SENSOR FUSION

Marginalized Particle Filter (MPF)

Sensor Fusion

Fredrik Gustafsson
fredrik.gustafsson@liu.se

Gustaf Hendeby
gustaf.hendeby@liu.se

Linköping University

Basic Particle Filter (SIR) Algorithm

The particle filter (PF) is an approximate solution to the Bayesian filtering recursion for nonlinear state-space models:

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

Particle Filter Algorithm

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

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

1. *Measurement update:*
$$\bar{w}_{k|k}^{(i)} = w_{k|k-1}^{(i)} p(y_k | x_k^{(i)}).$$
2. *Normalize:* $w_{k|k}^{(i)} := \bar{w}_{k|k}^{(i)} / \sum_j \bar{w}_{k|k}^{(j)}$.
3. *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 $w_{k|k}^{(i)} = 1/N$.
4. *Prediction:* Generate random process noise samples
$$v_k^{(i)} \sim p_{v_k}, \qquad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}) \qquad w_{k+1|k} = w_{k|k}.$$

The number of needed particles, N, scales rather poorly with the state dimension.

Marginalized Particle Filter

Objective

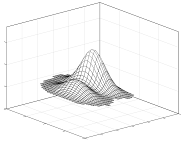
Decrease the number of particles for large state spaces (say $n_x > 3$) by utilizing partial linear Gaussian substructures.

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

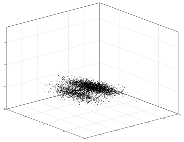
1. Representation of the filtering probability density function.
2. Propagation of this density during the time and measurement update stages.

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

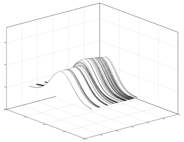
True



Particle



Mixed



Model with Linear Gaussian Substructure

Conditional Linear Gaussian 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.

Model with Linear Gaussian Substructure

Conditional Linear Gaussian Model

$$
\begin{aligned}
x^n_{k+1} &= f^n_k(x^n_k) + F^n_k(x^n_k)x^l_k + G^n_k(x^n_k)w^n_k,\\
x^l_{k+1} &= f^l_k(x^n_k) + F^l_k(x^n_k)x^l_k + G^l_k(x^n_k)w^l_k,\\
y_k &= h_k(x^n_k) + H_k(x^n_k)x^l_k + e_k.
\end{aligned}
$$

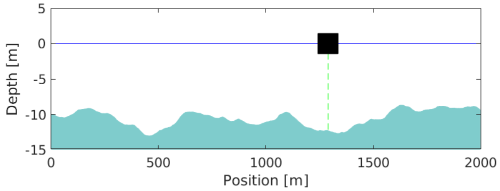
All of w^n, w^l, e_k and x^k_0 are Gaussian. x^n_0 can be general.

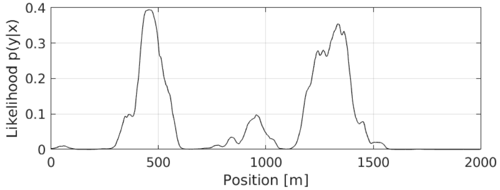
- Basic factorization holds: conditioned on $x^n_{1:k}$, the model is linear and Gaussian.
- This framework covers many navigation, tracking and SLAM problem formulations! Typically, position is the nonlinear state, while all other ones are (almost) linear where the (extended) KF is used.

Example: 1D terrain navigation (1/2)

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

$$x_{k+1} = x_k + u_k + \frac{T_s^2}{2} v_k$$
$$u_{k+1} = u_k + T_s v_k$$
$$y_k = h(x_k) + e_k.$$





Example: 1D terrain navigation (2/2)

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

$$u_{k+1} = u_k + T_s v_k \quad \text{dynamics}$$

$$x_{k+1} - x_k = u_k + \frac{T_s^2}{2} v_k \quad \text{measurement.}$$

  Given the trajectory, and the Kalman filter applies to this part.
- The reminder of the model,

$$x_{k+1} = x_k + \hat{u}_{k|k} + \frac{T_s^2}{2} v_k, \quad \operatorname{Cov}(\hat{u}_k) = P_{k|k}$$

$$y_k = h(x_k) + e_k,$$

  is nonlinear due to the measurement in the map. This can be handled using a particle filter to generate different trajectories for which to run the Kalman filters.
- The Kalman filters and the particle filter interact to obtain the full filter solution.

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 an implicit step)!
A method to derive the marginalized particle filter will now be outlined, in which the MPF can be interpreted as an stochastic filter bank.
Algorithm details can be found in the textbook.

Conditional Gaussian Linear Part: Kalman filter

Assume the trajectory of the nonlinear part, $x^n_{1:k}$ given, then $p(x^l_k | x^n_{1:k}, y_{1:k})$ is the solution to the estimation problem give by

$$
\begin{aligned}
x^l_{k+1} &= F^l_k(x^n_k)x^l_k + f^l_k(x^n_k) + G^l_k(x^n_k)w^l_k, && \text{(KF-TU)}\\
y_k &= H_k(x^n_k)x^l_k + h_k(x^n_k) + e_k && \text{(KF-MU1)}\\
x^n_{k+1} - f^n_k(x^n_k) &= F^n_k(x^n_k)x^l_k + G^n_k(x^n_k)w^n_k. && \text{(KF-MU2)}
\end{aligned}
$$

- The solution to (KF-TU) and (KF-MU1) is a Kalman filter.
- (KF-MU2) adds no value until conditioning on x^n_{k+1}, when it becomes a measurement equation where the statistics depends on how x^n_{k+1} was obtained.
- A separate Kalman filter is needed for each $x^n_{1:k}$ sequence, but sometimes computations can be reused.

Remaining Part: particle filter

The remaining factor $p(x^n_{1:k}|y_{1:k})$ matches what have not been resolved used in

$$x^n_{k+1} = f^n_k(x^n_k) + F^n_k(x^n_k)x^l_k + G^n_k(x^n_k)w^n_k \qquad \text{(PF-TU)}$$

$$x^l_{k+1} = f^l_k(x^n_k) + F^l_k(x^n_k)x^l_k + G^l_k(x^n_k)w^l_k,$$

$$y_k = h_k(x^n_k) + H_k(x^n_k)x^l_k + e_k. \qquad \text{(PF-MU)}$$

The estimation problem can be solved using a particle filter.

- The time update utilizes (PF-TU) and the distribution of x^l_k from the KF to generate new particles.
- The measurement update use (PF-MU) and the distribution of x^l_k to update the weights of the particles.

This can be interpreted as stochastic trajectory generation and pruning in a stochastic filter bank. Compare this to the generation and reduction of discrete and enumerable mode sequences in normal filter banks.

Marginalized Particle Filter Properties

Benefits of the marginalized particle filter compared to regular particle filters:

- Requires fewer particles.
- Improves the variance, given the same number of particles.
- Has lower risk of divergence.
- Requires less tuning of importance density and resampling.

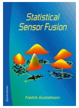
The price to paid is that the algorithm is more complex.

Summary

The marginalized particle filter (MPF) apply to problems with linear Gaussian substructure.

- Combines Kalman filters and a particle filter.
- Requires fewer particles than a regular PF.
- Higher accuracy than PF with the same number of particles.
- Improves robustness compared to PF.
- Is more complex to implement.

The MPF can be viewed as a filter bank with stochastic branching and pruning.



Section 9.8