Computer Vision 2
WS 2018/19

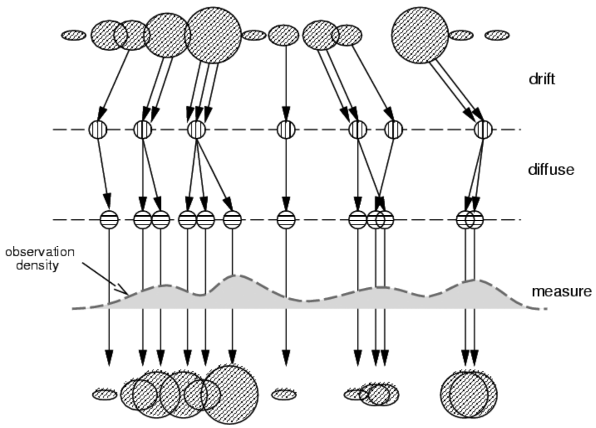
Part 9 – Particle Filters
21.11.2018

Prof. Dr. Bastian Leibe

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

Course Outline

- Single-Object Tracking
- Bayesian Filtering
  – Kalman Filters, EKF
  – Particle Filters
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis



Beyond Gaussian Error Models

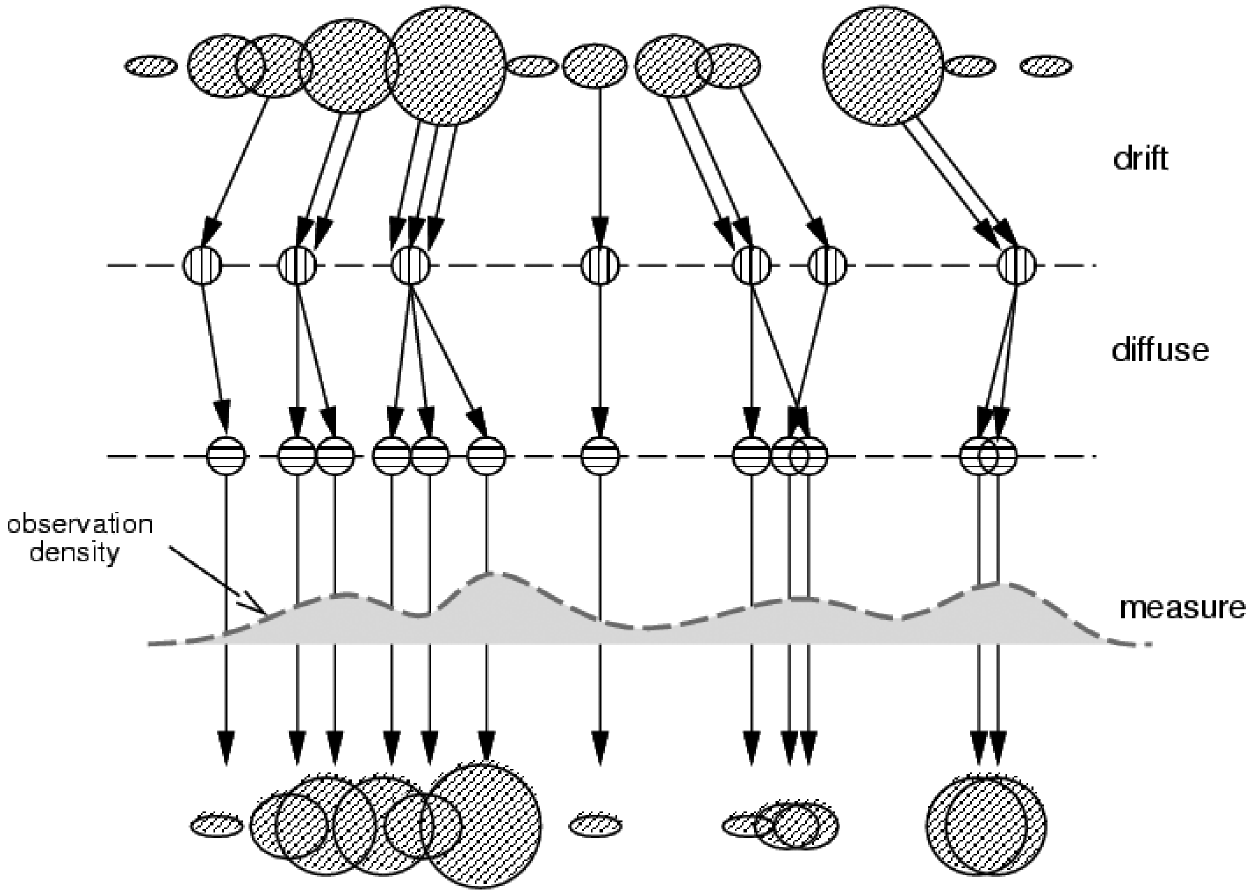


Figure from Isard & Blake

Topics of This Lecture

- Recap: Extended Kalman Filter
- Particle Filters: Detailed Derivation
  - Recap: Basic idea
  - Importance Sampling
  - Sequential Importance Sampling (SIS)
  - Transitional prior
  - Resampling
  - Generic Particle Filter
  - Sampling Importance Resampling (SIR)

Recap: Kalman Filter – Detailed Algorithm

- Algorithm summary
  - Assumption: linear model

$$
\begin{aligned}
\mathbf{x}_t &= \mathbf{D}_t \mathbf{x}_{t-1} + \varepsilon_t \\
\mathbf{y}_t &= \mathbf{M}_t \mathbf{x}_t + \delta_t
\end{aligned}
$$

  - Prediction step

$$
\begin{aligned}
\mathbf{x}_t^- &= \mathbf{D}_t \mathbf{x}_{t-1}^+ \\
\mathbf{\Sigma}_t^- &= \mathbf{D}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{D}_t^T + \mathbf{\Sigma}_{d_t}
\end{aligned}
$$

  - Correction step

$$
\begin{aligned}
\mathbf{K}_t &= \mathbf{\Sigma}_t^- \mathbf{M}_t^T \left(\mathbf{M}_t \mathbf{\Sigma}_t^- \mathbf{M}_t^T + \mathbf{\Sigma}_{m_t} \right)^{-1} \\
\mathbf{x}_t^+ &= \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{M}_t \mathbf{x}_t^- \right) \\
\mathbf{\Sigma}_t^+ &= \left(\mathbf{I} - \mathbf{K}_t \mathbf{M}_t \right) \mathbf{\Sigma}_t^-
\end{aligned}
$$

Extended Kalman Filter (EKF)

- Algorithm summary
  - Nonlinear model

$$\mathbf{x}_t = \mathbf{g}\left(\mathbf{x}_{t-1}\right) + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{h}\left(\mathbf{x}_t\right) + \delta_t$$

  - Prediction step

$$\mathbf{x}_t^- = \mathbf{g}\left(\mathbf{x}_{t-1}^+\right)$$
$$\mathbf{\Sigma}_t^- = \mathbf{G}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{G}_t^T + \mathbf{\Sigma}_{d_t}$$

  - Correction step

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \mathbf{\Sigma}_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{h}\left(\mathbf{x}_t^-\right)\right)$$
$$\mathbf{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{\Sigma}_t^-$$

with the Jacobians

$$\mathbf{G}_t = \left.\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{t-1}^+}$$

$$\mathbf{H}_t = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_t^-}$$

Topics of This Lecture

- Recap: Extended Kalman Filter
- Particle Filters: Detailed Derivation
  - Recap: Basic idea
  - Importance Sampling
  - Sequential Importance Sampling (SIS)
  - Transitional prior
  - Resampling
  - Generic Particle Filter
  - Sampling Importance Resampling (SIR)

Recap: Propagation of General Densities

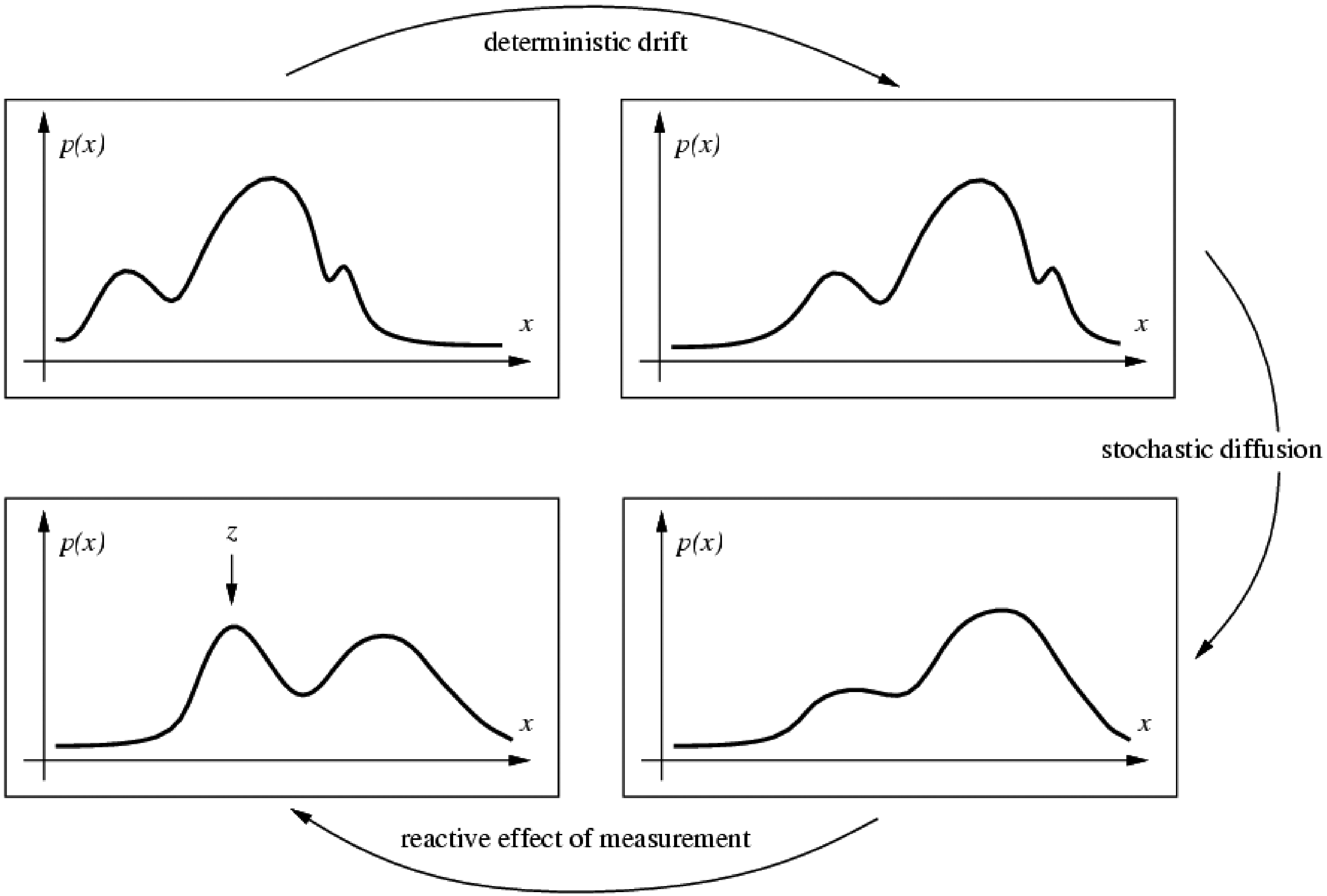
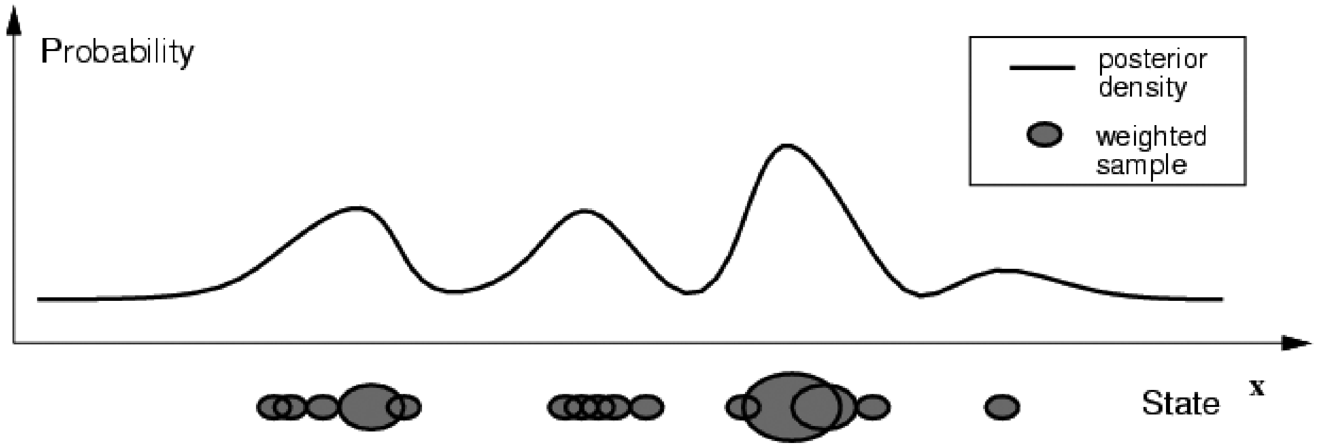


Figure from Isard & Blake

Recap: Factored Sampling



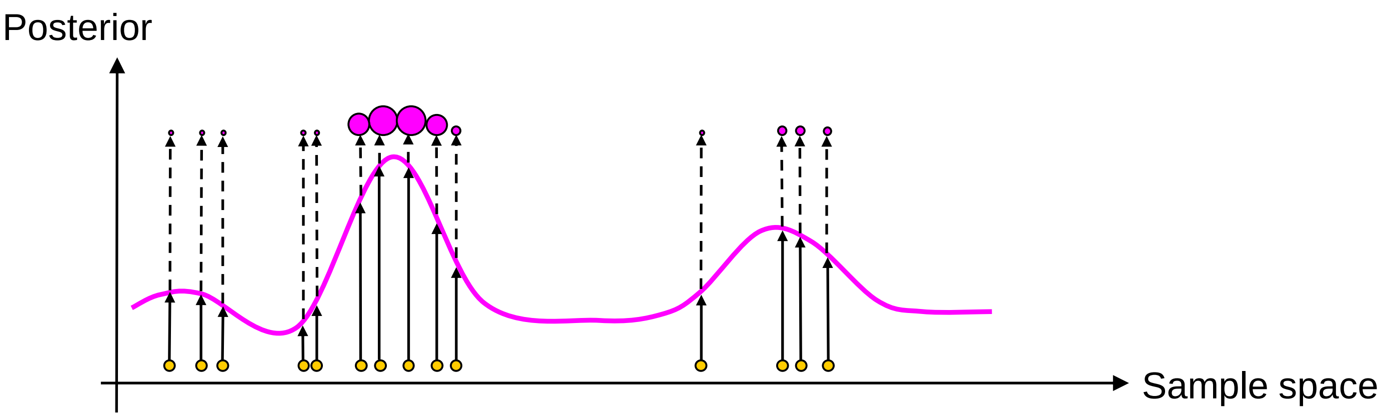
- Idea: Represent state distribution non-parametrically
  - Prediction: Sample points from prior density for the state, $P(X)$
  - Correction: Weight the samples according to $P(Y|X)$

$$P(X_t \mid y_0, \ldots, y_t) = \frac{P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1})}{\int P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1}) dX_t}$$

Slide credit: Svetlana Lazebnik

Particle Filtering

- Many variations, one general concept:
  - *Represent the posterior pdf by a set of randomly chosen weighted samples (particles)*



  - Randomly Chosen = Monte Carlo (MC)
  - As the number of samples become very large – the characterization becomes an equivalent representation of the true pdf.

Slide adapted from Michael Rubinstein

Particle filtering

- Compared to Kalman Filters and their extensions
  - Can represent any arbitrary distribution
  - Multimodal support
  - Keep track of as many hypotheses as there are particles
  - Approximate representation of complex model rather than exact representation of simplified model

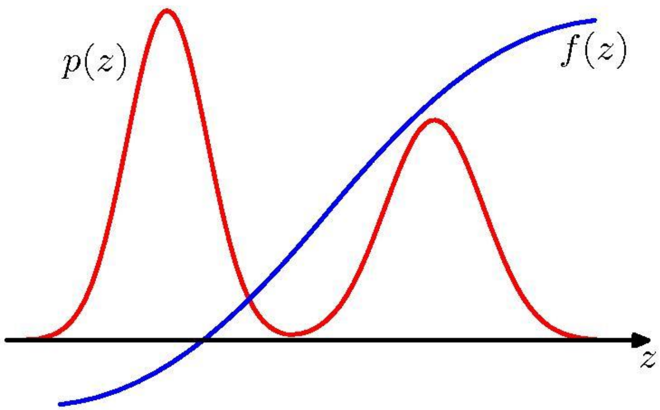
- The basic building-block: *Importance Sampling*

Slide adapted from Michael Rubinstein

Background: Monte-Carlo Sampling

- Objective:
  - Evaluate expectation of a function $f(\mathbf{z})$ w.r.t. a probability distribution $p(\mathbf{z})$.

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$



- Monte Carlo Sampling idea
  - Draw L independent samples $\mathbf{z}^{(l)}$ with l = 1,…,L from $p(\mathbf{z})$.
  - This allows the expectation to be approximated by a finite sum

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{l})$$

  - As long as the samples $\mathbf{z}^{(l)}$ are drawn independently from $p(\mathbf{z})$, then

$$\mathbb{E}[\hat{f}] = \mathbb{E}[f]$$

$\Rightarrow$ Unbiased estimate, independent of the dimension of $\mathbf{z}$!

Slide adapted from Bernt Schiele

Image source: C.M. Bishop, 2006

Monte Carlo Integration

- We can use the same idea for computing integrals
  - Assume we are trying to estimate a complicated integral of a function f over some domain D:

$$F = \int_D f(\vec{x})d\vec{x}$$

  - Also assume there exists some PDF p defined over D. Then

$$F = \int_D f(\vec{x})d\vec{x} = \int_D \frac{f(\vec{x})}{p(\vec{x})} p(\vec{x})d\vec{x}$$

  - For any pdf p over D, the following holds

$$\int_D \frac{f(\vec{x})}{p(\vec{x})} p(\vec{x})d\vec{x} = E\left[\frac{f(\vec{x})}{p(\vec{x})}\right], x \sim p$$

Slide adapted from Michael Rubinstein

Monte Carlo Integration

- Idea (cont'd)
  - Now, if we have i.i.d random samples $x_1,\ldots,\ x_N$ sampled from p, then we can approximate the expectation

$$E\left[\frac{f(\vec{x})}{p(\vec{x})}\right]$$

  - by

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\vec{x}_i)}{p(\vec{x}_i)}$$

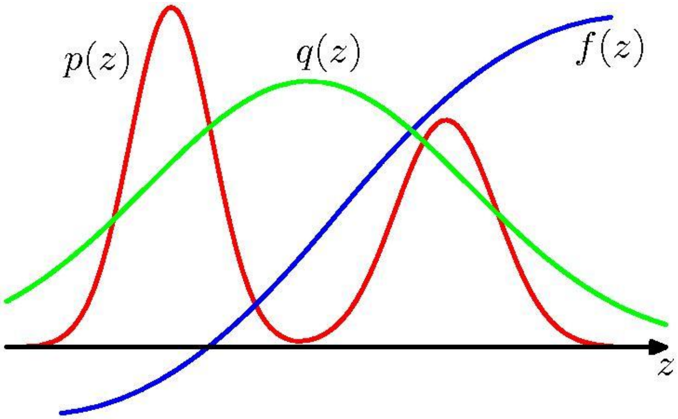
  - Guaranteed by law of large numbers:

$$N \to \infty, F_N \xrightarrow{a.s} E\left[\frac{f(\vec{x})}{p(\vec{x})}\right] = F$$

  - Since it guides sampling, p is often called a proposal distribution.

Slide adapted from Michael Rubinstein

Importance Sampling



- Let's consider an example

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\vec{x}_i)}{p(\vec{x}_i)}$$

  - f/p is the importance weight of a sample.
  - *What can go wrong here?*

- What if $p(x)=0$?
  - If p is very small, then f/p can get arbitrarily large!

$\Rightarrow$ Design p such that f/p is bounded.

  - Effect: get more samples in "important" areas of f, i.e., where f is large.

Slide adapted from Michael Rubinstein

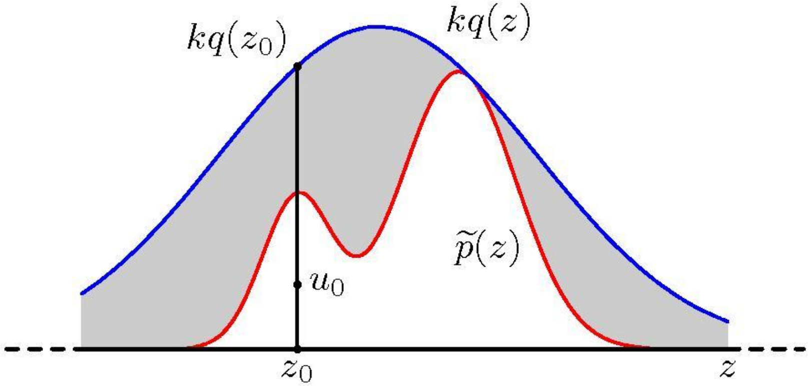
Image source: C.M. Bishop, 2006

Proposal Distributions: Other Uses

- Similar Problem
  - For many distributions, sampling directly from $p(\mathbf{z})$ is difficult.
  - But we can often easily *evaluate* $p(\mathbf{z})$ (up to some normalization factor Z_p):

$$p(\mathbf{z}) = \frac{1}{Z_p}\tilde{p}(\mathbf{z})$$

- Idea
  - Take some simpler distribution $q(\mathbf{z})$ as proposal distribution from which we can draw samples and which is non-zero.



Slide adapted from Bernt Schiele

Image source: C.M. Bishop, 2006

Background: Importance Sampling

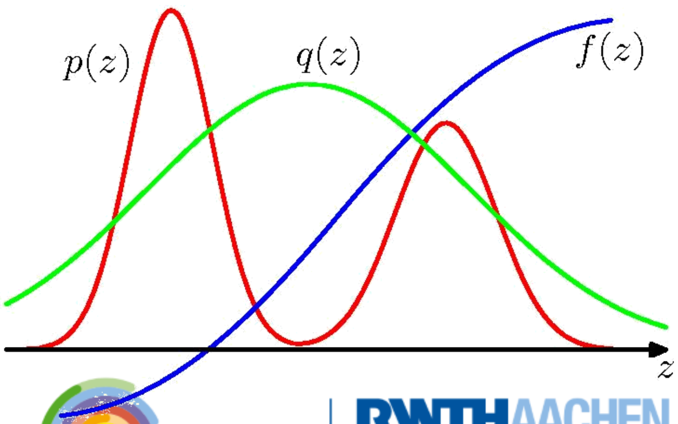
- Idea
  - Use a proposal distribution $q(\mathbf{z})$ from which it is easy to draw samples and which is close in shape to f.
  - Express expectations in the form of a finite sum over samples $\{\mathbf{z}^{(l)}\}$ drawn from $q(\mathbf{z})$.

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)})$$

  - with importance weights

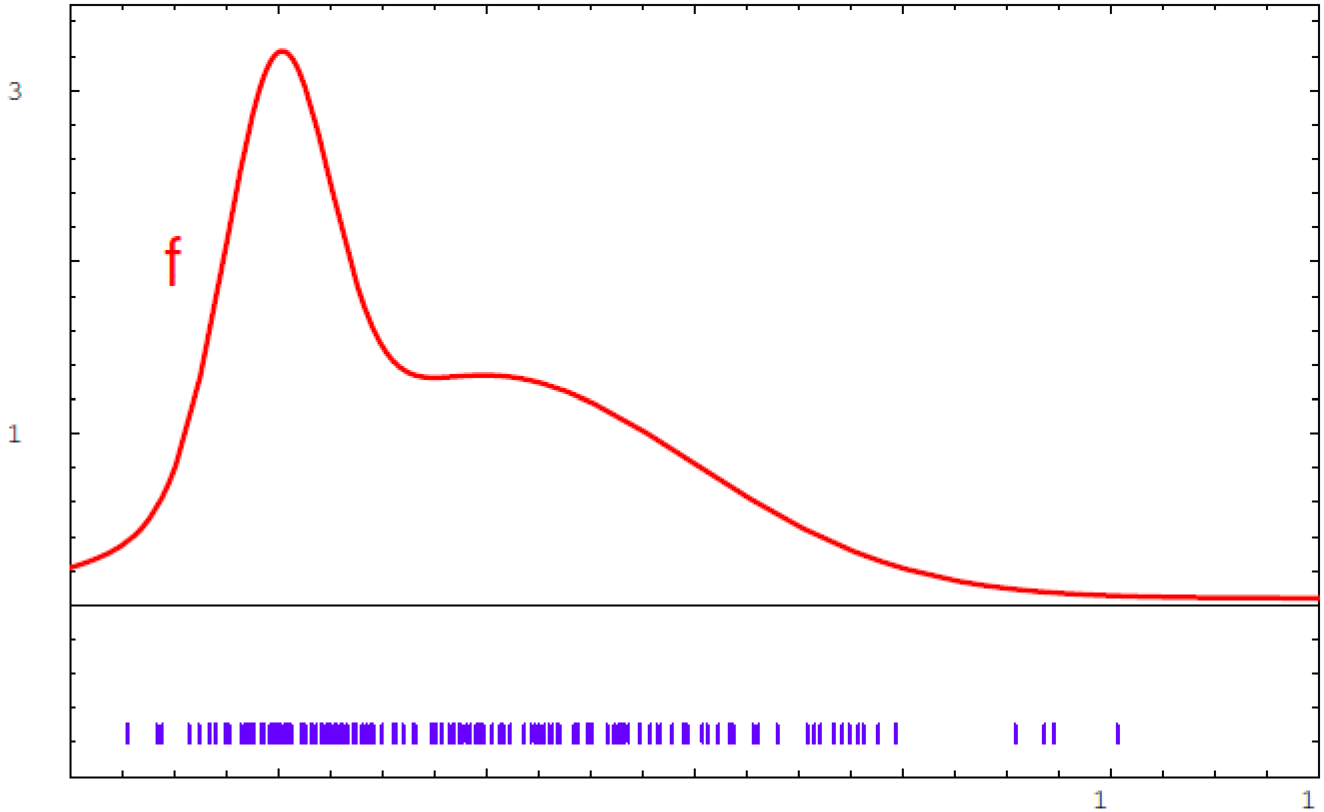
$$r_l = \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}$$



Slide adapted from Bernt Schiele

Image source: C.M. Bishop, 2006

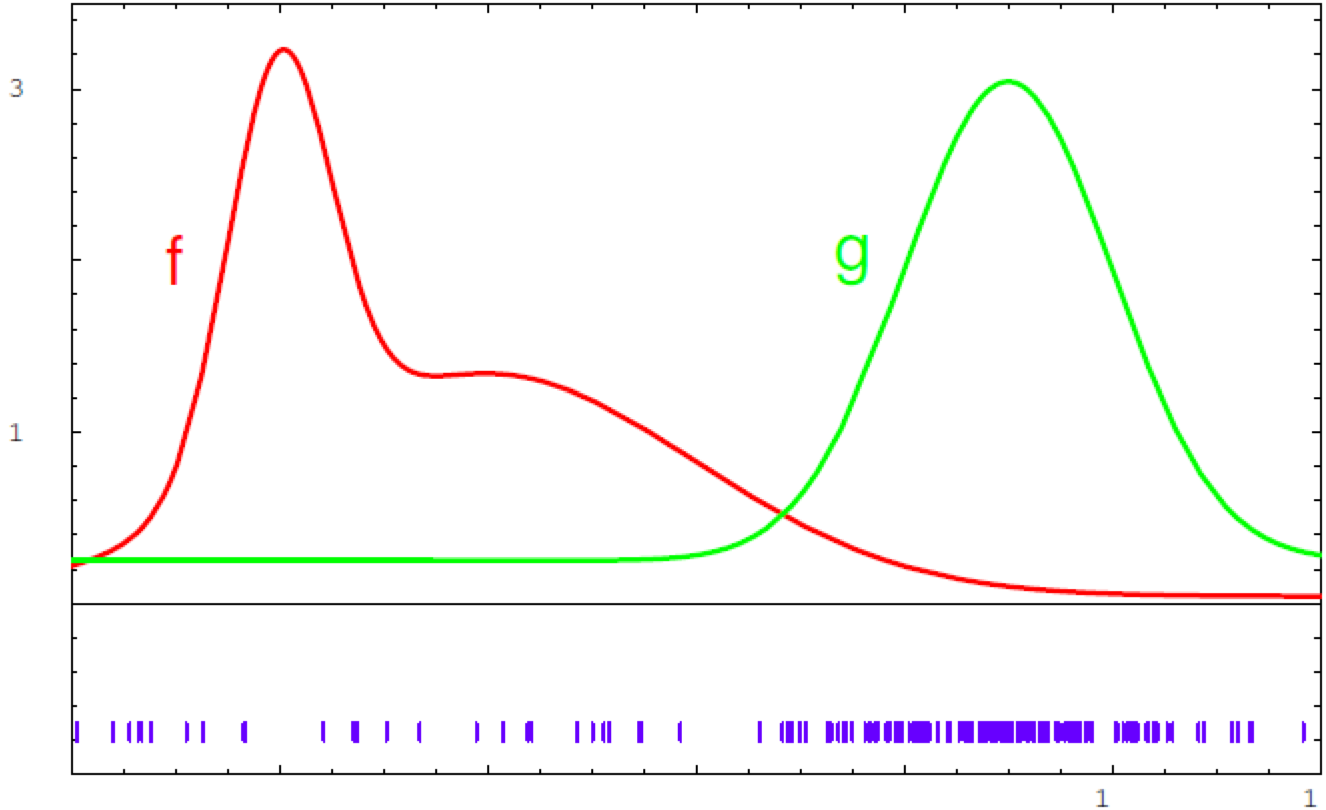
Illustration of Importance Factors



- Goal: Approximate target density f

Figure source: Thrun, Burgard, Fox

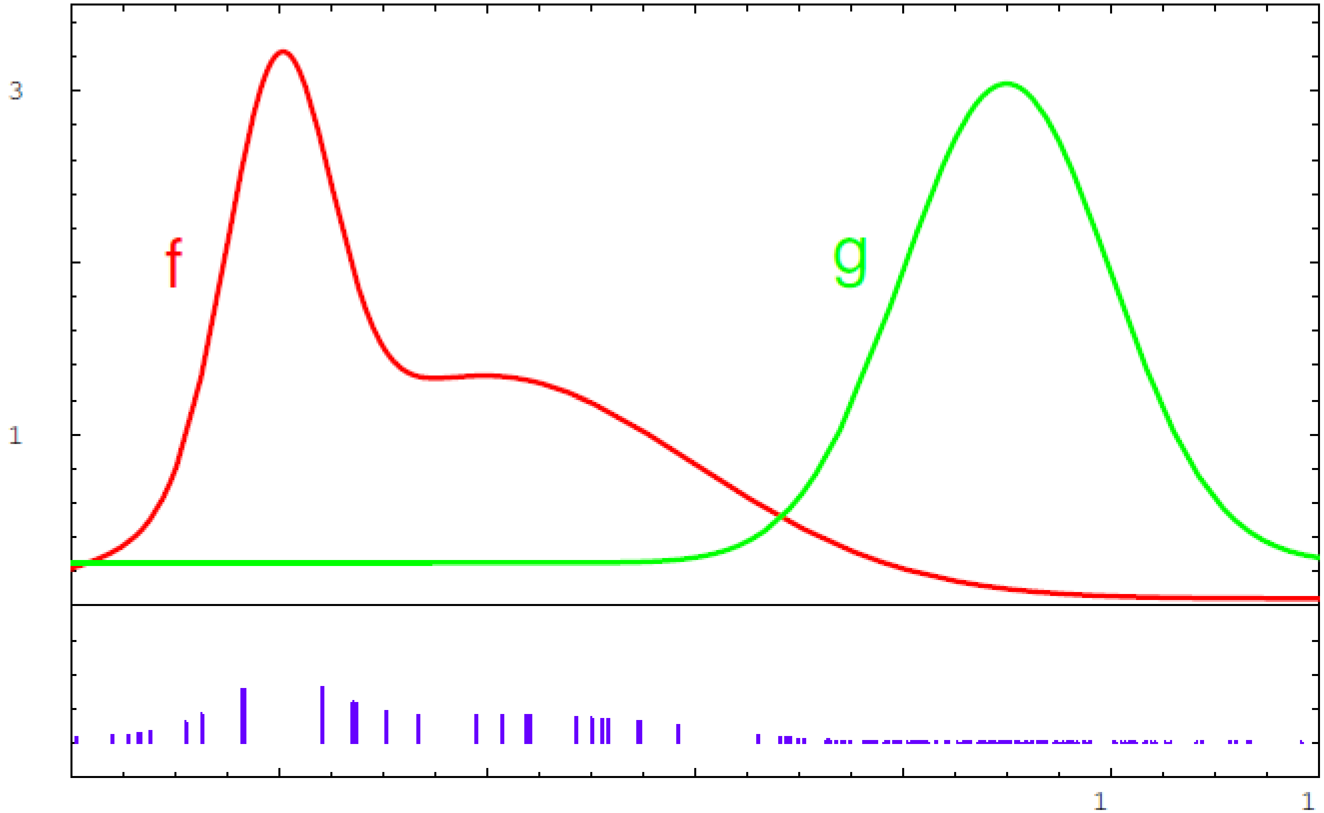
Illustration of Importance Factors



- Goal: Approximate target density f
  - Instead of sampling from f directly, we can only sample from g.

Figure source: Thrun, Burgard, Fox

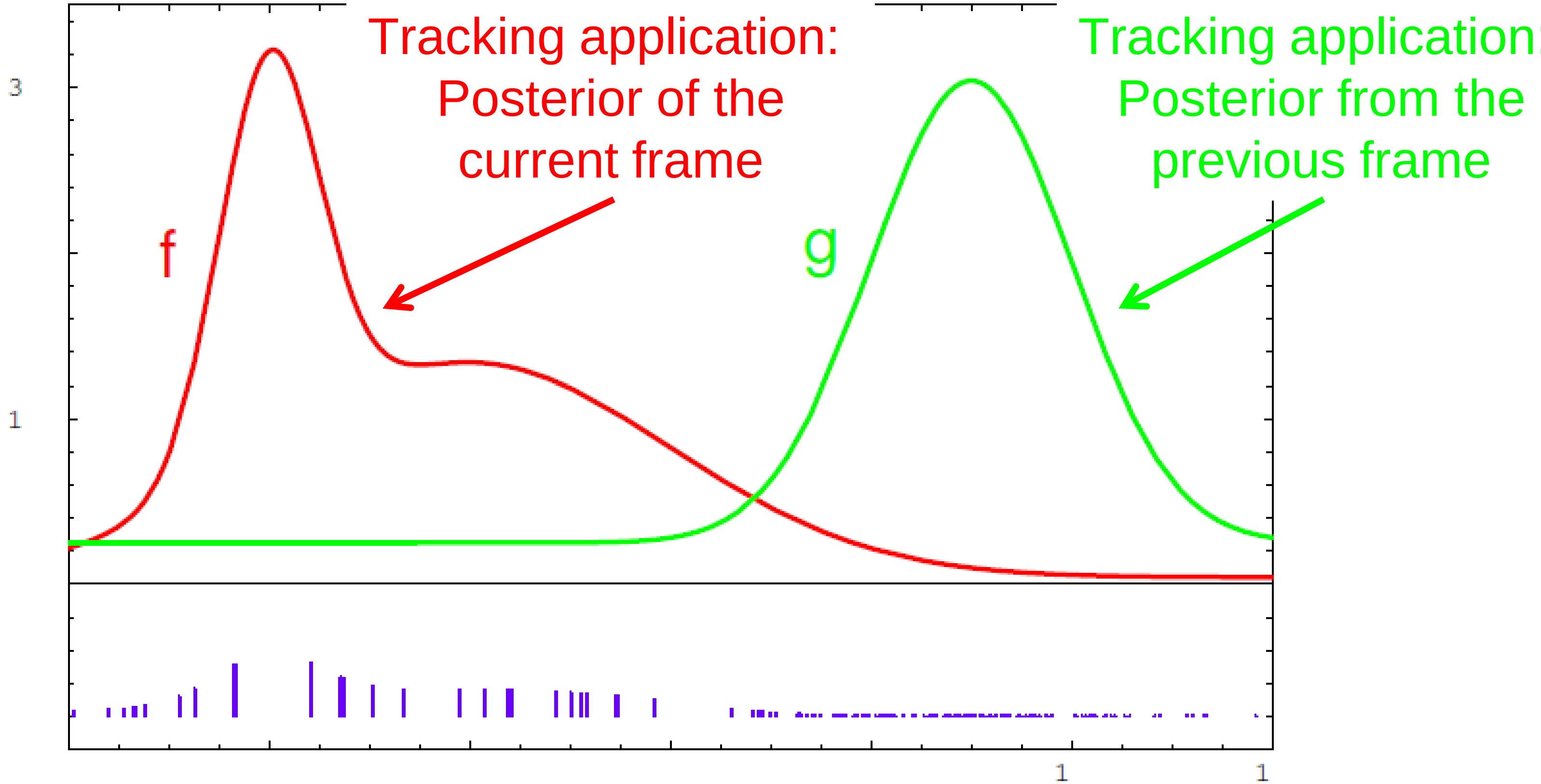
Illustration of Importance Factors



- Goal: Approximate target density f
  - Instead of sampling from f directly, we can only sample from g.
  - A sample of f is obtained by attaching the weight f/g to each sample $\mathbf{x}$

Figure source: Thrun, Burgard, Fox

Illustration of Importance Factors



- Goal: Approximate target density f
  - Instead of sampling from f directly, we can only sample from g.
  - A sample of f is obtained by attaching the weight f/g to each sample $\mathbf{x}$

Figure source: Thrun, Burgard, Fox

Importance Sampling for Bayesian Estimation

$$\mathbb{E}[f(X)] = \int_X f(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$
$$= \int_X f(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$

- Applying Importance Sampling
  - Characterize the posterior pdf using a set of samples (particles) and their weights

$$\left\{\mathbf{x}_{0:t}^i, w_t^i\right\}_{i=1}^N$$

  - Then the joint posterior is approximated by

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \sum_{i=1}^N w_t^i \delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^i)$$

Slide adapted from Michael Rubinstein

Importance Sampling for Bayesian Estimation

$$\mathbb{E}[f(X)] = \int_X f(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$
$$= \int_X f(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$

- Applying Importance Sampling
  - Draw the samples from the importance density $q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ with importance weights

    $$w_t^i \propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}$$

  - Sequential update (after some calculation)
    - Particle update $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$
    - Weight update $w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i) p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$

Slide adapted from Michael Rubinstein

Sequential Importance Sampling Algorithm

$$\textbf{function}\ \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^{N}\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^{N}, \mathbf{y}_t\right]$$

$\eta = 0$ — Initialize

$\textbf{for}\ \ i = 1{:}N$

$\quad \mathbf{x}_t^i \sim q(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$\quad w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t | \mathbf{x}_t^i) p(\mathbf{x}_t^i | \mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\quad \eta = \eta + w_t^i$ — Update norm. factor

$\textbf{end}$

$\textbf{for}\ \ i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — Normalize weights

$\textbf{end}$

Slide adapted from Michael Rubinstein

Sequential Importance Sampling Algorithm

function $\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\quad \mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$\quad w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\quad \eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — Normalize weights

end

For a concrete algorithm, we need to define the importance density $q(.|.)$!

Slide adapted from Michael Rubinstein

Choice of Importance Density

- Most common choice
  - Transitional prior

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$$

  - With this choice, the weight update reduces to

$$\begin{aligned} w_t^i &= w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)} \\ &= w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)\cancel{p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)}}{\cancel{p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)}} \\ &= w_{t-1}^i p(\mathbf{y}_t|\mathbf{x}_t^i) \end{aligned}$$

SIS Algorithm with Transitional Prior

$$\textbf{function} \;\; \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$\eta = 0$ — Initialize

$\textbf{for} \;\; i = 1{:}N$

$$\mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$$ Sample from proposal pdf

$$w_t^i = w_{t-1}^i p(\mathbf{y}_t | \mathbf{x}_t^i)$$ Update weights

$$\eta = \eta + w_t^i$$ Update norm. factor

end

$\textbf{for} \;\; i = 1{:}N$

$$w_t^i = w_t^i / \eta$$ Normalize weights

end

Slide adapted from Michael Rubinstein

Implementation of Sampling Step

$$\textbf{function}\ \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$\eta = 0$ — Initialize

for $i = 1{:}N$

Draw ε_t^i *from noise distribution*

$\mathbf{x}_t^i = \mathbf{g}\left(\mathbf{x}_{t-1}^i\right) + \varepsilon_t^i$ — Sample from proposal pdf

$w_t^i = w_{t-1}^i p(\mathbf{y}_t | \mathbf{x}_t^i)$ — Update weights

$\eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$w_t^i = w_t^i / \eta$ — Normalize weights

end

Slide adapted from Michael Rubinstein

The Degeneracy Phenomenon

- Unavoidable problem with SIS
  - After a few iterations, most particles have negligible weights.
  - Large computational effort for updating particles with very small contribution to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

- Measure of degeneracy
  - Effective sample size

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2}$$

  - Uniform: $N_{eff} = N$
  - Severe degeneracy: $N_{eff} = 1$

Slide adapted from Michael Rubinstein

Resampling

- Idea
  - Eliminate particles with low importance weights and increase the number of particles with high importance weight.

$$\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N \rightarrow \left\{\mathbf{x}_t^{i*}, \frac{1}{N}\right\}_{i=1}^N$$

  - The new set is generated by sampling with replacement from the discrete representation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ such that

$$Pr\left\{\mathbf{x}_t^{i*} = \mathbf{x}_t^j\right\} = w_t^j$$

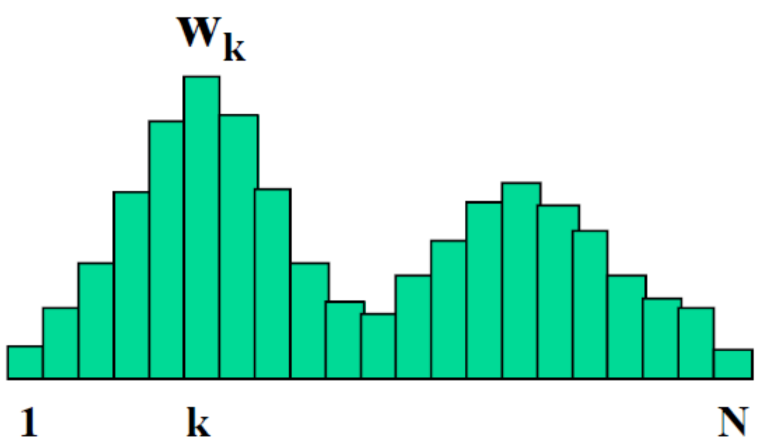
Slide adapted from Michael Rubinstein

Resampling

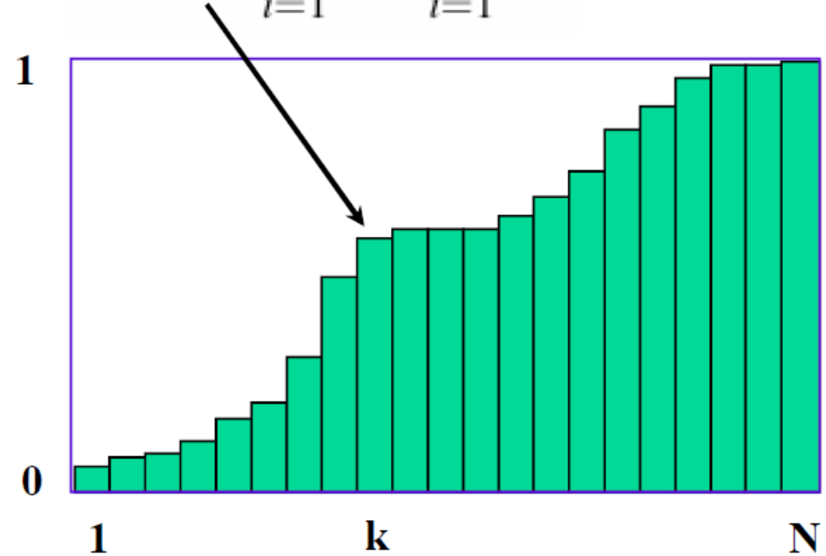
- How to do that in practice?
  - We want to resample $\left\{\mathbf{x}_t^i\right\}_{i=1}^N$ from the discrete pdf given by the weighted samples $\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N$
  - I.e., we want to draw N new samples $\left\{\mathbf{x}_t^i\right\}_{i=1}^N$ with replacement where the probability of drawing $\mathbf{x}_t^j$ is given by w_t^j.
- There are many algorithms for this
  - We will look at two simple algorithms here...

Inverse Transform Sampling

- Idea
  - It is easy to sample from a discrete distribution using the cumulative distribution function $F(x) = p(X \leq x)$



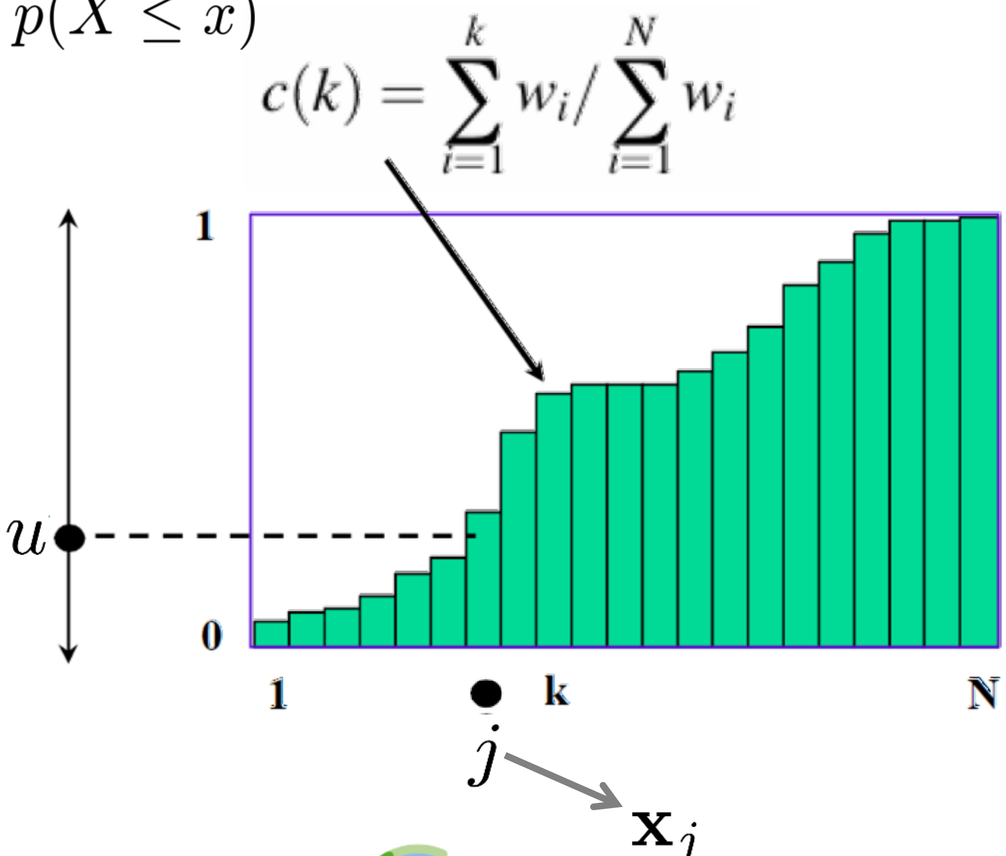
$$c(k) = \sum_{i=1}^{k} w_i / \sum_{i=1}^{N} w_i$$



Slide adapted from Robert Collins

Inverse Transform Sampling

- Idea
  - It is easy to sample from a discrete distribution using the cumulative distribution function $F(x) = p(X \leq x)$
- Procedure
  1. Generate uniform u in the range $[0,1]$.
  2. Visualize a horizontal line intersecting the bars.
  3. If index of intersected bar is j, output new sample $\mathbf{x}_j$.



Slide adapted from Robert Collins

More Efficient Approach

- From Arulampalam paper:

Algorithm 2: Resampling Algorithm

$[\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = \text{RESAMPLE}\ [\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

- Initialize the CDF: $c_1 = 0$
- FOR $i = 2: N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
- END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
- FOR $j = 1: N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j > c_i$
  - * $i = i + 1$
  - END WHILE
  - Assign sample: $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$
  - Assign weight: $w_k^j = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR

Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf. This is $\mathcal{O}(N)$!

Slide adapted from Robert Collins

Generic Particle Filter

$$\textbf{function} \quad \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = PF\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$$\textit{Apply SIS filtering} \quad \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$$\textit{Calculate } N_{eff}$$

$$\textbf{if} \quad N_{eff} < N_{thr}$$

$$\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = RESAMPLE\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right]$$

$$\textbf{end}$$

- We can also apply resampling selectively
  - Only resample when it is needed, i.e., N_{eff} is too low.
  - $\Rightarrow$ Avoids drift when the tracked state is stationary.

Slide adapted from Michael Rubinstein

Sampling-Importance-Resampling (SIR)

$$\textbf{function} \quad [\mathcal{X}_t] = SIR\,[\mathcal{X}_{t-1}, \mathbf{y}_t]$$

$$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$$ Initialize

$$\textbf{for} \quad i = 1{:}N$$

$$\quad Sample \;\; \mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$$ Generate new samples

$$\quad w_t^i = p(\mathbf{y}_t | \mathbf{x}_t^i)$$ Update weights

$$\textbf{end}$$

$$\textbf{for} \quad i = 1{:}N$$

$$\quad Draw\ i\ with\ probability \propto w_t^i$$

$$\quad Add\ \mathbf{x}_t^i\ \ to\ \mathcal{X}_t$$

Resample

$$\textbf{end}$$

Slide adapted from Michael Rubinstein

Other Variant of the Algorithm

$$\textbf{function} \quad [\mathcal{X}_t] = SIR\,[\mathcal{X}_{t-1}, \mathbf{y}_t]$$

$$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$$

$\textbf{for} \quad i = 1{:}N$

$$Sample \;\; \mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$$

$$w_t^i = p(\mathbf{y}_t | \mathbf{x}_t^i)$$

$\textbf{end}$

$\textbf{for} \quad i = 1{:}N$

$$Draw\; i\; with\; probability$$

$$Add\;\; \mathbf{x}_t^i \;\; to\; \mathcal{X}_t$$

$\textbf{end}$

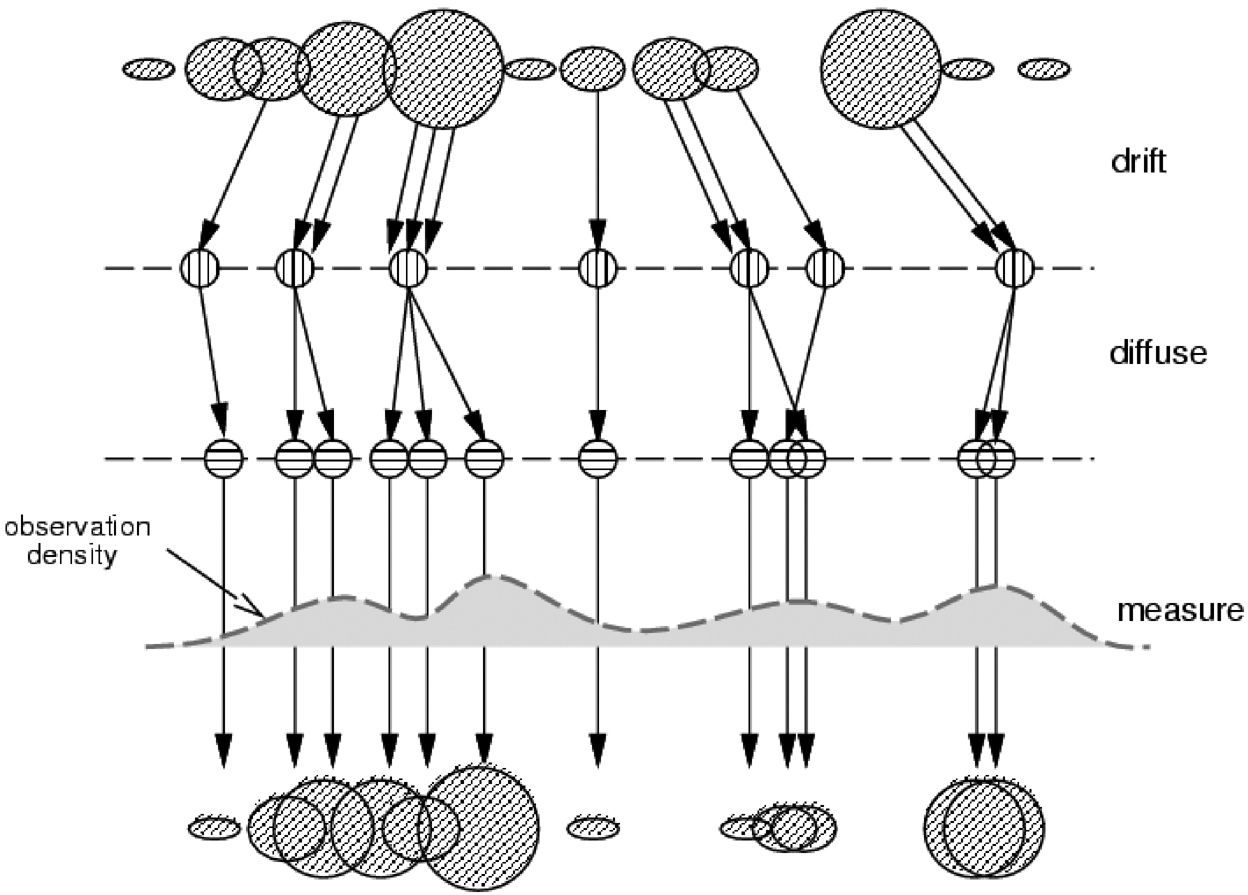
Important property:

Particles are distributed according to pdf from previous time step.

Particles are distributed according to posterior from this time step.

Slide adapted from Michael Rubinstein

Recap: Condensation Algorithm



Start with weighted samples from previous time step

Sample and shift according to dynamics model

Spread due to randomness; this is predicted density $P(X_t|Y_{t-1})$

Weight the samples according to observation density

Arrive at corrected density estimate $P(X_t|Y_t)$

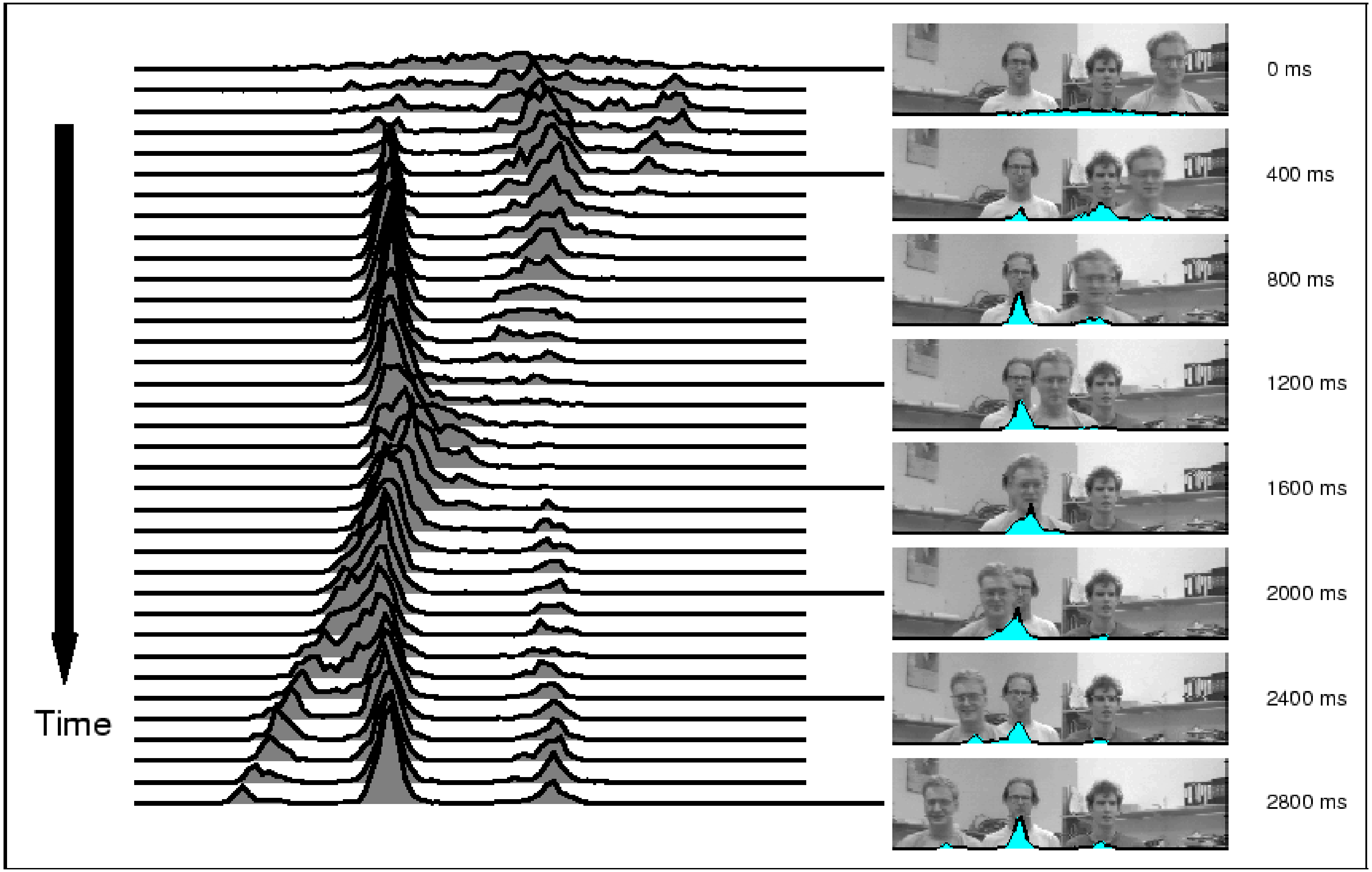
M. Isard and A. Blake, CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

Slide credit: Svetlana Lazebnik

Particle Filtering – Visualization

Code and video available from
http://www.robots.ox.ac.uk/~misard/condensation.html

Particle Filtering Results



http://www.robots.ox.ac.uk/~misard/condensation.html

Particle Filtering Results

- Some more examples

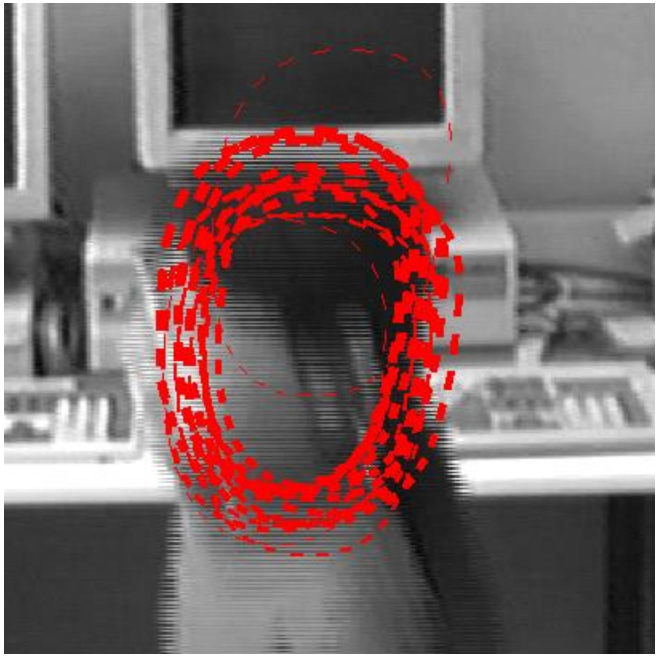
http://www.robots.ox.ac.uk/~misard/condensation.html

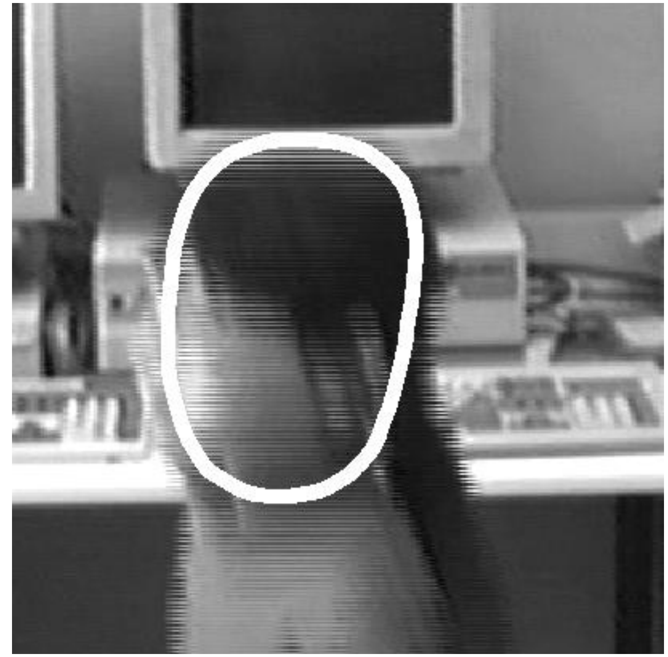
Videos from Isard & Blake

Sidenote: Obtaining a State Estimate

- Note that there's no explicit state estimate maintained, just a "cloud" of particles
- Can obtain an estimate at a particular time by querying the current particle set
- Some approaches
  - "Mean" particle
    - Weighted sum of particles
    - Confidence: inverse variance
  - Really want a mode finder—mean of tallest peak

Condensation: Estimating Target State





From Isard & Blake, 1998

State samples
(thickness proportional to weight)

Mean of weighted
state samples

Summary: Particle Filtering

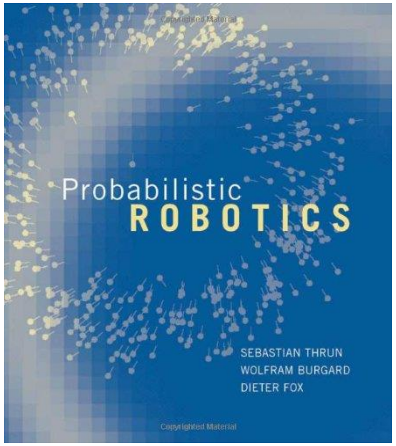
- Pros:
  - Able to represent arbitrary densities
  - Converging to true posterior even for non-Gaussian and nonlinear system
  - Efficient: particles tend to focus on regions with high probability
  - Works with many different state spaces
    - E.g. articulated tracking in complicated joint angle spaces
  - Many extensions available

Summary: Particle Filtering

- Cons / Caveats:
  - #Particles is important performance factor
    - Want as few particles as possible for efficiency.
    - But need to cover state space sufficiently well.
  - Worst-case complexity grows exponentially in the dimensions
  - Multimodal densities possible, but still single object
    - Interactions between multiple objects require special treatment.
    - Not handled well in the particle filtering framework (state space explosion).

References and Further Reading

- A good description of Particle Filters can be found in Ch.4.3 of the following book
  - S. Thrun, W. Burgard, D. Fox. Probabilistic Robotics. MIT Press, 2006.



- A good tutorial on Particle Filters
  - M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. In *IEEE Transactions on Signal Processing*, Vol. 50(2), pp. 174-188, 2002.

- The CONDENSATION paper
  - M. Isard and A. Blake, CONDENSATION - conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998