COL864: Special Topics in AI

Semester II, 2021-22

Sate Estimation - II

Rohan Paul

Today's lecture

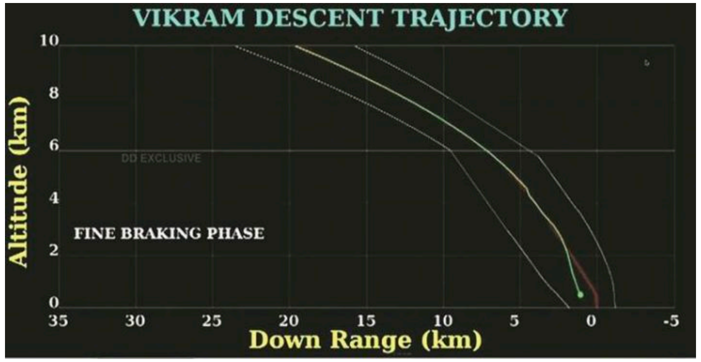
- Last Class
  - State Estimation - I
    - Recursive State Estimation
    - Bayes Filter
- This Class
  - State Estimation - II
    - Kalman Filter
    - Extended Kalman Filter
- References
  - Probabilistic Robotics Ch 3 (Sec. 3.1-3.3)
  - AIMA Ch 15 (Sec. 15.4)

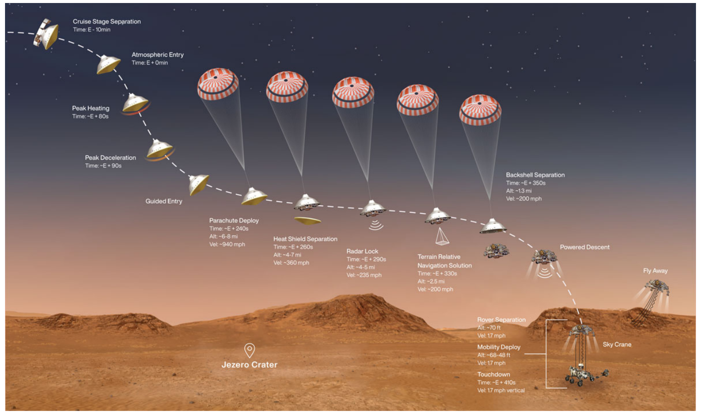
Acknowledgements

These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Nicholas Roy, Wolfram Burgard, Dieter Fox, Sebastian Thrun, Siddharth Srinivasa, Dan Klein, Pieter Abbeel and others.

State Estimation: Continuous Variables

- Bayes Filter till now
  - Discrete state variables
  - E.g., door open or closed.
  - Discrete conditional probability tables.
- Continuous variables
  - Example: we receive continuous measurements of the position or height and seek an estimate. Control the vehicle via velocities.
- Kalman Filter
  - Special case of a Bayes' filter for handling continuous variables.
  - Assumes that the motion model (dynamics/control) and the sensor model is linear Gaussian.
  - E.g., estimating the belief over the location of the agent given the sequence of observations and controls.





Estimating the true state from noisy observations is crucial for planning.

Multivariate Gaussians

- Distribution over a vector of variables
  - E.g., the agent's state in our case.
- Mean vector
  - Expected value of each variable.
- Covariance matrix
  - Covariance between each pair of elements of a given random vector.
  - Diagonals contain variance of each variable in the state.
  - Symmetric and positive semi-definite.

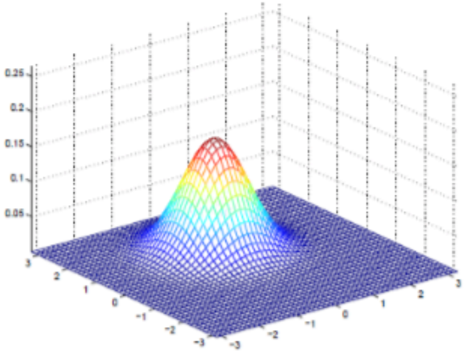
$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

$$\begin{aligned} \mathrm{E}_X[X_i] &= \int x_i p(x;\mu,\Sigma)dx = \mu_i \\ \mathrm{E}_X[X] &= \int x p(x;\mu,\Sigma)dx = \mu \end{aligned}$$

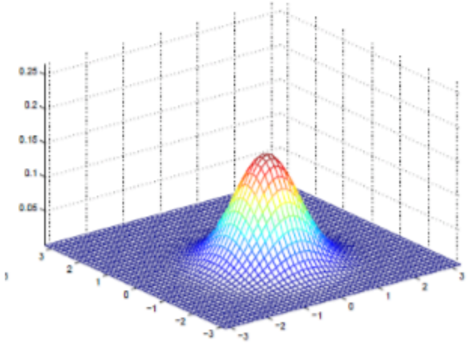
$$\begin{aligned} \mathrm{E}_X[(X_i-\mu_i)(X_j-\mu_j)] &= \int (x_i-\mu_i)(x_j-\mu_j)p(x;\mu,\Sigma)dx = \Sigma_{ij} \\ \mathrm{E}_X[(X-\mu)(X-\mu)^\top] &= \int [(X-\mu)(X-\mu)^\top p(x;\mu,\Sigma)dx = \Sigma \end{aligned}$$

Multivariate Gaussians: Examples

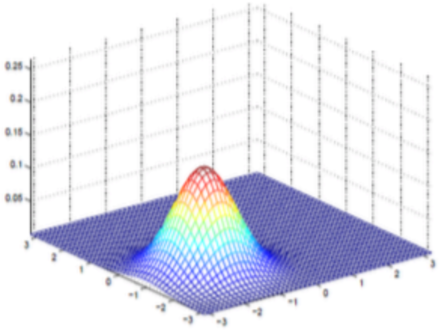
- Varying the mean or origin of the distribution.



- μ = [1; 0]
- Σ = [1 0; 0 1]



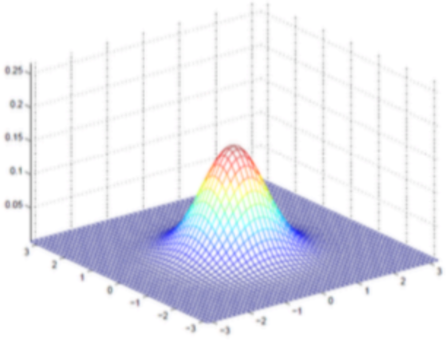
- μ = [-.5; 0]
- Σ = [1 0; 0 1]



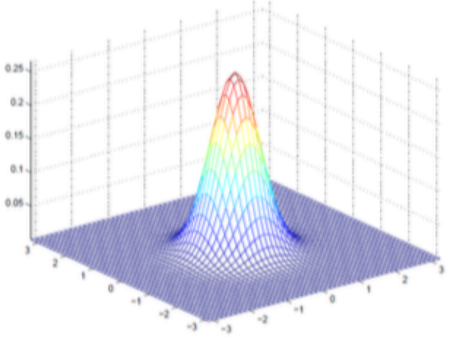
- μ = [-1; -1.5]
- Σ = [1 0; 0 1]

Multivariate Gaussians: Examples

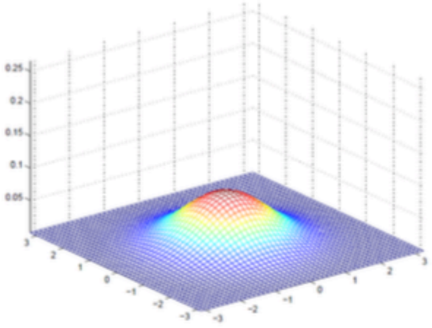
- Changing the variance in the state variables.



- $\mu = [0; 0]$
- $\Sigma = [1\ 0\ ;\ 0\ 1]$



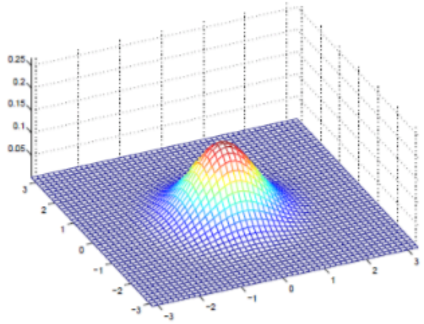
- $\mu = [0; 0]$
- $\Sigma = [.6\ 0\ ;\ 0\ .6]$



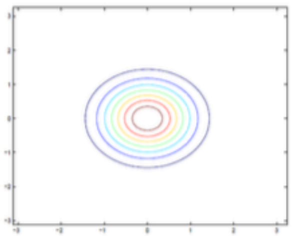
- $\mu = [0; 0]$
- $\Sigma = [2\ 0\ ;\ 0\ 2]$

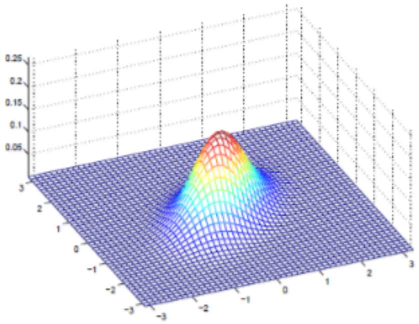
Multivariate Gaussians: Examples

- Changing the variance in the off-diagonal elements.
  - Model variance *between* state variables.

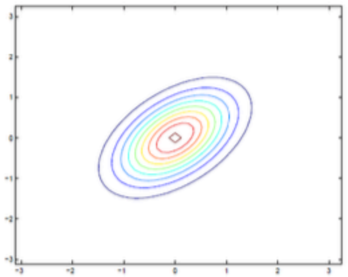


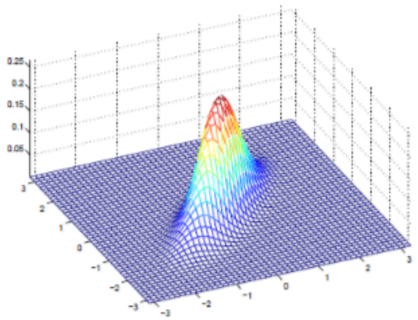
- $\mu = [0; 0]$
- $\Sigma = [1 \ \ 0; 0 \ \ 1]$



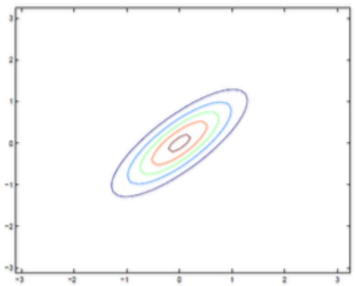


- $\mu = [0; 0]$
- $\Sigma = [1 \ \ 0.5; 0.5 \ \ 1]$

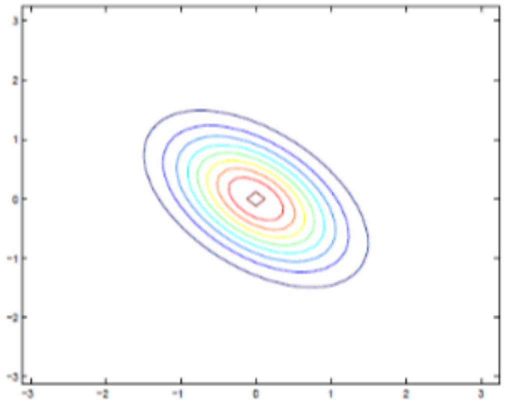




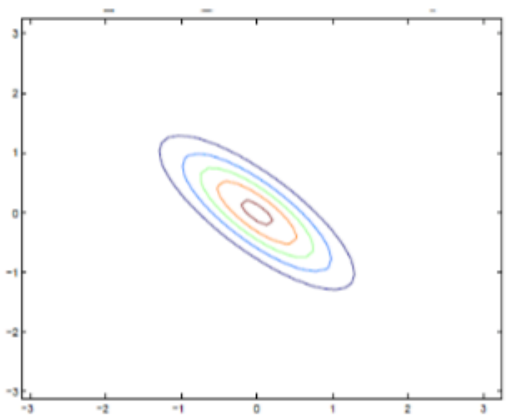
- $\mu = [0; 0]$
- $\Sigma = [1 \ \ 0.8; 0.8 \ \ 1]$



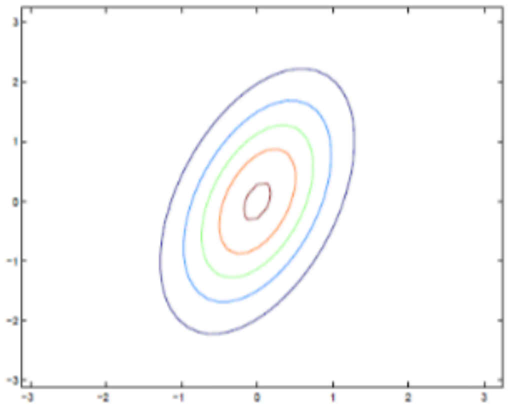
Multivariate Gaussians: Examples



- μ = [0; 0]
- Σ = [1 -0.5 ; -0.5 1]



- μ = [0; 0]
- Σ = [1 -0.8 ; -0.8 1]



- μ = [0; 0]
- Σ = [3 0.8 ; 0.8 1]

Joint Gaussian PDFs: Partitioning of variables

- Partition the random vector as variables as (X, Y).
  - Notice the block structure.
- Why?
  - Later, we would need to marginalize or condition on *some* of the variables.

$$\mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

$$p(\begin{bmatrix}x \\ y\end{bmatrix}; \mu, \Sigma) = \frac{1}{(2\pi)^{(n/2)}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\left(\begin{bmatrix}x \\ y\end{bmatrix} - \begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}\right)^\top \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}^{-1} \left(\begin{bmatrix}x \\ y\end{bmatrix} - \begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}\right)\right)$$

$$\begin{aligned}
\mu_X &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[X] \\
\mu_Y &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[Y] \\
\Sigma_{XX} &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[(X-\mu_X)(X-\mu_X)^\top] \\
\Sigma_{YY} &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[(Y-\mu_Y)(Y-\mu_Y)^\top] \\
\Sigma_{XY} &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[(X-\mu_X)(Y-\mu_Y)^\top] = \Sigma_{YX}^\top \\
\Sigma_{YX} &= \mathrm{E}_{(X,Y)\sim\mathcal{N}(\mu,\Sigma)}[(Y-\mu_Y)(X-\mu_X)^\top] = \Sigma_{XY}^\top
\end{aligned}$$

Joint Gaussian PDFs: Marginalization

- Marginalization
  - Integrating out the effect of a (sub)-set of variables.
  - Resulting is a normal distribution over a smaller set of variables.
  - The resulting distribution is Gaussian.

If

$$(X, Y) \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

Then

$$\begin{aligned} X &\sim \mathcal{N}(\mu_X, \Sigma_{XX}) \\ Y &\sim \mathcal{N}(\mu_Y, \Sigma_{YY}) \end{aligned}$$

Joint Gaussian PDFs: Conditioning

- Conditioning
  - Certain variables are observed (instantiated with observed values).
  - We seek the distribution over the remaining set of variables.
  - Conditioning a Gaussian results in another Gaussian distribution.

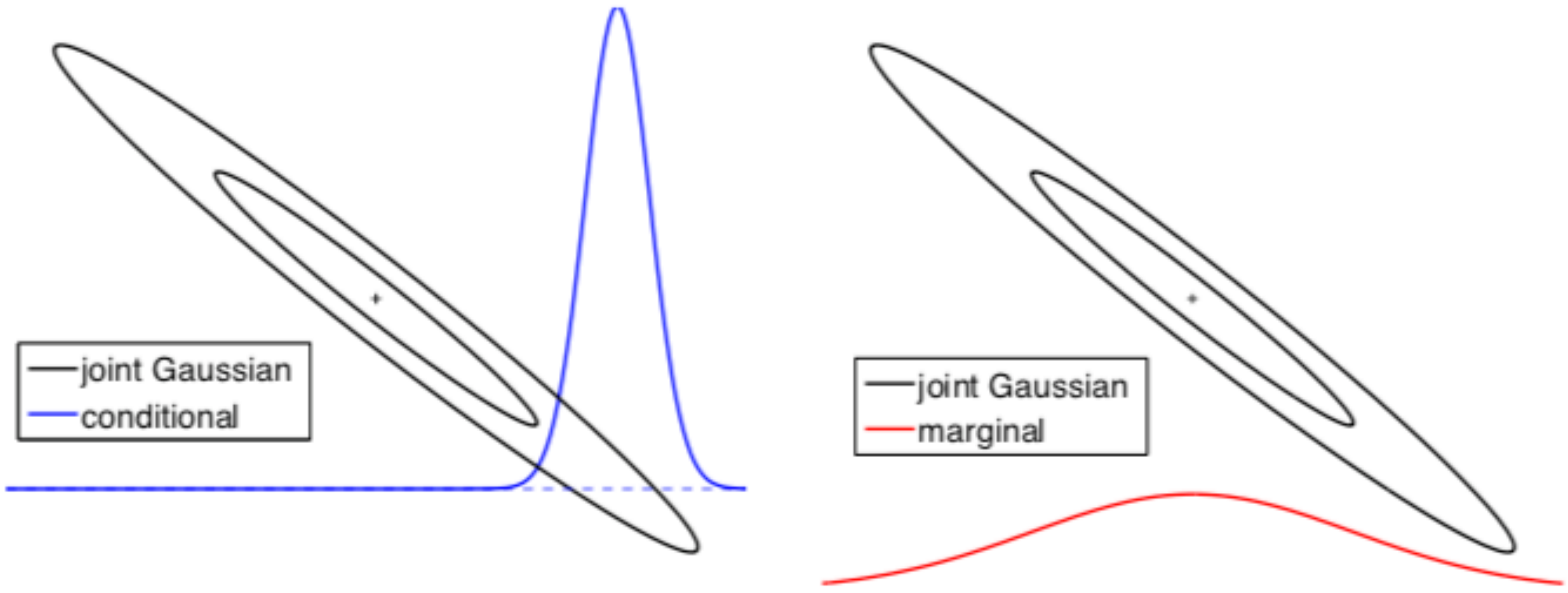
If

$$(X,Y) \sim \mathcal{N}(\mu,\Sigma) = \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

Then

$$\begin{aligned} X|Y = y_0 &\sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y_0 - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}) \\ Y|X = x_0 &\sim \mathcal{N}(\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(x_0 - \mu_X), \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}) \end{aligned}$$

Conditionals and Marginals of a Gaussian Distribution



Both the conditionals and the marginals of a joint Gaussian are again Gaussian.

Figure from Carl Rasmussen, Machine Learning Summer School 2009

Other Properties

- Linear transformation
- Product

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow p(X_1) \times p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2, \quad \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}} \right)$$

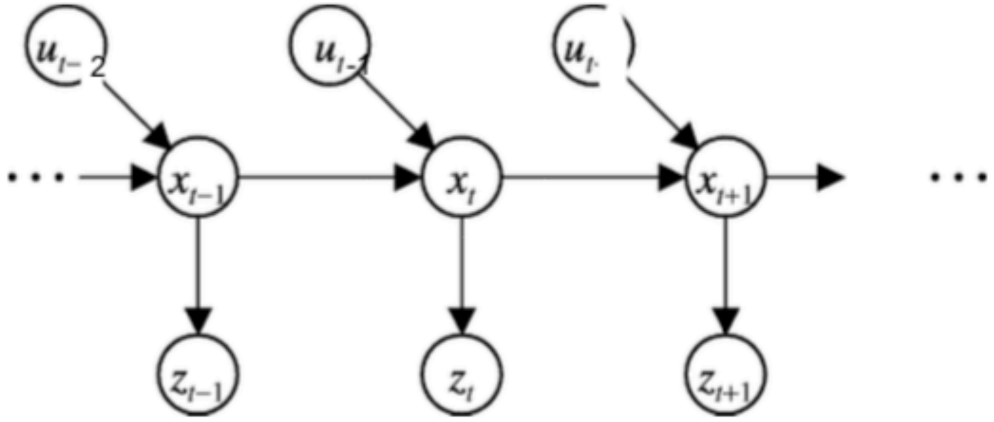
$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \times p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \quad \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}} \right)$$

Kalman Filter

- Provided
  - A belief over the initial state
  - Sensor model is linear Gaussian
  - Motion model is linear Gaussian
- What is our goal
  - Estimate a belief over the latent state at time t.

$$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$$



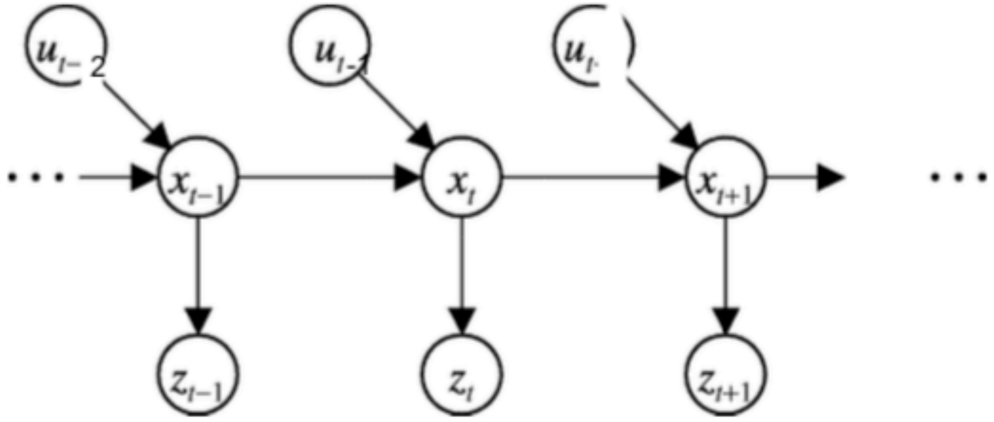
$$\begin{aligned} X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\ Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

Sometimes, explicit mention of d is dropped in the sensor model

Kalman Filter: Components

- A_t Matrix
  - Size ($n{\times}n$) that describes how the state evolves from 1 to t without controls or noise.
- B_t Matrix
  - Size ($n{\times}l$) that describes how the control u changes tl state from t-1 to t.
- Epsilon
  - Random variable (size n) representing the process noise that is assumed to be independent and normally distributed with covariance Q_t (size nxn).
- C_t Matrix
  - Size ($k{\times}n$) that describes how to map the state x_t to a observation z_t.
- d_t Vector
  - Size (k) constant offset added. Often explicit mentior of d is dropped from the sensor model.
- Delta
  - Random variable (size k) representing the measurement noise that is assumed to be independent and normally distributed with covariance R_t (size kxk).

$$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$$

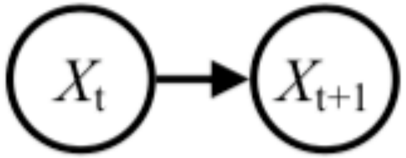


$$\begin{aligned} X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\ Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

Dynamics (Action) Update

- Assume we have current belief for $X_{t|0:t}$:

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



Update the belief using action

- Then, after one time step passes:

Marginalize out x_t

$$p(x_{t+1}|z_{0:t}, u_{0:t}) = \int_{x_t} p(x_{t+1}, x_t|z_{0:t}, u_{0:t})dx_t$$

Apply conditional independence

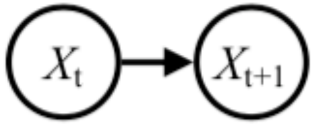
$$\begin{aligned} p(x_{t+1}, x_t|z_{0:t}, u_{0:t}) &= p(x_{t+1}|x_t, z_{0:t}, u_{0:t})p(x_t|z_{0:t}, u_{0:t}) \\ &= p(x_{t+1}|x_t, u_t)p(x_t|z_{0:t}, u_{0:t}) \end{aligned}$$

Product of two Gaussian distributions. We know that this is a Gaussian distribution.

Dynamics (Action) Update

- Assume we have current belief for $X_{t|0:t}$:

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



Update the belief using action

- Then, after one time step passes:

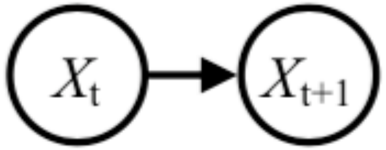
$$p(x_{t+1}|z_{0:t}, u_{0:t}) = \int_{x_t} p(x_{t+1}, x_t|z_{0:t}, u_{0:t})dx_t$$

$$\begin{aligned} p(x_{t+1}, x_t|z_{0:t}, u_{0:t}) &= p(x_{t+1}|x_t, u_t)p(x_t|z_{0:t}, u_{0:t}) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma_{t|0:t}|^{1/2}} e^{-\frac{1}{2}(x_t-\mu_{t|0:t})^\top \Sigma_{t|0:t}^{-1}(x_t-\mu_{t|0:t})} \\ &\quad \frac{1}{(2\pi)^{n/2}|Q_t|^{1/2}} e^{-\frac{1}{2}(x_{t+1}-(A_t x_t+B_t u_t))^\top Q_t^{-1}(x_{t+1}-(A_t x_t+B_t u_t))} \end{aligned}$$

Product of two Gaussian distributions - a Gaussian distribution.

Dynamics (Action) Update

- Assume we have



$$\begin{aligned} X_{t|0:t} &\sim \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t}) \\ X_{t+1} &= A_t X_t + B_t u_t + \epsilon_t, \\ \epsilon_t &\sim \mathcal{N}(0, Q_t), \text{ and independent of } x_{0:t}, z_{0:t}, u_{0:t}, \epsilon_{0:t-1} \end{aligned}$$

- Then we have

$$\begin{aligned} (X_{t|0:t}, X_{t+1|0:t}) &\sim \mathcal{N}\left(\begin{bmatrix} \mu_{t|0:t} \\ \mu_{t+1|0:t} \end{bmatrix}, \begin{bmatrix} \Sigma_{t|0:t} & \Sigma_{t,t+1|0:t} \\ \Sigma_{t+1,t|0:t} & \Sigma_{t+1|0:t} \end{bmatrix} \right) \\ &= \mathcal{N}\left(\begin{bmatrix} \mu_{t|0:t} \\ A_t \mu_{t|0:t} + B_t u_t \end{bmatrix}, \begin{bmatrix} \Sigma_{t|0:t} & \Sigma_{t|0:t} A_t^\top \\ A_t \Sigma_{t|0:t} & A_t \Sigma_{t|0:t} A_t^\top + Q_t \end{bmatrix} \right) \end{aligned}$$

- Marginalizing the joint, we immediately get

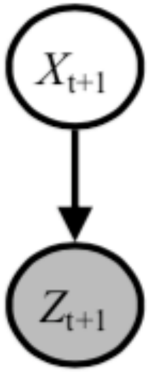
$$X_{t+1|0:t} \sim \mathcal{N}\left(A_t \mu_{t|0:t} + B_t u_t, A_t \Sigma_{t|0:t} A_t^\top + Q_t\right)$$

← A new Gaussian with the mean vector and the covariance matrix updated.

Notation: (I | j) implies an estimate of the quantity at time i using “observations” received till time j.

Measurement Update

Update the belief over the state by conditioning on the observation



- Assume we have:

$$\begin{aligned} X_{t+1|0:t} &\sim \mathcal{N}\left(\mu_{t+1|0:t}, \Sigma_{t+1|0:t}\right) \\ Z_{t+1} &\sim C_{t+1}X_{t+1} + d_{t+1} + \delta_{t+1} \\ \delta_{t+1} &\sim \mathcal{N}(0, R_t), \text{ and independent of } x_{0:t+1}, z_{0:t}, u_{0:t}, \epsilon_{0:t}, \end{aligned}$$

- Then:

$$(X_{t+1|0:t}, Z_{t+1|0:t}) \sim \mathcal{N}\left(\begin{bmatrix} \mu_{t+1|0:t} \\ C_{t+1}\mu_{t+1|0:t} + d \end{bmatrix}, \begin{bmatrix} \Sigma_{t+1|0:t} & \Sigma_{t+1|0:t}C_{t+1}^\top \\ C_{t+1}\Sigma_{t+1|0:t} & C_{t+1}\Sigma_{t+1|0:t}C_{t+1}^\top + R_{t+1} \end{bmatrix}\right)$$

- And, by conditioning on $Z_{t+1} = z_{t+1}$ (see lecture slides on Gaussians) we readily get:

$$\begin{aligned} X_{t+1}|z_{0:t+1}, u_{0:t} &= X_{t+1|0:t+1} \\ &\sim \mathcal{N}\left(\mu_{t+1|0:t} + \Sigma_{t+1|0:t}C_{t+1}^\top(C_{t+1}\Sigma_{t+1|0:t}C_{t+1}^\top + R_{t+1})^{-1}(z_{t+1} - (C_{t+1}\mu_{t+1|0:t} + d)),\right. \\ &\qquad \left.\Sigma_{t+1|0:t} - \Sigma_{t+1|0:t}C_{t+1}^\top(C_{t+1}\Sigma_{t+1|0:t}C_{t+1}^\top + R_{t+1})^{-1}C_{t+1}\Sigma_{t+1|0:t}\right) \end{aligned}$$

Kalman Filter

Core Idea: Recursively update the mean and the covariance using the action model and the sensor model.

Initial belief is a Gaussian → At time 0: $X_0 \sim \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$

Belief always remains Gaussian → For t = 1, 2, …

Prediction

- What would be the next state belief under the process model?
- Updates the mean and inflates the covariance.

→ Dynamics update:

$$\begin{aligned}\mu_{t+1|0:t} &= A_t \mu_{t|0:t} + B_t u_t \\ \Sigma_{t+1|0:t} &= A_t \Sigma_{t|0:t} A_t^\top + Q_t\end{aligned}$$

Correction

- Update the predicted belief with the observation.
- Updates the mean and deflates the covariance.

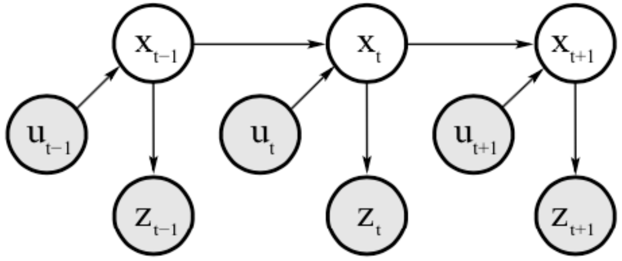
→ Measurement update:

$$\begin{aligned}\mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + \Sigma_{t+1|0:t} C_{t+1}^\top (C_{t+1} \Sigma_{t+1|0:t} C_{t+1}^\top + R_{t+1})^{-1} (z_{t+1} - (C_{t+1} \mu_{t+1|0:t} + d)) \\ \Sigma_{t+1|0:t+1} &= \Sigma_{t+1|0:t} - \Sigma_{t+1|0:t} C_{t+1}^\top (C_{t+1} \Sigma_{t+1|0:t} C_{t+1}^\top + R_{t+1})^{-1} C_{t+1} \Sigma_{t+1|0:t}\end{aligned}$$

- Often written as:

$$\begin{aligned}K_{t+1} &= \Sigma_{t+1|0:t} C_{t+1}^\top (C_{t+1} \Sigma_{t+1|0:t} C_{t+1}^\top + R_{t+1})^{-1} && \text{(Kalman gain)} \\ \mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + K_{t+1}(z_{t+1} - (C_{t+1} \mu_{t+1|0:t} + d)) && \text{“innovation”} \\ \Sigma_{t+1|0:t+1} &= (I - K_{t+1} C_{t+1}) \Sigma_{t+1|0:t}\end{aligned}$$

Kalman Filter: Alternate Notation



Previous belief

action

observation

Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

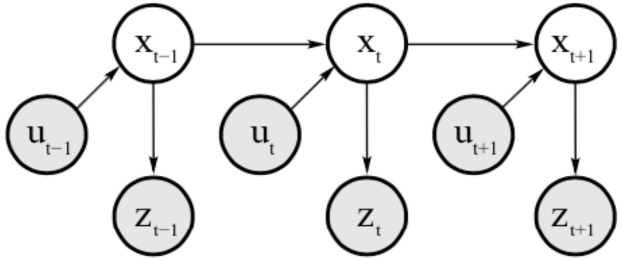
$$\bar{\mu}_t = A_t \, \mu_{t-1} + B_t \, u_t$$
$$\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t \, C_t^T (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \, \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t \, C_t) \, \bar{\Sigma}_t$$

return μ_t, Σ_t

Belief is gaussian

Probabilistic Robotics Ch 3.

Kalman Filter



Algorithm Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

$$\bar{\mu}_t = A_t \, \mu_{t-1} + B_t \, u_t$$

$$\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t$$

Action

$$K_t = \bar{\Sigma}_t \, C_t^T (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$$

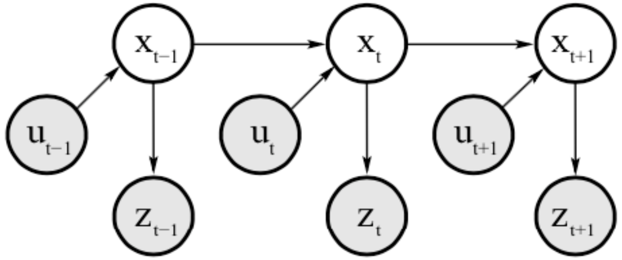
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \, \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t \, C_t) \, \bar{\Sigma}_t$$

return μ_t, Σ_t

Probabilistic Robotics Ch 3.

Kalman Filter



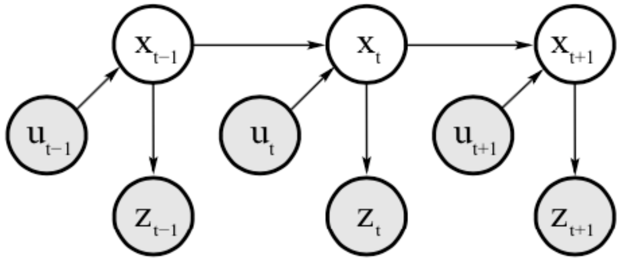
Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

$$\bar{\mu}_t = A_t \, \mu_{t-1} + B_t \, u_t$$
$$\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t \, C_t^T (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \, \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t \, C_t) \, \bar{\Sigma}_t$$

return μ_t, Σ_t

← *Kalman gain*: Degree at which observation factors into belief

Kalman Filter



Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

$$\bar{\mu}_t = A_t\, \mu_{t-1} + B_t\, u_t$$
$$\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t\, C_t^T (C_t\, \bar{\Sigma}_t\, C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t\, \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t\, C_t)\, \bar{\Sigma}_t$$

return μ_t, Σ_t

Compute mean from difference between expected and observed observations multiplied by Kalman Gain

"innovation"

Kalman Filter: Constant Velocity Case

- $X = [x, y, v_x, v_y]$
- Constant velocity motion:

$$f(X, v) = [x + \Delta t \cdot v_x, y + \Delta t \cdot v_y, v_x, v_y] + v$$

$$v \sim N(0, Q) \qquad Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & q^2 & 0 \\ 0 & 0 & 0 & q^2 \end{pmatrix}$$

- Only position is observed:

$$z = h(X, w) = [x, y] + w$$

$$w \sim N(0, R) \qquad R = \begin{pmatrix} r^2 & 0 \\ 0 & r^2 \end{pmatrix}$$

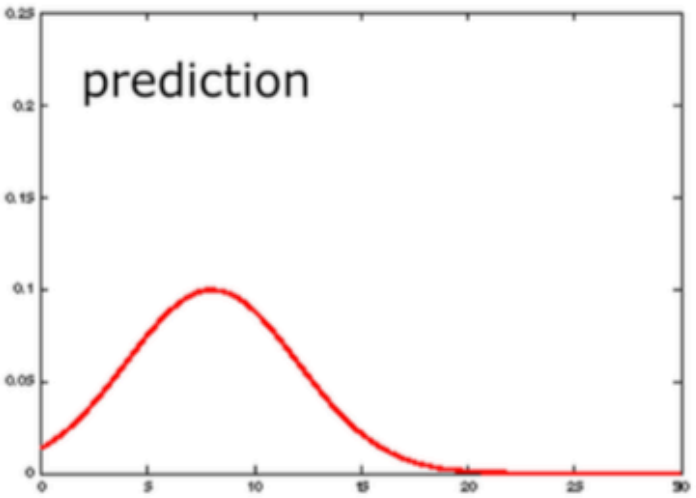
Kalman Filter: Constant Velocity Case

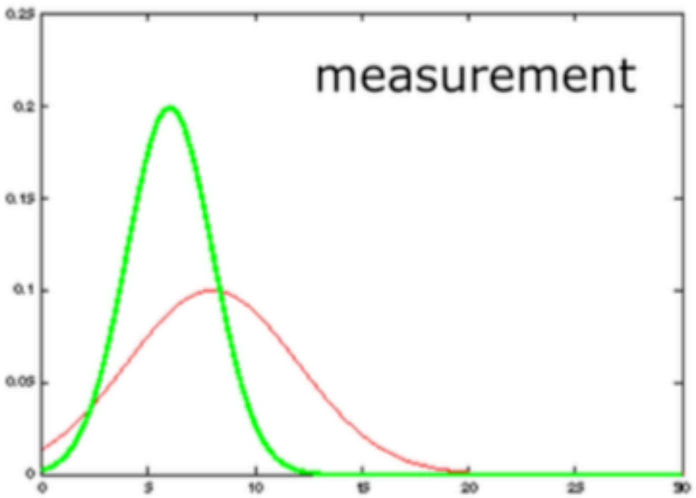
$$f(X,v) = [x + \Delta t \cdot v_x, y + \Delta t \cdot v_y, v_x, v_y] + v \qquad z = h(X,w) = [x, y] + w$$

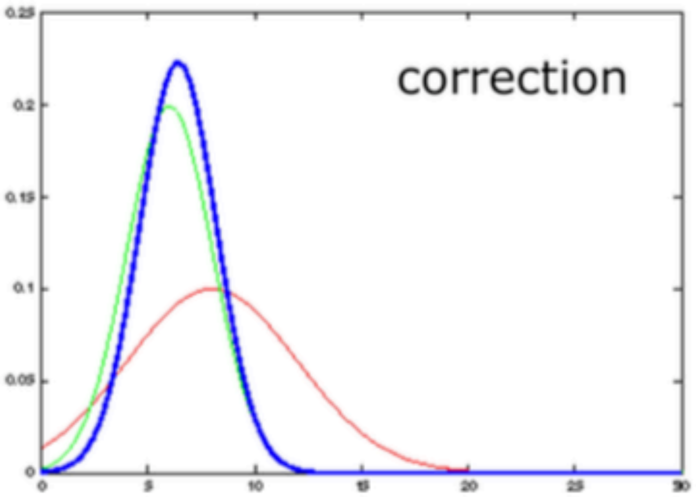
$$\begin{pmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A_t} \begin{pmatrix} x_k \\ y_k \\ v_{x,k-1} \\ v_{y,k-1} \end{pmatrix} + N(0, Q_k) \qquad \begin{pmatrix} x_{obs} \\ y_{obs} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{C_t} \begin{pmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{pmatrix} + N(0, R_k)$$

If there were actions (e.g., changes to velocity) then the B matrix would be added in the motion model.

Example: 1D Gaussian Case





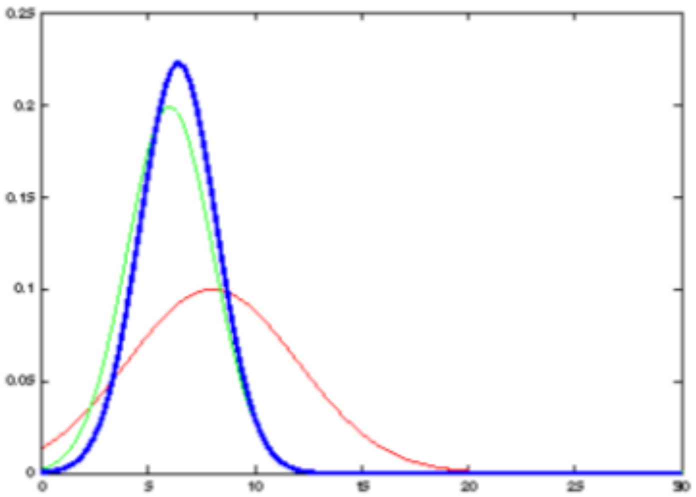


The corrected mean lies between the predicted and the mean of the measurement model. Weighted sum.

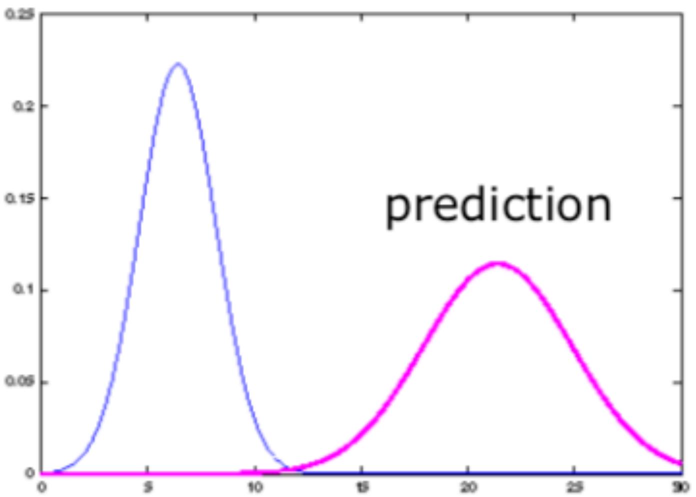
$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases} \quad \text{with} \quad K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$$

Example: 1D Gaussian Case



Belief after last measurement update.

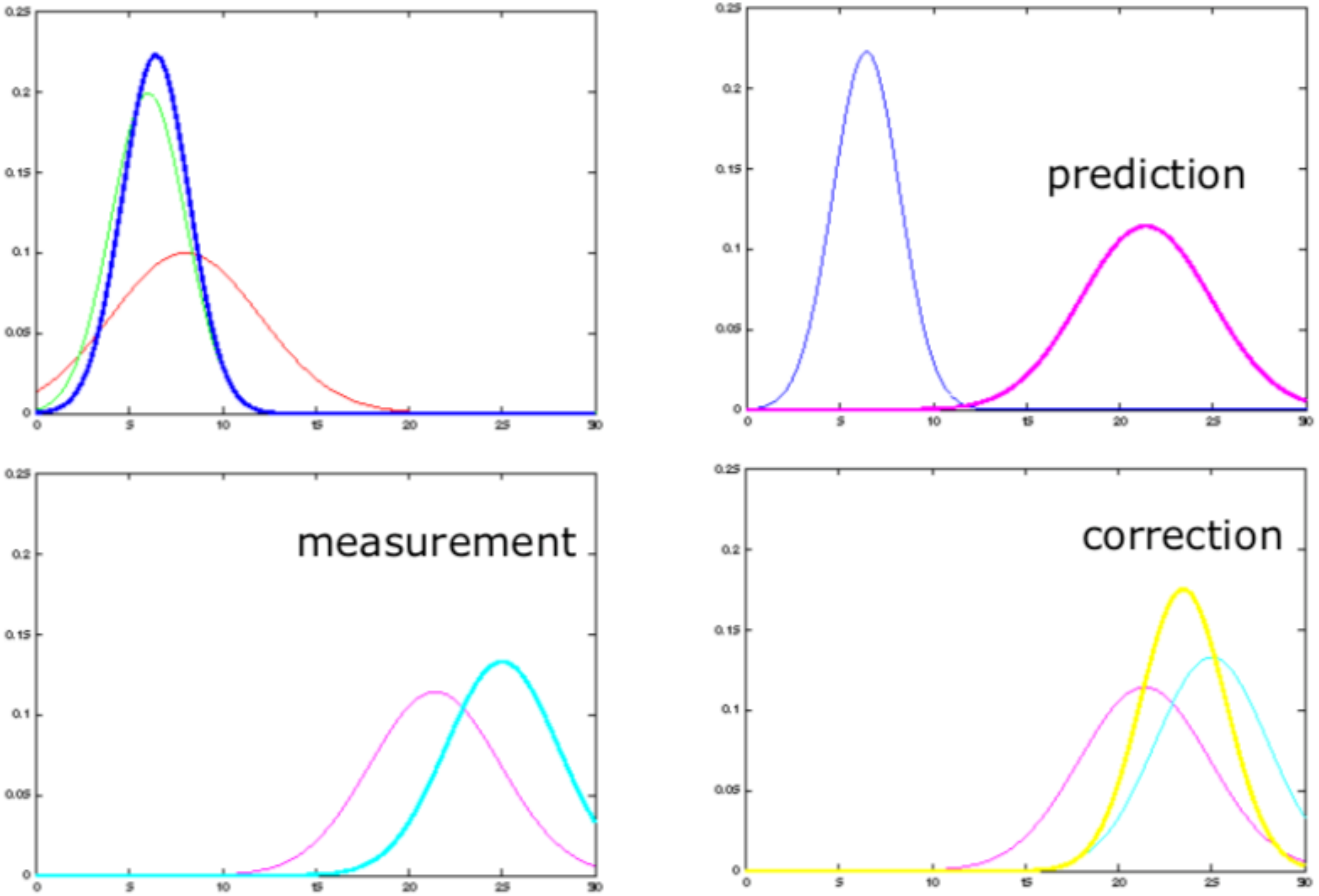


Magenta is the state after the prediction step is applied. The belief becomes less – localized.

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t \end{cases}$$

Example: 1D Gaussian Case



New measurement and correction.

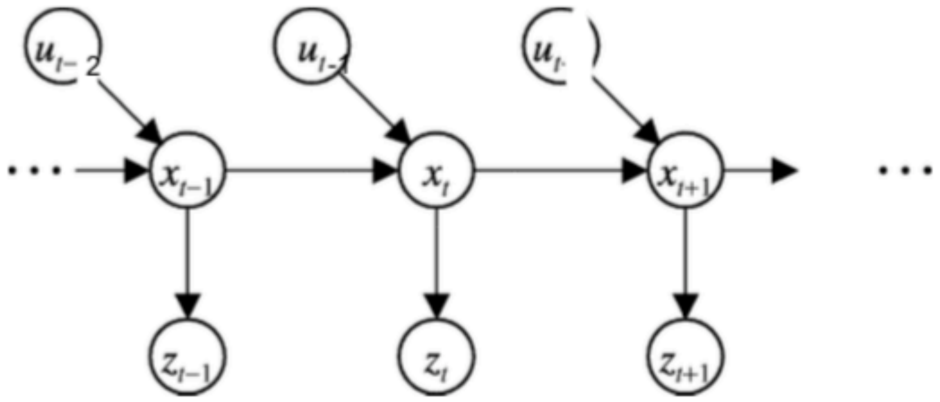
Kalman Filter: Other Takeaways

- **Optimal estimator**
  - Kalman filter is the optimal estimator for linear Gaussian case (i.e., we can't do better under the assumptions).
- **Efficient**
  - Polynomial in the measurement dimensionality k and the state dimensionality n: $O(k^{2.376} + n^2)$
- **Structure**
  - Asynchronisity: if no observations then propagate the motion model.
  - The measurement need not fully determine the latent state. Inherently, updating with partial observations.
  - Requires an initial prior mean and covariance. Predictor and corrector architecture.
- **Assumes and maintains a Gaussian Belief**
  - Unimodal and Gaussian.
  - Problem: in real life belief is often non-Gaussian and multi-modal.

Non-linearity: Extended Kalman Filter

- Kalman Filter (KF)
  - Assumed linear motion and observation models.
- Non-linearity
  - In several cases the sensor and the motion may be non-linear.
- Extended Kalman Filter
  - The EKF provides a way to handle non-linear motion and observation models.
  - “Extends” the use of the KF to non-linear problems.

$$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$$



$$\begin{aligned} X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\ Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

Non-linear Models

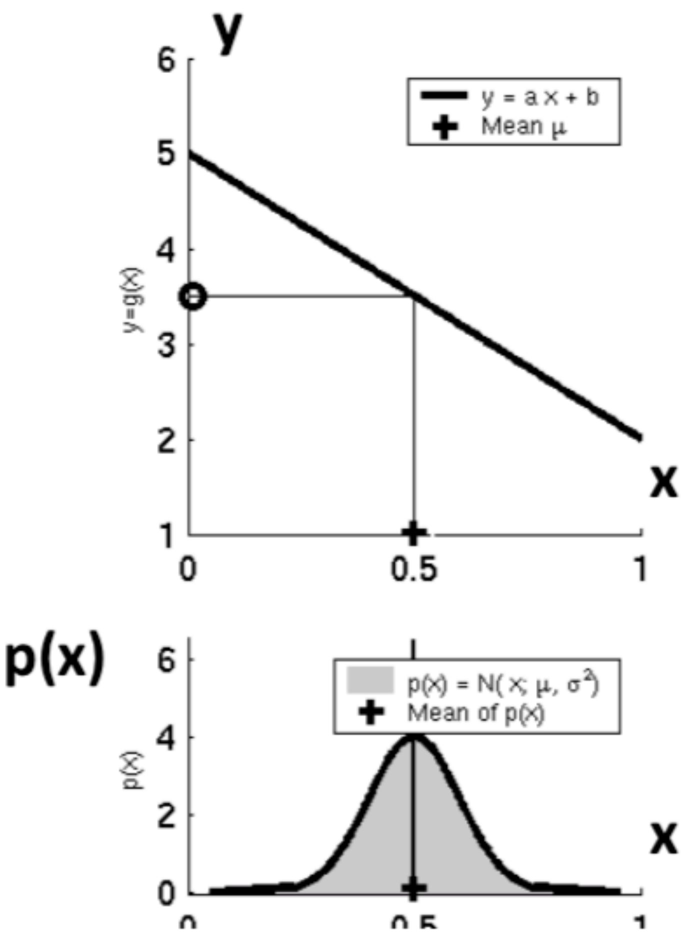
- Non-linear setting
  - The next state is a non-linear function of the current state and actions.
    - *Example: if the control input is a velocity then the velocity components have cosine/sine terms.*
  - The observation is a a non-linear function of the state.
    - *Example: observation is a distance to a landmark instead of (x,y) positions. Distance is a non-linear operation.*
- Linear setting
  - As discussed for KF.

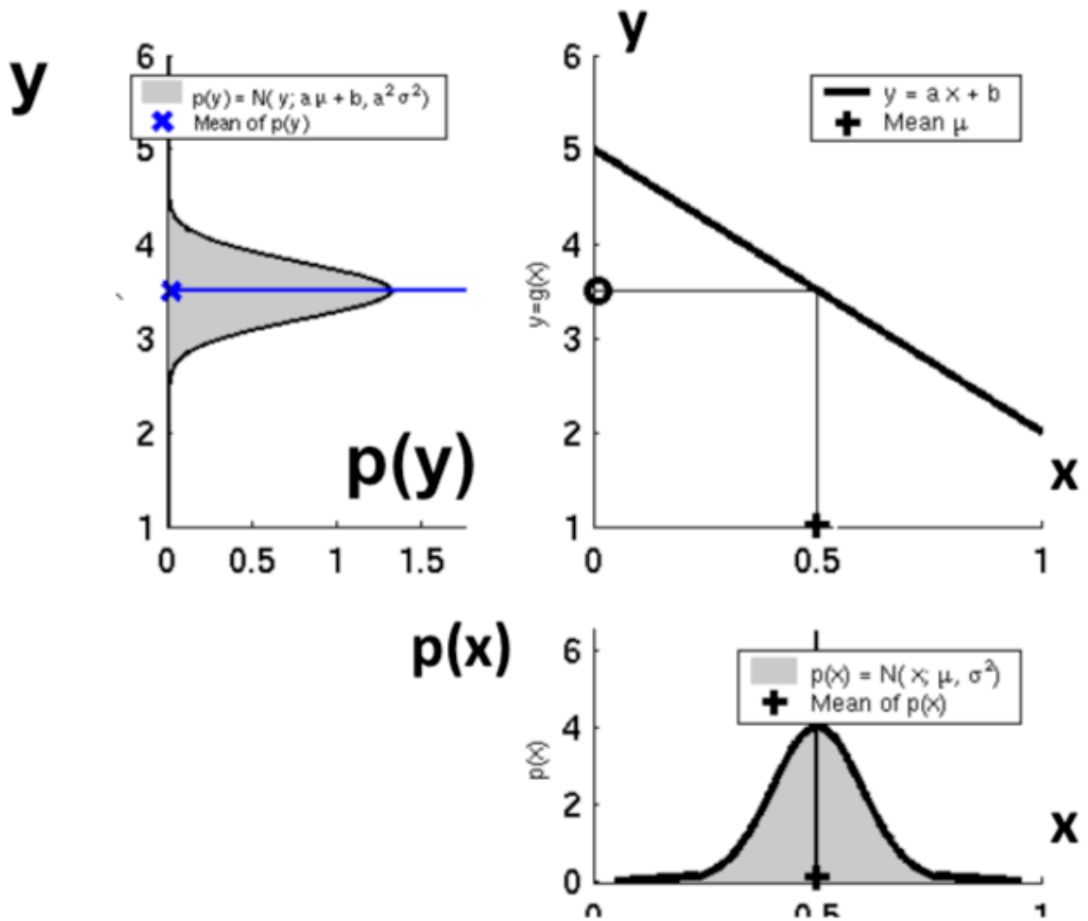
$$
\begin{aligned}
X_{t+1} &= f_t(X_t, u_t) + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t &= h_t(X_t) + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\end{aligned}
$$

$$
\begin{aligned}
X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\end{aligned}
$$

How do we update the belief over the state when there are non-linear dynamics and measurement functions are present.

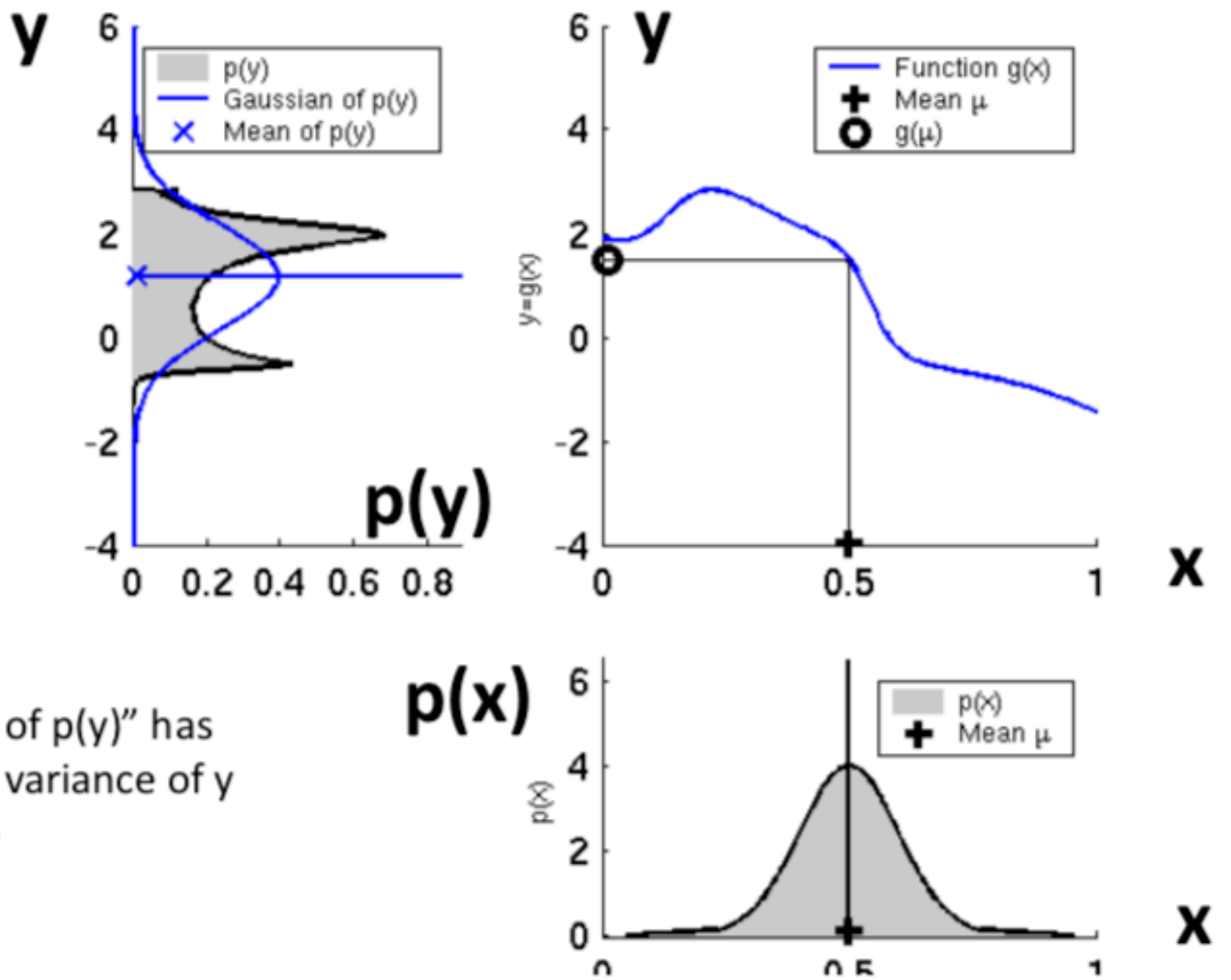
Applying a linear function on Gaussian Belief





Applying a *non-linear* function on a Gaussian Belief

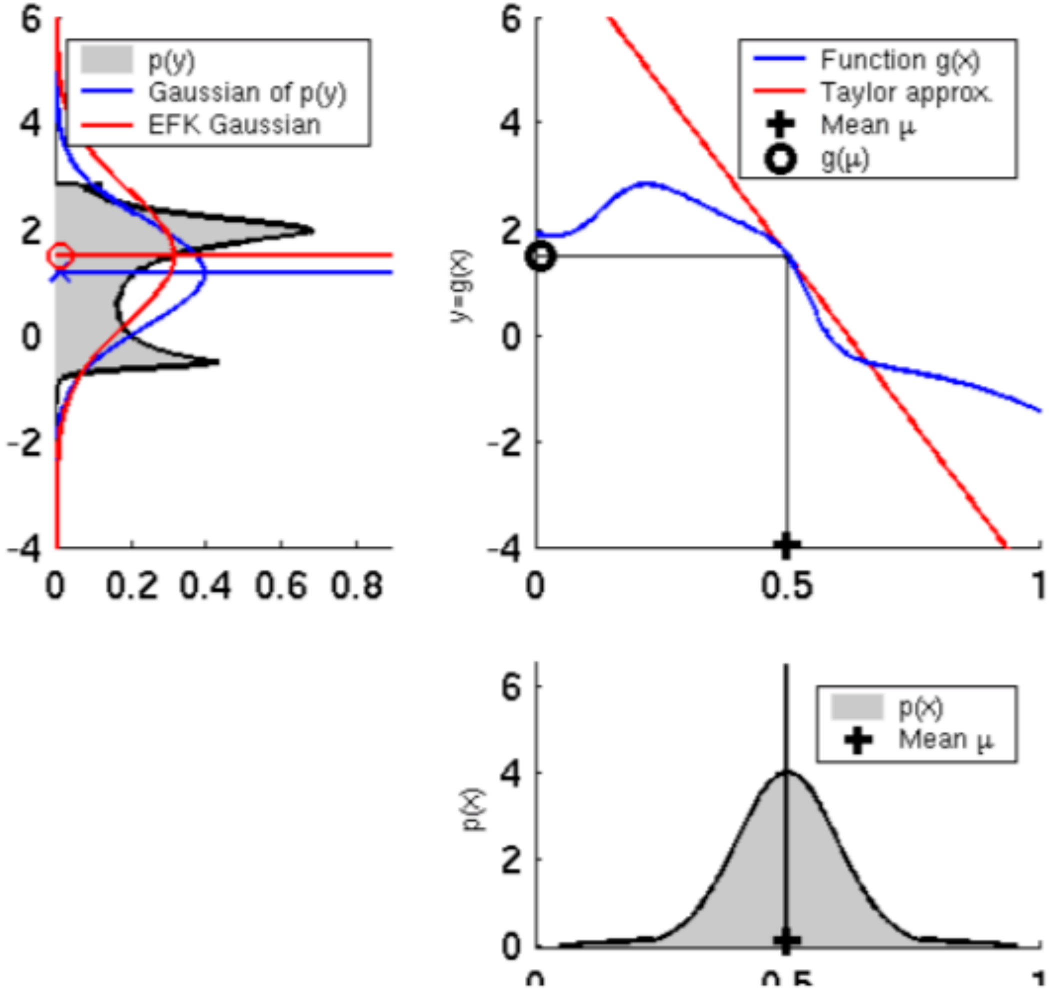
- A Gaussian random variable passed through a non-linear transformation.



“Gaussian of p(y)” has mean and variance of y under p(y)

EKF Linearization

- Problem
  - With a non-linear transformation, the resulting belief is non-Gaussian.
- Solution
  - Can the non-linear function be **linearized** or (locally) approximated as a linear function?
  - Once linearized, the transformed belief can be approximated as a Gaussian.
- EKF Linearization
  - Instead of passing the Gaussian through a non-linear function, pass it through a locally linear approximation to the function.



EKF Linearization: First-Order Taylor Series Expansion

- **Dynamics model:** for x_t "close to" μ_t we have:

$$\begin{aligned} f_t(x_t, u_t) &\approx f_t(\mu_t, u_t) + \frac{\partial f_t(\mu_t, u_t)}{\partial x_t}(x_t - \mu_t) \\ &= f_t(\mu_t, u_t) + F_t(x_t - \mu_t) \end{aligned}$$

- **Measurement model:** for x_t "close to" μ_t we have:

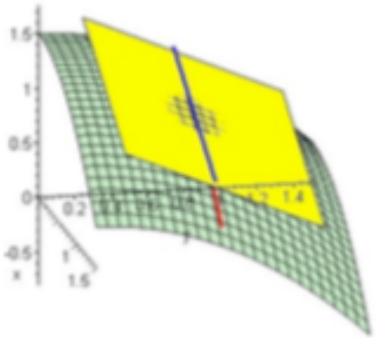
$$\begin{aligned} h_t(x_t) &\approx h_t(\mu_t) + \frac{\partial h_t(\mu_t)}{\partial x_t}(x_t - \mu_t) \\ &= h_t(\mu_t) + H_t(x_t - \mu_t) \end{aligned}$$

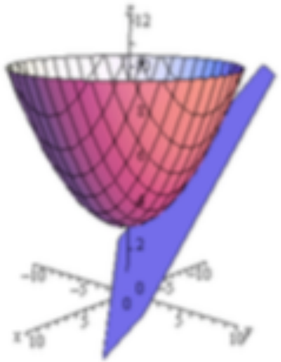
Note: linearization is around the current mean estimate of the belief over the state.

Jacobian Matrix

- Given a vector valued function f(x) from dimension n to m.
- The Jacobian matrix F_x is of size (n x m).
- The orientation of the tangent plane to the vector-valued function at a given point
- Generalizes the gradient of a scalar valued function

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \qquad \mathbf{F}_\mathbf{x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

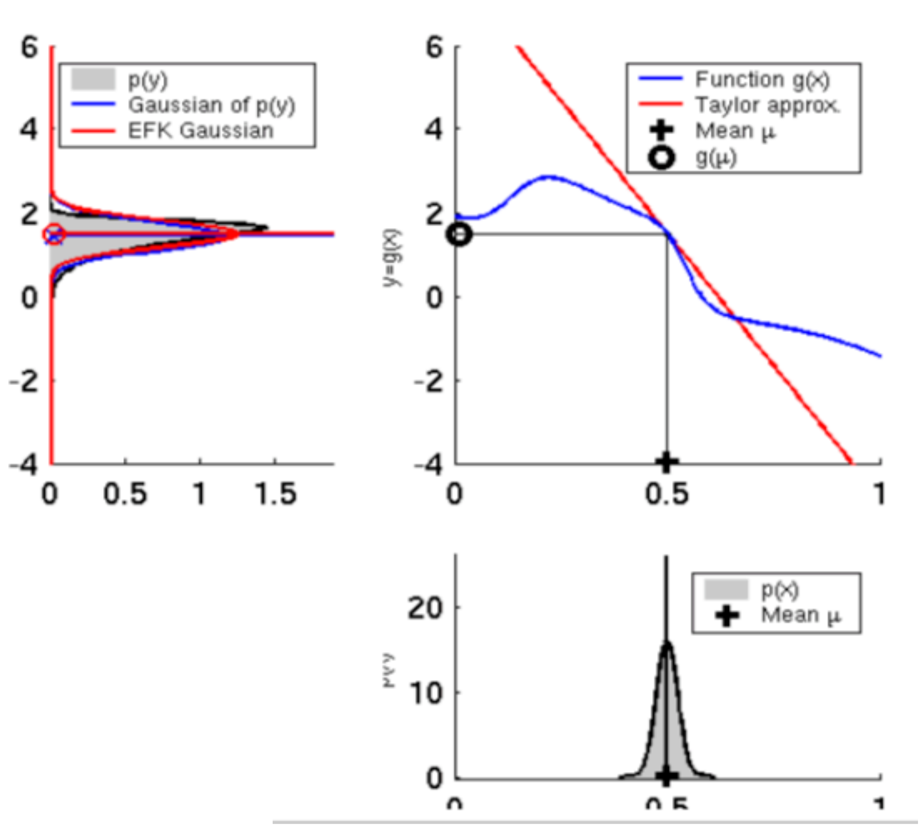


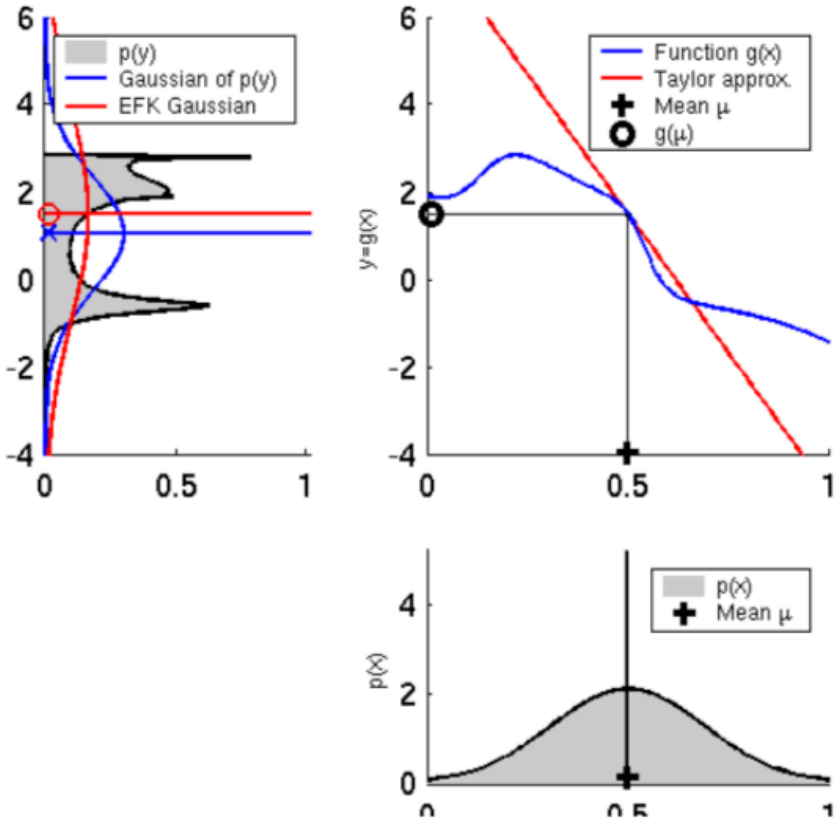


Courtesy: Wolfram Burgard

EKF Linearization

- Dependence of the approximation quality on the uncertainty.
- Cases: when p(X) initial belief has low and high variance relative to the region in which the linearization is accurate.





EKF Algorithm

- At time 0: $X_0 \sim \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$
- For t = 1, 2, …
  - Dynamics update: $f_t(x_t, u_t) \approx a_{0,t} + F_t(x_t - \mu_{t|0:t})$

$$
\begin{aligned}
(a_{0,t}, F_t) &= \text{linearize}(f_t, \mu_{t|0:t}, \Sigma_{t|0:t}, u_t) \\
\mu_{t+1|0:t} &= a_{0,t} \\
\Sigma_{t+1|0:t} &= F_t \Sigma_{t|0:t} F_t^\top + Q_t
\end{aligned}
$$

  - Measurement update: $h_{t+1}(x_{t+1}) \approx c_{0,t+1} + H_{t+1}(x_{t+1} - \mu_{t+1|0:t})$

$$
\begin{aligned}
(c_{0,t+1}, H_{t+1}) &= \text{linearize}(h_{t+1}, \mu_{t+1|0:t}, \Sigma_{t+1|0:t}) \\
K_{t+1} &= \Sigma_{t+1|0:t} H_{t+1}^\top (H_{t+1} \Sigma_{t+1|0:t} H_{t+1}^\top + R_{t+1})^{-1} \\
\mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + K_{t+1}(z_{t+1} - c_{0,t+1}) \\
\Sigma_{t+1|0:t+1} &= (I - K_{t+1} H_{t+1}) \Sigma_{t+1|0:t}
\end{aligned}
$$

EKF Algorithm

Linearization of the motion and the observation models.

- Prediction:

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

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

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$

Jacobian matrices

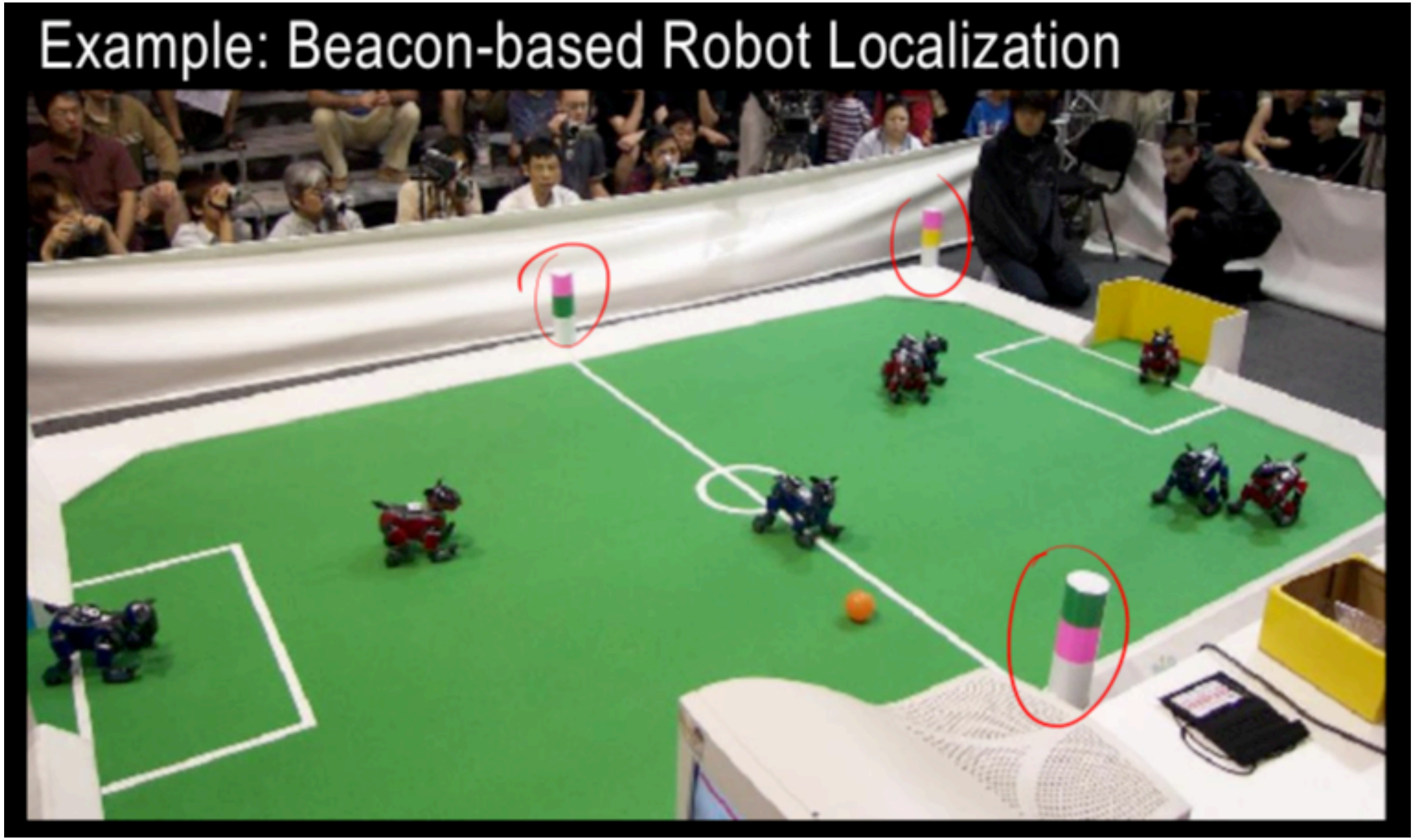
Once the motion and the observation models have been linearized, perform the similar updates as the Kalman Filter.

1. **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction:
3. $\bar{\mu}_t = g(u_t, \mu_{t-1})$ ⟵ $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$ ⟵ $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$
5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$ ⟵ $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$ ⟵ $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$ ⟵ $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
9. Return μ_t, Σ_t

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

Thrun et al. (Probabilistic Robotics) Ch 3 (Sec 3.3). Note the minor differences in notation from the previous slide.

Application



Application

Example Motion Model

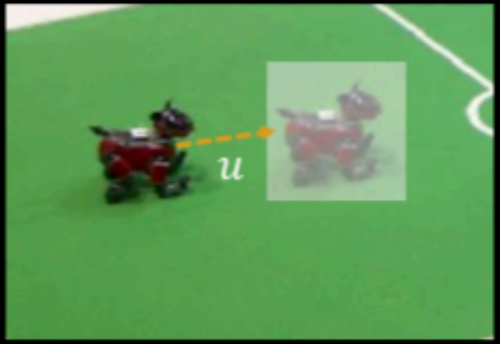
- State is $x_t = (x_t, y_t, \theta_t)$
- Command is rotation, translation, rotation

$$u_t = (\delta_{rot_1}, \delta_{trans}, \delta_{rot_2})$$

- Actual motion is $(\tilde{\delta}_{rot_1}, \tilde{\delta}_{trans}, \tilde{\delta}_{rot_2})$, a noisy version of the command

- Motion model g is:

$$x_{t+1} = x_t + \tilde{\delta}_{trans} \cos(\theta_t + \tilde{\delta}_{rot_1})$$
$$y_{t+1} = y_t + \tilde{\delta}_{trans} \sin(\theta_t + \tilde{\delta}_{rot_1})$$
$$\theta_{t+1} = \theta_t + \tilde{\delta}_{rot_1} + \tilde{\delta}_{rot_2}$$



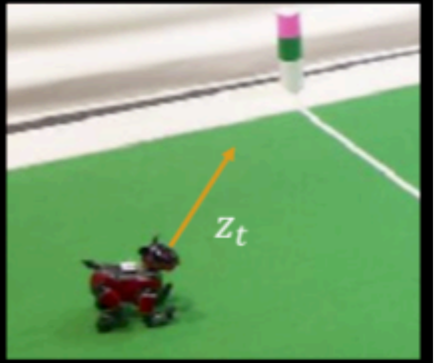
Application

Example sensor model

- The map is known
  - Beacons are at known positions
- Sensor reports noisy bearing $\tilde{\theta}$ and exact landmark ID L (discrete)
  - Only one beacon is observed at one time

$$z_t = \begin{pmatrix} \tilde{\theta} \\ L \end{pmatrix} = \begin{pmatrix} \text{atan2}(y_{rob} - y_L, x_{rob} - x_L) \\ L \end{pmatrix}$$

Not linear!

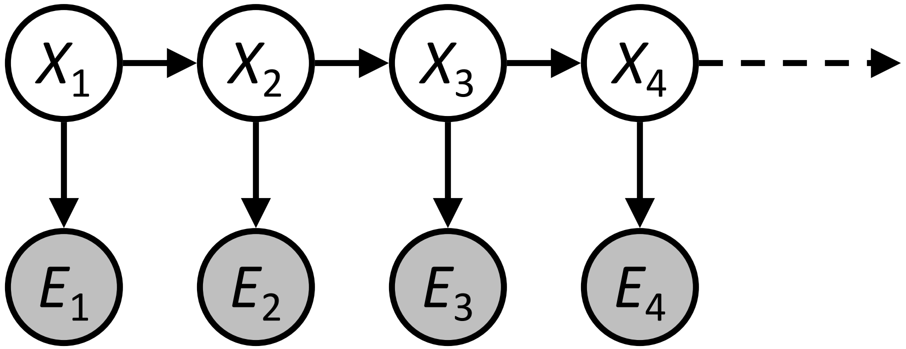


EKF: Other Takeaways

- Non-optimal.
  - EKF is *approximate* and can diverge if the non-linearities are large.
  - Note that Kalman Filter was the optimal filter.
- Effectiveness
  - Handles Non-Gaussian sensor and motion models.
  - Note: still does not handle multi-modality (other methods such as histogram filters and particle filters that address multi-modality).
- Efficient
  - Polynomial in the measurement dimensionality k and the state dimensionality n: $O(k^{2.376} + n^2)$

Hidden Markov Models

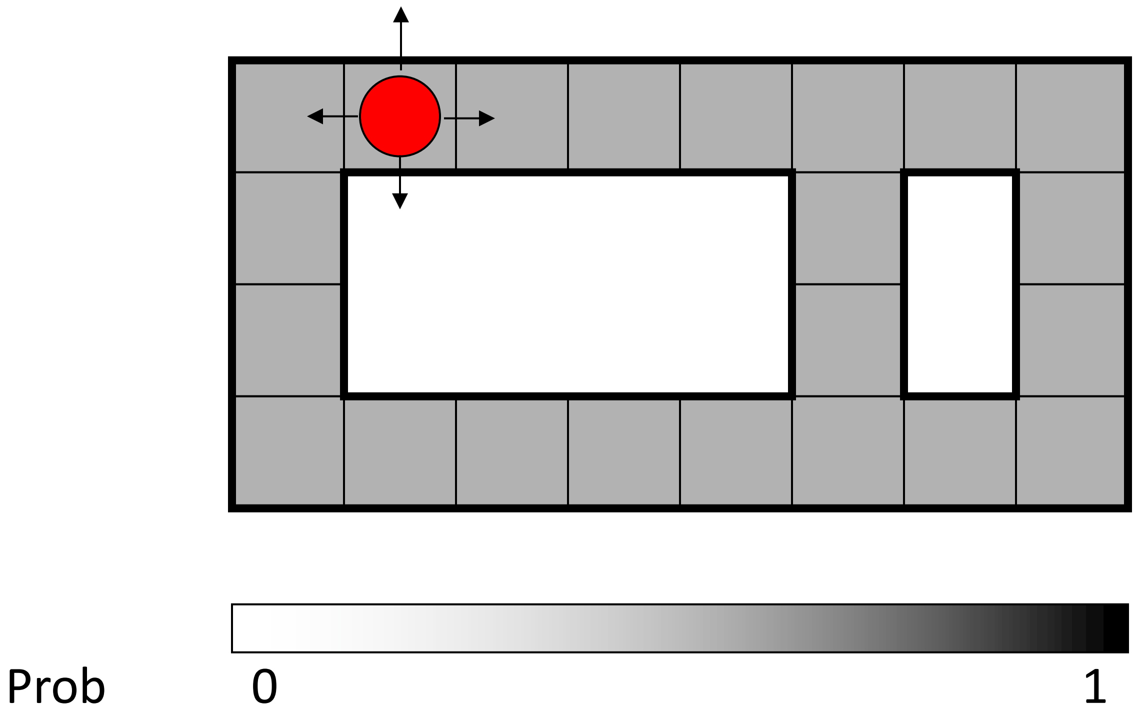
- No explicit notion of controls or actions
  - The state of the world changes with time.
  - Predict it with successive observations.
- Discrete states and observations
- Assumptions
  - Future depends on past via the present
  - Current observation independent of all else given current state



$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$$

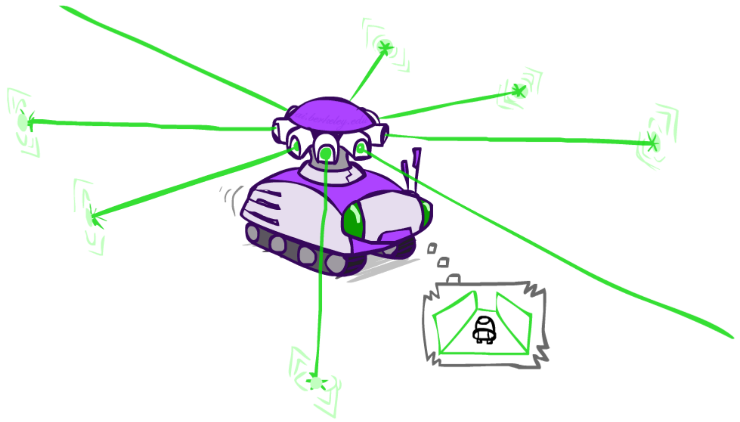
$$\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$$

Example: Robot Localization



t=0

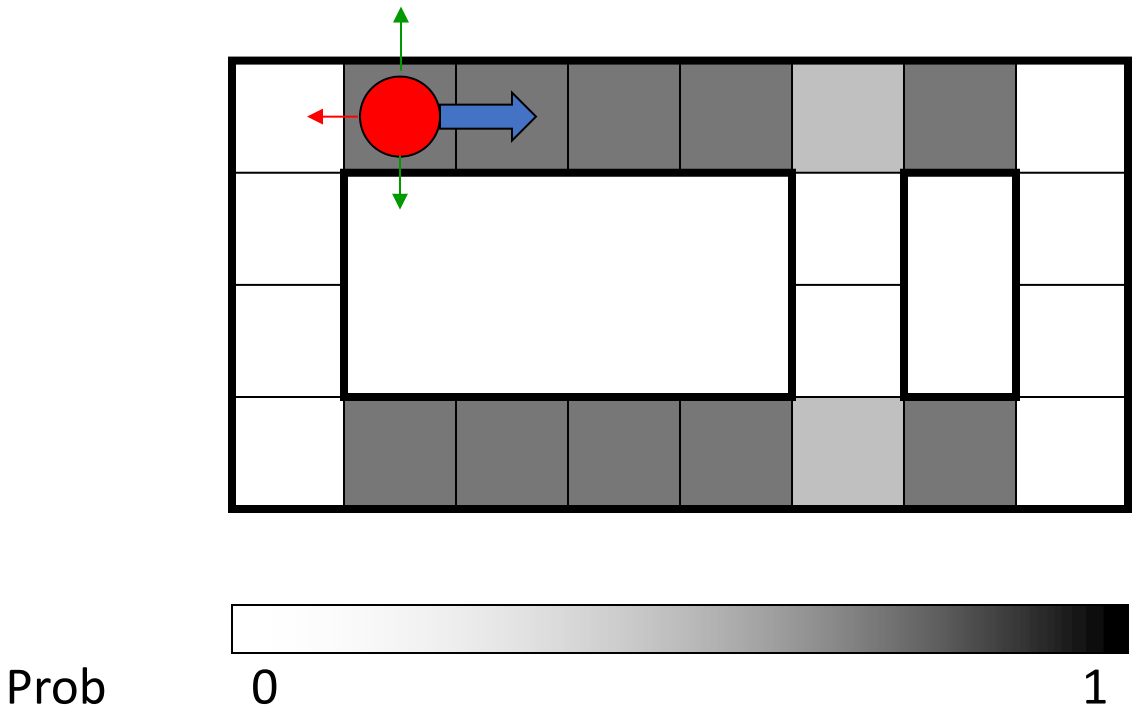
Robot can take actions N, S, E, W
Detects walls from its sensors

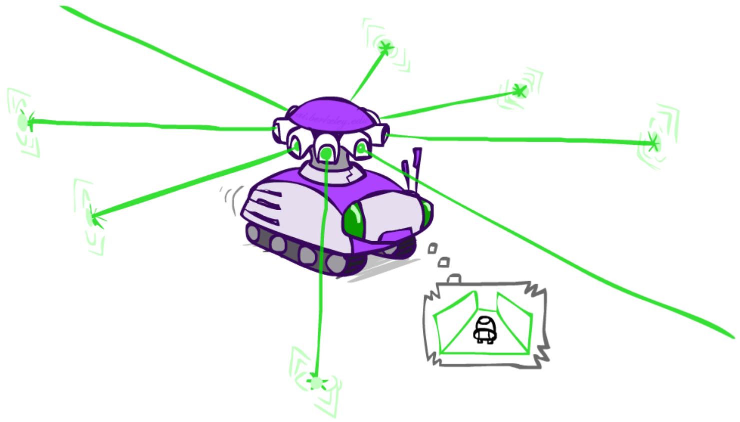


Sensor model: can read in which directions there is a wall, never more than 1 mistake

Motion model: may not execute action with small prob.

Example: Robot Localization

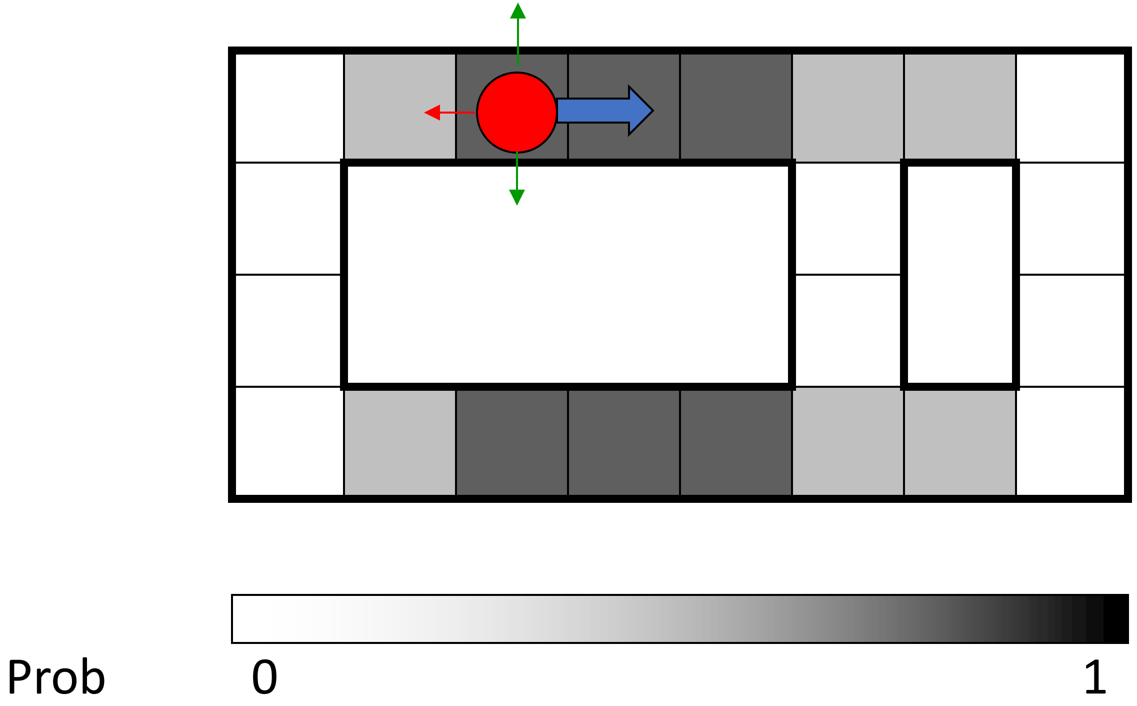


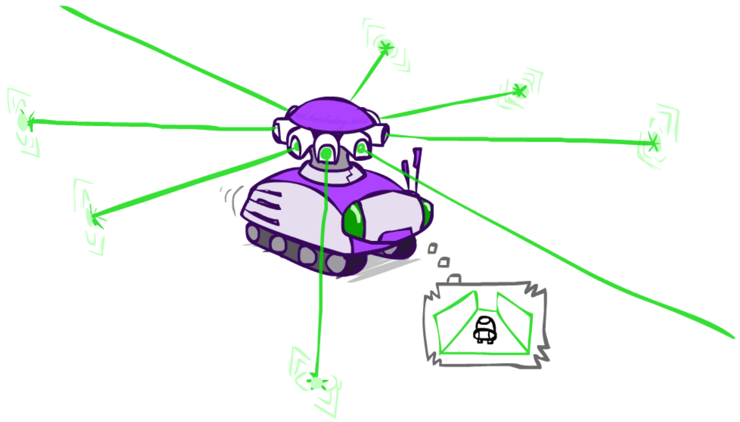


t=1

Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

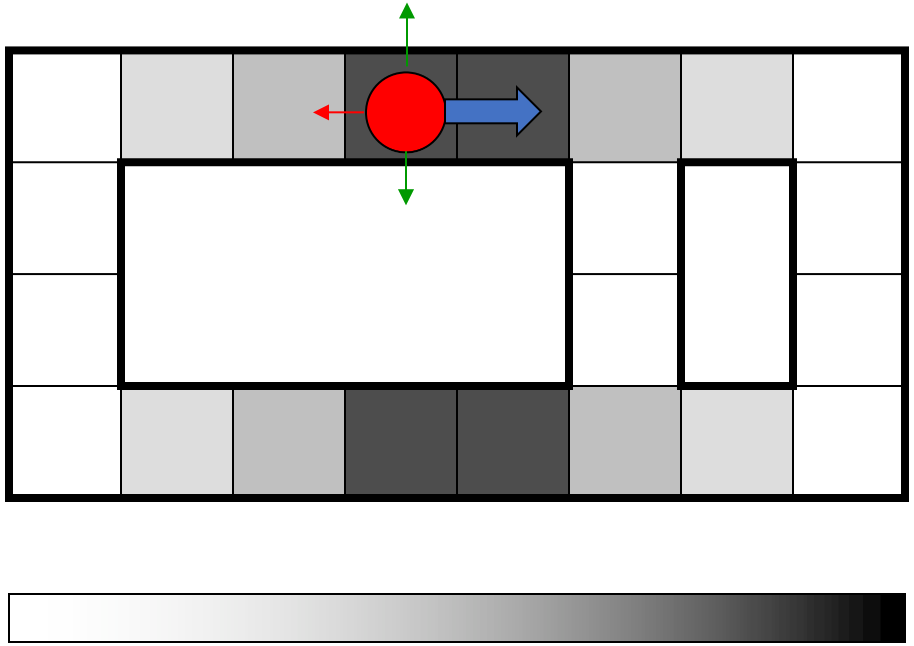
Example: Robot Localization



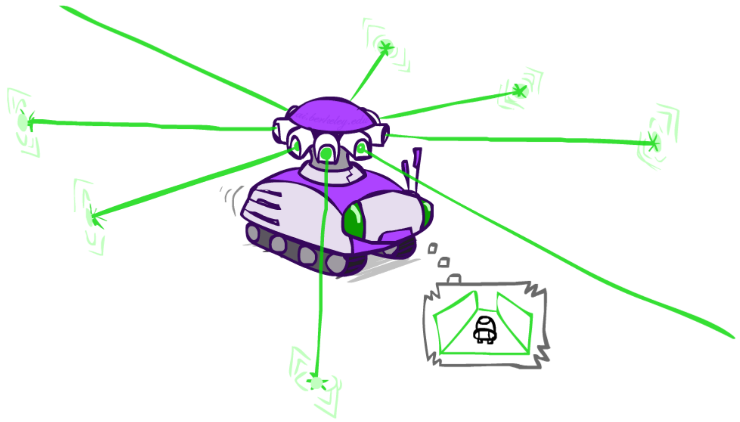


t=2

Example: Robot Localization

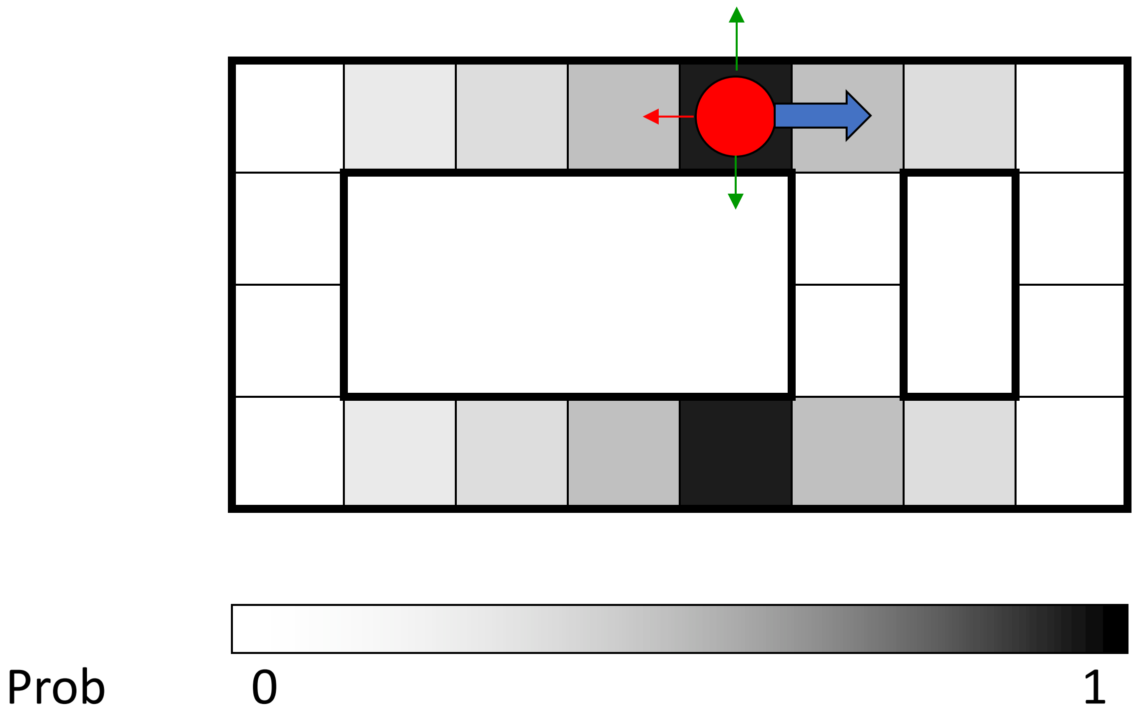


Prob 0 1

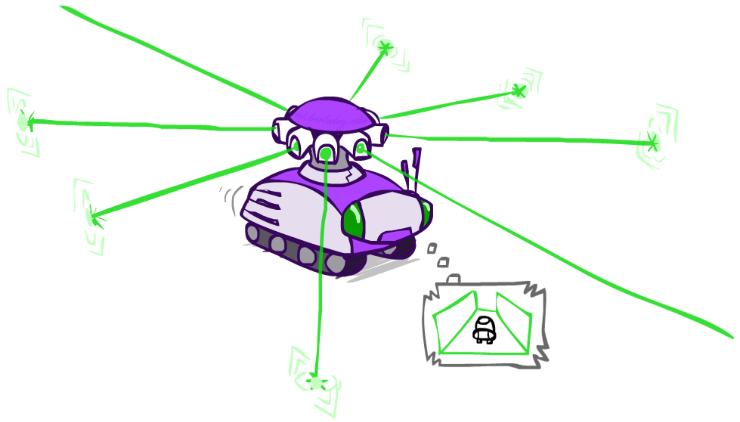


t=3

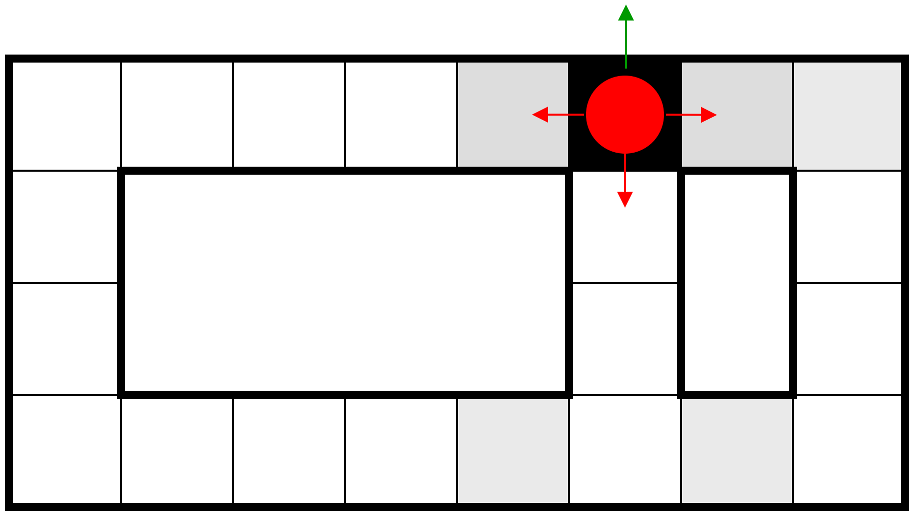
Example: Robot Localization

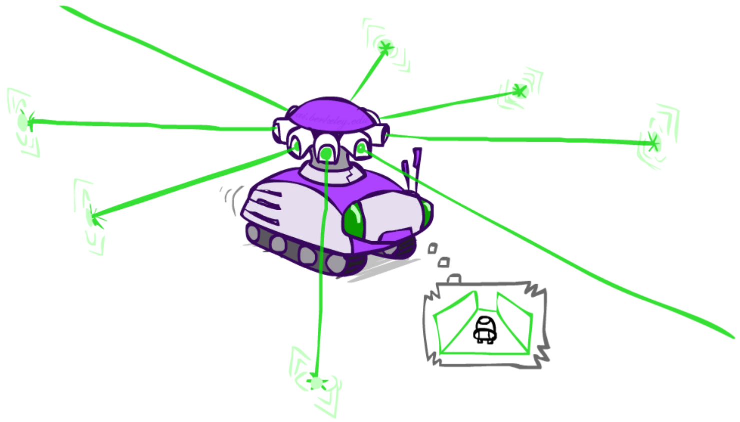


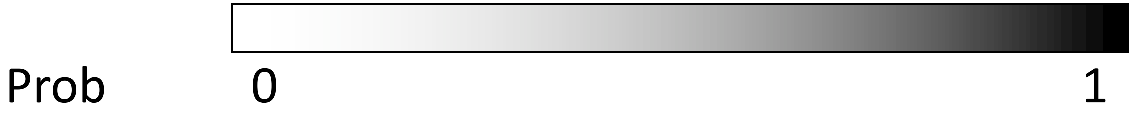
t=4



Example: Robot Localization







t=5

Range of Inference Tasks

Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$
 to compute the current belief state given all evidence
 better name: state estimation

Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$
 to compute a **future** belief state, given current evidence
 (it's like filtering without all evidence)

Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$
 to compute a better estimate of past states

Most likely explanation: $\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$
 to compute the state sequence that is most likely, given the evidence

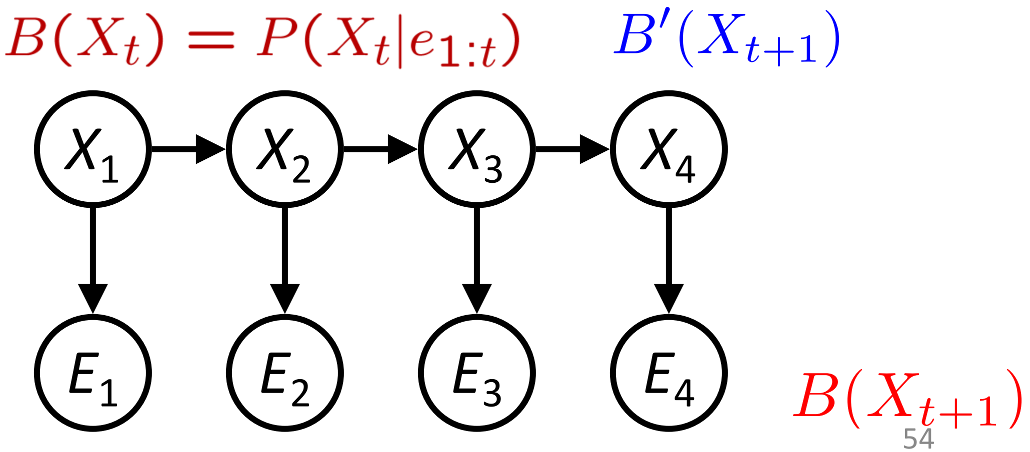
Inference: Estimate State Given Evidence

- We are given evidence at each time and want to know

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

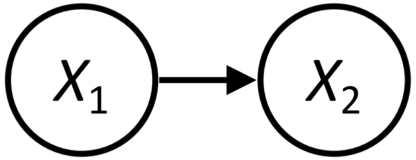
- Approach: start with $P(X_1)$ and derive B_t in terms of B_{t-1}
  - Equivalently, derive B_{t+1} in terms of B_t

- Two Steps:
  - Passage of time
  - Evidence incorporation



Passage of Time (Dynamics Update)

Assume we have current belief P(X | evidence to date)



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

Then, after one time step:

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

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

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

Basic idea: the beliefs get "pushed" through the transitions

Measurement Update

Assume we have current belief P(X | previous evidence):

$$B'(X_{t+1}) = P(X_{t+1}|e_{1:t})$$

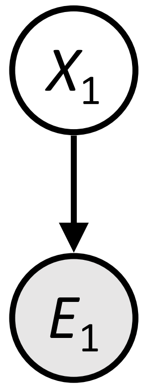
Then, after evidence comes in:

$$P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t})$$

$$\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t})$$

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

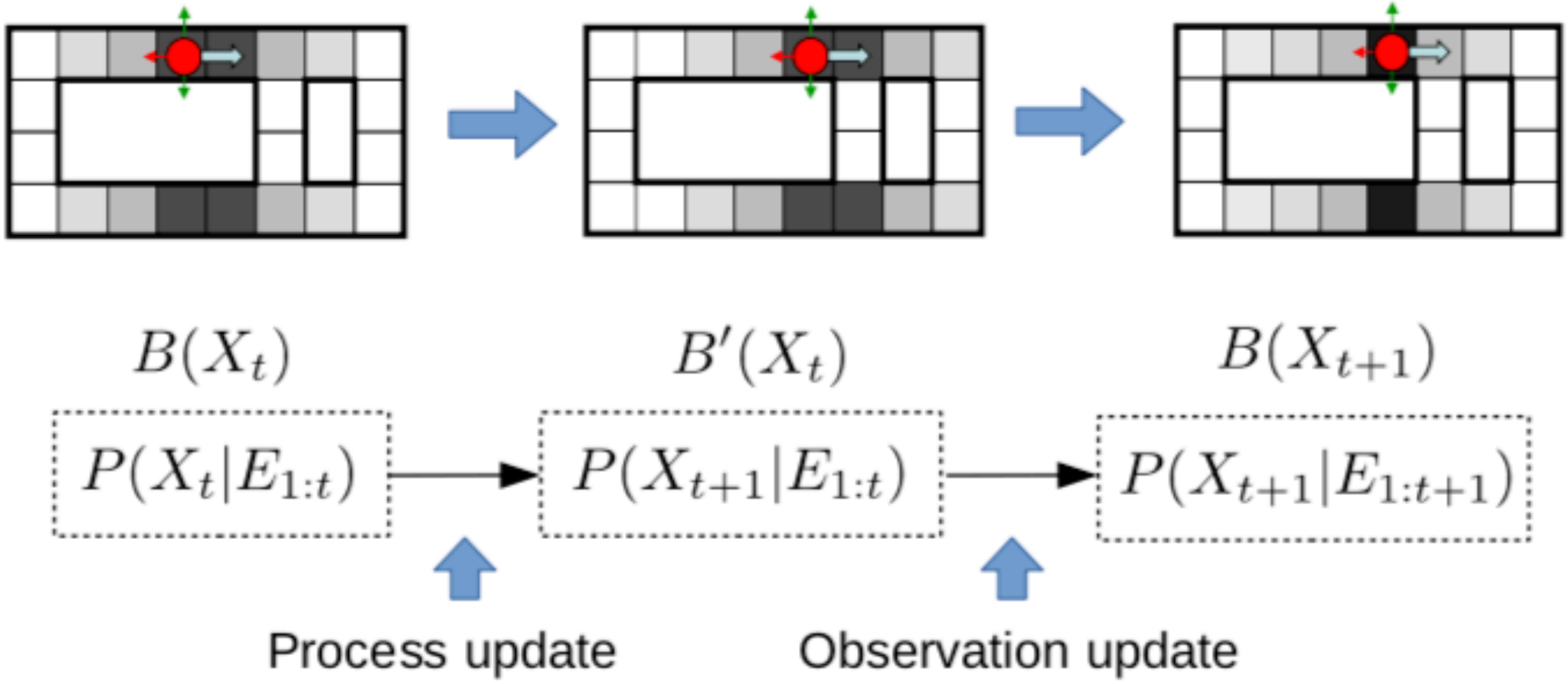
$$= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})$$



View it as a "correction" of the belief using the observation

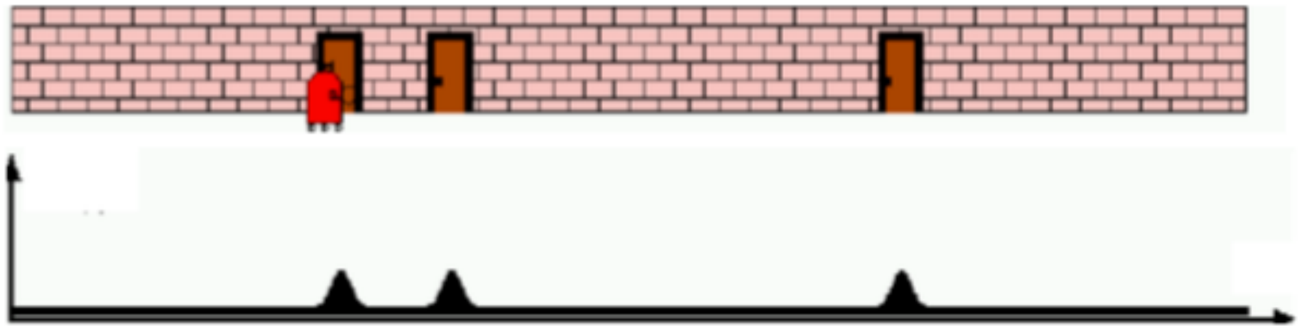
$$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1}|X_{t+1})B'(X_{t+1})$$

Dynamics Update and Measurement Update

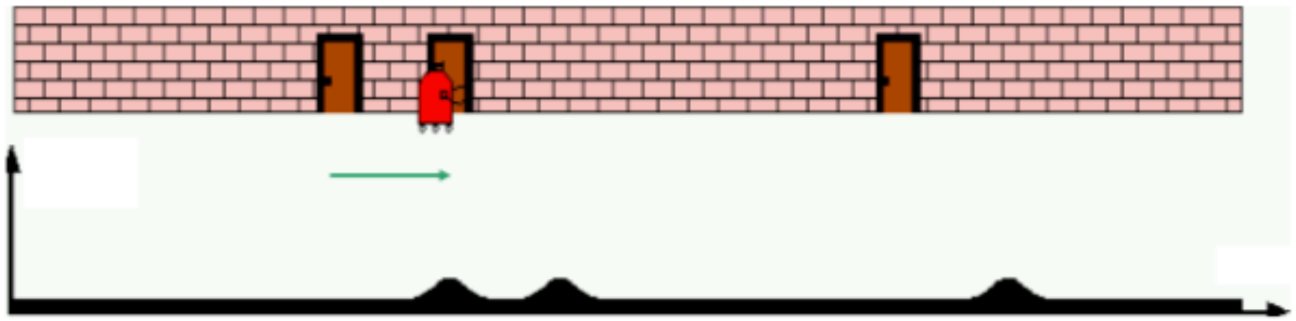


Dynamics Update and Measurement Update

Before process update



After process update

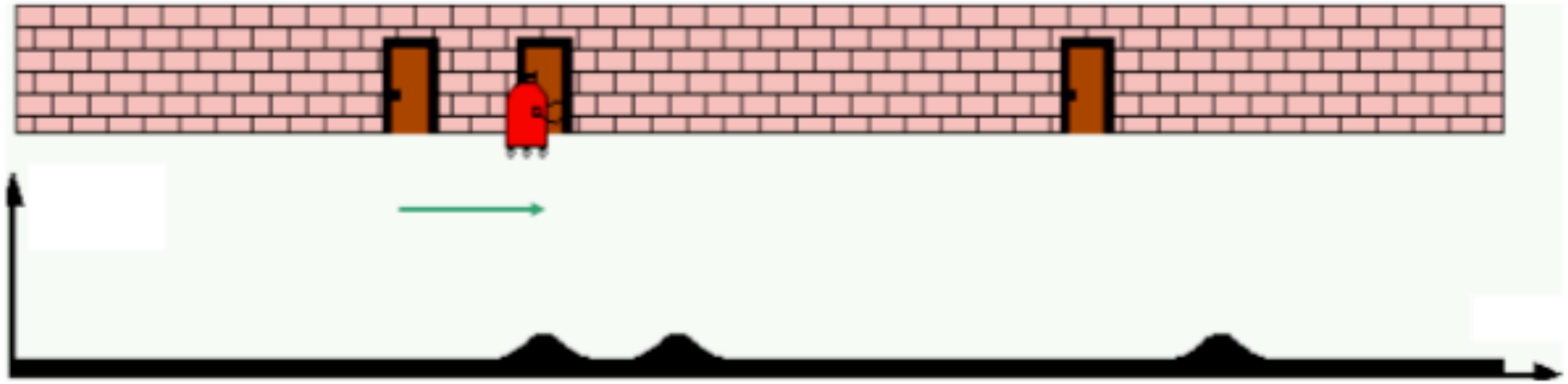


$$B'(X_{t+1}) = \sum_{X_t} P(X_{t+1}|X_t, e_{1:t})B(X_t)$$

This is a little like convolution...

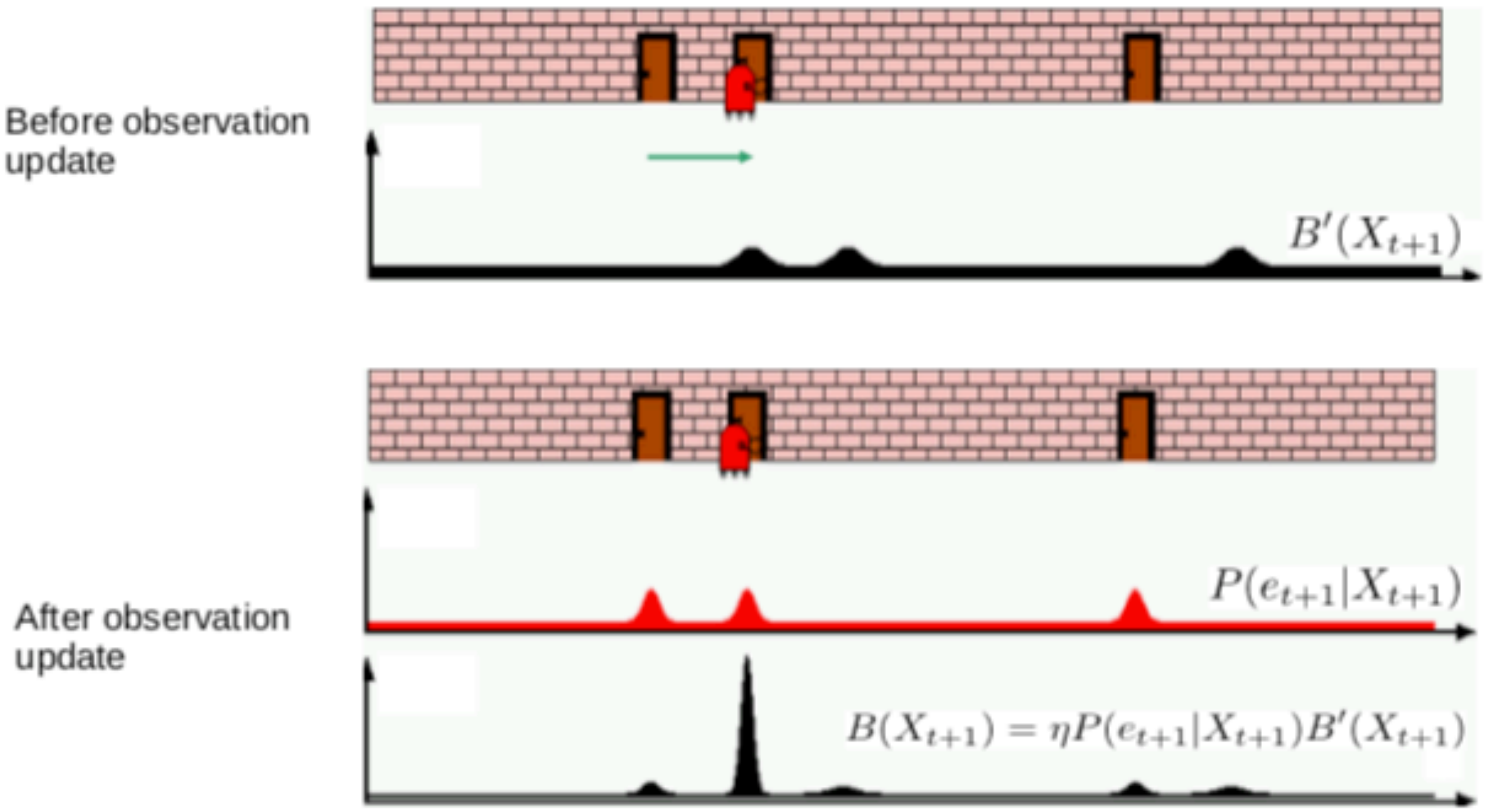
Dynamics Update and Measurement Update

After process update



Each time you execute a process update, belief gets more disbursed
- *i.e.* Shannon entropy increases
- this makes sense: as you predict state further into the future, your uncertainty grows.

Dynamics Update and Measurement Update

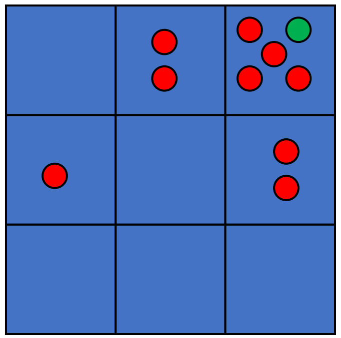


Particles in continuous space instead of grids

- Problem:
  - |X| may be too big to even store B(X)

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|

- P(x) approximated by number of particles with value x
  - Several x can have P(x) = 0. Note that (3,3) has half the number of particles.



Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Updating Particles

Each particle is moved by sampling its next position from the transition model

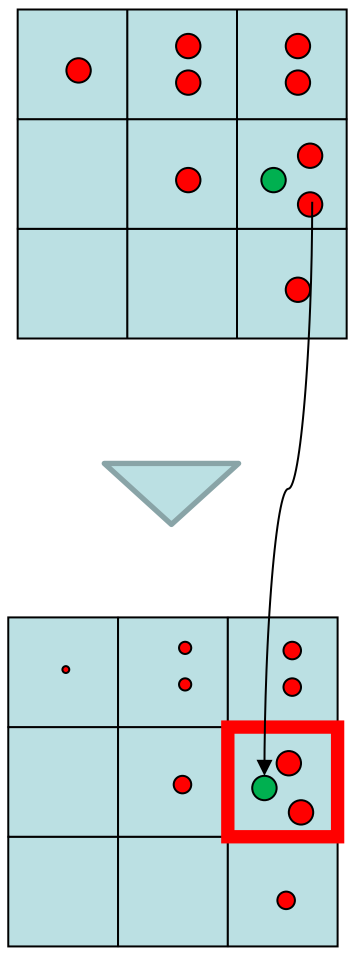
$$x' = \text{sample}(P(X'|x))$$

Attach a weight to each sample. Weigh the samples based on the likelihood of the evidence.

$$w(x) = P(e|x)$$

$$B(X) \propto P(e|X)B'(X)$$

Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



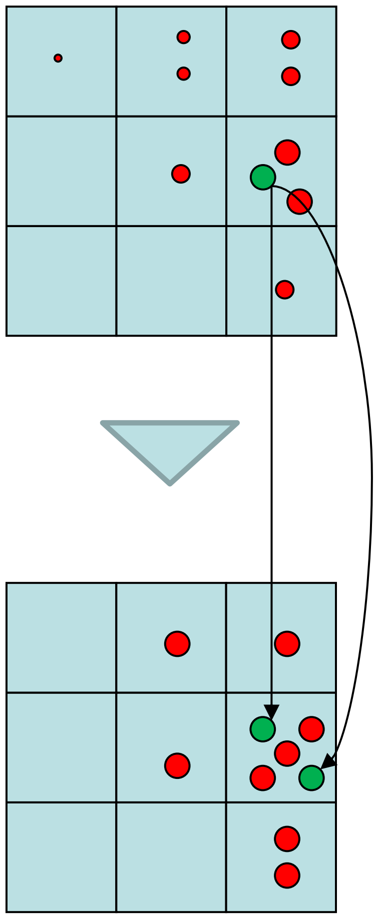
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

Resampling Particles

- Resample particles
  - Sample N times, from the weighted sample distribution (i.e. draw **with** replacement)
- Key idea:
  - maintain hypotheses (particles) in the region of probable states, discard others. Note that the sampling is with replacement.

Particles:

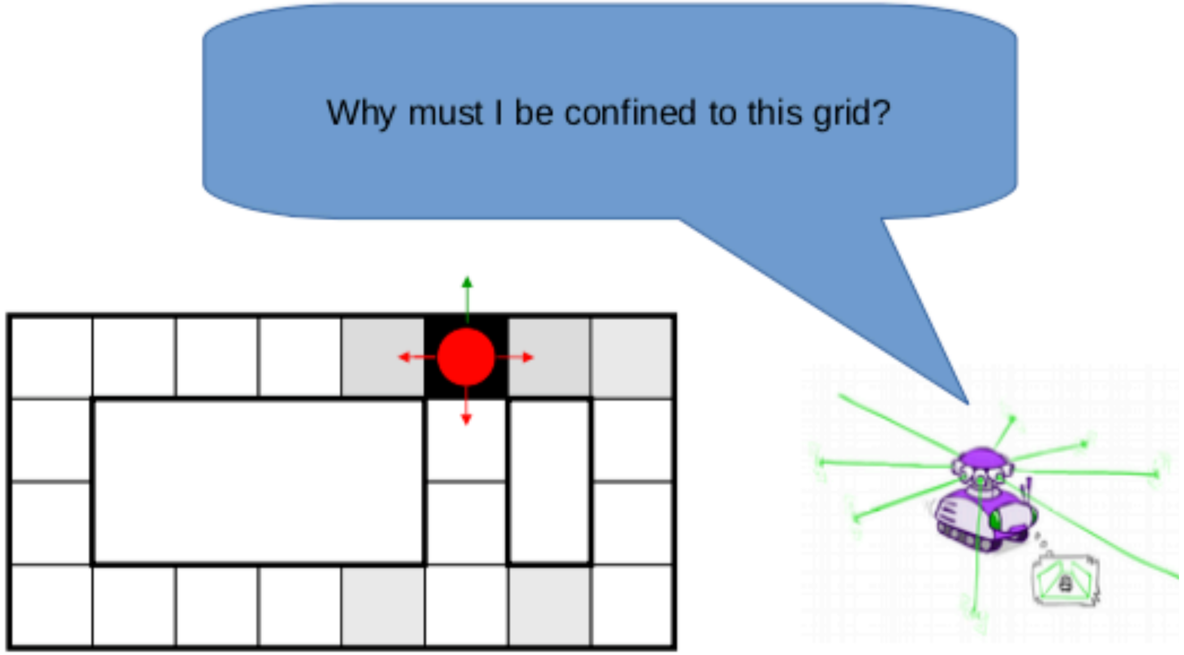
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

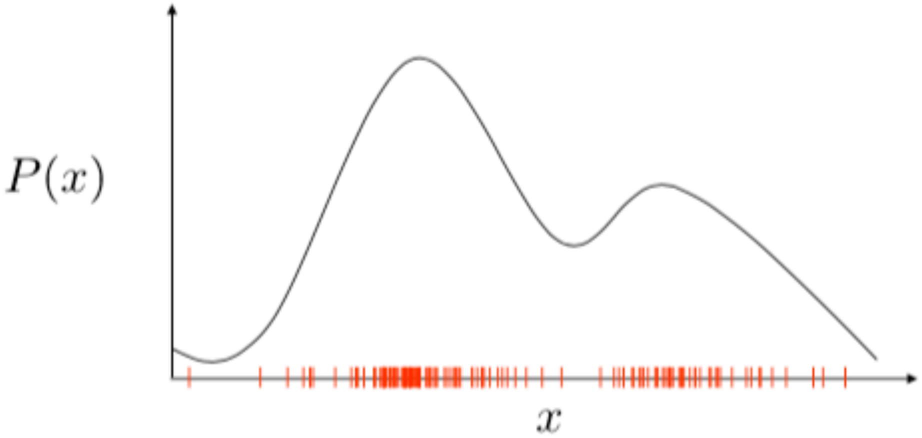


(New) Particles:

(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)

Belief over continuous space & multi-modality





Standard Bayes filtering requires discretizing state space into grid cells

Can do Bayes filtering w/o discretizing?

– yes: particle filtering or Kalman filtering

Key idea: represent a probability distribution as a finite set of points

– density of points encodes probability mass.

– particle filtering is an adaptation of Bayes filtering to this particle representation

Particle Filtering

$B(X_t)$ $\quad P(X_t|E_{1:t})$

↓

$B'(X_t)$ $\quad P(X_{t+1}|E_{1:t})$

↓

$B(X_{t+1})$ $\quad P(X_{t+1}|E_{1:t+1})$

Do this *n* times →

Prior distribution

$$x_t^1, \ldots, x_t^n \qquad w_t^1, \ldots, w_t^n = 1$$

Process update

$$\bar{x}_{t+1}^i \sim P(X_{t+1}|x_t^i, e_{1:t})$$

Observation update

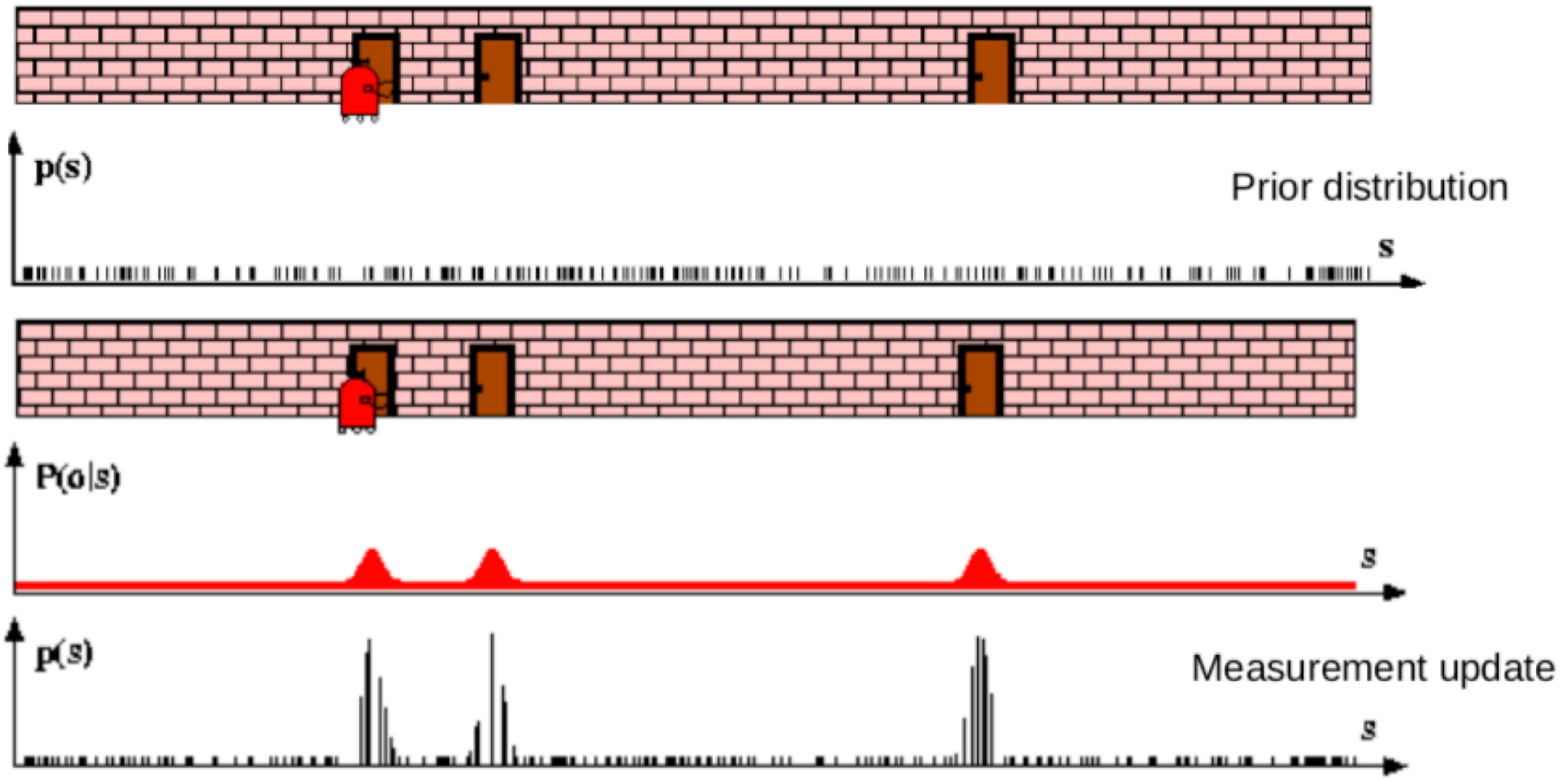
$$w_{t+1}^i = P(e_{t+1}|\bar{x}_{t+1}^i)w_t^i$$

Resample

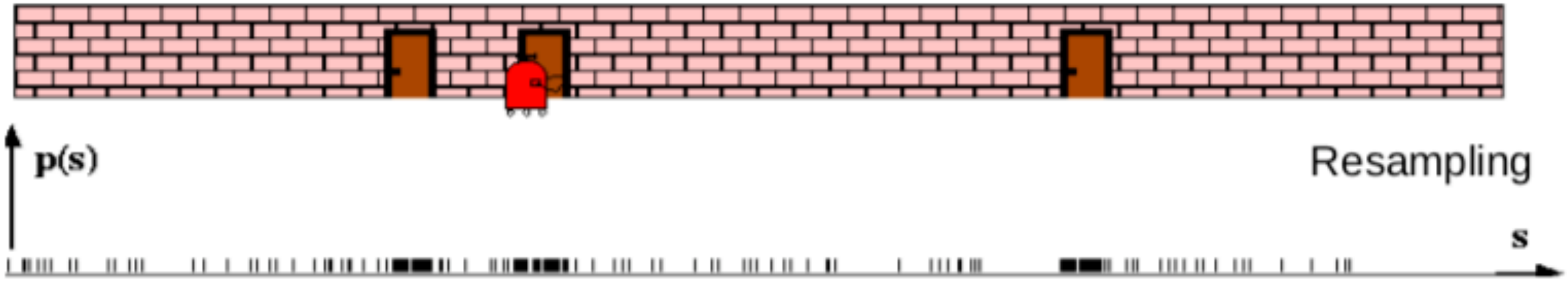
$$X_{t+1} = \{\}$$

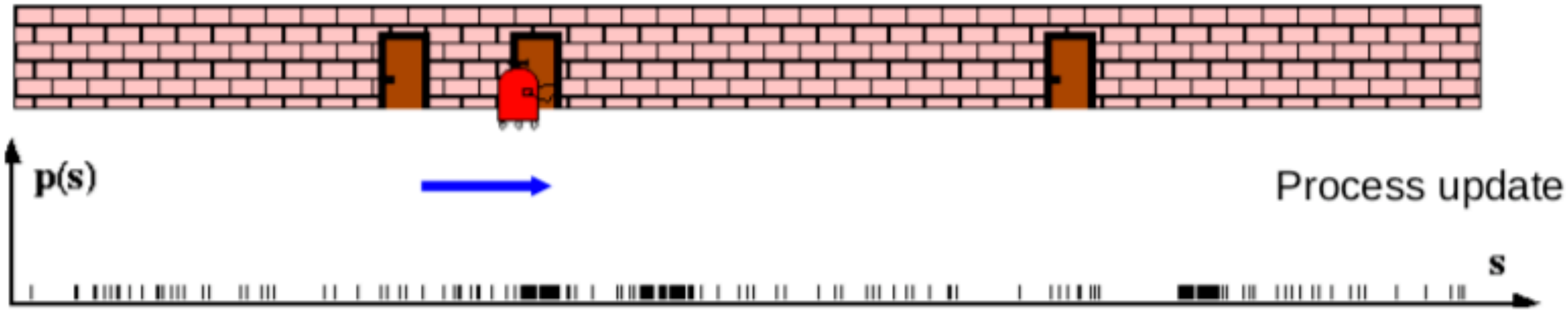
$X_{t+1} = X_{t+1} \cup \bar{x}_{t+1}^i$ w/ prob w_{t+1}^i

Example: Measurement Update to Particles



Example: Resampling and Process Update






Global localization with
sonar sensors
40000