The particle filter and its application to positioning

Literature
Monte Carlo Sampling
Stochastic estimation
MCMC
Particle filter
Marginalization
Applications
1. Terrain nav.
2. Underwater nav
3. Car nav.
4. Cell phone pos
The particle filter and its application to positioning

Fredrik Gustafsson
Prof in Communication Systems
Department of Electrical Engineering
Linköping University
fredrik@isy.liu.se
Background and acknowledgement

Presentation based on

- Work in competence center ISIS with
  - SAAB Aircraft: aircraft terrain aided positioning and navigation
  - SAAB Dynamics: missile tracking and torpedo positioning
  - NIRA Dynamics: Map-Aided Positioning (MAP) for cars
  - Ericsson: wireless networks
Literature

- Robert and Casella, Monte Carlo Statistical Methods, Springer, 1999
- Gilks, Richardson and Spiegelhalter, Markov Chain Monte Carlo in Practice, 1996
Random number generation

- Uniform distribution $U \in [0,1]$ – integer multiplication with truncation – shift registers
- General PDF $f(x)$, compute distribution function $F(x)$, take
  \[ X = F^{-1}(U) \]
- This inverse may be complicated to compute, resort to numerical methods
Monte Carlo based estimation

- Goal: estimate \( I = E(g(X)) = \int g(x)p(x)dx \)
- Given: function \( g(x) \), pdf \( p(x) \), random number generator with pdf \( q(x) \)

The Monte Carlo method. Generate \( N \) samples of \( X \) and take

\[
\hat{I} = \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)})
\]

The law of large numbers assures convergence and the central limit theorem assures asymptotic Gaussian distribution.

Problem if \( p(x) \) can be evaluated but not sampled (\( F(x) \) hard to compute and invert analytically), so we cannot generate samples of \( X \)
Accept-reject methods

1. Let U be uniform and generate X from arbitrary pdf q(x)
2. Accept Y=X if

\[ U \leq \frac{p(X)}{Mq(X)} \]

otherwise return to 1

Proof:

\[ P(Y \leq y) = P(X \leq y | U \leq \frac{p(X)}{Mq(X)}) \]

\[ = \frac{P\left(X \leq y, U \leq \frac{p(X)}{Mq(X)}\right)}{P\left(U \leq \frac{p(X)}{Mq(X)}\right)} = \int_{-\infty}^{y} p(x)dx \]
Accept-reject example

- \( p(x) \) truncated Gaussian, \( q(x) \) uniform, both defined on the interval \([-4,4]\), and \( M=4 \).
- Left figure shows true and estimated distributions, the right one the chain of random numbers.
- The larger \( M \), the more rejections and the slower convergence.
Accept-reject example

- Convergence in distribution
Example: Matlab’s `randn`

- **Ancient versions:** \( \text{randn}(1,1) = \text{sum}(\text{rand}(12,1)) - 6 \)
- **Prior ver. 5:** Made use of polar transformation

\[
(X_1, X_2) = F^{-1}(U_1, U_2)
\]

where U and V are uniformly distributed on the unit disc. Matlab code:

```matlab
r = 2;
while r > 1
    U = 2 * rand(2, 1) - 1;
    r = U' * U;
end
X = sqrt(sqrt(-2 * log(r) / r)) * U;
```

The particle filter and its application to positioning

Fredrik Gustafsson
Example: Matlab’s `randn`

- After ver. 5.3: the Ziggurat algorithm (Marsaglia, Knuth).
- Accept-reject idea: accept uniform 2D numbers in the upper half plane that fall below \( p(x) \)
- Compute once in each session a rectangle approximation of \( p(x) \)

\[
x_k : (f(x_k) - f(x_{k+1}))x_k = c, \quad k = 1, \ldots, n-1
\]

\[
\sigma_k = x_{k-1} / x_k
\]

Accept rule (97% accepted) code:

```matlab
K = ceil(128*rand);
U = 2*rand-1;
if abs(U) < sigma(k)
    X = U * x(k);
    return
end
```

Fredrik Gustafsson
Importance sampling

Generate $N$ samples $X$ from proposal pdf $q(x)$ (support of $q(x)$ includes support of $p(x)$) and take

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x^{(i)})}{q(x^{(i)})} g(x^{(i)}) \approx \int \frac{p(x)g(x)}{q(x)} q(x) dx = I$$

Example: Mean of $(g(x)=x)$
$p(x)=0.5N(x;0,1)+0.5N(x;3,0.5)$
using $q(x)$ uniform on $[-5,5]$.

Histogram of importance weights $\frac{p(x^{(i)})}{q(x^{(i)})}$
This is a combination of
1. Monte Carlo methods,
2. Accept-reject
3. Importance sampling,
with the difference that the proposal distribution depends on the previous sample,

\[ Z \propto q(z \mid x^{(i)}) \]

and thus forms a Markov Chain. The idea is that the sequence of samples will converge to \( p(x) \)
1. Let $U$ be uniform and generate $Z \sim q(z \mid x^{(i)})$

2. Accept $x^{(i+1)} = Z$ if

$$U \leq \min\left(1, \frac{p(Z)q(x^{(i)} \mid Z)}{p(x^{(i)})q(Z \mid x^{(i)})}\right)$$

Otherwise $x^{(i+1)} = x^{(i)}$

The independent M-H algorithm is very similar to the accept-reject method. Take

$$Z \sim q(z) \quad U \leq \min\left(1, \frac{p(Z)q(x^{(i)})}{p(x^{(i)})q(Z)}\right)$$
Algorithm in special case:

- Use Bayes rule
  \[ p(x \mid y) \propto p(y \mid x) p(x) \]
- Random walk Markov chain
  where \( Z \) from \( q(z) \)
- Acceptance rule

\[
U \leq \min \left( 1, \frac{p(y \mid x^{(i)} + Z) p(x^{(i)} + Z) q(Z)}{p(y \mid x^{(i)}) p(x^{(i)}) q(-Z)} \right)
\]

In the symmetric case \( p(Z) = p(-Z) \) (Gaussian random walk for instance) with non-informative prior (\( p(x) = c \)), then we always accept the new state if the likelihood increases. The smaller likelihood, the lower acceptance probability.

The case when using the prior as proposal distribution gives a similar rule and interpretation.
Gibbs sampling

- For high-dimensional pdf’s p(x), the marginals may be much simpler to describe than the multivariate distribution. Then sample from one marginal at the time.

Example: Generate samples X from

\[ X \mid \sigma^2 \sim N(0, \sigma^2) \]

\[ \sigma^2 \mid X \sim Exp(1) \]

Theoretical distribution is Laplace
• Idea: given a sequence of iid random variables $X(i)$ from $p(x)$, generate a new sequence by randomly resampling $N$ samples from this set, with replacement.

• Motivation:

$$ I = \int g(x) p(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)}) \approx \sum_{i=1}^{N} \frac{N(i)}{N} g(x^{(i)}) $$

if $E(N(i)) = 1$

The point is that many data sets of $N$ samples can be generated from just one, and one can get an idea of variability in the estimates (that are non-linear transformation of the samples).
Transition to particle filter

- The particle filter contains ideas from accept-reject, Metropolis-Hastings, Gibbs, MCMC and Bootstrap!
- MCMC-interpretation: \((x(1),\ldots x(t)) \rightarrow (x(1),\ldots,x(t+1))\) forms a Markov chain. Since the state space is increasing, the standard results do not apply automatically.
Filtering model

**Model:**

\[
x_{t+1} = f(x_t, w_t),
\]

\[
y_t = h(x_t, e_t).
\]

\[
x_{t+1} \sim p(x_{t+1}|x_t)
\]

\[
y_t \sim p(y_t|x_t)
\]

**Goal:** Transform the information in the measurements \( Y_t = \{y_i\}_{i=1}^t \) to \( x_t \)

Compute \( p(x_t|Y_t) \)
Bayes

Thomas Bayes theorem is instrumental in all derivations

\[ p(x|y) = \frac{p(y|x)p(x)}{p(y)} \]
General Solution

\[ p(x_t|Y_t) = \frac{p(y_t|x_t)p(x_t|Y_{t-1})}{p(y_t|Y_{t-1})} \]

\[ p(y_t|Y_{t-1}) = \int_{\mathbb{R}^n_x} p(y_t|x_t)p(x_t|Y_{t-1}) \, dx_t \]

\[ p(x_{t+1}|Y_t) = \int_{\mathbb{R}^n_x} p(x_{t+1}|x_t)p(x_t|Y_{t}) \, dx_t \]

**Approximate solution**

**PF:** Approximate optimal filter on correct model

**EKF:** Optimal filter on approximate model
Numerical alternatives

- Point-mass filter: Design a grid for the state space, compute

$$p(x_{t+1} \mid Y_t) = \int p(x_{t+1} \mid x_t)p(x_t \mid Y_t)dx_t = \int p_w(x_{t+1} - f(x_t))p(x_t \mid Y_t)dx_t$$

$$p(x_t \mid Y_t) = \frac{p(y_t \mid x_t)p(x_t \mid Y_{t-1})}{p(y_t \mid Y_{t-1})} = C_{p_e}(y_t - h(x_t))p(x_t \mid Y_{t-1})$$

for each grid point!
  - Simple to implement.
  - Adaptive translation and scaling of state grid required.
  - State space soon becomes very sparse, that is, a large number of grid points are required.
  - Variants with Gaussian sums, splines, rectangles.
Numerical alternatives

- The Monte Carlo idea: Simulate a large number of trajectories of the system

\[ x_{t+1} = f(x_t) + w_t \]
\[ y_t = h(x_t) + e_t \]

Compute the likelihood of each trajectory, and compute the weighted state mean.

- Simple to implement.
- Does not work at all! The state space grows with time and becomes very large.
Particle filter algorithm

Generic Particle Filter

1. Generate random states \( x_{0}^{(i)} \in p(x_0) \)
2. Compute likelihood
   \[
   \omega_t^{(i)} = p_e(y_t - h(x_t^{(i)}))
   \]
3. Resampling: \( x_t^{(i)} \sim \omega_t^{(i)}, \omega_t^{(i)} = \frac{1}{N} \)
4. Prediction: \( x_{t+1}^{(i)} = f(x_t^{(i)}) + w_t^{(i)}, w_t^{(i)} \in p_w \)

Example: \( x(t+1) = x(t) + v(t) + w(t) \),
\[
y(t) = h(x(t)) + e(t)
\]
EKF versus PF

Example: $x(t+1) = x(t) + v(t) + w(t)$,
$y(t) = h(x(t)) + e(t)$

EKF: linearize $h(x)$ around current estimate.

This works well for good initialization and small noise

EKF linearizing around true trajectory gives Cramer-Rao lower bound as $P(t)$!
Particle filter interpretation

1. Point-mass filter with stochastic and adaptive grid
2. Monte-Carlo method with recursive cut and branch of state trajectories
3. Approximate MCMC method
4. Genetic algorithm with probability based rules

The main difference to pure numerical methods is that certain variants of the law of large numbers and the central limit theorem exist. Basically, these say the estimation error is bounded as

\[ \text{Var}(\hat{I}) \propto \frac{p(t)}{N} \]

where \( p(t) \) is a polynomial and \( N \) is the number of particles. At least in theory, the accuracy is independent of the state dimension.
Particle filter patches

1. Basic algorithm bootstrap inspired. One alternative is to resample only when needed, compute the effective number of samples $N_{eff}$ smaller than some threshold (Sampling Importance Resampling SIR)

$$1 \leq N_{eff} = \frac{1}{\sum_{i=1}^{N} \left( \bar{q}_{t}^{(i)} \right)^2} \leq N$$

2. Preview the next measurement in the time update (prior editing)

3. Add roughening noise to state equation to help the particles explore a larger surrounding.

4. Large number of particles needed in the transient, use clever initialization procedures or an adaptive number $N(t)$ of particles (and perhaps also sample time!).

5. Feasability: is the required number of particles possible to implement (memory and real-time constraints)? If not, try to marginalize linear states and exclude these in the particle filter. This gives a smaller state space, saving particles.
Marginalization I (The basic idea)

- A.k.a. Rao-Blackwellization
- The case when the same Riccati equation can be used for all KF’s
- General idea, exploit linear sub-structures

\[
\mathbf{x}_t = \begin{bmatrix}
\mathbf{x}^l_t \\
\mathbf{x}^n_t
\end{bmatrix}
\]

\[
\begin{bmatrix}
\mathbf{x}^l_t(i) \\
\mathbf{x}^n_t(i)
\end{bmatrix}
\]

\[
\mathbf{x}_t(i) P(i)
\]
Basically it is all about using Bayes’ theorem:

\[ p(x_t^l, X_t^n | Y_t) = p(x_t^l | X_t^n, Y_t)p(X_t^n | Y_t) \]

\[ \underbrace{\text{OptimalKF}}_{\text{Appr. PF}} \]

It is instructive to think about information flow when it comes to understanding marginalization.
Marginalization III

General model structure:

\[
\begin{align*}
    x_{t+1}^n &= f_t^n(x_t^n) + A_t^n(x_t^n)x_t^l + G_t^n(x_t^n)\omega_t^n, \\
    x_{t+1}^l &= f_t^l(x_t^n) + A_t^l(x_t^n)x_t^l + G_t^l(x_t^n)\omega_t^l, \\
    y_t &= h_t(x_t^n) + C_t(x_t^n)x_t^l + e_t.
\end{align*}
\]

Nonlinear states \( x_t^n \),
Linear states \( x_t^l \)
The marginalized particle filter algorithm:

1. Initialize the particles.
2. Evaluate weights.
3. Particle filter measurement update, i.e., resample.
4. Particle filter time update.
   4.a Kalman filter measurement update.
   4.b Predict new particles.
   4.c Kalman filter ”measurement” and time update.
5. Iterate from step 2.

Difference from the standard particle filter
A very important model class in applications:

\[
\begin{align*}
    x_{t+1}^n &= A_{n,t}^n x_t^n + A_{l,t}^n x_t^l + G_t^n w_t^n, \\
    x_{t+1}^l &= A_{n,t}^l x_t^n + A_{l,t}^l x_t^l + G_t^l w_t^l, \\
    y_t &= h_t(x_t^n) + e_t.
\end{align*}
\]

Many positioning applications fall into this class, where only the position measured.

The same Ricatti equation for all Kalman filters, thus the same Kalman gain!
Positioning: Dynamic model examples

- Model when velocity is measured:
  \[ x_t = (X_t, X_t, \Psi_t)^T \]
  \[ X_{t+1} = X_t + T v_t \cos(\Psi_t) + w_{X,t} \]
  \[ Y_{t+1} = Y_t + T v_t \sin(\Psi_t) + w_{Y,t} \]
  \[ \Psi_{t+1} = \Psi_t + T \dot{\Psi}_t \]
  \[ y_t = h(X_t, Y_t) + e_t \]

- Model when velocity is **not** measured (then estimate it!):
  \[ x_t = (X_t, \dot{X}_t, Y_t, \dot{Y}_t) \]
  \[ x_{t+1} = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & T^2 / 2 \\ 0 & T \end{pmatrix} w_t \]
  \[ y_t = h(X_t, Y_t) + e_t \]
Positioning: Sensor examples

- Generic measurement depends only on the current position \((X,Y)\)
  \[
y_t = h(X_t, Y_t) + e_t
\]

- Examples of target tracking standard sensors:
  - Radar measures range and bearing to \((X,Y)\)
  - IR and radar warning systems measure bearing to \((X,Y)\)

- Examples of navigation standard sensors:
  - INS measures acceleration and rotations of coordinates \((X,Y)\)
  - GPS
  - Wireless network measurements: Loran-C, GSM (cell ID, timing advance, time difference of arrival)
  - Maps!!

The particle filter and its application to positioning

Fredrik Gustafsson
Positioning: Sensor examples in com.

- Cell-Identifier (CI)
- Sector information or Angle of Arrival (AOA) (requires adaptive antennas)
- Timing Advance (TA) $d_{TA} = \frac{1}{2} \times 3.69\mu s$ symbol period $\times 3e8 \text{ m/s} = 554\text{m}$
- Enhancement: usage of received signal levels (RXLEV), see operator’s power attenuation map below.
Positioning: GIS as a sensor

Digital Terrain Elevation Database: 200,000,000 grid points
points between 50 meters
uncertainty 2.5 meters

Ground Cover Database: 14 types of vegetation
Obstacle Database: All man made obstacles above 40 m

(C) 1999 Saab, Stefan Ahlqvist
Aircraft navigation

New (2G) integrated navigation and landing system for JAS is based on particle filter for terrain navigation.

Next generation may be based on marginalized particle filter for sensor fusion (27 state EKF today).
Two-dimensional version of example:

\[ x(t+1) = x(t) + v(t) + w(t), \]
\[ y(t) = h(x(t)) + e(t) \]

Two applications:

1. \( h(x) \) terrain map and
   \[ y(t) = \text{barometric altitude} - \text{height radar} \]
   \[ v(t) \text{ from INS} \]

2. \( h(x) \) under-water depth map and
   \[ y(t) = \text{echo sound} \]
   \[ v(t) \text{ from rudder and propeller speed} \]

Note:

1. Cramer-Rao: position error > sqrt (altitude error * velocity error / terrain variation)
2. The particle filter normally attains the Cramer-Rao bound!
2D Example

- Simulated flight trajectory on GIS
- Snapshots at t=0, 20 and 31 seconds
- Red: true  Green: estimate
Terrain-aided navigation III

Animation of terrain navigation in 2D using real GIS
Underwater terrain navigation

- Underwater navigation using sonar sensors with aid from rudder and log (propeller speed) for motion model.

- Underwater map built in Swedish lake, using surface ship with GPS for validation.

- Particle filter satisfies Cramer-Rao lower bound (depends on the map)
Car positioning I

- Initialization using manual marking or GSM positioning
Car positioning II

• Initializalization using manual marking or GSM positioning
• After slight bend, four particle clusters left
Car positioning III

- Initialization using manual marking or GSM positioning
- After slight bend, four particle clusters left
- Convergence after turn
Car positioning IV

- Initialisation using manual marking or GSM positioning
- After slight bend, four particle clusters left
- Convergence after turn
- Spread along the road
Car positioning V

- Particle filter using street map and $v(t), \Psi(t)$ from car’s ABS sensors.
- Green - true position
- Blue – estimate
- Red - particles
Car positioning VI

- Particle filter using street map and \( v(t), \dot{\Psi}(t) \) from car’s ABS sensors and GSM cell ID and sector for initialization
- Purple: GPS
- Blue: particles
- Light blue: estimate
- Photo background
Car positioning VII

- Red – GPS
- Light green: particles
- Blue: estimate (after convergence)
- Real-time implementation on Compac iPAQ
- Works without or with GPS
- Map database background
Car positioning VIII

- Complete navigator with voice guidance!
- Integer implementation of the particle filter (ISIS project)
- PF in simulation mode off-road
- # particles up to 15000 (without GPS) or as small as 50 (with GPS)
- On-going R&D work at NIRA Dynamics AB and ISIS
1. Cell ID and Received signal strength:
   - Sector information (usually 60 degrees)
   - One or more sectors:
     - Connected antenna (yellow pages service)
     - Power measurements from 5 (GSM) or 6 (WCDMA) antennas
   - Power attenuation: Hata’s formula or map-based
TDOA static positioning

- Particle filter with decaying roughening noise
Network positioning

- Positioning of cellular phones using network measurements
- Focus on dynamic estimation and automotive application
- Flexible framework configurable with an asynchronous mixture of information sources:
  1. Map: street, altitude, attenuation
  2. Angle and range to fix-point: AOA, TOA, TDOA, GPS
  3. Velocity and turn rate: \(v(t), \dot{\Psi}(t)\)
- Particle filter suitable because of its ability to include:
  1. Non-linear constraints (map)
  2. Non-Gaussian noise (Rice and Rayleigh fading, operator’s power attenuation map, etc.)
Conclusions

- Flexible framework for **positioning** configurable with an asynchronous mixture of information sources:
  1. Map: street, altitude, attenuation
  2. Wireless measurements as angle and range to fix-point: AOA, TOA, TDOA, GPS
  3. Inertial measurements like velocity and turn rate: \( v(t), \dot{\Psi}(t) \)
  4. Distance and range measurements from radar, IR, etc.

- **Particle filter** suitable because of its ability to include:
  1. Non-linear dynamic models
  2. Non-linear constraints (map)
  3. Non-Gaussian noise (radar loop, geometric transformation, Rice and Rayleigh fading, operator’s power attenuation map, etc.)