Particle flow for particle filtering

Yunpeng Li [1] Lingling Zhao [2] Mark Coates [1]

[1]Dept. of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

March 22, 2016

Motivation

- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.

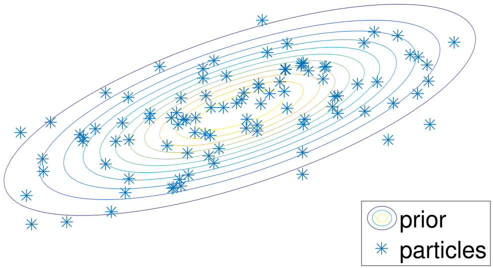
Motivation

- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.
- Particle flow migrates, instead of samples, particles to combat particle degeneracy [1]

[1] F. Daum and J. Huang, "Nonlinear filters with log-homotopy, in *Proc. SPIE Signal and Data Processing of Small Targets*, Sep. 2007.

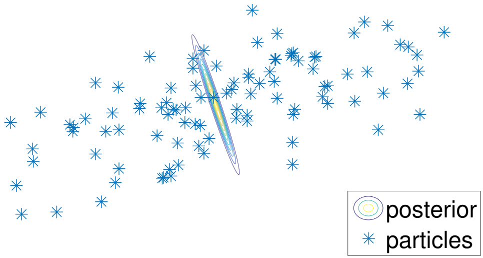
Motivation

- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.
- Particle flow migrates, instead of samples, particles to combat particle degeneracy [1]



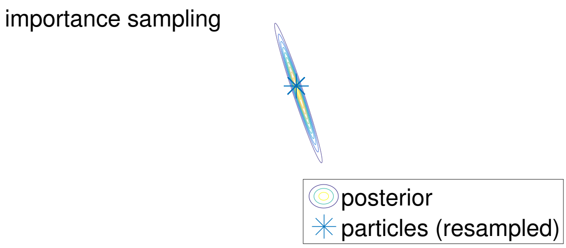
Motivation

- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.
- Particle flow migrates, instead of samples, particles to combat particle degeneracy [1]



Motivation

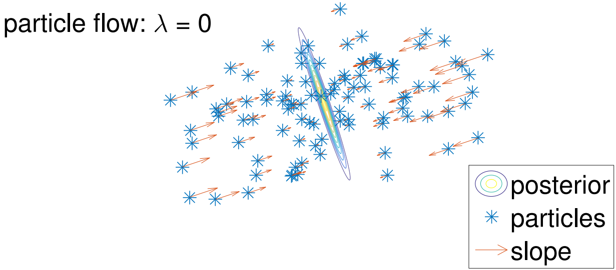
- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.
- Particle flow migrates, instead of samples, particles to combat particle degeneracy [1]



[1] F. Daum and J. Huang, "Nonlinear filters with log-homotopy, in *Proc. SPIE Signal and Data Processing of Small Targets*, Sep. 2007.

Motivation

- Particle filter suffers weight degeneracy in high dimensional state space or when observations are highly informative.
- Particle flow migrates, instead of samples, particles to combat particle degeneracy [1]



[1] F. Daum and J. Huang, "Nonlinear filters with log-homotopy, in *Proc. SPIE Signal and Data Processing of Small Targets*, Sep. 2007.

Motivation

- Most solutions to particle flow are computationally intractable; we need approximations in implementation.

Motivation

- Most solutions to particle flow are computationally intractable; we need approximations in implementation.
- Alternative: utilize particle flow to generate a proposal distribution [2] [3]; often computationally expensive.

[2] P. Bunch and S. Godsill, "Approximations of the optimal importance density using Gaussian particle flow importance sampling," *J. Amer. Statist. Assoc.*, 2015.

[3] S. Reich, "A guided sequential Monte Carlo method for the assimilation of data into stochastic dynamical systems," in *Recent Trends in Dynamical Systems*, 2013.

Motivation

- Most solutions to particle flow are computationally intractable; we need approximations in implementation.
- Alternative: utilize particle flow to generate a proposal distribution [2] [3]; often computationally expensive.
- Proposed method: construct particle flows with the invertible mapping property to efficiently evaluate importance weights.

[2] P. Bunch and S. Godsill, "Approximations of the optimal importance density using Gaussian particle flow importance sampling," *J. Amer. Statist. Assoc.*, 2015.

[3] S. Reich, "A guided sequential Monte Carlo method for the assimilation of data into stochastic dynamical systems," in *Recent Trends in Dynamical Systems*, 2013.

Problem statement

A nonlinear filtering task with the following model:

$$x_k = g(x_{k-1}, v_k)$$
$$z_k = h(x_k, w_k)$$

x_k the unobserved state at time step k

z_k the observation

v_k, w_k noise terms

Problem statement

A nonlinear filtering task with the following model:

$$x_k = g(x_{k-1}, v_k)$$
$$z_k = h(x_k, w_k)$$

x_k the unobserved state at time step k

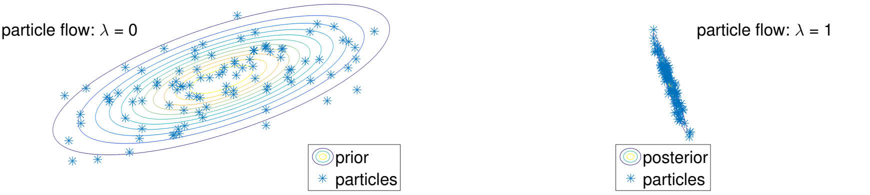
z_k the observation

v_k, w_k noise terms

Task: track the marginal posterior distribution $p(x_k | z_1, \dots, z_k)$.

Particle flow

- The particle flow process is modeled as a stochastic process η_λ for $\lambda \in [0, 1]$.
- The distribution of η_0 is the prior distribution of x_k and the distribution of η_1 is the posterior distribution of x_k.



Zero-diffusion particle flow

The trajectory of η_λ^i in realization i follows $\frac{d\eta_\lambda^i}{d\lambda} = \zeta(\eta_\lambda^i, \lambda)$, governed by the Fokker-Planck equation with zero diffusion:

$$\frac{\partial p(\eta_\lambda^i, \lambda)}{\partial \lambda} = -\mathrm{div}(p(\eta_\lambda^i, \lambda)\zeta(\eta_\lambda^i, \lambda))$$

Zero-diffusion particle flow

The trajectory of η_λ^i in realization i follows $\dfrac{d\eta_\lambda^i}{d\lambda} = \zeta(\eta_\lambda^i, \lambda)$, governed by the Fokker-Planck equation with zero diffusion:

$$\frac{\partial p(\eta_\lambda^i, \lambda)}{\partial \lambda} = -\text{div}(p(\eta_\lambda^i, \lambda)\zeta(\eta_\lambda^i, \lambda))$$

- Different flow assumptions:
  incompressible flow, exact flow with zero diffusion, etc.

Zero-diffusion particle flow

The trajectory of η^i_λ in realization i follows $\dfrac{d\eta^i_\lambda}{d\lambda} = \zeta(\eta^i_\lambda, \lambda)$, governed by the Fokker-Planck equation with zero diffusion:

$$\frac{\partial p(\eta^i_\lambda, \lambda)}{\partial \lambda} = -\mathrm{div}(p(\eta^i_\lambda, \lambda)\zeta(\eta^i_\lambda, \lambda))$$

- Different flow assumptions:
  incompressible flow, exact flow with zero diffusion, etc.
- Prior and posterior are Gaussian: exact Daum-Huang (EDH) filter[4], Localized exact DH (LEDH) filter [5].

$$\zeta(\eta^i_\lambda, \lambda) = A^i(\lambda)\eta^i_\lambda + b^i(\lambda) \qquad \text{(for LEDH)}$$

[4] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Orlando, FL, USA, Apr. 2010.

[5] T. Ding and M. J. Coates, "Implementation of the Daum-Huang exact-flow particle filter," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, Ann Arbor, MI, USA, Aug. 2012.

Importance weight calculation

$$w_k^i \propto \frac{p(\eta_1^i | x_{k-1}^i) p(z_k | \eta_1^i)}{p(\eta_1^i | x_{k-1}^i ; z_k)} w_{k-1}^i$$

x_{k-1}^i the i-th particle at time step $k-1$

z_k the observation at time step k

η_1^i the i-th particle after the flow

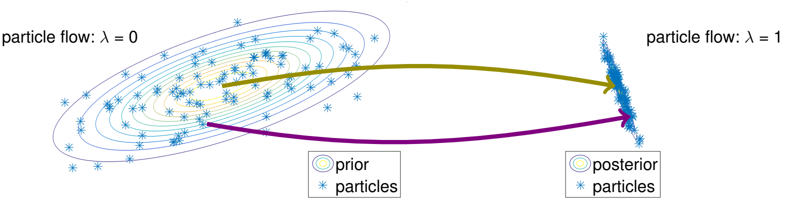
Importance weight calculation

$$w_k^i \propto \frac{p(\eta_1^i | x_{k-1}^i) p(z_k | \eta_1^i)}{p(\eta_1^i | x_{k-1}^i ; z_k)} w_{k-1}^i$$

- x_{k-1}^i the i-th particle at time step $k-1$
- z_k the observation at time step k
- η_1^i the i-th particle after the flow
- η_0^i the i-th particle before the flow

invertible mapping

$$\eta_1^i \neq \eta_1^j \iff \eta_0^i \neq \eta_0^j$$



Importance weight calculation

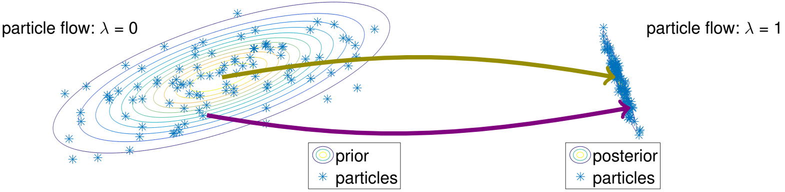
$$w_k^i \propto \frac{p(\eta_1^i|x_{k-1}^i)p(z_k|\eta_1^i)}{p(\eta_1^i|x_{k-1}^i; z_k)} w_{k-1}^i$$

x_{k-1}^i	the i-th particle at time step $k-1$
z_k	the observation at time step k
η_1^i	the i-th particle after the flow
η_0^i	the i-th particle before the flow

invertible mapping

$$\eta_1^i \neq \eta_1^j \iff \eta_0^i \neq \eta_0^j$$

$$\begin{aligned} p(\eta_1^i|x_{k-1}^i; z_k) &= p(\eta_0^i|x_{k-1}^i; z_k) \\ &= p(\eta_0^i|x_{k-1}^i) \end{aligned}$$



Particle flow with invertible mapping

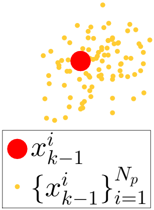
Algorithm 1 Particle flow particle filter (LEDH)

```
Initialization.
for k = 1 to TotalTimeStep do
    for i = 1 to NumParticle do
        calculate η̄_0^i = g(x_{k−1}^i, 0)
        sample η_0^i = g(x_{k−1}^i, v_k)
        for λ = [0, λ_1, λ_2, ..., 1) do
            propagate η̄_λ^i to obtain A^i(λ) and b^i(λ)
            propagate η_λ^i using A^i(λ) and b^i(λ)
        end for
        w_k^i = [p(η_1^i | x_{k−1}^i) p(z_k | η_1^i) / p(η_0^i | x_{k−1}^i)] w_{k−1}^i
    end for
end for
```

$$\bar{\eta}_0^i = g(x_{k-1}^i, 0)$$

$$\eta_0^i = g(x_{k-1}^i, v_k)$$

$$w_k^i = \frac{p(\eta_1^i | x_{k-1}^i)\, p(z_k | \eta_1^i)}{p(\eta_0^i | x_{k-1}^i)} w_{k-1}^i$$



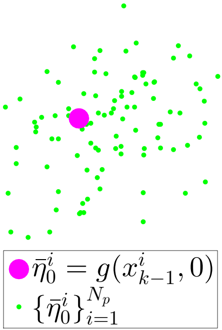
Particle flow with invertible mapping

Algorithm 2 Particle flow particle filter (LEDH)

```
Initialization.
for k = 1 to TotalTimeStep do
    for i = 1 to NumParticle do
        calculate η̄_0^i = g(x_{k−1}^i, 0)
        sample η_0^i = g(x_{k−1}^i, v_k)
        for λ = [0, λ_1, λ_2, ..., 1) do
            propagate η̄_λ^i to obtain A^i(λ) and b^i(λ)
            propagate η_λ^i using A^i(λ) and b^i(λ)
        end for
        w_k^i = [p(η_1^i | x_{k−1}^i) p(z_k | η_1^i) / p(η_0^i | x_{k−1}^i)] w_{k−1}^i
    end for
end for
```

Where step 4 is: calculate $\bar{\eta}_0^i = g(x_{k-1}^i, 0)$; step 5: sample $\eta_0^i = g(x_{k-1}^i, v_k)$; step 6: **for** $\lambda = [0, \lambda_1, \lambda_2, \ldots, 1)$ **do**; step 7: propagate $\bar{\eta}_\lambda^i$ to obtain $A^i(\lambda)$ and $b^i(\lambda)$; step 8: propagate η_λ^i using $A^i(\lambda)$ and $b^i(\lambda)$; step 10:

$$w_k^i = \frac{p(\eta_1^i | x_{k-1}^i) p(z_k | \eta_1^i)}{p(\eta_0^i | x_{k-1}^i)} w_{k-1}^i$$



Particle flow with invertible mapping

Algorithm 3 Particle flow particle filter (LEDH)

```
Initialization.
for k = 1 to TotalTimeStep do
    for i = 1 to NumParticle do
        calculate η̄_0^i = g(x_{k-1}^i, 0)
        sample η_0^i = g(x_{k-1}^i, v_k)
        for λ = [0, λ_1, λ_2, ..., 1) do
            propagate η̄_λ^i to obtain A^i(λ) and b^i(λ)
            propagate η_λ^i using A^i(λ) and b^i(λ)
        end for
        w_k^i = p(η_1^i | x_{k-1}^i) p(z_k | η_1^i) / p(η_0^i | x_{k-1}^i) · w_{k-1}^i
    end for
end for
```

Step 4: calculate $\bar{\eta}_0^i = g(x_{k-1}^i, 0)$

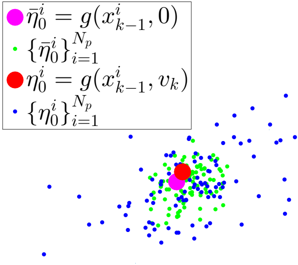
Step 5: sample $\eta_0^i = g(x_{k-1}^i, v_k)$

Step 6: for $\lambda = [0, \lambda_1, \lambda_2, \ldots, 1)$ do

Step 7: propagate $\bar{\eta}_\lambda^i$ to obtain $A^i(\lambda)$ and $b^i(\lambda)$

Step 8: propagate η_λ^i using $A^i(\lambda)$ and $b^i(\lambda)$

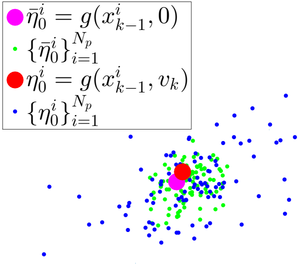
Step 10:
$$w_k^i = \frac{p(\eta_1^i|x_{k-1}^i)p(z_k|\eta_1^i)}{p(\eta_0^i|x_{k-1}^i)} w_{k-1}^i$$



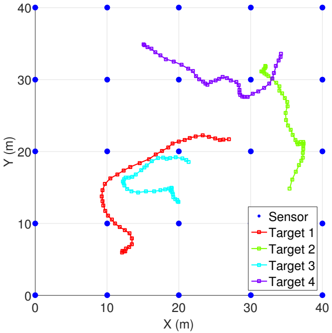
Particle flow with invertible mapping

Algorithm 4 Particle flow particle filter (LEDH)

Initialization.
for k = 1 to TotalTimeStep **do**
  for i = 1 to NumParticle **do**
    calculate $\bar{\eta}_0^i = g(x_{k-1}^i, 0)$
    sample $\eta_0^i = g(x_{k-1}^i, v_k)$
    for $\lambda = [0, \lambda_1, \lambda_2, \ldots, 1)$ **do**
      propagate $\bar{\eta}_\lambda^i$ to obtain $A^i(\lambda)$ and $b^i(\lambda)$
      propagate η_λ^i using $A^i(\lambda)$ and $b^i(\lambda)$
    end for
    $$w_k^i = \frac{p(\eta_1^i | x_{k-1}^i) p(z_k | \eta_1^i)}{p(\eta_0^i | x_{k-1}^i)} w_{k-1}^i$$
  end for
end for



Experiment setup

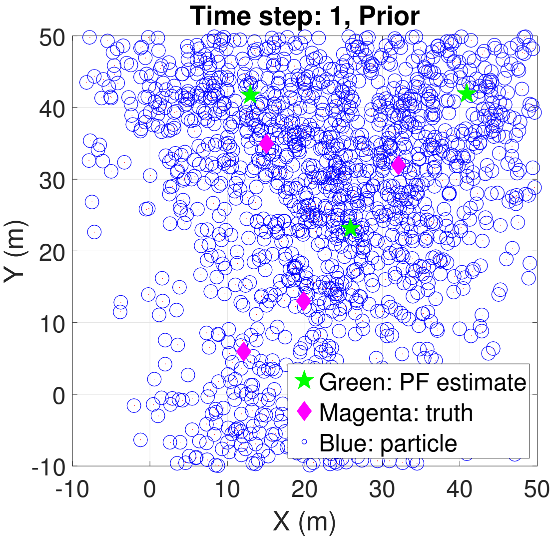


- 4 targets, constant velocity model, 16×1 state vector.
- 25 acoustic amplitude sensors in a 40m x 40m grid.
- Additive measurement model:

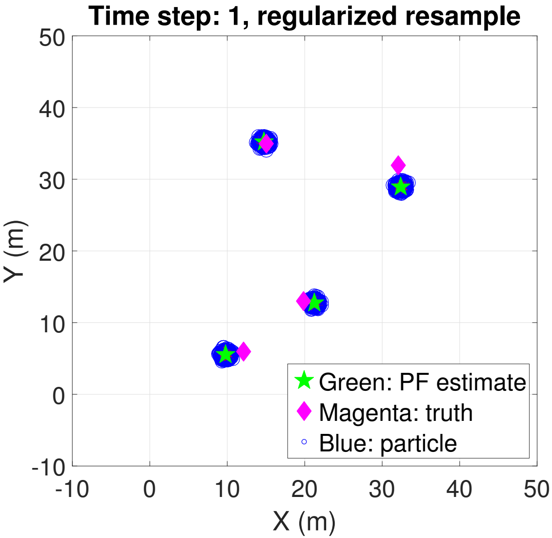
$$\bar{z}^s(x_k) = \sum_{m=1}^{NumTarget} \frac{A}{||(\mathrm{x}_k^{(m)}, \mathrm{y}_k^{(m)})^T - \zeta^s||^\kappa + d_0}$$

- Small measurement noise.

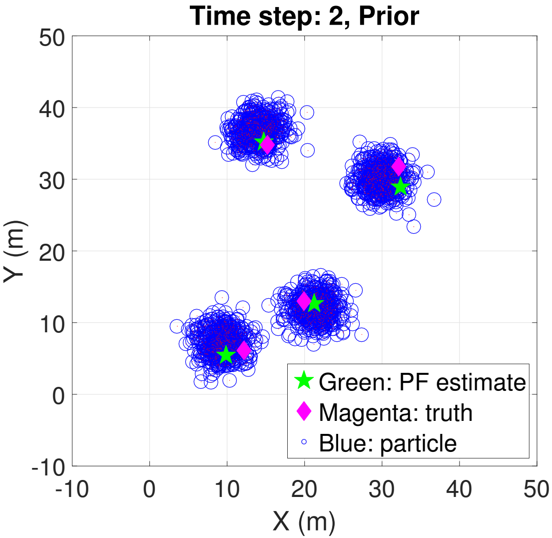
Tracking demo



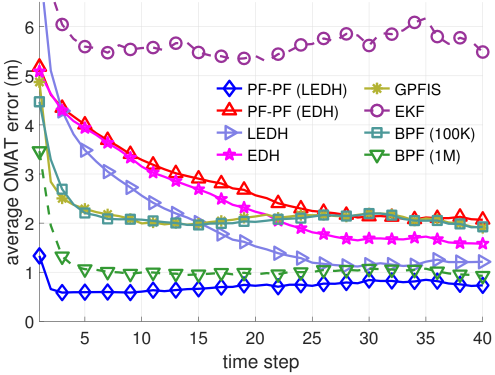
Tracking demo



Tracking demo



Performance evaluation



PF-PF	particle flow particle filtering
LEDH	localized exact Daum and Huang filter
EDH	exact Daum and Huang filter
GPFIS	Gaussiam particle flow importance sampling
EKF	extended Kalman filter
BPF	bootstrap particle filter

500 particles used in all algorithms except BPF.

optimal mass transfer (OMAT) metric

$$d_p(X, Y) = (\frac{1}{M} \min_{\pi \in \Pi} \sum_{i=1}^{M} d(x_i, y_{\pi(i)})^p)^{1/p}$$

M	number of targets
$d()$	Euclidean distance
Π	the set of possible permutations of 1,2,...,M

Performance evaluation

- Average error, effective sample size, and execution time per step.
- Results are produced with an Intel i7 4770K 3.50GHz CPU.
- 500 particles used in all algorithms except BPF.

Algorithm	Avg. OMAT	Avg. ESS	Avg. Execution time
PF-PF (LEDH)	0.72	29.3	1.45
PF-PF (EDH)	2.81	29.8	0.01
LEDH	2.05	N/A	1.44
EDH	2.52	N/A	0.01
GPFIS	2.20	14.2	124
EKF	5.73	N/A	0.002
BPF (10^5 particles)	2.18	2.13	0.52
BPF (10^6 particles)	1.10	7.43	5.29

PF-PF particle flow particle filtering
LEDH localized exact Daum and Huang filter
EDH exact Daum and Huang filter
GPFIS Gaussiam particle flow importance sampling
EKF extended Kalman filter
BPF bootstrap particle filter

Conclusion

- Proposed a particle filtering algorithm that uses particle flow to construct the proposal distribution.
- Proved that the applied particle flows are invertible mappings, so we can evaluate importance weights in a simple fashion.
- Proposed algorithm retains the statistical consistency of particle filter, and acquires desirable properties of particle flow.

Thank you!