TSRT14: Sensor Fusion
Lecture 8

— Particle filter theory
— Marginalized particle filter

Gustaf Hendeby

gustaf.hendeby@liu.se

LINKÖPING
UNIVERSITY

Le 8: particle filter theory, marginalized particle filter

Whiteboard:

- PF tuning and properties

Slides:

- Proposal densities and SIS PF
- Marginalized PF (MPF)

Lecture 7: summary

Basic SIR PF algorithm

Choose the number of particles N.

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

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

1. *Measurement update:* For $k = 1, 2, \dots,$

$$\bar{w}_{k|k}^{(i)} = w_{k|k-1}^{(i)} p(y_k | x_k^{(i)}).$$

2. *Normalize:* $w_{k|k}^{(i)} := \bar{w}_{k|k}^{(i)} / \sum_i \bar{w}_{k|k}^{(i)}$.
3. *Estimation:* MMSE $\hat{x}_{k|k} \approx \sum_{i=1}^{N} w_{k|k}^{(i)} x_k^{(i)}$ 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 $w_{k|k}^{(i)}$. Let $w_{k|k}^{(i)} = 1/N$.
5. *Prediction:* Generate random process noise samples

$$v_k^{(i)} \sim p_{v_k}, \qquad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}) \qquad w_{k+1|k} = w_{k|k}.$$

Particle Filter Theory

LINKÖPING
UNIVERSITY

Particle Filter Design: design choices

1. Choice of N is a complexity *vs.* performance trade-off. Complexity is linear in N, while the error in theory is bounded as g_k/N, where g_k is polynomial in k but independent of n_x.
2. $N_{\text{eff}} = \frac{1}{\sum_i (w_k^{(i)})^2}$ controls how often to resample. Resample if $N_{\text{eff}} < N_{th}$. $N_{\text{th}} = N$ gives SIR. Resampling increases variance in the weights, and thus the variance in the estimate, but it is needed to avoid depletion.
3. The proposal density. An appropriate proposal makes the particles explore the most critical regions, without wasting efforts on meaningless state regions.
4. Pretending the process (and measurement) noise is larger than it is (dithering, jittering, roughening) is as for the EKF and UKF often a sound *ad hoc* solution to avoid filter divergence.

Common Particle Filter Extensions

- Main problem with basic SIR PF: **depletion**. After a while, only one or a few particles are contributing to $\hat{x}$.
- The effective number of samples, N_{eff} is a measure of this. $N_{\text{eff}} = N$ means that all particles contribute equally, and $N_{\text{eff}} = 1$ means that only one has a non-zero weight.
- Too few design parameters, more degrees of freedom:
  - Sequential importance sampling: means that you only resample when needed, $N_{\text{eff}} < N_{\text{th}}$.
  - The theory allows for a general proposal distribution $q(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k})$ for how to sample a new state in the time update. The "prior" $q(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k}) = p(x_k^{(i)}|x_{k-1}^{(i)})$ is the standard option, but there might be better ones.

SIS PF Algorithm

Choose the number of particles N, a proposal density $q(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k})$, and a threshold N_{th} (for instance $N_{th} = \frac{2}{3}N$).

- *Initialization:* Generate $x_0^{(i)} \sim p_{x_0}$ and $\omega_{1|0}^{(i)}$, $i = 1, \ldots, N$.

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

1. *Measurement update:* For $i = 1, 2, \ldots, N$: $w_{k|k}^{(i)} \propto w_{k|k-1}^{(i)} p(y_k|x_k^{(i)})$, normalize $w_{k|k}^{(i)}$.
2. *Estimation:* MMSE $\hat{x}_{k|k} \approx \sum_{i=1}^{N} w_{k|k}^{(i)} x_{k|k}^{(i)}$.
3. *Resampling:* Resample with replacement when $N_{\text{eff}} = \frac{1}{\sum_i (w_{k|k}^{(i)})^2} < N_{th}$.
4. *Prediction:* Generate samples $x_{k+1}^{(i)} \sim q(x_k|x_{k-1}^{(i)}, y_k)$, update the weights $w_{k+1|k}^{(i)} \propto w_{k|k}^{(i)} \frac{p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$, normalize $w_{k+1|k}^{(i)}$.

Sampling from Proposal Density

SIR PF samples from the prior $x_{k+1}^{(i)} \sim p(x_{k+1}|x_k^{(i)})$. In general, one can sample from any proposal density,

$$x_{k+1}^{(i)} \sim q(x_{k+1}|x_k^{(i)}, y_{k+1}).$$

Note that we are allowed to "cheat" and look at the next measurement y_{k+1} when we sample. Note that the time update can be written

$$p(x_{k+1}|y_{1:k}) = \int_{\mathbb{R}^{n_x}} q(x_{k+1}|x_k, y_{k+1}) \frac{p(x_{k+1}|x_k)}{q(x_{k+1}|x_k, y_{k+1})} p(x_k|y_{1:k})\, dx_k.$$

The new approximation becomes

$$\hat{p}(x_{1:k+1}|y_{1:k}) = \sum_{i=1}^{N} \underbrace{\frac{p(x_{k+1}^{(i)}|x_k^{(i)})}{q(x_{k+1}^{(i)}|x_k^{(i)}, y_{k+1})} w_{k|k}^{(i)}}_{w_{k+1|k}^{(i)}} \delta(x_{1:k+1} - x_{1:k+1}^{(i)}).$$

Choice of Proposal Density

1. Factorized form

$$q(x_{0:k}|y_{1:k}) = q(x_k|x_{0:k-1}, y_{1:k})q(x_{0:k-1}|y_{1:k}).$$

   In the original form, we sample *trajectories*.

2. Approximate filter form

$$q(x_{0:k}|y_{1:k}) \approx q(x_k|x_{0:k-1}, y_{1:k}).$$

   In the approximate form, we keep the previous trajectory and just append x_k.

Proposals: (1) optimal form

$$q(x_k|x_{k-1}^{(i)}, y_k) = p(x_k|x_{k-1}^{(i)}, y_k) = \frac{p(y_k|x_k)p(x_k|x_{k-1}^{(i)})}{p(y_k|x_{k-1}^{(i)})},$$

$$w_{k|k}^{(i)} = w_{k-1|k-1}^{(i)} p(y_k|x_{k-1}).$$

Optimal since the sampling process of x_k does not influence (that is, increase the variance of) the weights.

Drawbacks:

- It is generally hard to sample from this proposal density.
- It is generally hard to compute the weight update needed for this proposal density, since it would require to integrate over the whole state space to obtain something computable $p(y_k|x_{k-1}) = \int p(y_k|x_k)p(x_k|x_{k-1})\,dx$.

For linear (linearized) Gaussian likelihood and additive Gaussian process noise, the integral can be solved, leading to a (extended) KF time update.

Proposals: (2) prior

$$\begin{aligned} q(x_k|x_{k-1}^{(i)}, y_k) &= p(x_k|x_{k-1}^{(i)}), \\ w_{k|k}^{(i)} &= w_{k-1|k-1}^{(i)} p(y_k|x_k^{(i)}). \end{aligned}$$

The absolutely simplest and most common choice.

Proposals: (3) likelihood

$$
\begin{aligned}
q(x_k|x_{k-1}^{(i)}, y_k) &= p(y_k|x_k), \\
w_{k|k}^{(i)} &= w_{k-1|k-1}^{(i)} p(x_k|x_{k-1}^{(i)}).
\end{aligned}
$$

Good in high SNR applications, when the likelihood contains more information about x than the prior.

Drawback:
The likelihood is not always invertible in x.

Marginalized Particle Filter

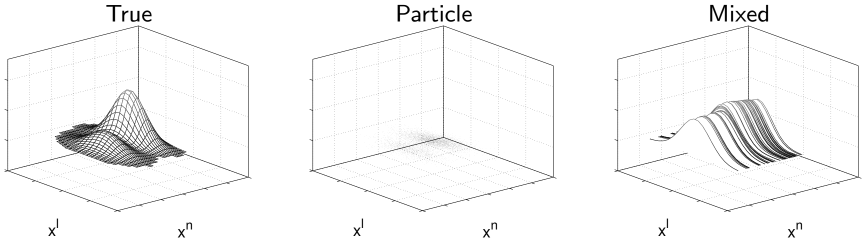
Marginalized Particle Filter (1/2)

Objective: decrease the number of particles for large state spaces (say $n_x > 3$) by utilizing partial linear Gaussian substructures.

The task of nonlinear filtering can be split into two parts:

1. Representation of the filtering probability density function.
2. Propagation of this density during the time and measurement update stages.

Possible to mix a parametric distribution in some dimensions with grid/particle represention in the other dimensions.



Marginalized Particle Filter (2/2)

- Model

$$
\begin{aligned}
x_{k+1}^n &= f_k^n(x_k^n) + F_k^n(x_k^n)x_k^l + G_k^n(x_k^n)w_k^n, \\
x_{k+1}^l &= f_k^l(x_k^n) + F_k^l(x_k^n)x_k^l + G_k^l(x_k^n)w_k^l, \\
y_k &= h_k(x_k^n) + H_k(x_k^n)x_k^l + e_k.
\end{aligned}
$$

  All of w^n, w^l, e_k and x_0^k are Gaussian. x_0^n can be general.
- Basic factorization holds: conditioned on $x_{1:k}^n$, the model is linear and Gaussian.
- This framework covers many navigation, tracking and SLAM problem formulations! Typically, position is the nonlinear state, while all other ones are (almost) linear where the (extended) KF is used.

Marginalized Particle Filter (2/2)

- Model

$$
\begin{aligned}
x_{k+1}^n &= f_k^n(x_k^n) + F_k^n(x_k^n)x_k^l + G_k^n(x_k^n)w_k^n, \\
x_{k+1}^l &= f_k^l(x_k^n) + F_k^l(x_k^n)x_k^l + G_k^l(x_k^n)w_k^l, \\
y_k &= h_k(x_k^n) + H_k(x_k^n)x_k^l + e_k.
\end{aligned}
$$

  All of w^n, w^l, e_k and x_0^k are Gaussian. x_0^n can be general.
- Basic factorization holds: conditioned on $x_{1:k}^n$, the model is linear and Gaussian.
- This framework covers many navigation, tracking and SLAM problem formulations! Typically, position is the nonlinear state, while all other ones are (almost) linear where the (extended) KF is used.

Marginalized Particle Filter: key factorization

Split the state vector into two parts ('linear' and 'nonlinear')

$$x_k = \begin{pmatrix} x_k^n \\ x_k^l \end{pmatrix}.$$

The key idea in the MPF is the factorization

$$p(x_k^l, x_{1:k}^n | y_{1:k}) = p(x_k^l | x_{1:k}^n, y_{1:k}) p(x_{1:k}^n | y_{1:k}).$$

The KF provides the first factor, and the PF the second one (requires *marginalization* as in implicit step)!

Marginalized Particle Filter: factorization

KF factor provided by the Kalman filter

$$p(x_k^l | x_{1:k}^n, y_{1:k}) = \mathcal{N}(\hat{x}_{k|k}^l, P_{k|k}^l).$$

PF factor given by marginalization procedure

$$\begin{aligned} p(x_{1:k+1}^n | y_{1:k}) &= p(x_{k+1}^n | x_{1:k}^n, y_{1:k}) p(x_{1:k}^n | y_{1:k}) \\ &= p(x_{1:k}^n | y_{1:k}) \int p(x_{k+1}^n | x_k^l, x_{1:k}^n, y_{1:k}) p(x_k^l | x_{1:k}^n, y_{1:k}) \, dx_k^l \\ &= p(x_{1:k}^n | y_{1:k}) \int p(x_{k+1}^n | x_k^l, x_{1:k}^n, y_{1:k}) \mathcal{N}(\hat{x}_{k|k}^l, P_{k|k}^l) \, dx_k^l. \end{aligned}$$

Example: marginalized particle filter

Terrain navigation in 1D. Unknown velocity considered as a state:

$$\begin{aligned} x_{k+1} &= x_k + u_k + \tfrac{T_s^2}{2} v_k \\ u_{k+1} &= u_k + T_s v_k \\ y_k &= h(x_k) + e_k. \end{aligned}$$

Conditional on trajectory $x_{1:k}$, the velocity is given by a linear and Gaussian model:

$$\begin{aligned} u_{k+1} &= u_k + T_s v_k \quad \text{dynamics} \\ x_{k+1} - x_k &= u_k + \tfrac{T_s^2}{2} v_k \quad \text{measurement.} \end{aligned}$$

Given this trajectory, KF time updates linear part:

$$\begin{aligned} x_{k+1} &= x_k + \hat{u}_{k|k} + \tfrac{T_s^2}{2} v_k, \quad \mathrm{cov}(\hat{u}_k) = P_{k|k} \\ y_k &= h(x_k) + e_k. \end{aligned}$$

Marginalized Particle Filter: principal algorithm

1. PF time update using where x_k^l is interpreted as process noise.
2. KF time update using for each particle $x_{1:k}^{n,(i)}$.
3. KF extra measurement update using for each particle $x_{1:k}^{n,(i)}$.
4. PF measurement update and resampling where x_k^l is intepreted as measurement noise.
5. KF measurement update for each particle $x_{1:k}^{n,(i)}$.

If there is no linear term in the measurement equation, the KF measurement update in 5 disappears, and the Ricatti equation for P becomes the same for all sequences $x_{1:k}^n$. That is, only one Kalman gain for all particles!

Marginalized Particle Filter: information flow

- There are five indeces k in the right hand side factorization of the prior.
- Each index is stepped up separately.
- The order is important!

Prior	$p(x_k^l, x_{1:k}^p \mid y_{1:k})$	$= p(x_k^l \mid x_{1:k}^p, y_{1:k}) p(x_{1:k}^p \mid y_{1:k})$
1. PF TU	$p(x_{1:k}^p \mid y_{1:k})$	$\Rightarrow p(x_{1:k+1}^p \mid y_{1:k})$
2. KF TU	$p(x_k^l \mid x_{1:k}^p, y_{1:k})$	$\Rightarrow p(x_{k+1}^l \mid x_{1:k}^p, y_{1:k})$
3. KF dyn MU	$p(x_{k+1}^l \mid x_{1:k}^p, y_{1:k})$	$\Rightarrow p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k})$
4. PF MU	$p(x_{1:k+1}^p \mid y_{1:k})$	$\Rightarrow p(x_{1:k+1}^p \mid y_{1:k+1})$
5. KF obs MU	$p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k})$	$\Rightarrow p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k+1})$
Posterior	$p(x_{k+1}^l, x_{1:k+1}^p \mid y_{1:k+1})$	$= p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k+1}) p(x_{1:k+1}^p \mid y_{1:k+1})$

Marginalized Particle Filter: properties

MPF compared to full PF gives:

- Fewer particles needed.
- Less variance.
- Less risk of divergence.
- Less tuning of importance density and resampling needed.

The price to paid is that the algorithm is more complex.

Variance Formula

The *law of total variance* says that

$$\operatorname{cov}(U) = \operatorname{cov}\big(\mathsf{E}(U|V)\big) + \mathsf{E}\big(\operatorname{cov}(U|V)\big).$$

Example

$$x \sim 0.5\mathcal{N}(-1,1) + 0.5\mathcal{N}(1,1)$$

Let $U = \mathcal{N}(0,1)$ and V the mode ± 1. Then

$$\mathsf{E}(x) = 0,$$
$$\operatorname{cov}(x) = \big(0.5\cdot(1-0)^2 + 0.5\cdot(-1-0)^2\big) + (0.5\cdot 1 + 0.5\cdot 1) = 2$$

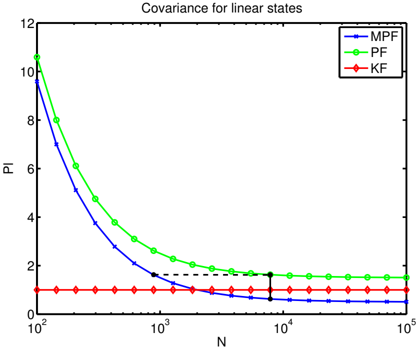
Property: variance reduction

Letting $U = x_k^l$ and $V = x_{1:k}^n$ gives the following decomposition of the variance of the PF:

$$\underbrace{\mathsf{cov}(x_k^l)}_{PF} = \mathsf{cov}\big(\mathsf{E}(x_k^l|x_{1:k}^n)\big) + \mathsf{E}\big(\mathsf{cov}(x_k^l|x_{1:k}^n)\big)$$

$$= \underbrace{\mathsf{cov}\big(\hat{x}_{k|k}^l(x_{1:k}^{n,i})\big)}_{MPF} + \sum_{i=1}^{N} w_k^i \underbrace{P_{k|k}(x_{1:k}^{n,i})}_{KF}.$$

Decreasing variance or N



Schematic view of how the covariance of the linear part of the state vector depends on the number of particles for the PF and MPF, respectively. The gain in MPF is given by the Kalman filter covariance, which is more efficiently estimated without using particles.

Summary

LINKÖPING UNIVERSITY

Filter Summary

- Approximate the model to a case where an optimal algorithm exists:
  - EKF1 approximates the model with a linear one.
  - UKF and EKF2 apply higher order approximations.

  Gives an approximate Gaussian posterior.
- Approximate the optimal nonlinear filter for the original model:
  - Point-mass filter (PMF) which uses a *regular* grid of the state space and applies the Bayesian recursion.
  - Particle filter (PF) which uses a *random* grid of the state space and applies the Bayesian recursion.

  Gives a sample-based numerical approximation of the posterior.

Gustaf Hendeby

hendeby@isy.liu.se

www.liu.se

LINKÖPING UNIVERSITY