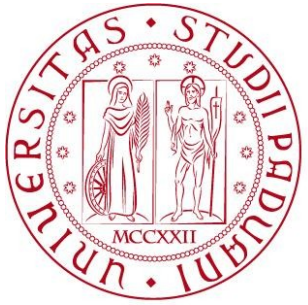


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

IAS-Lab
Intelligent Autonomous
Systems Laboratory

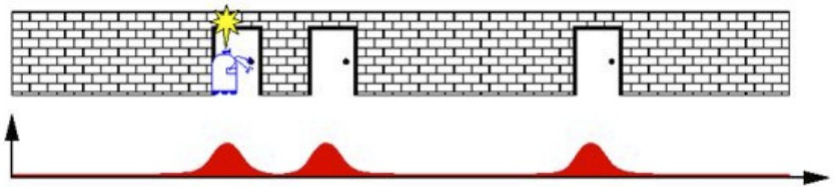
A Very Brief Introduction to the Localization and SLAM Problems
Part 2

Alberto Pretto

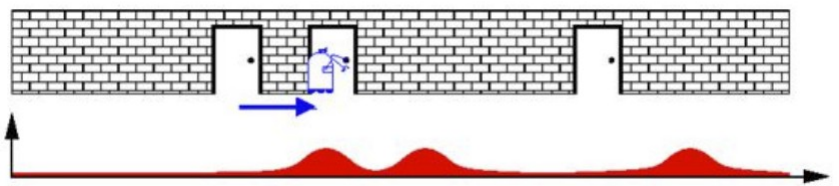
Thanks to Wolfram Burgard, Giorgio Grisetti, Davide Scaramuzza and Cyrill Stachniss for some slides!

Bayes Filter Recap

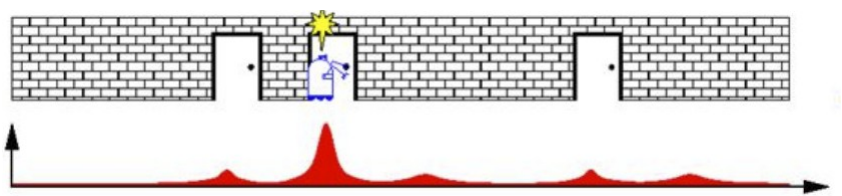
- Represent the state in a probabilist way (e.g., in a discrete space, a probability value for each possible state realization
  $\boldsymbol{Bel(x_i) = p(x_i)}$
- Update (i.e., ~"**move**") the probability density using the actions (e.g., motion)
  $\boldsymbol{Bel'(x_{i+1}) = \int P(x_{i+1} \mid u, x_i)\, Bel(x_i)\, dx_i}$
- Update (i.e., ~"**re-weight**", **or** "**correct**") the probability density using the observations (sensory measurements)
  $\boldsymbol{Bel(x_{i+1}) = \eta\, P(z \mid x_{i+1})\, Bel'(x_{i+1})}$



Prediction

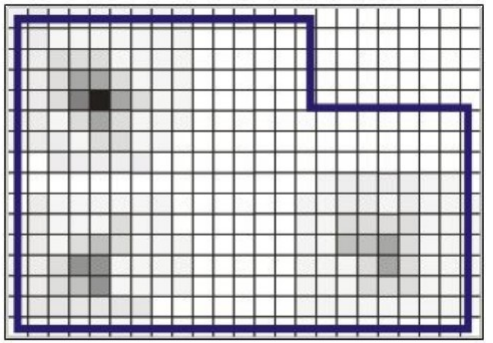


Correction (or Update)



How to Represent the State?

- So far: discrete way (i.e., "grid based")
  - Non parametric



- Why not in an analytical way?

Prediction

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

Correction

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

Intractable!

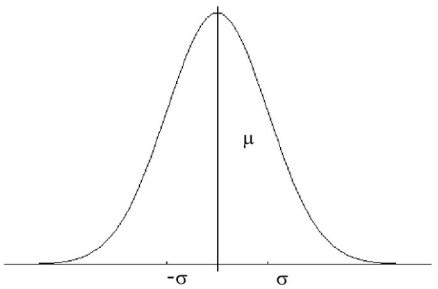
Gaussian Densities

Parametric, unimodal densities with special properites

- Univariate

$$p(x) \sim N(\mu, \sigma^2):$$

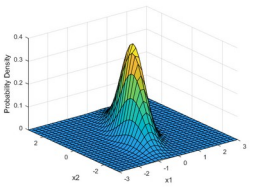
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

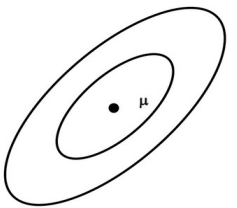


- Multivariate

$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$





Covariance Matrix

- Given a n-dimensional random vector X with Gaussian density, define μ_k the expected value (***mean***) of the k^{th} element of X, i.e., $\mu_k = E(X_k)$, with $\boldsymbol{\mu} = E(\boldsymbol{X})$
- The covariance matrix is defined as:

$$\Sigma = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])\,(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\top}\right]$$

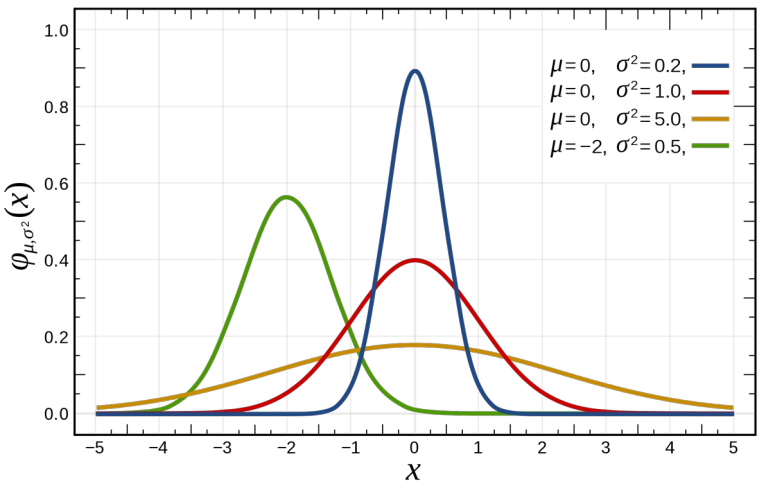
$$= \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}$$

Covariance Matrix

- The *(i,i)* element is the variance of X_i
- The *(i,j)* element is the covariance between X_i and X_j:
  - if positive, if X_i increases, also X_j *tends to increases*
  - if negative, if X_i increases, X_j *tends to decreases (and vice versa)*
  - If zero, X_i and X_j are independent hence uncorrelated.

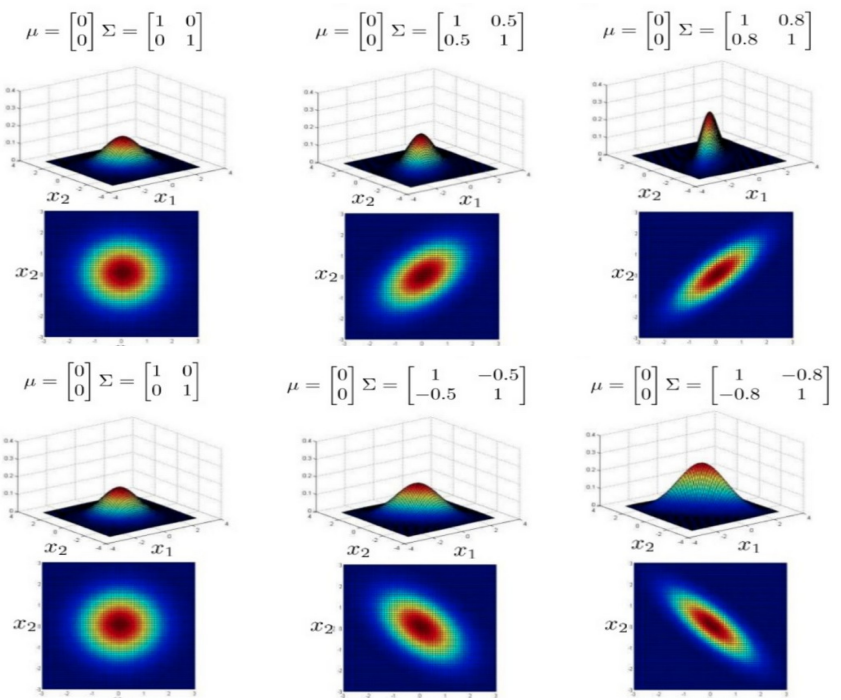
Gaussian Densities

- Univariate



[Source: Wikipedia]

- Bivariate



[Source: Anomaly detection, David Khosid]

Properties of Gaussians

Linear combinations of Gaussian random varabiles (or vectors) are still Gaussian

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

Products of Gaussian probability density functions are still Gaussian PDF

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

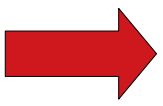
Properties of Gaussians

Chain rule and linear combinations for Gaussian variables/vectors

- General chain rule : $\mathrm{P}(X, Y) = \mathrm{P}(X \mid Y) \cdot P(Y)$
- Given: $p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a; \mu_a, \mathbf{\Sigma}_a).$

  $p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_b; \mathbf{A}\mathbf{x}_a + \mathbf{c}, \mathbf{\Sigma}_{b|a})$
- We want to compute: $p(\mathbf{x}_a, \mathbf{x}_b) = \mathcal{N}\left(\mathbf{x}_{a,b}; \mu_{a,b}, \mathbf{\Sigma}_{a,b}\right)$

  The parameters are:



$$\mu_{a,b} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} \mu_a \\ \mathbf{A}\mu_a + \mathbf{c} \end{pmatrix}$$

$$\mathbf{\Sigma}_{a,b} = \begin{pmatrix} \mathbf{\Sigma}_a & \mathbf{\Sigma}_a\mathbf{A}^T \\ \mathbf{A}\mathbf{\Sigma}_a & \mathbf{\Sigma}_{b|a} + \mathbf{A}\mathbf{\Sigma}_a\mathbf{A}^T \end{pmatrix}$$

Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by a linear (or linearized) stochastic equation:

$$x_t = A_t x_{t-1}$$

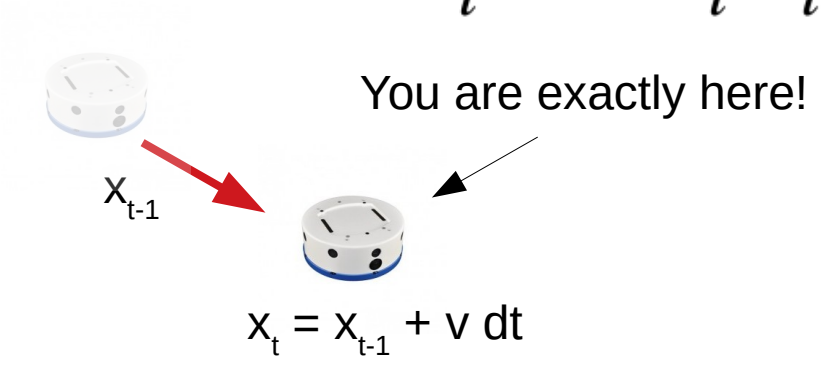
$x_t = x_{t-1}$

Matrix (*nxn*) that describes how the state evolves from *t-1* to *t* without controls or noise. E.g., **In many cases is just an identity matrix**!

Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by a linear (or linearized) stochastic equation:

$$x_t = A_t x_{t-1} + B_t u_t$$

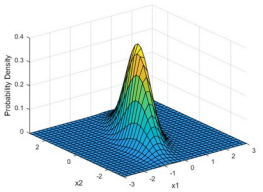


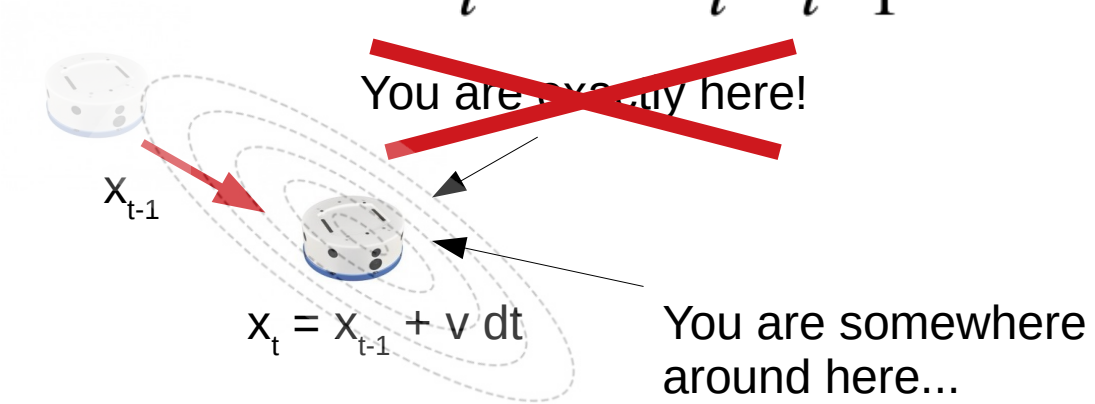
Matrix (*nxl*) that describes how the control u_t changes the state from *t-1* to *t*. E.g., u_t may be a speed command, B_t "maps" such command into a state x_t update.

Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by a linear (or linearized) **stochastic** equation:

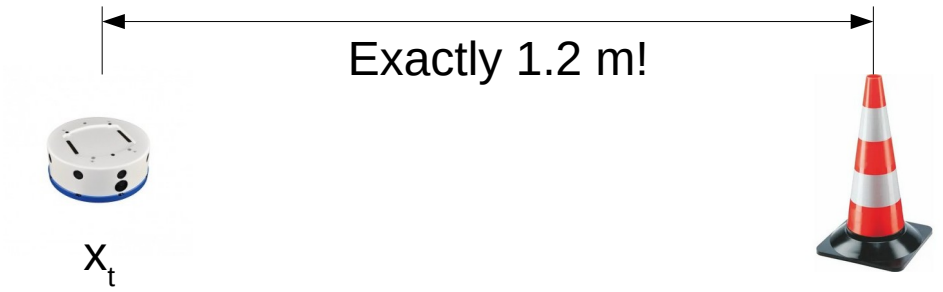
$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$





Normally distributed (i.e., **Gaussian**) random variable representing the process ("transition") noise with covariance R_t

Kalman Filter



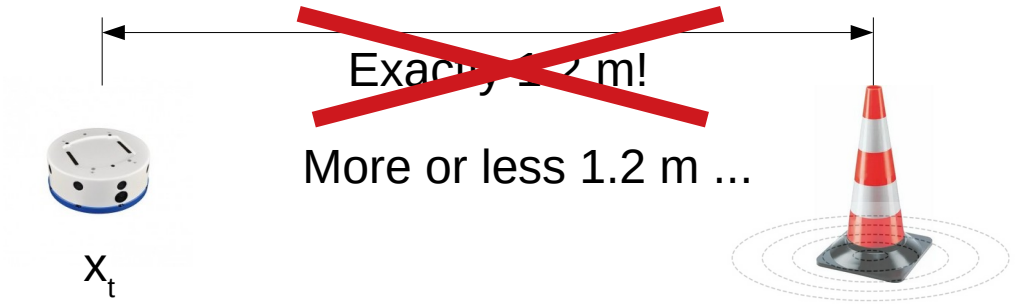
with stochastic measurements:

''Given a landmark with known position (I have the map!), I want to calculate the measurement I would get from my sensor given the current robot position x_t"

$$z_t = C_t x_t$$

Matrix (*kxn*) that describes how to map the state x_t to an observation z_t

Kalman Filter

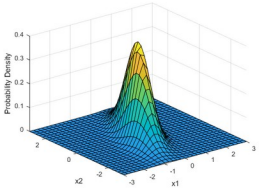


with **stochastic** measurements:

''Given a landmark with known position (I have the map!), I want to calculate the measurement I would get from my sensor given the current robot position x_t"

$$z_t = C_t x_t + \delta_t$$

Normally distributed (i.e., **Gaussian**) random variable representing the measurement noise with covariance Q_t



Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by a linear (or linearized) stochastic equation:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

with stochastic measurements:

$$z_t = C_t x_t + \delta_t$$

Kalman Filter

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

$$z_t = C_t x_t + \delta_t$$

Kalman Filter

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

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

$$z_t = C_t x_t + \delta_t$$

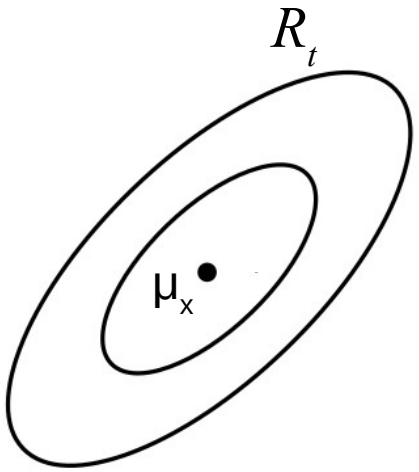
$$p(z_t \mid x_t)$$ ➡ Sensor model

Kalman Filter

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

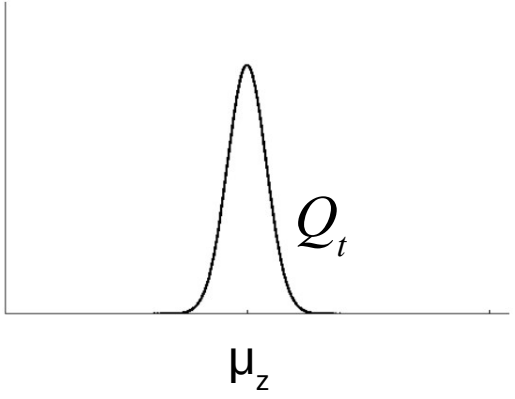
$$p(z_t \mid x_t)$$

Kalman Filter



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

$$= N(x_t; \underbrace{A_t x_{t-1} + B_t u_t}_{\mu_x}, R_t)$$



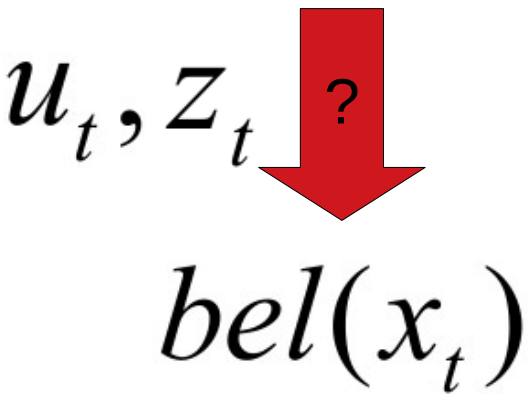
$$p(z_t \mid x_t)$$

$$= N(z_t; \underbrace{C_t x_t}_{\mu_z}, Q_t)$$

Kalman Filter

Given a **Gaussian** belief at time *t-1*, how to incorporate actions and sensor measurements to obtain the belief at time *t*?

$$bel(x_{t-1}) = N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$



Kalman Filter: Prediction

$$p(x_t \mid u_t, x_{t-1}) = N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right)$$

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

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

Kalman Filter: Prediction

Derivation 1 (intuition): solve analytically

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

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

$$\Downarrow$$

$$\overline{bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right\}$$

$$\exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\} dx_{t-1} \quad \ldots\ldots\ldots\ldots\ldots$$

……

Kalman Filter: Prediction

Derivation 1 (intuition): solve analytically

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

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

$$\Downarrow$$

$$\overline{bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right\}$$

$$\exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\} dx_{t-1} \quad \ldots\ldots\ldots\ldots\ldots$$

$\ldots\ldots\ldots\ldots\ldots$

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

Kalman Filter: Prediction

Derivation 2 (intuition): exploit the fact that any affine transformation of a (multivariate) Gaussian random variable is (multivariate) Gaussian:

$$\left.\begin{array}{l} x_{t-1} \rightarrow X \sim N(\mu, \Sigma) \\ x_t \rightarrow Y = AX + B \end{array}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

We have:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

$$bel(x_{t-1}) = N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

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

Action noise

Kalman Filter: Corretion

$$p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)$$

$$bel(x_t) = \eta \; p(z_t \mid x_t) \qquad \overline{bel}(x_t)$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N(z_t; C_t x_t, Q_t) \qquad \sim N(x_t; \overline{\mu}_t, \overline{\Sigma}_t)$$

Kalman Filter: Corretion

Derivation 1 (intuition): solve analytically

$$bel(x_t) = \eta \quad p(z_t \mid x_t) \qquad \overline{bel}(x_t)$$

$$\Downarrow \qquad \Downarrow$$

$$\sim N(z_t; C_t x_t, Q_t) \qquad \sim N(x_t; \overline{\mu}_t, \overline{\Sigma}_t)$$

$$\Downarrow$$

$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \exp\left\{-\frac{1}{2}(x_t - \overline{\mu}_t)^T \overline{\Sigma}_t^{-1}(x_t - \overline{\mu}_t)\right\}$$

....

Kalman Filter: Corretion

Derivation 1 (intuition): solve analytically

$$bel(x_t) = \eta \quad p(z_t \mid x_t) \qquad \overline{bel}(x_t)$$

$$\Downarrow \qquad \Downarrow$$

$$\sim N(z_t; C_t x_t, Q_t) \qquad \sim N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)$$

$$\Downarrow$$

$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \exp\left\{-\frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\right\}$$

....

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

Kalman Filter: Corretion

Derivation 2 (intuition 1/2): exploit the chain rule:

$$\mathrm{P}(X,Y) = \mathrm{P}(X \mid Y) \cdot P(Y)$$

to compute

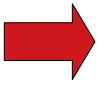
$$p(\mathbf{z}_t, \mathbf{x}_t)$$

Given:

$$\overline{bel}(x_t) \quad \text{and} \quad p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)$$

Remembering that:

$$p(\mathbf{x}_a, \mathbf{x}_b) = \mathcal{N}(\mathbf{x}_{a,b}; \mu_{a,b}, \boldsymbol{\Sigma}_{a,b})$$



$$\mu_{a,b} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} \mu_a \\ \mathbf{A}\mu_a + \mathbf{c} \end{pmatrix}$$

$$\boldsymbol{\Sigma}_{a,b} = \begin{pmatrix} \boldsymbol{\Sigma}_a & \boldsymbol{\Sigma}_a \mathbf{A}^T \\ \mathbf{A}\boldsymbol{\Sigma}_a & \boldsymbol{\Sigma}_{b|a} + \mathbf{A}\boldsymbol{\Sigma}_a \mathbf{A}^T \end{pmatrix}$$

Kalman Filter: Corretion

Derivation 2 (intuition, 2/2): once we have:

$$p(\mathbf{z}_t, \mathbf{x}_t)$$

Condition on the **current** measurement (e.g., z_t = 1.3 m) to obtain:

$$p(\mathbf{x}_t \mid \mathbf{z}_t)$$

Current measurement

Predicted measurement

And obtain:

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}$$

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

Kalman gain

Intuition: The ''smaller'' Q_t is (i.e., a less noisy sensor), the larger K_t is

Kalman Filter Algorithm

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

2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

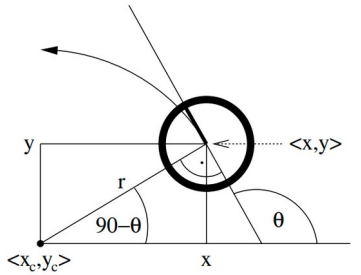
5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9. Return μ_t, Σ_t

Localization: An example

Velocity motion model control:

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix}$$



After time Δt of motion, our ideal robot will be at:

$$\mathbf{x}_{t-1} \rightarrow \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ y_c - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix}$$

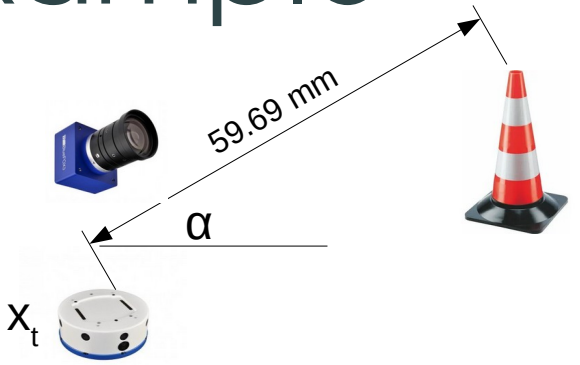
$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix} \rightarrow \mathbf{x}_t$$

Non linear relationship between x_{t-1} , u_t and x_t !

~~$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$~~

Localization: An example

Landmark detection: provide both distance and angle from a landmark (e.g., by using an RGB-D camera).



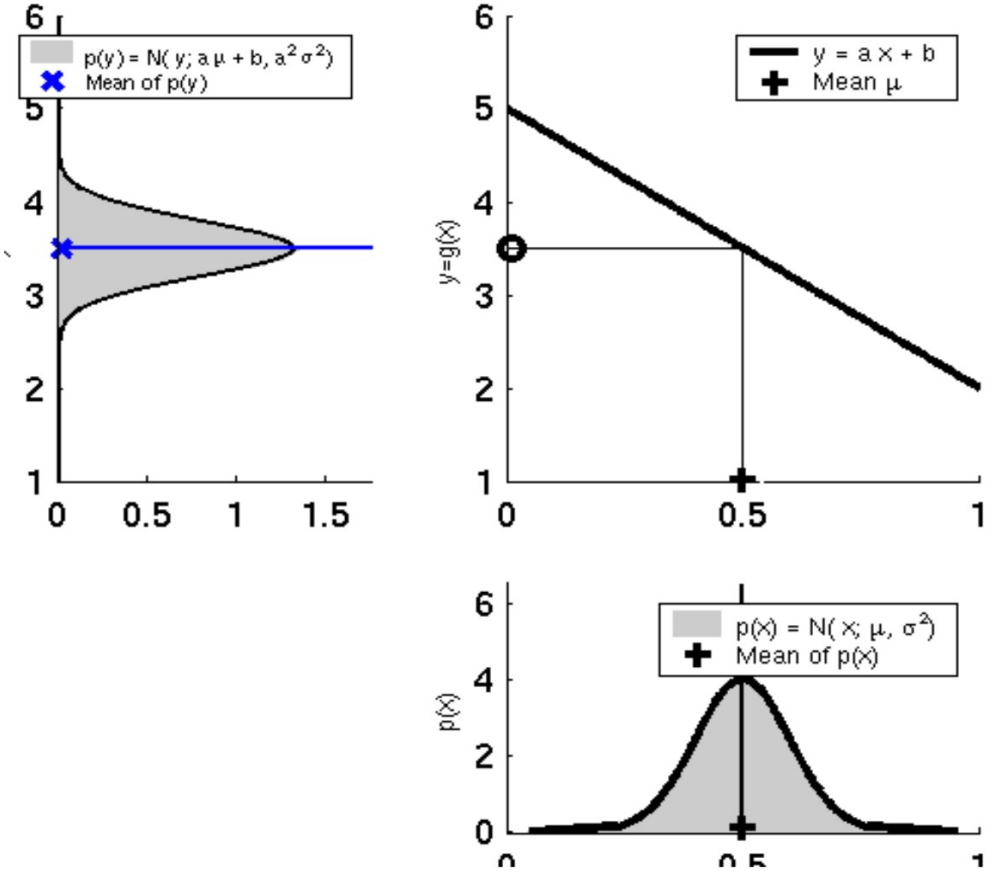
I know the landmark position (m_x, m_y) in my map: let's calculate the measurement the robot should get at the current (estimated) position

$$\hat{z}_t = \begin{pmatrix} \sqrt{(m_x - \bar{\mu}_{t,x})^2 + (m_y - \bar{\mu}_{t,y})^2} \\ \operatorname{atan2}(m_y - \bar{\mu}_{t,y}, m_x - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \end{pmatrix}$$

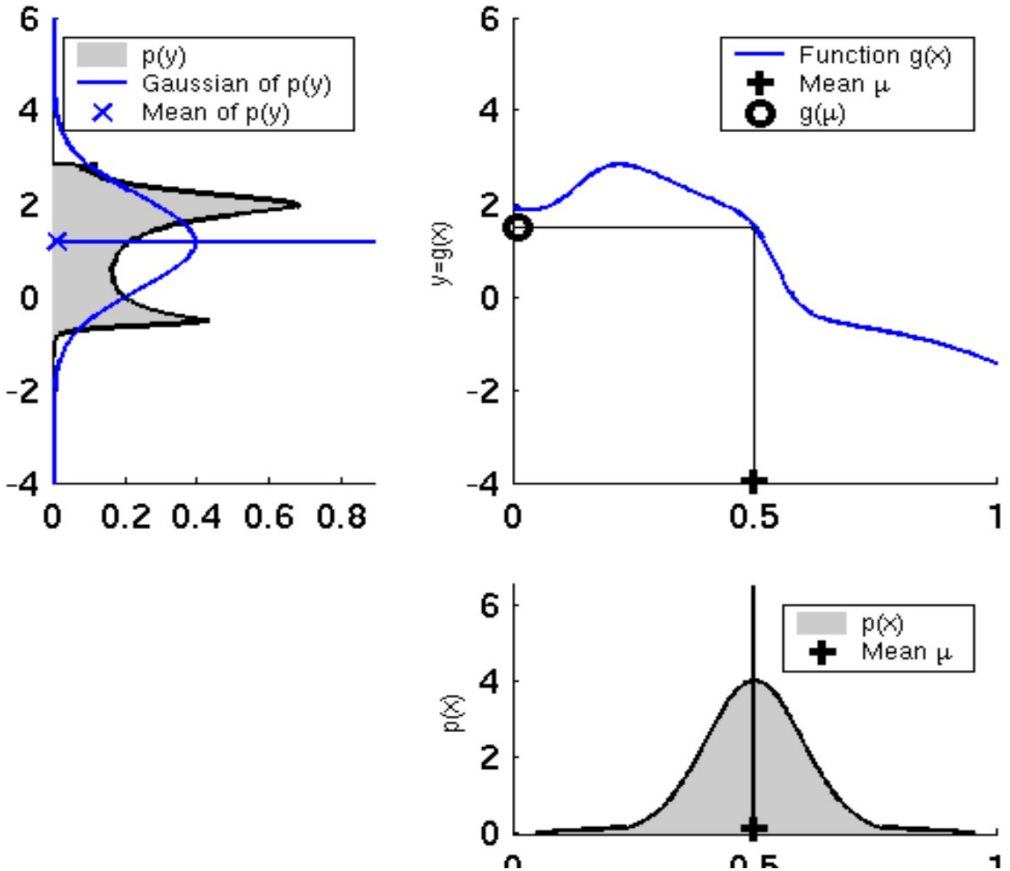
Non linear relationship between state x_t and z_t !

~~$z_t = C_t x_t + \delta_t$~~

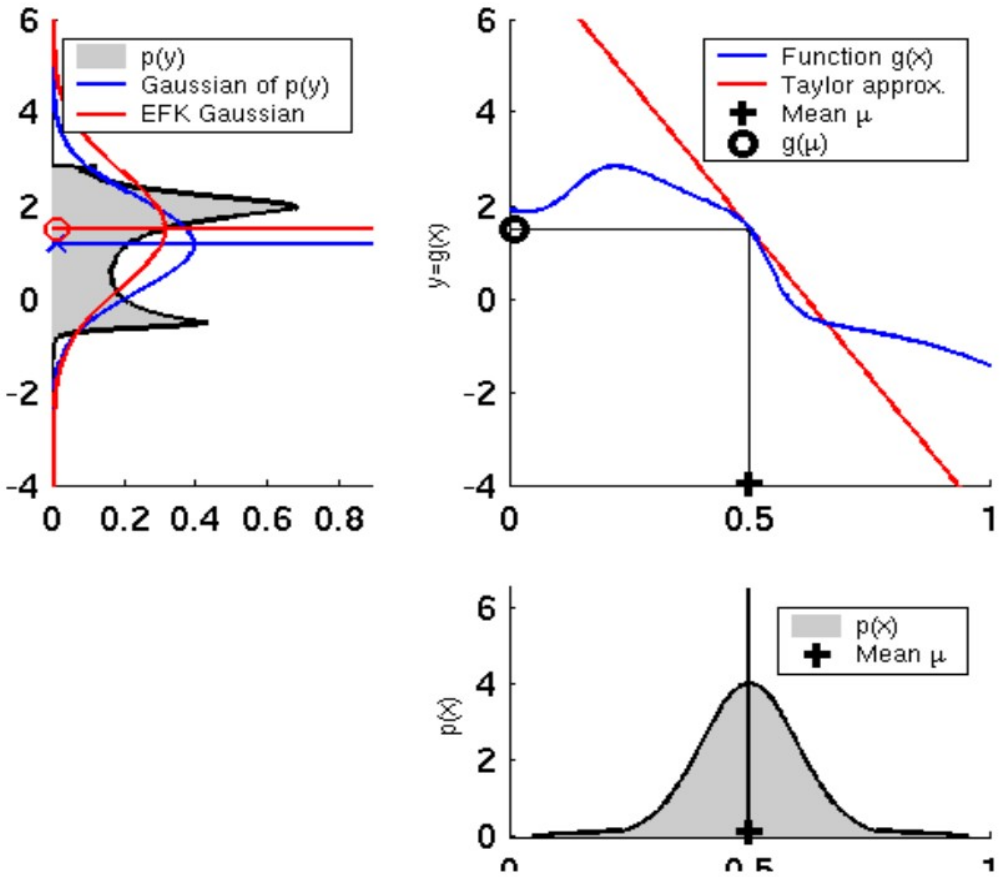
With Linear Models



Without Linear Models



Without Linear Models: Linearize!



Jacobian Matrix

- Given $x \in \mathbb{R}_n$ *and a* general multi-dimensional functions $f(x)$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ (i.e., $f(x) : \{f_1(x), .., f_m(x)\}$). **How to linearize *f(x)*?**
- Use the Jacobian matrix! ➡ $\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$
  - The idea is to approximate $f(x)$ with $J \cdot x$ $f(x) \approx J \cdot x$

Extended Kalman Filter (EKF)

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

5. Correction:

6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$ ⟵ $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$

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

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

Landmark-based EKF Localization

1. EKF_localization ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$):

Prediction:

2. $G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} = \begin{pmatrix} \frac{\partial x'}{\partial \mu_{t-1,x}} & \frac{\partial x'}{\partial \mu_{t-1,y}} & \frac{\partial x'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial y'}{\partial \mu_{t-1,x}} & \frac{\partial y'}{\partial \mu_{t-1,y}} & \frac{\partial y'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial \theta'}{\partial \mu_{t-1,x}} & \frac{\partial \theta'}{\partial \mu_{t-1,y}} & \frac{\partial \theta'}{\partial \mu_{t-1,\theta}} \end{pmatrix}$ Jacobian of *g* w.r.t location

3. $V_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial u_t} = \begin{pmatrix} \frac{\partial x'}{\partial v_t} & \frac{\partial x'}{\partial \omega_t} \\ \frac{\partial y'}{\partial v_t} & \frac{\partial y'}{\partial \omega_t} \\ \frac{\partial \theta'}{\partial v_t} & \frac{\partial \theta'}{\partial \omega_t} \end{pmatrix}$ Jacobian of *g* w.r.t control

4. $M_t = \begin{pmatrix} (\alpha_1 |v_t| + \alpha_2 |\omega_t|)^2 & 0 \\ 0 & (\alpha_3 |v_t| + \alpha_4 |\omega_t|)^2 \end{pmatrix}$ Motion noise

5. $\bar{\mu}_t = g(u_t, \mu_{t-1})$ Predicted mean

6. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$ Predicted covariance

1. **EKF_localization** ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$):

Correction:

2. $\hat{z}_t = \begin{pmatrix} \sqrt{(m_x - \bar{\mu}_{t,x})^2 + (m_y - \bar{\mu}_{t,y})^2} \\ \operatorname{atan2}(m_y - \bar{\mu}_{t,y}, m_x - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \end{pmatrix}$ Predicted measurement mean

3. $H_t = \frac{\partial h(\bar{\mu}_t, m)}{\partial x_t} = \begin{pmatrix} \frac{\partial r_t}{\partial \bar{\mu}_{t,x}} & \frac{\partial r_t}{\partial \bar{\mu}_{t,y}} & \frac{\partial r_t}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \varphi_t}{\partial \bar{\mu}_{t,x}} & \frac{\partial \varphi_t}{\partial \bar{\mu}_{t,y}} & \frac{\partial \varphi_t}{\partial \bar{\mu}_{t,\theta}} \end{pmatrix}$ Jacobian of *h* w.r.t location

4. $Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{pmatrix}$

5. $S_t = H_t \bar{\Sigma}_t H_t^T + Q_t$ Pred. measurement covariance

6. $K_t = \bar{\Sigma}_t H_t^T S_t^{-1}$ Kalman gain

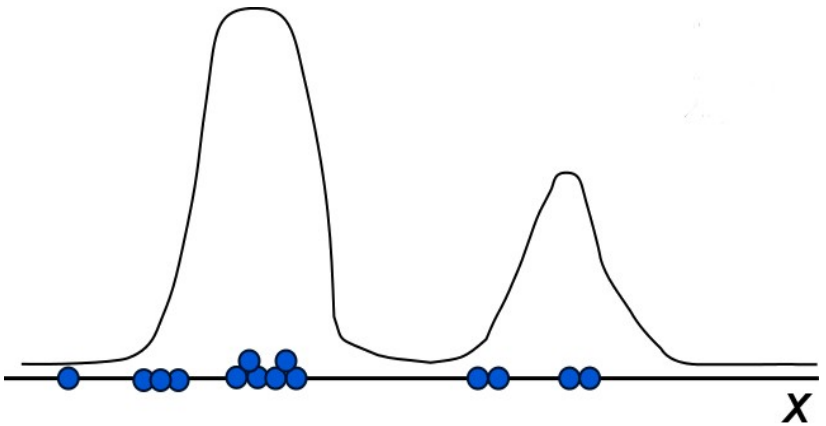
7. $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$ Updated mean

8. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$ Updated covariance

Particle Filters

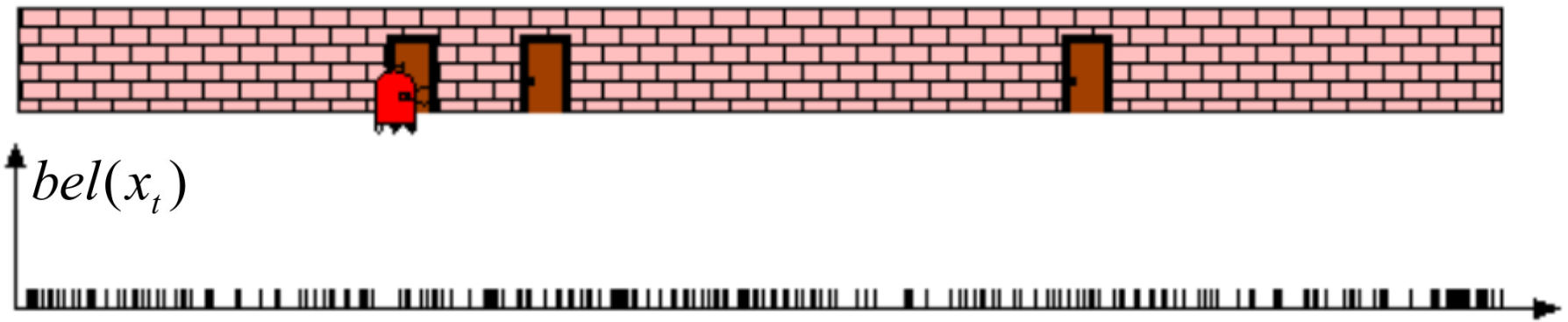
- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

  "**More the samples (*particles*) are dense in a neighborhood, more the probabilty is high in such neighborhood**"



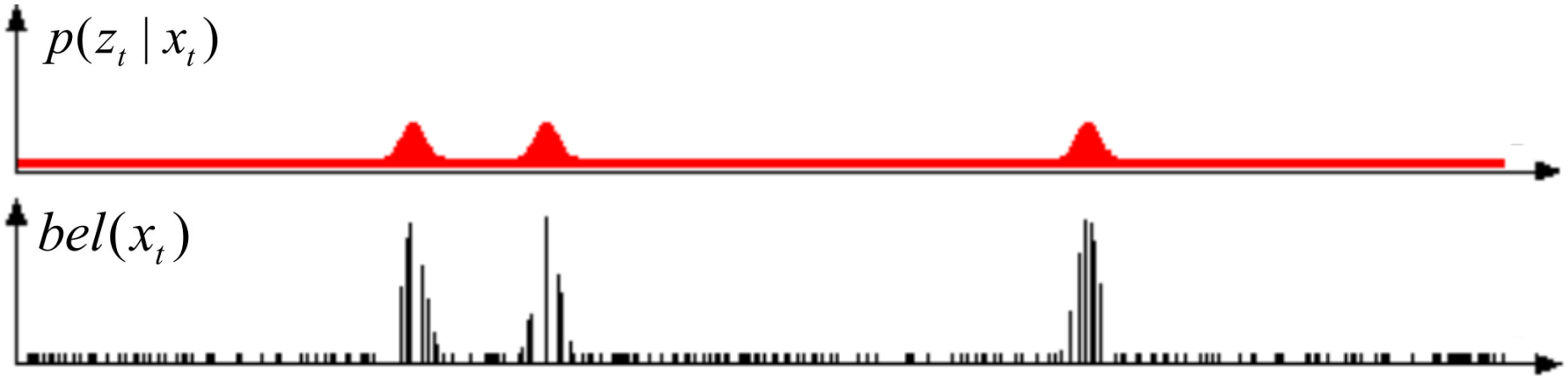
Particle Filters

Start drawing particles from an uniform distribution over the whole space domain



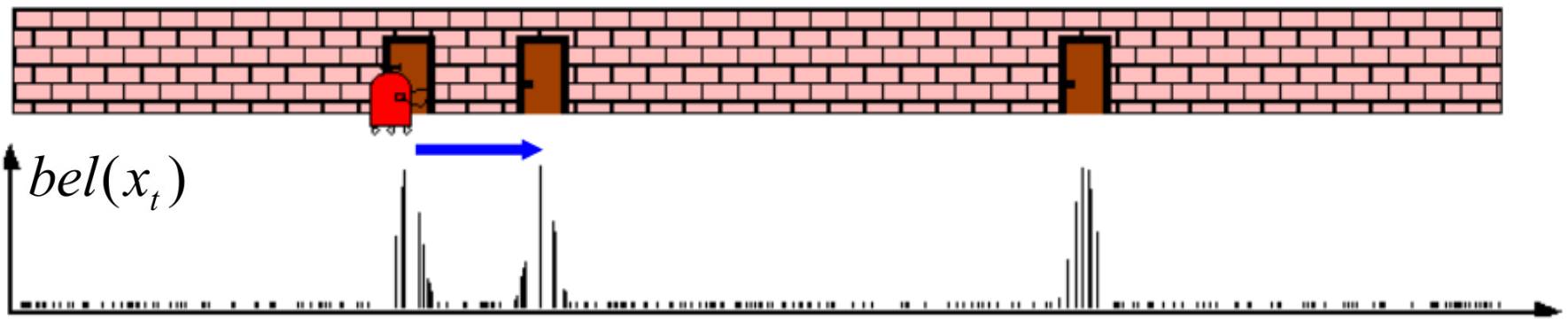
Particle Filters

For each particle, compute the **importance weight** directly from the sensor model $p(z_t \mid x_t)$



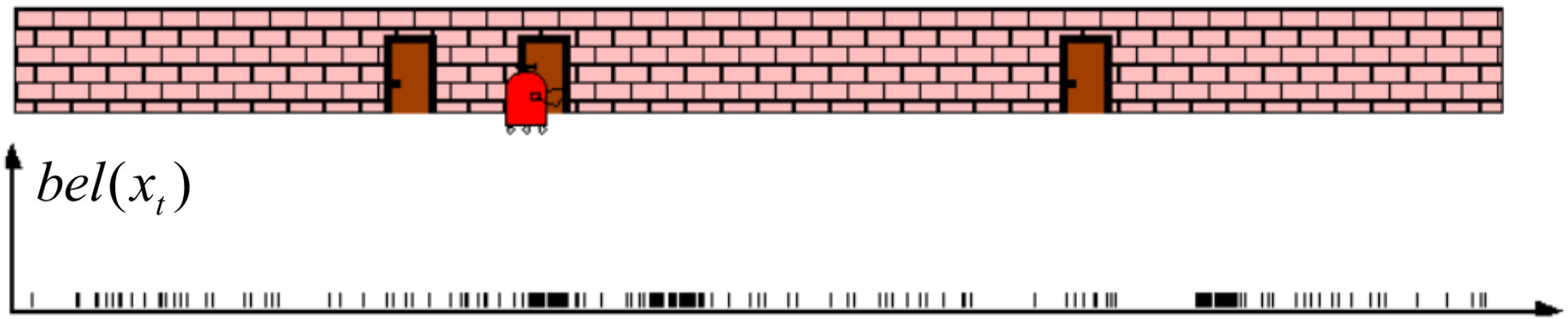
Particle Filters

For each particle x(i), sample a new particle x(i)' by using the motion model $p(x_t \mid u_t, x_{t-1})$



Particle Filters

Resample a new generation of particles in accordance with their importance weight: **particles with high weight (i.e., likelihood) will be easily duplicated, vice versa will be easily deleted**.



Particle Filter Algorithm

$$Bel(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

draw x^i_{t-1} from $Bel(x_{t-1})$

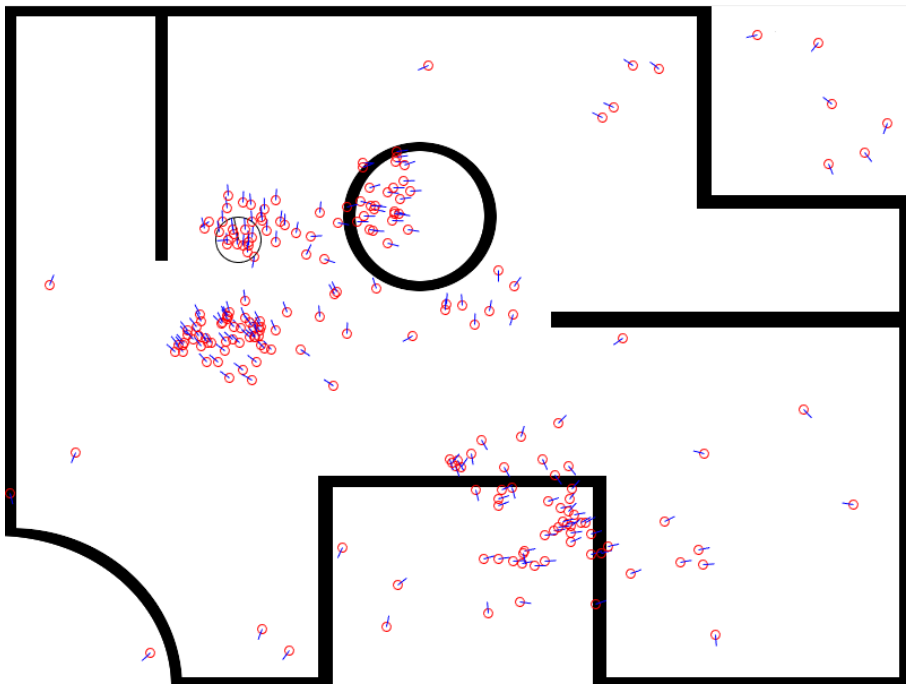
draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

Importance factor for x^i_t:

Localization with Particle Filters

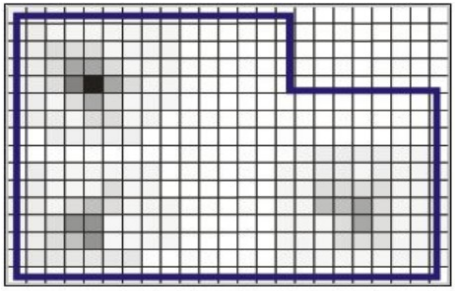
Particle are **pose hypoteses**

1) Update each pose hypoteses with the more recent motion
2) "Weight" each pose hypoteses with the more recent sensor reading.
3) **Resample** in accordance with their importance weight.
4) Repeat from 1)



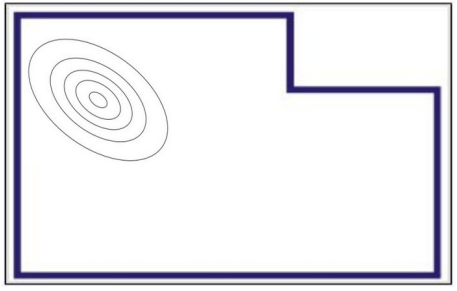
Localization with Particle Filters

Bayes Filter solutions: recap

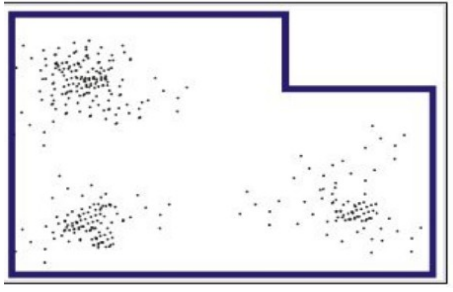


Grid based approaces

- Also called **Hidden Markow Models**



Parametric in countinous space



Particle Filters