras that measure entire lightfields

http://graphics.cs.cmu.edu/courses/15-463/
Overview of today’s lecture

• Leftover from last time: Mean-shift tracking.
• Temporal state models.
• Temporal inference.
• Kalman filtering.
• Extended Kalman filtering.
• Mono SLAM.
Most of these slides were adapted from:

Temporal state models
Represent the ‘world’ as a set of random variables $X$

$X = \{x, y\}$  
location on the ground plane

$X = \{x, y, z\}$  
position in the 3D world

$X = \{x, \dot{x}\}$  
position and velocity

$X = \{x, \dot{x}, f_1, \ldots, f_n\}$

position, velocity and location of landmarks
Object tracking (localization)

\[ X = \{x, y\} \]

e.g., location on the ground plane

Object location and world landmarks
(localization and mapping)

\[ X = \{x, \dot{x}, f_1, \ldots, f_n\} \]

e.g., position and velocity of robot and location of landmarks
The state of the world changes over time
The state of the world changes over time

So we use a sequence of random variables:

\[ X_0, X_1, \ldots, X_t \]
The state of the world changes over time

So we use a sequence of random variables:

$$X_0, X_1, \ldots, X_t$$

The state of the world is usually uncertain so we think in terms of a distribution

$$P(X_0, X_1, \ldots, X_t)$$

How big is the space of this distribution?
If the state space is \( X = \{x, y\} \) the location on the ground plane

\[
P(X_0, X_1, \ldots, X_t)
\]

is the probability over all possible trajectories through a room of length \( t+1 \)
When we use a sensor (camera), we don’t have direct access to the state but noisy observations of the state

\[ E_t \]

\[ X_0, X_1, \ldots, X_t, E_1, E_2, \ldots, E_t \]

(all possible ways of observing all possible trajectories)

How big is the space of this distribution?
all possible ways of observing all possible trajectories of length t

true trajectory

observations $E$
So we think of the world in terms of the distribution

\[ P(X_0, X_1, \ldots, X_t, E_1, E_2, \ldots, E_t) \]

- unobserved variables (hidden state)
- observed variables (evidence)
So we think of the world in terms of the distribution

\[ P(X_0, X_1, \ldots, X_t, E_1, E_2, \ldots, E_t) \]

unobserved variables (hidden state) \hspace{1cm} observed variables (evidence)

**How big is the space of this distribution?**
So we think of the world in terms of the distribution

\[ P(X_0, X_1, \ldots, X_t, E_1, E_2, \ldots, E_t) \]

unobserved variables (hidden state)  
observed variables (evidence)

*How big is the space of this distribution?*

*Can you think of a way to reduce the space?*
Reduction 1. Stationary process assumption:

‘a process of change that is governed by laws that do not themselves change over time.’

\[ P(E_t|X_t) = P_t(E_t|X_t) \]

the model doesn’t change over time
Reduction 1. Stationary process assumption:

‘a process of change that is governed by laws that do not themselves change over time.’

\[ P(E_t|X_t) = P_t(E_t|X_t) \]

the model doesn’t change over time

Only have to store one model.

Is this a reasonable assumption?
Reduction 2. Markov Assumption:

‘the current state only depends on a finite history of previous states.’

First-order Markov Model: \( P(X_t | X_{t-1}) \).

Second-order Markov Model: \( P(X_t | X_{t-1}, X_{t-2}) \)

(this relationship is called the motion model)
Reduction 2. Markov Assumption:

‘the current observation only depends on current state.’

The current observation is usually most influenced by the current state

\[ P(E_t | X_t) \]

(this relationship is called the observation model)

Can you think of an observation of a state?
For example, GPS is a noisy observation of location.

But GPS tells you that you are here with probability $P(E_t | X_t)$.
Reduction 3. Prior State Assumption:

‘we know where the process (probably) starts’
Applying these assumptions, we can decompose the joint probability:

\[ P(X_0, X_1, \ldots, X_T, E_1, E_1, \ldots, E_T) = P(X_0) \prod_{t=1}^{T} P(X_t | X_{t-1}) P(E_t | X_t) \]

**Stationary process assumption:**
only have to store ____ models
(assuming only a single variable for state and observation)

**Markov assumption:**
This is a model of order ___

We have significantly reduced the number of parameters
Joint Probability of a Temporal Sequence

\[ P(X_0) \prod_{t=1}^{T} P(X_t|X_{t-1}) P(E_t|X_t) \]

state prior prior
motion model transition model
sensor model observation model
Joint Probability of a Temporal Sequence

\[ P(X_0) \prod_{t=1}^{T} P(X_t | X_{t-1}) P(E_t | X_t) \]

- State prior
- Motion model
- Sensor model

Joint Distribution for a Dynamic Bayesian Network

Specific instances of a DBN covered in this class

Hidden Markov Model

(typically taught as discrete but not necessarily)

Kalman Filter

(Gaussian motion model, prior and observation model)
Hidden Markov Model
Hidden Markov Model example

‘In the trunk of a car of a sleepy driver’ model

binary random variable (left lane or right lane)

\[ x = \{x_{\text{left}}, x_{\text{right}}\} \]
From a hole in the car you can see the ground

binary random variable (road is yellow or road is gray)

\[ e = \{ e_{\text{gray}}, e_{\text{yellow}} \} \]
What needs to sum to one?

What's the probability of staying in the left lane if I'm in the left lane?

What lane am I in if I see yellow?
visualization of the motion model

\[
P(x_t|x_{t-1}) | \quad x_{t-1} = R \quad x_{t-1} = S
\]

<table>
<thead>
<tr>
<th>(x_t = R)</th>
<th>(x_t = S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Is the stationary assumption true?
Is the stationary assumption true?

visibility at night?
visibility after a day in the car?
still swerving after one day of driving?
Is the stationary assumption true?

visibility at night?
visibility after a day in the car?
still swerving after one day of driving?

Is the Markov assumption true?
Is the stationary assumption true?

- visibility at night?
- visibility after a day in the car?
- still swerving after one day of driving?

Is the Markov assumption true?

- what can you learn with higher order models?
- what if you have been in the same lane for the last hour?

In general, assumptions are not correct but they simplify the problem and work most of the time when designed appropriately.
Temporal inference
Basic Inference Tasks

**Filtering**

\[ P(X_t | e_{1:t}) \]

Posterior probability over the **current** state, given all evidence up to present

**Prediction**

\[ P(X_{t+k} | e_{1:t}) \]

Posterior probability over a **future** state, given all evidence up to present

**Smoothing**

\[ P(X_k | e_{1:t}) \]

Posterior probability over a **past** state, given all evidence up to present

**Best Sequence**

\[
\arg \max_{X_{1:t}} P(X_{1:t} | e_{1:t})
\]

Best state sequence given all evidence up to present
Filtering

$$P(X_t | e_{1:t})$$

Posterior probability over the current state, given all evidence up to present

Where am I now?
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})
\]

posterior  observation model  motion model  prior
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})
\]

What is this?
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})
\]

same type of ‘message’
Filtering

Can be computed with recursion (Dynamic Programming)

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t}) \]

same type of ‘message’

called a belief distribution

sometimes people use this annoying notation instead: \( Bel(x_t) \)

a belief is a reflection of the systems (robot, tracker) knowledge about the state \( X \)
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})
\]

*Where does this equation come from?*

*(scary math to follow…)*
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})
\]

just splitting up the notation here

\[
P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t})
\]
Filtering

Can be computed with recursion (Dynamic Programming)

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t}) \]

\[ P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t}) \]

Apply Bayes' rule (with evidence)
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})
\]

\[
P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t})
\]

\[
= \frac{P(e_{t+1}|X_{t+1}, e_{1:t}) P(X_{t+1}|e_{1:t})}{P(e_{t+1}|e_{1:t})}
\]

Apply Markov assumption on observation model
Filtering

Can be computed with recursion (Dynamic Programming)

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t}) \]

\[ P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t}) \]

\[ = \frac{P(e_{t+1}|X_{t+1}, e_{1:t})P(X_{t+1}|e_{1:t})}{P(e_{t+1}|e_{1:t})} \]

\[ = \alpha P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \]

Condition on the previous state \( X_t \)
Filtering

Can be computed with recursion (Dynamic Programming)

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t}) \]

\[ P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t}) \]

\[ = \frac{P(e_{t+1}|X_{t+1}, e_{1:t})P(X_{t+1}|e_{1:t})}{P(e_{t+1}|e_{1:t})} \]

\[ = \alpha P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \]

\[ = \alpha P(e_{t+1}|X_{t+1})\sum_{X_t} P(X_{t+1}|X_t, e_{1:t})P(X_t|e_{1:t}) \]

Apply Markov assumption on motion model
Filtering

Can be computed with recursion (Dynamic Programming)

\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})
\]

\[
P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{t+1}, e_{1:t})
\]

\[
= \frac{P(e_{t+1}|X_{t+1}, e_{1:t})P(X_{t+1}|e_{1:t})}{P(e_{t+1}|e_{1:t})}
\]

\[
= \alpha P(e_{t+1}|X_{t+1}) P(X_{t+1}|e_{1:t})
\]

\[
= \alpha P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t, e_{1:t}) P(X_t|e_{1:t})
\]

\[
= \alpha P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})
\]
Hidden Markov Model example

‘In the trunk of a car of a sleepy driver’ model

binary random variable (left lane or right lane)

\[ x = \{ x_{\text{left}}, x_{\text{right}} \} \]
From a hole in the car you can see the ground

binary random variable (center lane is yellow or road is gray)

\[ e = \{ e_{\text{gray}}, e_{\text{yellow}} \} \]
What needs to sum to one?

This is filtering!

What's the probability of being in the left lane at t=4?
What is the belief distribution if I see yellow at $t=1$ $p(x_1|e_1 = e_{\text{yellow}}) =$?

Prediction step: 
$$p(x_1) = \sum_{x_0} p(x_1|x_0)p(x_0)$$

Update step: 
$$p(x_1|e_1) = \alpha p(e_1|x_1)p(x_1)$$
<table>
<thead>
<tr>
<th>$P(x_0)$</th>
<th>$x_{\text{left}}$</th>
<th>$x_{\text{right}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

| $P(x_t|x_{t-1})$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|---|---|---|
| $x_{\text{left}}$ | 0.7 | 0.3 |
| $x_{\text{right}}$ | 0.3 | 0.7 |

| $P(e_t|x_t)$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|---|---|---|
| $e_{\text{yellow}}$ | 0.9 | 0.2 |
| $e_{\text{gray}}$ | 0.1 | 0.8 |

**Filtering:**

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})$$

*What is the belief distribution if I see yellow at $t=1$? $p(x_1|e_1 = e_{\text{yellow}}) = ?$*

**Prediction step:**

$$p(x_1) = \sum_{x_0} p(x_1|x_0)p(x_0)$$

$$= [0.7 \ 0.3] (0.5) + [0.3 \ 0.7] (0.5)$$

$$= \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$
Filtering:  
\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t}) \]

What is the belief distribution if I see **yellow** at \( t=1 \)  \( p(x_1|e_1 = e_{\text{yellow}}) = ? \)

Update step:  
\[ p(x_1|e_1) = \alpha \ p(e_1|x_1)\ p(x_1) \]
<table>
<thead>
<tr>
<th>$P(x_0)$</th>
<th>$x_{\text{left}}$</th>
<th>$x_{\text{right}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

| $P(x_t|x_{t-1})$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|-----------------|----------------|----------------|
| $x_{\text{left}}$ | 0.7            | 0.3            |
| $x_{\text{right}}$ | 0.3            | 0.7            |

| $P(e_t|x_t)$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|--------------|----------------|----------------|
| $e_{\text{yellow}}$ | 0.9            | 0.2            |
| $e_{\text{gray}}$ | 0.1            | 0.8            |

Filtering:  

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})$$

What is the belief distribution if I see yellow at $t=1$?  

$p(x_1|e_1 = e_{\text{yellow}}) =$

Update step:  

$$p(x_1|e_1) = \alpha \ p(e_1|x_1)p(x_1)$$

$$= \alpha \ (0.9 \ 0.2) . \ (0.5 \ 0.5)$$

$$= \alpha \ \begin{bmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0.1 \end{bmatrix}$$

more likely to be in which lane?
| \( P(x_0) \) | \( X_{\text{left}} \) | \( X_{\text{right}} \) | \( P(x_t|x_{t-1}) \) | \( X_{\text{left}} \) | \( X_{\text{right}} \) | \( P(e_t|x_t) \) | \( X_{\text{left}} \) | \( X_{\text{right}} \) |
|---|---|---|---|---|---|---|---|---|
| 0.5 | 0.5 | \( x_{\text{left}} \) | 0.7 | 0.3 | \( e_{\text{yellow}} \) | 0.9 | 0.2 | \( e_{\text{gray}} \) | 0.1 | 0.8 |

**Filtering:** 
\[
P(X_{t+1|e_{1:t+1}} \propto P(e_{t+1|X_{t+1}}) \sum_{X_t} P(X_{t+1|X_t})P(X_t|e_{1:t})
\]

*What is the belief distribution if I see yellow at t=1* \( p(x_1|e_1 = e_{\text{yellow}}) = ? \)

**Summary**

**Prediction step:** 
\[
p(x_1) = \sum_{x_0} p(x_1|x_0)p(x_0)
\]
\[
= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}
\]

**Update step:** 
\[
p(x_1|e_1) = \alpha p(e_1|x_1)p(x_1)
\]
\[
\approx \begin{bmatrix} 0.818 \\ 0.182 \end{bmatrix}
\]
<table>
<thead>
<tr>
<th>$P(x_0)$</th>
<th>$x_{\text{left}}$</th>
<th>$x_{\text{right}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

| $P(x_t|x_{t-1})$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|---|---|---|
| $x_{\text{left}}$ | 0.7 | 0.3 |
| $x_{\text{right}}$ | 0.3 | 0.7 |

| $P(e_t|x_t)$ | $x_{\text{left}}$ | $x_{\text{right}}$ |
|---|---|---|
| $e_{\text{yellow}}$ | 0.9 | 0.2 |
| $e_{\text{gray}}$ | 0.1 | 0.8 |

**Filtering:**  
$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})$$

*What if you see yellow again at $t=2*  
$p(x_2|e_1, e_2) =$?
Filtering: \[ P(X_{t+1|e_{1:t+1}}) \propto P(e_{t+1|X_{t+1}}) \sum_{X_t} P(X_{t+1|X_t}) P(X_t|e_{1:t}) \]

What if you see yellow again at \( t=2 \) \[ p(x_2|e_1, e_2) = ? \]

Prediction step: \[ p(x_2|e_1) = \sum_{x_1} p(x_2|x_1)p(x_1|e_1) \]

Update step: \[ p(x_1|e_1, e_2) = \alpha p(e_1|x_1)p(x_1) \]
Filtering: 

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t}) \]

What if you see yellow again at \( t=2 \) 

\[ p(x_2|e_1, e_2) =? \]

Prediction step: 

\[ p(x_2|e_1) = \sum_{x_1} p(x_2|x_1)p(x_1|e_1) \]

\[
= \begin{bmatrix}
0.7 & 0.3 \\
0.3 & 0.7
\end{bmatrix} \begin{bmatrix}
0.818 \\
0.182
\end{bmatrix} = \begin{bmatrix}
0.627 \\
0.373
\end{bmatrix}
\]

Why does the probability of being in the left lane go down?
Filtering: $P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t) P(X_t|e_{1:t})$

What if you see yellow again at $t=2$  

$p(x_2|e_1, e_2) =$

Update step: 

$p(x_2|e_1, e_2) = \alpha \cdot \begin{bmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.627 \\ 0.373 \end{bmatrix}$

$= \alpha \begin{bmatrix} 0.883 \\ 0.117 \end{bmatrix}$
Basic Inference Tasks

**Filtering**

\[ P(X_t|e_{1:t}) \]

Posterior probability over the current state, given all evidence up to present

**Prediction**

\[ P(X_{t+k}|e_{1:t}) \]

Posterior probability over a future state, given all evidence up to present

**Smoothing**

\[ P(X_k|e_{1:t}) \]

Posterior probability over a past state, given all evidence up to present

**Best Sequence**

\[ \arg \max_{X_{1:t}} P(X_{1:t}|e_{1:t}) \]

Best state sequence given all evidence up to present
Prediction

\[ P(X_{t+k} | e_{1:t}) \]

Posterior probability over a future state, given all evidence up to present

Where am I going?
Prediction

What happens as you try to predict further into the future?

\[ P(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1}|x_{t+k})P(x_{t+k}|e_{1:t}) \]

same recursive form as filtering but...

no new evidence!
Prediction

\[ P(X_{t+k+1} | e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1} | x_{t+k}) P(x_{t+k} | e_{1:t}) \]

What happens as you try to predict further into the future?

Approaches its ‘stationary distribution’
Basic Inference Tasks

**Filtering**

$$P(X_t | e_{1:t})$$

Posterior probability over the *current* state, given all evidence up to present

**Prediction**

$$P(X_{t+k} | e_{1:t})$$

Posterior probability over a *future* state, given all evidence up to present

**Smoothing**

$$P(X_k | e_{1:t})$$

Posterior probability over a *past* state, given all evidence up to present

**Best Sequence**

$$\arg \max_{X_{1:t}} P(X_{1:t} | e_{1:t})$$

Best state sequence given all evidence up to present
Smoothing

\[ P(X_k | e_{1:t}) \]

Posterior probability over a past state, given all evidence up to present

Wait, what did I do yesterday?
**Smoothing**

\[ P(X_k|e_{1:t}) \quad 1 \leq k < t \]

\[
P(X_k|e_{1:t}) = P(X_k|e_{1:k}, e_{k+1:t})
\]

\[
= \alpha P(X_k|e_{1:k}) P(e_{k+1:t}|X_k, e_{1:k})
\]

\[
= \alpha P(X_k|e_{1:k}) P(e_{k+1:t}|X_k)
\]

'forward' message

'backward' message

some time in the past

this is just filtering

this is backwards filtering

Let me explain…
Backward message

\[ P(e_{k+1:t} | X_k) = \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k) \]
Backward message

\[ P(e_{k+1:t} | X_k) = \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k) \]

\[ = \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1}) P(x_{k+1} | X_k) \]
Backward message

\[ P(e_{k+1:t} | X_k) = \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k) \]  

conditioning

copied from last slide

\[ = \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1}) P(x_{k+1} | X_k) \]  

Markov Assumption

\[ = \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t} | x_{k+1}) P(x_{k+1} | X_k) \]  

split
Backward message

\[ P(e_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(e_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) \]

conditioning

copied from last slide

= \sum_{x_{k+1}} P(e_{k+1:t} \mid x_{k+1}) P(x_{k+1} \mid X_k)

Markov Assumption

= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k)

split

= \sum_{x_{k+1}} P(e_{k+1} \mid x_{k+1}) P(e_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k)

observation model recursive message motion model

This is just a 'backwards' version of filtering where

initial message \[ P(e_{t-1:t} \mid X_t) = 1 \]
Basic Inference Tasks

**Filtering**

\[ P(X_t | e_{1:t}) \]

Posterior probability over the **current** state, given all evidence up to present

**Prediction**

\[ P(X_{t+k} | e_{1:t}) \]

Posterior probability over a **future** state, given all evidence up to present

**Smoothing**

\[ P(X_k | e_{1:t}) \]

Posterior probability over a **past** state, given all evidence up to present

**Best Sequence**

\[ \arg \max_{X_{1:t}} P(X_{1:t} | e_{1:t}) \]

Best state sequence given all evidence up to present
Best Sequence

\[ \arg \max_{X_{1:t}} P(X_{1:t} | e_{1:t}) \]

Best state sequence given all evidence up to present

I must have done something right, right?
Best Sequence

\[
\max_{x_1, \ldots, x_t} P(x_1, \ldots, x_t, X_{t+1} | e_{1:t+1})
\]

\[
= \alpha P(e_{t+1} | X_{t+1}) \max_{x_t} \left[ P(X_{t+1} | x_t) \max_{x_1, \ldots, x_{t-1}} P(x_1, \ldots, x_{t-1}, X_t | e_{1:t}) \right]
\]

Identical to filtering but with a max operator

Recall: Filtering equation

\[
P(X_{t+1} | e_{1:t+1}) \propto P(e_{t+1} | X_{t+1}) \sum_{X_t} P(X_{t+1} | X_t) P(X_t | e_{1:t})
\]
Now you know how to answer all the important questions in life:

Where am I now?

Where am I going?

Wait, what did I do yesterday?

I must have done something right, right?
Kalman filtering
Examples up to now have been **discrete** (binary) random variables

Kalman ‘filtering’ can be seen as a special case of a temporal inference with continuous random variables

Everything is continuous…

\[ x \quad e \quad P(x_0) \quad P(e|x) \quad P(x_t|x_{t-1}) \]

probability distributions are no longer tables but functions
Making the connection to the ‘filtering’ equations

(Discrete) Filtering

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t}) \]

Kalman Filtering

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{\mathbf{x}_t} P(X_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|e_{1:t})d\mathbf{x}_t \]

integral because continuous PDFs
Simple, 1D example…
System (motion) model

\[ x_t = x_{t-1} + s + r_t \]

\[ r_t \sim \mathcal{N}(0, \sigma_R) \]

'sampled from'
How do you represent the motion model?

\[ x_t = x_{t-1} + s + r_t \]

\[ r_t \sim \mathcal{N}(0, \sigma_R) \]

know velocity \quad noise

\[ P(x_t | x_{t-1}) \]
A linear Gaussian (continuous) transition model:

\[ x_t = x_{t-1} + s + r_t \]

\[ r_t \sim \mathcal{N}(0, \sigma_R) \]

How do you represent the motion model?

\[ P(x_t | x_{t-1}) = \mathcal{N}(x_t; x_{t-1} + s, \sigma_r) \]

How can you visualize this distribution?
A linear Gaussian (continuous) transition model

\[ P(x_t|x_{t-1}) = \mathcal{N}(x_t; x_{t-1} + s, \sigma_r) \]

Why don’t we just use a table as before?
GPS measurement $z_1$  

$q_1$ sensor error

$x_1$ True position

$$z_t = x_t + q_t$$

$q_t \sim \mathcal{N}(0, \sigma_Q)$

sampled from a Gaussian

Observation (measurement) model
How do you represent the observation (measurement) model?

\[ z_t = x_t + q_t \]
\[ q_t \sim \mathcal{N}(0, \sigma_Q) \]

\[ P(e | x) \]

e represents z
How do you represent the observation (measurement) model?

Also a linear Gaussian model

$$P(z_t | x_t) = \mathcal{N}(z_t; x_t, \sigma_Q)$$
How do you represent the observation (measurement) model?

Also a linear Gaussian model

\[ P(z_t | x_t) = \mathcal{N}(z_t; x_t, \sigma_Q) \]
Prior (initial) State

$x_0$  true position

$\hat{x}_0$  initial estimate

$\sigma_0$  initial estimate uncertainty

Prior (initial) State
How do you represent the prior state probability?
How do you represent the prior state probability?

Also a linear Gaussian model!

\[ P(\hat{x}_0) = \mathcal{N}(\hat{x}_0; x_0, \sigma_0) \]
How do you represent the prior state probability?

Also a linear Gaussian model!

\[ P(\hat{x}_0) = \mathcal{N}(\hat{x}_0; x_0, \sigma_0) \]
Inference

So how do you do temporal filtering with the KL?
Recall: the first step of filtering was the ‘prediction step’

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})dx_t \]

compute this!
It’s just another Gaussian

need to compute the ‘prediction’ mean and variance…
Prediction
(Using the motion model)

How would you predict $\hat{x}_1$ given $\hat{x}_0$?

using this ‘cap’ notation to denote ‘estimate’

\[
\hat{x}_1 = \hat{x}_0 + s \tag{This is the mean}
\]

\[
\sigma^2_1 = \sigma^2_0 + \sigma_r^2 \tag{This is the variance}
\]
The second step after prediction is …

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{X_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})dx_t \]
\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})dx_t \]

... update step!

compute this (using results of the prediction step)
In the **update step**, the **sensor measurement corrects** the system **prediction**

Which estimate is correct? Is there a way to know? *Is there a way to merge this information?*
Intuitively, the smaller variance means less uncertainty.

This happens naturally in the Bayesian filtering (with Gaussians) framework!
Recall the filtering equation:

\[
P(\mathbf{X}_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|\mathbf{X}_{t+1}) \int_{\mathbf{x}_t} P(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|e_{1:t}) d\mathbf{x}_t
\]

What is the product of two Gaussians?
Recall …

When we multiply the prediction (Gaussian) with the observation model (Gaussian) we get …

… a product of two Gaussians

\[ \mu = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_2^2 + \sigma_1^2} \]

\[ \sigma = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \]

applied to the filtering equation…
\[
P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t}) dx_t
\]

- Mean: \(z_1\)
- Variance: \(\sigma_q\)

- Mean: \(\hat{x}_1\)
- Variance: \(\sigma_1\)

**New Mean:**
\[
\hat{x}_1^+ = \frac{\hat{x}_1 \sigma_q^2 + z_1 \sigma_1^2}{\sigma_q^2 + \sigma_1^2}
\]

**New Variance:**
\[
\hat{\sigma}_1^2 = \frac{\sigma_q^2 \sigma_1^2}{\sigma_q^2 + \sigma_1^2}
\]

'plus' sign means post 'update' estimate
With a little algebra…

\[
\hat{x}_1^+ = \frac{\hat{x}_1 \sigma_q^2 + z_1 \sigma_1^2}{\sigma_q^2 + \sigma_1^2} = \hat{x}_1 \frac{\sigma_q^2}{\sigma_q^2 + \sigma_1^2} + z_1 \frac{\sigma_1^2}{\sigma_q^2 + \sigma_1^2}
\]

We get a weighted state estimate correction!
Kalman gain notation

With a little algebra...

\[
\hat{x}_1^+ = \hat{x}_1 + \frac{\sigma_1^2}{\sigma_q^2 + \sigma_1^2} (z_1 - \hat{x}_1) = \hat{x}_1 + K(z_1 - \hat{x}_1)
\]

‘Kalman gain’  ‘Innovation’

With a little algebra...

\[
\sigma_1^+ = \frac{\sigma_1^2 \sigma_q^2}{\sigma_1^2 + \sigma_q^2} = \left(1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2}\right) \sigma_1^2 = (1 - K) \sigma_1^2
\]
Summary (1D Kalman Filtering)

To solve this...

\[ P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \int_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})dx_t \]

Compute this...

\[ \hat{x}_1^+ = \hat{x}_1 + \frac{\sigma^2_1}{\sigma^2_1 + \sigma^2_q} (z_1 - \hat{x}_1) \quad \sigma^2_{1+} = \sigma^2_1 - \frac{\sigma^2_1}{\sigma^2_1 + \sigma^2_q} \sigma^2_1 \]

\[ K = \frac{\sigma^2_1}{\sigma^2_1 + \sigma^2_q} \]

‘Kalman gain’

\[ \hat{x}_1^+ = \hat{x}_1 + K(z_1 - \hat{x}_1) \quad \sigma^2_{1+} = \sigma^2_1 - K\sigma^2_1 \]

mean of the new Gaussian

variance of the new Gaussian
Simple 1D Implementation

\[ \begin{bmatrix} x \\ p \end{bmatrix} = KF(x,v,z) \]

\[ x = x + s; \]
\[ v = v + q; \]

\[ K = v/(v + r); \]

\[ x = x + K * (z - x); \]
\[ p = v - K * v; \]

Just 5 lines of code!
or just 2 lines

\[
\begin{align*}
[x & \ P] &= \text{KF}(x, v, z) \\
x &= (x+s) + (v+q) / ((v+q) + r) \times (z - (x+s)) \\
p &= (v+q) - (v+q) / ((v+q) + r) \times v
\end{align*}
\]
Bare computations (algorithm) of Bayesian filtering:

\[
\text{KalmanFilter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)
\]

\[
\bar{\mu}_t = A_t \mu_{t-1} + Bu_t \\
\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R
\]

\[
K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}
\]

\[
\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)
\]

\[
\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t
\]
Simple Multi-dimensional Implementation
(also 5 lines of code!)

\[
[x \ P] = KF(x, P, z)
\]

\[
x = A*x;
\]
\[
P = A*P*A' + Q;
\]

\[
K = P*C' / (C*P*C' + R);
\]

\[
x = x + K*(z - C*x);
\]
\[
P = (eye(size(K,1)) - K*C)*P;
\]
2D Example
Constant position Motion Model

\[ x_t = Ax_{t-1} + Bu_t + \epsilon_t \]
Constant position Motion Model

\[ x_t = Ax_{t-1} + Bu_t + \epsilon_t \]

System noise

\[ \epsilon_t \sim \mathcal{N}(0, R) \]

State measurement

\[
\begin{align*}
x &= \begin{bmatrix} x \\ y \end{bmatrix} \\
z &= \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]
Measurement Model

\[ z_t = C_t x_t + \delta_t \]
$x$  

state  

$z$  

measurement  

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

$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}$  

Measurement Model  

$\mathbf{z}_t = \mathbf{C}_t \mathbf{x}_t + \delta_t$  

zero-mean measurement noise  

$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  

$\delta_t \sim \mathcal{N}(0, Q)$  

$Q = \begin{bmatrix} \sigma^2_q & 0 \\ 0 & \sigma^2_q \end{bmatrix}$
Algorithm for the 2D object tracking example

General Case

\[
\begin{align*}
\bar{\mu}_t &= A_t \mu_{t-1} + Bu_t \\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R \\
K_t &= \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
\mu_t &= \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\
\Sigma_t &= (I - K_t C_t) \bar{\Sigma}_t
\end{align*}
\]

Constant position Model

\[
\begin{align*}
\bar{x}_t &= x_{t-1} \\
\bar{\Sigma}_t &= \Sigma_{t-1} + R \\
K_t &= \bar{\Sigma}_t (\bar{\Sigma}_t + Q)^{-1} \\
x_t &= \bar{x}_t + K_t (z_t - \bar{x}_t) \\
\Sigma_t &= (I - K_t) \bar{\Sigma}_t
\end{align*}
\]

\[
A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

motion model

observation model
Just 4 lines of code

\[
[x \ P] = KF\_constPos(x, P, z)
\]

\[
P = P + Q;
\]

\[
K = P / (P + R);
\]

\[
x = x + K * (z - x);
\]

\[
P = (eye(size(K,1)) - K) * P;
\]

Where did the 5th line go?
General Case

\[
\begin{align*}
\bar{\mu}_t &= A_t \mu_{t-1} + B u_t \\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^\top + R \\
K_t &= \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1} \\
\mu_t &= \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\
\Sigma_t &= (I - K_t C_t) \bar{\Sigma}_t
\end{align*}
\]

Constant position Model

\[
\begin{align*}
\bar{x}_t &= x_{t-1} \\
\bar{\Sigma}_t &= \Sigma_{t-1} + R \\
K_t &= \bar{\Sigma}_t (\bar{\Sigma}_t + Q)^{-1} \\
x_t &= \bar{x}_t + K_t (z_t - \bar{x}_t) \\
\Sigma_t &= (I - K_t) \bar{\Sigma}_t
\end{align*}
\]
Extended Kalman filter
Motion model of the Kalman filter is linear

\[ x_t = A x_{t-1} + B u_t + \epsilon_t \]

but motion is not always linear
Can we use the Kalman Filter?

(motion model and observation model are linear)
Visualizing **non-linear** models

1D motion model example

Output: **NOT** Gaussian

\[ x_t = g(x_{t-1}) \]

Input: Gaussian **(Belief)**

Can we use the Kalman Filter?

*(motion model is not linear)*
How do you deal with non-linear models?

1D motion model example

Input:
Gaussian

Output:
NOT
Gaussian

$p(x_t)$

$x_t = g(x_{t-1})$

$p(x_{t-1})$
How do you deal with non-linear models?

When does this trick work?
Extended Kalman Filter

- Does not assume linear Gaussian models
- Assumes Gaussian noise
- Uses local linear approximations of model to keep the efficiency of the KF framework

Kalman Filter

- Linear motion model
  \[ x_t = Ax_{t-1} + Bu_t + \epsilon_t \]
- Linear sensor model
  \[ z_t = C_t x_t + \delta_t \]

Extended Kalman Filter

- Non-linear motion model
  \[ x_t = g(x_{t-1}, u_t) + \epsilon_t \]
- Non-linear sensor model
  \[ z_t = H(x_t) + \delta_t \]
Motion model linearization

\[ g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]

Taylor series expansion
Motion model linearization

\[ g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1}) \]

\[ \approx g(\mu_{t-1}, u_t) + G_t (x_{t-1} - \mu_{t-1}) \]

What's this called?
Motion model linearization

\[ g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]

\[ \approx g(\mu_{t-1}, u_t) + G_t \quad (x_{t-1} - \mu_{t-1}) \]

Jacobian Matrix

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?

What's this called?
Motion model linearization

\[ g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \]

\[ \approx g(\mu_{t-1}, u_t) + G_t (x_{t-1} - \mu_{t-1}) \]

Jacobian Matrix

‘the rate of change in x’
‘slope of the function’

Sensor model linearization

\[ h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_{t-1} - \bar{\mu}_t) \]

\[ \approx h(\bar{\mu}_t) + H_t (x_t - \bar{\mu}_t) \]
New EKF Algorithm
(pretty much the same)

Kalman Filter

\[ \bar{\mu}_t = A_t \mu_{t-1} + B u_t \]
\[ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R \]
\[ K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1} \]
\[ \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \]
\[ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \]

Extended KF

\[ \bar{\mu}_t = g(\mu_{t-1}, u_t) \]
\[ \bar{\Sigma}_t = G_t \bar{\Sigma}_{t-1} G_t^\top + R \]
\[ K_t = \bar{\Sigma}_t H_t^\top (H_t \bar{\Sigma}_t H_t^\top + Q)^{-1} \]
\[ \mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t)) \]
\[ \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \]
2D example
state: position-velocity

\[ x = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \]

constant velocity motion model

\[ A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 \end{bmatrix} \]

Motion model is linear but …
**measurement**: range-bearing

\[
\begin{bmatrix}
  r \\
  \theta
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  \sqrt{x^2 + y^2} \\
  \tan^{-1}(y/x)
\end{bmatrix}
\]

**measurement model**

Is the measurement model linear?

\[
z = h(r, \theta)
\]

with additive Gaussian noise
measurement: range-bearing

\[ z = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix} \]

measurement model

Is the measurement model linear?

\[ z = h(r, \theta) \]

with additive Gaussian noise

non-linear!

What should we do?
linearize the observation/measurement model!

\[ z = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix} \]

What is the Jacobian?

\[ H = \frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \end{bmatrix} \]
Jacobian used in the Taylor series expansion looks like …

\[
H = \begin{bmatrix}
\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial y} \\
\frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial y}
\end{bmatrix}
= \begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) & 0 \\
-sin(\theta)/r & 0 & \cos(\theta)/r & 0
\end{bmatrix}
\]
\[ [x \ P] = \text{EKF}(x, P, z, dt) \]

\[
\begin{align*}
    r &= \sqrt{x(1)^2 + x(3)^2}; \\
    b &= \text{atan2}(x(3), x(1)); \\
    y &= [r; b]; \\
    H &= \begin{bmatrix} \cos(b) & 0 & \sin(b) & 0; \\ -\sin(b)/r & 0 & \cos(b)/r & 0 \end{bmatrix};
\end{align*}
\]

\[
\begin{align*}
    x &= Fx; \\
    P &= FPF' + Q; \\
    K &= P\ H' / (H\ P\ H' + R); \\
    x &= x + K(z - y); \\
    P &= (\text{eye}(\text{size}(K,1)) - K\ H) \ P;
\end{align*}
\]

**Parameters:**
- \( Q = \text{diag}([0 \ 0.1 \ 0 \ 0.1]); \)
- \( R = \text{diag}([50^2 \ 0.005^2]); \)
- \( F = \begin{bmatrix} 1 & dt & 0 & 0; \\ 0 & 1 & 0 & 0; \\ 0 & 0 & 1 & dt; \\ 0 & 0 & 0 & 1 \end{bmatrix}; \)

*extra computation for the EKF measurement model Jacobian*
Problems with EKFs

Taylor series expansion = poor approximation of non-linear functions
success of linearization depends on limited uncertainty and amount of
local non-linearity

Computing partial derivatives is a pain

Drifts when linearization is a bad approximation

Cannot handle multi-modal (multi-hypothesis) distributions
SLAM
MonoSLAM: Real-Time Single Camera SLAM

Andrew J. Davison, Ian D. Reid, Member, IEEE, Nicholas D. Molton, and Olivier Stasse, Member, IEEE

Abstract—We present a real-time algorithm which can recover the 3D trajectory of a monocular camera, moving rapidly through a previously unknown scene. Our system, which we dub MonoSLAM, is the first successful application of the SLAM methodology from mobile robotics to the “pure vision” domain of a single uncontrolled camera, achieving real time but drift-free performance inaccessible to Structure from Motion approaches. The core of the approach is the online creation of a sparse but persistent map of natural landmarks within a probabilistic framework. Our key novel contributions include an active approach to mapping and measurement, the use of a general motion model for smooth camera movement, and solutions for monocular feature initialization and feature orientation estimation. Together, these add up to an extremely efficient and robust algorithm which runs at 30 Hz with standard PC and camera hardware. This work extends the range of robotic systems in which SLAM can be usefully applied, but also opens up new areas. We present applications of MonoSLAM to real-time 3D localization and mapping for a high-performance full-size humanoid robot and live augmented reality with a hand-held camera.

Index Terms—Autonomous vehicles, 3D/stereo scene analysis, tracking.
Simultaneous Localization and Mapping

Given a **single camera** feed, estimate the 3D **position of the camera** and the 3D **positions of all landmark** points in the world.
Real-Time Camera Tracking in Unknown Scenes
MonoSLAM is just EKF!

\[ P(x_t|z_{1:t}) \propto P(z_t|x_t) \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1} \]

**Step 1: Prediction**

\[ P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1} \]

**Step 2: Update:**

\[ P(x_t|z_{1:t}) = P(z_t|x_t)P(x_t|z_{1:t-1}) \]
MonoSLAM is just EKF!

\[
P(x_t|z_{1:t}) \propto P(z_t|x_t) \int_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|z_{1:t-1}) \, dx_{t-1}
\]

What is the state representation?

Step 1: Prediction

\[
P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|z_{1:t-1}) \, dx_{t-1}
\]

Step 2: Update:

\[
P(x_t|z_{1:t}) = P(z_t|x_t)P(x_t|z_{1:t-1})
\]
What is the camera (robot) state?

\[
x_c = \begin{bmatrix}
    r \\
    q \\
    v \\
    \omega
\end{bmatrix}
\]

- position
- rotation (quaternion)
- velocity
- angular velocity

13 total
What is the camera (robot) state?

\[ x_c = \begin{bmatrix}
  r \\
  q \\
  v \\
  \omega
\end{bmatrix} \]

- **position**: 3 dimensions
- **rotation (quaternion)**: 4 dimensions
- **velocity**: 3 dimensions
- **angular velocity**: 3 dimensions

**13 total**

What are the dimensions?
What is the world (robot+environment) state?

\[ x = \begin{bmatrix} x_c \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \]

- state of the camera
- location of feature 1
- location of feature 2
- location of feature N
What is the world (robot+environment) state?

\[ x = \begin{bmatrix} x_c \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \]

- state of the camera: 13
- location of feature 1: 3
- location of feature 2: 3
- location of feature N: 3

13 + 3N total
What is the covariance (uncertainty) of the world state?

\[
\Sigma = \begin{bmatrix}
\Sigma_{x_c x_c} & \Sigma_{x_c y_1} & \cdots & \Sigma_{x_c y_N} \\
\Sigma_{y_1 x_c} & \Sigma_{y_1 y_1} & \cdots & \Sigma_{y_1 y_N} \\
\vdots & \vdots & \ddots & \vdots \\
\Sigma_{y_N x_c} & \Sigma_{y_N y_1} & \cdots & \Sigma_{y_N y_N}
\end{bmatrix}
\]

What are the dimensions?

\((13+3N) \times (13+3N)\)
MonoSLAM is just EKF!

\[
P(x_t|z_{1:t}) \propto P(z_t|x_t) \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1}
\]

*What are the observations?*

**Step 1: Prediction**

\[
P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1}
\]

**Step 2: Update:**

\[
P(x_t|z_{1:t}) = P(z_t|x_t)P(x_t|z_{1:t-1})
\]
MonoSLAM is just EKF!

$$P(x_t|z_{1:t}) \propto P(z_t|x_t) \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1}$$

**What are the observations?**

**Step 1: Prediction**

$$P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1}$$

**Step 2: Update:**

$$P(x_t|z_{1:t}) = P(z_t|x_t)P(x_t|z_{1:t-1})$$
Observations are...

detected visual features of landmark points.
(e.g., Harris corners)
MonoSLAM is just EKF!

\[ P(x_t | z_{1:t}) \propto P(z_t | x_t) \int_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | z_{1:t-1}) dx_{t-1} \]

**Step 1: Prediction**

\[ P(x_t | z_{1:t-1}) = \int_{x_{t-1}} P(x_t | x_{t-1})P(x_{t-1} | z_{1:t-1}) dx_{t-1} \]

**What does the prediction step look like?**

**Step 2: Update:**

\[ P(x_t | z_{1:t}) = P(z_t | x_t)P(x_t | z_{1:t-1}) \]
What is the motion model? \[ P(x_t|x_{t-1}) \]

What is the form of the belief? \[ P(x_t|z_{1:t-1}) \]
What is the motion model?

**Landmarks:**
constant position
(identity matrix)

**Camera:**
constant velocity
(not identity matrix and non-linear)

What is the form of the belief?

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

\[ P(x_t | z_{1:t-1}) \]
What is the motion model? $P(x_t|x_{t-1})$

Landmarks:
constant position
(identity matrix)

Camera:
constant velocity
(not identity matrix and non-linear)

What is the form of the belief? $P(x_t|z_{1:t-1})$

Gaussian!
(everything will be parametrized by a mean and variance)
Constant Velocity Motion Model

\[ r_t = r_{t-1} + v_{t-1} \Delta t \]

\[ q_t = q_{t-1} \times [q(\omega) \Delta t] \]

\[ v_t = v_{t-1} \]

\[ \omega_t = \omega_{t-1} \]
Gaussian noise uncertainty (only on velocity)

\[ \mathbf{v}_t = \mathbf{v}_{t-1} + \mathbf{V} \]

\[ \omega_t = \omega_{t-1} + \Omega \]

\[ \mathbf{V} \sim \mathcal{N}(0, \begin{bmatrix} \sigma_v & 0 & 0 \\ 0 & \sigma_v & 0 \\ 0 & 0 & \sigma_v \end{bmatrix}) \]

\[ \Omega \sim \mathcal{N}(0, \begin{bmatrix} \sigma_w & 0 & 0 \\ 0 & \sigma_w & 0 \\ 0 & 0 & \sigma_w \end{bmatrix}) \]
Prediction (mean of camera state):

\[ P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1} \]

\[
\mathbf{f}_t = \begin{bmatrix} r_t \\ q_t \\ v_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{t-1} + \mathbf{v}_{t-1}\Delta t \\ \mathbf{q}_{t-1} + \mathbf{q}(\omega)_{t-1}\Delta t \\ \mathbf{v}_{t-1} \\ \omega_{t-1} \end{bmatrix}
\]
Prediction (covariance of camera state):

\[ P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1} \]

\[
\tilde{\Sigma}_{xx} = \begin{bmatrix} \frac{\partial f_t}{\partial x} \end{bmatrix} \Sigma_{xx} \begin{bmatrix} \frac{\partial f_t}{\partial x} \end{bmatrix}^\top + Q_t
\]

Where does this motion model approximation come from?
What are the dimensions?
Skipping over many details…

\[
\frac{\partial f_t}{\partial x_{t-1}} = \begin{bmatrix}
I & 0 & I\Delta t & 0 \\
0 & \frac{\partial q_t}{\partial q_{t-1}} & 0 & \frac{\partial \omega_t}{\partial q_{t-1}} \\
0 & 0 & I & 0 \\
0 & 0 & 0 & I
\end{bmatrix}
\]
Prediction (covariance of camera state):

\[
P(x_t | z_{1:t-1}) = \int_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | z_{1:t-1}) dx_{t-1}
\]

\[
\Sigma_{xx} = \left[ \frac{\partial f_t}{\partial x} \right] \Sigma_{xx} \left[ \frac{\partial f_t}{\partial x} \right]^\top + Q_t
\]

new covariance  
change around new state  
old covariance  
change around new state  
system noise (process noise)

Bit of a pain to compute this term…
We just covered the **prediction** step for the camera state

\[
P(x_t|z_{1:t-1}) = \int_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1}) dx_{t-1}
\]

\[
f_t = \begin{bmatrix} r_t \\ q_t \\ v_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} r_{t-1} + v_{t-1} \\ q_{t-1} + q(\omega)_{t-1} \\ v_{t-1} \\ \omega_{t-1} \end{bmatrix}
\]

\[
\Sigma_{xx} = \frac{\partial f_t}{\partial x} \Sigma_{xx} \frac{\partial f_t}{\partial x} \top + Q_t
\]

Now we need to do the **update** step!
General Filtering Equations

\[
P(x_t | z_{1:t}) \propto P(z_t | x_t) \int_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | z_{1:t-1}) \, dx_{t-1}
\]

Prediction:

\[
P(x_t | z_{1:t-1}) = \int_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | z_{1:t-1}) \, dx_{t-1}
\]

Update:

\[
P(x_t | z_{1:t}) = P(z_t | x_t) P(x_t | z_{1:t-1})
\]
Predicted State

What are the observations?

2D projections of 3D landmarks
Recall, the state includes the 3D location of landmarks.

What is the projection from 3D point to 2D image point?

\[
x = \begin{bmatrix}
  x_c \\
  y_1 \\
  y_2 \\
  \vdots \\
  y_N
\end{bmatrix}
\]
Observation Model

\[ P(z_t | x_t) \]

If you know the 3D location of a landmark, what is the 2D projection?

Non-linear observation model

\[ h \sim P_y \]

2D Image Point \hspace{2cm} Camera matrix \hspace{2cm} 3D World Point

\[ P = K [ R | T ] \]

What do we know about \( P \)?

How do we make the observation model linear?
\[ H = \frac{\partial h}{\partial x} \]

\( n \): number of visible points

I will spare you the pain of deriving the partial derivative…
\[ P(x_t | z_{1:t}) = P(z_t | x_t) P(x_t | z_{1:t-1}) \]

**Update step (mean):**

\[
x_t = x_t + K_t(z_t - h(y; x_t))
\]

- **Updated state**
- **Predicted state**
- **Matched 2D features**
- **2D projection of 3D point**

**Update step (covariance):**

\[
\Sigma_t = (I - K_t H_t) \Sigma_t
\]

- **Covariance (updated)**
- **Identity**
- **Kalman gain**
- **Jacobian**
- **Covariance (predicted)**
Kintinuous: Spatially Extended Kinect Fusion

Thomas Whelan, John McDonald
National University of Ireland Maynooth, Ireland

Michael Kaess, Maurice Fallon, Hordur Johannsson, John J. Leonard
Computer Science and Artificial Intelligence Laboratory, MIT, USA
References

Basic reading:
• Szeliski, Appendix B.