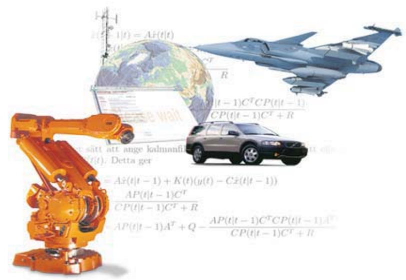
Bonus Lecture – Nonlinear State Estimation



Thomas Schön,
Division of Automatic Control,
Department of Electrical Engineering,
Linköping University.

Email: schon@isy.liu.se

Nonlinear state estimation is all about finding approximations of probability density functions

Content – Bonus Lecture

1. Problem formulation
2. Conceptual solution
3. The Kalman filter and the extended Kalman filter (EKF)
4. The key idea underlying the particle filter
5. Random number generation
    a. Perfect sampling
    b. Importance sampling
    c. Importance sampling resampling
6. A first particle filter
7. General particle filters
8. Application Examples

The Setting

We are dealing with dynamic systems!

Mathematics

State variable

$$x_{t+1} = f_t(x_t) + w_t,$$
$$y_t = h_t(x_t) + e_t.$$

Measurement

Stochastic noise

"The output of a dynamical system depends on all the previous inputs"

Application examples

Dynamical Systems – Modelling

Nonlinear state-space model (stochastic difference equation)

State variable

Input signal

Process model $x_{t+1} = f_t(x_t, u_t) + v_t$ ← Stochastic process noise

Measurement model $y_t = h_t(x_t, u_t) + e_t$ ← Stochastic measurement noise

Measurement

Alternative formulation
$$x_{t+1} \sim p(x_{t+1}|x_t)$$
$$y_t \sim p(y_t|x_t)$$

The two model descriptions are related,
$$p(x_{t+1}|x_t) = p_{v_t}(x_{t+1} - f_t(x_t))$$
$$p(y_t|x_t) = p_{e_t}(y_t - h_t(x_t))$$

Dynamical Systems – Estimation

The aim is to obtain information about the state x_t using the information present in the measurements $y_{1:s} = \{y_i\}_{i=1}^{s}$

$\Downarrow$

Compute $p(x_t|y_{1:s})$

Three different estimation problems:

- The filtering problem, $t = s$ Filtering density $p(x_t|y_{1:t})$
- The prediction problem, $t > s$
- The smoothing problem, $t < s$

Conceptual Solution

We have now shown that for the dynamic model

$$x_{t+1} \sim p(x_{t+1}|x_t)$$
$$y_t \sim p(y_t|x_t)$$

the filtering and the one-step ahead prediction densities are

Measurement update
$$p(x_t|y_{1:t}) = \frac{\overbrace{p(y_t|x_t)}^{\text{Likelihood}}\overbrace{p(x_t|y_{1:t-1})}^{\text{Previous prediction density}}}{p(y_t|y_{1:t-1})}$$

Time update
$$p(x_{t+1}|y_{1:t}) = \int \underbrace{p(x_{t+1}|x_t)}_{\text{Dynamics}}\underbrace{p(x_t|y_{1:t})}_{\text{Previous filtering density}} dx_t$$

State Estimation

The **aim** is to obtain information about the state x_t using the information present in the measurements $y_{1:t} = \{y_i\}_{i=1}^{t}$

$$x_{t+1} = f_t(x_t) + w_t, \qquad y_t = h_t(x_t) + e_t.$$

$$x_{t+1} \sim p(x_{t+1}|x_t), \qquad y_t \sim p(y_t|x_t).$$

f, h linear equations
w_t, e_t Gaussian noise
$\Rightarrow$ The **Kalman filter** is the best solution

General case $\Rightarrow$ **Particle filter**

Point mass filters

Extended Kalman filter

Sigma point Kalman filter

State Estimation – The Kalman Filter

Important special case, linear equations with Gaussian noise

$$\begin{aligned} x_{t+1} &= Ax_t + v_t, & v_t &\sim \mathcal{N}(0, Q), \\ y_t &= Cx_t + e_t, & e_t &\sim \mathcal{N}(0, R) \end{aligned}$$

This allows for a closed form solution

$$p(x_t|y_{1:t}) = \mathcal{N}(x_t; \hat{x}_{t|t}, P_{t|t}),$$
$$p(x_{t+1}|y_{1:t}) = \mathcal{N}(x_{t+1}; \hat{x}_{t+1|t}, P_{t+1|t}).$$

Gaussian variables and linear transformations implies that complete information is provided by the mean and the covariance.

State Estimation – The Kalman filter

Measurement update

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \overbrace{(y_t - C\hat{x}_{t|t-1})}^{\text{innovation}},$$
$$S_t = CP_{t|t-1}C^T + R_t,$$
$$K_t = P_{t|t-1}C^T S_t^{-1},$$
$$P_{t|t} = P_{t|t-1} \underbrace{- P_{t|t-1}C^T S_t^{-1} C P_{t|t-1}}_{\text{Decrease uncertainty}},$$

Time update

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t},$$
$$P_{t+1|t} = AP_{t|t}A^T \underbrace{+Q}_{\text{Increase uncertainty}}.$$

The Key Idea Underlying the Particle Filter

The particle filter computes estimates of the filtering PDF

The key idea is to form a nonparametric estimate,

$$p^N(x_t|y_{1:t}) = \sum_{i=1}^{N} \underbrace{\boxed{\tilde{q}_t^i}}_{\text{weights}} \delta(x_t - \underbrace{\boxed{x_t^i}}_{\text{particles}}), \quad \sum_{i=1}^{N} \tilde{q}_t^i = 1, \quad \tilde{q}_t^i \geq 0, \forall i$$

The estimate converge as $N \to \infty$

This implies that the multidimensional integrals are reduced to finite sums

$$\boxed{\text{“}\ \delta + \int \to \sum\ \text{”}}$$

Random Number Generation – Perfect Sampling

Assumption: We have access to samples from the density we seek to estimate,

$$\hat{t}_N(x) = \sum_{i=1}^{N} \frac{1}{N}\delta\left(x - x^i\right)$$

Recall that we are seeking an estimate of according to

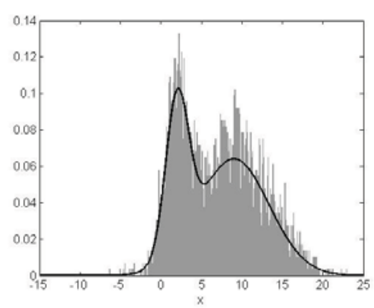
$$\hat{I}_N\left(g(x)\right) = \int g(x)\hat{t}_N(x)dx = \sum_{i=1}^{N} \frac{1}{N} g(x^i).$$

The strong law of large numbers $\lim_{N\to\infty} \hat{I}_N(g(x)) \xrightarrow{\text{a.s.}} I(g(x))$

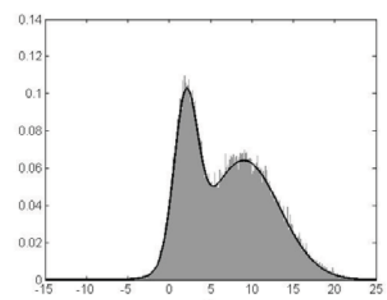
Central limit theorem $\lim_{N\to\infty} \frac{\sqrt{N}}{\sigma}\left(\hat{I}_N(g(x)) - I(g(x))\right) \xrightarrow{d} \mathcal{N}(0,1)$

Perfect Sampling – Example and Problem

$$t(x) = 0.3\mathcal{N}\left(x\,|\,2,2\right) + 0.7\mathcal{N}\left(x\,|\,9,19\right)$$



5000 samples



50000 samples

Obvious problem:

In general we are **NOT** able to sample from the density we are interested in!

Random Number Generation

$$\text{Target density} \longrightarrow t(\tilde{x}) \propto \underbrace{w(\tilde{x})}_{\text{Acceptance prob.}} q(\tilde{x}) \longleftarrow \text{Proposal density}$$

Target density – We seek samples distributed according to this density

Proposal density – This density is simple to generate samples from

Acceptance probability – Used to decide whether the sample is OK

Three common algorithms based on this idea:

1. Rejection sampling
2. **Importance sampling**
3. Metropolis-Hastings

Bonus Lecture

Random Number Generation – Importance Sampling

Algorithm 3.1 (Importance sampling)

1. *Generate N i.i.d. samples $\{\tilde{x}^i\}_{i=1}^N$ from the proposal density $q(x)$ and compute the importance weights*

$$\tilde{w}^i = t(\tilde{x}^i)/q(\tilde{x}^i), \qquad i = 1, \dots, N. \tag{3.22}$$

2. *Form the acceptance probabilities by normalization,*

$$w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j, \qquad i = 1, \dots, N. \tag{3.23}$$

This algorithm provides the following approximation of the target density

$$\tilde{t}_N(x) = \sum_{i=1}^N w^i \delta\left(x - \tilde{x}^i\right)$$

Central limit theorem and a law of large numbers can be formulated

Bonus Lecture

Sampling Importance Resampling

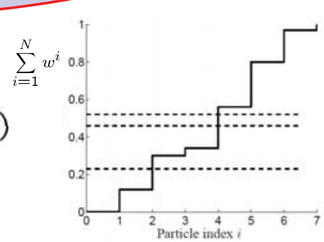
$$\tilde{t}_N(x) \approx \sum_{i=1}^N w^i \delta(x - \tilde{x}^i)$$

Resampling

Generates an unweighted set of samples by drawing according to

$$\boxed{\Pr\left(x^i = \tilde{x}^j\right) = w^j, \qquad j = 1, \dots, N.}$$

$$\hat{t}_N(x) = \sum_{i=1}^N \frac{1}{N} \delta(x - x^i)$$



Bonus Lecture

Sampling Importance Resampling – Algorithm

This part is exactly the same as the importance sampling algorithm

Algorithm 3.3 (Sampling Importance Resampling (SIR))

1. *Generate N i.i.d. samples $\{\tilde{x}^i\}_{i=1}^N$ from the proposal density $q(x)$ and compute the importance weights*

$$\tilde{w}^i = t(\tilde{x}^i)/q(\tilde{x}^i), \qquad i = 1, \dots, N. \tag{3.28}$$

2. *Compute the acceptance probabilities by normalization*

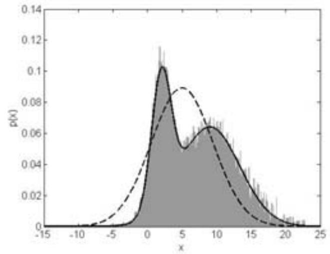
$$w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j, \qquad i = 1, \dots, N. \tag{3.29}$$

3. *For each $i = 1, \dots, N$ draw a new particle x_t^i with replacement (resample) according to,*

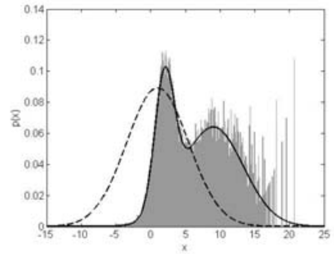
$$P\left(x^i = \tilde{x}^j\right) = w^j, \qquad j = 1, \dots, N. \tag{3.30}$$

Bonus Lecture

The Importance of a Good Importance Density



$q_1(x) = \mathcal{N}(5, 20)$



$q_2(x) = \mathcal{N}(1, 20)$

(50 000 samples were used in both experiments)

Lesson learned:

It is very important to be careful in selecting the importance density.

A First Particle Filter

Algorithm 3.3 (A first particle filter)

1. *Initialize the particles,* $\{x_0^i\}_{i=1}^N \sim p(x_0)$.
2. *Predict the particles by drawing* N *i.i.d. samples according to*

$$\tilde{x}_t^i \sim p(x_t | x_{t-1}^i), \qquad i = 1, \dots, N. \tag{3.45}$$

3. *Compute the importance weights* $\{\tilde{w}_t^i\}_{i=1}^N$,

$$\tilde{w}_t^i = p(y_t | \tilde{x}_t^i), \qquad i = 1, \dots, N. \tag{3.46}$$

and normalize $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.

4. *For each* $i = 1, \dots, N$ *draw a new particle* x_t^i *with replacement (resample) according to,*

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \qquad j = 1, \dots, N. \tag{3.47}$$

5. *Set* $t := t + 1$ *and repeat from step 2.*

State estimate:

$$\hat{x}_t = \sum_{i=1}^N w_t^i \tilde{x}_t^i$$

The Particle Filter in Terms of the Kalman Filter

The Kalman filter

1. Initialize

$$x_{0|0} = \bar{x},$$
$$P_{0|0} = P_0$$

2. Time update

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1},$$
$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q.$$

3. Measurement update

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - C\hat{x}_{t|t-1}),$$
$$P_{t|t} = P_{t|t-1} - K_t C P_{t|t-1}.$$

$$\hat{p}(x_t|y_{1:t}) = \mathcal{N}(x_t;\, \hat{x}_{t|t},\, P_{t|t})$$

The particle filter

1. Initialize:

$$x_0^i \sim p(x_0)$$

2. Time update

$$\tilde{x}_t^i = f(x_{t-1}^i) + v_{t-1}^i$$

3. Measurement update

$$q_t^i = p(y_t|\tilde{x}_t^i) / \sum_{j=1}^N p(y_t|\tilde{x}_t^j),$$
$$\mathrm{P}(x_t^i = \tilde{x}_t^j) = w_t^j.$$

$$\hat{p}(x_t|y_{1:t}) = \sum_{i=1}^N \frac{1}{N}\delta(x_t - x_t^i)$$

The Particle Filter – Simple Example

Let us study one-dimensional motion and build a simple positioning system.

Continuous-time model:

$$\begin{pmatrix} \dot{p} \\ \dot{v} \\ \dot{a} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ v \\ a \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w, \qquad w \sim \mathcal{N}(0, Q_t)$$

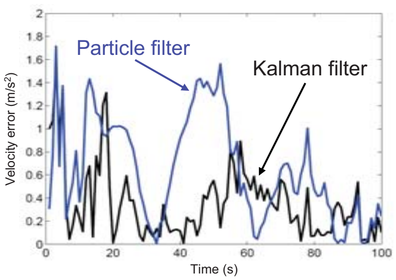
$$y = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ v \\ a \end{pmatrix} + e, \qquad e_t \sim \mathcal{N}(0, R_t)$$

Discrete-time model:

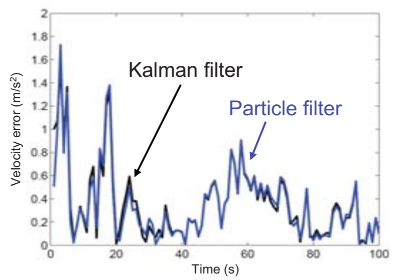
$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \\ a_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + \begin{pmatrix} T^3/6 \\ T^2/2 \\ T \end{pmatrix} w_t, \qquad w_t \sim \mathcal{N}(0, Q_t)$$

$$y_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R_t)$$

The Particle Filter – Simple Example



Using 100 particles.



Using 10 000 particles.

Clearly the result improves as we increase the number of particles.

Motivating Toy Example – Revisited in MATLAB


```
x=repmat(x0,1,N)+sqrtm(P0)*randn(nx,N);

for t=1:t_final

  x=A*x + Bw*sqrtm(Q)*randn(1,N);

  e=repmat(y(:,t),1,N) - C*x;
  w=exp(-0.5*sum(e.*(inv(R)*e)));
  w=w/sum(w);
  index=sysresampling(w);
  x=x(:,index);

  xHat(:,t)=mean(x,2);

end;
```

1. Initialize: $x_0 \sim p(x_0)$

2. Time update: $\tilde{x}_t^i = f_{t-1}(x_{t-1}^i) + v_{t-1}^i$

3. Measurement update: $\tilde{w}_t^i = p(y_t|\tilde{x}_t^i)$ $\Pr(x_t^i = \tilde{x}_t^j) = w_t^j$

Obtaining the estimate

A 5 minute MATLAB implementation is available in the course material

A General Particle Filter

Algorithm 3.4 (Particle filter)

1. *Initialize the particles,* $x_0^i \sim p(x_0), i = 1, \dots, N$ *and the weights,* $w_0^i = 1/N, i = 1, \dots, N$ *and let* $t := 1$.
2. *Generate* N *i.i.d. new particles by drawing from the importance density*

$$\tilde{x}_t^i \sim q(x_t|x_{t-1}^i, y_t), \qquad i = 1, \dots, N. \tag{3.62}$$

and update the weights accordingly

$$\tilde{w}_t^i = \frac{p(\tilde{x}_t^i|x_{t-1}^i)}{q(\tilde{x}^i|x_{t-1}^i, y_t)} w_{t-1}^i, \qquad i = 1, \dots, N. \tag{3.63}$$

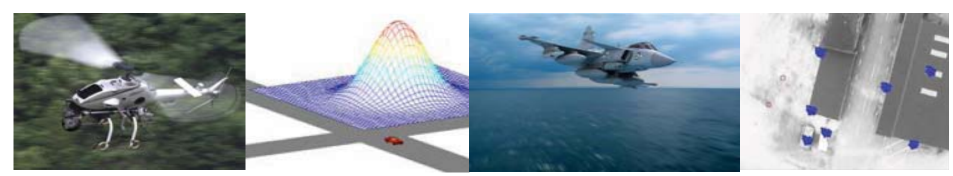
3. *Update the weights according to,*

$$w_t^i = \frac{p(y_t|x_t^i)\tilde{w}_t^i}{\sum_{j=1}^N p(y_t|x_t^j)\tilde{w}_t^j}, \qquad i = 1, \dots, N. \tag{3.64}$$

4. *Resample according to Algorithm 3.5 or Algorithm 3.7.*
5. *Set* $t := t + 1$ *and iterate from step 2.*

$q(x_t|x_{t-1}^i, y_t) = p(x_t|x_{t-1}^i)$ ➡ The algorithm is reduced to the “first particle filter”

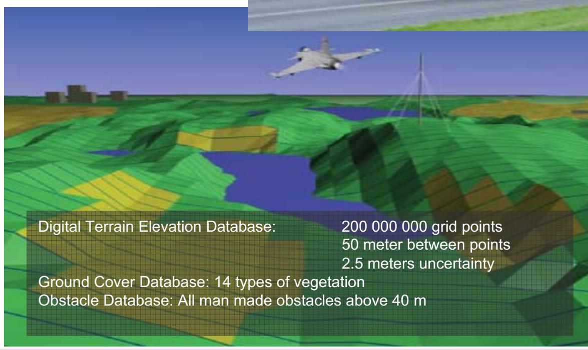
Two Applications



Application Example – Fighter Aircraft Navigation

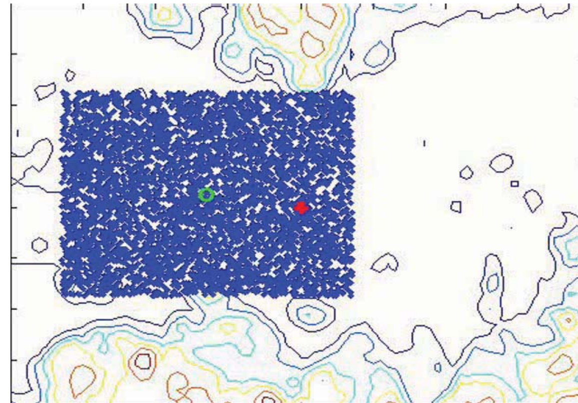
With P-J Nordlund (SAAB Aerosystems)

- Dead-reckoning inertial measurements (gyros and accelerometers)
- TAP aiding sensor to prevent long-term drift
- Many states (27)
- Sensors
  - Barometric altitude
  - Radar altitude
  - Terrain elevation DB
- Solve this using a marginalized particle filter.



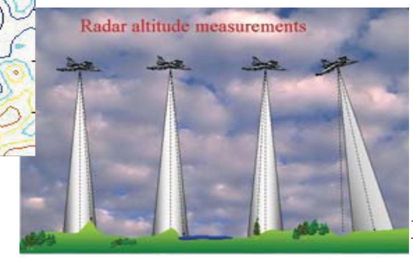
Bonus Lecture

Application Example – Fighter Aircraft Navigation



Green (O) Marginalized particle filter estimate

Red (+) True horizontal position



Bonus Lecture

UAV Simultaneous Localization and Mapping (SLAM)

Joint work with David Törnqvist and others.

Aim: Compute the pose of the UAV and a map without using GPS and already existing maps.

Sensors used:

- IMU (3 acc. 3 gyro.)
- Camera
- Barometer (altitude)

Moviii

www.moviii.liu.se

Bonus Lecture

Key Factorization

State vector

$$x_t = \begin{pmatrix} x_t^p \\ x_t^k \\ m_t \end{pmatrix}$$

Nonlinear vehicle states (x_t^p)

Linear vehicle states (x_t^k)

Linear(ized) map states (3D feature positions) (m_t)

$$p(x_{1:t}^p, x_t^k, m_t | y_{1:t})$$

$$= \underbrace{p(x_{1:t}^p | y_{1:t})}_{\text{particle filter}} \underbrace{p(x_t^k | x_{1:t}^p, y_{1:t}) \prod_{j=1}^{M_t} p(m_{j,t} | x_{1:t}^p, x_t^k, y_{1:t})}_{\text{(extended) Kalman filter}}$$

Generalizes the so called FastSLAM algorithm to high dimensional state vectors

Bonus Lecture

UAV SLAM (Experimental Data)

Yamaha Rmax helicopter

Test facility, Revinge, Sweden

- ○ New map entry
- + Measured map entry
- • Particle for position