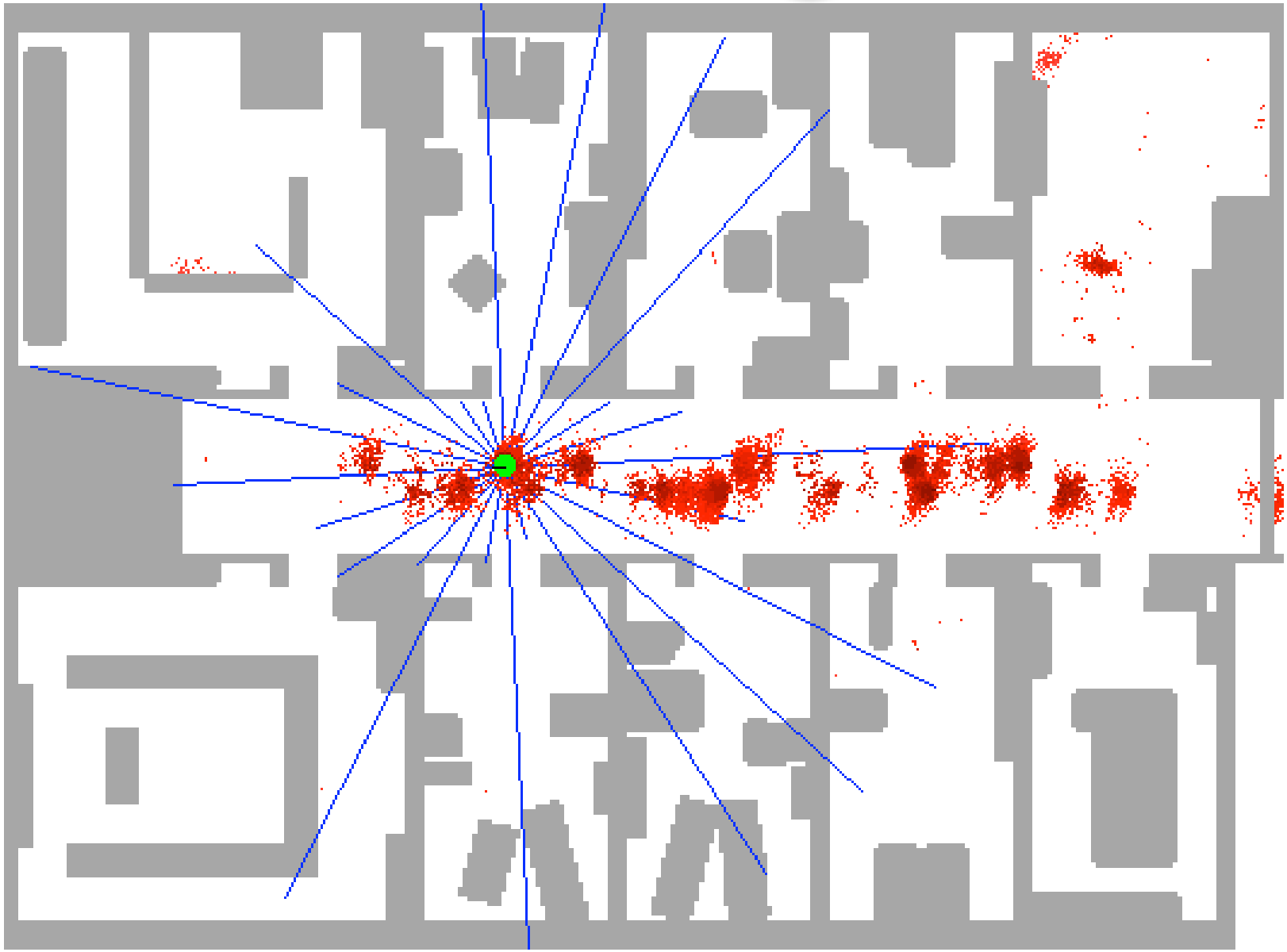
Particle Filtering

- Monte Carlo methods (& Importance Sampling)
- Particle Filtering

Particle Filter Movies

- State: Location
  (+ direction, speed)
- Observations:
  Camera, Sonar , Lazer,
  Wheel speed.
- Task: track position (while performing policy)
- Goal: put out fires.

Sonar Navigation

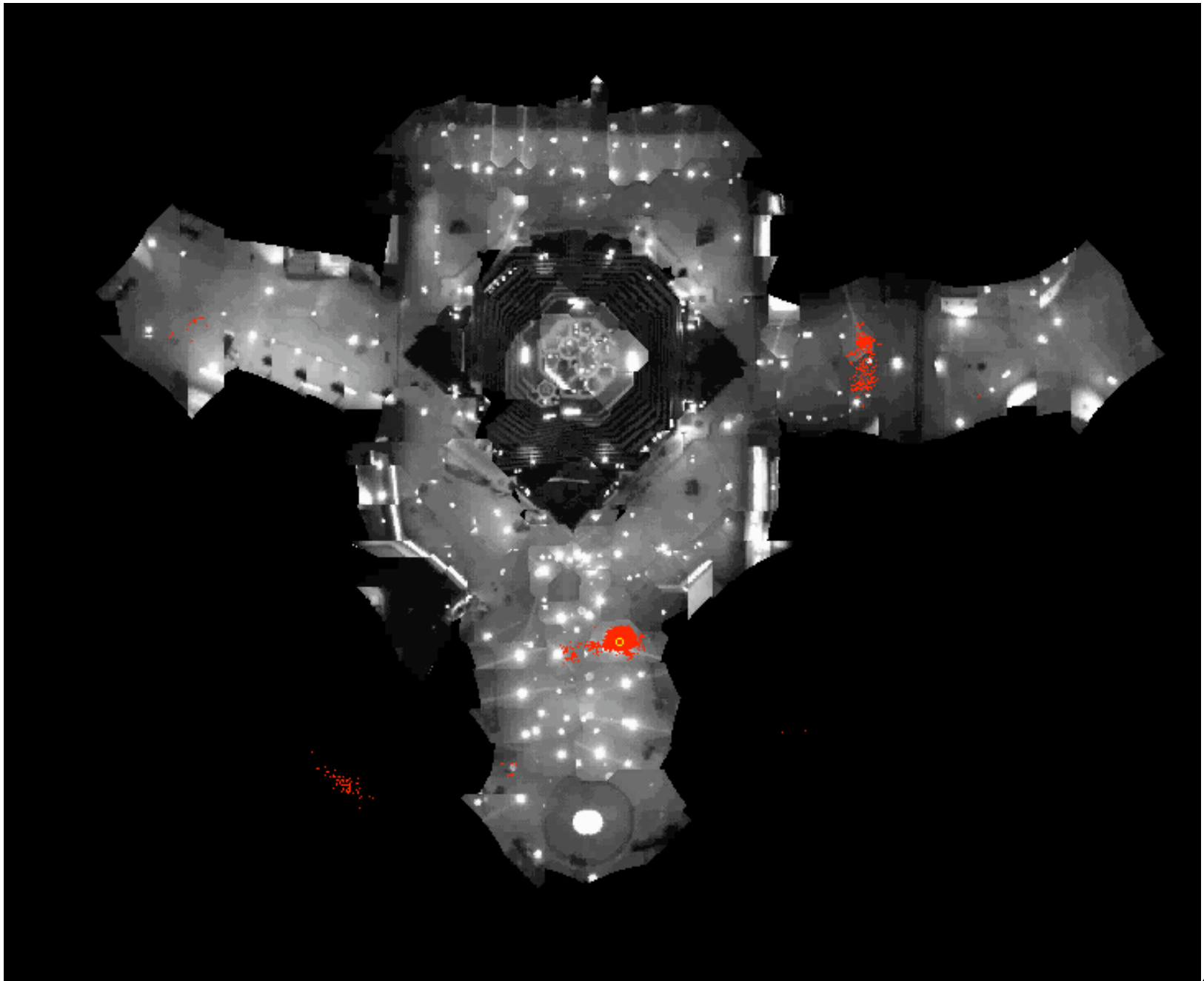


Dieter Fox

Using ring of 24 sonar sensors. Initial estimated position is wrong

Smithsonian Tour Guide

Navigates using camera pointed at ceiling.



Sebastian Thrun

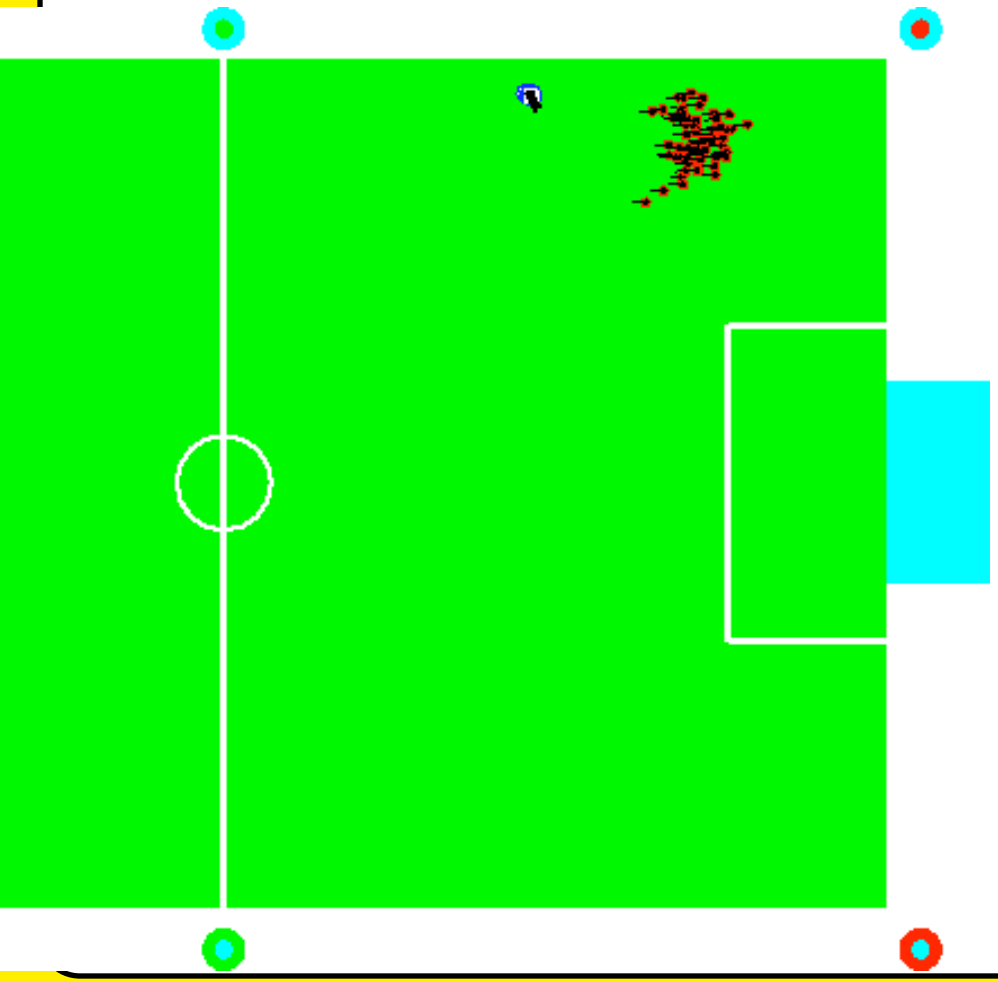
Object Tracking

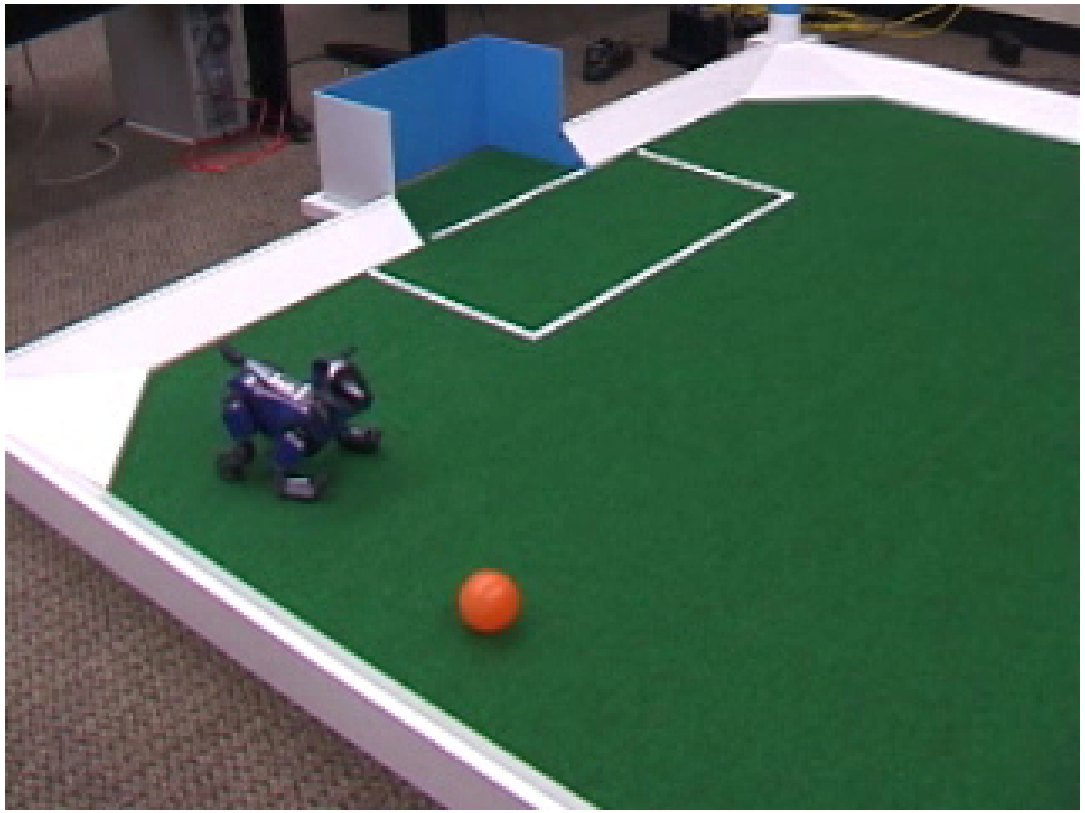
- "Condensation" Alg.
- Also uses shape primitives.

Michal Issard & Andrew Blake

Aibo Tracking (RoboCup)

- Both Aibo and Ball are tracked using particle filtering.





Dieter Fox

Particle Filtering

- *Monte Carlo* methods use a sample (particle cloud) to represent a distribution.
- In noisy dynamic systems we wish to know the distribution of state values as they evolve in time
- When the initial distributions are not Gaussian or the transitions are not linear the state distributions are not Gaussian.
- Particle filtering uses Monte Carlo methods to represent distributions and *Dynamic Programming* to propagate them in time.

Intro to Monte Carlo Methods

- Monte Carlo methods deal with the following two problems:
  - Find an i.i.d sample, $\{\mathbf{x}^m\}$ from a probability distribution, $p(\mathbf{x})$.
  - Estimate expected function values under a probability:

$$\Phi = E_p[f(\mathbf{x})] = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

Solving problem 1 gives a solution to problem 2:

$$\hat{\Phi} = \frac{1}{d}\sum_r f(\mathbf{x}^r)$$

- Obviously, $E(\hat{\Phi}) = \Phi$ and the variance of the estimate is

$$var(\hat{\Phi}) = \frac{var(f)}{d} = \frac{1}{d}\int (f(\mathbf{x}) - \Phi)^2 p(\mathbf{x})d\mathbf{x}$$

- More examples lower the variance (linearly).
- Variance does not grow with N, the dimension of $\mathbf{x}$. A small number of samples may be sufficient even if N is large. But, sampling from $p(\mathbf{x})$ may be difficult ("curse of dimensionality")

Importance Sampling

- Being able to calculate $p(\mathbf{x})$ does *not* mean you can sample from $p(\mathbf{x})$.
- Basic idea: create a random sample and weigh the particles according to their probabilities:
- If you can calculate $p(\mathbf{x})$ and can sample from some other distribution, $q(\mathbf{x})$ and q is such that $supp(p) \subseteq supp(q)$ (i.e. if $p(x)>0$ then $q(x)>0$) then:

$$E_q\left[f(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}\right] = \int f(\mathbf{x})q(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}d\mathbf{x} = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} = E_p[f(\mathbf{x})]$$

- So, given a sample from distribution q above, we can estimate $E_p[f(\mathbf{x})]$:

$$\hat{\Phi}_p(f) = \frac{1}{M}\sum_{m=1}^{M} f(\mathbf{x}^m)\frac{p(\mathbf{x}^m)}{q(\mathbf{x}^m)}$$

- Again, $E(\hat{\Phi}) = \Phi$

- The variance is

$$\begin{aligned} var(\hat{\Phi}) &= \frac{1}{M}\left(E_q\left[\left(f(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)^2\right] - \left(E_q\left[f(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}\right]\right)^2\right) \\ &= \frac{1}{M}\left(E_q\left[\left(f(x)\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)^2\right] - \left(E_p\left[f(\mathbf{x})\right]\right)^2\right) \end{aligned}$$

- For $f(\mathbf{x})=1$

$$var(\hat{\Phi}) = \frac{1}{M} \quad var\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)$$

- So the variance diminishes as $q \rightarrow p$.
- Suppose q is simply uniform over the whole (finite) space, if p is concentrated, then $p/q \rightarrow \infty$ as the space expands (exponentially in N).

- Quite often, p is known up to a normalization constant, i.e. $\pi(\mathbf{x}) = \alpha p(\mathbf{x})$ and α is not known. We can still do importance sampling!

Define $$w(\mathbf{x}) = \frac{\pi(\mathbf{x})}{q(\mathbf{x})}$$

note that $$E_q[w(\mathbf{x})] = \int \frac{\pi(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) d\mathbf{x} = \int \pi(\mathbf{x}) d\mathbf{x} = \alpha$$

therefore
$$\begin{aligned}
\Phi = E_p(f(\mathbf{x}) &= \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\
&= \int f(\mathbf{x}) q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \\
&= \frac{1}{\alpha} \int f(\mathbf{x}) q(\mathbf{x}) \frac{\pi(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \\
&= \frac{E_q[f(\mathbf{x}) w(\mathbf{x})]}{\alpha} \\
&= \frac{E_q[f(\mathbf{x}) w(\mathbf{x})]}{E_q[w(\mathbf{x})]}
\end{aligned}$$

$$\Phi = \frac{E_q[f(\mathbf{x})w(\mathbf{x})]}{E_q[w(\mathbf{x})]}$$

- We can estimate Φ with a sample by:

$$\hat{\Phi} = \frac{\sum_{m=1}^{M} f(\mathbf{x}^m)w(x^m)}{\sum_{m=1}^{M} w(x^m)}$$

- Note that this estimate is biased. For M=1

$$E_q(\hat{\Phi}) = E_q\left[\frac{f(\mathbf{x}^1)w(x^1)}{w(x^1)}\right] = E_q[f(\mathbf{x}^1)] \neq E_p[f(\mathbf{x}^1)]$$

  but for $M \rightarrow \infty$ it converges to Φ

- Importance sampling is problematic when q is very different from p.
- Other methods (e.g. rejection sampling and the Metropolis method) try to sample from p by sampling from q and throwing away examples. If q and p are very different many particles get thrown away (and sampling takes a long time).

Particle Filtering

- A Monte Carlo method for sequences (a.k.a Sequential Monte Carlo).
- The particles are sequences that get propagated.
- Given a general (Markov) dynamical system,

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \nu_{k-1})$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mu_k)$$

  Where υ and μ are i.i.d process and observation noises. The initial state, **f**,**h** and the noise distributions define the following probabilities:

$$p(\mathbf{x}_0) \quad , \quad p(x_k|x_{k-1}) \quad , \quad p(\mathbf{y}_k|\mathbf{x}_k)$$

- We wish to know $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. In general, this distribution does not belong to a simple parametric family.
- We therefore represent it with a sample of particles and propagate it in time using the Markov assumptions.

Developing a Recursion Eq

- We develop an equation for $p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$.
- Suppose we know $p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$ then from Bayes and Markovity:

$$p(\mathbf{x}_{0:k}|y_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_{0:k},\mathbf{y}_{1:k-1})p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}$$

Bayes →

$$= \frac{p(\mathbf{y}_k|\mathbf{x}_{0:k},\mathbf{y}_{1:k-1})p(\mathbf{x}_k|\mathbf{x}_{0:k-1},\mathbf{y}_{1:k-1})p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}$$

Markov →

$$= \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}$$

$$\propto p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$$

"known" (as particles)

- i.e. we know the distributions that make up $p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$ up to a constant factor (i.e we have $\pi(\mathbf{x})$ but can't sample from it)
- We will therefore use importance sampling

- We choose a sampling distribution, q that factors:

$$q(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})q(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$$

- Define:

$$w^i = \frac{p(\mathbf{x}^i_{0:k}|\mathbf{y}_{1:k})}{q(\mathbf{x}^i_{0:k}|\mathbf{y}_{1:k})}$$

- Plugging in q and p

$$\begin{aligned}\tilde{w}^i &= \frac{p(\mathbf{y}_k|\mathbf{x}^i_k)p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})p(\mathbf{x}^i_{0:k-1}|\mathbf{y}_{1:k-1})}{q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})q(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})} \\ &= \frac{p(\mathbf{y}_k|\mathbf{x}^i_k)p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} w^i_{k-1}\end{aligned}$$

and

$$w^i = \frac{\tilde{w}^i}{\sum_j \tilde{w}^j}$$

- So, taking the particles $\{\mathbf{x}^i_{0:k}\}^M_{i=1}$ sampled from q and weighing them by w_i gives the required estimate $p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$.

Problems

- Given a sample, how do you choose the predicted state? The mean is easy to approximate but the MAP estimate is more difficult. The probability of a specific value of **x** is

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \lim_{M\to\infty} \sum_{i=1}^{M} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i)$$

- To find the approximate MAP value one could replace δ with a Gaussian and search for the maximum value of the mixture..

- As mentioned earlier, *q* should be close to *p*. If *p* is Markov then so can *q*. I.e. we can limit *q* to the form $q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{y}_k)$. A popular choice is $q=p(\mathbf{x}_k|\mathbf{x}_{k-1})$ if it can be sampled from. In this case, $\tilde{w}^i = p(\mathbf{y}_k|\mathbf{x}_k^i)w_{k-1}^i$

- As the sequence advances w^i go to 0 or 1. This happens quickly if *q* differs greatly from *p*. To avoid this, *resampling* is applied, splitting particles with large *w* and eliminating particles with small *w*.
This method is knows as Sequential Importance Resampling (SIR)

Algorithm

Algorithm 1 Particle filter (SIR)

$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^M] = PF(\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^M, \mathbf{y}_k)$

- for i=1:M
  - draw $\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{y}_k)$
  - assign $\tilde{w}_k^i = \frac{p(\mathbf{y}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} w_{k-1}^i$,
- for each i, $w^i = \frac{\tilde{w}^i}{\sum_j \tilde{w}^j}$
- optionally, $[\{\bar{\mathbf{x}}^i, \bar{w}^i\}_{i=1}^M] = RESAMPLE(\{\mathbf{x}^i, w^i\}_{i=1}^M)$

Resampling

Algorithm 2 Resample

$[\{\bar{\mathbf{x}}^i, \bar{w}^i\}_{i=1}^M] = RESAMPLE(\{\mathbf{x}^i, w^i\}_{i=1}^M)$

- for each i in $\{0, \ldots, M\}$, $c_i = \sum_{j=1}^{i} w^i$
- for each i in $\{1, \ldots, M\}$
  - $u_i = uniform\ random\ in\ [0, 1]$
  - $\bar{\mathbf{x}}^i = \mathbf{x}^j$ such that $c_{j-1} \leq u_i < c_j$
  - $\bar{w}^i = \frac{1}{M}$

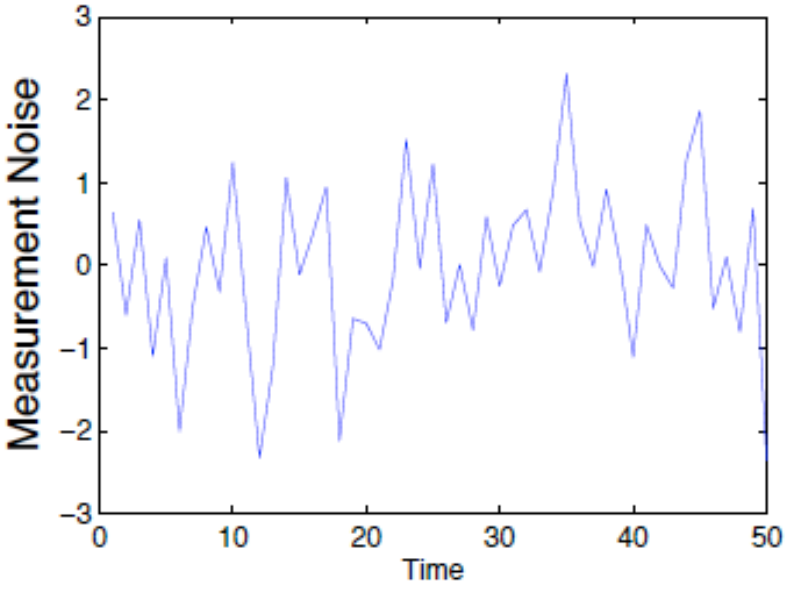
Example

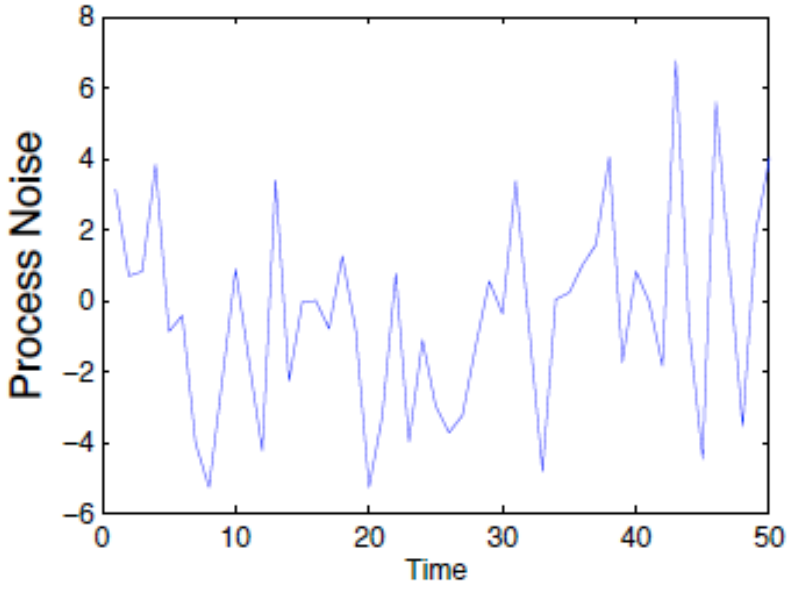
- Dynamic system:

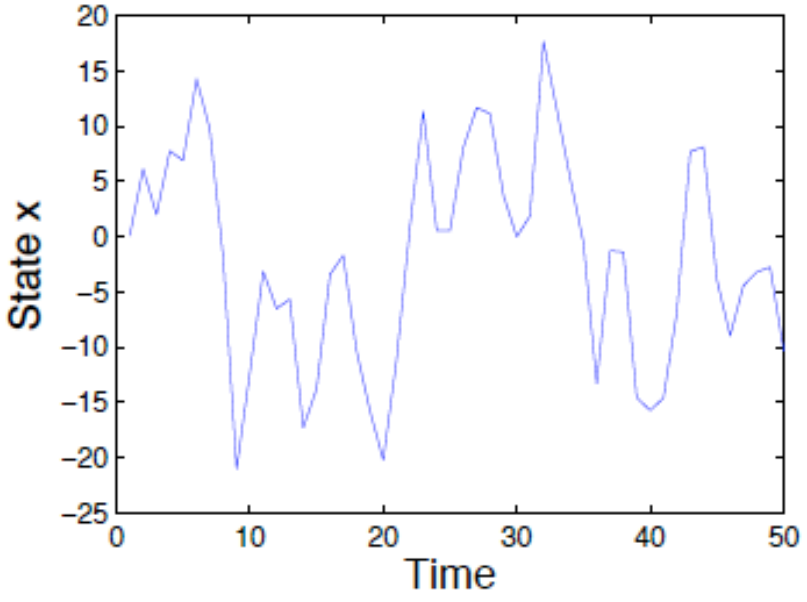
$$
\begin{aligned}
x_k &= \frac{x_{k-1}}{2} + \frac{25 x_{k-1}}{1 + x_{k-1}^2} + 8cos(1.2k) + \nu_{k-1} \\
y_k &= \frac{x_k^2}{20} + \mu_k \\
\nu &\sim \mathcal{N}(0, 10) \\
\mu &\sim \mathcal{N}(0, 1)
\end{aligned}
$$

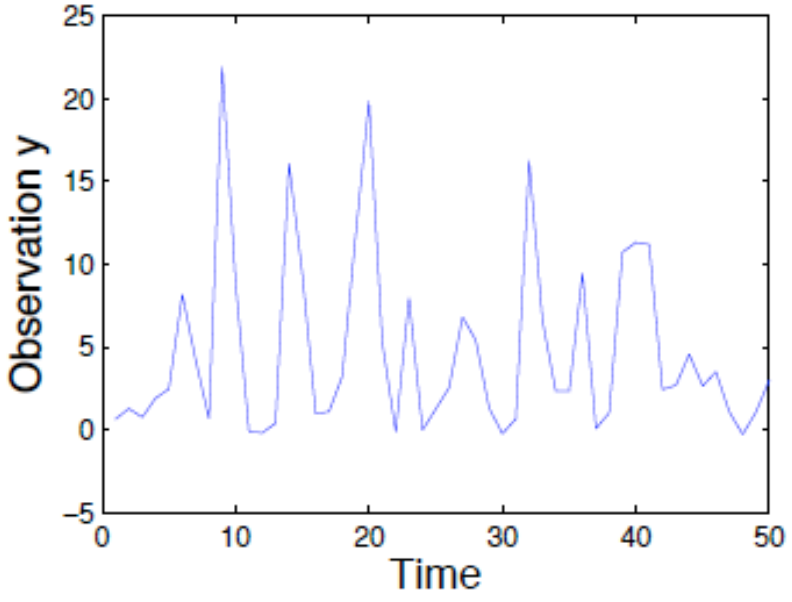
- Choose $q(\mathbf{x}_k^i | \mathbf{x}_{k-1}^i) = \mathcal{N}(\mathbf{x}_{k-1}^i, 10)$

True Values

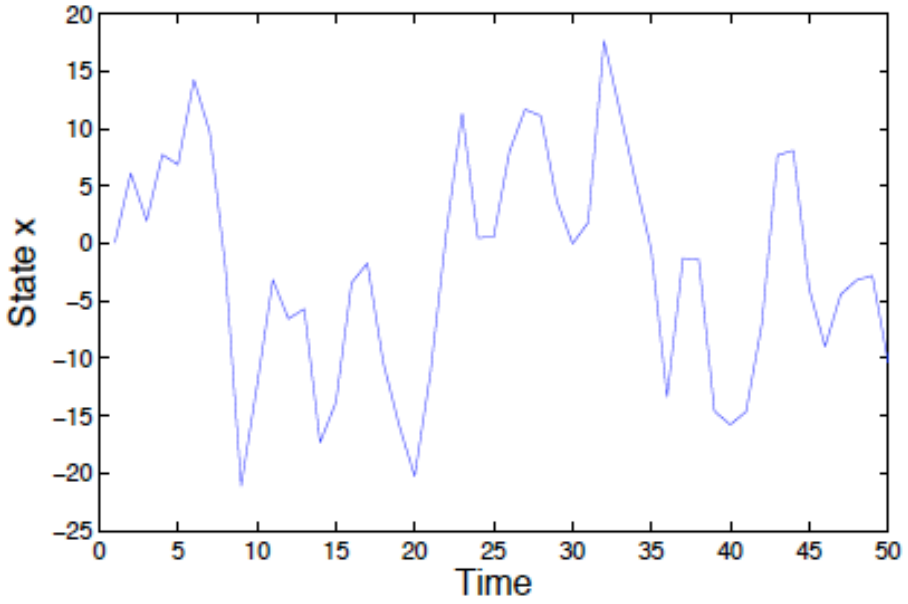

Measurement Noise
Time
3
2
1
0
−1
−2
−3
0
10
20
30
40
50

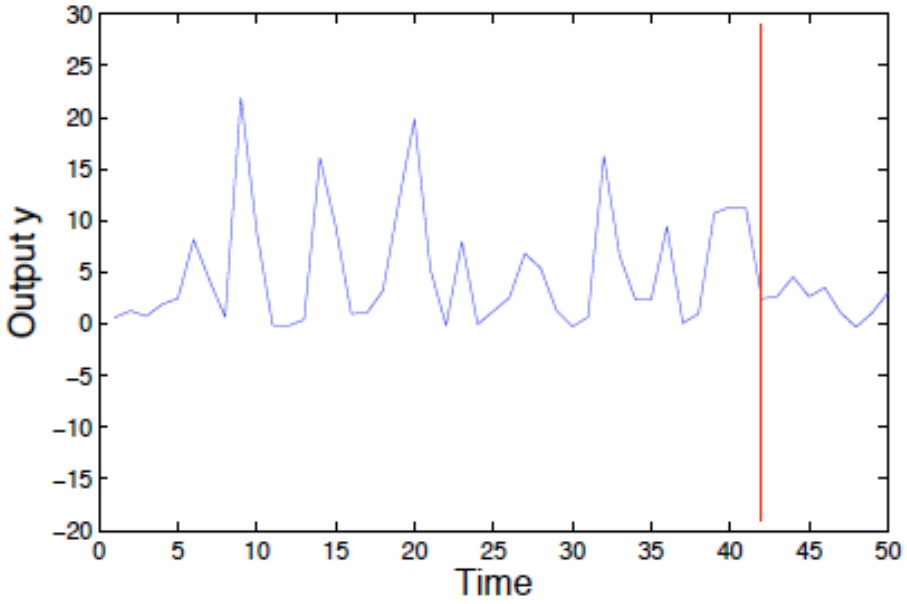

Process Noise
Time
8
6
4
2
0
−2
−4
−6
0
10
20
30
40
50

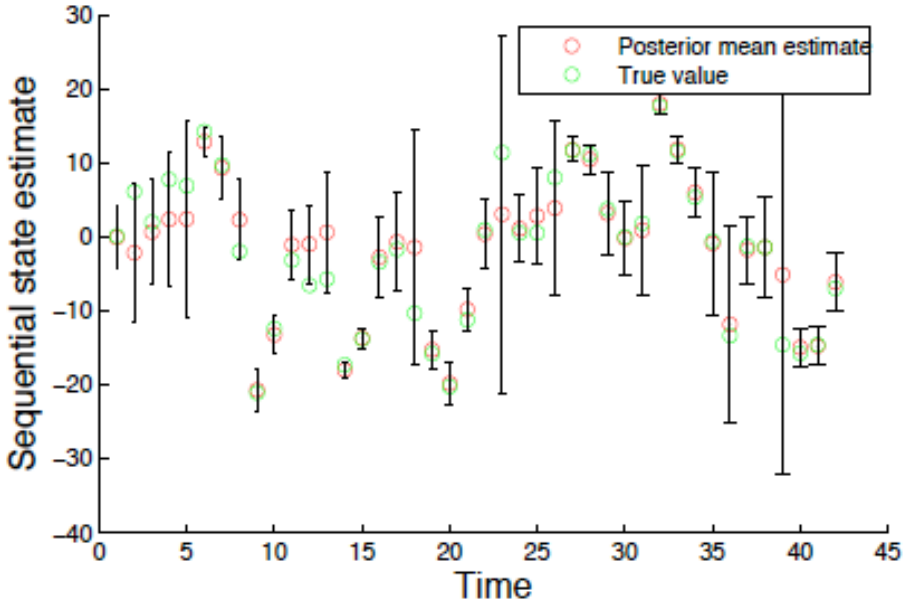

State x
Time
20
15
10
5
0
−5
−10
−15
−20
−25
0
10
20
30
40
50

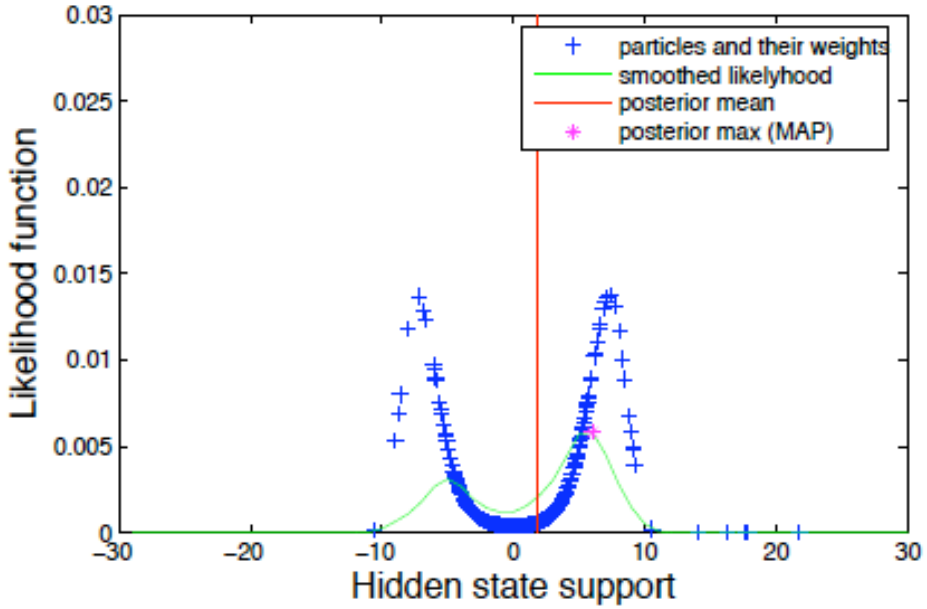

Observation y
Time
25
20
15
10
5
0
−5
0
10
20
30
40
50

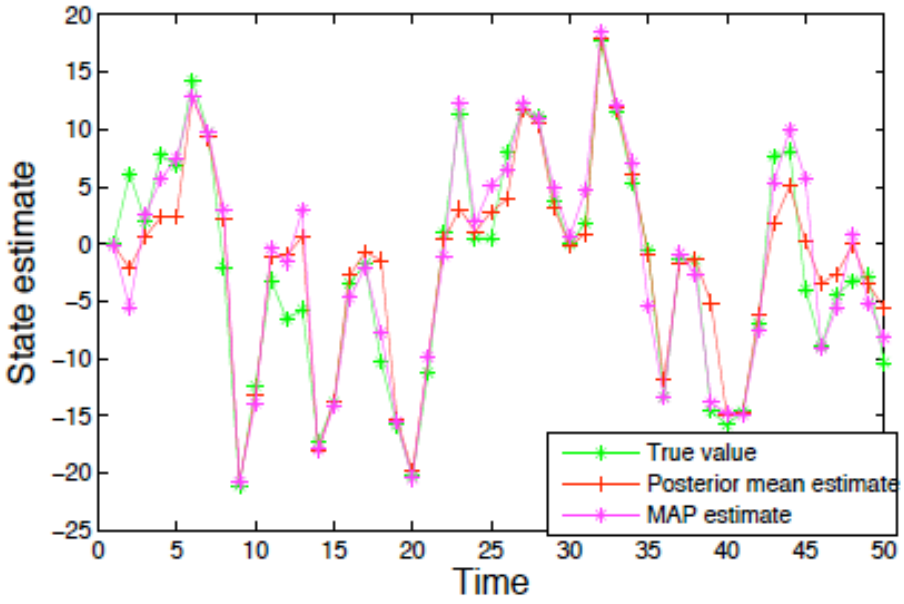
- at 42nd step:

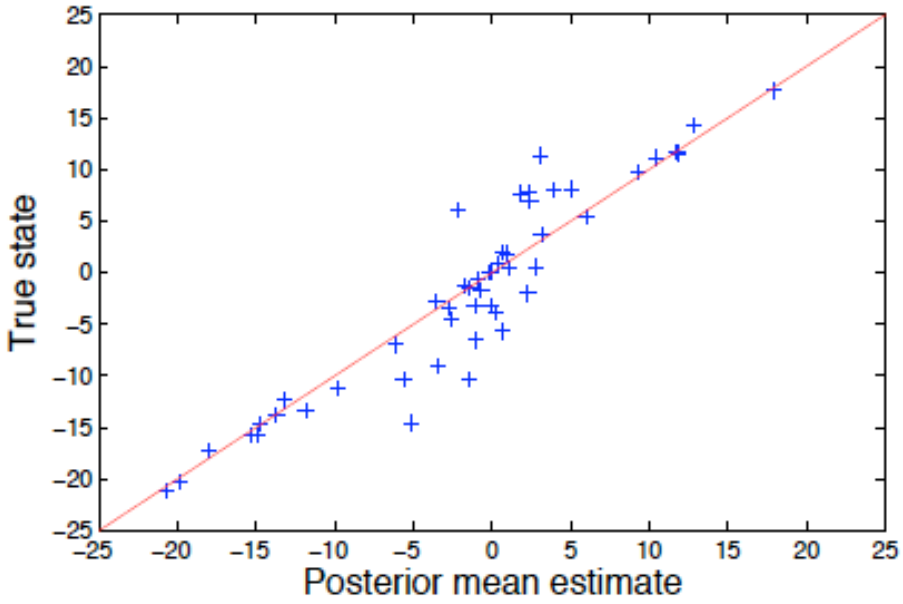

State x
Time

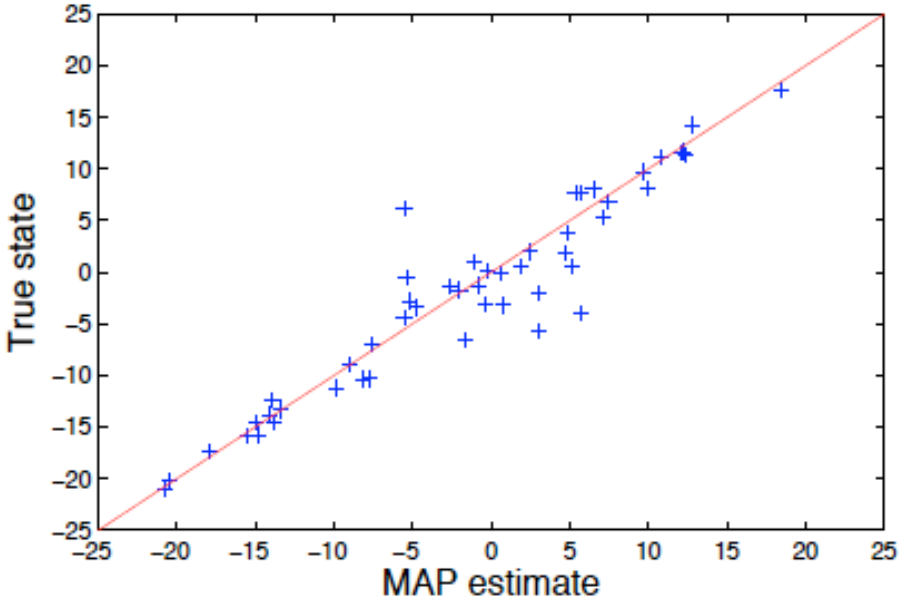

Output y
Time

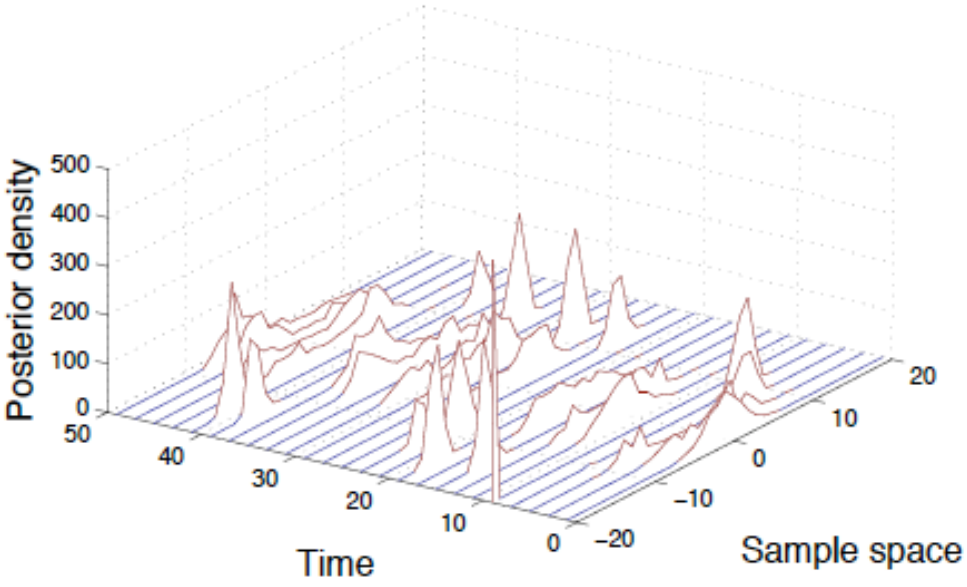

Posterior mean estimate
True value
Sequential state estimate
Time


particles and their weights
smoothed likelyhood
posterior mean
posterior max (MAP)
Likelihood function
Hidden state support


State estimate
Time
True value
Posterior mean estimate
MAP estimate


True state
Posterior mean estimate


True state
MAP estimate


Posterior density
Time
Sample space