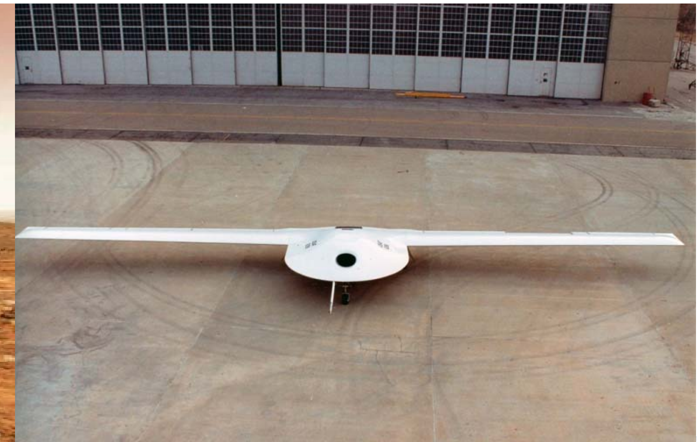
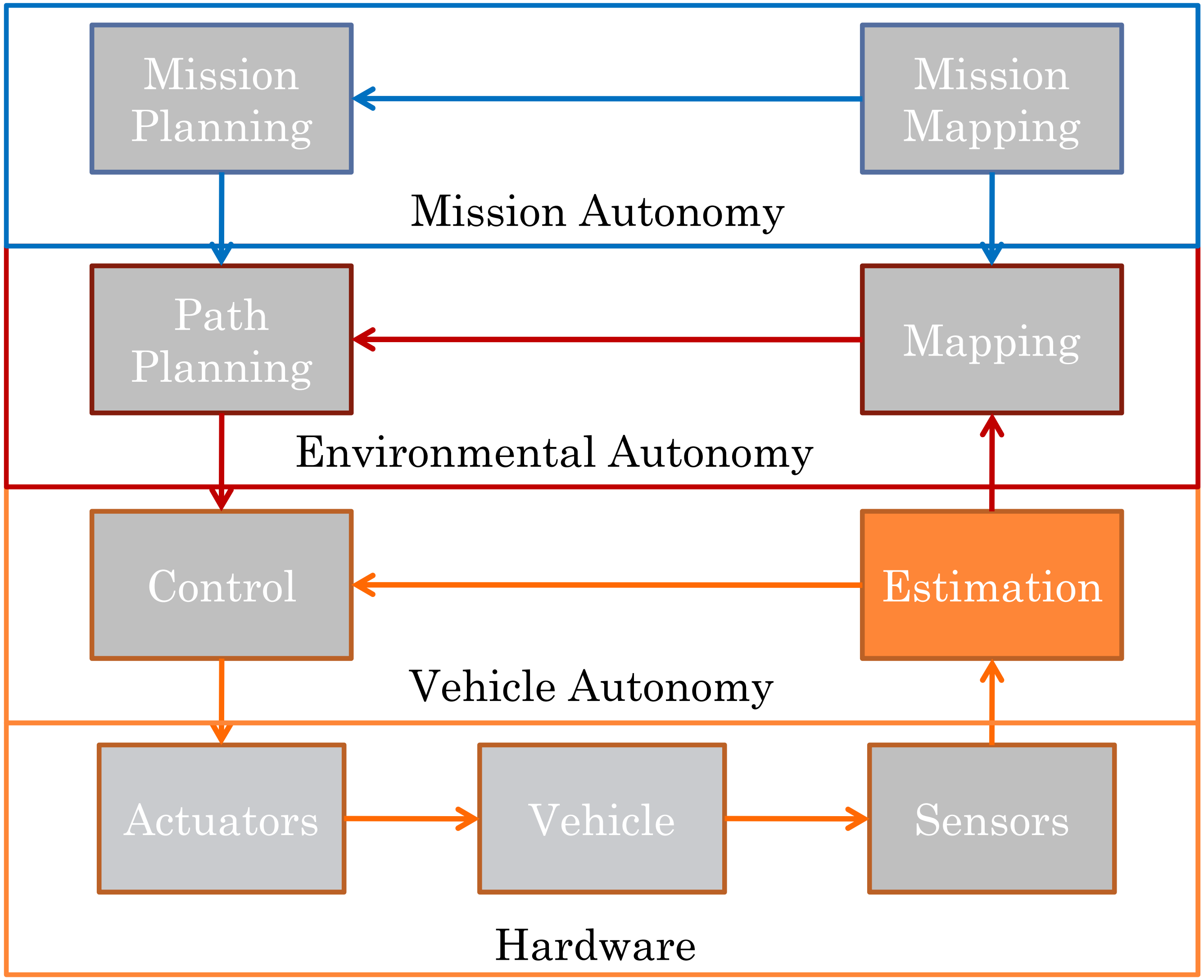





ME 597: Autonomous Mobile Robotics
Section 7 – Estimation II

Prof. Steven Waslander

Components



Mission Planning
Mission Mapping
Mission Autonomy
Path Planning
Mapping
Environmental Autonomy
Control
Estimation
Vehicle Autonomy
Actuators
Vehicle
Sensors
Hardware

Outline

- Review of Probability
- Bayes Filter Framework
- Kalman Filter
- Extended Kalman Filter
- Unscented Particle Filter
- Particle Filter

PARTICLE FILTERS

- The Bayes Filter Framework has now been adapted to
  - Kalman Filter
    - Linear models with additive Gaussian noise
  - Extended Kalman Filter
    - Nonlinear models with additive Gaussian noise

- Both continuous Gaussian methods are computationally appealing
  - Even for large numbers of state, measurement variables
  - Benefit arises from ability to maintain Gaussian beliefs
    - Track only mean and covariance throughout filtering process

Particle Filters

- In both cases, modeling requirements rule out a significant portion of real systems
  - Nonlinear systems where linearization is a poor approximation over distribution range
  - Systems with multiple reasonable hypotheses

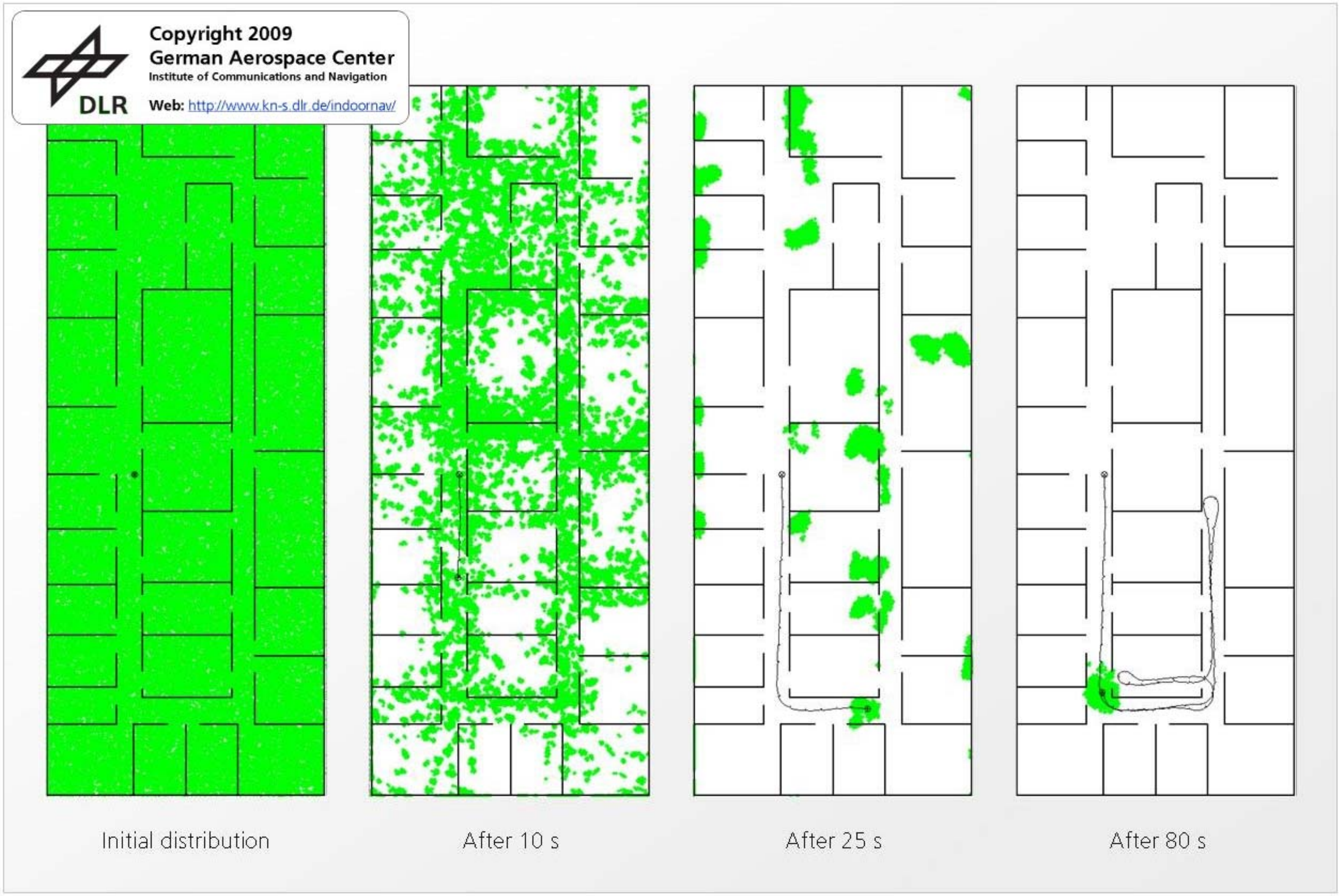
- Alternatives include non-parametric filters
  - Filters that do not track distribution parameters
  - Bayes/Histogram Filter
    - Discrete state systems with known probabilities
    - Explodes computationally for higher dimensional models
  - Particle Filter
    - Maintain a sample set representation of beliefs
    - Results can be poor in higher dimensional models
    - Also called Sequential Monte Carlo methods

Particle Filters

- Modeling assumption
  - Instead of assuming Gaussian, tracking μ_t, Σ_t , generate a set of sample states from each distribution
    - Each sample is a hypothesis about the current state
    - Properties of the whole collection of samples are used to generate estimates
  - Not possible to sample belief distribution directly, must apply Importance Sampling

Particle Filter

- Example particle sets – density of points defines probability



Particle Filters

- Generating samples from a known distribution
  - Given a probability density function, draw samples with the appropriate probability
  - Easy for uniform, Gaussian
    - Use built in Matlab functions
  - Harder for arbitrary pdfs, but approximation is possible
  - Needed in Particle filters to perform measurement update

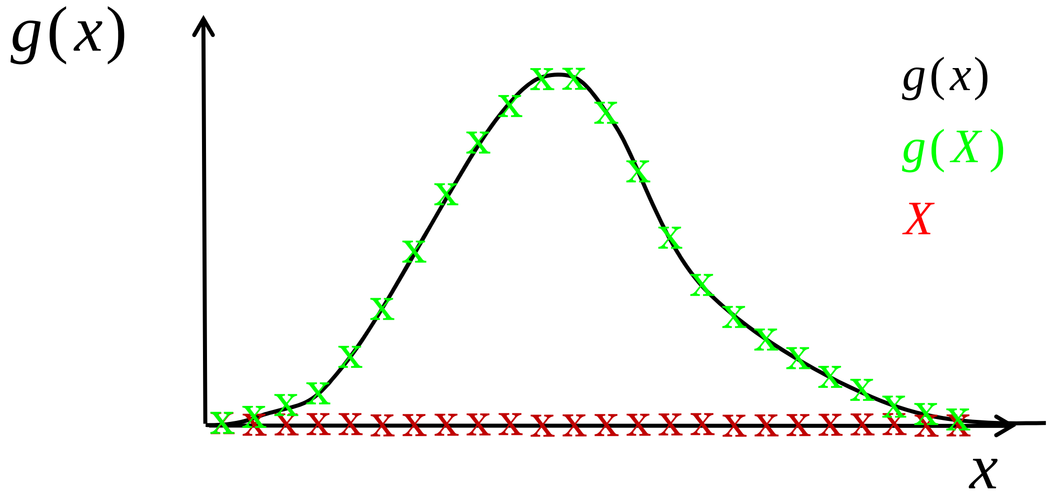
Particle Filters

- Generating samples from a distribution
  - Given a state $x \in \mathbb{R}$ and a distribution $g(x): \mathbb{R} \rightarrow [0,1]$

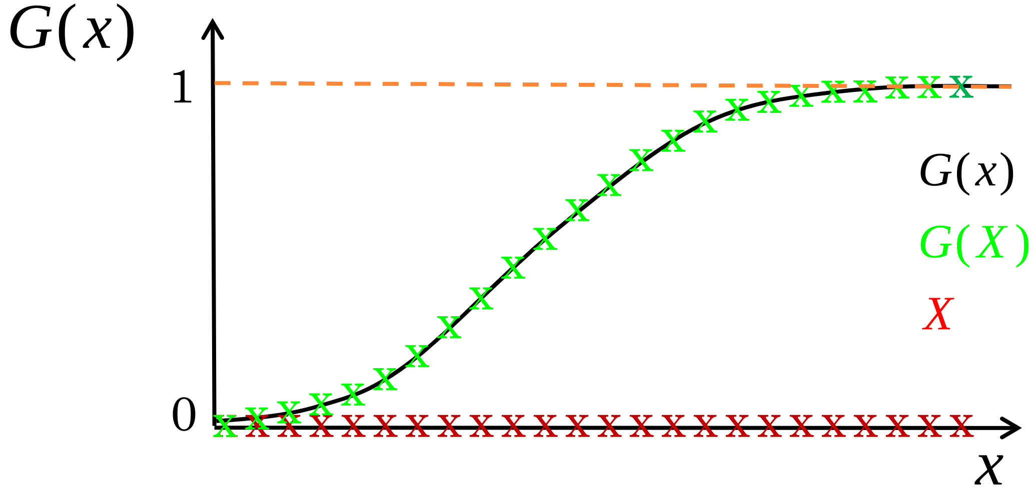
1. Create a vector *X* of evenly spaced values of *x* over the range of interest
   - e.g. If g(x) is Gaussian, create *X* to span ±5σ about mean
2. Create an exact/approximate cumulative distribution vector, *G(X)*
   - Integrate probability distribution *g(x)* to get *G(x),* and create the vector *G(X)*
   - Or sum probabilities *g(X)* and normalize to get vector *G(X)*
3. Draw samples from a uniform distribution over [0,1]
4. Find closest value to sample in G(X)
5. Corresponding value of *x* is a sample, denoted $x_g^{[i]}$

Particle Filters

- Sampling of *g(x)*
  - *1.*

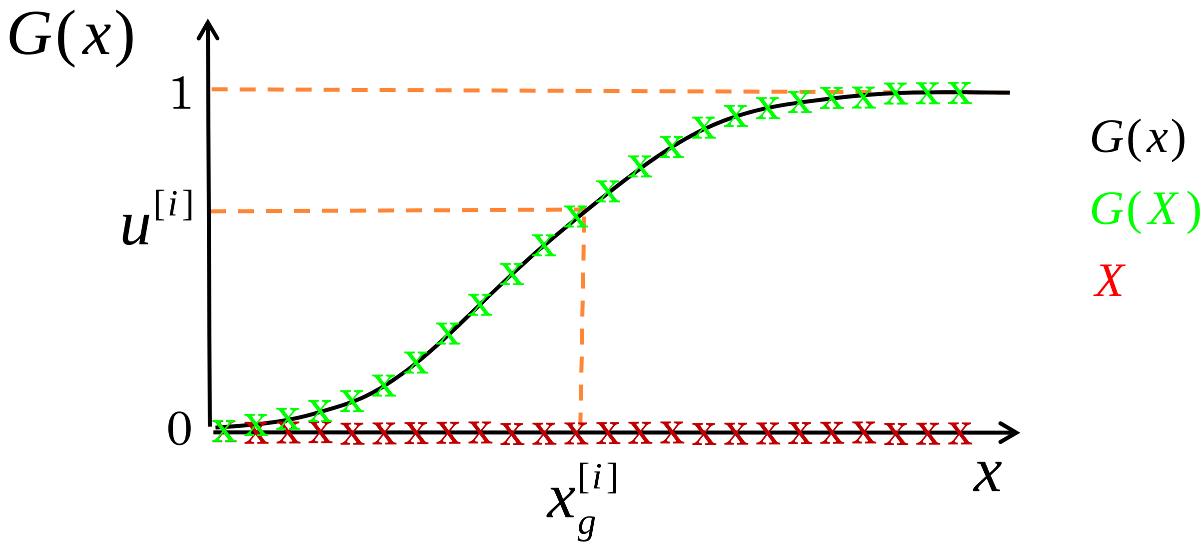


  - *2.*



Particle Filters

- Sampling of *g(x)*
  - *3.-5.*

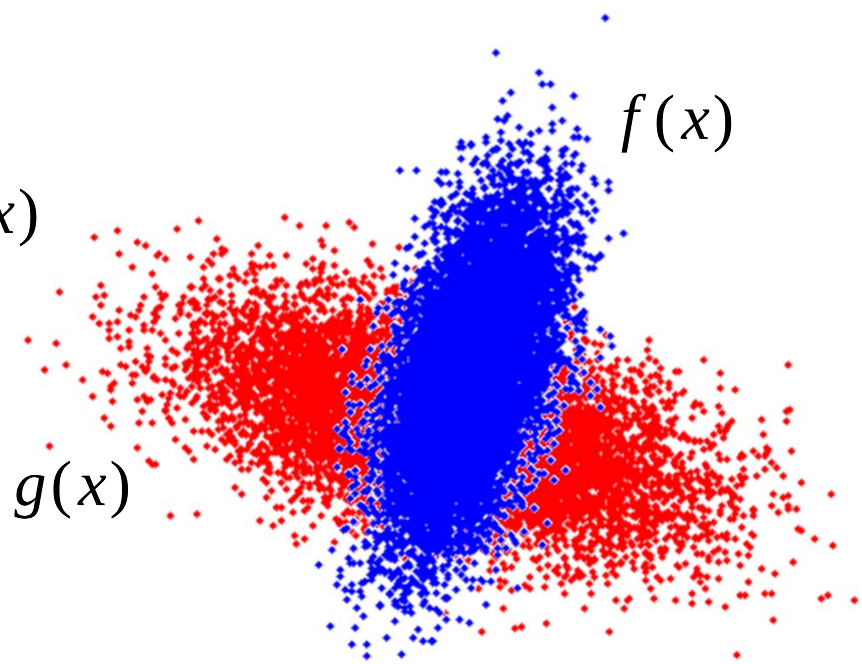


PARTICLE FILTERS

- Importance Sampling
  - Goal: perform a calculation using a distribution, *f(x)*, but without being able to sample it directly
    - *f(x)* = Target distribution, unknown
  - Can first sample a different distribution, *g(x)*,
    - *g(x)* = Proposal distribution, known

$$f(x) = F(g(x))$$

$$f(x) = \frac{f(x)}{g(x)} g(x)$$



Particle Filters

- Importance Sampling
  - Then use relationship between distributions if known to define the weighting factor as

$$w(x) = \frac{f(x)}{g(x)}$$

- Finally, resample from *g(x)*, with weights *w(x)* to generate samples of *f(x)*

- If weighting factor is known, can perform this calculation without knowing *f*(*x*)
  - Note that $g(x)>0$ wherever $f(x)>0$ for this to be valid

Particle Filters

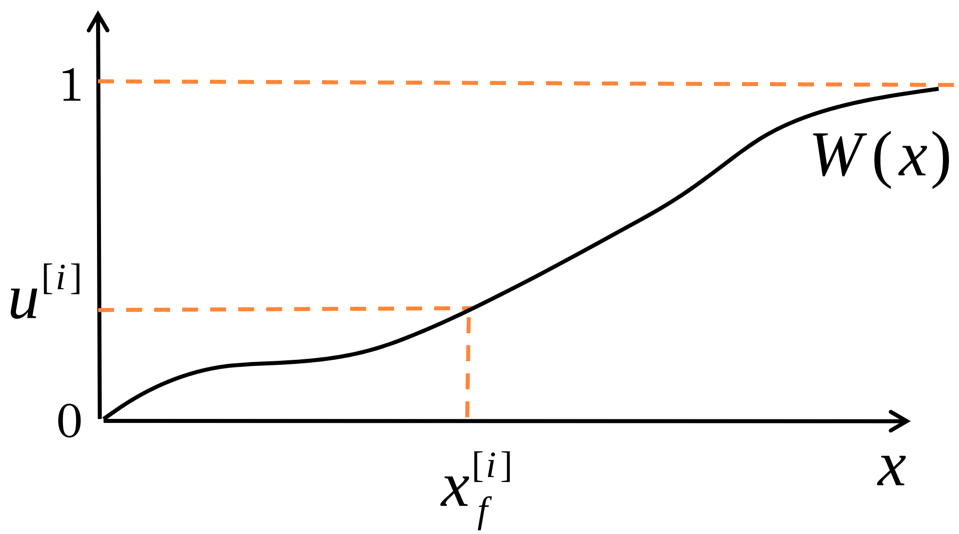
- Importance Sampling
  - Define weights for each sample $x_g^{[i]}$ in S

$$w(x) = \frac{f(x)}{g(x)}$$

  - Weights are the probability that we should include sample $x_g^{[i]}$ in our final sample set
    - The importance of sample $x_g^{[i]}$
  - Not obvious how to calculate the weight at this point, will become clear in derivation of particle filter
  - For now, found by dividing $f(x_g^{[i]})$ by $g(x_g^{[i]})$
    - This assumes complete knowledge of $f(x)$,
      - (yes, cheating)

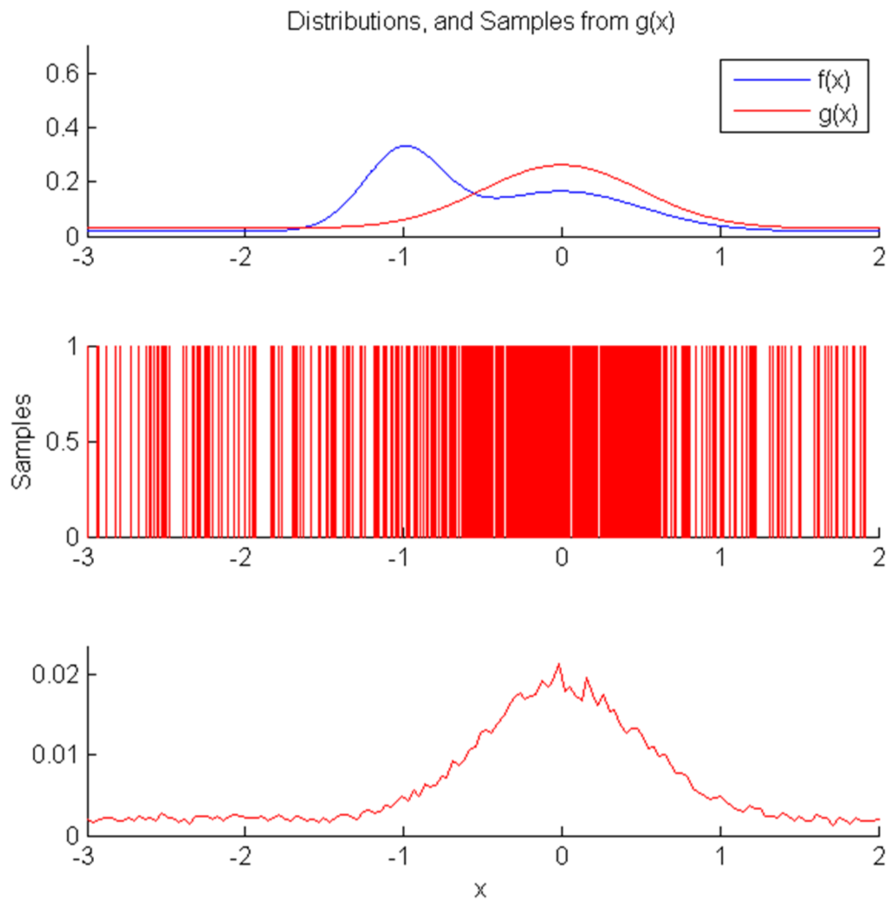
Particle Filters

- Importance Sampling
  - Importance sampling of *f(x)*
    1. Define cumulative distribution *W(x)* based on weights *w(x)* as before (samples need not be ordered)
    2. For each sample
       1. Take uniform sample, $u^{[i]}$
       2. Find first element of *W(x)* that exceeds current sample
       3. Add corresponding value of $x_g^{[i]}$ as a sample to sample set, *S'*



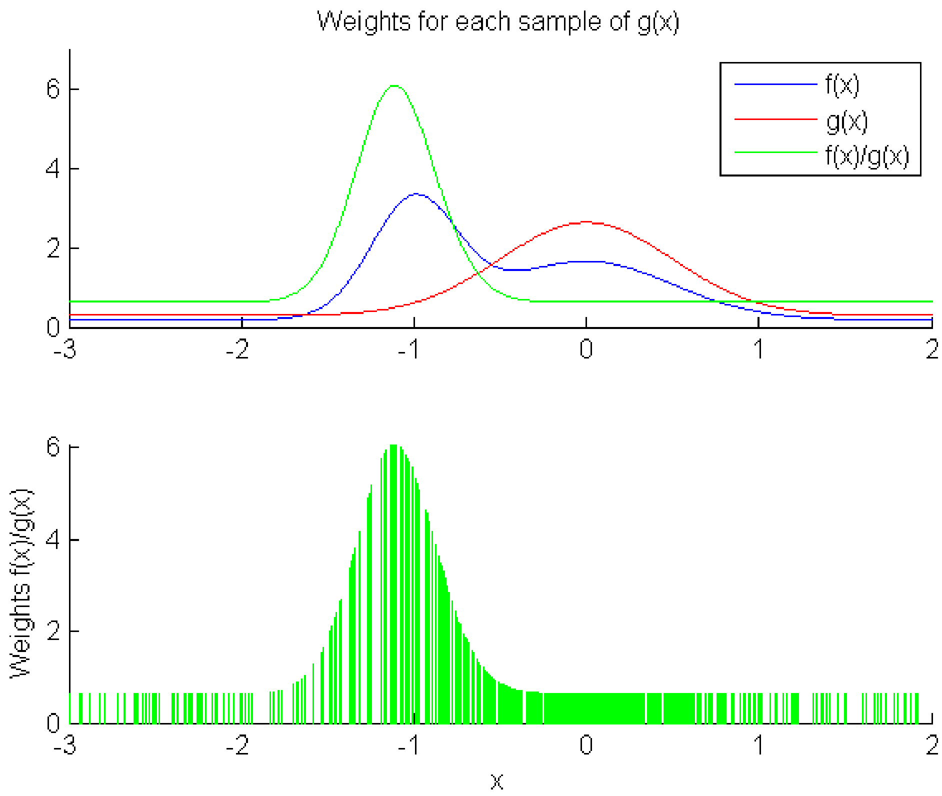
Particle Filters

- Example
  - Target distribution *f(x)*, proposal distribution *g(x)*
  - 20000 samples drawn from *g(x)* (1/25th of samples shown)



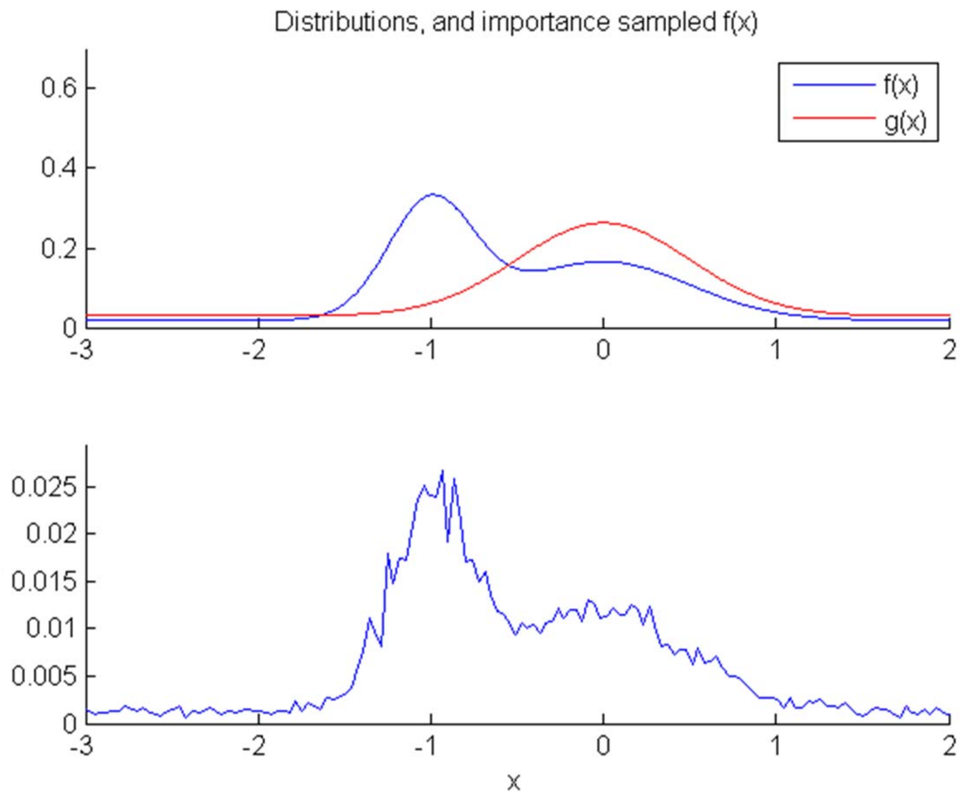
Particle Filters

- Example
  - Weights for all x and sample weights for each sample



Particle Filters

- Example
  - Importance sample points to generate new set
  - New set is distributed according to *f(x)*



Particle Filters

- The Particle Set
  - A sample can be drawn from a *proposal distribution*

$$x_p^{[i]}$$

  - The sample is assigned a weight

$$w_p^{[i]}$$

  - The combination of sample and weight is a particle

$$s_p^{[i]} = \{x_p^{[i]}, w_p^{[i]}\}$$

  - The particle set is used to generate an approximation to the *target distribution*

$$s^{[i]} = \{x^{[i]}, w^{[i]}\} \qquad S = \{s^{[1]}, \ldots, s^{[I]}\}$$

  - I is the total number of particles in the set
  - The approximation improves as $I \rightarrow \infty$

Particle Filters

- Defining the usual model elements, in general probabilistic form
  - State prior

$$p(x_0)$$

  - Motion Model

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

  - Measurement Model

$$p(y_t \mid x_t)$$

  - Only restrictions on model elements are that samples can be drawn from them, (probabilities known for all conditional values)

Particle Filters

- Beliefs
  - In particle filters, the belief distributions will be represented by particle sets
    - The belief

$$x_t^{[i]} \sim bel(x_t) = p(x_t \mid y_{1:t}, u_{1:t})$$

$$S_t = \{s_t^{[1]}, \ldots, s_t^{[I]}\}$$

    - The predicted belief

$$\bar{x}_t^{[i]} \sim \overline{bel}(x_t) = p(x_t \mid y_{1:t-1}, u_{1:t})$$

$$\bar{S}_t = \{\bar{s}_t^{[1]}, \ldots, \bar{s}_t^{[I]}\}$$

Particle Filters

- Particle Filter Algorithm
  1. Prediction Transformation
     - Transform prior belief particle set to predicted belief through sampling
  2. Importance factor
     - Using measurement, calculate particle importance factor
       - Probability of the measurement occurring, given the state was defined by the current particle
  3. Resampling
     - Transform predicted belief particle set to belief using importance sampling

- Note: steps 1,2 can be combined into a single loop, if prediction and measurement steps are combined

PARTICLE FILTERS

- Particle Filter Components
  1. Prediction Update
     - The samples $x_{t-1}^{[i]}$ are known from previous iteration
     - The motion model and input are known
     - It is therefore possible to generate samples of $\overline{bel}(x_t)$

$$\bar{x}_t^{[i]} \sim p(x_t \mid x_{t-1}^{[i]}, u_t)$$

     - One new sample is drawn from each distribution defined by the prior samples
     - The set of I new samples defines an approximation to $\overline{bel}(x_t)$
     - Unit weighting on each particle

$$\bar{S}_t = \{\bar{s}_t^{[1]}, \ldots, \bar{s}_t^{[I]}\}$$

Particle Filters

- Particle Filter Components
  2. Measurement update
     - The measurement is known but the state is not
     - Would like to generate a particle set to represent $bel(x_t)$
       - Target distribution
     - Have particle set representation of predicted belief $\overline{bel}(x_t)$
       - Proposal distribution
     - Use importance sampling to generate belief update

$$w_t^{[i]} = \eta \frac{bel(x_t)}{\overline{bel}(x_t)}$$

     - The proper weighting to use turns out to be

$$w_t^{[i]} = p(y_t \mid \overline{x}_t^{[i]})$$

PARTICLE FILTERS

- Particle Filter Expanded Algorithm
  1. Prediction update
     1. For each particle in S_{t-1}
        1. Sample $\bar{x}_t^{[i]} \sim p(x_t \mid x_{t-1}^{[i]}, u_t)$
        2. Weight $\bar{w}_t^{[i]} = 1$
        3. Add to $\bar{S}_t$
  2. Measurement update
     1. For each particle in $\bar{S}_t$
        1. Calculate weighting

$$w_t^{[i]} = p(y_t \mid \bar{x}_t^{[i]})$$

     2. For j = 1 to I
        1. Draw particle $\bar{x}_t^{[i]}$ with probability $\propto w_t^{[i]}$
        2. Add to S_t as $s_t^{[i]} = \left\{x_t^{[i]}, 1\right\}$

Particle Filters

- Particle Filter Algorithm (simplified)
  1. For each particle in S_{t-1}
     1. Propagate sample forward using motion model (sampling)

$$\bar{x}_t^{[i]} \sim p(x_t \mid x_{t-1}^{[i]}, u_t)$$

     2. Calculate weight (importance)

$$w_t^{[i]} = p(y_t \mid \bar{x}_t^{[i]})$$

     3. Store in interim particle set

$$S'_t = S'_t + \{s_t^{[i]}\}$$

  2. Normalize weights
  3. For $j = 1$ to I
     1. Draw index i with probability $\propto w_t^{[i]}$ (resampling)
        1. Add to final particle set

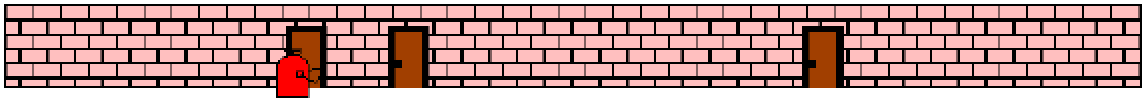
$$S_t = S_t + \{s_t^{[i]}\}$$

Particle Filters

- Example
  - Robot Localization
    - Robot travels along hallway, can detect doors within a range with noisy sensor
    - Knows probability of detecting a door, given a specific location

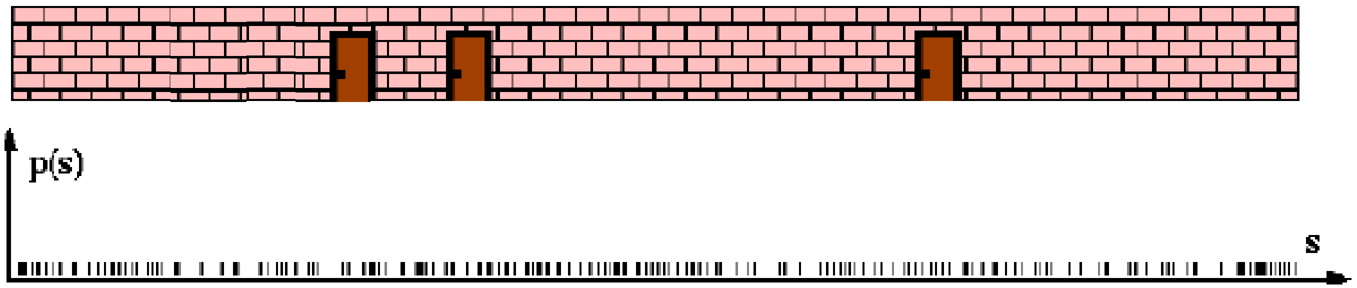
$$p(y_t \mid x_t)$$

    - Knows motion model, and has uniform initial belief



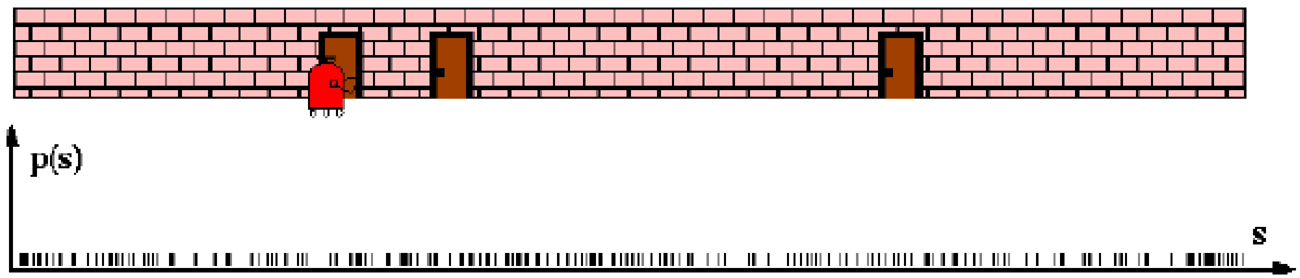
PARTICLE FILTERS

- Example
  - Step one
    - Sample uniformly over state space



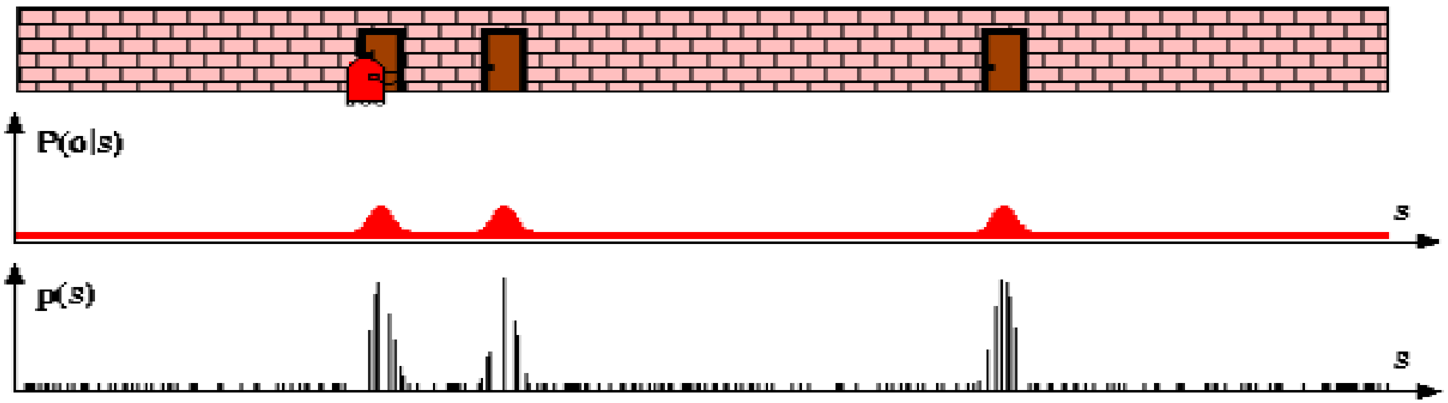
Particle Filters

- Example
  - Step Two
    - Propagate samples through motion model



PARTICLE FILTERS

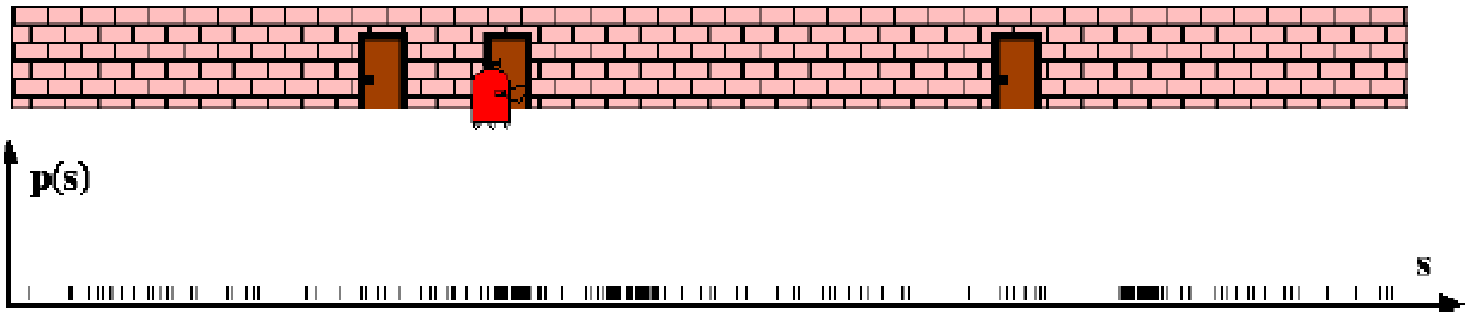
- Example
  - Step three
    - Take a measurement, and use $p(y_t \mid x_t)$ to calculate weights



    - Particles that are more likely have higher weights
      - Starting to narrow down position options
      - Still difficult to estimate state (mean?)

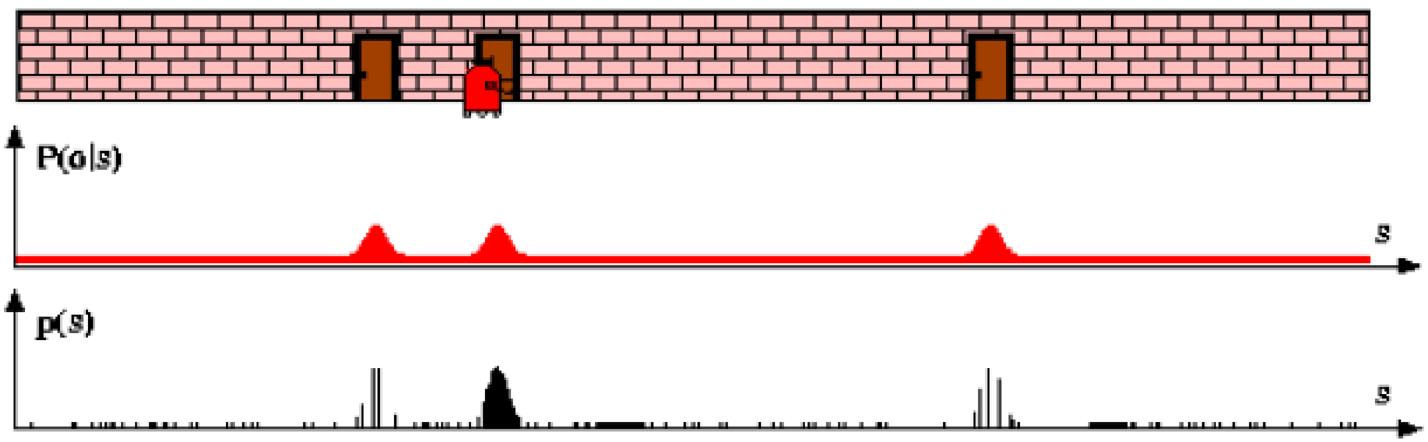
Particle Filters

- Example
  - Step four
    - Perform resampling to get more particles in areas of higher probability
    - Reset weights to 1, as particle locations capture probability information
  - Repeat
  - The following particle set shows how the motion model distributes the identical particles that result from resampling



Particle Filters

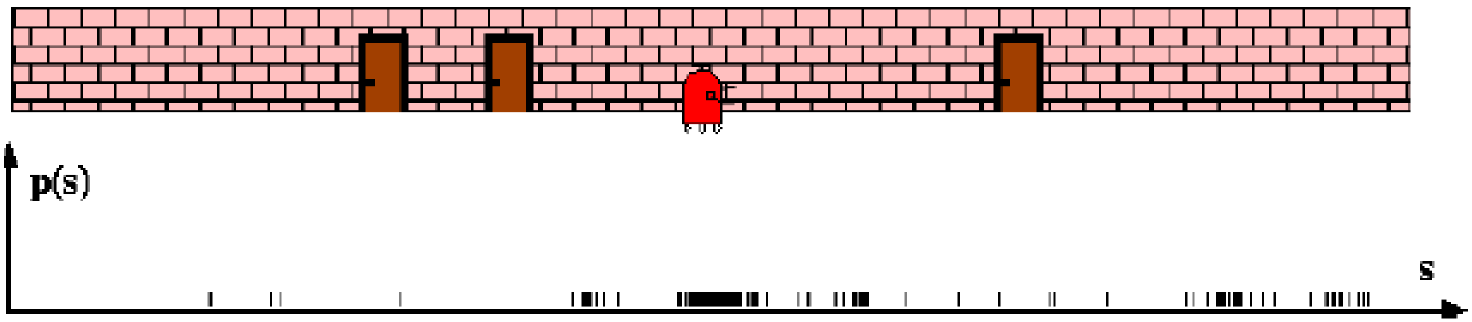
- Example
  - After a second measurement, weights are again assigned to the particles



  - True state starts to become apparent

Particle Filters

- Example
  - After resampling again, propagate with motion model sampling



Particle Filters

- Derivation
  - Consider the particles as state sequence samples

$$x_{0:t}^{[i]} = x_0^{[i]}, x_1^{[i]}, \ldots, x_t^{[i]}$$

  - Form belief over entire sequence

$$bel(x_{0:t}) = p(x_{0:t} \mid u_{1:t}, y_{1:t})$$

    - Instead of just

$$bel(x_t) = p(x_t \mid u_{1:t}, y_{1:t})$$

  - This is an enormous state to approximate with a set of particles, but no matter, for derivation only

Particle Filters

- Derivation
  - Using Bayes Theorem, expand belief about last measurement

$$bel(x_{0:t}) = p(x_{0:t} \mid u_{1:t}, y_{1:t})$$
$$= \eta\, p(y_t \mid x_{0:t}, u_{1:t}, y_{1:t-1})\, p(x_{0:t} \mid u_{1:t}, y_{1:t-1})$$

  - The Markov assumption remains valid

$$= \eta\, p(y_t \mid x_t)\, p(x_{0:t} \mid u_{1:t}, y_{1:t-1})$$

Particle Filters

- Derivation
  - Conditional probability can be used to expand the last distribution

$$bel(x_{0:t}) = \eta p(y_t \mid x_t) p(x_t \mid x_{0:t-1}, u_{1:t}, y_{1:t-1}) p(x_{0:t-1} \mid u_{1:t}, y_{1:t-1})$$

  - Apply the Markov assumption again yields

$$bel(x_{0:t}) = \eta p(y_t \mid x_t) p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid u_{1:t}, y_{1:t-1})$$

Particle Filters

- Derivation
  - The sequence $x_{0:t-1}$ does not depend on u_t

$$bel(x_{0:t}) = \eta p(y_t \mid x_t) p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid u_{1:t-1}, y_{1:t-1})$$
$$= \eta p(y_t \mid x_t) p(x_t \mid x_{t-1}, u_t) bel(x_{0:t-1})$$

  - Breaking into steps
    - Prediction

$$\overline{bel}(x_{0:t}) = p(x_t \mid x_{t-1}, u_t) bel(x_{0:t-1})$$

    - i^{th} particle generated by this distribution is an element of the predicted belief particle set

Particle Filters

- Derivation
  - The measurement update uses importance sampling to generate a particle set representation of belief
    - Weighting, based on relation to predicted belief is

$$
\begin{aligned}
w_t^{[i]} &= \eta \frac{bel(x_{0:t})}{\overline{bel}(x_{0:t})} \\
&= \frac{\eta p(y_t \mid x_t) p(x_t \mid x_{t-1}, u_t) bel(x_{0:t-1})}{p(x_t \mid x_{t-1}, u_t) bel(x_{0:t-1})} \\
&= \eta p(y_t \mid x_t)
\end{aligned}
$$

    - Which confirms use of measurement model as weighting parameter
    - This confirms that particles sets are distributed according to full belief sequences, which means must hold for current state too

Particle Filters

- Example
  - Returning to the temperature control problem
    - State is current temperature
    - One dimensional example
    - Prior: Uniform over temperature range
    - Motion Model: Decaying temperature + furnace input + disturbances (opening doors, outside effects)

$$x_t = 0.8x_{t-1} + 3u_t + r_t$$

$$A = 0.8, \quad B = 3$$

$$r_t \sim N(0,2)$$

Particle Filters

- Example
  - Measurement Model
    - Directly measure the current temperature

$$y(t) = x(t) + \delta_t$$
$$\delta_t \sim N(0,4)$$

  - Controller design
    - Bang bang control, based on current estimate of temperature

$$u(t) = \begin{cases} 1 & \mu_t < 2 \\ 0 & \mu_t > 10 \\ u(t-1) & \text{otherwise} \end{cases}$$

Particle Filters

- Particle filter calculations
  1. Transform and sample from Gaussian

$$\bar{x}_t^{[i]} \sim p(x_t \mid x_{t-1}^{[i]}, u_t) = 0.8x_{t-1}^{[i]} + 3u_t + r_t$$

  2. Define weights from measurement model, with Gaussian noise centered at predicted measurement location

$$w_t^{[i]} = p(y_t \mid \bar{x}_t^{[i]})$$

  3. Resample from predicted belief particle set using weights to generate belief particle set

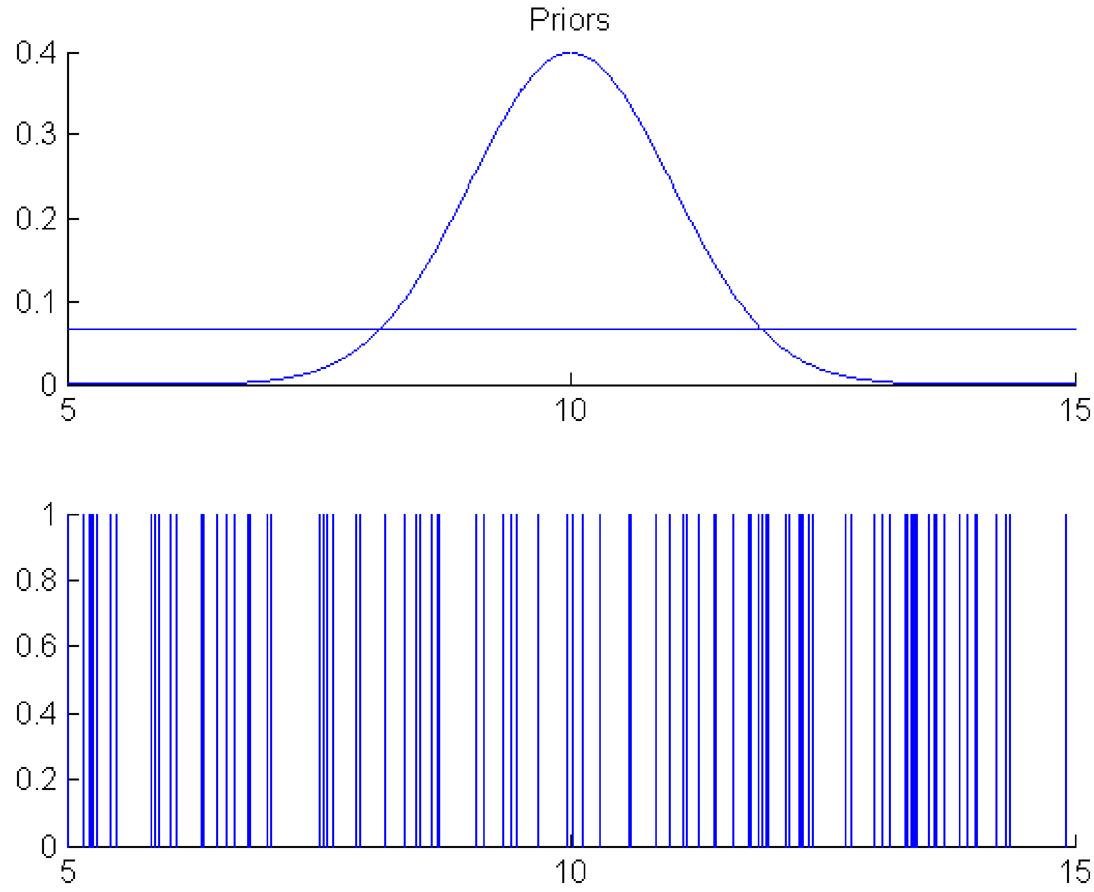
PARTICLE FILTERS

- Resulting particle filter code

```
%Particle filter estimation
 for i=1:I
     e = sqrt(R)*randn(1);
     Xp(i) = A*X(i) + B*u(t) + e;
     w(i) = normpdf(y(t),C*Xp(i),Q);
 end
 W = cumsum(w);
 W = W/max(W);
 for j=1:I
     i = find(W>rand(1),1);
     X(j) = Xp(i);
 end
```

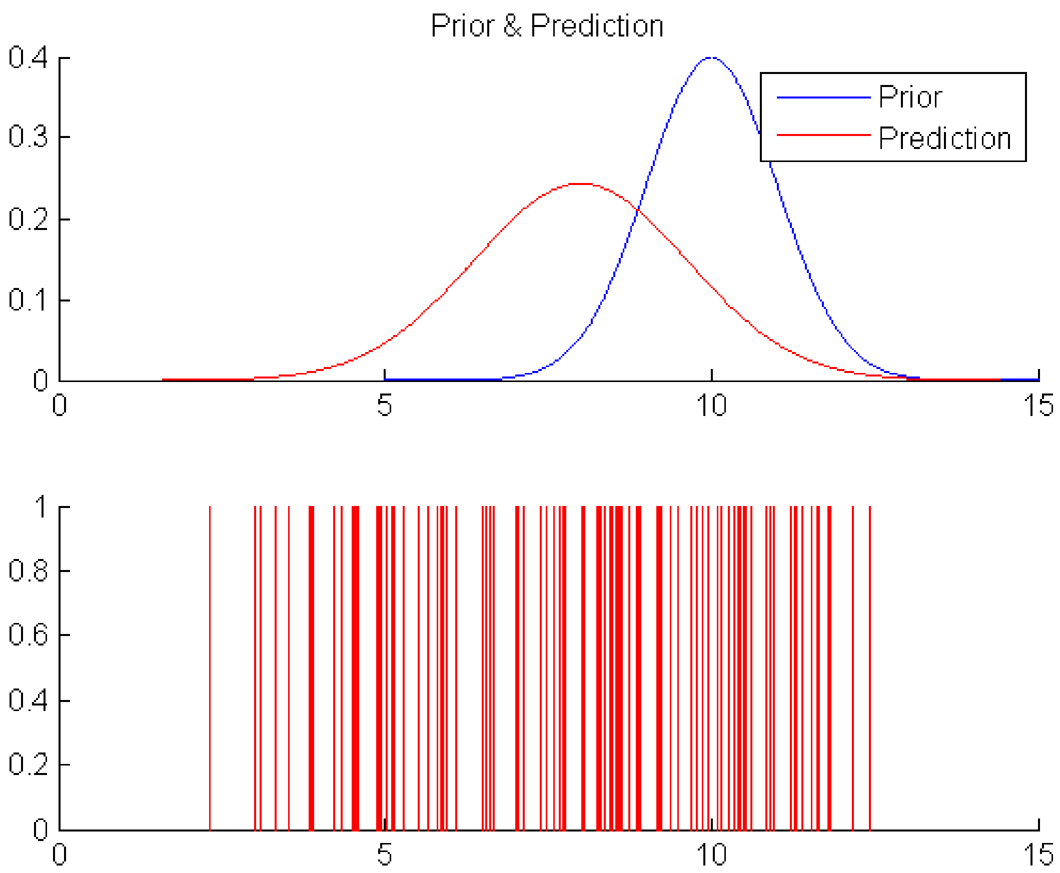
PARTICLE FILTERS

- Priors, comparing KF and Particle filter



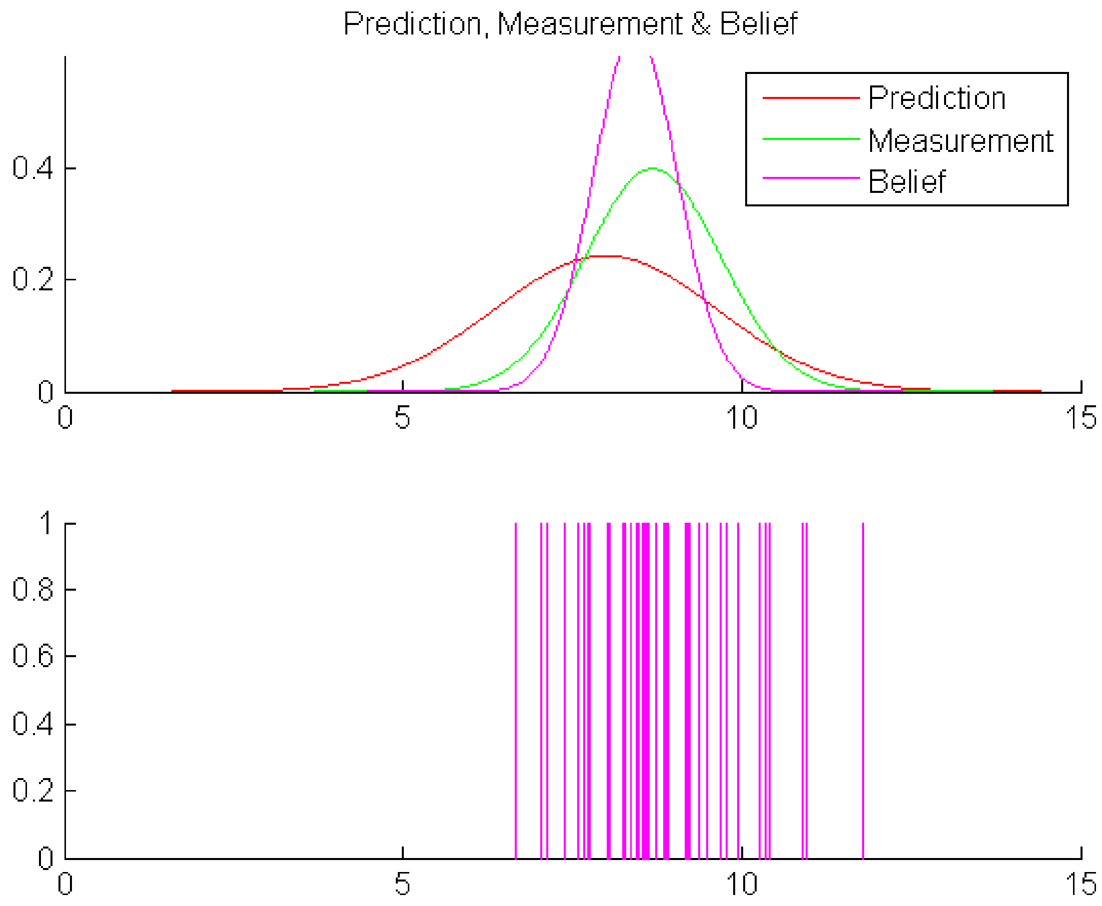
Particle Filters

- Prediction



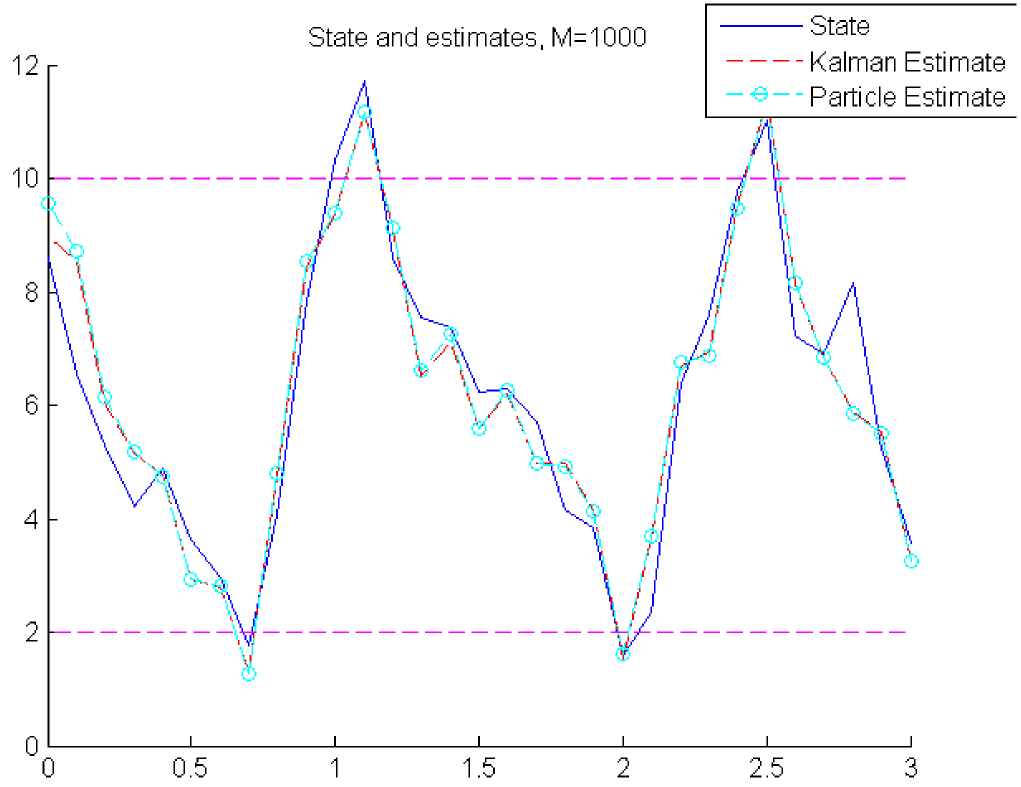
Particle Filters

- Belief



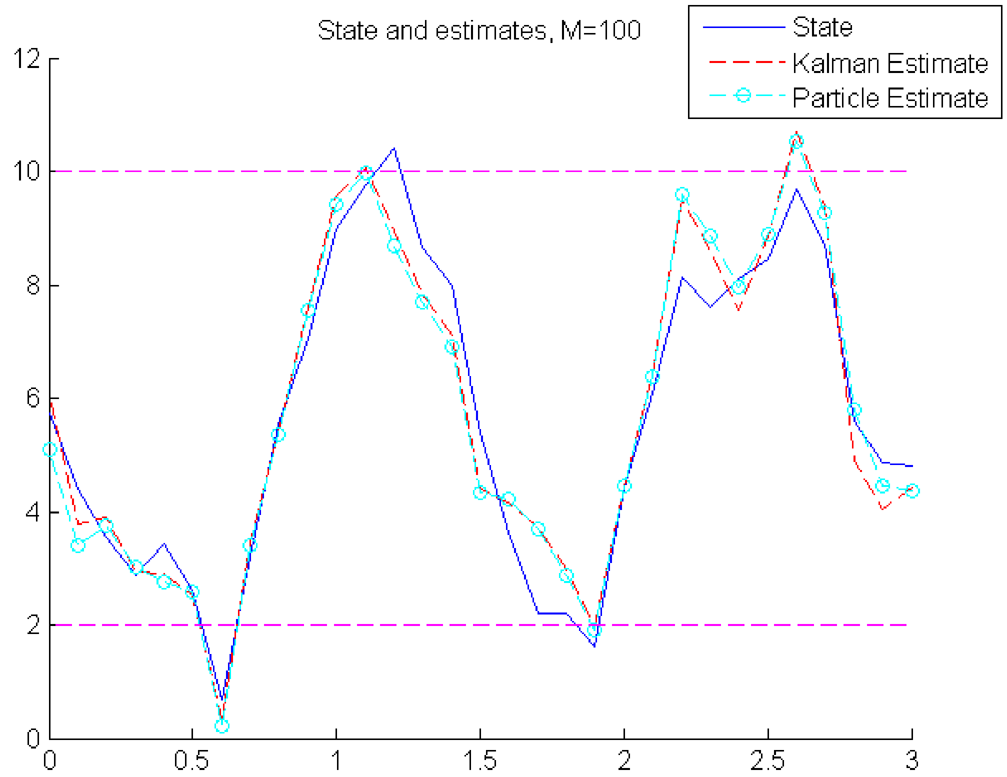
PARTICLE FILTERS

- Comparison of Gaussian parameters
  - KF vs PF (1000, 100, 10)



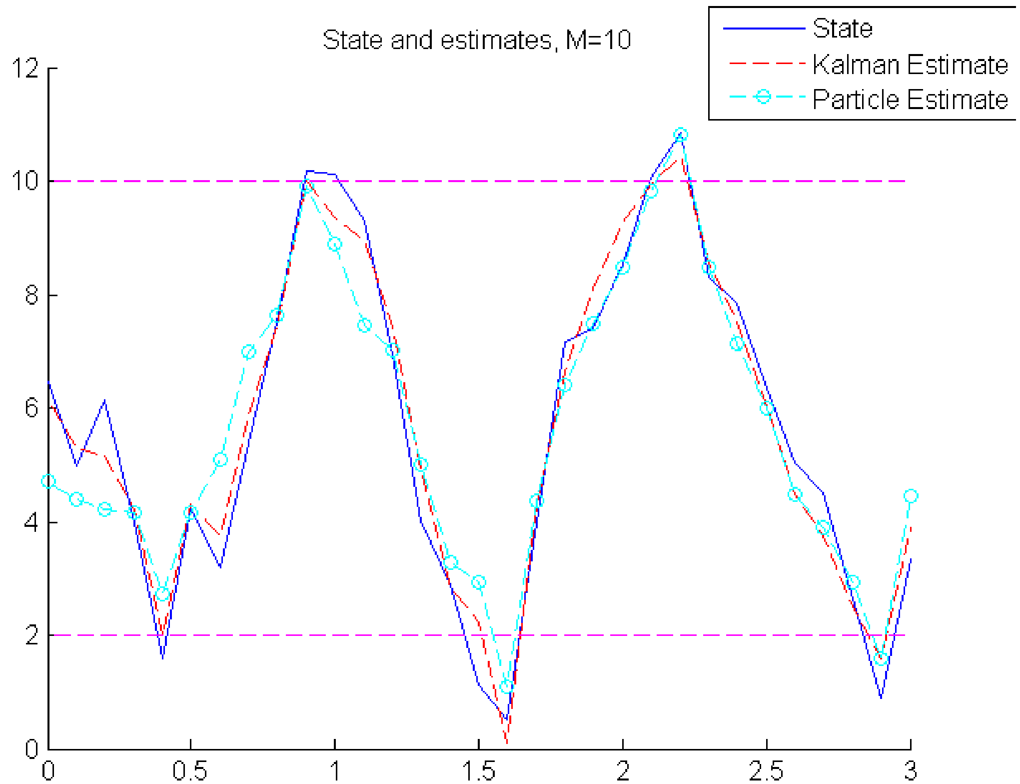
Particle Filters

- Comparison of Gaussian parameters
  - KF vs PF (1000, 100, 10)



Particle Filters

- Comparison of Gaussian parameters
  - KF vs PF (1000, 100, 10)



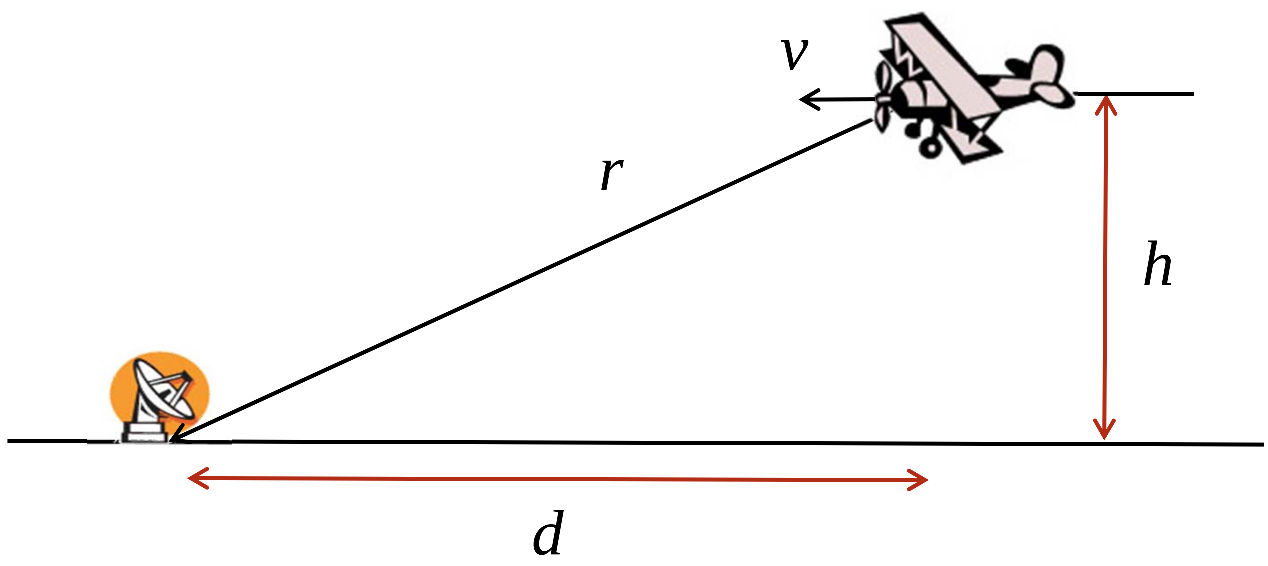
Particle Filters

- Comparison of run times

Algorithm	Run Time
Kalman Filter	0.005164
Particle Filter -10	0.043191
Particle Filter -20	0.06965
Particle Filter -100	0.2188
Particle Filter -1000	1.8740

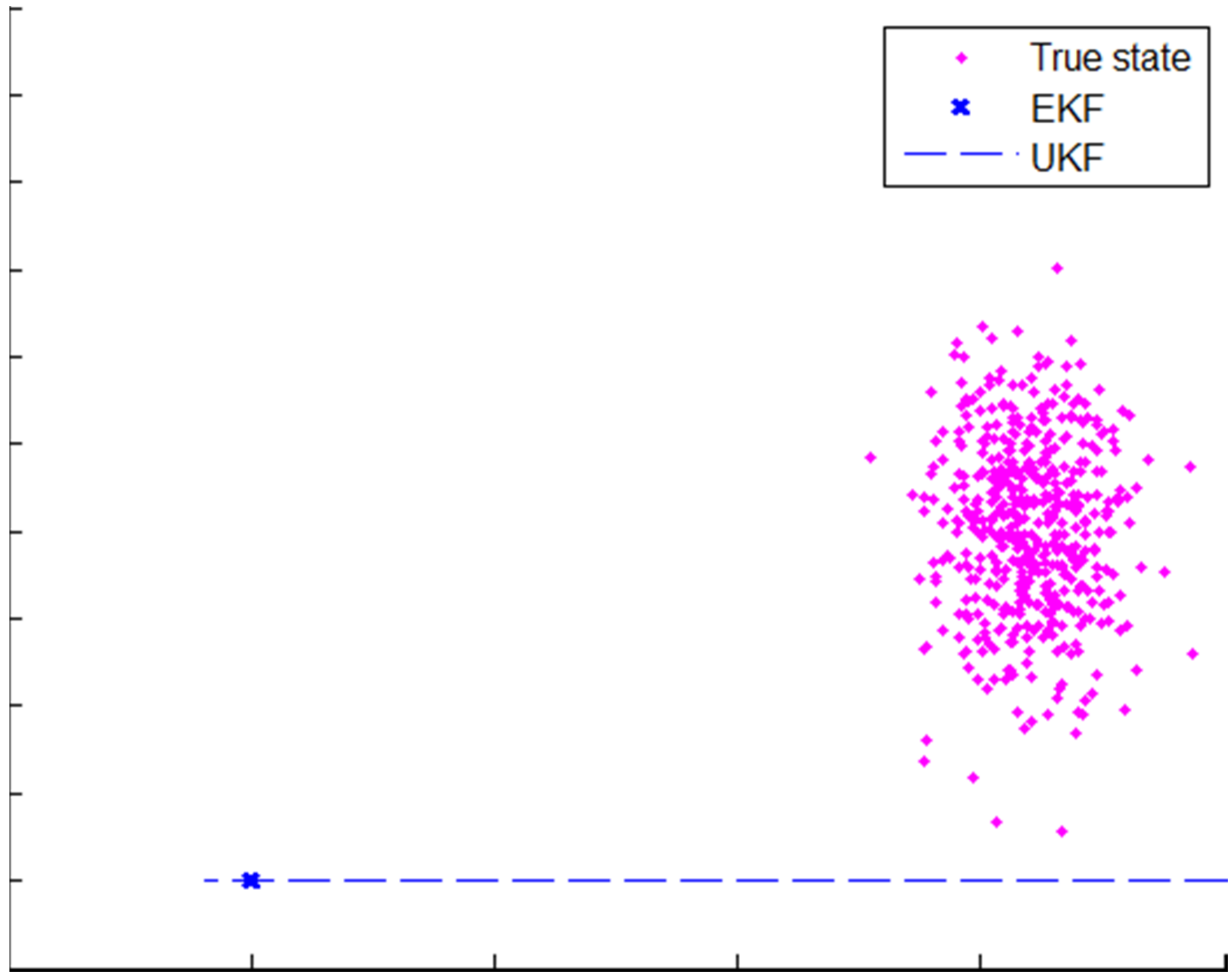
Particle Filters

- EKF/UKF/Particle showdown
  - Aircraft flyover example



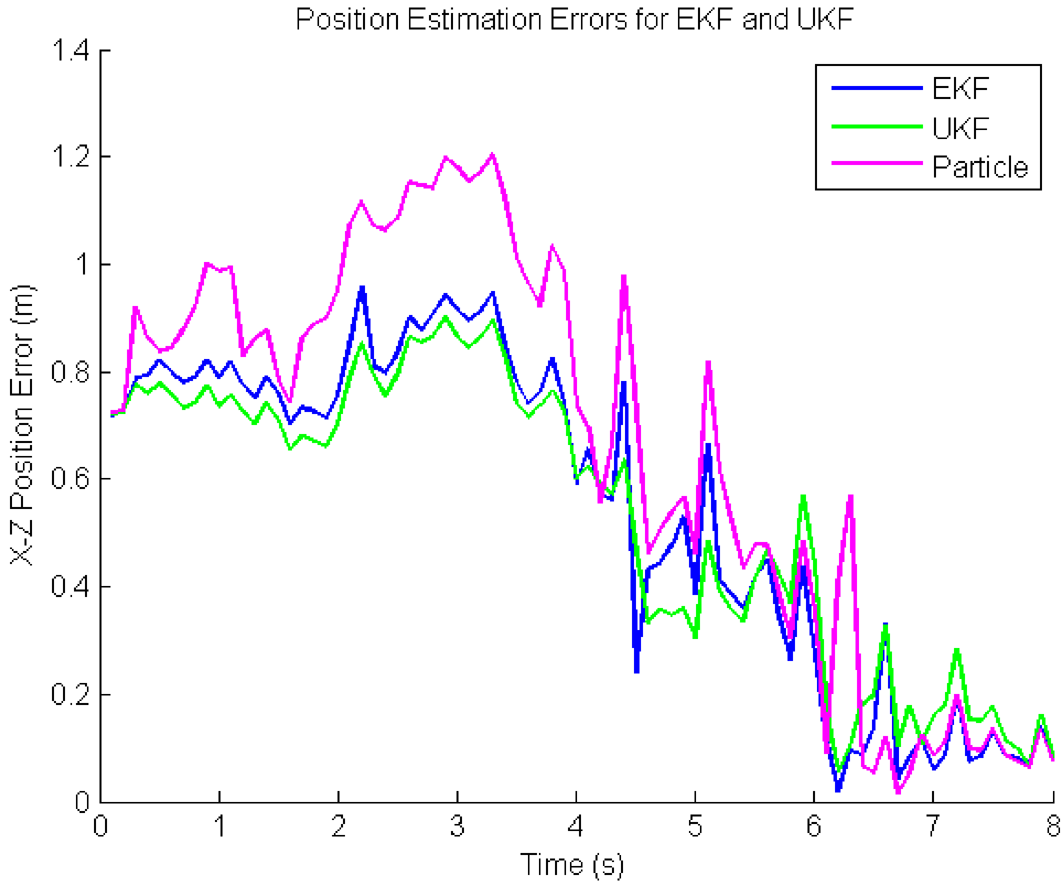
PARTICLE FILTER

- Results



Particle Filter

- Not always better, hard to tune



Particle Filters

- Interpreting particle sets
  - In order to use a particle filter, must somehow extract relevant information from particles
  - Density Extraction
    - Determining a probability density function from a set of particles
      - Gaussian approximation
        - Simply calculate mean and covariance of set
        - Only really useful for unimodal distributions
        - Used most often for control applications
      - K-means algorithm
        - Approximate density with a mixture of K Gaussians
        - Requires clustering of particles
      - Kernel density estimation
        - Use each particle as the center of a continuous kernel function
        - Add all kernels together to generate a pdf
        - Linear in the number of particles

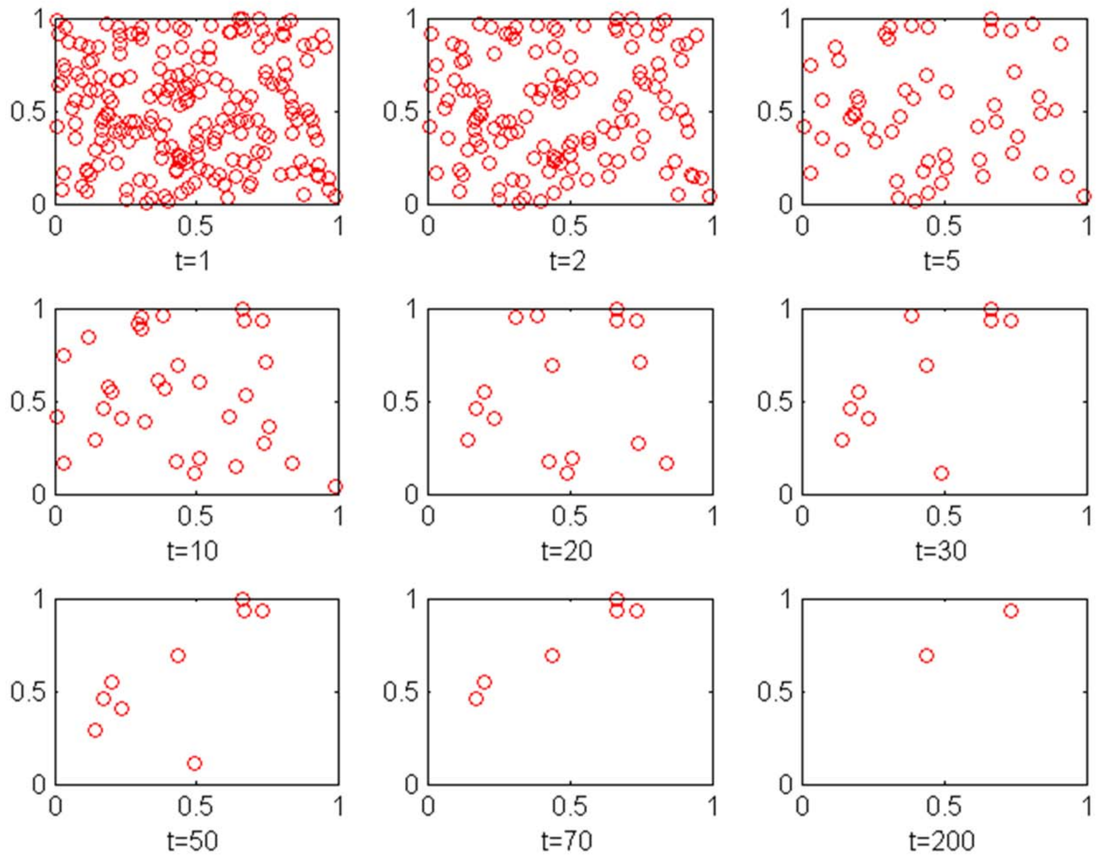
Particle Filters

- Sample Variance
  - Since continuous distributions are approximated by a discrete set of samples, errors occur
  - Each time a particle filter is run (with random sampling) a different particle set will result

  - Extreme case:
    - No motion $x_t = x_{t-1}$
    - No measurements, uniform weights on each particle
    - Uniform prior over 2D space
    - What will happen to the particle set as we update the particle filter?
      - Essentially repeating the resampling step with uniform weight on all particles.

Particle Filters

- Example
  - Particle deprivation



Particle Filters

- Excessive resampling can lead to particle deprivation
  - Motion sampling adds variety to particle set
  - Do not resample when no motion occurs
    - Instead update weights multiplicatively for each measurement
  - If problems arise
    - Apply low variance sampling
    - Artificially disperse samples as well
    - Add random samples after resampling
  - Referred to as variance reduction
    - Reducing the variance in the particle set approximation

Particle Filters

- Summary
  - Use particle sets instead of parameterizations to represent distributions
  - Inherently an approximation, introduces errors
  - Propagate samples through motion model by sampling from model distribution
  - Weight samples using measurement probability given sample state as true state
  - Define belief distribution through samples and weights (particles) or post resampling
  - Many extensions, nuances, issues, advanced techniques