TSRT14: Sensor Fusion
Lecture 7

— Point-mass filter (PMF)
— Particle filter (PF)

Gustaf Hendeby

gustaf.hendeby@liu.se

LINKÖPING UNIVERSITY

Le 7: point-mass filter and particle filter

Whiteboard:

- Derivation and explanation of the PMF and PF.

Slides:

- Point-mass filter (PMF)
- Particle filter (PF)
- Examples and applications

Lecture 6: summary

Key tool for a unified derivation of KF, EKF, UKF.

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy} & P_{yy} \end{pmatrix}\right)$$
$$\implies (X|Y=y) \sim \mathcal{N}(\mu_x + P_{xy}P_{yy}^{-1}(y-\mu_y), P_{xx} - P_{xy}P_{yy}^{-1}P_{yx}).$$

The Kalman gain is in this notation given by $K_k = P_{xy}P_{yy}^{-1}$.

- In KF, P_{xy} and P_{yy} follow from a linear Gaussian model.
- In EKF, P_{xy} and P_{yy} can be computed from a linearized model (requires analytic gradients).
- In EKF and UKF, P_{xy} and P_{yy} computed by NLT for transformation of $(x^T, v^T)^T$ and $(x^T, e^T)^T$, respectively. No gradients required, just function evaluations.

Point-Mass Filter

LINKÖPING UNIVERSITY

Chapter 9 Overview

Particle filter

- Algorithms and derivation.
- Practical and theoretical issues.
- Computational aspects.
- Marginalization to beat the curse of dimensionality.

Model and Bayesian Recursion

General nonlinear state-space model:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, v_k) \\ y_k &= h(x_k, u_k, e_k) \end{aligned}$$

or even more general Markov model:

$$\begin{aligned} x_k | x_{k-1} &\sim p(x_k | x_{k-1}) \\ y_k | x_k &\sim p(y_k | x_k) \end{aligned}$$

Bayes Optimal Filter: measurement update

Bayes' rule

$$p(A|B,C) = \frac{p(B|A,C)p(A|C)}{p(B|C)},$$

using $A = x_k$, $(B,C) = y_{1:k}$, $B = y_k$ and $C = y_{1:k-1}$, yields

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k, y_{1:k-1})p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}.$$

The Markov property $p(y_k|x_k, y_{1:k-1}) = p(y_k|x_k)$ is used in the last equality.

Bayes Optimal Filter: time update

The definition of conditional probability $p(A, B|C) = p(A|B,C)p(B|C)$ gives

$$p(x_{k+1}, x_k | y_{1:k}) = p(x_{k+1}|x_k, y_{1:k})\, p(x_k|y_{1:k}) = p(x_{k+1}|x_k)\, p(x_k|y_{1:k}).$$

Again, the Markov property is used in the last step.

Marginalization of x_k by integrating both sides with respect to x_k yields

$$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})\, dx_k.$$

This is known as the *Chapman-Kolmogorov* equation.

Bayes Optimal Filter: summary

General Bayesian recursion (time and measurement updates)

$$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})\,dx_k,$$

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}.$$

- Analytic solution available in a few special cases: linear Gaussian model (KF) and finite state-space model (HMM).
- However, for a given trajectory $x_{1:k}$, the recursion can be computed (neglect the integral).
- We can numerically evaluate different trajectories by comparing their likelihoods. But there are many possible trajectories, so Monte Carlo sampling of trajectories directly does not work.

Numerical Approximation

Basic idea: postulate a discrete approximation of the posterior. For the predictive density, we have

$$\hat{p}(x_k|y_{1:k-1}) = \sum_{i=1}^{N} w_{k|k-1}^{(i)} \delta(x_k - x_k^{(i)}).$$

The first moments (mean and covariance) are simple to compute from this approximation:

$$\hat{x}_{k|k-1} = \mathsf{E}(x_k) = \sum_{i=1}^{N} w_{k|k-1}^{(i)} x_k^{(i)},$$

$$P_{k|k-1} = \mathsf{cov}(x_k) = \sum_{i=1}^{N} w_{k|k-1}^{(i)} (x_k^{(i)} - \hat{x}_{k|k-1})(x_k^{(i)} - \hat{x}_{k|k-1})^T.$$

Also, the MAP estimate can be useful:

$$\hat{x}_{k|k-1}^{\mathrm{MAP}} = \arg\max_{x_k^{(i)}} \hat{p}(x_k|y_{1:k-1}).$$

Measurement Update

The measurement follows directly, without any extra approximations

$$\hat{p}(x_k|y_{1:k}) = \sum_{i=1}^{N} \underbrace{\frac{1}{c_k} p(y_k|x_k^{(i)}) w_{k|k-1}^{(i)}}_{w_{k|k}^{(i)}} \delta(x_k - x_k^{(i)})$$

$$c_k = \sum_{i=1}^{N} p(y_k|x_k^{(i)}) w_{k|k-1}^{(i)}.$$

The normalization constant c_k corresponds to assuring that $\sum_{i=1}^{N} w_{k|k}^{(i)} = 1$.

Time Update

Bayesian time update gives a continuous distribution

$$\hat{p}(x_{k+1}|y_{1:k}) = \sum_{i=1}^{N} w_{k|k}^{(i)} p(x_{k+1}|x_k^{(i)}).$$

To keep the approximation form, the distribution is sampled at points $x_{k+1}^{(i)}$, and the weights are updated as

$$w_{k+1|k}^{(i)} = \hat{p}(x_{k+1}^{(i)}|y_{1:k}) = \sum_{j=1}^{N} w_{k|k}^{(j)} p(x_{k+1}^{(i)}|x_k^{(j)}), \quad i = 1, 2, \ldots, N.$$

Two principles:

- Keep the same grid, so $x_{k+1}^{(i)} = x_k^{(i)}$, which yields the point mass filter.
- Generate new samples from the posterior distribution $x_{k+1}^{(i)} \sim \hat{p}(x_{k+1}|y_{1:k})$, which yields the marginal particle filter.

Both alternatives have quadratic complexity (N weights $w_{k+1|k}^{(i)}$, each one involving a sum with N terms).

Point-Mass Filter (PMF)

Advantages:

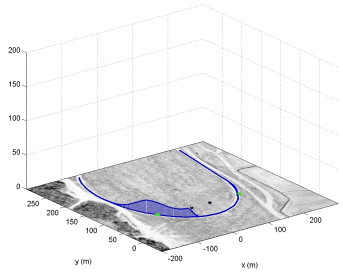
- Simple to implement.
- Works excellent when $n_x \leq 2$.
- Gives the complete posterior, not only $\hat{x}$ and P.
- Global search, no local minima.

Problems:

- Grid inefficient in higher dimensions, since the probability to be at one grid point depends on the transition probability from all other grid points.
- The grid should be adaptive (rough initially, then finer).
- Sparse matrices can be used for multi-modal distributions.

PMF Example: 2 DOA sensors, 2 targets

- Data from the FOCUS project (map overlay).
- Two microphone arrays (black x) compute two DOA.
- Two road-bound targets (green *).
- One grid point (stem plot) every meter on the road.



http://youtu.be/VcdI

Particle Filter

Particle Filter

- Trick to avoid cubic complexity: sample trajectories, not states
- Time update for trajectory:

$$p(x^{(i)}_{1:k+1}|y_{1:k}) = \underbrace{p(x^{(i)}_{k+1}|x^{(i)}_{1:k}, y_{1:k})}_{p(x^{(i)}_{k+1}|x^{(i)}_{k})} \underbrace{p(x^{(i)}_{1:k}|y_{1:k})}_{w^{(i)}_{k|k}}$$
$$= w^{(i)}_{k|k} p(x^{(i)}_{k+1}|x^{(i)}_{k}) = w^{(i)}_{k+1|k}.$$

  No sum involved here!

- The new sample is sampled from the prior in the original PF (*sequential importance resampling* (SIR) or bootstrap PF)

$$x^{(i)}_{k+1} \sim p(x_{k+1}|x^{(i)}_{k}).$$

Basic SIR PF Algorithm

Choose the number of particles N.

Initialization: Generate $x_0^{(i)} \sim p_{x_0}, i = 1, \ldots, N$ particles.

Iterate for $k = 1, 2, \ldots, t$:

1. *Measurement update:* For $k = 1, 2, \ldots,$
$$w_k^{(i)} = w_{k-1}^{(i)} p(y_k | x_k^{(i)}).$$
2. *Normalize:* $w_k^{(i)} := w_k^{(i)} / \sum_j w_k^{(j)}$.
3. *Estimation:* MMSE $\hat{x}_k \approx \sum_{i=1}^N w_k^{(i)} x_k^{(i)}$ or MAP.
4. *Resampling:* Bayesian bootstrap: Take N samples with replacement from the set $\{x_k^{(i)}\}_{i=1}^N$ where the probability to take sample i is $w_k^{(i)}$. Let $w_k^{(i)} = 1/N$.
5. *Prediction:* Generate random process noise samples
$$v_k^{(i)} \sim p_{v_k}, \quad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}).$$

PF Code

Input arguments: NL object `m`, SIG object `z`.

Output arguments: SIG object `zhat`.

```
y = z.y.';
u = z.u.';
xp = m.x0.' + rand(m.px0, Np);  % Initialization
for k = 1:N
   % Time update
   v = rand(m.pv, Np);  % Random process noise
   xp = m.f(k, xp.', u(:,k), m.th).' + v;  % State prediction
   % Measurement update
   yp = m.h(k, xp.', u(:, k).', m.th).';  % Measurement prediction
   w = pdf(m.pe, y(:,k).'-yp);  % Likelihood
   xhat(k,:) = mean(w(:).*xp);  % Estimation
   [xp, w] = resample(xp, w);  % Resampling
   xMC(:, k, :) = xp;  % MC uncertainty repr.
end
```


Example: 1D terrain navigation

Problem: Measured velocity u_k, unknown velocity disturbance v_k, known altitude profile $h(x)$ which is observed with y_k.

Model:

$$x_{k+1} = x_k + u_k + v_k,$$
$$y_k = h(x_k) + e_k,$$

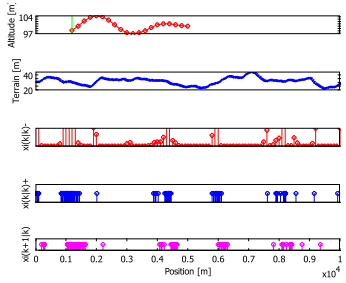
y_1

$h(x)$

Step 1. MUP $p(x_1|y_1)$

Step 4. Resampling $p(x_1|y_1)$

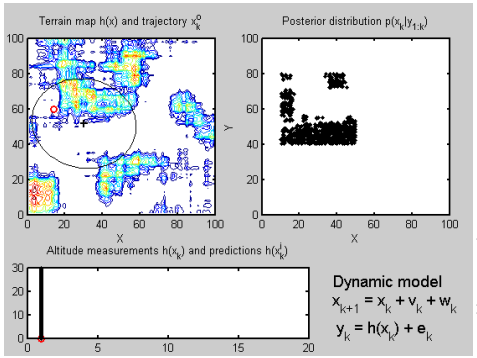
Step 5. TUP $p(x_2|y_1)$



http://youtu.be/tlNhOE6tmV0

Example: 2D terrain navigation

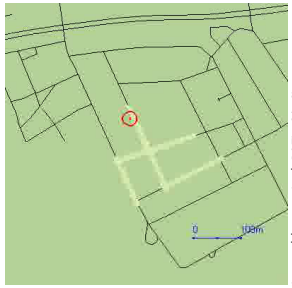
Same assumptions as in 1D: aircraft measures ground altitude as measurement y_k and noisy speed $u_k = v_k + w_k$, terrain elevation map (TAM) provides $h(x_k)$.



http://youtu.be/1uT8U22Pmw0

Example: 2D street navigation by odometry

Wheel speed sensors provide speed and yaw rate, street map provides constraints on turns.

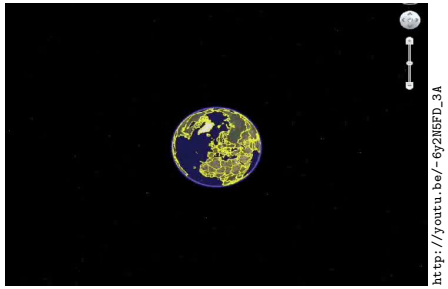


http://youtu.be/aLOTbt9Aw6k

Video illustrates how a uniform prior over a part of the road network eventually converge to a single particle cluster when sufficient information is obtained (but many local particle clusters initially).

Example: 2D street navigation by fingerprinting

WiMAX network (Brussels here) provides signal strength measurements, RSS map provides a fingerprint $h(x_k)$, street map (optional) provides constraints on turns.

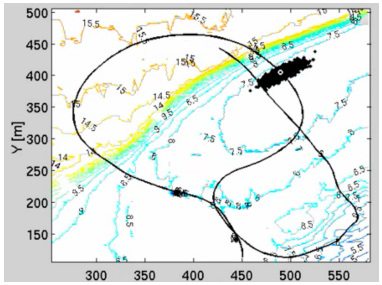


http://youtu.be/-6y2N5FD_3A

First half of video shows PF without street constraint, second half with road constraint (with superior performance).

Example: 2D underwater navigation

Underwater vessel measures its own depth and distance to bottom, and sea chart provides depth $h(x_k)$.

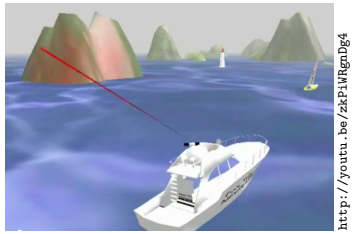


http://youtu.be/JxUjVEn87yE

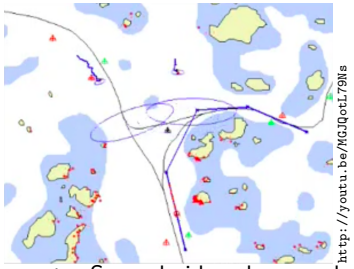
Video shows how a uniform prior quickly converges to a unimodal particle cloud. Note how the cloud changes form when passing the ridge.

Example: 2D surface navigation

On-board radar measures range to shore and possible its speed, sea chart provides conditional distance to shore $h(x_k)$.



http://youtu.be/zkP1WRgnDg4



http://youtu.be/MGQJotL79Ms

First video shows animated ship and radar measurements. Second video shows radar measurements overlayed on sea chart (given estimated position), the estimated (PF) position of the own ship, and the estimated (EKF) positions of other ships and 1 minute prediction of their position (collision warning).

Summary

LINKÖPING UNIVERSITY

Lecture 7: summary

Basic SIR PF algorithm

Choose the number of particles N.

- *Initialization:* Generate $x_0^{(i)} \sim p_{x_0}, i = 1, \ldots, N$, particles.

Iterate for $k = 1, 2, \ldots, t$:

1. *Measurement update:* For $k = 1, 2, \ldots,$
$$\bar{w}_{k|k}^{(i)} = w_{k|k-1}^{(i)} p(y_k | x_k^{(i)}).$$
2. *Normalize:* $w_{k|k}^{(i)} := \bar{w}_{k|k}^{(i)} / \sum_j \bar{w}_{k|k}^{(j)}$.
3. *Estimation:* MMSE $\hat{x} \approx \sum_{i=1}^{N} w^{(i)} x^{(i)}$ or MAP.
4. *Resampling:* Bayesian bootstrap: Take N samples with replacement from the set $\{x_k^{(i)}\}_{i=1}^{N}$ where the probability to take sample i is $w_{k|k}^{(i)}$ Let $w_{k|k}^{(i)} = 1/N$.
5. *Prediction:* Generate random process noise samples
$$v_k^{(i)} \sim p_{v_k}, \qquad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}) \qquad w_{k+1|k} = w_{k|k}.$$