Localization in Mobile Robotics

Part II.

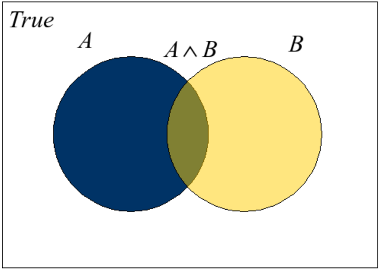
Miroslav Kulich

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

Wednesday 19/10/2016

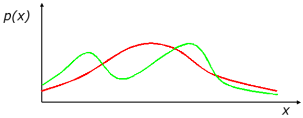
Gentle introduction to probability theory

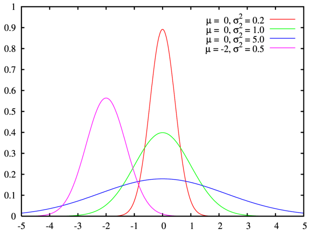
- Idea: explicit representation of uncertainty using calculus of the probability theory
- p(X=x) probability that the random variable X is x
- $0 \leq p(x) \leq 1$
- $p(true) = 1,\ p(false) = 0$
- $p(A \vee B) = p(A) + p(B) - p(A \wedge B)$



Discrete and continuous random variable

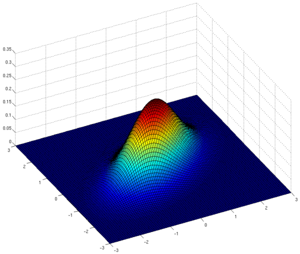
- **Discrete**: X is countable, i.e. $X = x_1, x_2, \ldots, x_n$
- **Continuous**: X can have an uncountable number of values (from some interval)
- p is probability density
- Various distributions
- Most known: Normal (Gaussian)
- $p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

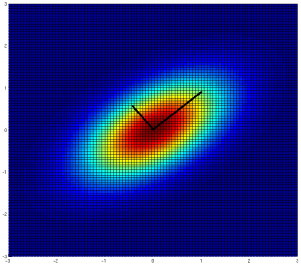




Multi-dimensional normal distribution

$$p(\mathbf{x} = x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})},$$





- Eigenvalues and eigenvectors of the covariance matrix define an ellipse.

Joint and conditional probability distribution

- $p(X = x \text{ a } Y = y) = p(x, y)$
- If X and Y are independent then

$$p(x, y) = p(x)p(y)$$

- $p(x|y)$ is probability x given y

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

$$p(x, y) = p(x|y)p(y)$$

- If X a Y are independent then

$$p(x|y) = p(x)$$

Total probability theorem

Discrete case

$$\sum_x p(x) = 1$$

$$p(x) = \sum_y p(x, y)$$

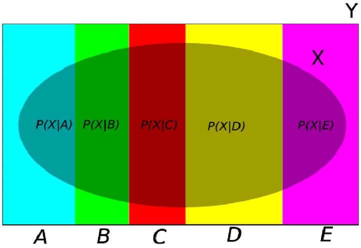
$$p(x) = \sum_y p(x|y)p(y)$$

Continuous space

$$\int_x p(x)dx = 1$$

$$p(x) = \int_y p(x, y)dy$$

$$p(x) = \int_y p(x|y)p(y)dy$$



Bayes' theorem

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$$

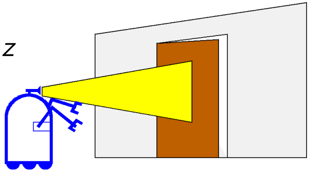
$$\Rightarrow$$

$$\boxed{p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{\textit{likelihood} \cdot \textit{prior}}{\textit{evidence}}}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \eta p(y|x)p(x)$$

$$\eta = p(y)^{-1} = \frac{1}{\sum_x p(y|x)p(x)}$$

Simple example of state estimation



- Assume 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

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

$$p(open|z_2 z_1) = \frac{p(z_2|open)p(open|z_1)}{p(z_2|open)p(open|z_1) + p(z_1|\neg open)p(\neg open|z_1)}$$

$$= \frac{\frac{1}{2}\cdot\frac{2}{3}}{\frac{1}{2}\cdot\frac{2}{3} + \frac{3}{5}\cdot\frac{1}{3}} = \frac{5}{8} = 0.625$$

- z_2 lowers the probability that the door is open.

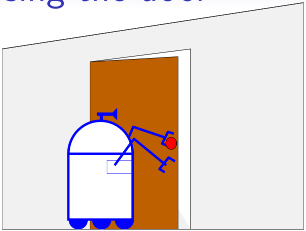
Actions

- 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

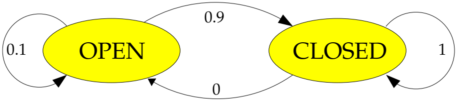
$$p(x|u, x')$$

- 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"



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

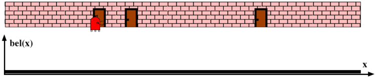
If the door is open, the action "close door" succeeds in 90% of all cases.

Continuing the example - closing the door

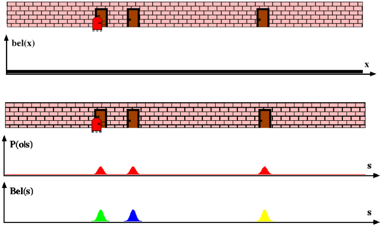
$$
\begin{aligned}
p(closed|u) &= \sum_{x'} p(closed|u,x')p(x') \\
&= p(closed|u,open)p(open) \\
&+ p(closed|u,closed)p(closed) \\
&= \frac{9}{10}\cdot\frac{5}{8}+\frac{1}{1}\cdot\frac{3}{8}=\frac{15}{16}
\end{aligned}
$$

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

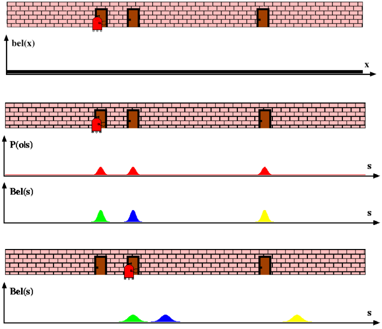
Motivation



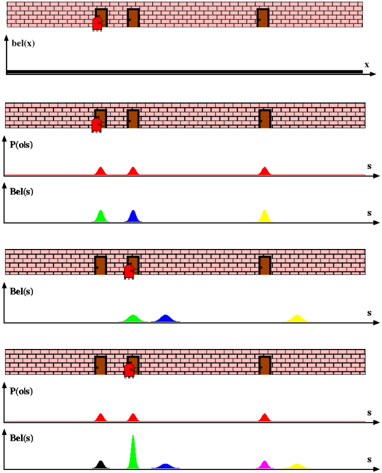
Motivation



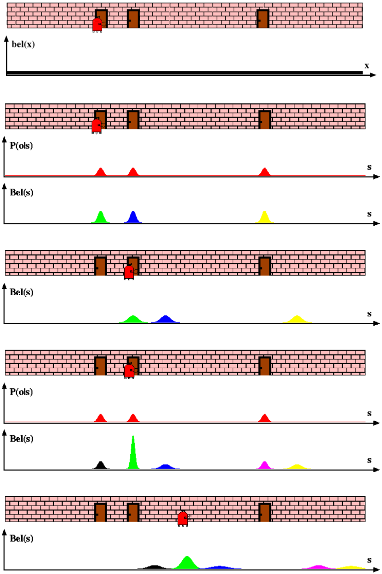
Motivation



Motivation



Motivation



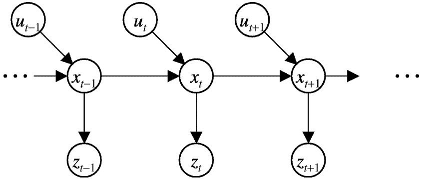
Bayes filter: the framework

- Given:
    - Stream of observations z and actions u:

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

    - 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: $Bel(x_t) = p(x_t|u_1, z_1, \ldots, u_t, z_t)$

Markov assumption



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

Underlying assumptions

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

Bayes filter - derivation

$$
\begin{aligned}
\boxed{Bel(x_t)} \quad &= p(x_t|u_1, z_1, \ldots, u_t, z_t) \\
\text{Bayes} \quad &= \eta p(z_t|x_t, u_1, z_1, \ldots, u_t) p(x_t|u_1, z_1, \ldots, u_t) \\
\text{Markov} \quad &= \eta p(z_t|x_t) p(x_t|u_1, z_1, \ldots, u_t) \\
\text{Total prob.} \quad &= \eta p(z_t|x_t) \int p(x_t|u_1, z_1, \ldots, u_t, x_{t-1}) \\
&\quad p(x_{t-1}|u_1, z_1, \ldots, u_t) dx_{t-1} \\
\text{Markov} \quad &= \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) p(x_{t-1}|u_1, z_1, \ldots, u_t) dx_{t-1} \\
\text{Markov} \quad &= \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) p(x_{t-1}|u_1, z_1, \ldots, z_{t-1}) dx_{t-1} \\
&= \boxed{\eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}}
\end{aligned}
$$

Bayes filter

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

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

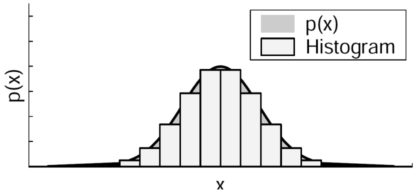
return $Bel'(x)$

Bayes filters are familiar

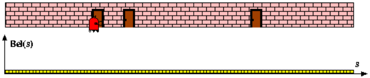
- Kalman filters
- Histogram filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

Non-parametric filters

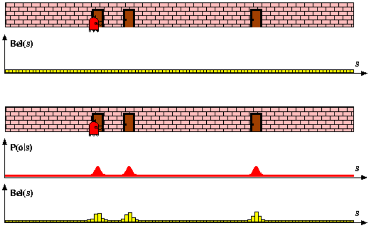
- Don't rely on a fixed functional form of the posterior.
- Approximation of probability density by a finite number of values.
- Adaptive (based on discretization), they handle nonlinearities.
- The number of samples biases the speed of the algorithm and the quality of the filter.
- Histogram x particle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



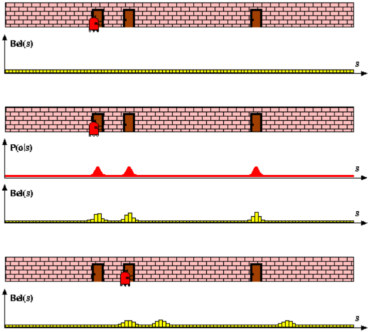
Histogram filter



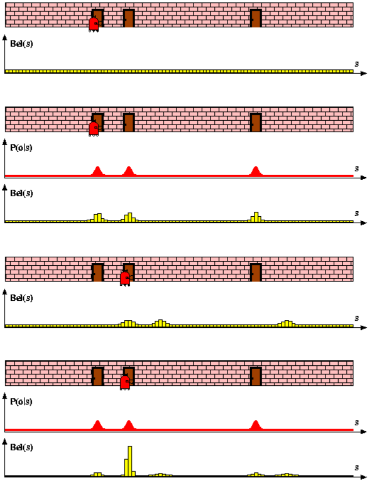
Histogram filter



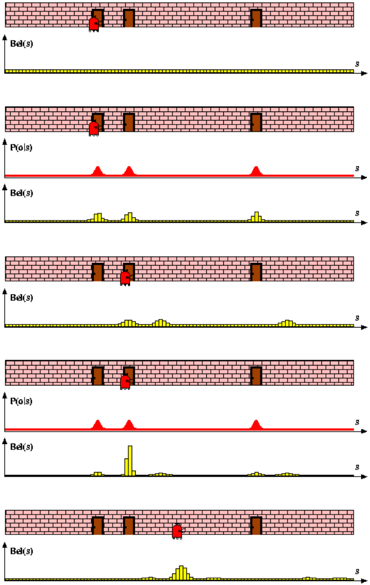
Histogram filter



Histogram filter

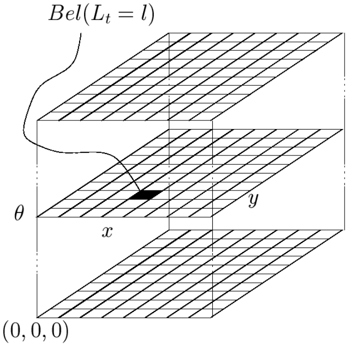


Histogram filter



Histogram filter

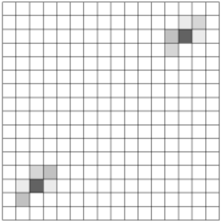
if d is a measurement z **then**
  $\eta = 0$
  for all x **do**
    $Bel'(x) = p(z|x)Bel(x)$
    $\eta = \eta + Bel'(x)$
  end for
  for all x **do**
    $Bel'(x) = \eta^{-1}Bel'(x)$
  end for
else if d is a action u **then**
  for all x **do**
    $Bel'(x) = \int p(x|u,x')Bel(x')dx'$
  end for
end if
return $Bel'(x)$

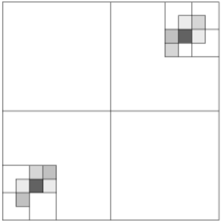


$Bel(x_t = \langle x, y, \phi \rangle)$

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)



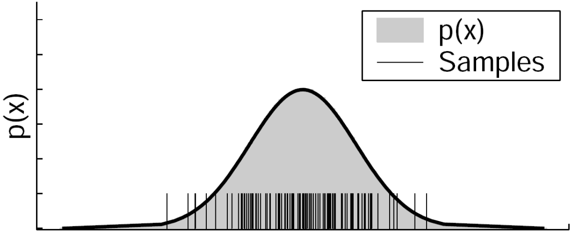


Particle filter

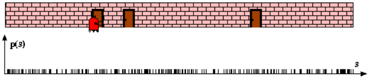
- Probability density represented by "appropriately" (randomly) placed particles:

$$Bel(x_t) \approx \left\{x_{(i)}, w_{(i)}\right\}_{i=1,\ldots,m}$$

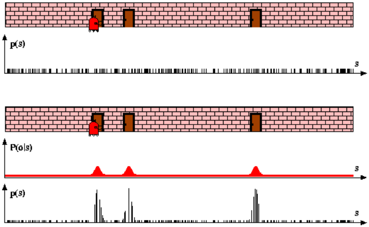
- Particles are weighted.
- Particles for time t are chosen according to the weights in time $t-1$.
- Really simple to implement.
- Most universal Bayes filter: representation of non-Gaussian distributions and non-linear processes



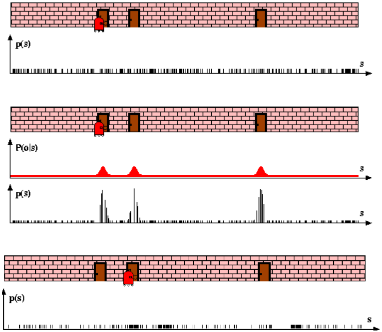
Particle filter



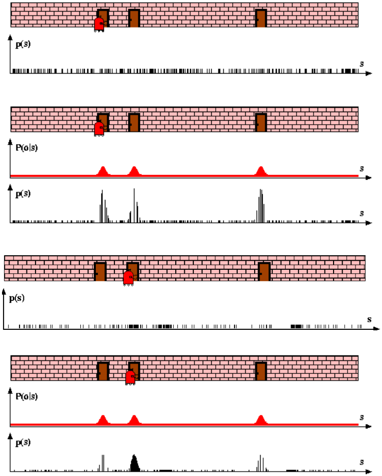
Particle filter



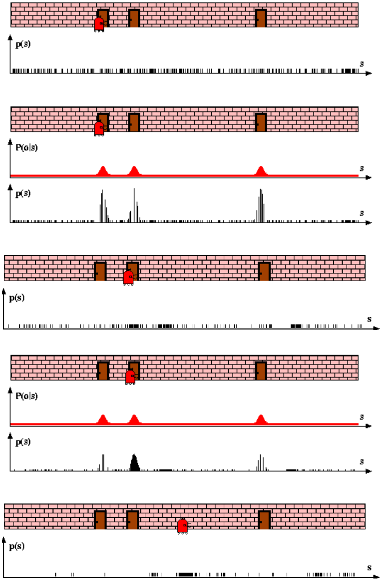
Particle filter



Particle filter



Particle filter



Particle filter - the algorithm

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

$S_t = \emptyset$, $\eta = 0$

for $i = 1, \ldots, n$ **do** — Generate new particles

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

  Sample x_t^i z $p(x_t | x_{t-1}, u_{t-1})$ 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

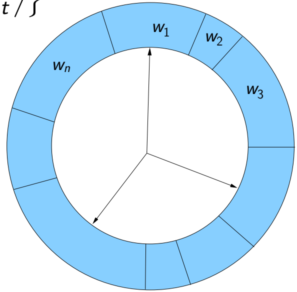
  $S_t = S_t \cup \{\langle x_t^i, w_t^i \rangle\}$ — Insert a particle

end for

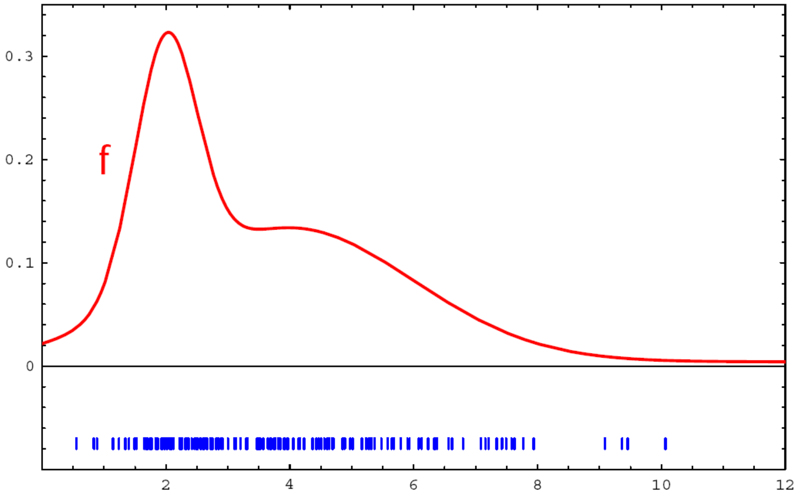
for $i = 1, \ldots, n$ **do** — Normalize weights

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

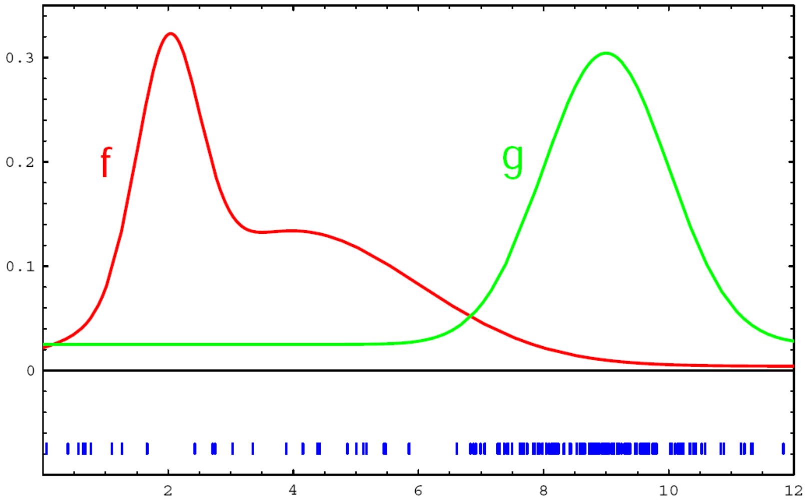
end for



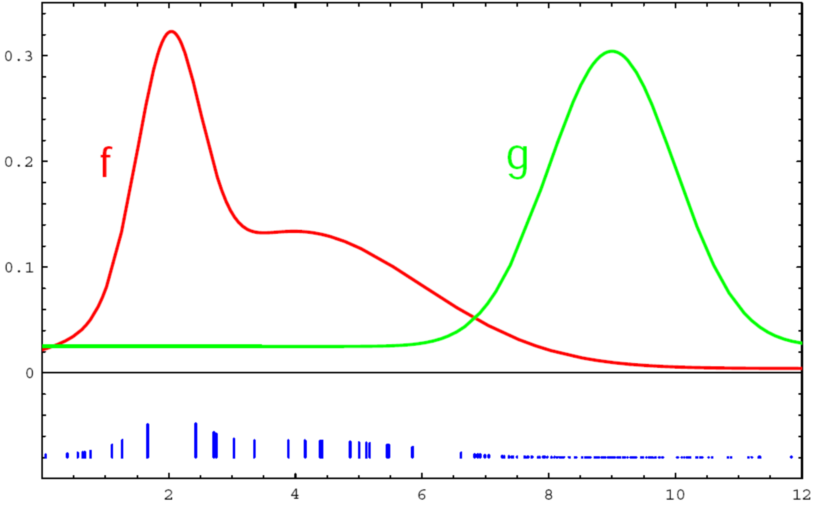
Importance sampling



Importance sampling



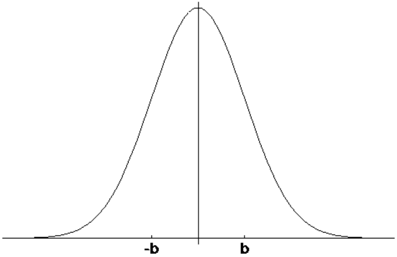
Importance sampling



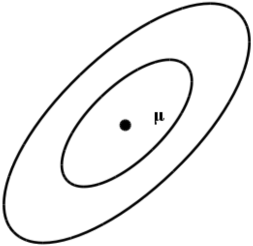
Video 1 Video 2

Kalman filter

- Unimodal representation of probability density by a Gaussian



$$
\begin{aligned}
p(x) &\sim N(\mu, \sigma^2) \\
p(x) &= \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
p(x) &\sim N(\mu, \boldsymbol{\Sigma}) \\
p(x) &= \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x
\end{aligned}
$$



Linear transformation

- Linear transformation preserves normal distribution.

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

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

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

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

Fundamental assumptions

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

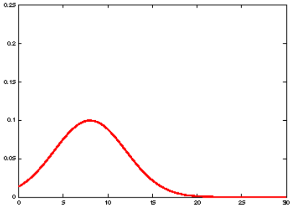
$$x_t = Ax_{t-1} + B_t u_t + \varepsilon_t$$

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

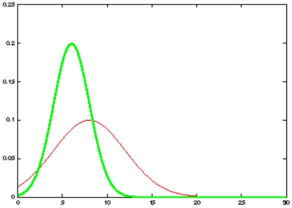
$$z_t = Cx_t + \delta_t$$

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

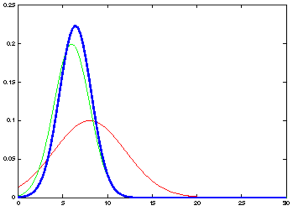
Kalman filter



A-priory belief

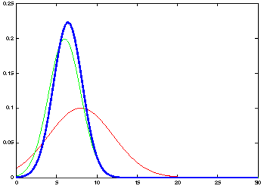


New measurement

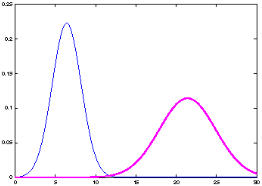


Integration of the new measurement

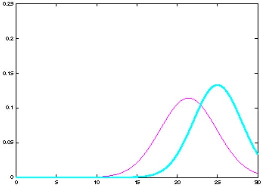
Kalman filter



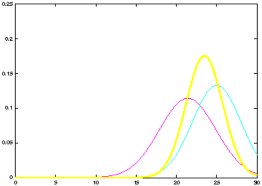
Actual belief



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)

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

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

Correction (measurement integration)

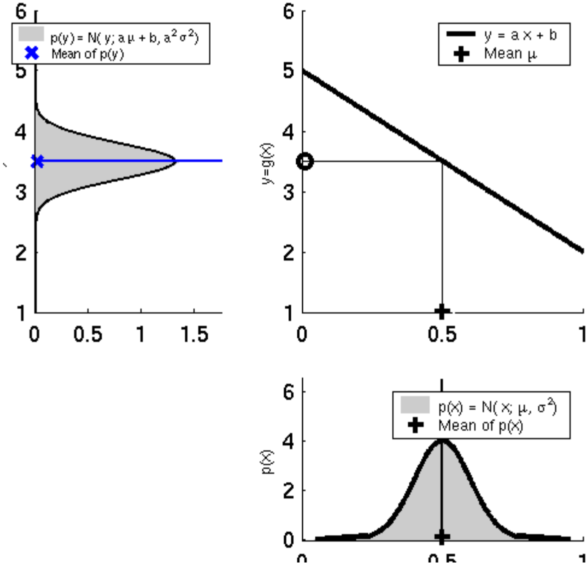
$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$

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

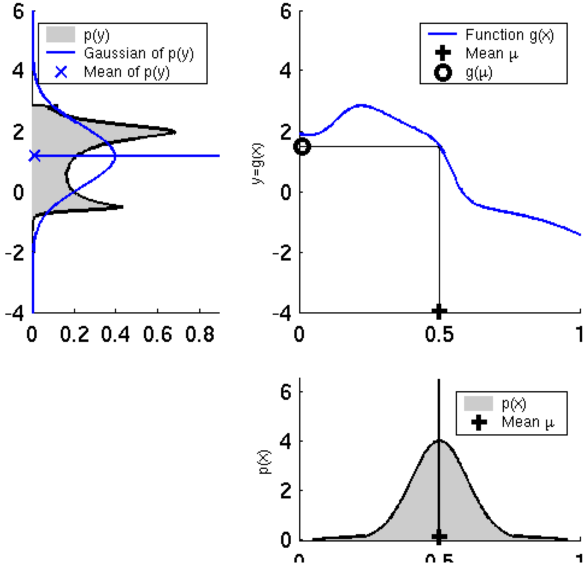
$\Sigma_t = (E - K_t C_t) \bar{\Sigma}_t$

return μ_{t,Σ_t}

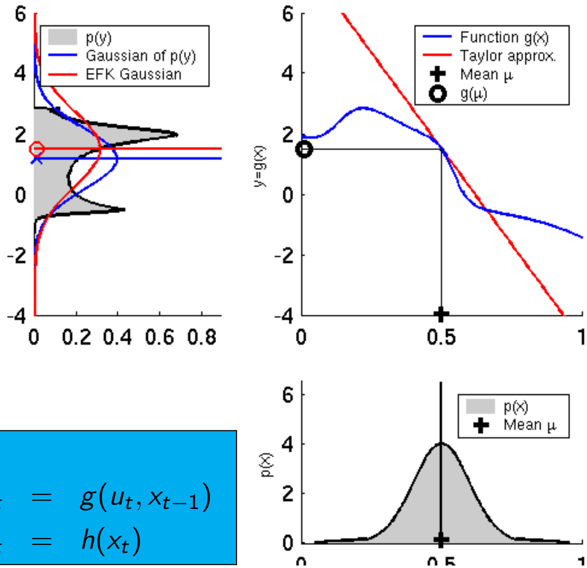
Kalman filter - linear transformation



Non-linear transformation



Extended Kalman filter



$$
\begin{aligned}
x_t &= g(u_t, x_{t-1}) \\
z_t &= h(x_t)
\end{aligned}
$$

Extended Kalman filter

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

$$\begin{aligned} g(u_t, x_t - 1) &\approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1}) \\ g(u_t, x_t - 1) &\approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \end{aligned}$$

- Correction:

$$\begin{aligned} h(x_t) &\approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_t - \bar{\mu}_t) \\ h(x_t) &\approx h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t) \end{aligned}$$

Extended Kalman filter - the algorithm

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

Prediction (action integration)

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

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

Correction (measurement integration)

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

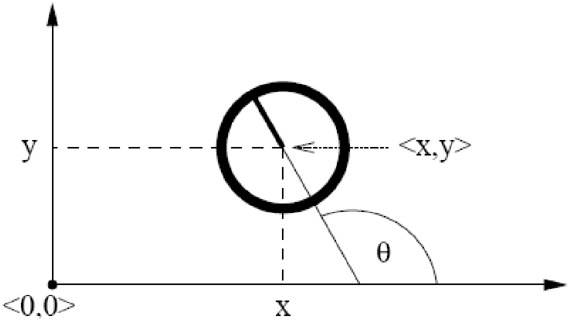
$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$

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

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.)



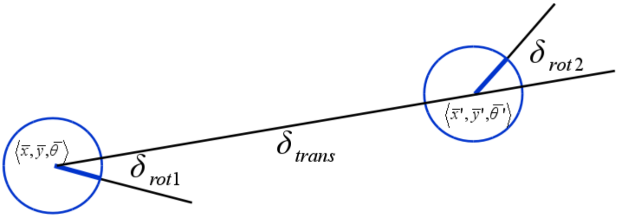
Odometry-based motion model

- 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 $u = \langle \delta_{rot_1}, \delta_{rot_2}, \delta_{trans} \rangle$

$$
\begin{aligned}
\delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\
\delta_{rot_1} &= atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\phi} \\
\delta_{rot_2} &= \bar{\phi}' - \bar{\phi} - \delta_{rot_1}
\end{aligned}
$$



Adding noise

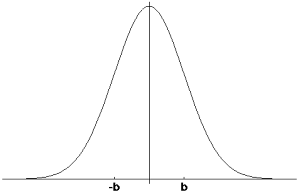
- The measured motion is given by the true motion corrupted with noise.

$$
\begin{aligned}
\hat{\delta}_{rot_1} &= \delta_{rot_1} + \varepsilon_{\alpha_1|\delta_{rot_1}|+\alpha_2|\delta_{trans}|} \\
\hat{\delta}_{trans} &= \delta_{trans} + \varepsilon_{\alpha_3|\delta_{trans}|+\alpha_4(|\delta_{rot_1}|+|\delta_{rot_2}|)} \\
\hat{\delta}_{rot_2} &= \delta_{rot_2} + \varepsilon_{\alpha_1|\delta_{rot_2}|+\alpha_2|\delta_{trans}|}
\end{aligned}
$$

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

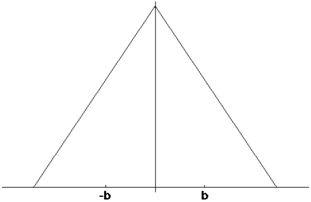
Typical distributions for probabilistic motion models

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

$u = \langle \delta_{rot_1}, \delta_{rot_2}, \delta_{trans} \rangle, x = \langle x, y, \phi \rangle => x' = \langle x', y', \phi' \rangle$

Random control

$$
\begin{aligned}
\hat{\delta}_{rot_1} &= \delta_{rot_1} + sample(\alpha_1|\delta_{rot_1}| + \alpha_2|\delta_{trans}|) \\
\hat{\delta}_{trans} &= \delta_{trans} + sample(\alpha_3|\delta_{trans}| + \alpha_4(|\delta_{rot_1}| + |\delta_{rot_2}|)) \\
\hat{\delta}_{rot_2} &= \delta_{rot_2} + sample(\alpha_1|\delta_{rot_2}| + \alpha_2|\delta_{trans}|)
\end{aligned}
$$

Position determination

$$
\begin{aligned}
x' &= x + \hat{\delta}_{trans} cos(\phi + \hat{\delta}_{rot_1}) \\
y' &= y + \hat{\delta}_{trans} sin(\phi + \hat{\delta}_{rot_1}) \\
\phi' &= \phi + \hat{\delta}_{rot_1} + \hat{\delta}_{rot_1}
\end{aligned}
$$

return $\langle x', y', \phi' \rangle$

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

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

Odometry values (u)

$$\begin{aligned}
\delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\
\delta_{rot_1} &= atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\phi} \\
\delta_{rot_2} &= \bar{\phi}' - \bar{\phi} - \delta_{rot_1}
\end{aligned}$$

Ideal case

$$\begin{aligned}
\hat{\delta}_{trans} &= \sqrt{(x' - x)^2 + (y' - y)^2} \\
\hat{\delta}_{rot_1} &= atan2(y' - y, x' - x) - \phi \\
\hat{\delta}_{rot_2} &= \phi' - \phi - \delta_{rot_1}
\end{aligned}$$

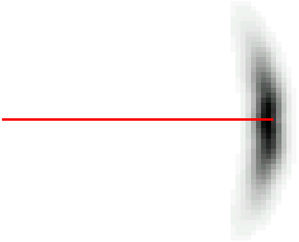
Probability calculation

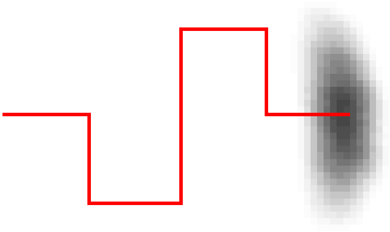
$$\begin{aligned}
p_1 &= prob(\delta_{rot_1} - \hat{\delta}_{rot_1}, \alpha_1|\delta_{rot_1}| + \alpha_2\delta_{trans}) \\
p_2 &= prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\delta_{trans} + \alpha_4(|\delta_{rot_1}| + |\delta_{rot_2}|)) \\
p_3 &= prob(\delta_{rot_2} - \hat{\delta}_{rot_2}, \alpha_1|\delta_{rot_2}| + \alpha_2\delta_{trans})
\end{aligned}$$

return $p_1 p_2 p_3 \longleftarrow$ independence assumption

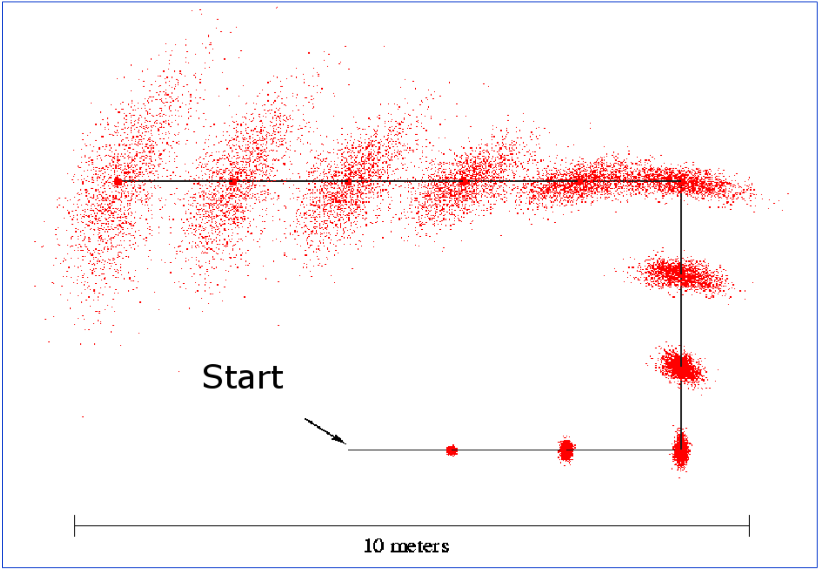
Application

- 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)



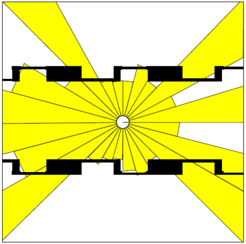


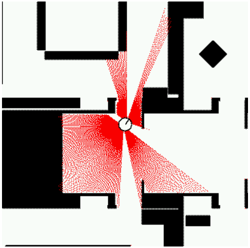
Trajectory composition



Sensor model

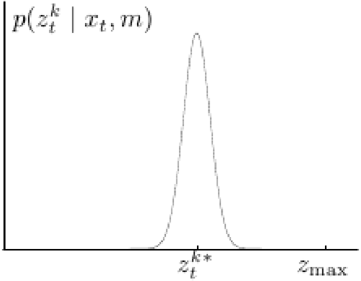
- The aim is to determine $p(z|m, x)$.
- We will use proximity sensor (laser, sonar).
- Scan is composed from k measurements (beams): $z = \{z_1, z_2, \ldots, z_k\}$
- Individual measurements are independent given the robot position (strong assumption): $P(z|x, m) = \prod_{k=1} P(z_k|x, m)$





Beam-based model - components

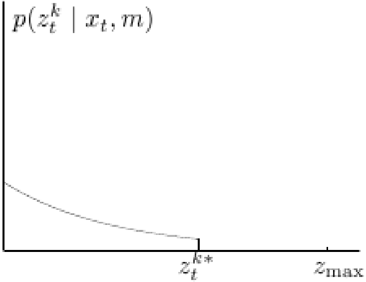
Measurement noise



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

Normal distribution

Unexpected obstacles

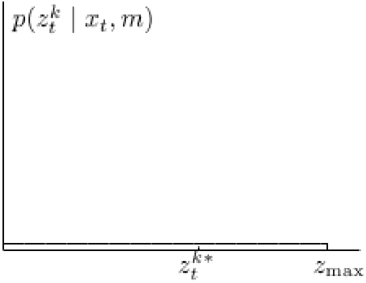


$$p_{short}(z|x,m) = \begin{cases} \eta \lambda_{short} e^{-\lambda_{short} z} & \text{if } 0 \leq z \leq z^* \\ 0 & \text{otherwise} \end{cases}$$

Exponential distribution

Beam-based model - components

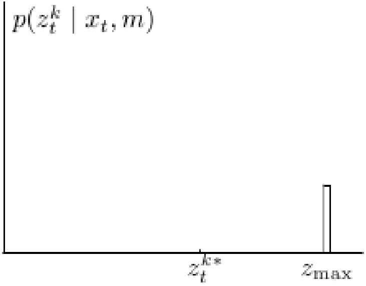
Random measurement



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

Uniform distribution

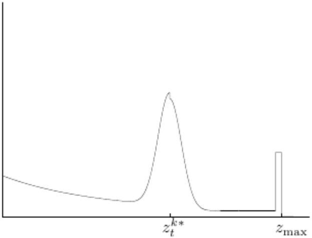
Max range



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

Discrete distribution :-(

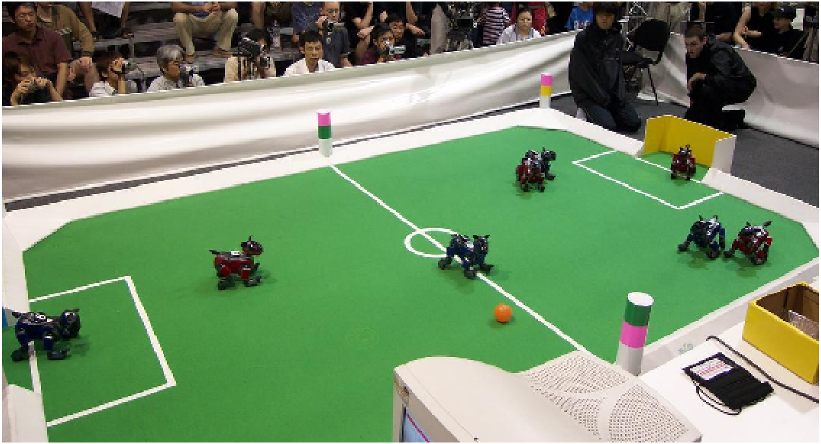
Resulting mixture density



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

- 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 $=>$ approximate determination of probability density can miss the right state.

Motivation



EKF-based localization

- Velocity motion model ($u = (v, \omega)$)

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

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

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

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

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

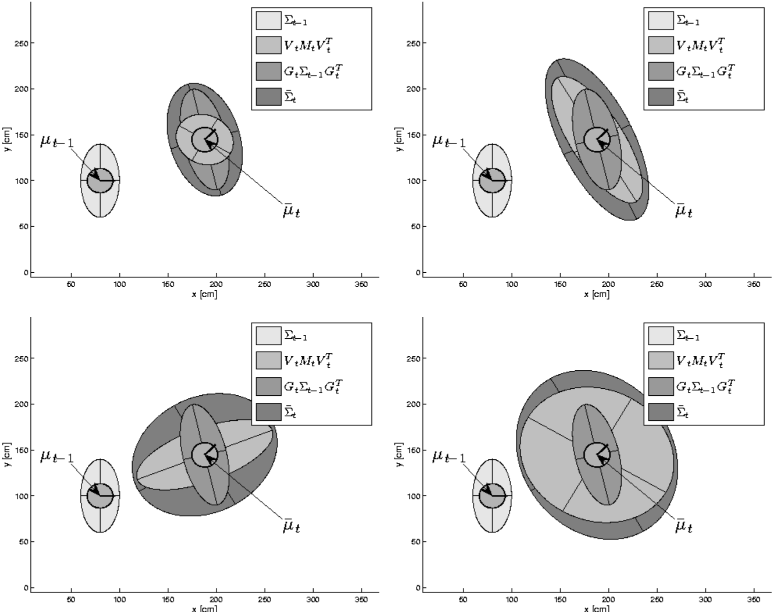
Predicted mean

$$\overline{\mu_t} = g(u_t, \mu_{t-1})$$

Predicted covatiance

$$\overline{\Sigma_t} = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$$

Prediction


Σ_{t-1}
$V_t M_t V_t^T$
$G_t \Sigma_{t-1} G_t^T$
$\bar{\Sigma}_t$
μ_{t-1}
$\bar{\mu}_t$
x [cm]
y [cm]

Correction

Predicted measurement mean

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

Jacobian of h w.r.t location

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

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$$

Predicted measurement covariance

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

Gain

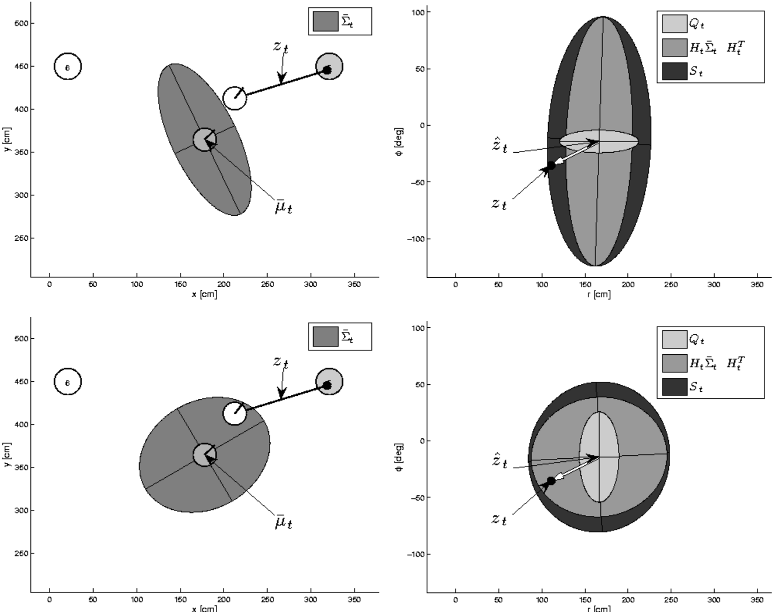
$$K_t = \overline{\Sigma}_t H_t^T S_t^{-1}$$

Updated mean and covariance

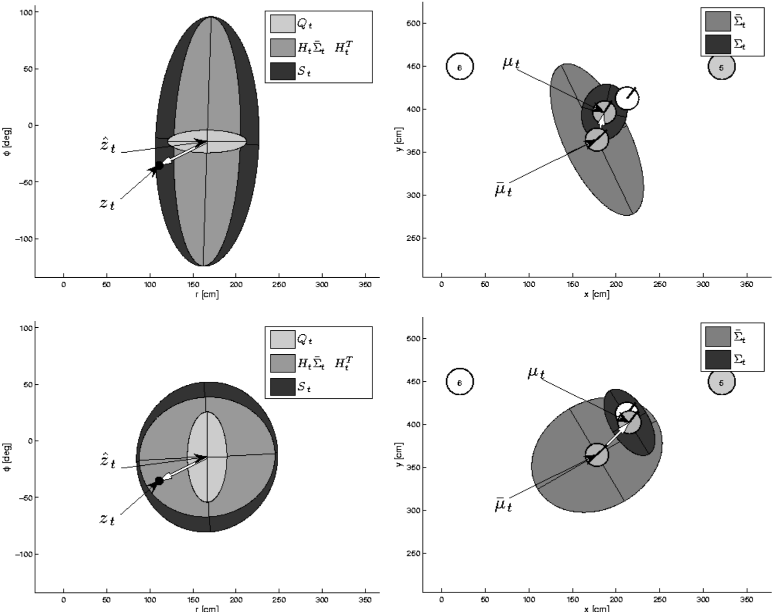
$$\mu_t = \overline{\mu}_t + K_t(z_t - \hat{z}_t)$$

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

Observation

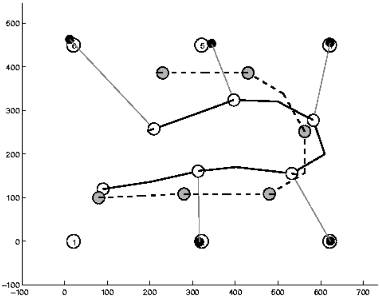


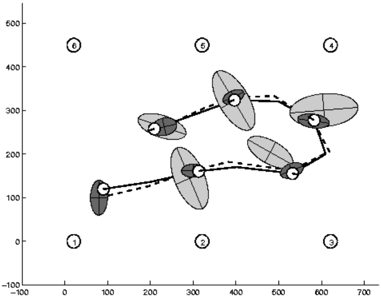
Correction



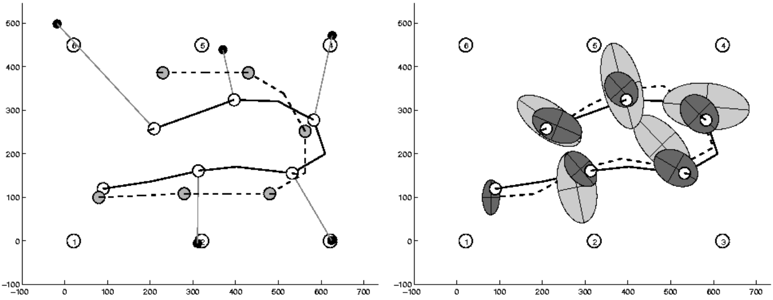
Q_t
$H_t \bar{\Sigma}_t \; H_t^T$
S_t
$\hat{z}_t$
z_t
φ [deg]
r [cm]
$\bar{\Sigma}_t$
Σ_t
μ_t
$\bar{\mu}_t$
y [cm]
x [cm]
6
5

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.