Non-parametric Filters: Particle Filters

Particle Filter

- Kalman-like filter – all densities are Gaussian
- histogram filter – represent density as histogram over the entire domain of the state
- particle filter – represent density as a (large) set of samples drawn from the density
  - samples are called particles

$$\chi_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

  - each particle $x_t^{[m]}, 1 \le m \le M$, is a concrete instantiation of the state at time t

Particle Filter

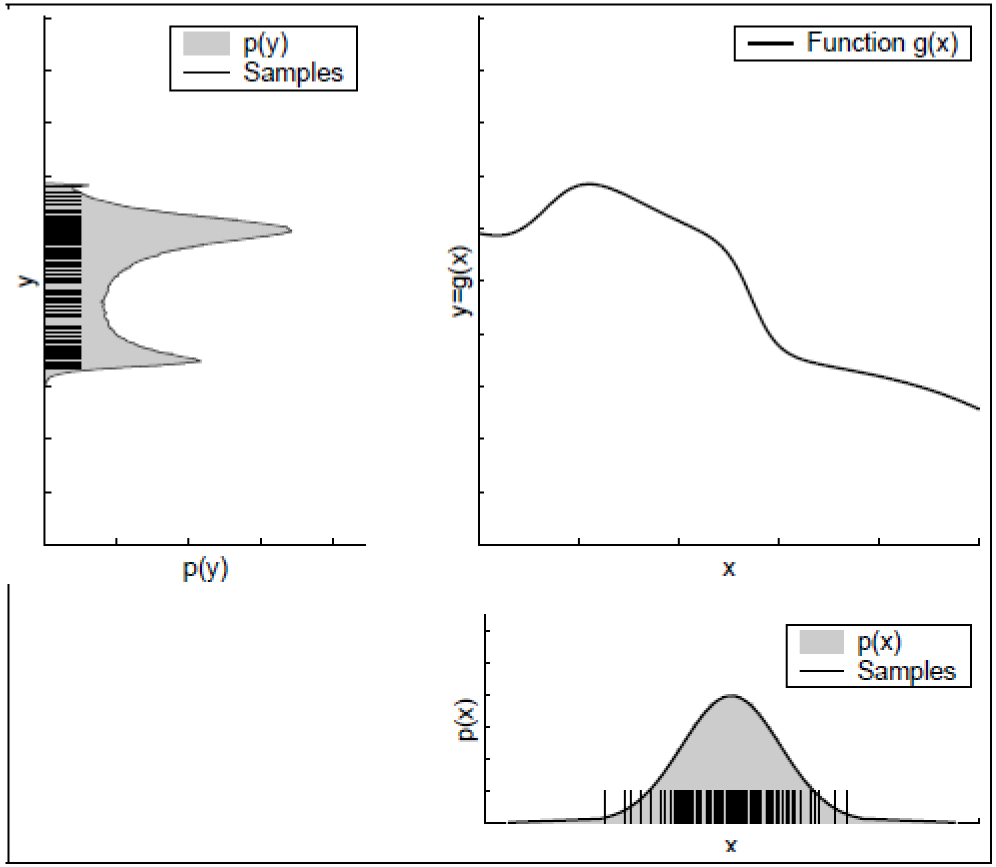


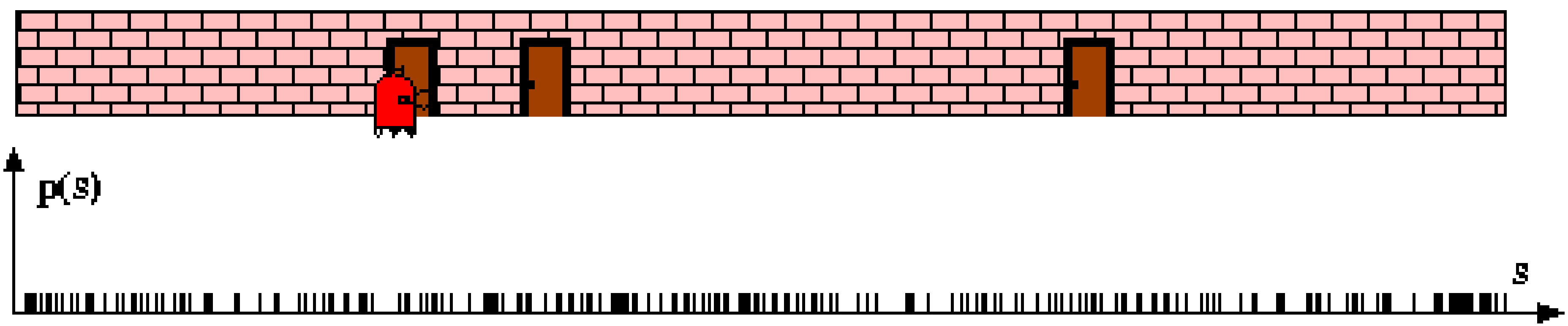
Figure 4.3 The "particle" representation used by particle filters. The lower right graph shows samples drawn from a Gaussian random variable, X. These samples are passed through the nonlinear function shown in the upper right graph. The resulting samples are distributed according to the random variable Y.

Particle Filter Localization

- consider a robot moving down a hall equipped with a sensor that measures the presence of a door beside the robot
  - the pose of the robot is simply its location on a line down the middle of the hall
  - the robot starts out having no idea how far down the hallway it is located
  - robot has a map of the hallway showing it where the doors are
- very similar (and very well done) example here:
  - https://www.youtube.com/watch?v=aUkBa1zMKv4

Particle Filter Localization

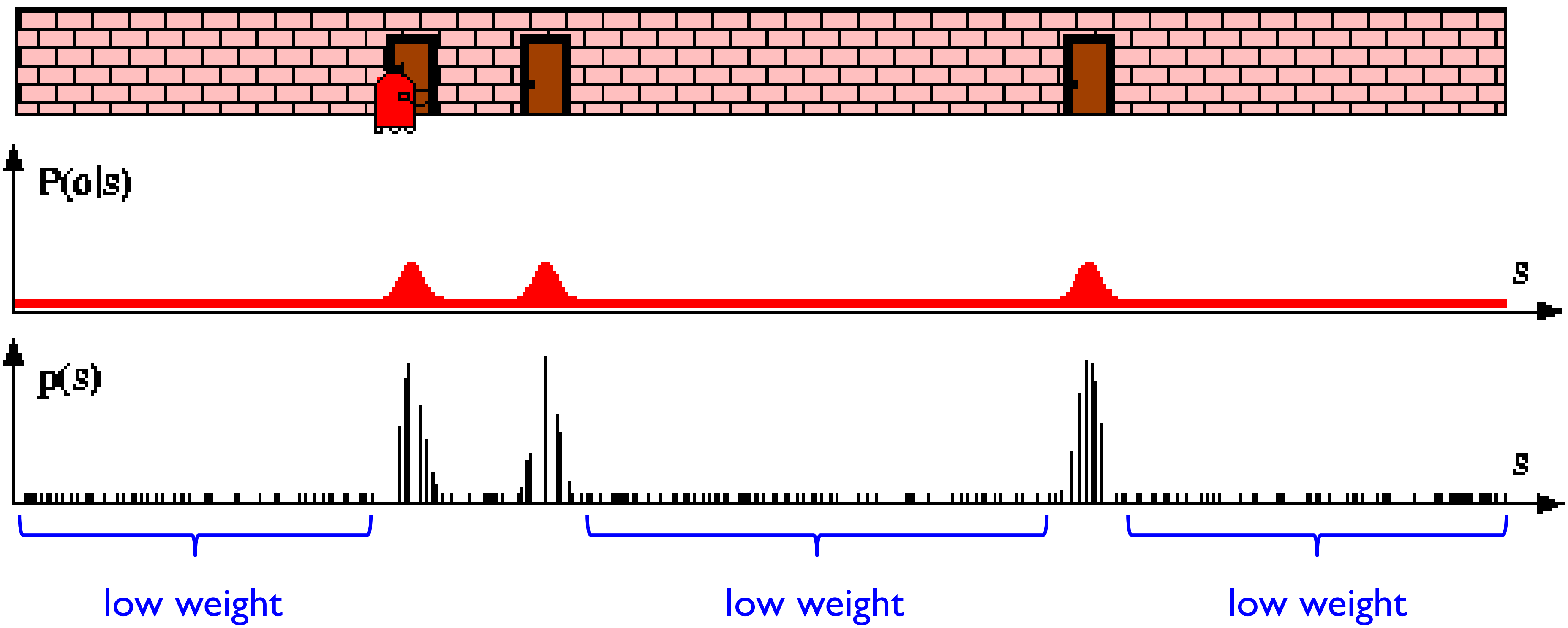
- the robot starts out having no idea how far down the hallway it is located
  - particles *with equal weights* are randomly drawn from a uniform state density



• height of particle is proportional to its weight
• the weights are called *importance weights*

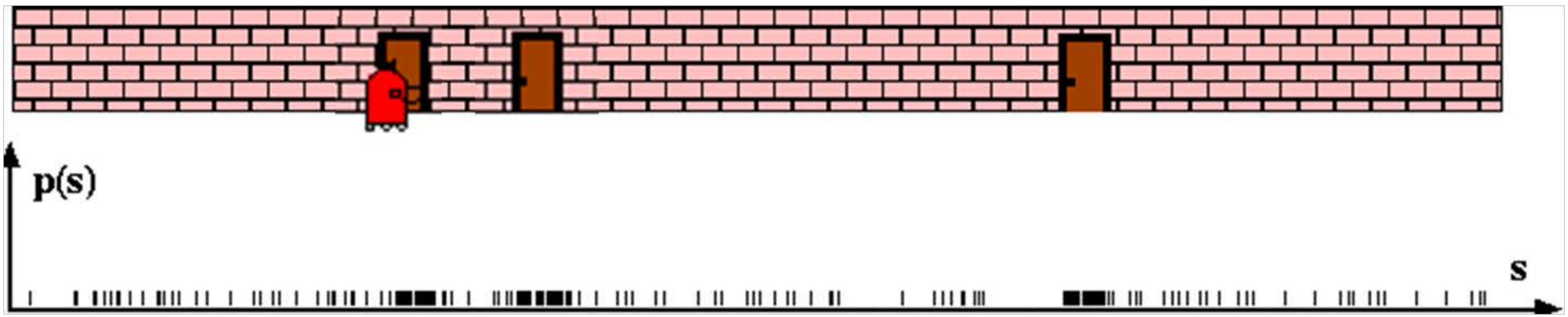
Particle Filter Localization

- because the robot is beside a door, it has a measurement
  - it can incorporate this measurement into its state estimate
  - particles are reweighted based on how consistent each particle is with the measurement



Particle Filter Localization

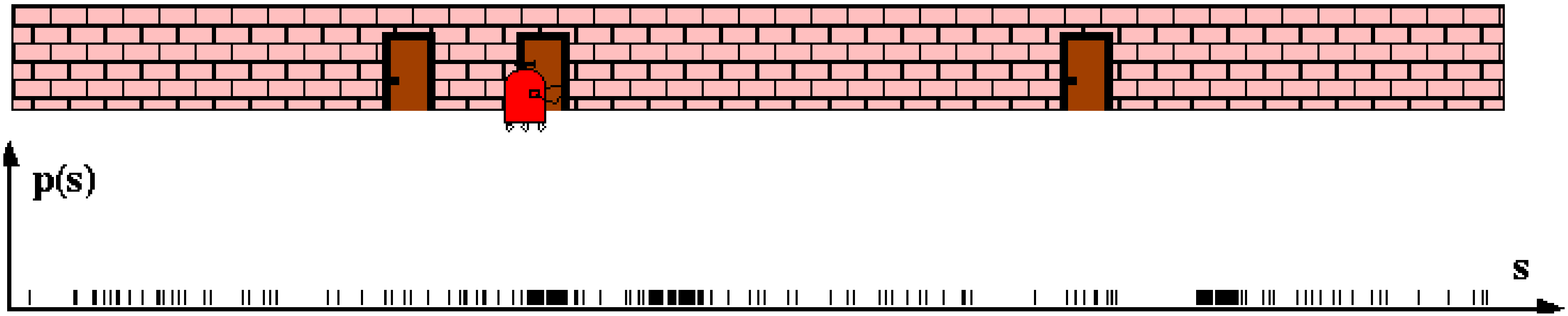
- the existing particles are resampled with replacement where the probability of drawing a particle is proportional to its importance weight



- resampling produces a set of particles with equal importance weights that approximates the density
- the resampled set usually contains many duplicate particles (those with high importance weights)
- the resampled set will be missing many particles from the original set (those with low importance weights)

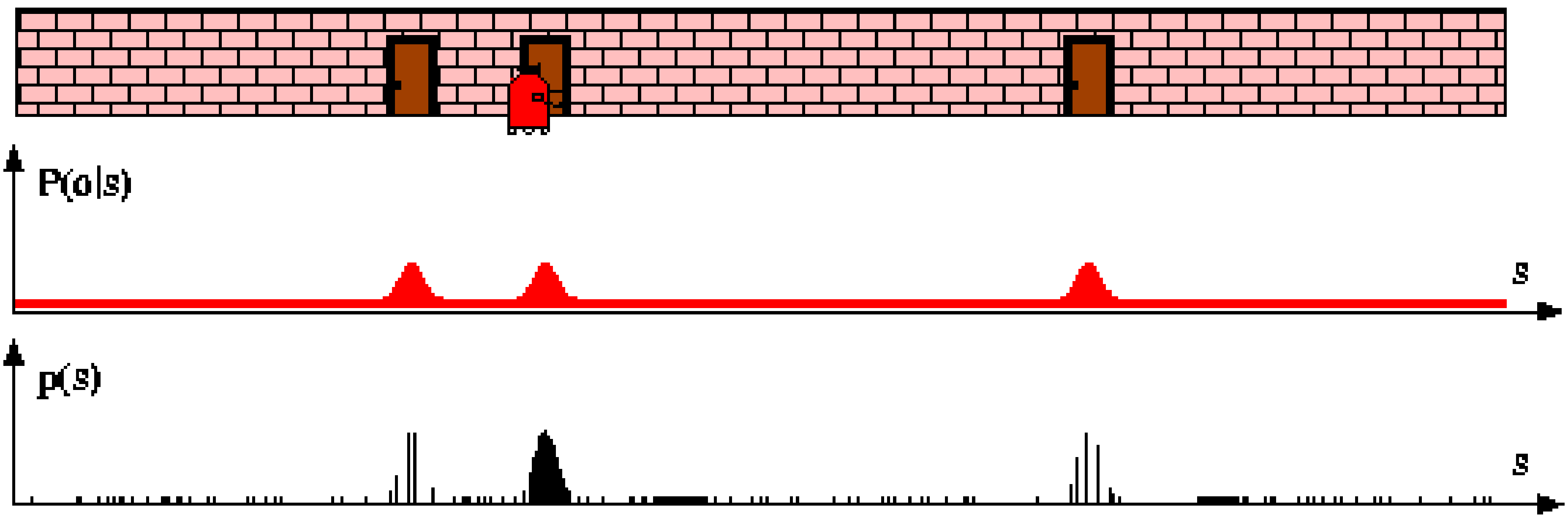
Particle Filter Localization

- the particles are projected forward in time using the motion model



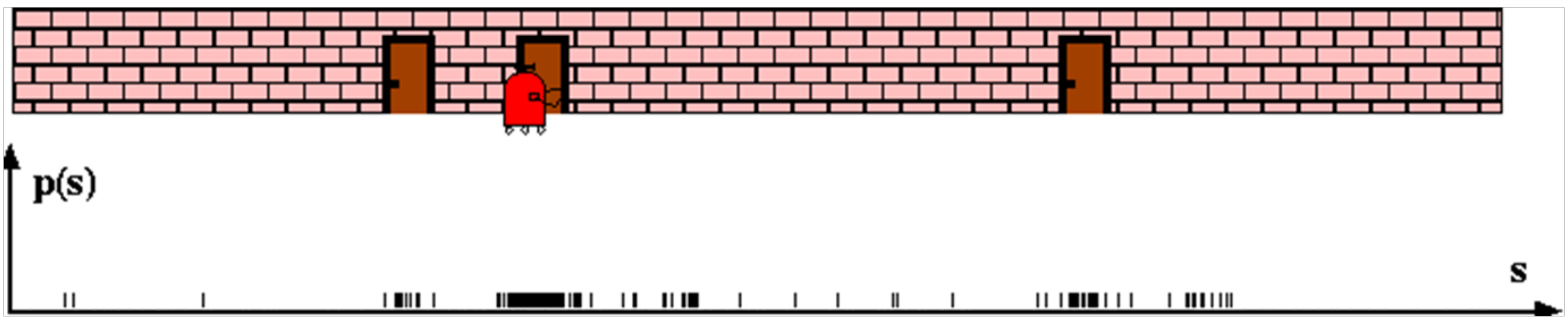
Particle Filter Localization

- because the robot is beside a door, it has a measurement
  - it can incorporate this measurement into its state estimate
  - particles are reweighted based on how consistent each particle is with the measurement



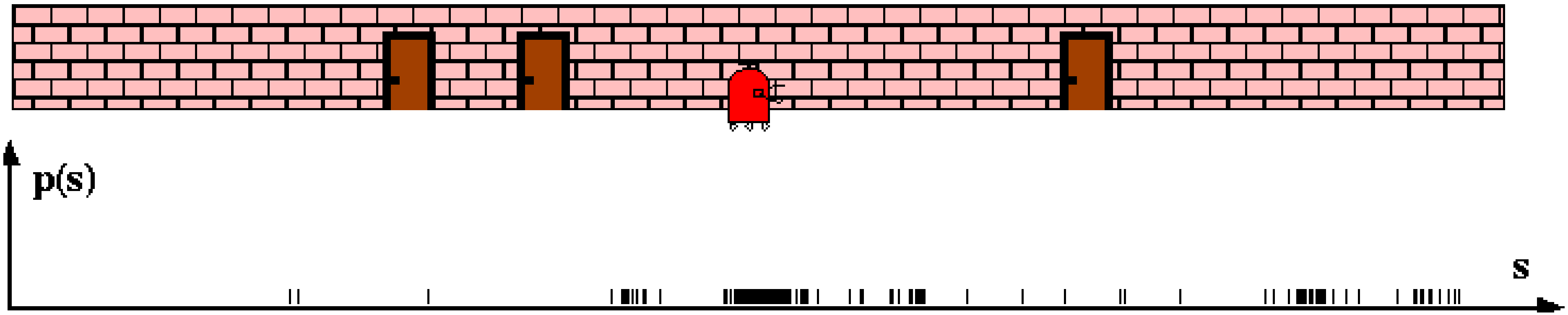
Particle Filter Localization

- the existing particles are resampled with replacement where the probability of drawing a particle is proportional to its importance weight



Particle Filter Localization

- the particles are projected forward in time using the motion model



Particle Filter Localization Algorithm

1. algorithm pf_localization(χ_{t-1}, u_t, z_t, m)
2. $\bar{\chi}_t = \chi_t =$ empty set
3. for $m = 1$ to M
4. $\quad x_t^{[m]} =$ sample_motion_model($u_t, x_{t-1}^{[m]}$)
5. $\quad w_t^{[m]} =$ measurement_model($z_t, x_t^{[m]}, m$)
6. $\bar{\chi}_t = \bar{\chi}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
7. endfor
8. $\chi_t =$ resample ($\bar{\chi}_t$)
9. return χ_t

χ_t	set of particles
u_t	control input
z_t	measurement
m	map

Resampling Algorithm

1. algorithm resample($\bar{\chi}$)
2. for $m = 1$ to M
3. draw i with probability $\propto w^{[m]}$
4. add $x^{[i]}$ to χ
5. endfor
6. return χ

Drawing Particles

		compute this	then this
i	**importance weights**	**cumulative sum**	**normalized sum**
1	0.0846	0.0846	0.0235
2	0.0769	0.1615	0.0449
3	0.0895	0.2510	0.0698
4	0.4486	0.6995	0.1945
5	0.9505	1.6500	0.4588
6	0.6019	2.2519	0.6262
7	0.1720	2.4239	0.6740
8	0.2853	2.7092	0.7534
9	0.0301	2.7393	0.7618
10	0.8567	3.5960	1.0000

then generate M random number uniformly distributed between 0 and 1

Drawing Particles

find the first normalized sum
entry that this is less than

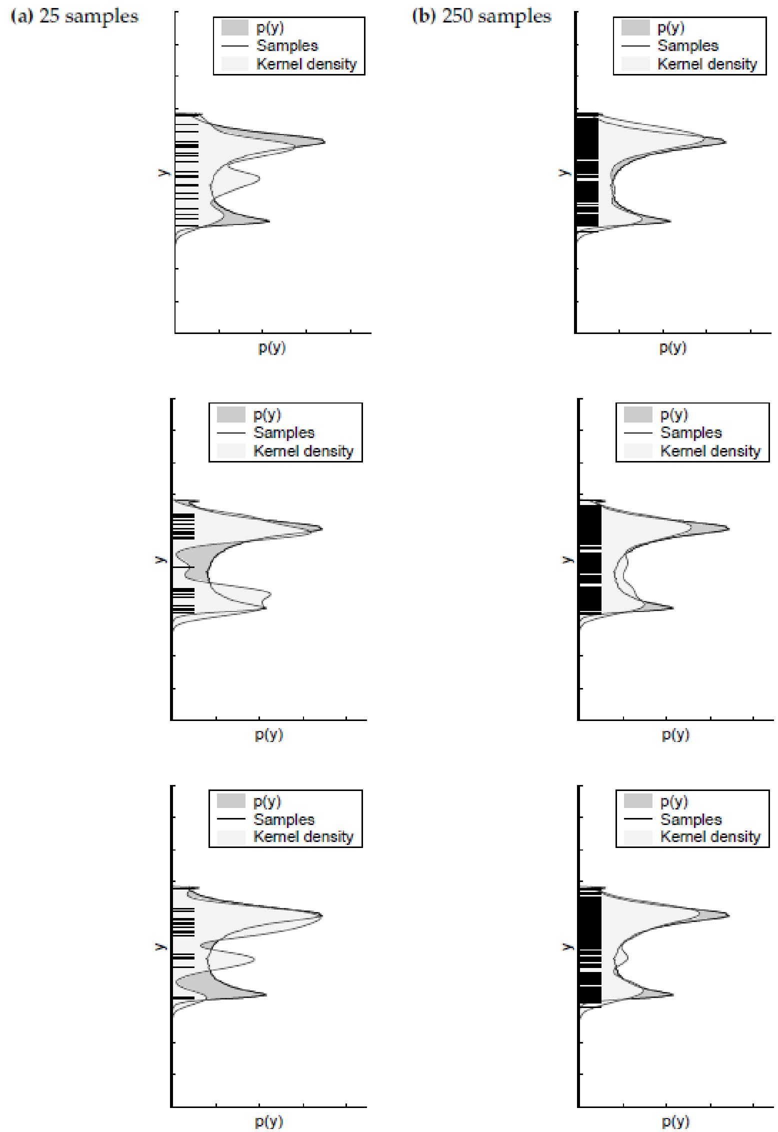
i	importance weights	cumulative sum	normalized sum	random numbers	particle
1	0.0846	0.0846	0.0235	0.5261	6
2	0.0769	0.1615	0.0449	0.5154	6
3	0.0895	0.2510	0.0698	0.8847	10
4	0.4486	0.6995	0.1945	0.0286	2
5	0.9505	1.6500	0.4588	0.3836	5
6	0.6019	2.2519	0.6262	0.5928	6
7	0.1720	2.4239	0.6740	0.4528	5
8	0.2853	2.7092	0.7534	0.3306	5
9	0.0301	2.7393	0.7618	0.5034	6
10	0.8567	3.5960	1.0000	0.7134	8

- this algorithm is known as "roulette wheel sampling/selection"
- inefficient as it requires generating M random numbers and M binary searches
- "stochastic universal sampling" is often used instead

Sampling Variance

- an important source of error in the particle filter is the variation caused by random sampling
- whenever a finite number of samples is drawn from a probability density, the statistics extracted from the samples will differ slightly from the statistics of the original density
  - e.g., if you draw 2 samples from a 1D Gaussian and compute the mean and variance you will probably get a different mean and variance from the original probability density
    - however, if you draw 100 samples then the mean and variance will probably be very close to the correct values

Sampling Variance



Resampling Issues

- there are many issues related to resampling and how to perform good resampling
- notice that resampling as we have described it causes some particles to be eliminated and some to be duplicated
  - continuous resampling will eventually cause all of the particles to be duplicates of a small number of states
  - some PF implementations will add a small amount of noise to the particles so that they are not exact duplicates

Particle Deprivation

- it may happen that there are no particles near the correct state
  - this can happen because of the variance in random sampling
    - an unlucky series of random numbers can wipe out all of the particles near the correct state
  - when this occurs the filter estimate can become arbitrarily incorrect
- occurs mostly when the number of particles is too small for the dimensionality of the state