Filtering & State Estimation

EEL 4930/5934: Autonomous Robots

Spring 2023

Md Jahidul Islam

Lecture 9

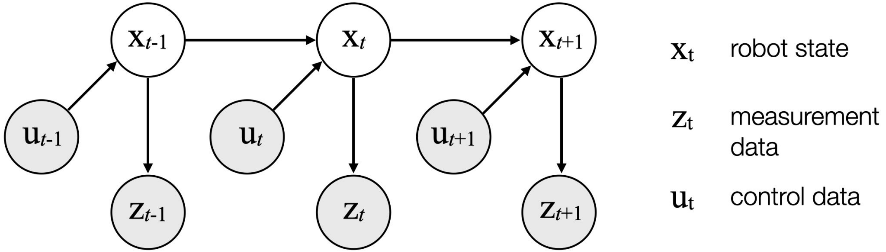
ECE | Florida
Electrical & Computer Engineering
UF | UNIVERSITY of FLORIDA

State Estimation and Filtering

State Estimation

- *A* *state* *is the collection of all aspects about a robot and the environment that we need to track*
  - Robot pose and motion information
  - Positions of: landmarks, static obstacles
  - Motion estimates of dynamic obstacles

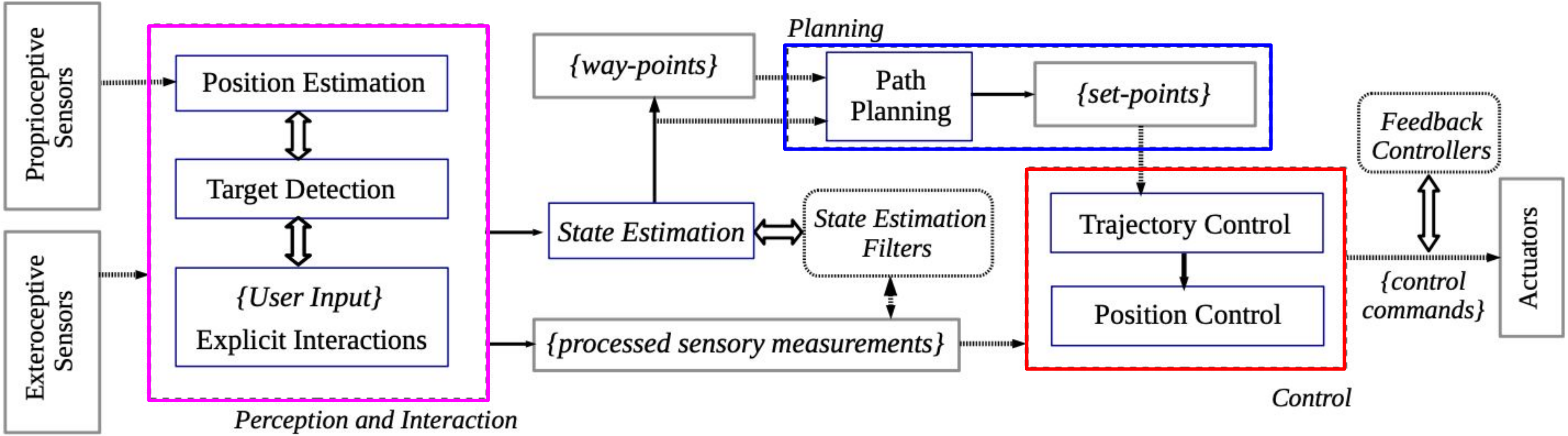
- State is represented by a vector (numerically)
  - Used for planning and control
  - Tracked and updated regularly



What is Filtering?

- *State estimation of* *dynamic systems* *from a series of noisy sensory measurements*
- <u>Examples:</u>
  - Kalman filters (EF, EKF, UKF)
  - Particle filters

Perception–Planning–Control



<u>Pipeline Example:</u> autonomous target following by mobile robots

The SLAM Problem

SLAM: Simultaneous Localization and Mapping

- Estimating robot pose and mapping the environment simultaneously

- <u>We briefly covered:</u>
    - Visual SLAM
    - LiDAR-based SLAM

⇒ ***Given:***

- *The robots controls*

  $$u_{1:T} = \{u_1, u_2, u_3 \ldots, u_T\}$$

- *The measurements / observations*

  $$z_{1:T} = \{z_1, z_2, z_3 \ldots, z_T\}$$

⇒ ***Wanted:***

- *The environment map*

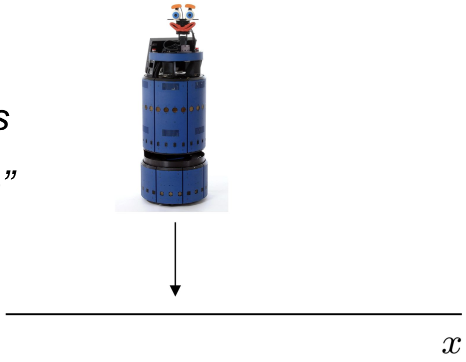
  $$m$$

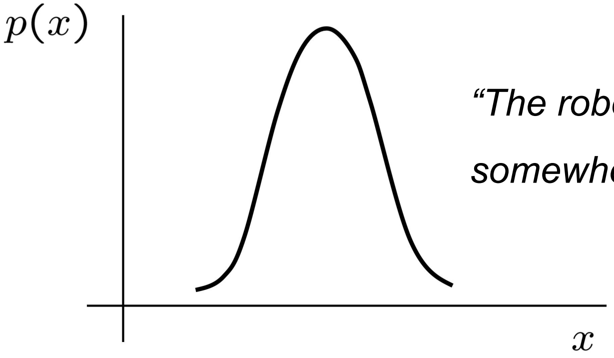
- *The robot pose*

  $$x_{0:T} = \{x_0, x_1, x_2 \ldots, x_T\}$$

Probabilistic Representation

"The robot is exactly here"



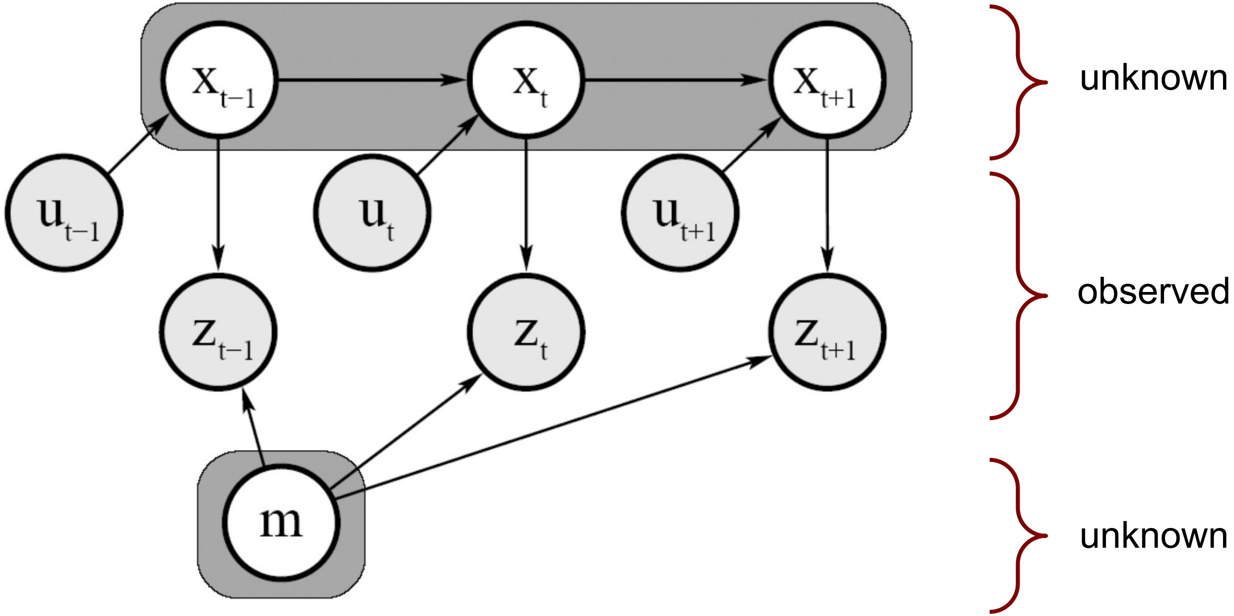


"The robot is somewhere here"

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution path map given observations controls

Graphical Representation



$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

Dr. Cyrill Stachniss

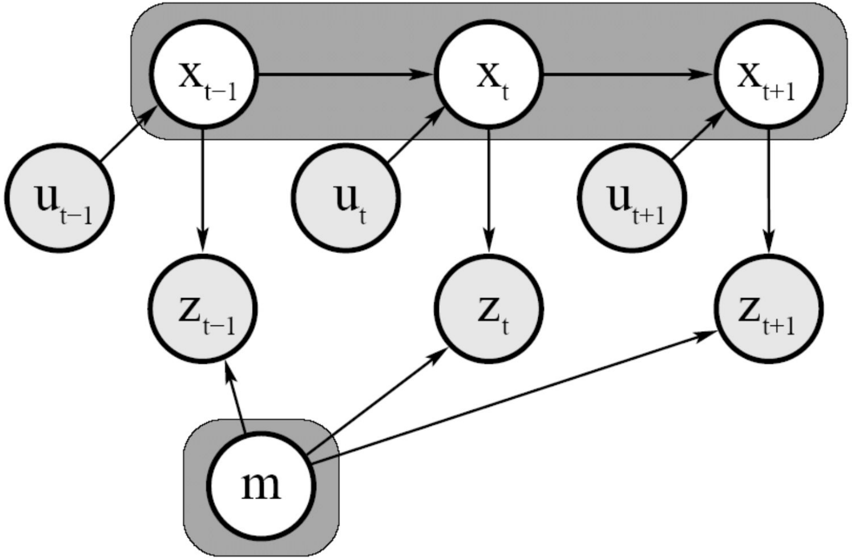
Full SLAM vs Online SLAM

⇒ **Full SLAM estimates the entire pose (state)**

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

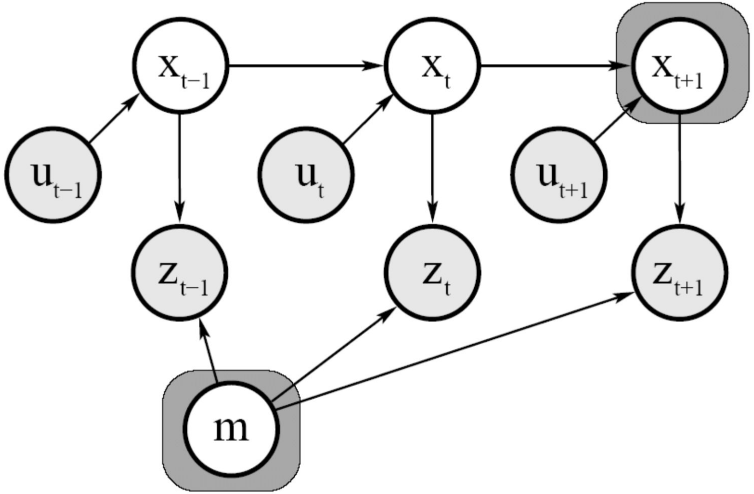
⇒ **Online SLAM estimates the most recent pose (state)**

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$



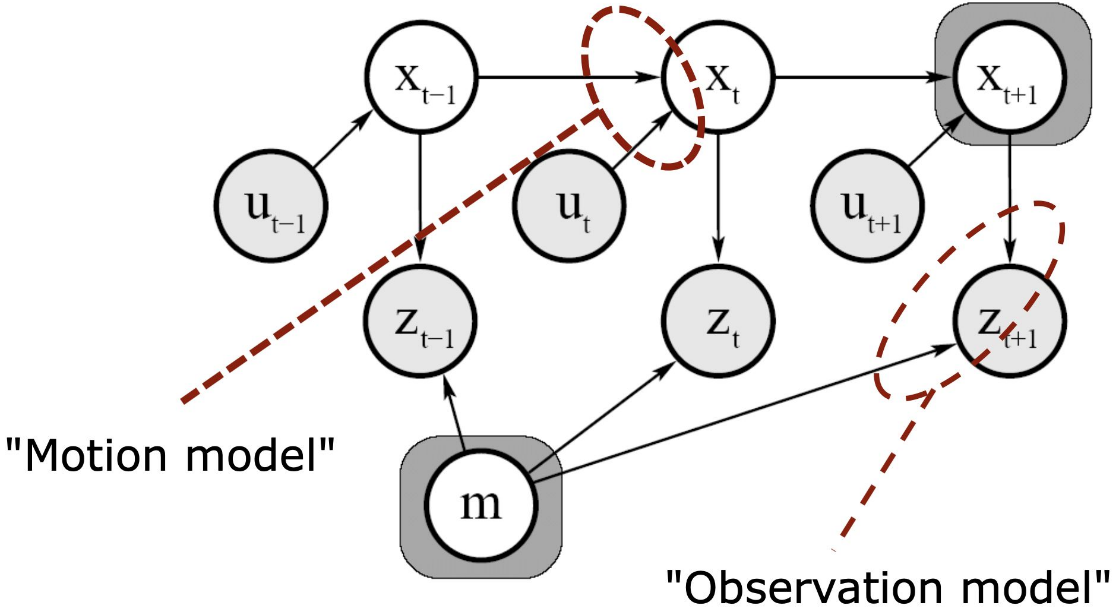
Ref: Probabilistic Robotics (Chapter 7, 10)

Graphical Model: Online SLAM



$$p(x_{t+1}, m \mid z_{1:t+1}, u_{1:t+1}) = \int_{x_0} \cdots \int_{x_t} p(x_{0:t+1}, m \mid z_{1:t+1}, u_{1:t+1}) \, dx_t \ \ldots \ dx_0$$

Motion and Observation Model

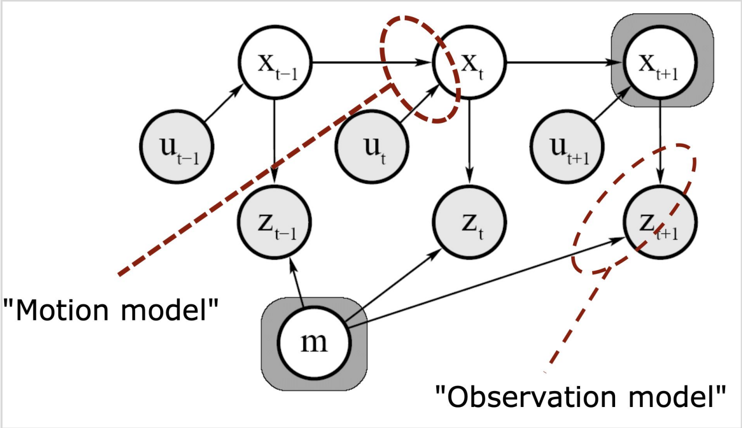


Motion Model

$$p(x_t \mid x_{t-1}, u_t)$$

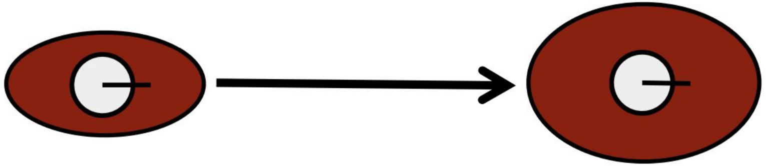
distribution new pose given old pose control

Robot motion 'rules' given current state & control input

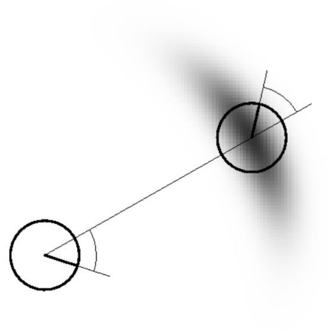


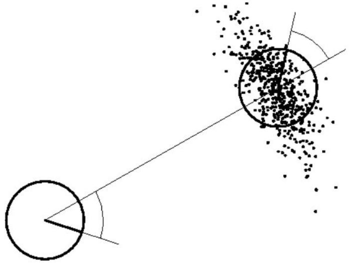
Motion Model Examples

⇒ **Gaussian model**



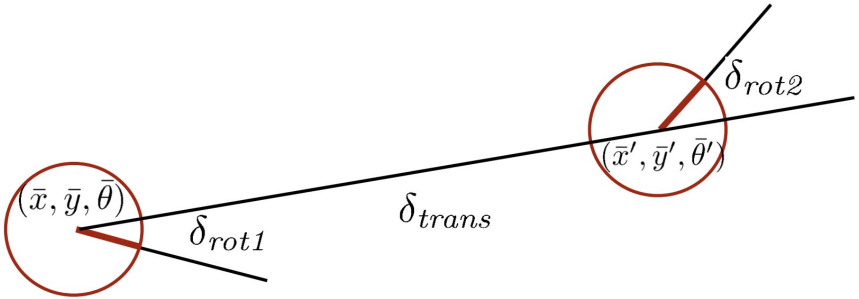
⇒ **Non-Gaussian model**





Odometry Model: 2D Robots

- ⇒ **Robot moves from** $(\bar{x}, \bar{y}, \bar{\theta})$ **to** $(\bar{x}', \bar{y}', \bar{\theta}')$
- ⇒ **Odometry information** $u = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$



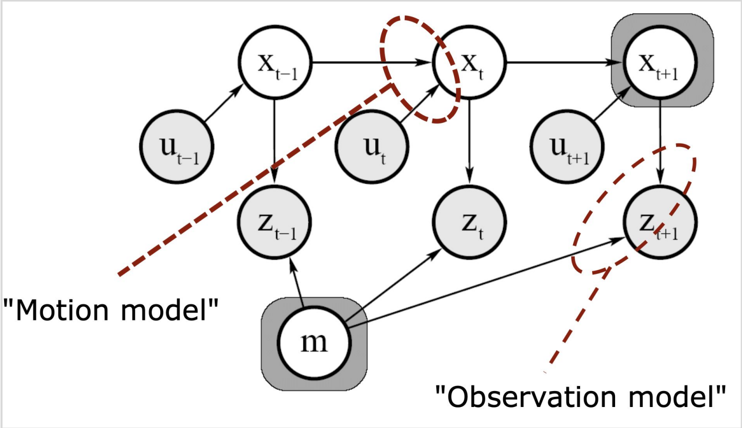
$$
\begin{aligned}
\delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\
\delta_{rot1} &= \operatorname{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\
\delta_{rot2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot1}
\end{aligned}
$$

Observation (Sensor) Model

$$p(z_t \mid x_t)$$

distribution observation given pose

Beliefs of sensor measurements given robot's state



State Estimation: Bayesian Filter

Problem formulation

- Belief = Possible state
- x_t = State (numerical vector)
- z_t = Measurement data (from sensors)
- u_t = Control input (from user / autopilot)

Set $bel(x_t)$ at time $t=0$

For $t = 1, 2, \ldots$

- **Predict** current belief

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

- **Update** belief based on sensory observation

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \bar{bel}(x_{t-1})$$

$$p(x|z) = \frac{p(z|x)\, p(x)}{p(z)} \propto p(z|x)\, p(x)$$

Bayes Rule

$p(x)$ = *Prior probability distribution*

$p(z)$ = *Measurement data distribution*

$p(x|z)$ = *Posterior probability distribution*

State Estimation: Bayesian Filter

Problem formulation

- belief = Possible state
- x_t = State (numerical vector)
- z_t = Measurement data (from sensors)
- u_t = Control input (from user / autopilot)

Set $bel(x_t)$ at time $t=0$

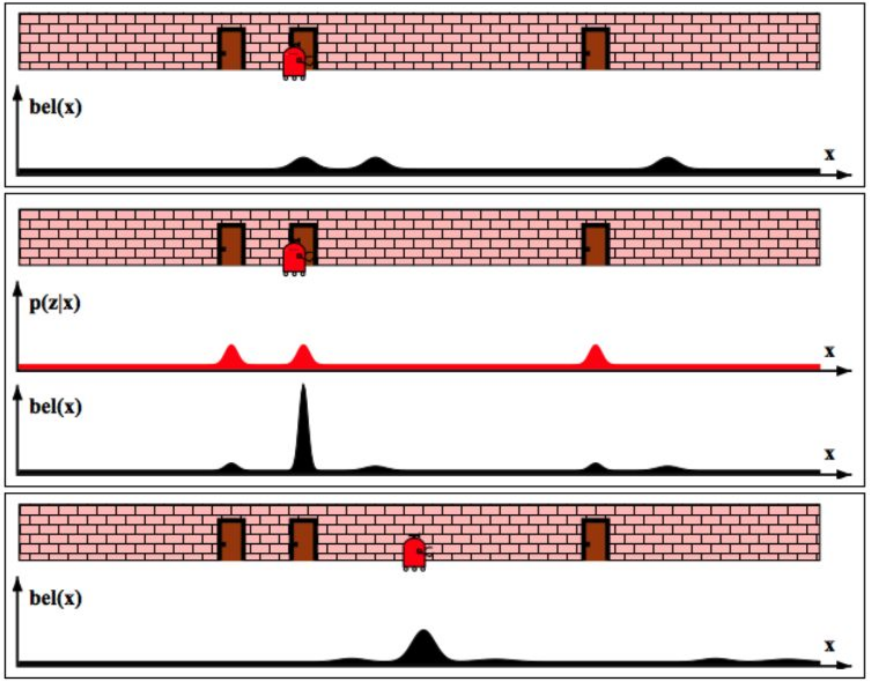
For $t = 1, 2, ...$

- **Predict** current belief

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\, dx_{t-1}$$

- **Update** belief based on sensory observation

$$bel(x_t) = \eta\, p(z_t \mid x_t)\ \bar{bel}(x_{t-1})$$



Dr. Allen

State Estimation: Kalman Filter

Kalman Filter (KF)

- A Bayes filter
- Gives optimal solution for linear models and for Gaussian distributions
- Popular versions:
    - Extended Kalman Filter (EKF)
    - Unscented Kalman Filter (UKF)

KF Assumptions:

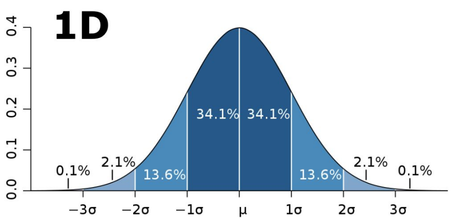
- A linear state transition model

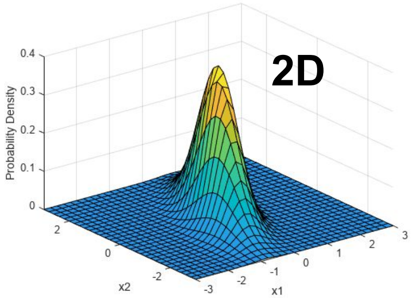
$$x_t = \mathbf{F}_t x_{t-1} + \mathbf{B}_t u_t + \epsilon_t$$

- A linear observation model

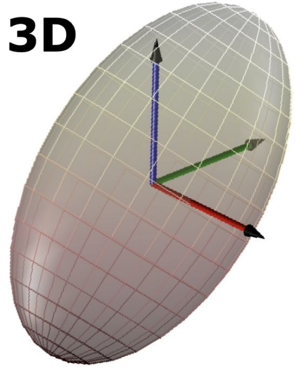
$$z_t = \mathbf{H}_t x_t + \delta_t$$

- Zero mean Gaussian noise





KF assumes a Gaussian world!

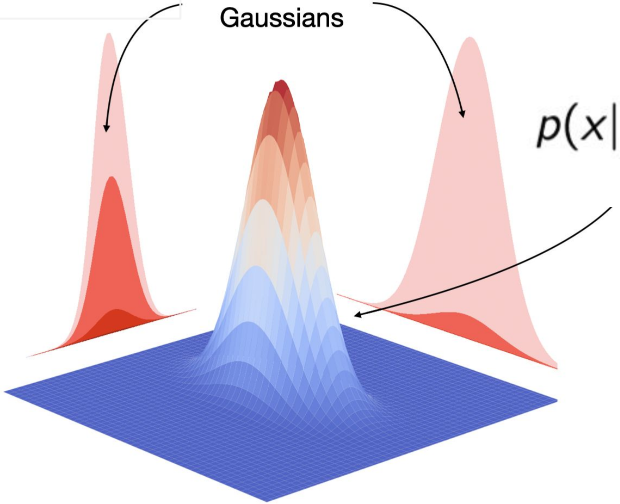


Gaussian Distributions

1D Gaussian: *p(x | μ, σ²)*

- x = 1D scalar
- μ = 1D mean
- σ^2 = variance

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\frac{(x-u)^2}{\sigma^2}}$$



$$p(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} e^{-\frac{1}{2}(x-u)^T\Sigma^{-1}(x-u)}$$

Multivariate Gaussian: *p(x | μ, Σ)*

- x = n dimensional state vector
- μ = n dimensional mean vector

  μ = **E**[x]

- Σ = n x n covariance matrix

  Σ = **E**[(x-μ)(x-μ)^T]

Kalman Filtering

State transition: $x_t = \mathbf{F}_t\, x_{t-1} + \mathbf{B}_t\, u_t + \epsilon_t$

Observation: $z_t = \mathbf{H}_t\, x_t + \delta_t$

KF Assumptions:

- A linear state transition model
- A linear observation model
- Zero mean Gaussian noise model
  - $\boldsymbol{Q_t}$ describes the process/motion noise
  - $\boldsymbol{R_t}$ describes the measurement noise

Ref: Probabilistic Robotics (Chapter 2, 3)

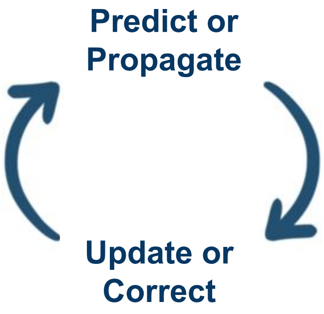
$\boldsymbol{F_t}$ Matrix $(n \times n)$ that describes how the state evolves from $t-1$ to t without controls or noise.

$\boldsymbol{B_t}$ Matrix $(n \times l)$ that describes how the control u_t changes the state from $t-1$ to t.

$\boldsymbol{H_t}$ Matrix $(k \times n)$ 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 $\boldsymbol{Q_t}$ and $\boldsymbol{R_t}$ respectively.

Kalman Filter Algorithm



Predict / Propagate

$$\text{Predict State:} \quad x_t = \mathbf{F}_t \, x_{t-1} + \mathbf{B}_t \, u_t$$

$$\text{Predict covariance:} \quad \mathbf{P}_t = \mathbf{F}_t \, \mathbf{P}_{t-1} \, \mathbf{F}_\mathbf{t}^\mathbf{T} + \mathbf{Q}_t$$

Update / Correct

$$\text{Observation residual:} \quad y_t = z_t - \mathbf{H}_t \, x_{t-1}$$

$$\text{Observation covariance:} \quad \mathbf{S}_t = \mathbf{H}_t \, \mathbf{P}_{t-1} \, \mathbf{H}_\mathbf{t}^\mathbf{T} + \mathbf{R}_t$$

$$\text{Kalman gain:} \quad \mathbf{K}_t = \mathbf{P}_t \, \mathbf{H}_\mathbf{t-1}^\mathbf{T} \, \mathbf{S}_\mathbf{t}^{-1}$$

$$\text{Update state:} \quad x_t = x_t + \mathbf{K}_t \, y_t$$

$$\text{Update covariance:} \quad \mathbf{P}_t = (\mathbf{I}_t - \mathbf{K}_t \, \mathbf{H}_\mathbf{t}) \, \mathbf{P}_t$$

Derivation: Probabilistic Robotics (Ch 3.2)

Simple Derivation: 1D KF

$$p(x|z) = \frac{p(z|x)\,p(x)}{p(z)} \propto p(z|x)\,p(x)$$

x = state value; $p(x) \sim N(\mu_x, \sigma_x^2)$
z = measurement; $p(z \mid x) \sim N(x, \sigma_z^2)$

MAP (maximum a posteriori) estimate

$$\begin{aligned} x^* &= \operatorname{argmax} p(x|z) \\ &= \operatorname{argmax} \log p(x|z) \end{aligned}$$

$$\begin{aligned} \log p(x|z) &= \log p(z|x) + \log p(x) - \log p(z) \\ &= \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(z-x)^2}{\sigma_z^2}} + \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu_x)^2}{\sigma_x^2}} - \log p(z) \\ &= -\frac{1}{2}\frac{(z-x)^2}{\sigma_z^2} - \frac{1}{2}\frac{(x-\mu_x)^2}{\sigma_x^2} + constants \end{aligned}$$

$$\frac{\partial \log p(x|z)}{\partial x} = 0 \rightarrow \boxed{\frac{z-x}{\sigma_z^2} + \frac{x-\mu_x}{\sigma_x^2} = 0}$$

$$\boxed{x^* = \mu_x + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} \cdot (z - \mu_z)}$$

Optimal State =

<u>Current belief</u>: The mean

+

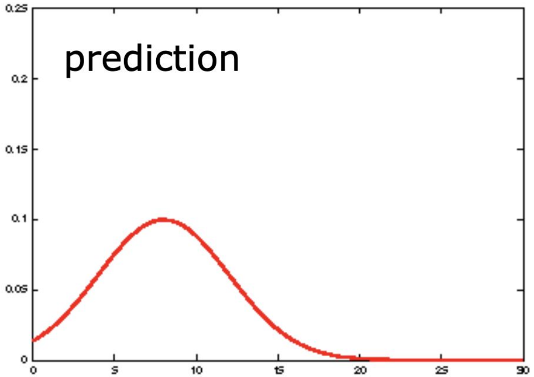
<u>Weight</u>: Kalman gain!

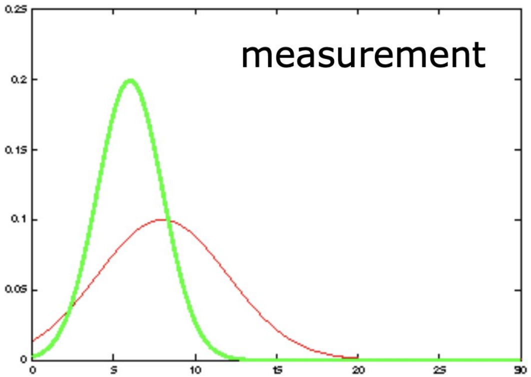
*****

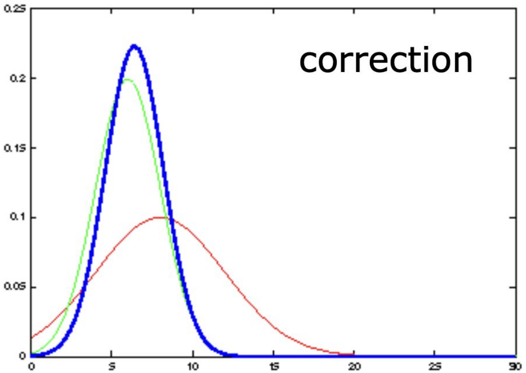
Measurement residual

*The **Kalman gain** determines how much your estimate needs to change given a measurement*
- *How erroneous is your process?*
- *How erroneous is your measurement?*

Example #1: 1D KF







It's a weighted mean!

General KF

$$\text{State transition:} \quad x_t = \mathbf{F}_t \, x_{t-1} + \mathbf{B}_t \, u_t + \epsilon_t$$

$$\text{Observation:} \quad z_t = \mathbf{H}_t \, x_t + \delta_t$$

Predict or Propagate

Update or Correct

$$\text{Predict State:} \quad x_t = \mathbf{F}_t \, x_{t-1} + \mathbf{B}_t \, u_t$$

$$\text{Predict covariance:} \quad \mathbf{P}_t = \mathbf{F}_t \, \mathbf{P}_{t-1} \, \mathbf{F}_\mathbf{t}^\mathbf{T} + \mathbf{Q}_t$$

$$\text{Observation residual:} \quad y_t = z_t - \mathbf{H}_t \, x_{t-1}$$

$$\text{Observation covariance:} \quad \mathbf{S}_t = \mathbf{H}_t \, \mathbf{P}_{t-1} \, \mathbf{H}_\mathbf{t}^\mathbf{T} + \mathbf{R}_t$$

$$\text{Kalman gain:} \quad \mathbf{K}_t = \mathbf{P}_t \, \mathbf{H}_{\mathbf{t-1}}^\mathbf{T} \, \mathbf{S}_\mathbf{t}^{-1}$$

$$\text{Update state:} \quad x_t = x_t + \mathbf{K}_t \, y_t$$

$$\text{Update covariance:} \quad \mathbf{P}_t = (\mathbf{I}_t - \mathbf{K}_t \, \mathbf{H}_\mathbf{t}) \, \mathbf{P}_t$$

MAP vs MLE Estimation

$$p(x|z) = \frac{p(z|x)\,p(x)}{p(z)} \propto p(z|x)\,p(x)$$

x = state value; $p(x) \sim N(\mu_x, \sigma_x^2)$
z = measurement; $p(z|x) \sim N(x, \sigma_z^2)$

MAP (maximum a posteriori) estimate

$$\begin{aligned} x^* &= \operatorname{argmax} p(x|z) \\ &= \operatorname{argmax} \log p(x|z) \end{aligned}$$

Finds the Kalman solution!

$$\boxed{x^* = \mu_x + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} \cdot (z - \mu_z)}$$

MLE (maximum likelihood estimator) → ?

$$\begin{aligned} x^* &= \operatorname{argmax} p(z|x) \\ &= \operatorname{argmin} \sum_i (z_i - x)^2 \end{aligned}$$

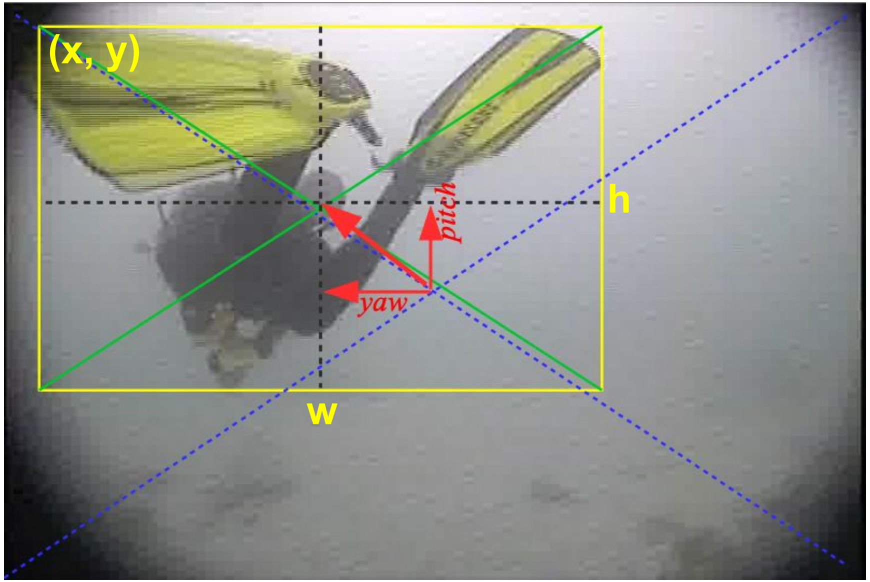
$$\begin{aligned} p(z|x) &= p(z_1, z_2, \cdots, z_t|x) \\ &= p(z_1|x) \cdot p(z_2|x) \cdot p(z_3|x) \cdots p(z_t|x) \\ &= \prod_i \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\frac{(z_i-x)^2}{\sigma_z^2}} \\ &= \left(\frac{1}{2\pi\sigma^2}\right)^{|t|/2} \cdot e^{-\frac{\sum_i (z_i-x)^2}{2\sigma_z^2}} \end{aligned}$$

Finds the least squared solution!

$$\boxed{x^* = \operatorname{argmin} \sum_i (z_i - x)^2}$$

Example #2: Bounding Box Tracker

Remember the BBox-reactive Yaw-Pitch Controller!




```
x_c, y_c = (x + w/2), (y + h/2)
x0, y0 = im_width/2, im_height/2
offset_yaw = (x_c - x0)/im_width
offset_pitch = (y_c - y0)/im_height
```

```
yaw_angle ∝ offset_yaw
pitch_angle ∝ offset_pitch
velocity_forward ∝ distance
```

Detector gives BBox: (x, y, w, h)

- *How to formulate the state, control, and transition relationships?*
- *How to design a KF-based tracker for this?*

BBox Tracker

$$x_t = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ w \\ \dot{w} \\ h \\ \dot{h} \end{bmatrix}$$

The state vector

$$\mathbf{F}_t = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Assume constant velocity!

$$\mathbf{B}_t = \mathbf{0}$$

No control inputs u_t

State transition: $x_t = \mathbf{F}_t x_{t-1} + \epsilon_t$

Observation: $z_t = \mathbf{H}_t x_t + \delta_t$

Predict

```
x = dot(F, x_state)

P = dot(F, P).dot(F.T) + Q
```

Update

```
y = z - dot(H, x) # residual

S = dot(H, P).dot(H.T) + R

K = dot(P, H.T).dot(inv(S)) # gain

x = x + dot(K, y)

P = P - dot(K, H).dot(P)# update P_t

x_state = x.astype(int) # update x_t
```

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

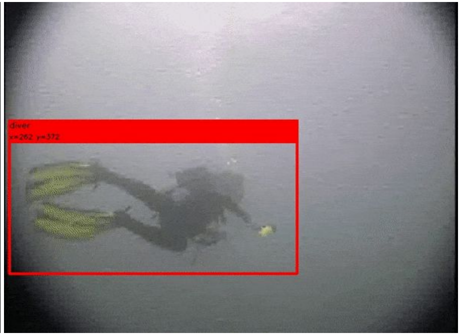
we can only measure the coordinates!

Process covariance $\mathbf{Q}_t$=?

Observation covariance $\mathbf{R}_t$=?

Homework 5C/5D

BBox Tracker



$$\text{State transition:} \quad x_t = \mathbf{F}_t x_{t-1} + \epsilon_t$$

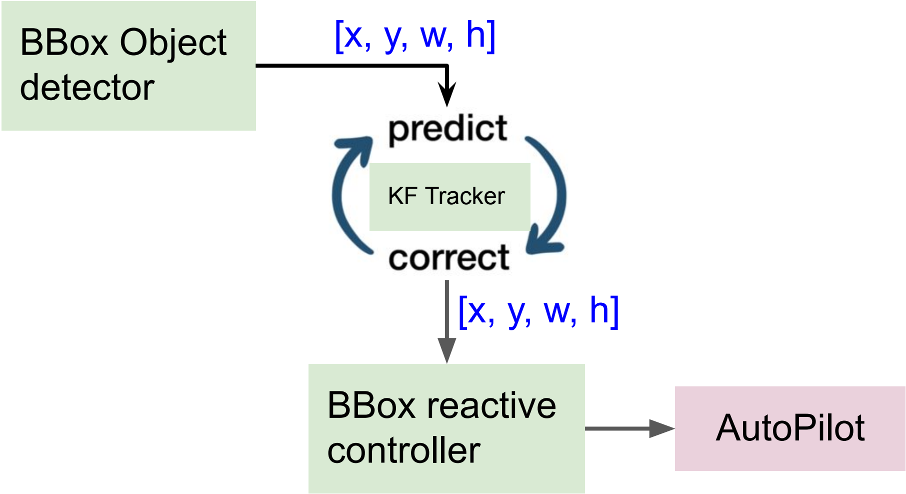
$$\text{Observation:} \quad z_t = \mathbf{H}_t x_t + \delta_t$$

Predict

```
x = dot(F, x_state)

P = dot(F, P).dot(F.T) + Q
```

Update

```
y = z - dot(H, x) # residual

S = dot(H, P).dot(H.T) + R

K = dot(P, H.T).dot(inv(S)) # gain

x = x + dot(K, y)

P = P - dot(K, H).dot(P)# update P_t

x_state = x.astype(int) # update x_t
```



Non-linear Dynamic Systems

⇒ Most realistic problems (in robotics) involve nonlinear functions

⇒ The non-linear functions lead to non-Gaussian distributions

⇒ The non-linear functions lead to non-Gaussian distributions

$$x_t = \mathbf{F}_t x_{t-1} + \mathbf{B}_t u_t + \epsilon_t$$

$$\Downarrow$$

$$\boxed{x_t = g(u_t, x_{t-1}) + \epsilon_t}$$

Solution:
Non-linear versions
- Extended KF (EKF)
- Iterative EKF

$$z_t = \mathbf{H}_t x_t + \delta_t$$

$$\Downarrow$$

$$\boxed{z_t = h(x_t) + \delta_t}$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=:\, G_t} (x_{t-1} - \mu_{t-1})$$

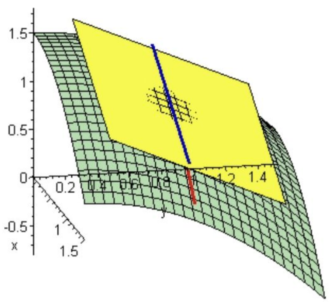
$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=:\, H_t} (x_t - \bar{\mu}_t)$$

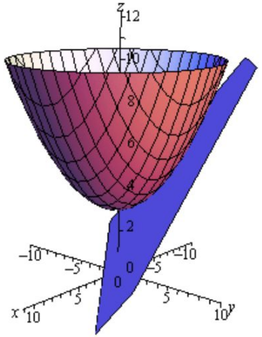
Jacobian matrices

What is a Jacobian?

$$f(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$



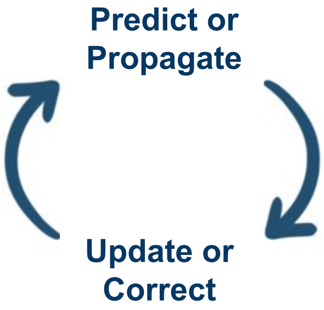


Jacobian J:

- Consider a function $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$
- Then **J** is an *m x n* matrix of gradients!
- Each entry represents a partial derivative: slope of the surface along that direction
- See: https://www.rosroboticslearning.com/jacobian

Extended Kalman Filter

$$\text{State transition:} \quad x_t = \mathbf{f}(x_t, u_t) + \epsilon_t$$
$$\text{Observation:} \quad z_t = \mathbf{h}(x_t) + \delta_t$$



$$\text{Predict State:} \quad x_t = \mathbf{f}(x_t, u_t)$$

Predict covariance:

$$\mathbf{F}_t = \nabla_x \mathbf{f}(x_t, u_t)$$
$$\mathbf{G}_t = \nabla_\epsilon \mathbf{f}(x_t, u_t)$$
$$\mathbf{P}_t = \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_{\mathbf{t}}^{\mathbf{T}} + \mathbf{Q}_t$$

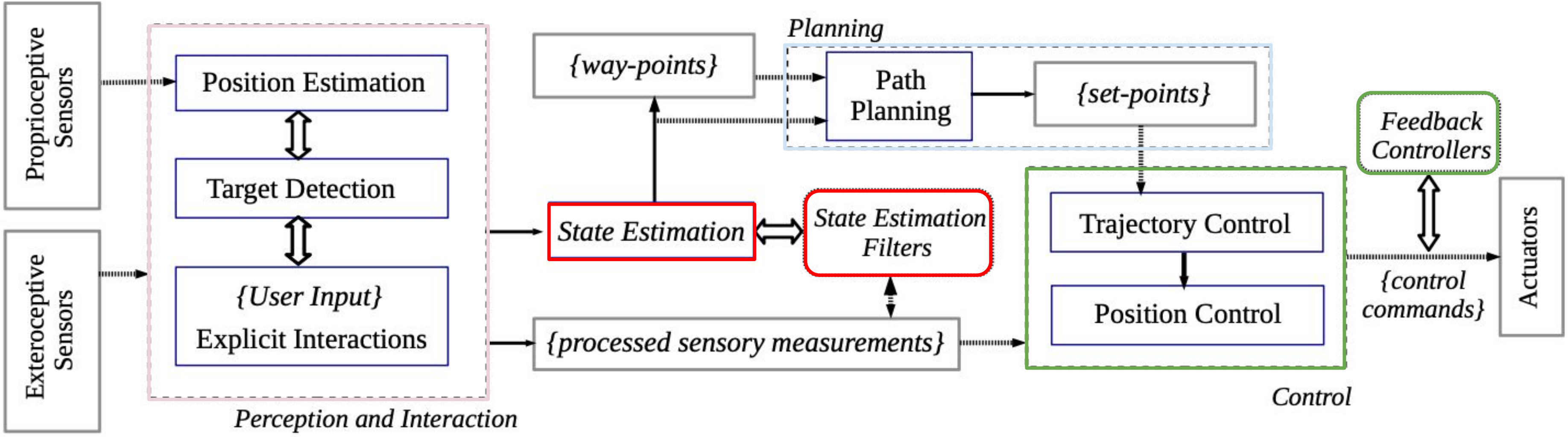
$$\text{Observation residual:} \quad y_t = z_t - \mathbf{h}(x_t)$$

Observation covariance:

$$\mathbf{H}_t = \nabla_x \mathbf{h}(x_t)$$
$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t-1} \mathbf{H}_{\mathbf{t}}^{\mathbf{T}} + \mathbf{R}_t$$
$$\text{Kalman gain:} \quad \mathbf{K}_t = \mathbf{P}_t \mathbf{H}_{\mathbf{t-1}}^{\mathbf{T}} \mathbf{S}_{\mathbf{t}}^{-1}$$
$$\text{Update state:} \quad x_t = x_t + \mathbf{K}_t y_t$$
$$\text{Update covariance:} \quad \mathbf{P}_t = (\mathbf{I}_t - \mathbf{K}_t \mathbf{H}_{\mathbf{t}}) \mathbf{P}_t$$

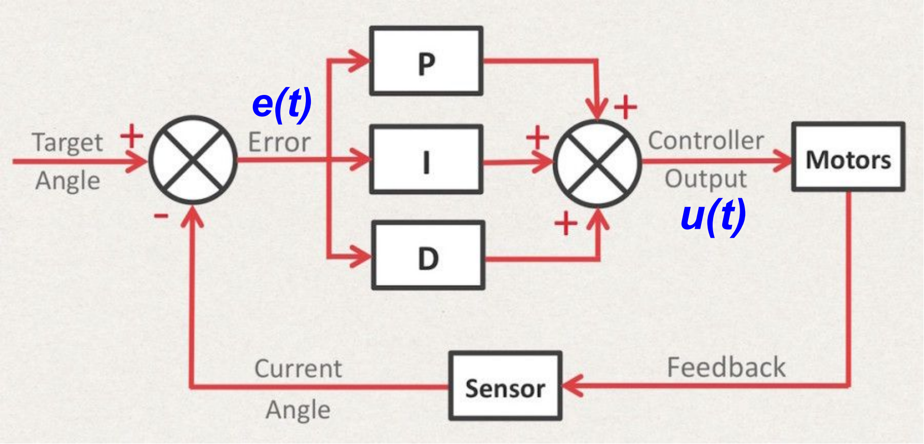
Derivation: Probabilistic Robotics (Ch 3.3)

State Estimation Filter vs Feedback Controller

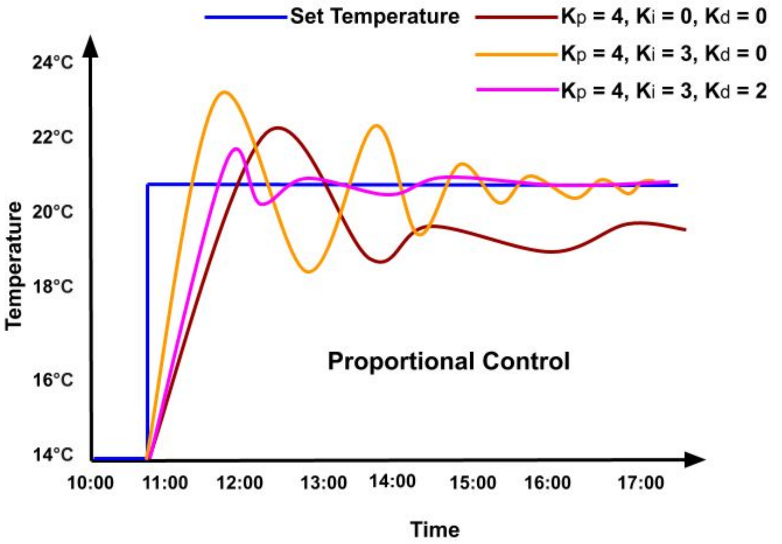


Pipeline Example: autonomous target following by mobile robots

PID Controllers



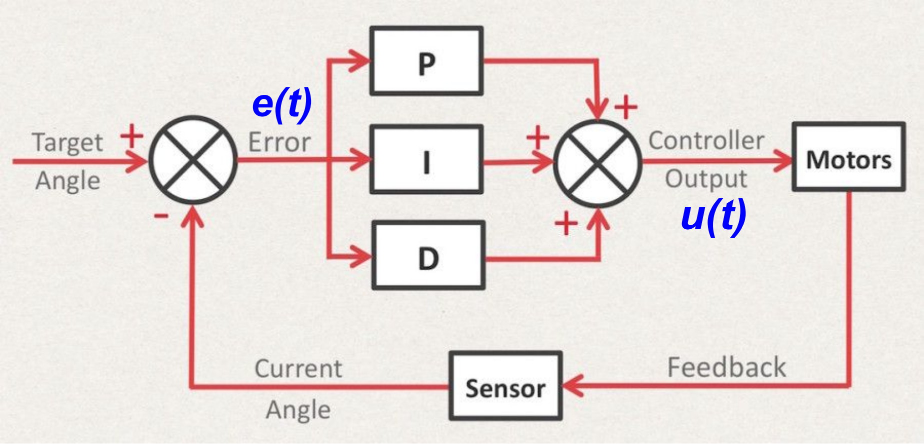
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$



- **Proportional (P) Part**: compensates for the error difference
- **Derivative (D) Part**: reacts for the change of error (restricts oscillation)
- **Integral (I) Part**: responds to the steady-state response

Need to tune K_p, K_i, K_d experimentally

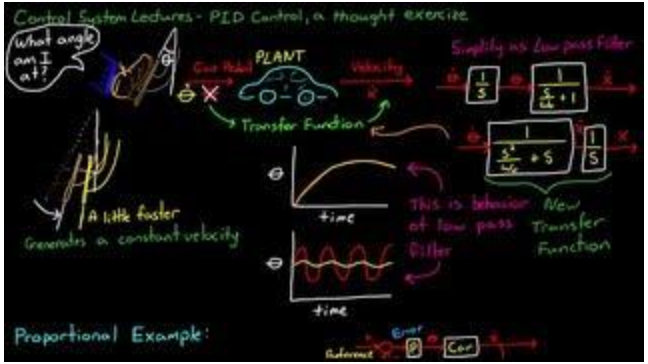
PID Controllers



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

- **Proportional (P) Part**: compensates for the error difference
- **Derivative (D) Part**: reacts for the change of error (restricts oscillation)
- **Integral (I) Part**: responds to the steady-state response

Need to tune K_p, K_i, K_d experimentally

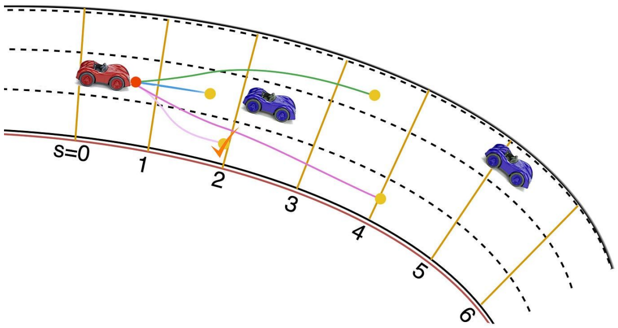


https://youtu.be/XfAt6hNV8XM

https://youtu.be/JFTJ2SS4xyA

Position Control vs Trajectory Control

- **Perception module**: uses sensory measurements for state estimation
- **State estimation filters**: smooths state estimation with noisy measurements
- **Path planner**: finds set-points for executing the robot's goal
- **Position and trajectory controllers:** execute that goal
- **Feedback (PID) controllers:** smooth the controller outputs



Why position controller and trajectory controller?

- With raw position control, the controller simply tries to go to the next setpoint
  - Not smooth or consistent motion
  - Vulnerable to dynamic agents' uncertainties
- A trajectory controller tunes the feedback gains more aggressively
  - Reject disturbance while keeping smooth motion

Advanced topics for the 'Robotics II' course

- Designing a trajectory controller for self-driving car scenario
- Filtering and state estimation pipelines for
  - Search-and-rescue robot
  - Autonomous exploratory robot
- Multi-robot cooperative localization and task execution