
Localization in Mobile Robotics - Part II



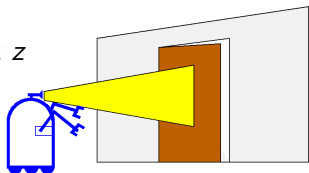
Miroslav Kulich

Czech Technical University in Prague
Czech Institute of Informatics, Robotics and Cybernetics
Intelligent and Mobile Robotics Group

<http://imr.ciirc.cvut.cz/people/Mirek>

Simple example of state estimation

- Suppose a robot obtains measurement z
- What is $p(open|z)$?
- $p(open|z)$ is **diagnostic**
- $p(z|open)$ is **causal**
- Often **causal** knowledge is easier to obtain (counting frequencies)
- Bayes rule allows us to use causal:



$$p(open|z) = \frac{p(z|open)p(open)}{p(z)}$$

Example - open doors

- $p(z|open) = 0.6$ $p(z|\neg open) = 0.3$
- $p(open) = p(\neg) = 0.5$

$$p(open|z) = \frac{p(z|open)p(open)}{p(z|open)p(open) + p(z|\neg open)p(\neg open)}$$

$$p(open|z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

- z raises probability that the door is open.

Example - second measurement

- $p(z_2|open) = 0.5$ $p(z_2|\neg open) = 0.6$
- $p(open|z_1) = \frac{2}{3}$

$$\begin{aligned} p(open|z_2z_1) &= \frac{p(z_2|open)p(open|z_1)}{p(z_2|open)p(open|z_1) + p(z_2|\neg open)p(\neg open|z_1)} \\ &= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625 \end{aligned}$$

- z_2 lowers the probability that the door is open.

Actions

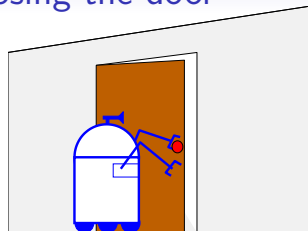
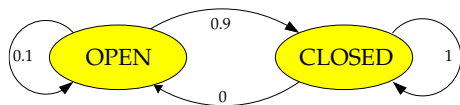
- Often the world is dynamic since
 - actions carried out by the robot,
 - actions carried out by other agents,
 - or just the time passing by change the world (plants grow).
- Actions are never carried out with absolute certainty.
- In contrast to measurements, actions generally decrease the uncertainty.
- To incorporate the outcome of an action u into the current “belief”, we use the conditional pdf

$$p(x|u, x')$$

- This term specifies the pdf that executing u changes the state from x' to x .

Continuing the example - closing the door

$p(x|u, x')$ for $u =$ "close door"



$$p(x|u) = \sum_{x'} p(x|u, x')p(x')$$

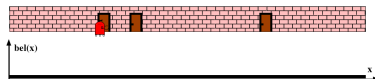
If the door is open, the action "close door" succeeds in 90% of all cases.

Continuing the example - closing the door

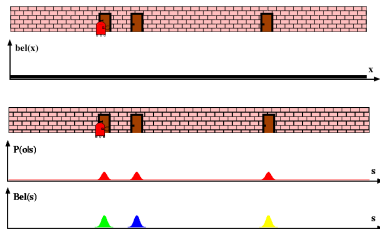
$$\begin{aligned}p(\text{closed}|u) &= \sum_{x'} p(\text{closed}|u, x')p(x') \\&= p(\text{closed}|u, \text{open})p(\text{open}) \\&+ p(\text{closed}|u, \text{closed})p(\text{closed}) \\&= \frac{9}{10} \cdot \frac{5}{8} + \frac{1}{1} \cdot \frac{3}{8} = \frac{15}{16}\end{aligned}$$

$$\begin{aligned}p(\text{open}|u) &= \sum_{x'} p(\text{open}|u, x')p(x') \\&= p(\text{open}|u, \text{open})p(\text{open}) \\&+ p(\text{open}|u, \text{closed})p(\text{closed}) \\&= \frac{1}{10} \cdot \frac{5}{8} + \frac{0}{1} \cdot \frac{3}{8} = \frac{1}{16} \\&= 1 - p(\text{closed}|u)\end{aligned}$$

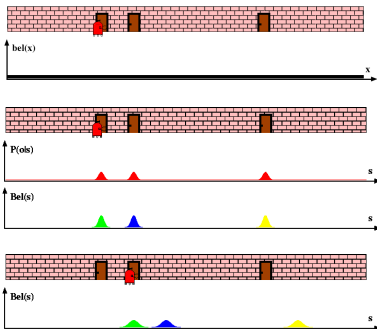
Motivation



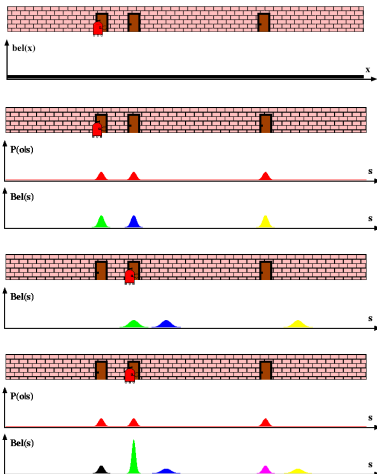
Motivation



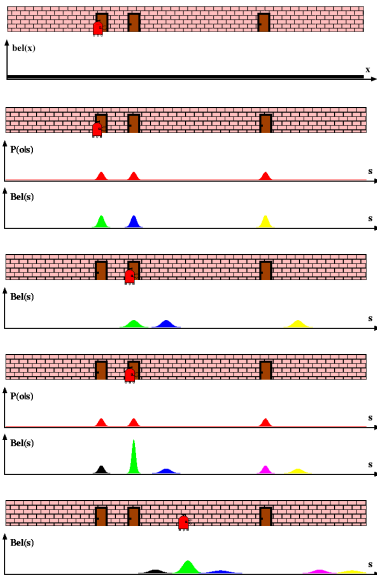
Motivation



Motivation



Motivation



Bayes filter: the framework

- Given:

- Stream of observations z and actions u :

$$d_t = \{u_1, z_1, \dots, u_t, z_t\}$$

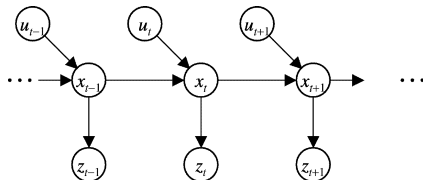
- **Sensor model** $p(z|x)$
- **Action model** $p(x|u, x')$
- **Prior** probability of the system state $p(x)$

- Wanted:

- Estimate of the state X of a **dynamic system**
- The posterior of the state is also called **belief**:

$$Bel(x_t) = p(x_t | u_1, z_1, \dots, u_t, z_t)$$

Markov assumption



$$p(z_t | x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t | x_t)$$
$$p(x_t | x_{1:t-1}, z_{1:t}, u_{1:t}) = p(x_t | x_{t-1}, u_t)$$

Underlying assumptions

- Static world
- Independent noise
- Perfect model, no approximation errors

Bayes filter - derivation

 $Bel(x_t)$

$$= p(x_t | u_1, z_1, \dots, u_t, z_t)$$

Bayes

$$= \eta p(z_t | x_t, u_1, z_1, \dots, u_t) p(x_t | u_1, z_1, \dots, u_t)$$

Markov

$$= \eta p(z_t | x_t) p(x_t | u_1, z_1, \dots, u_t)$$

Total prob.

$$= \eta p(z_t | x_t) \int p(x_t | u_1, z_1, \dots, u_t, x_{t-1}) p(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$$

Markov

$$= \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) p(x_{t-1} | u_1, z_1, \dots, u_t) dx_{t-1}$$

Markov

$$= \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) p(x_{t-1} | u_1, z_1, \dots, z_{t-1}) dx_{t-1}$$

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

Bayes filter

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

Algorithm Bayes_filter($Bel(x), d$)

if d is a **measurement** z **then**

$$\eta = 0$$

for all x **do**

$$Bel'(x) = p(z|x) Bel(x)$$

$$\eta = \eta + Bel'(x)$$

end for

for all x **do**

$$Bel'(x) = \eta^{-1} Bel'(x)$$

end for

end if

if d is a **action** u **then**

for all x **do**

$$Bel'(x) =$$

$$\int p(x|u, x') Bel(x') dx'$$

end for

end if

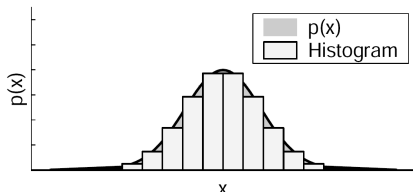
return $Bel'(x)$

Bayes filters are familiar

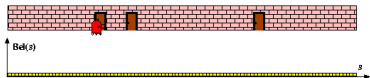
- Kalman filters
- Histogram filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

Non-parametric filters

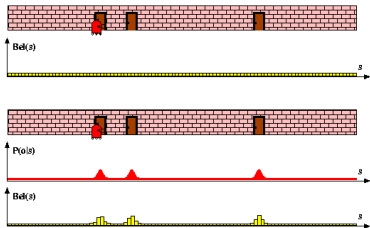
- Don't rely on a fixed functional form of the posterior.
- Approximation of probability density by a finite number of values.
- Adaptive (based on discretization), they handle nonlinearities.
- The number of samples biases the speed of the algorithm and the quality of the filter.
- Histogram \times particle filter



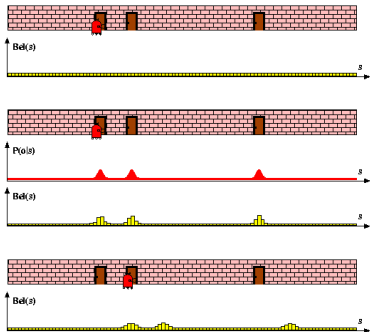
Histogram filter



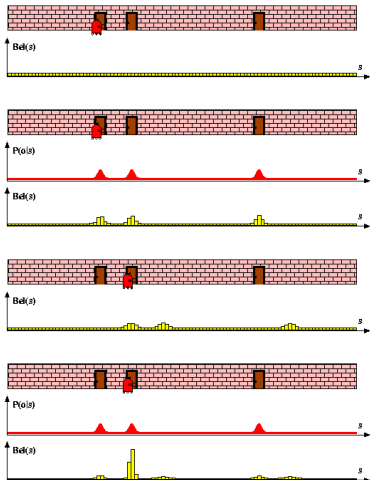
Histogram filter



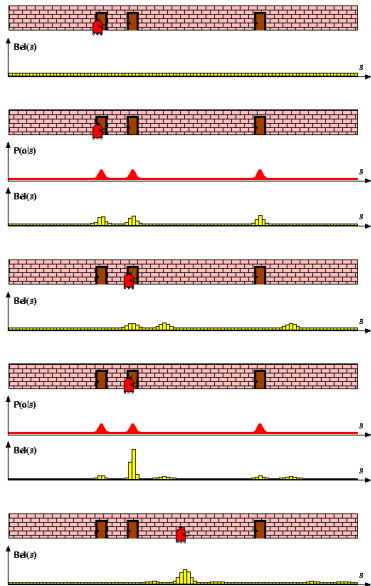
Histogram filter



Histogram filter



Histogram filter



Histogram filter

if d is a measurement z then

$$\eta = 0$$

for all x do

$$Bel'(x) = p(z|x)Bel(x)$$

$$\eta = \eta + Bel'(x)$$

end for

for all x do

$$Bel'(x) = \eta^{-1} Bel'(x)$$

end for

else if d is an action u then

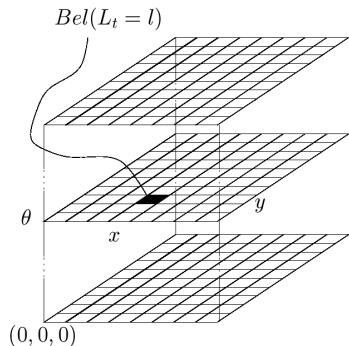
for all x do

$$Bel'(x) = \int p(x|u, x')Bel(x')dx'$$

end for

end if

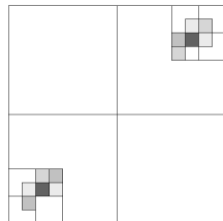
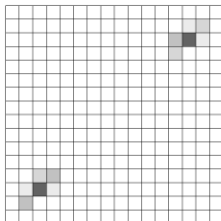
return $Bel'(x)$



$$Bel(x_t = \langle x, y, \phi \rangle)$$

Histogram filter

- To update the belief upon sensory input and to carry out the normalization one has to iterate over all cells of the grid => complexity $O(n^2)$
- Selective update
 - Only a part of state space is updated ...
 - ... but the quality of the localization should be monitored
- Dynamic state space decomposition – kd-trees (density trees):
division grain depends on probability density (higher pdf => finer grain)

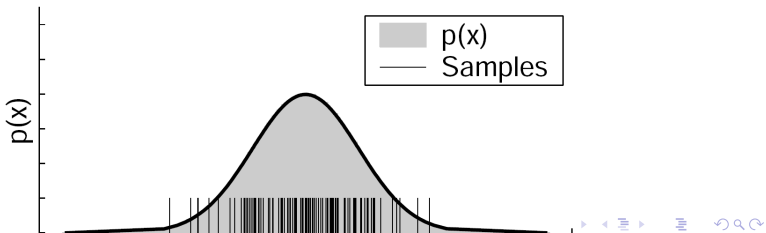


Particle filter

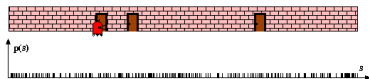
- Probability density represented by “appropriately” (randomly) placed particles:

$$Bel(x_t) \approx \{x^{(i)}, w^{(i)}\}_{i=1, \dots, m}$$

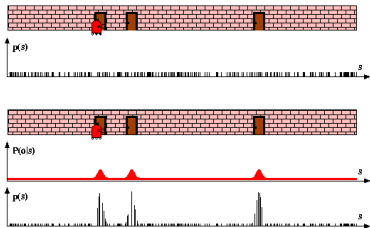
- Particles are weighted.
- Particles for time t are chosen according to the weights in time $t - 1$.
- Really simple to implement.
- Most universal Bayes filter: representation of **non-Gaussian** distributions and **non-linear** processes



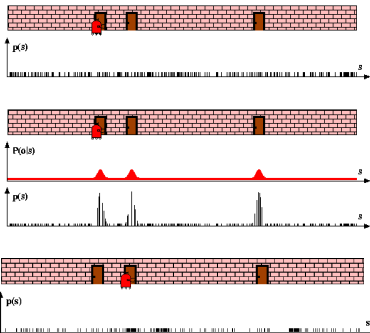
Particle filter



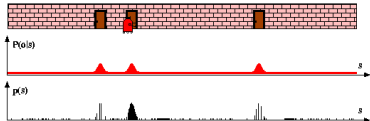
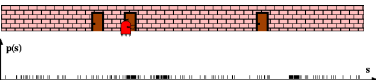
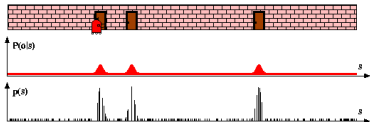
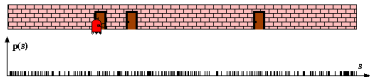
Particle filter



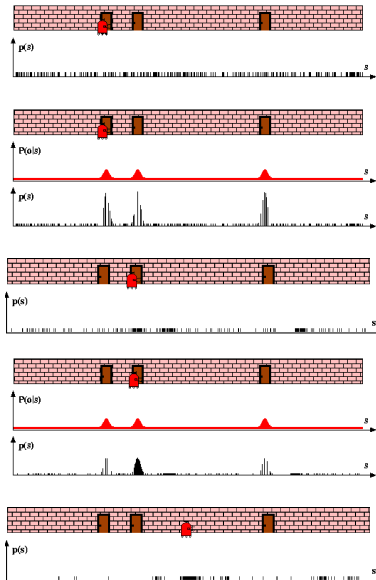
Particle filter



Particle filter



Particle filter



Particle filter - the algorithm

Particle_filter(S_{t-1}, u_{t-1}, z_t)

$S_t = \emptyset, \eta = 0$

for $i = 1, \dots, n$ **do**

Generate new particles

Sample index j_i from the discrete distribution given by w_{t-1}

Sample $x_t^i \sim p(x_t | x_{t-1}, u_{t-1})$ using $x_{t-1}^{j_i}$ and u_{t-1}

$w_t^i = p(z_t | x_t^i)$

Compute importance weight

$\eta = \eta + w_t^i$

Update normalization factor

$S_t = S_t \cup \{ \langle x_t^i, w_t^i \rangle \}$

Insert a particle

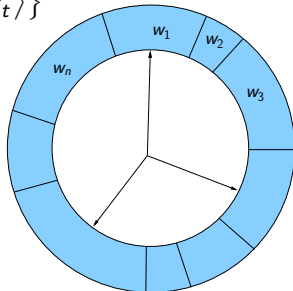
end for

for $i = 1, \dots, n$ **do**

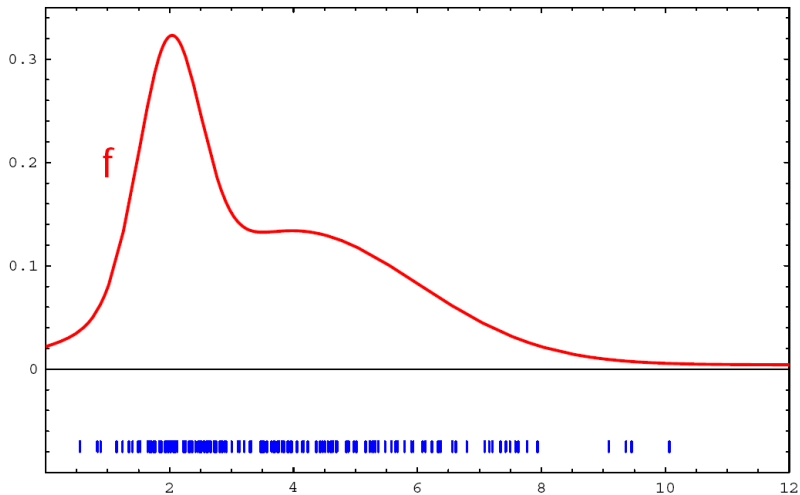
$w_t^i = \frac{w_t^i}{\eta}$

Normalize weights

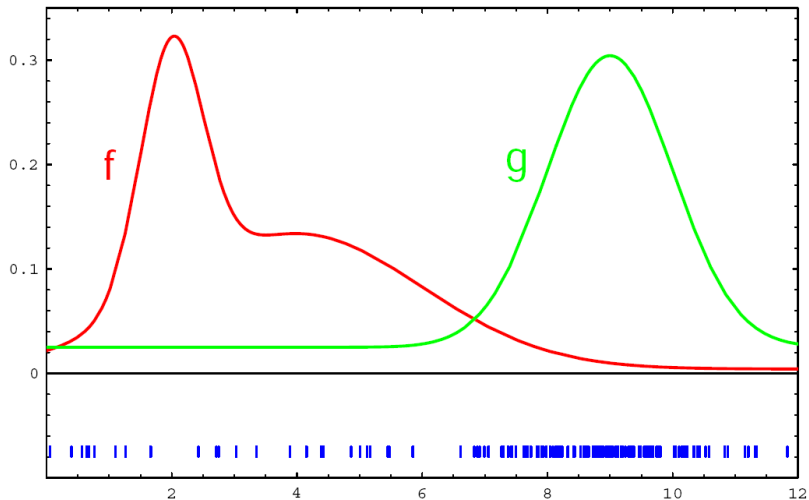
end for



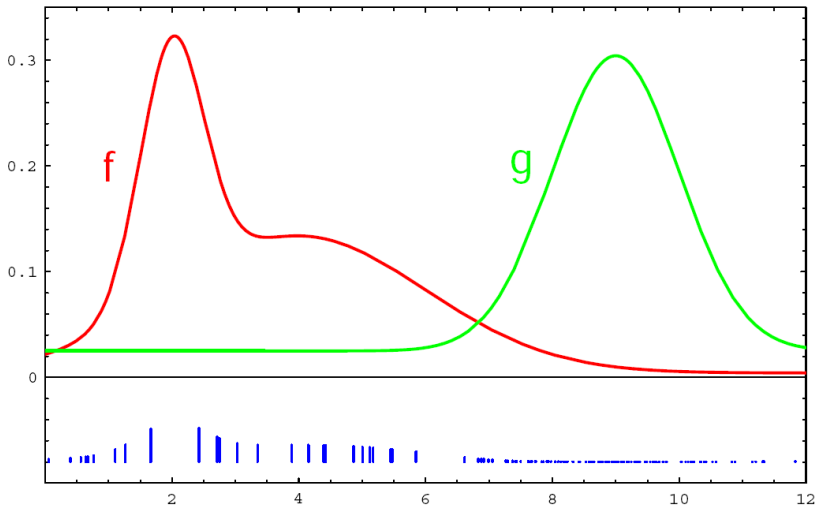
Importance sampling



Importance sampling



Importance sampling



Kalman filter

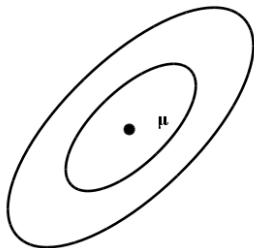
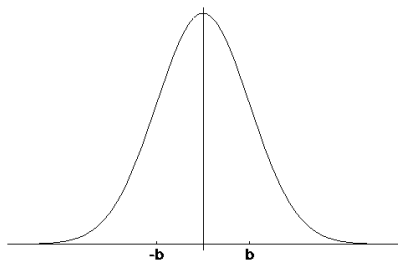
- Unimodal representation of probability density by a Gaussian

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

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

$$p(x) \sim N(\mu, \Sigma)$$

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



Linear transformation

- Linear transformation preserves normal distribution.

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

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

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

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

Fundamental assumptions

- Markov assumption + the three following:
 - Probability of state transition (prediction/motion model) $p(x|u, x')$ is linear with added Gaussian noise:

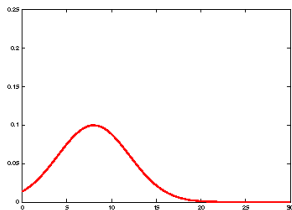
$$x_t = Ax_{t-1} + B_t u_t + \varepsilon_t$$

- Sensor model (correction) is linear with added Gaussian noise

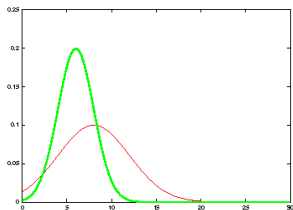
$$z_t = Cx_t + \delta_t$$

- A-priory information about the state (belief) must have normal distribution.

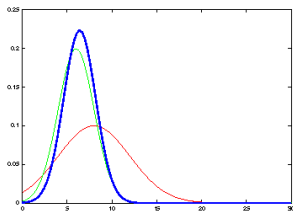
Kalman filter



A-priory belief

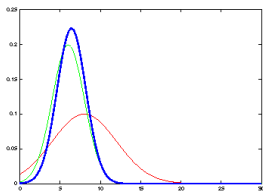


New measurement

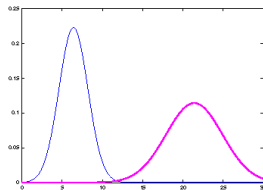


Integration of the new measurement

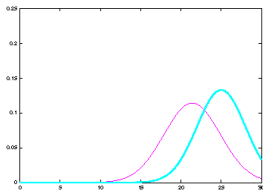
Kalman filter



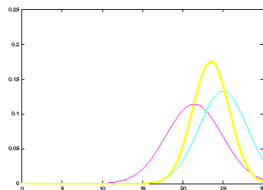
Actual belief



Action



New measurement



Integration of the new
measurement

Kalman filter - the algorithm

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

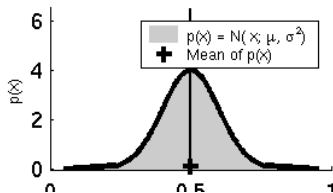
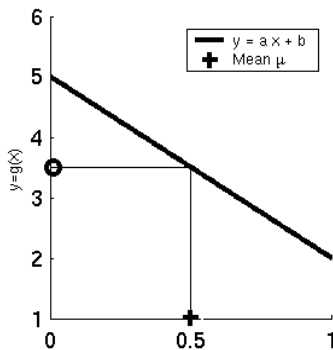
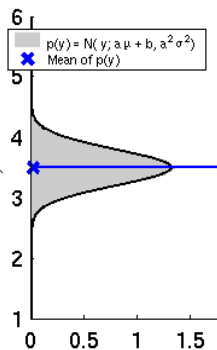
Prediction (action integration)

$$\begin{aligned}\bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t\end{aligned}$$

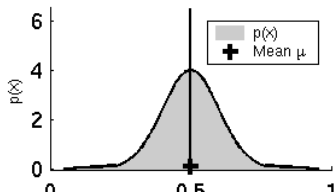
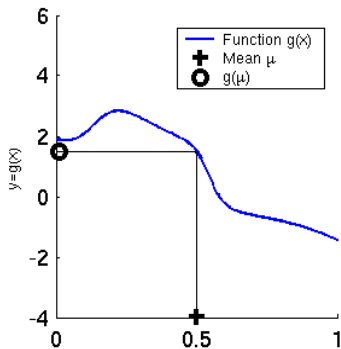
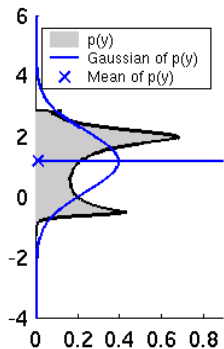
Correction (measurement integration)

$$\begin{aligned}K_t &= \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\ \mu_t &= \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \Sigma_t &= (E - K_t C_t) \bar{\Sigma}_t \\ \text{return } &\mu_t, \Sigma_t\end{aligned}$$

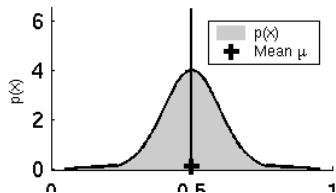
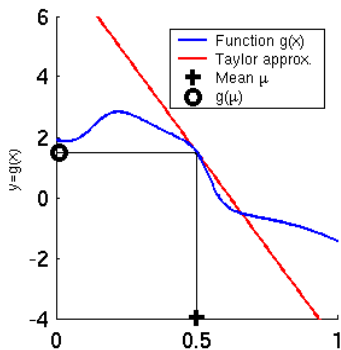
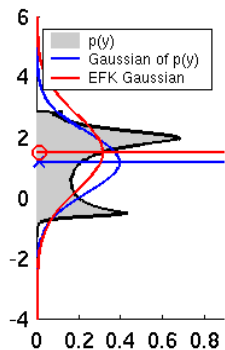
Kalman filter - linear transformation



Non-linear transformation



Extended Kalman filter



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

$$z_t = h(x_t)$$

Extended Kalman filter

- Taylor expansion v μ used for linearisation, i.e. matrix of functions derivations – Jacobians
- 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:

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

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

Extended Kalman filter - the algorithm

Algorithm `Extended_Kalman_filter`($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)

Prediction (action integration)

$$\begin{aligned}\bar{\mu}_t &= g(u_t, \mu_{t-1}) \\ \bar{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t\end{aligned}$$

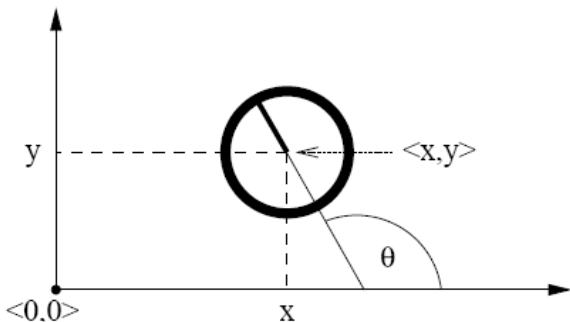
Correction (measurement integration)

$$\begin{aligned}K_t &= \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \\ \mu_t &= \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))\end{aligned}$$

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

Motion model

- Motion model $p(x|x', u)$ is needed for implementation of Bayes filter.
- Motion model defines probability, that the robot will be in the state x after realization of the action u in the state x' . The robot operates in the plane, i.e. $x = \langle x, y, \phi \rangle$
- Different models (depending on control type, whether kinematics is considered, etc.)



Odometry-based motion model

- Used when systems are equipped with wheel encoders.

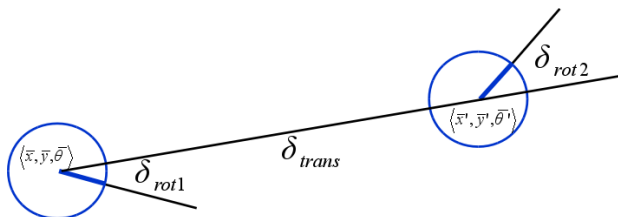
Ideal case

- Robot moves from $\langle \bar{x}, \bar{y}, \bar{\phi} \rangle$ to $\langle \bar{x}', \bar{y}', \bar{\phi}' \rangle$
- Odometry information $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$$

$$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\phi}$$

$$\delta_{rot2} = \bar{\phi}' - \bar{\phi} - \delta_{rot1}$$



Adding noise

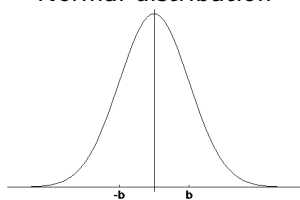
- The measured motion is given by the true motion corrupted with noise.

$$\begin{aligned}\hat{\delta}_{rot1} &= \delta_{rot1} + \varepsilon_{\alpha_1}|\delta_{rot1}| + \alpha_2|\delta_{trans}| \\ \hat{\delta}_{trans} &= \delta_{trans} + \varepsilon_{\alpha_3}|\delta_{trans}| + \alpha_4(|\delta_{rot1}| + |\delta_{rot2}|) \\ \hat{\delta}_{rot2} &= \delta_{rot2} + \varepsilon_{\alpha_1}|\delta_{rot2}| + \alpha_2|\delta_{trans}|\end{aligned}$$

- Noise is determined by four parameters.
- Most difficult thing is to get the noise parameters – experimentally.

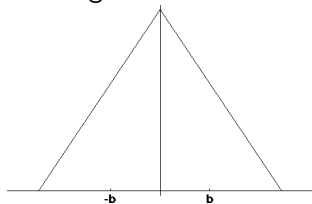
Typical distributions for probabilistic motion models

Normal distribution



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

Triangular distribution



$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0 & \text{if } |x| > \sqrt{6\sigma^2} \\ \frac{\sqrt{6\sigma^2} - |x|}{6\sigma^2} & \text{otherwise} \end{cases}$$

Odometry-based model for sampling

$$u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \phi \rangle \Rightarrow x' = \langle x', y', \phi' \rangle$$

Random control

$$\begin{aligned}\hat{\delta}_{rot1} &= \delta_{rot1} + \text{sample}(\alpha_1 |\delta_{rot1}| + \alpha_2 |\delta_{trans}|) \\ \hat{\delta}_{trans} &= \delta_{trans} + \text{sample}(\alpha_3 |\delta_{trans}| + \alpha_4 (|\delta_{rot1}| + |\delta_{rot2}|)) \\ \hat{\delta}_{rot2} &= \delta_{rot2} + \text{sample}(\alpha_1 |\delta_{rot2}| + \alpha_2 |\delta_{trans}|)\end{aligned}$$

Position determination

$$\begin{aligned}x' &= x + \hat{\delta}_{trans} + \cos(\phi + \hat{\delta}_{rot1}) \\ y' &= x + \hat{\delta}_{trans} + \sin(\phi + \hat{\delta}_{rot1}) \\ \phi' &= \phi + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \\ &\text{return } \langle x', y', \phi' \rangle\end{aligned}$$

sample (normal distribution): $\frac{1}{2} \sum_{i=1}^{12} \text{rand}(-b, b)$

Calculating $p(x|u, x')$

Odometry values (u)

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

Ideal case

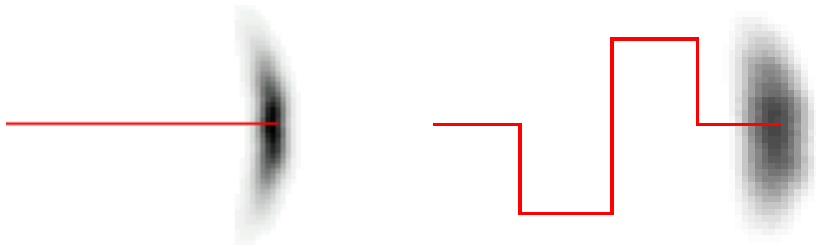
$$\begin{aligned}\hat{\delta}_{trans} &= \sqrt{(x' - x)^2 + (y' - y)^2} \\ \hat{\delta}_{rot1} &= \text{atan2}(y' - y, x' - x) - \phi \\ \hat{\delta}_{rot2} &= \phi' - \phi - \delta_{rot1}\end{aligned}$$

Probability calculation

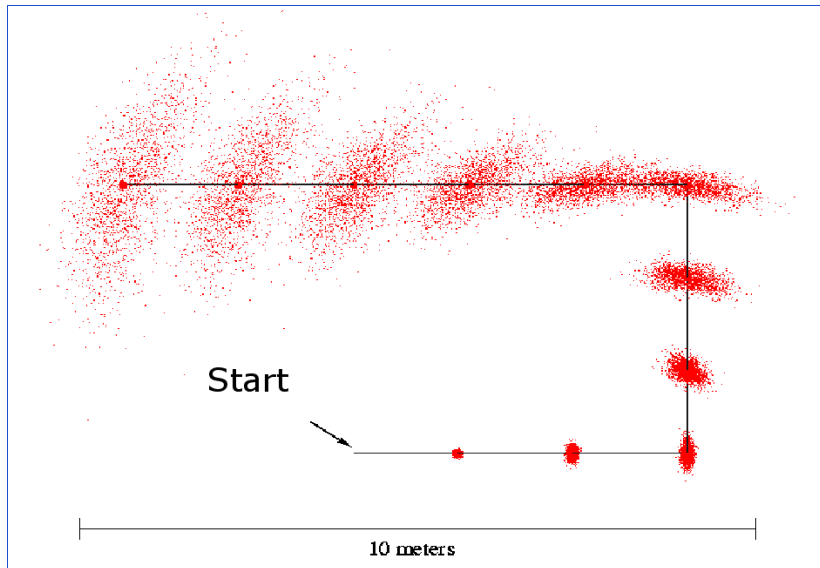
$$\begin{aligned}p_1 &= \text{prob}(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1 |\delta_{rot1}| + \alpha_2 \delta_{trans}) \\ p_2 &= \text{prob}(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3 \delta_{trans} + \alpha_4 (|\delta_{rot1}| + |\delta_{rot2}|)) \\ p_3 &= \text{prob}(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1 |\delta_{rot2}| + \alpha_2 \delta_{trans}) \\ &\text{return } p_1 p_2 p_3 \quad \leftarrow \text{independence assumption}\end{aligned}$$

Application

- Resulting probability density depends on trajectory traversed, not only on the final robot position!
- For complex cases, repeat the above algorithm accordingly.
- A typical example of the distribution (2D projection)



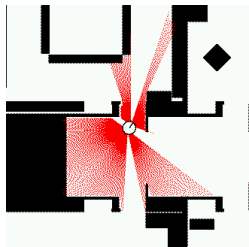
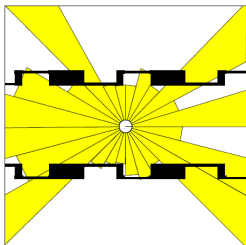
Trajectory composition



Sensor model

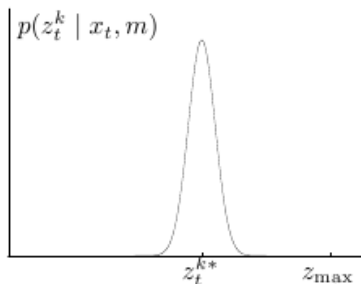
- The aim is to determine $p(z|m, x)$.
- We will use proximity sensor (laser, sonar).
- Scan is composed from k measurements (beams):

$$z = \{z_1, z_2, \dots, z_k\}$$
- Individual measurements are independent given the robot position (strong assumption): $P(z|x, m) = \prod_{k=1}^k P(z_k|x, m)$



Beam-based model - components

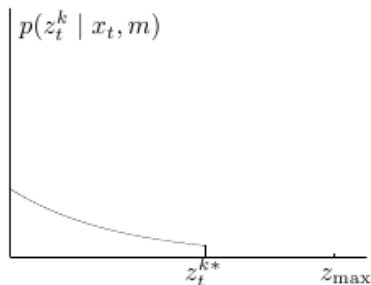
Measurement noise



$$p_{hit}(z|x, m) = \begin{cases} \eta N(z, z^*, \sigma_{hit}^2) & \text{if } 0 \leq z \leq z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Normal distribution

Unexpected obstacles

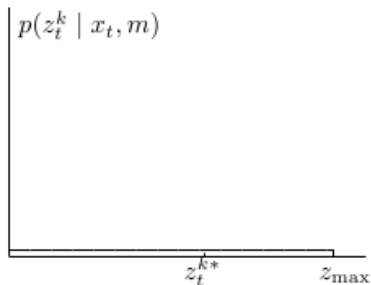


$$p_{short}(z|x, m) = \begin{cases} \eta \lambda_{short} e^{-\lambda_{short} z} & \text{if } 0 \leq z \leq z^* \\ 0 & \text{otherwise} \end{cases}$$

Poisson distribution

Beam-based model - components

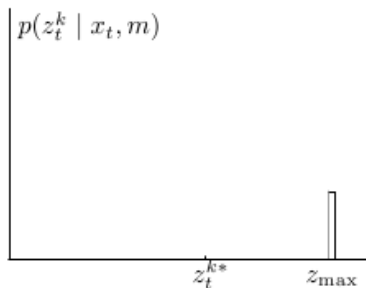
Random measurement



$$p_{\text{rand}}(z|x, m) = \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z \leq z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Uniform distribution

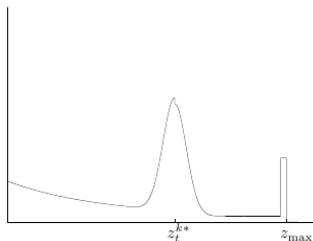
Max range



$$p_{\text{max}}(z|x, m) = \begin{cases} 1 & \text{if } z = z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Discrete distribution :-)

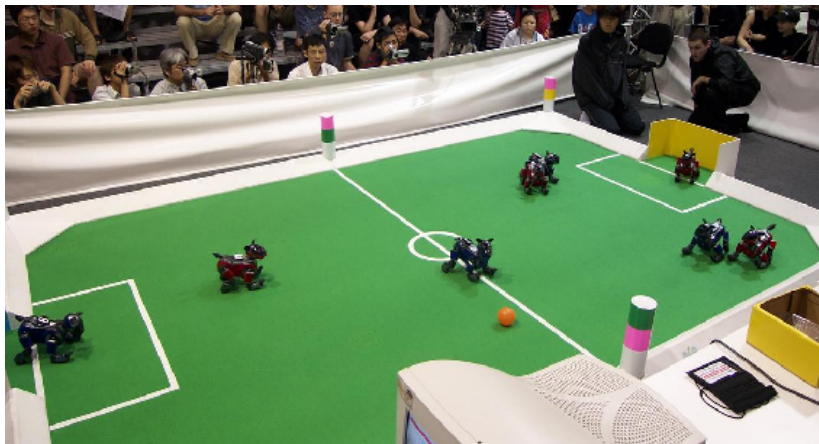
Resulting mixture density



$$p(z|x, m) = \begin{pmatrix} \alpha_{hit} \\ \alpha_{short} \\ \alpha_{rand} \\ \alpha_{max} \end{pmatrix}^T \begin{pmatrix} p_{hit}(z|x, m) \\ p_{short}(z|x, m) \\ p_{rand}(z|x, m) \\ p_{max}(z|x, m) \end{pmatrix}$$

- Model parameters are learned based on real data (E-M, GA)
- Expected distances z_{exp} are determined by raytracing (time consuming).
- Only selected beams are considered (e.g. eight); it increases independence also.
- Expected distances can be pre-processed (for each $\langle x, y, z, \phi \rangle$)
- Multiplication of sensor model by $\lambda < 1$ reduces sensor impact.
- Model is noncontinuous \Rightarrow approximate determination of probability density can miss the right state.

Motivation



EKF-based localization

- Velocity motion model

$$\underbrace{\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}}_{x_t} = \underbrace{\begin{pmatrix} x^* + r\sin(\phi + \omega\Delta t) \\ y^* + r\cos(\phi + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix}}_{g(u_t, x_{t-1})} + N(0, M_t)$$

- Map (list of landmarks $\langle x_i, y_i, \phi_i \rangle$)

$$\underbrace{\begin{pmatrix} r_i \\ \phi_i \end{pmatrix}}_{z_t^i} = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{pmatrix}}_{h(x_t, j, m)} + N(0, Q_t)$$

Prediction

Jacobian of g w.r.t location

$$G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} = \begin{pmatrix} \frac{\partial x'}{\partial \mu_{t-1,x}} & \frac{\partial x'}{\partial \mu_{t-1,y}} & \frac{\partial x'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial y'}{\partial \mu_{t-1,x}} & \frac{\partial y'}{\partial \mu_{t-1,y}} & \frac{\partial y'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial \theta'}{\partial \mu_{t-1,x}} & \frac{\partial \theta'}{\partial \mu_{t-1,y}} & \frac{\partial \theta'}{\partial \mu_{t-1,\theta}} \end{pmatrix}$$

Motion noise

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

Jacobian of g w.r.t control

$$V_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial u_t} = \begin{pmatrix} \frac{\partial x'}{\partial v_t} & \frac{\partial x'}{\partial \omega_t} \\ \frac{\partial y'}{\partial v_t} & \frac{\partial y'}{\partial \omega_t} \\ \frac{\partial \theta'}{\partial v_t} & \frac{\partial \theta'}{\partial \omega_t} \end{pmatrix}$$

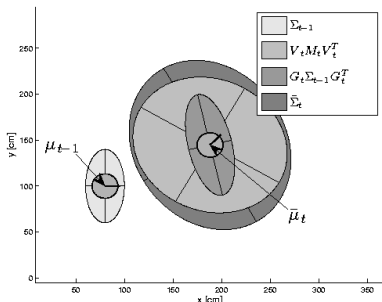
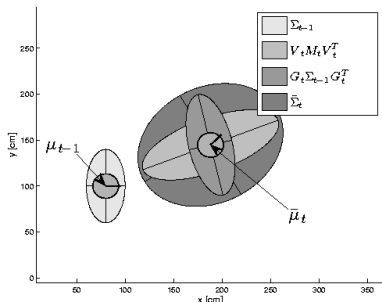
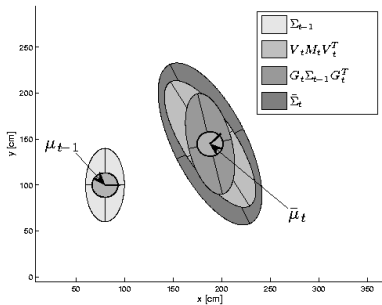
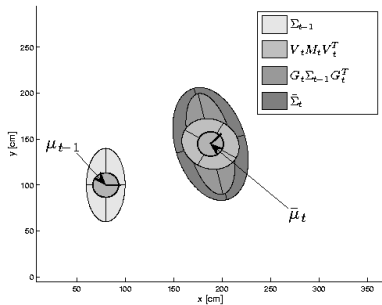
Predicted mean

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

Predicted covariance

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

Prediction



Correction

Predicted measurement mean

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

Jacobian of h w.r.t location

$$H_t = \frac{\partial h(\bar{\mu}_t, m)}{\partial x_t} = \begin{pmatrix} \frac{\partial r_t}{\partial \bar{\mu}_{t,x}} & \frac{\partial r_t}{\partial \bar{\mu}_{t,y}} & \frac{\partial r_t}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \phi_t}{\partial \bar{\mu}_{t,x}} & \frac{\partial \phi_t}{\partial \bar{\mu}_{t,y}} & \frac{\partial \phi_t}{\partial \bar{\mu}_{t,\theta}} \end{pmatrix}$$

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \text{sigma}_\phi^2 \end{pmatrix}$$

Predicted measurement covariance

$$S_t = H_t \bar{\Sigma}_t H_t^T + Q_t$$

Gain

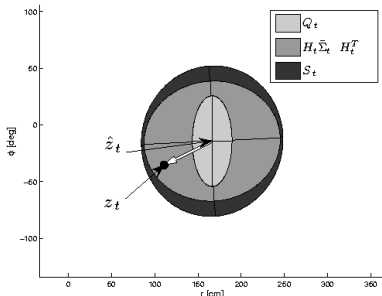
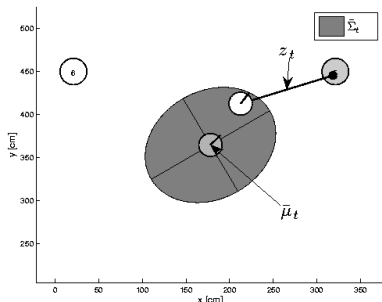
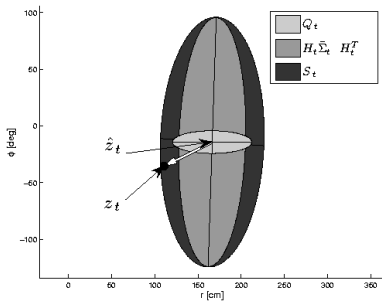
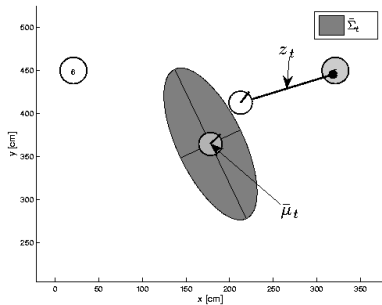
$$K_t = \bar{\Sigma}_t H_t^T S_t^{-1}$$

Updated mean and covariance

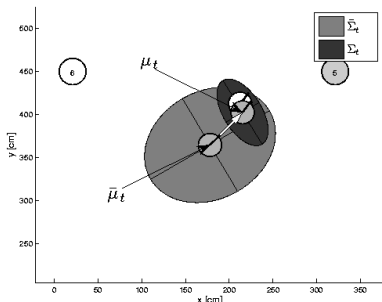
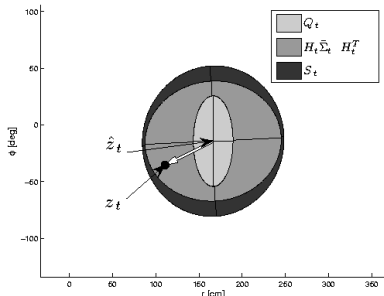
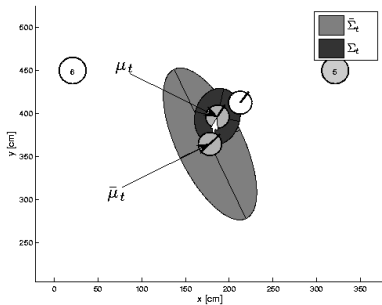
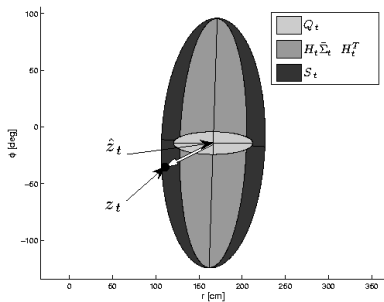
$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

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

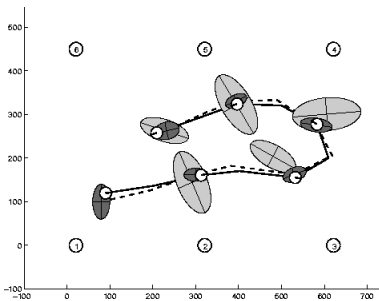
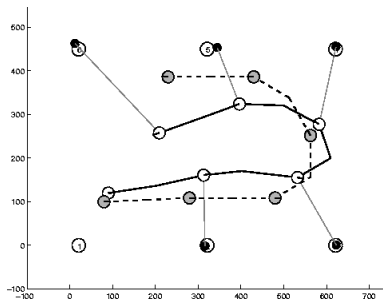
Observation



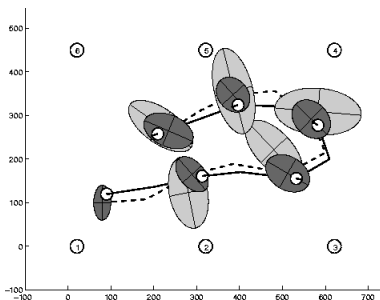
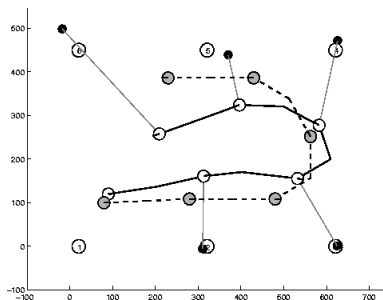
Correction



Estimation sequence 1



Estimation sequence 2



Acknowledgement

I was inspired by lessons of Sebastian Thrun, from which majority of the presented figures comes. These (and many others) can be found and download from

<http://www.probabilistic-robotics.org/>.

I also recommend the book

S. Thrun, W. Burgard, and D. Fox: *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.