Non Parametric Filters

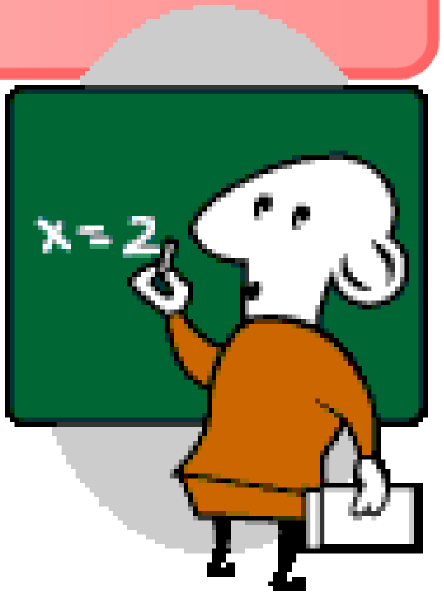
Histogram and Particle Filters

Department of Electrical and Electronics Engineering

Dr. Afşar Saranlı

Lecture slides heavily use material from the textbook and
Sebastian Thrun, Lecture Slides; http://www.probabilistic-robotics.org/

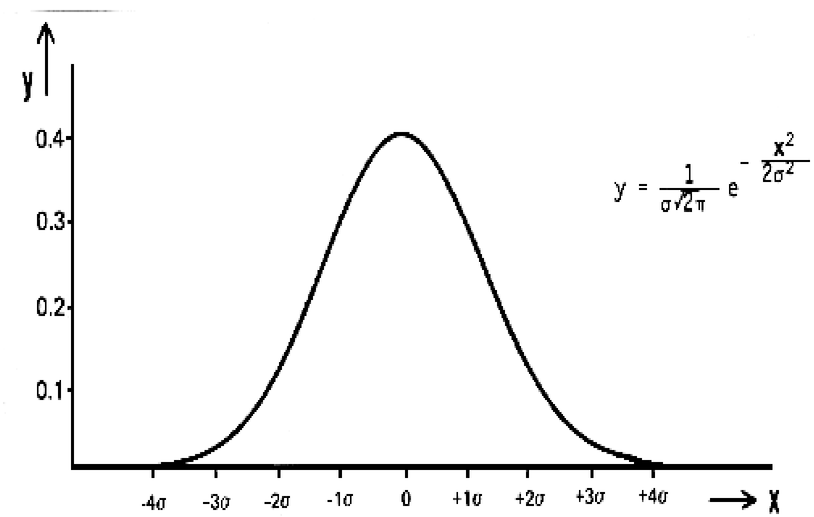
What we will discuss

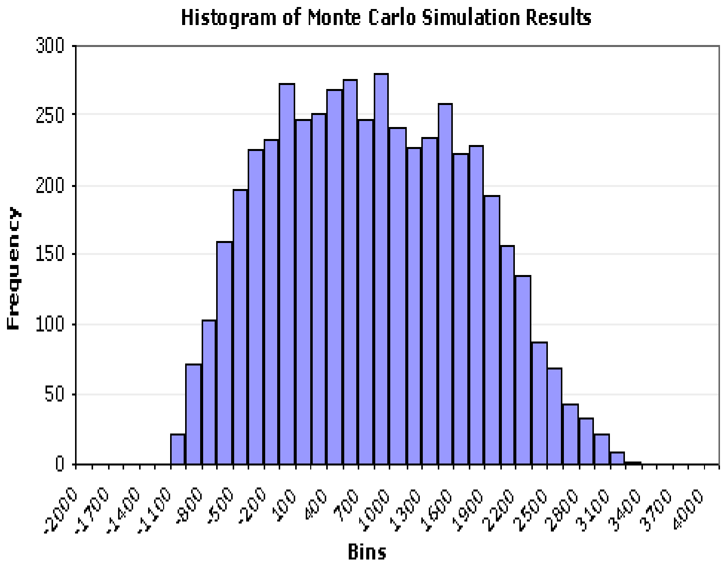


- Discuss the merits of non-parametric density representations,
- Remember histograms and introduce the *Histogram Filters*,
- Introduce *Particle Filters*,
- Introduce *resampling* and *Importance Sampling,*
- Review practical issues and properties of particle filters,
- Try to conclude…

Non-Parametric Representations

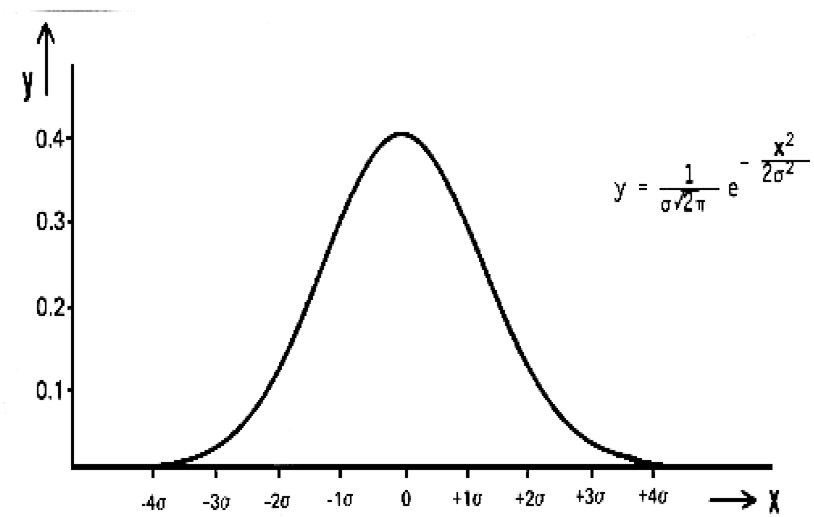
- **What is a non-parametric density representation?**
- **Finite set of values (samples) instead of a parametric closed form expression**
- **Example: *Gaussian* versus its *histogram***

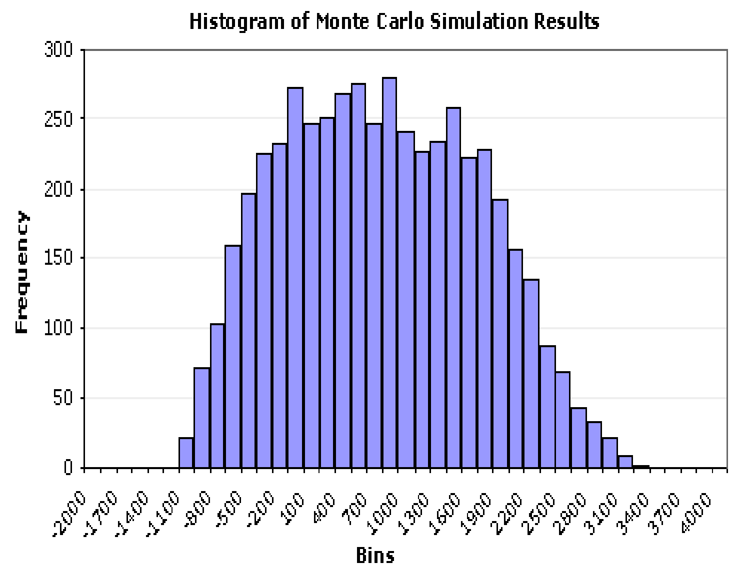




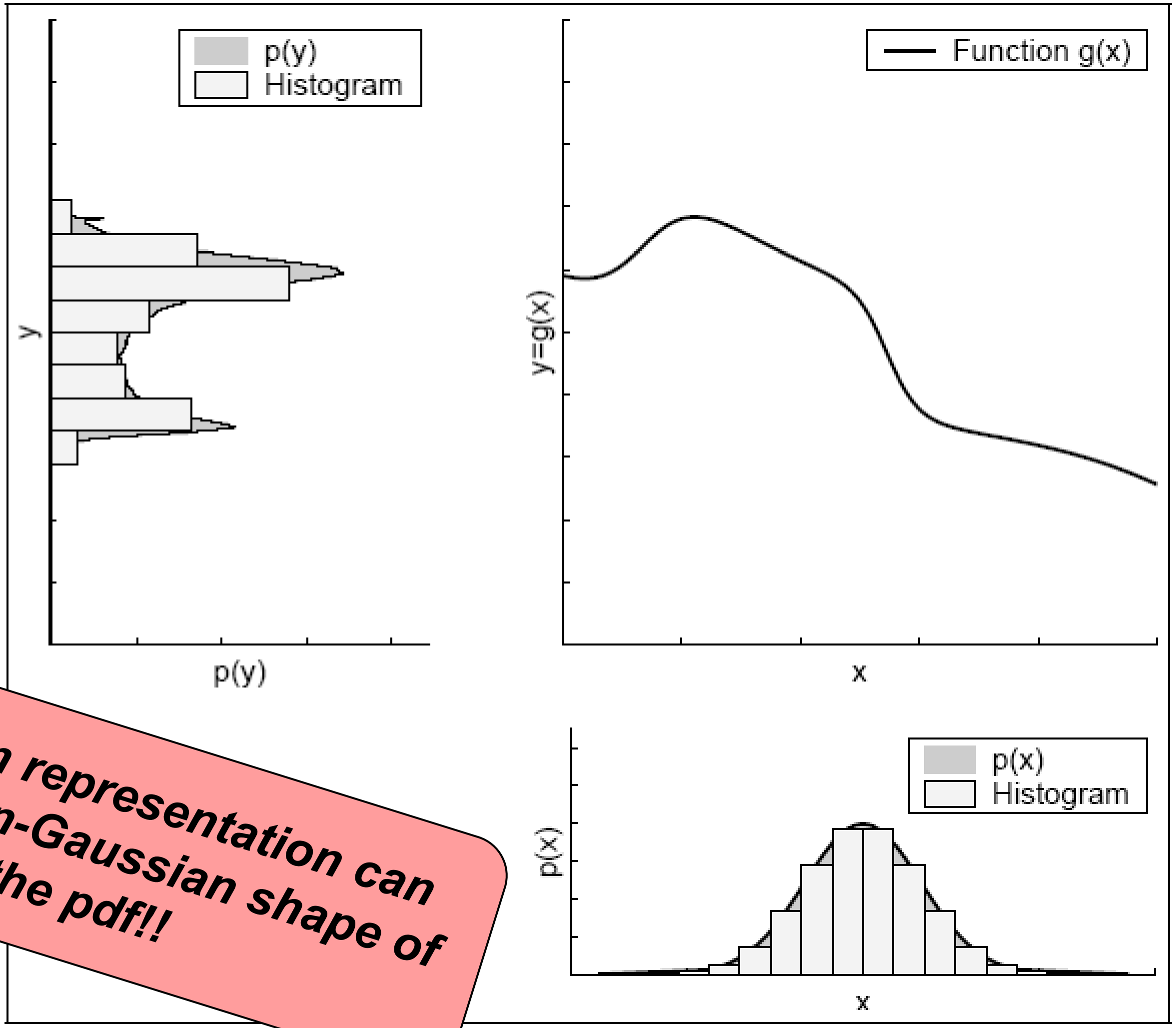
Why Non-Parametric?

- **Why would we want to use non-parametric?**
- **Why would we not want to use non-parametric?**
- **Quality? Computational Complexity?**
- **As N→∞, non-parametric rep. converges uniformly to the true density.**





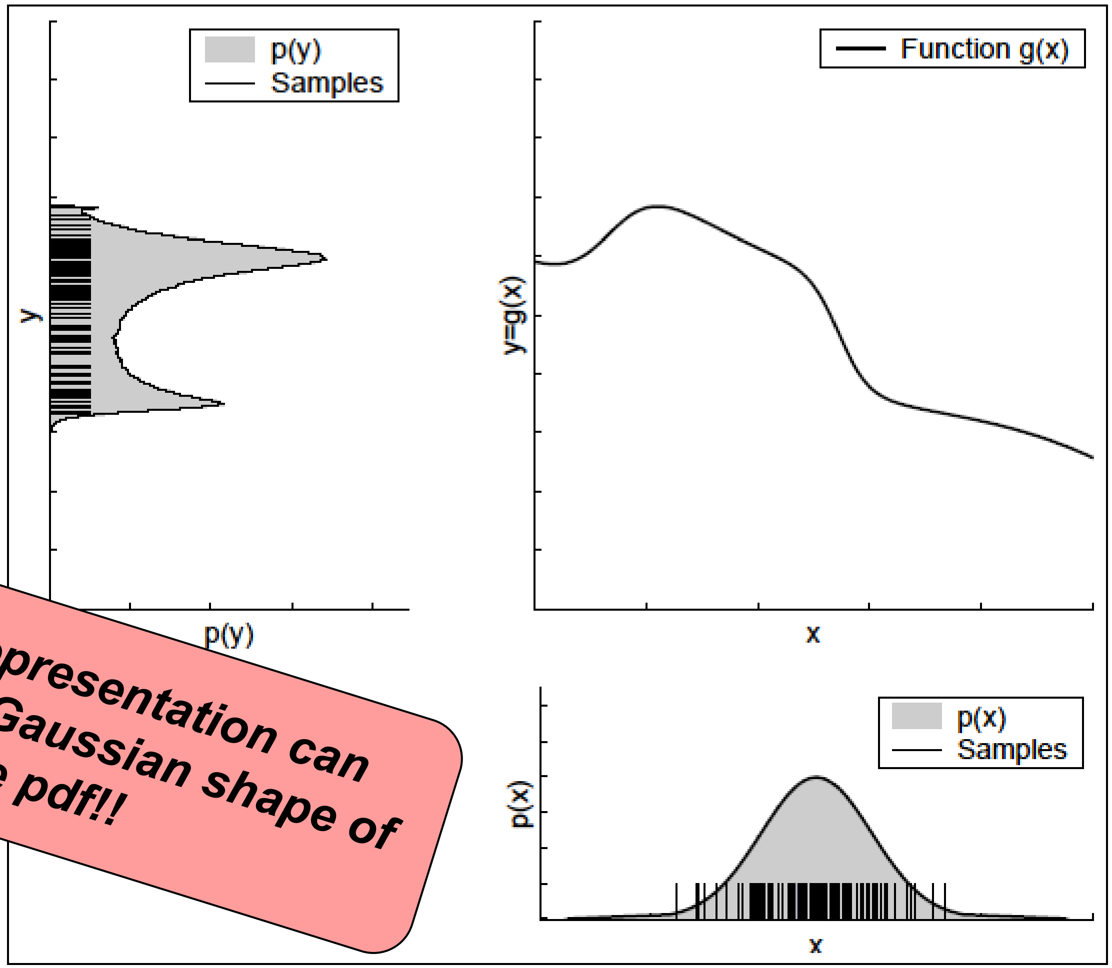
What do you see?



Histogram representation can capture Non-Gaussian shape of the pdf!!

Other Non-Parametric Reps

- **Particles (Sample based) Representation**



"Partile" representation can capture Non-Gaussian shape of the pdf!!

Non-Parametric Advantages

- **Both histograms and particle sets have:**
- **No strong parametric assumptions on density (any arbitrary shape can be represented),**
- **Accuracy of the representation can be adjusted as required (by setting *N*)**
- **Results in conceptually much simpler program implementations,**
- **Well suited for *complex multi-modal beliefs* (e.g. in *global localization* with hard data association problems)**

Non-Parametric Disadvantages

- **Simply: *Computational Complexity!!***
- **A naïve implementation can be orders of magnitude more complex than parametric implementations (e.g. Kalman Filters)**
- **Problem becomes compounded if state-space dimension increases. (much larger *N* needed!)**
- **Fortunately:**
  - **Computational complexity can be adapted by adapting number of parameters *N*,**
  - **Complexity and accuracy can be traded off using: *Resource Adaptation*,**
  - **Resource adaptive algorithms very important in robotics and embedded systems**

Question

- **How would we use non-parametric density representations to implement recursive state estimation?**

Use the Bayes Filtering framework in discrete-form

Part1: The Histogram Filter

Part1: The Histogram Filter

- **Remember the generic Bayes Filter:**

Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)**:**

  for all x_t *do*

    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$

    $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$

  endfor

  return $bel(x_t)$

Table 2.1 The general algorithm for Bayes filtering.

The Histogram Filter - DBF

- **Discrete Bayes Filter from the Bayes Filter**

Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$**):**
  for all x_t do
    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
    $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

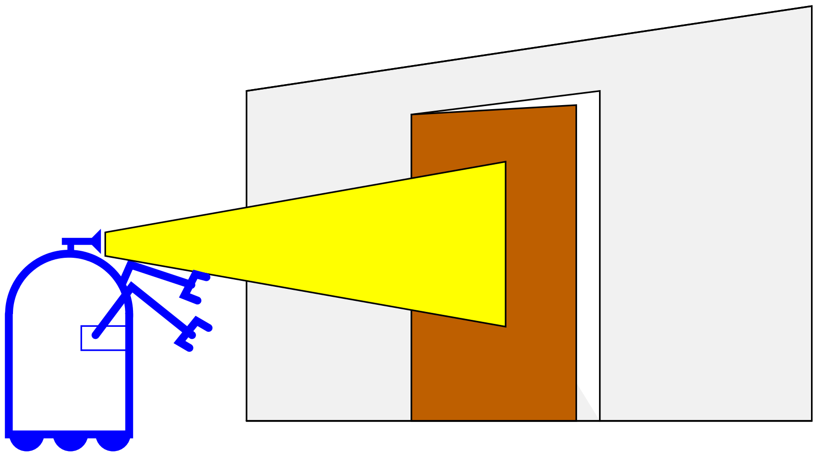
Algorithm Discrete_Bayes_filter($\{p_{k,t-1}\}, u_t, z_t$**):**
  for all k do
    $\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$
    $p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$
  endfor
  return $\{p_{k,t}\}$

The Histogram Filter - Discretization

- **Some problems are already discrete in nature (state of a door, states of a game board),**
- **Some problems are *represented* in discrete form because of the *resolution of interest*. (E.g. occupancy grid maps)**
- **Entirely continuous problems may also be discretized through *various approximations.* (e.g. the orientation of a robot with 5º steps),**
- **Granularity (resolution) of the discretization may be very important (not only for performance but also for proper operation of the filters)**

Example – Estimation of Door State

- **Problem is to estimate a single number!!**
  Prob(door=open|all past states, all past measurements)



Continuous State Space

- **When DBF is applied to a continuous state space, it is called the *Histogram Filter,***
- **Histogram Filter decompose such a space into “bins” through a suitable “partitioning”,**

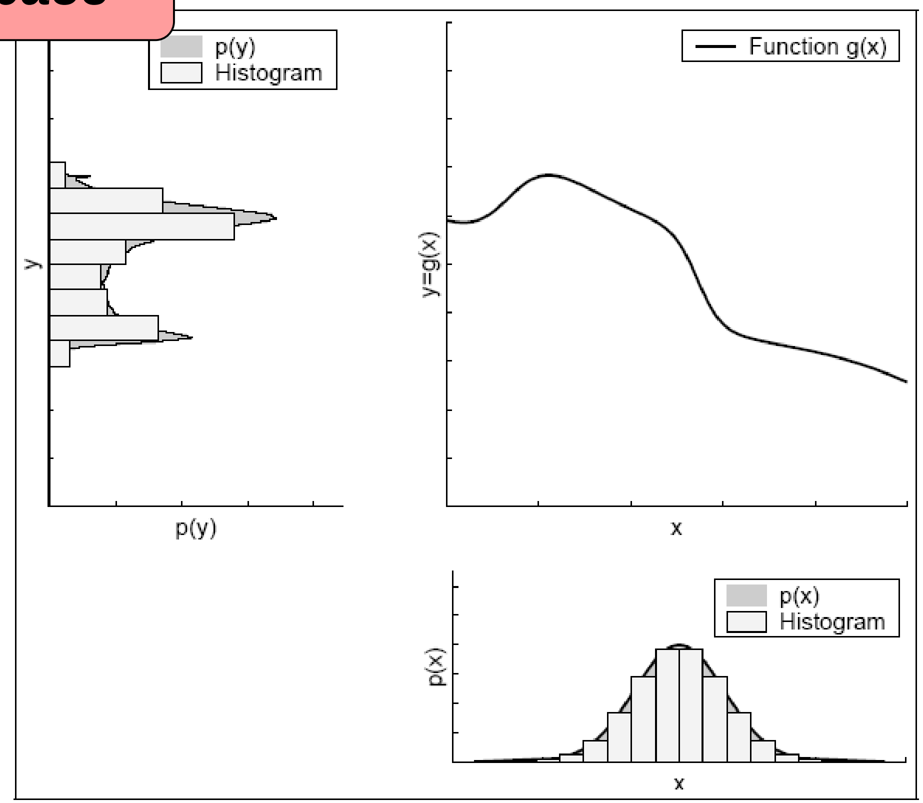
$$\mathrm{dom}(X_t) \quad = \quad \mathbf{x}_{1,t} \cup \mathbf{x}_{2,t} \cup \ldots \mathbf{x}_{K,t}$$

$$\mathbf{x}_{i,t} \cap \mathbf{x}_{k,t} = \emptyset \text{ and } \bigcup_k \mathbf{x}_{k,t} = \mathrm{dom}(X_t)$$

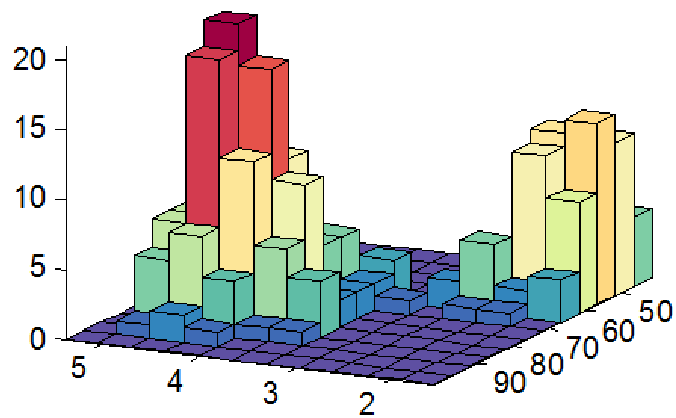
- **Most common partitioning is a multi-dimensional grid representation,**
- **Resulting Pdf approximation is a piecewise constant pdf.**

Example – 1D / 2D spaces

1D space



2D space



Use in Continuous Spaces

- **We might be given the continuous densities,**

$$p(x_t \mid u_t, x_{t-1}) \text{ and } p(z_t \mid x_t)$$

- **These are defined for a continuum of states, not for the discrete “bins”,**

- **How can we discretize the given continuous densities?**

Use in Continuous Spaces

- **Given continuous** $p(x_t \mid u_t, x_{t-1})$ and $p(z_t \mid x_t)$
- **For each bin, we can pick a *representative "mean" state:***

$$\hat{x}_{k,t} = |\mathbf{x}_{k,t}|^{-1} \int_{\mathbf{x}_{k,t}} x_t \, dx_t$$

- ***Then approximate the discrete probability mass functions as:***

There is a correction in the book here.

$$p(z_t \mid \mathbf{x}_{k,t}) \approx p(z_t \mid \hat{x}_{k,t})$$

$$p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \approx \eta \, |\mathbf{x}_{k,t}| \, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})$$

Discrete "probability" is calculated from the continuous "likelihood" value at the mean state which is integrated over the "bin"

Use in Continuous Spaces

- **Then, the Discrete Bayes Filter can be used directly:**

Algorithm Discrete_Bayes_filter($\{p_{k,t-1}\}, u_t, z_t$):

for all k do

$$\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$$

$$p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$$

endfor

return $\{p_{k,t}\}$

$$p(z_t \mid \mathbf{x}_{k,t}) \approx p(z_t \mid \hat{x}_{k,t})$$

$$p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \approx \eta\, |\mathbf{x}_{k,t}|\, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})$$

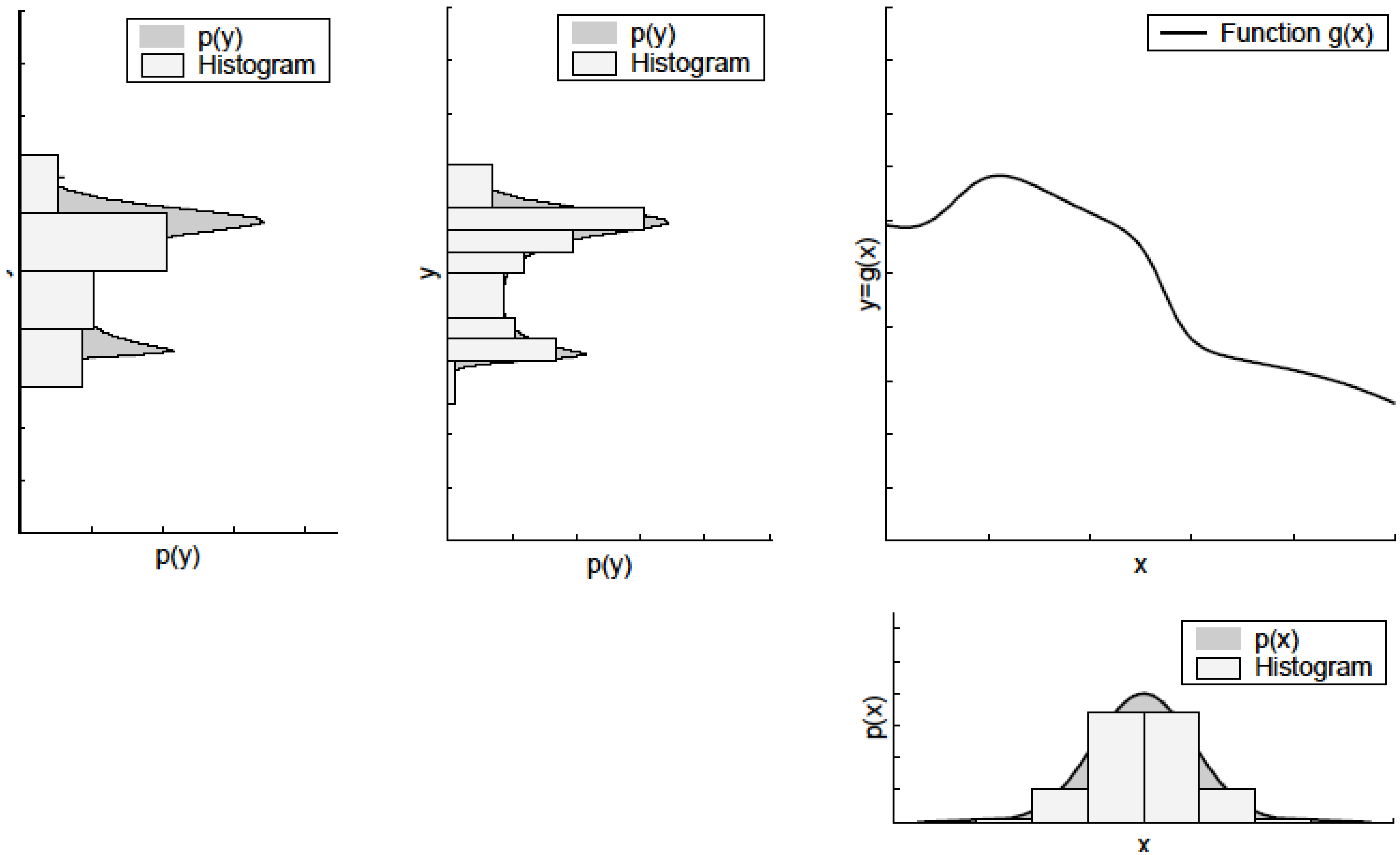
Practical Issues: Decomposition

- Histogram filters can trade-off *accuracy* with *computational complexity* but…
- Desired accuracy may come at a <u>prohibitive computational price</u>!!
- A Naïve uniform grid decomposition with full update may be unusable.
- Some ideas:
  - Density trees,
  - Selective Updating,
  - "Topological" representations,

Decomposition: Density Trees

- **Important example of "dynamic decomposition",**
- **Dynamic techniques adapt to the shape of the posterior being approximated,**
- **Static techniques *easier to implement*, Dynamic techniques *more efficient and hence faster,***
- ***Density Trees:* A recursive decomposition that takes into account the distribution,**
  - ***The more likely a region, the finer the decomposition (more bins) and vice versa.***
  - ***Achieves higher approximation quality with the same computational complexity,***
  - ***OR: Cuts the complexity by orders of magnitude for the same approximation quality***

Dynamic Decomposition: Example

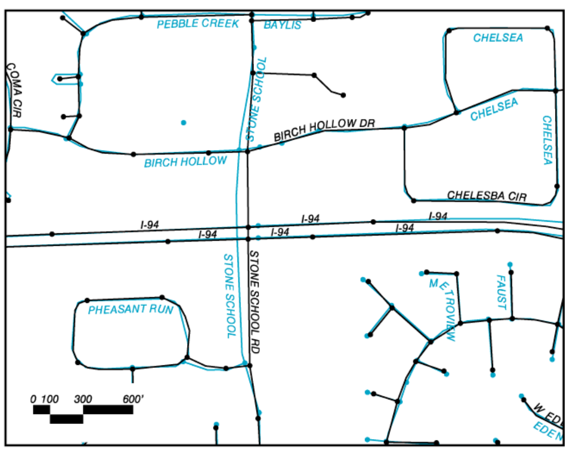


Practical Issues: Selective Update

- **Instead of worrying about the decomposition, worry about the filter updates,**
- **E.g., generate a uniform grid decomposition but…**
- **Update only a fraction of the cells (probabilities) at a time, specifically…**
- **Those that have the posterior probabilities that exceeds a certain threshold.**
- **It can also be viewed as a dynamic decomposition technique,**
- **Can save orders of magnitude in computational complexity. What about storage space?**

Practical Issues: Topological Reps

- **“Metric” vs “Topological” representations,**
- ***Topological*: Coarse, graph-like representations where only *significant places* or *features* are stored,**
- **E.g. corridors, intersections, dead ends,**
- **“Topological” representations usually more efficient but much less precise,**



Special: Binary Bayes w Static State

- **A special case of Discrete Bayes Filter,**
- **Best approximation for certain problems (e.g. occupancy grid maps)**
- **State is static: belief at time *t* is only a function of the measurements:**

$$bel_t(x) = p(x \mid z_{1:t}, u_{1:t}) = p(x \mid z_{1:t})$$

- **An elegant and efficient formulation using the so called "log-odds ratios"**
- **Uses the inverse measurement model $p(x|z_t)$ contrary to our usual model $p(z_t|x)$**

Special: Binary Bayes w Static State

P(x): Prior probability of state x

1: **Algorithm binary_Bayes_filter**(l_{t-1}, z_t):

2: $l_t = l_{t-1} + \log \frac{p(x|z_t)}{1-p(x|z_t)} - \log \frac{p(x)}{1-p(x)}$

3: return l_t

$$l_t(x) = \log \frac{p(x \mid z_{1:t})}{1 - p(x \mid z_{1:t})}$$

All terms in terms of
“Log-odds ratios”

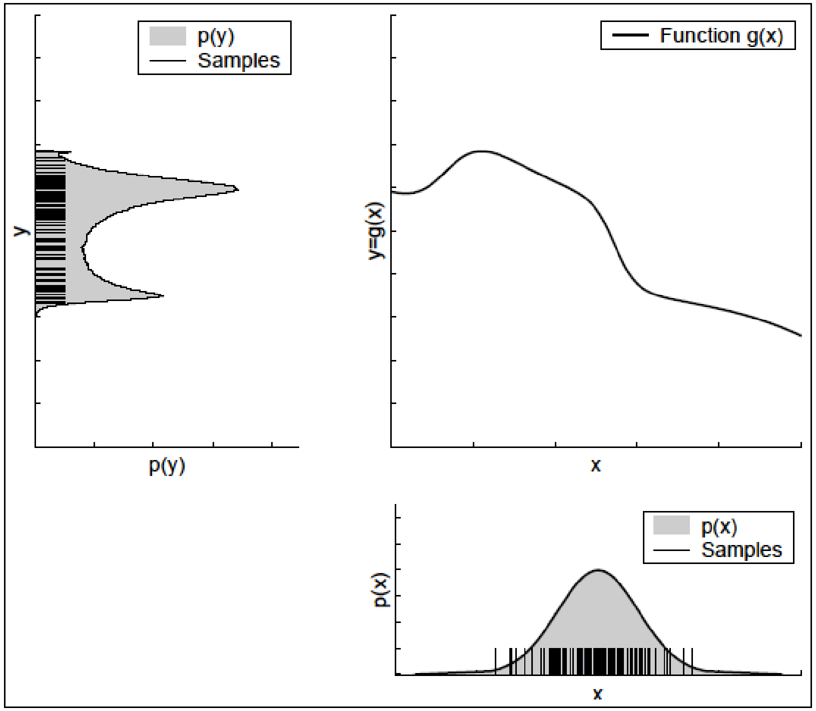
Inverse measurement model
(Typically used when measurements are more complex than the state)

- **This additive form avoids truncation problems with probabilities close to 1 or 0.**

Particle Filters

- **“Particles representation” of density:**
- **Density represented by samples drawn from it**
- **For a Bayes Filter implementation:**

  We need to be able to draw samples from:
  $p(x'|x,u)$ (motion)
  and evaluate:
  $p(z|x)$ (sensor)



- **Usually easier to do!**

Particle Filters

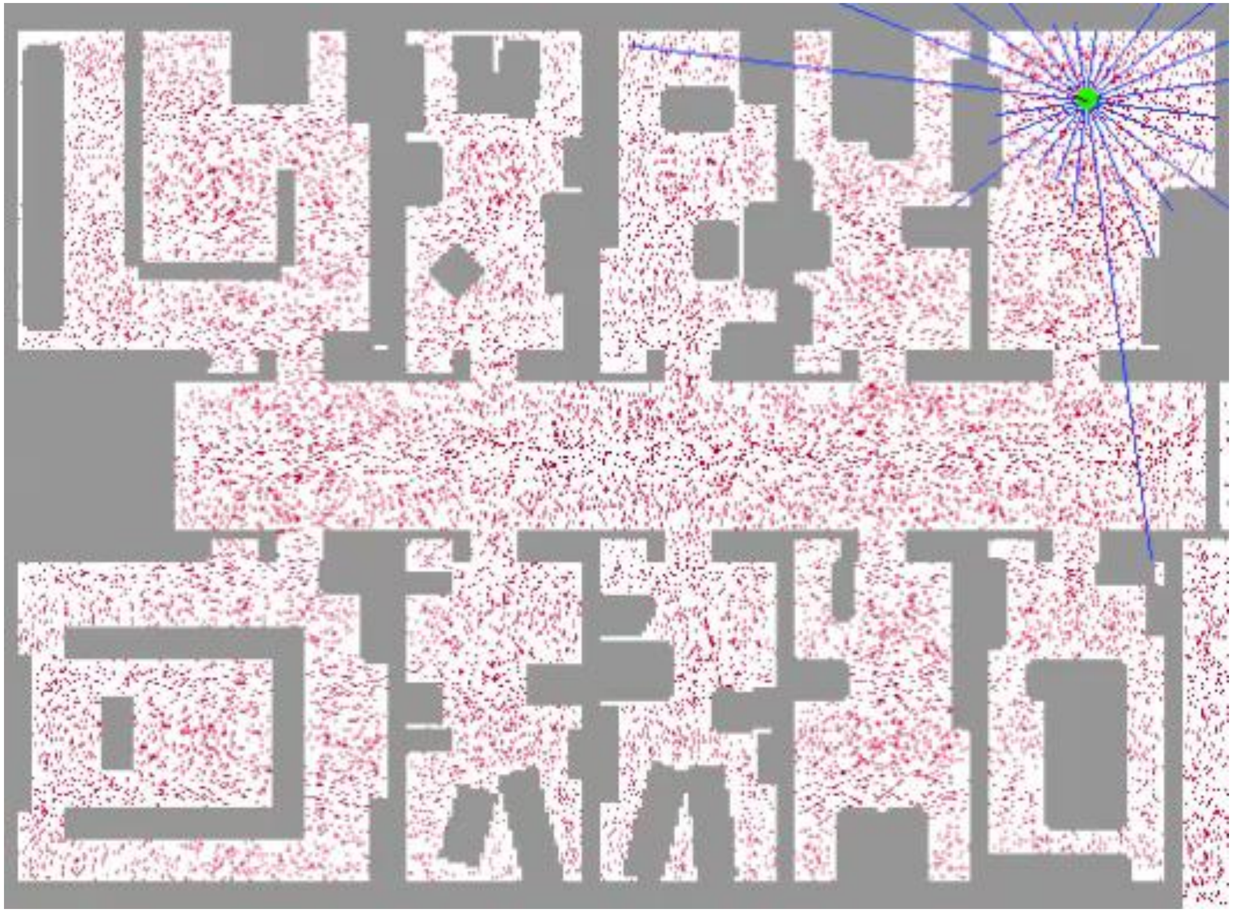
- The "belief" $bel(x_t)$, i.e. the posterior density of robot pose is represented by a particle set:

$$\mathcal{X}_t \quad := \quad x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

- The particle set is recursively updated at each iteration,
- "*Condensation*" of particles around a state indicates "*high posterior likelihood*" of state; given "*measurements*" and "*commands*"

Particle Filters - Example

- **How it looks like:**



http://robots.stanford.edu/movies/sca80a0.avi

Particle Filters – How it Works?

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$)**:**

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

for $m = 1$ *to* M *do*

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ *to* M *do*

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Particle Filters – How it Works?

```
Algorithm Particle_filter(X_{t-1}, u_t, z_t):
    X̄_t = X_t = ∅
    for m = 1 to M do
        sample x_t^[m] ~ p(x_t | u_t, x_{t-1}^[m])
        w_t^[m] = p(z_t | x_t^[m])
        X̄_t = X̄_t + ⟨x_t^[m], w_t^[m]⟩
    endfor
    for m = 1 to M do
        draw i with probability ∝ w_t^[i]
        add x_t^[i] to X_t
    endfor
    return X_t
```

Previous set of "particles", command and current measurement is used

Intermediate *and* ***Posterior particle*** *sets are initialized to empty*

We will consider ***M*** *paricles*

A new sample is drawn from the ***motion model****: "prediction step"*

A ***"weight" ("importance factor")*** *is assigned for each sample to integrate the measurement:* ***measurement model: "Update Step"***

New ***sample*** *added to the set, paired with its* ***weight***

Particle Filters – How it Works?

Algorithm Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$:

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

for $m = 1$ *to* M *do*

sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

$w_t^{[m]} = p(z_t \mid x_t^{[m]})$

$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ *to* M *do*

draw i with probability $\propto w_t^{[i]}$

add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

At this point, we have something different than what we started with!

The “real trick” of particle filters:
importance sampling

A new sample set is generated:
Probability of drawing a sample is proportional to its weight

Samples drawn “with replacement”
I.e., same particles may be picked more than once for the new set
(And some others may be lost)

Resampling: Importance Sampling

- **Interesting and necessary step of particle filters,**
- **Transform a set of (x_i, w_i) pairs into a new proper particle set (no weights) for time t,**
- **Probability of drawing (x_i) should be proportional to its weight (w_i)**
- **M particles are chosen "with replacement",**
- **(Same particle may be chosen multiple times, some particles may be lost)**

Importance Sampling: Intuition

- **When we pass the particles from the motion model, we have the "*prediction*" step,**
- **We have a set of particles representing:**

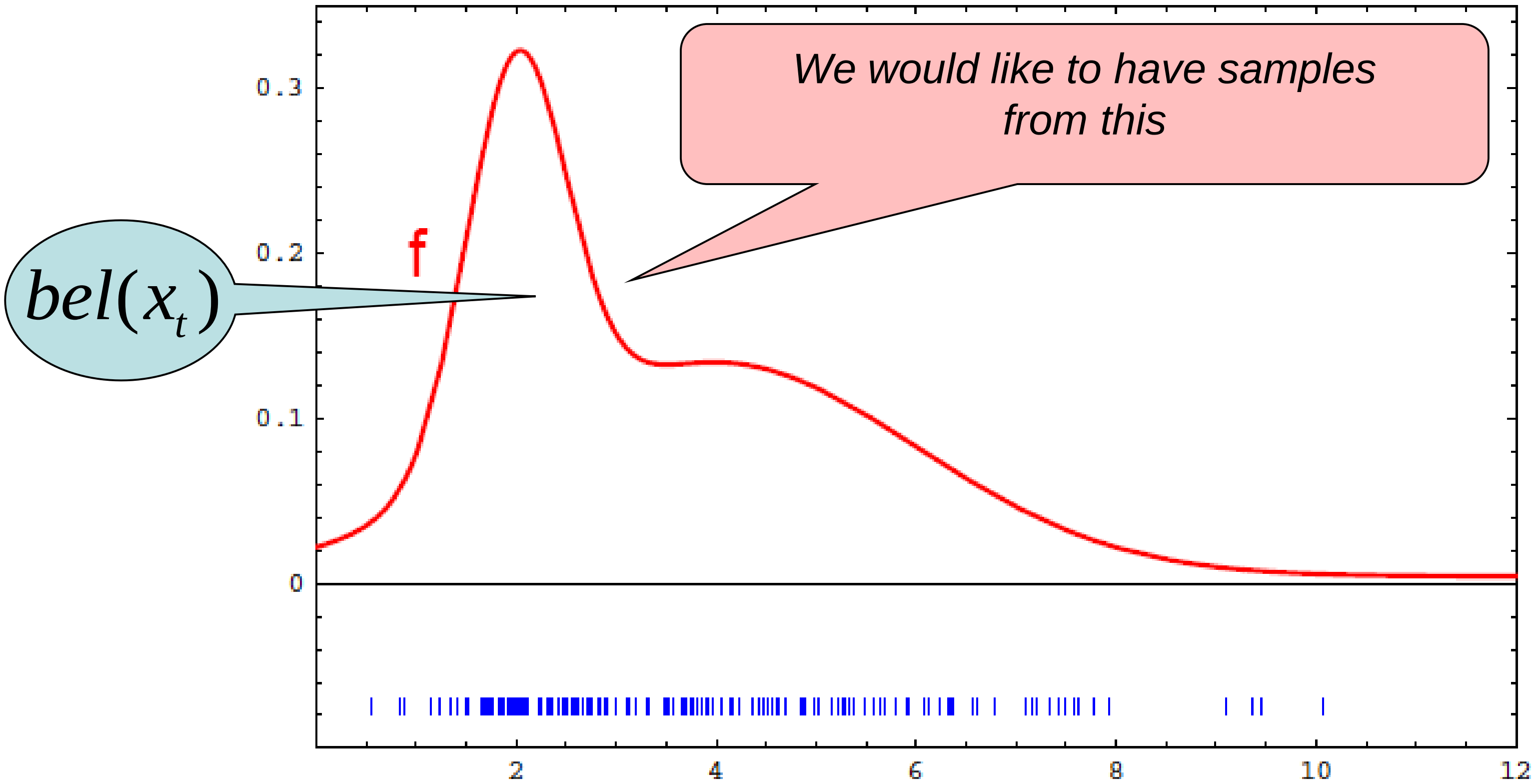
$$\overline{bel}(x_t)$$

- **How can we obtain a set of samples distributed (approximately) according to**

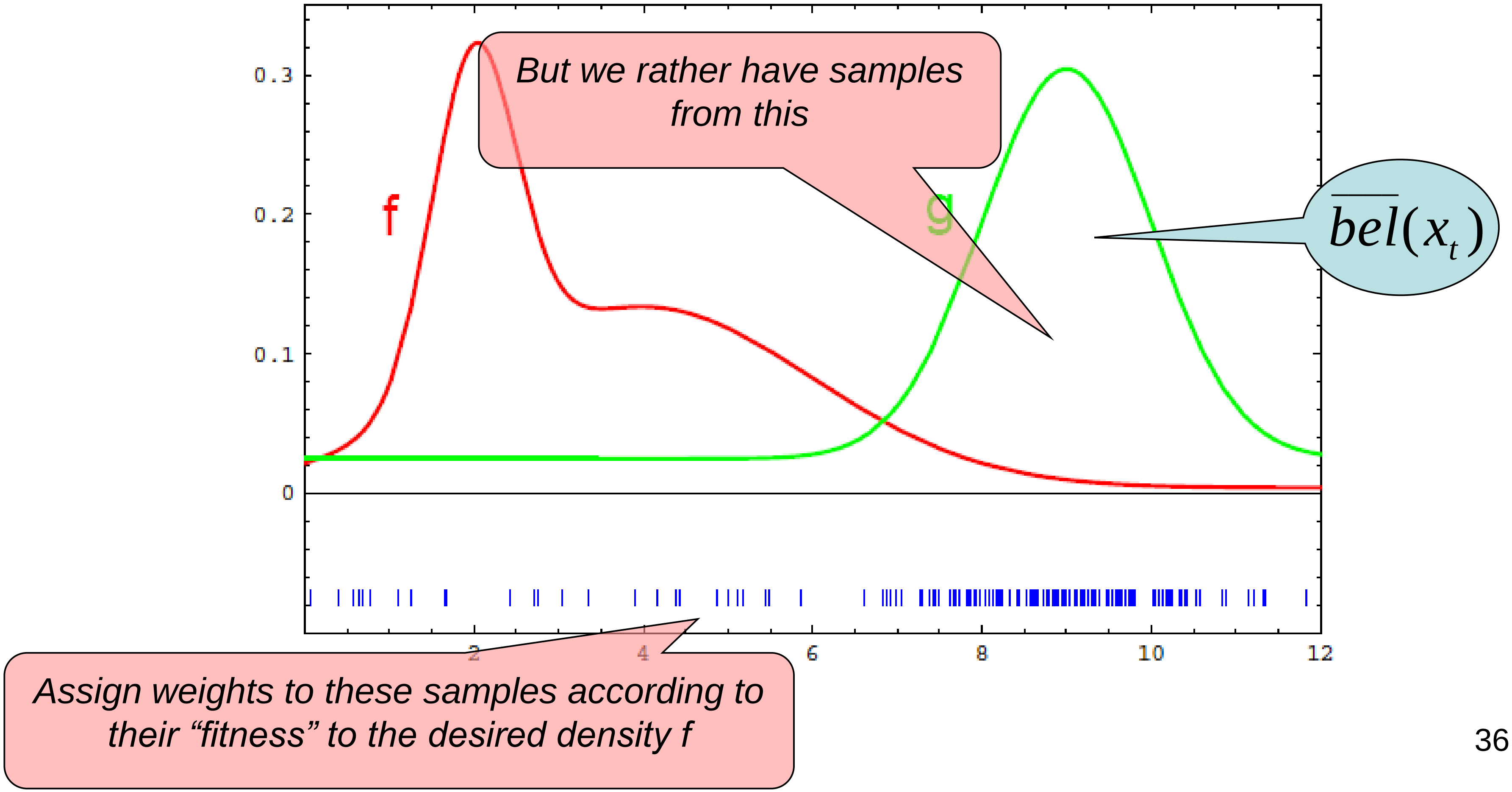
$$bel(x_t)$$

which also integrates the measurements?

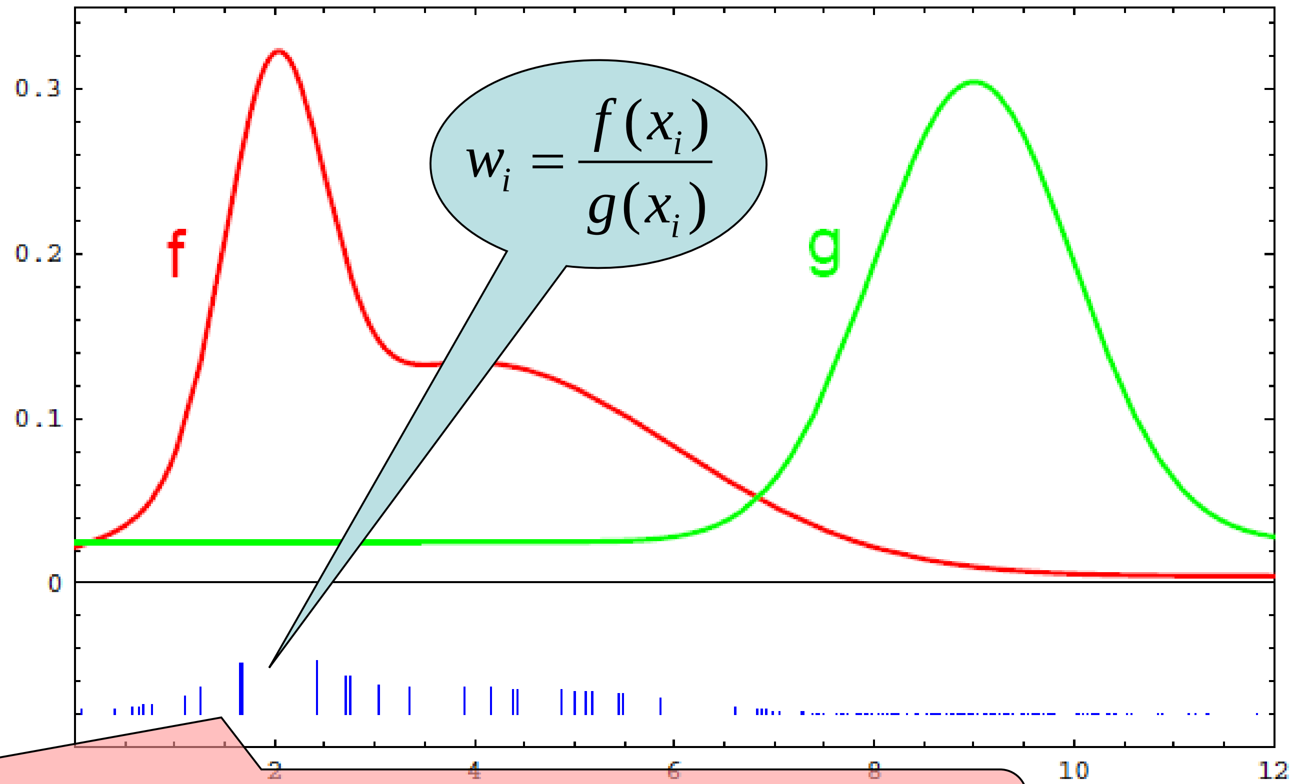
Importance Sampling: Intuition



Importance Sampling: Intuition



Importance Sampling: Intuition



Samples of g can represent f ***only if*** *they are weighted accordingly*

Importance Weights

- **Consider the update step of the Bayes Filter:**

$$bel(x_t) = \eta \, p(z_t \mid x_t) \, \overline{bel}(x_t)$$

- **Hence we have:**

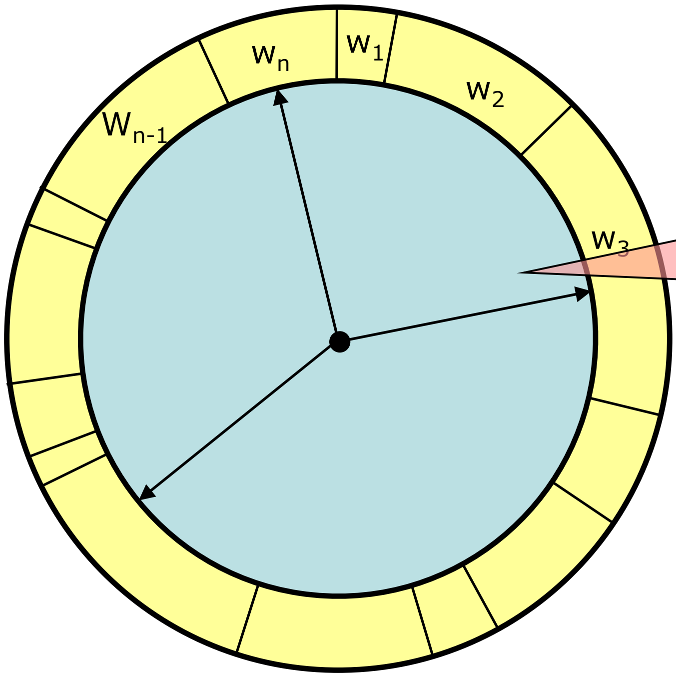
$$w_i = \frac{f(x_i)}{g(x_i)} = \frac{bel(x_i)}{\overline{bel}(x_i)} = \frac{\text{target distribution}}{\text{proposal distribution}} = \eta \, p(z_t \mid x_t)$$

- **The normalizer in the w_i (hence its actual value) is irrelevant because we only resample with probabilities *proportional* to w_i.**
- **Hence we can use in the algorithm:**

$$w_i = p(z_t \mid x_t)$$

The *Resampling* Step

- **Now, let us resample according to weights:**
- **One way: *Roulette Wheel approach*:**



A random number is generated from a uniform distribution.

Probability that a particle is selected is *proportional to its weight.*

Problems With Sampling/Resampling

- **Estimator Variance:**

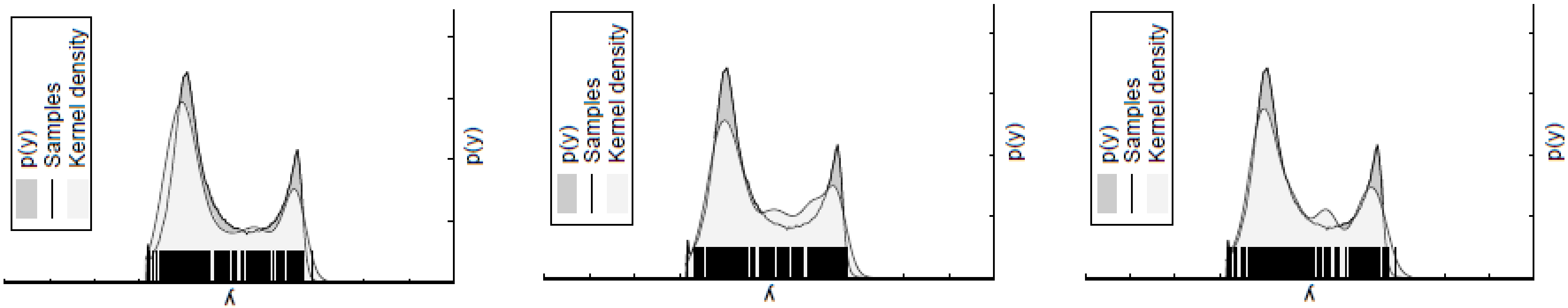
  (Variance of particles as a density estimator)

  Is a source of performance loss in Particle Filters

  "The fact that statistics (mean, variance, …) computed from *M* finite samples drawn from a distribution will differ from the statistics of the original distribution"

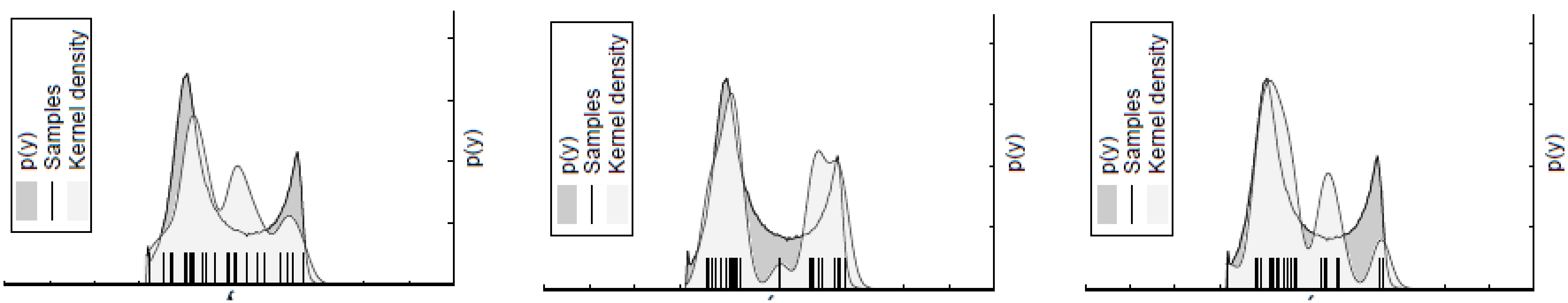
  That means *M* samples may be a poor representation of the true density (in particular if *M* is small).

Estimator Variance Illustration



250 samples

Every particular sample set "instance" results in a different density approximation
(The continuous approximation of particles illustrated with "Kernel density")



25 samples

Problems With Sampling/Resampling

(Continued)

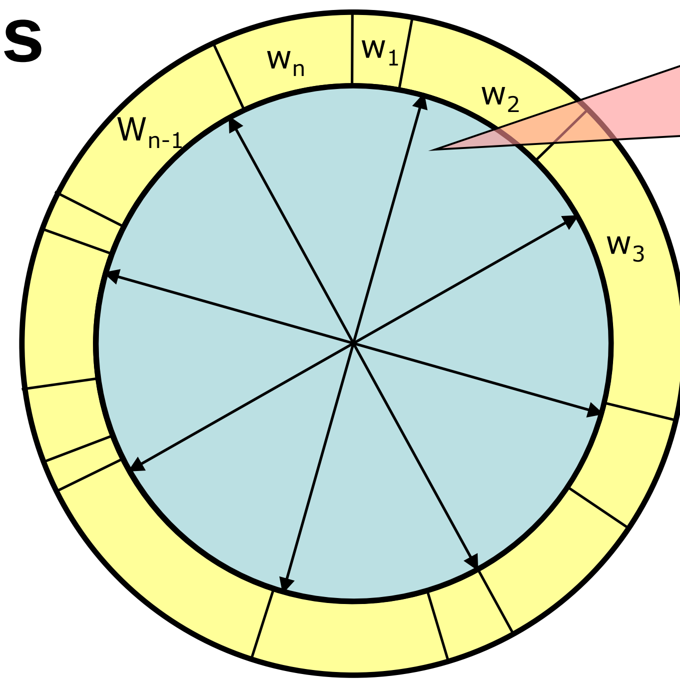
- **Estimator Variance Divergence**
  Major failure mode of original particle filter with Roulette wheel resampling
- ***Example:*** **Stationary robot (constant state) and no sensor (no measurement of state).**
- **Repetitive resampling monotonically increases estimator variance (particle set converges into a spurious local maxima)**
- ***End effect:*** **Particles may be gradually lost (resulting in a single particle) due to random resampling, resulting in "localized" robot (!!!)**

Problems With Sampling/Resampling
(Continued)

- Estimator Variance: One fix
- Resampling step may be done less frequently, (multiple measurements can still be multiplicatively integrated into the weight factors)
- Stop resampling if the robot stops / new measurements are not coming,
- If resampling is not done, particles may be wasted in regions of low probability,
- The proper precision-cost balance requires experience and *tuning* for a particular application.

Problems With Sampling/Resampling
(Continued)

- **Estimator Variance: Another Fix**
- **Use of a “low variance sampler” instead of a roulette wheel strategy**
- **Use a sequencial stochastic process to pick particles**



M equally spaces samples.

One random number generates *M* *universally sampled particles*

- **Stochastic universal sampling**
- **Systematic resampling**
- **Linear time complexity**
- **Easy to implement, low variance**

Problems With Sampling/Resampling

(Continued)

- **Particle Deprivation**
- **Divergence of the filter characterized by *no samples remaining in the vicinity of the correct state.***
- **Result of Estimator Variance problem,**
- **Can be diminished (but not fixed) by:**
  - **Increasing sample size (*M*),**
  - **Adding uniformly distributed samples at each step,**

Again: Particle Filtering Algorithm

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

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

for $m = 1$ *to* M *do*

sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

$w_t^{[m]} = p(z_t \mid x_t^{[m]})$

$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ *to* M *do*

draw i with probability $\propto w_t^{[i]}$

add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Summary

- **We discussed non-parametric Bayes Filter implementations: Histogram and Particle Filters,**
- **Approximation of posterior by finite set of values,**
- **Histogram Filter: Decompose state space into *M convex regions – assign a probability for each region,***
- **Decomposition technique plays important role: Static and Dynamic versions, as well as Topological/Non-topo versions determine performance and implementation,**
- **Particle Filter: Represent the posterior by random samples of state. Recursively update these.**
- **Very easy implementation and flexible posterior,**
- **Some failure modes that need special attention**