CSE-571
Robotics

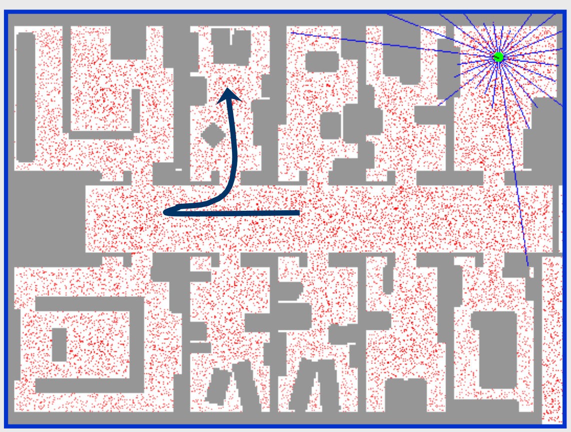
Bayes Filter Implementations

Particle filters

Motivation

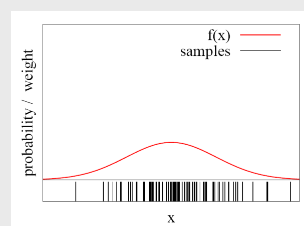
- So far, we discussed the
  - Kalman filter: Gaussian, linearization problems
- Particle filters are a way to **efficiently** represent **non-Gaussian distributions**
- Basic principle
  - Set of state hypotheses (“particles”)
  - Survival-of-the-fittest

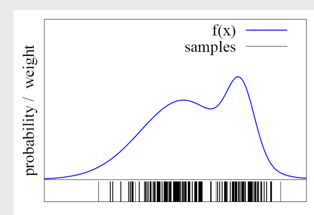
Sample-based Localization (sonar)



Density Approximation

- Particle sets can be used to approximate densities

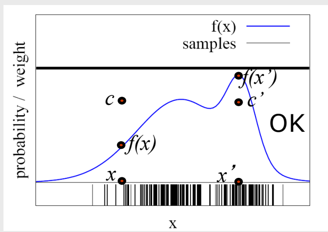




- The more particles fall into an interval, the higher the probability of that interval
- How to draw samples form a function/distribution?

Rejection Sampling

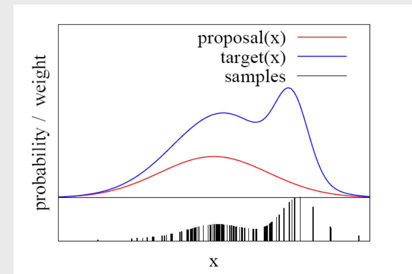
- Let us assume that $f(x)<=1$ for all x
- Sample x from a uniform distribution
- Sample c from [0,1]
- if $f(x) > c$ keep the sample
  otherwise reject the sampe



5

Importance Sampling Principle

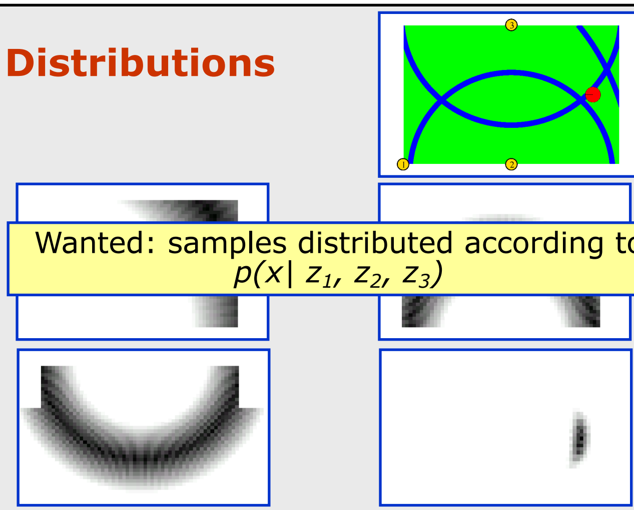
- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the "differences between g and f"
- $w = f / g$
- f is often called target
- g is often called proposal



6

Importance Sampling with Resampling: Landmark Detection Example

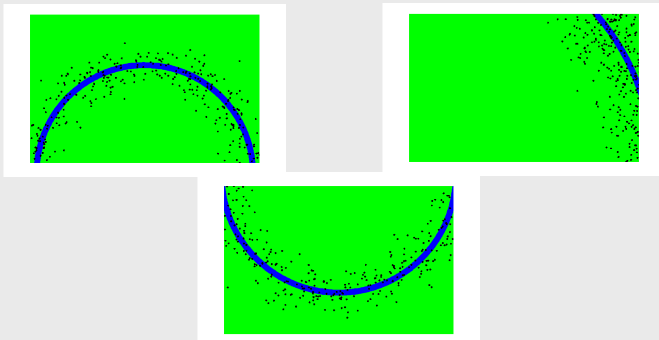
Distributions



Wanted: samples distributed according to $p(x| z_1, z_2, z_3)$

This is Easy!

We can draw samples from $p(x|z_l)$ by adding noise to the detection parameters.



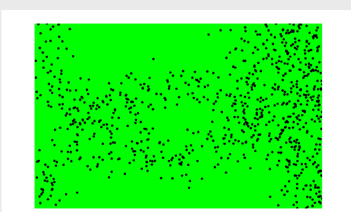
Importance Sampling with Resampling

$$\text{Target distribution f}: p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)}$$

$$\text{Sampling distribution g}: p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)}$$

$$\text{Importance weights w}: \frac{f}{g} = \frac{p(x \mid z_1, z_2, \ldots, z_n)}{p(x \mid z_l)} = \frac{p(z_l) \prod_{k \neq l} p(z_k \mid x)}{p(z_1, z_2, \ldots, z_n)}$$

Importance Sampling with Resampling



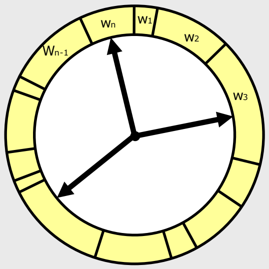
Weighted samples

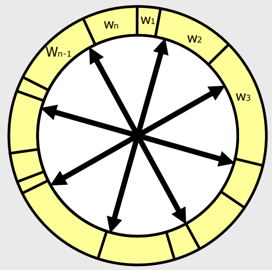
After resampling

Resampling

- **Given**: Set S of weighted samples.
- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.
- Typically done n times with replacement to generate new sample set S'.

Resampling





- Roulette wheel
- Binary search, n log n

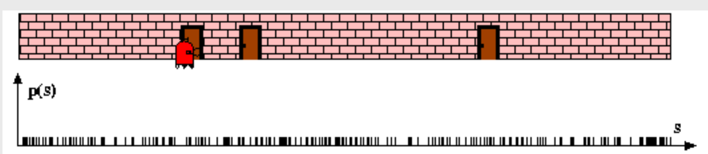
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Resampling Algorithm

1. Algorithm **systematic_resampling**(S,n):
2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ — ***Generate cdf***
4. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ — ***Initialize threshold***
6. **For** $j = 1 \ldots n$ — ***Draw samples …***
7. $\quad$ **While** ($u_j > c_i$) — ***Skip until next threshold reached***
8. $\quad\quad i = i + 1$
9. $\quad S' = S' \cup \{< x^i, n^{-1} >\}$ — ***Insert***
10. $\quad u_j = u_j + n^{-1}$ — ***Increment threshold***
11. **Return** S'

Also called **stochastic universal sampling**

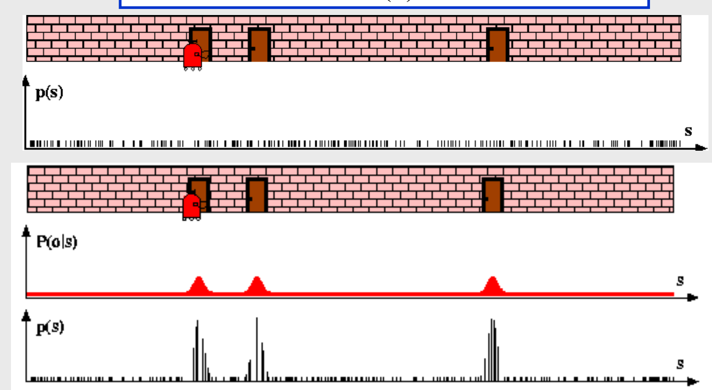
Particle Filters



Sensor Information: Importance Sampling

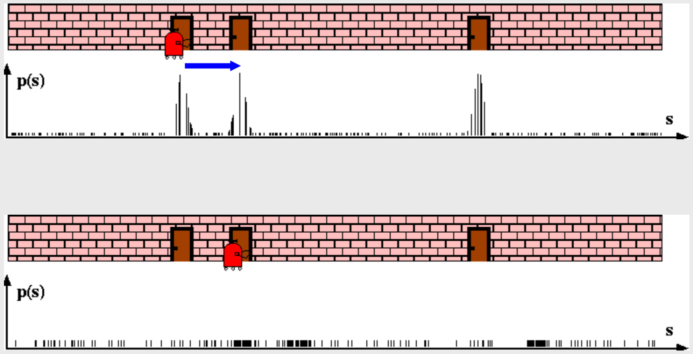
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



Robot Motion

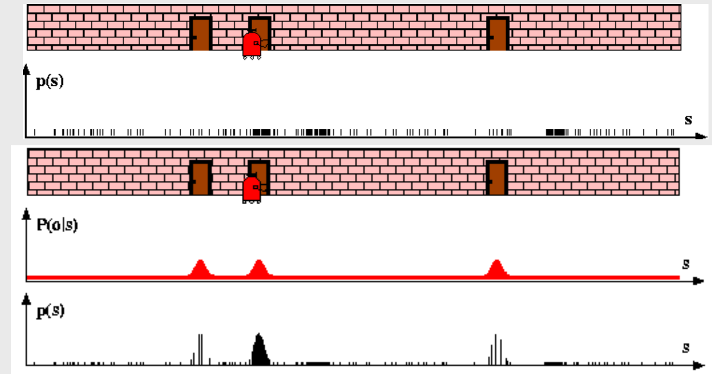
$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



Sensor Information: Importance Sampling

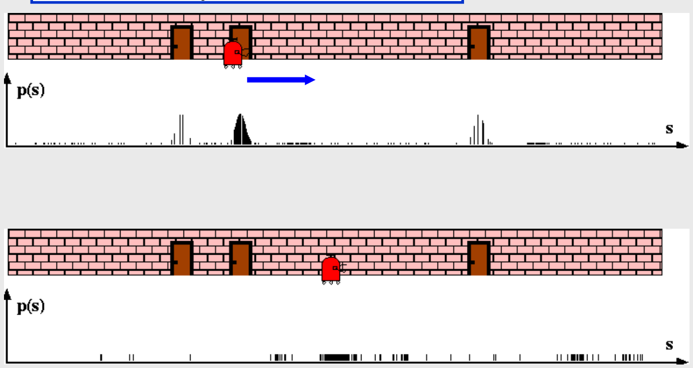
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



Robot Motion

$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Generate new samples***
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weights***

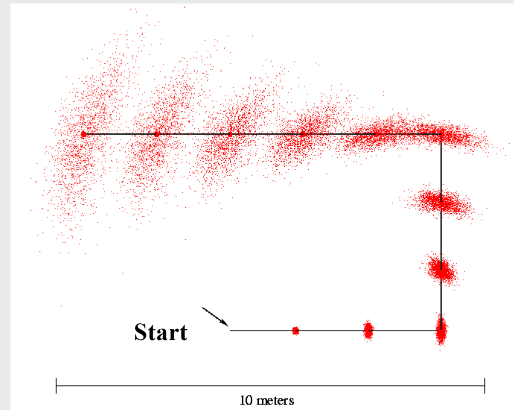
Particle Filter Algorithm

$$Bel\,(x_t) = \eta\; p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})\; dx_{t-1}$$

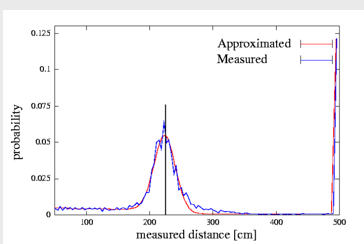
- draw x^i_{t-1} from $Bel(x_{t-1})$
- draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$
- Importance factor for x^i_t:

$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta\; p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})} \propto p(z_t \mid x_t)$$

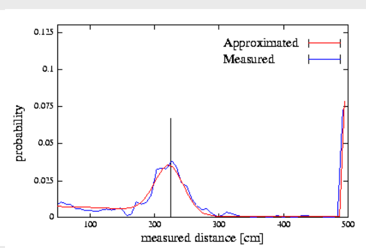
Motion Model Reminder



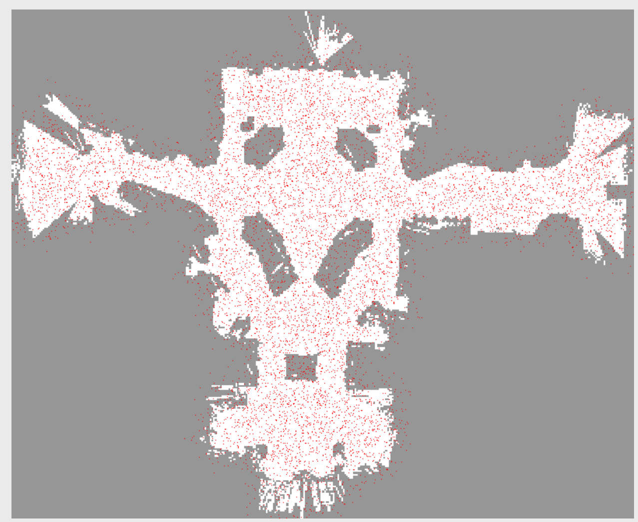
Proximity Sensor Model Reminder



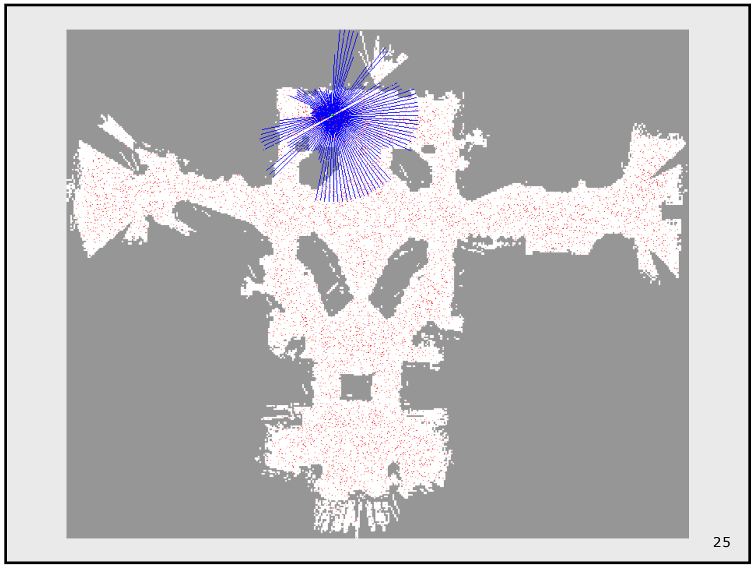
Laser sensor

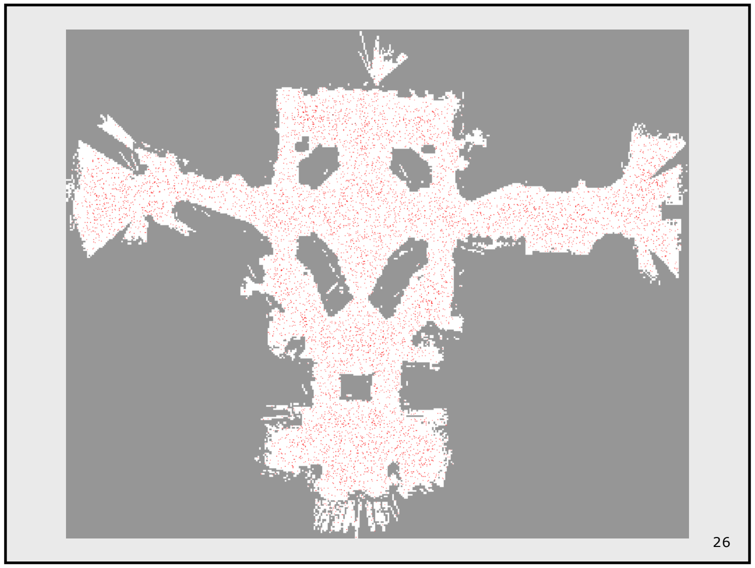


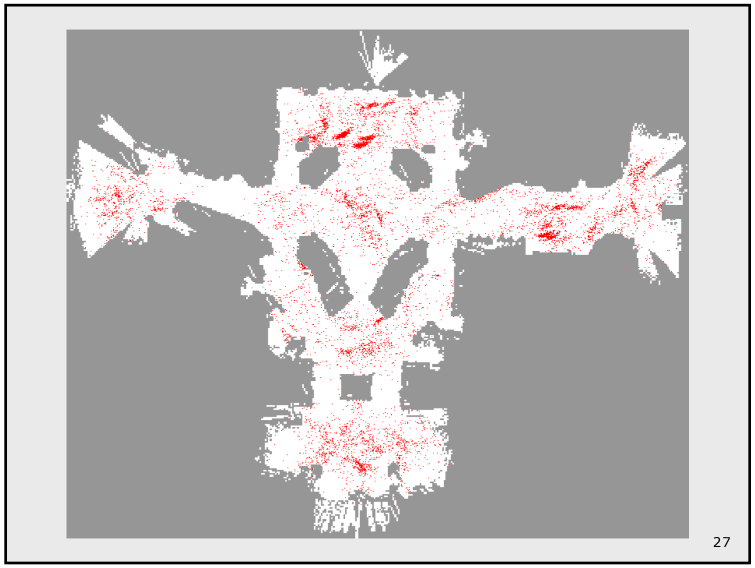
Sonar sensor

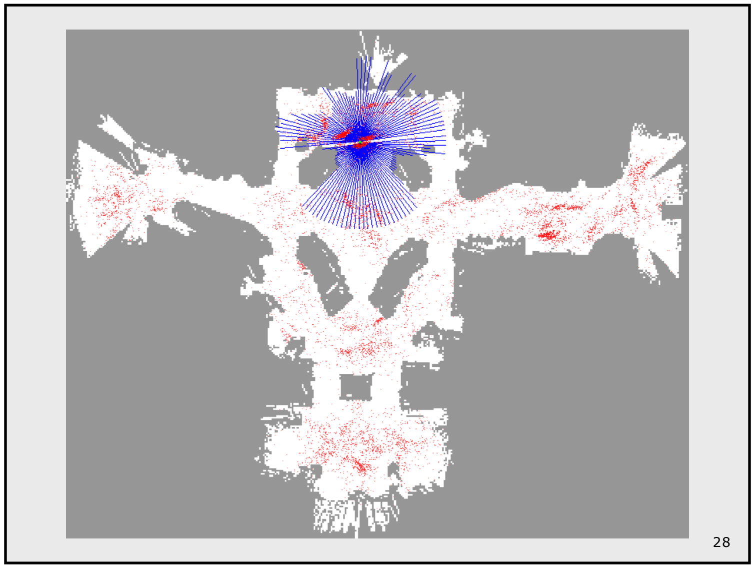


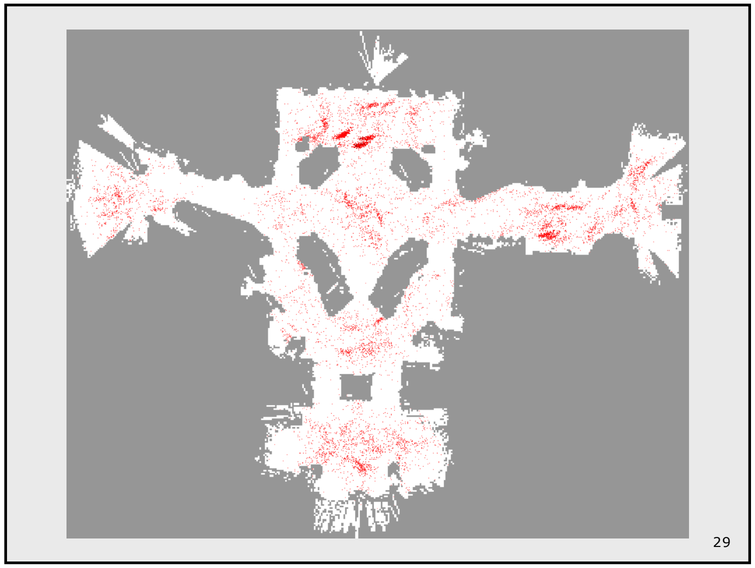
24



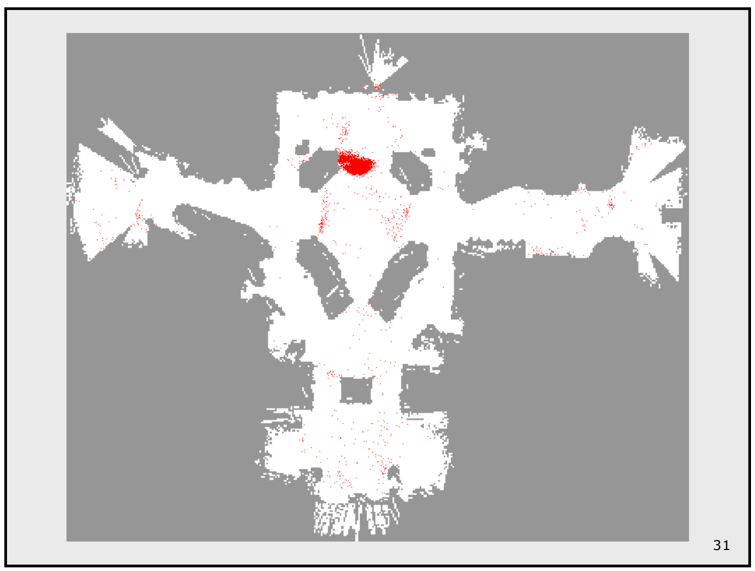


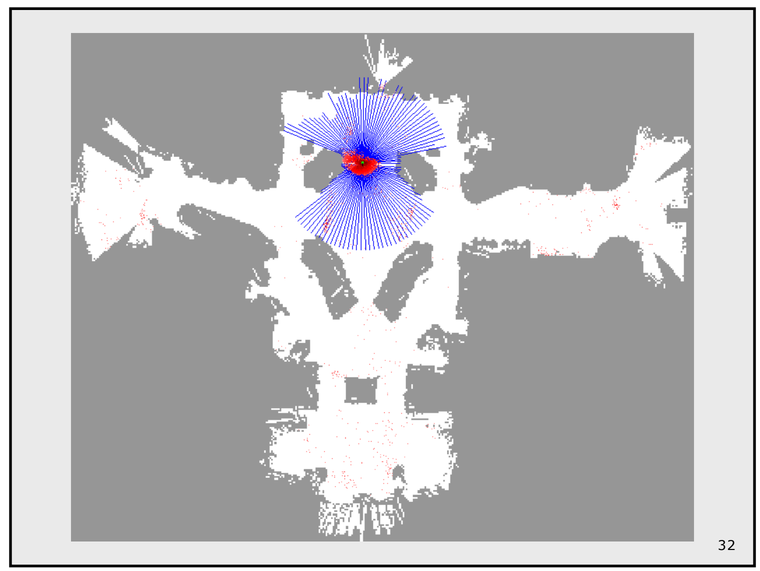


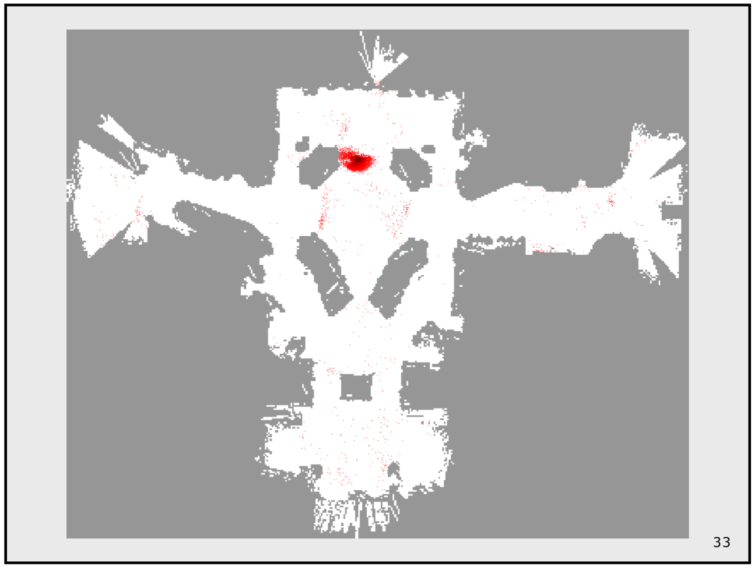


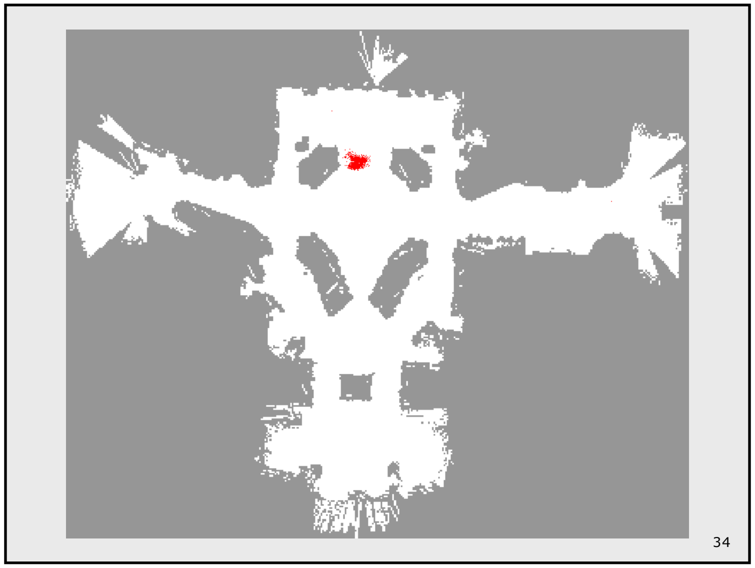

29

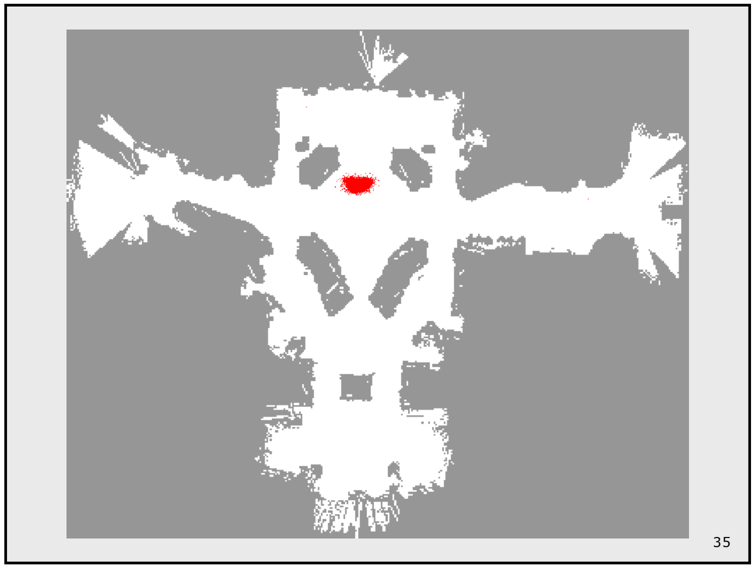

30

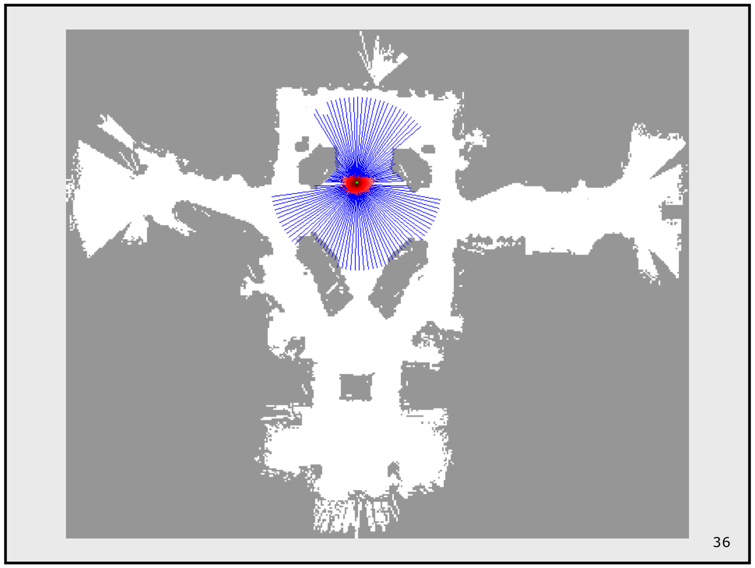

31

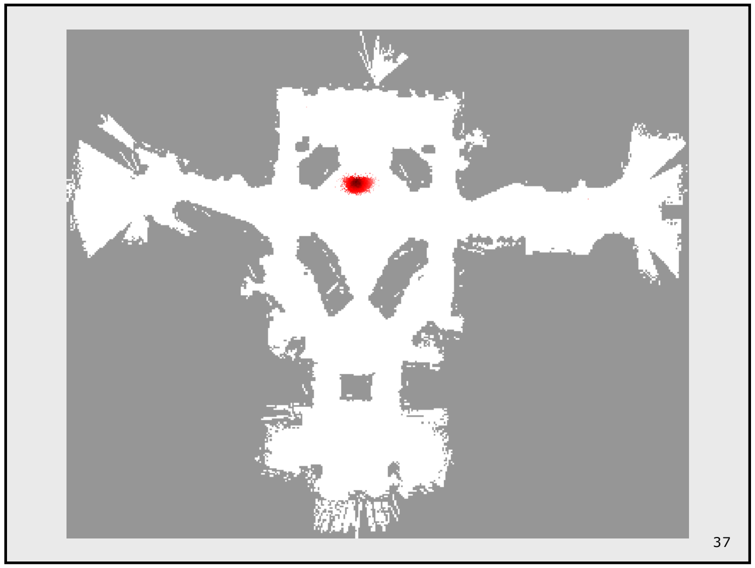

32

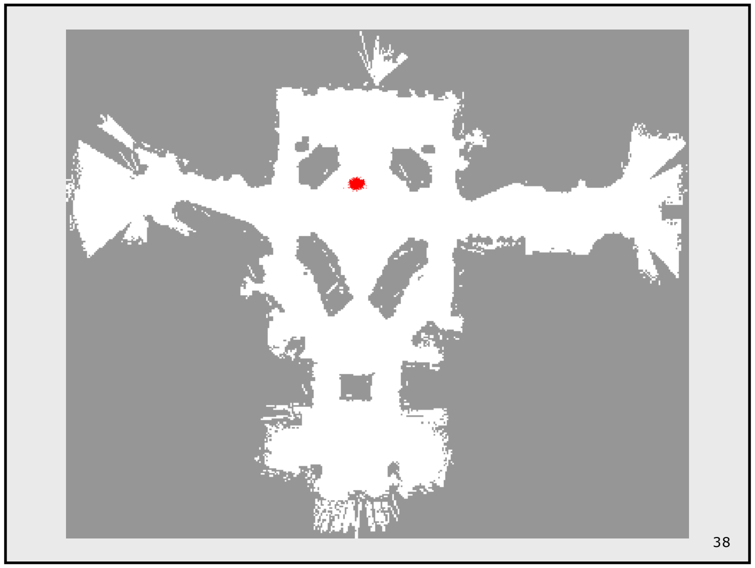

33

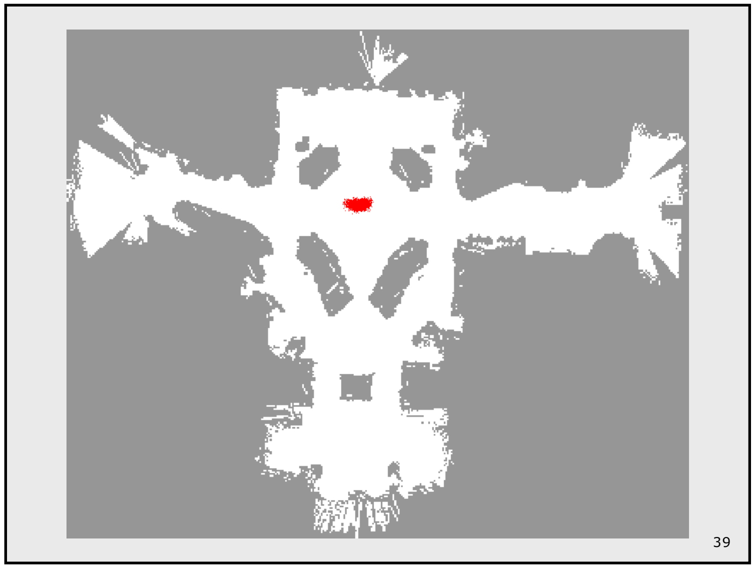

34

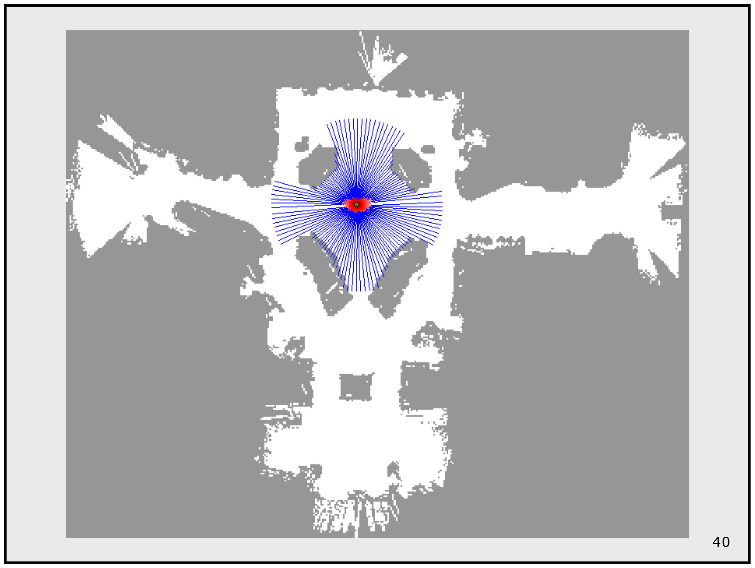

35

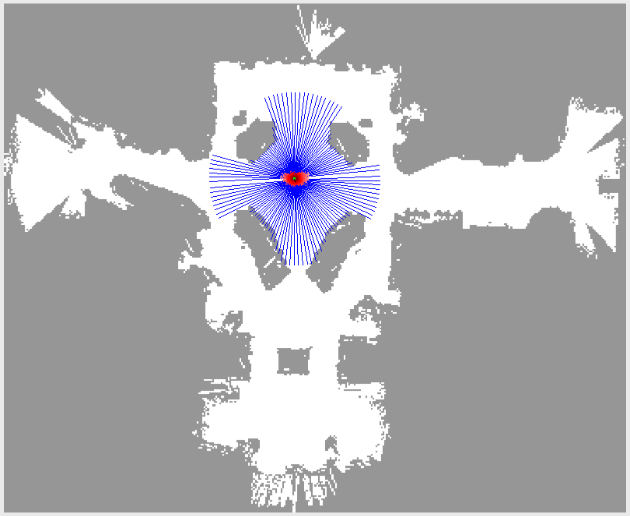

36


37


38

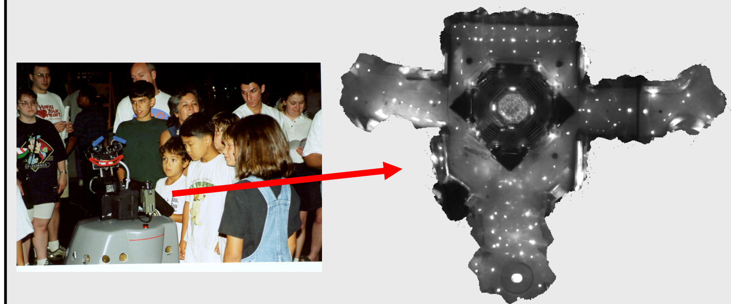

39


40



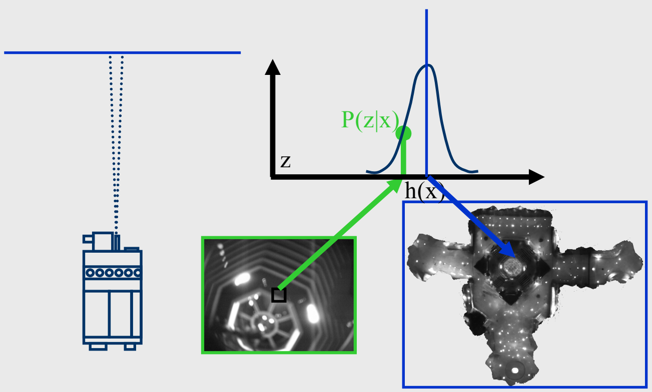
41

Using Ceiling Maps for Localization



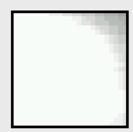
[Dellaert et al. 99]

Vision-based Localization

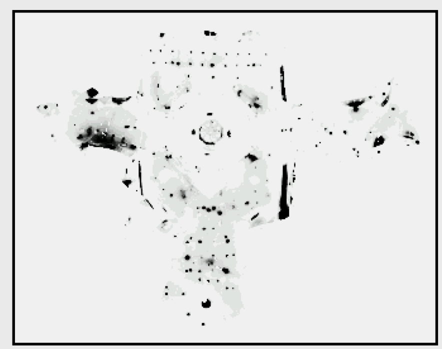


Under a Light

Measurement z:

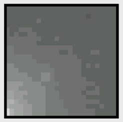


***P(z|x)*:**

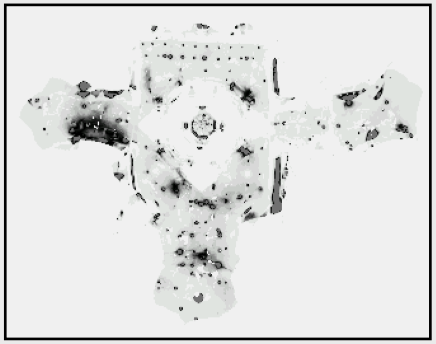


Next to a Light

Measurement z:



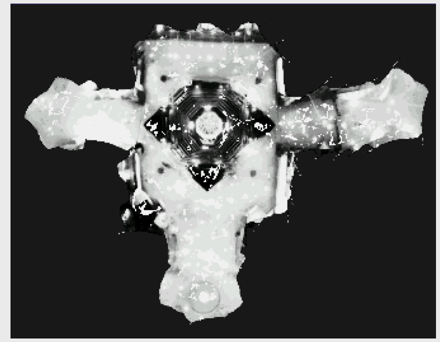
***P(z|x)*:**



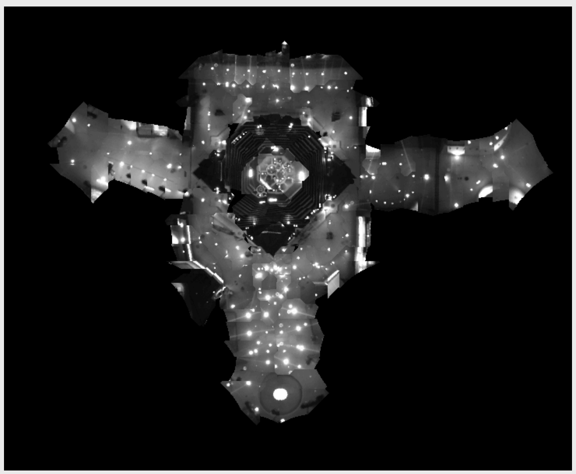
Elsewhere

Measurement z:

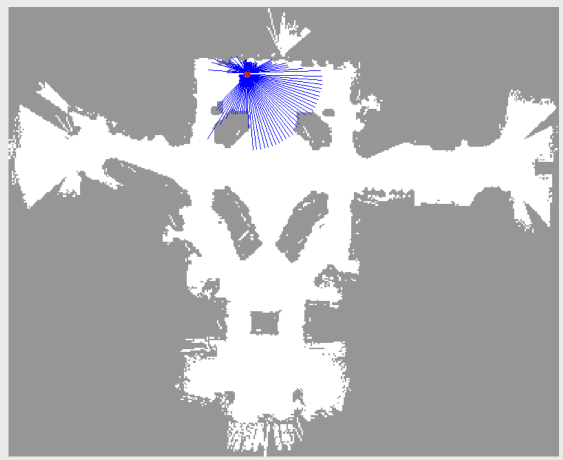
***P(z|x)*:**



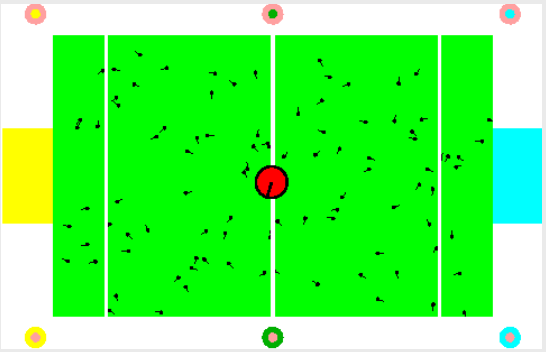
Global Localization Using Vision



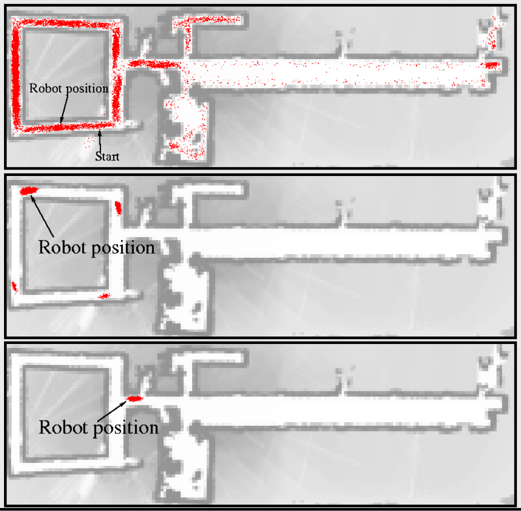
Recovery from Failure



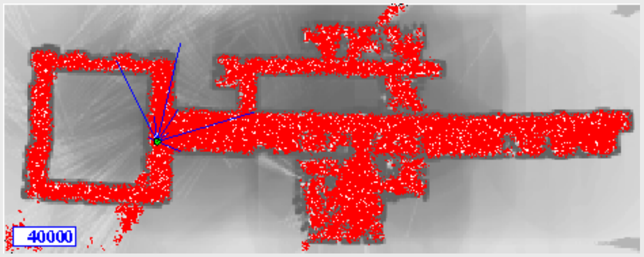
Localization for AIBO robots



Adaptive Sampling

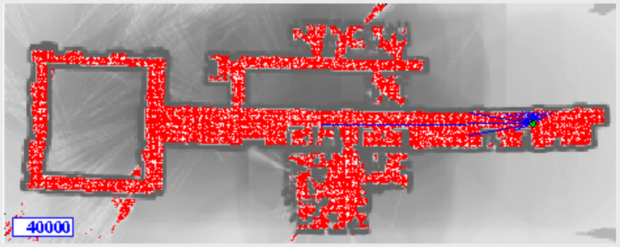


KLD-Sampling Sonar

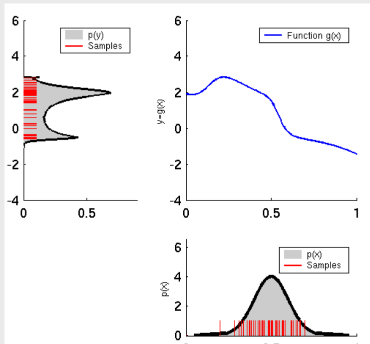


Adapt number of particles on the fly based on statistical approximation measure

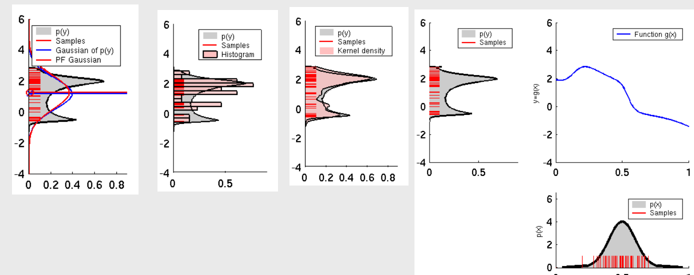
KLD-Sampling Laser



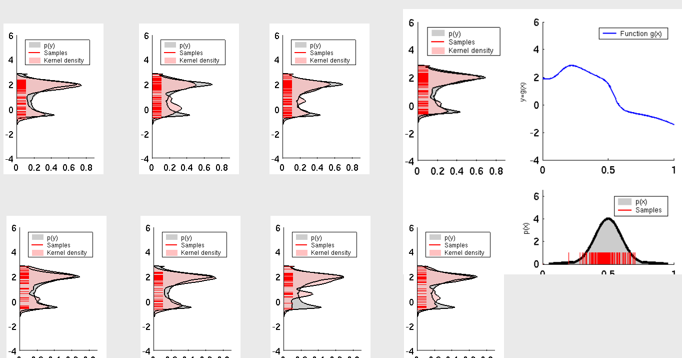
Particle Filter Projection



Density Extraction



Sampling Variance

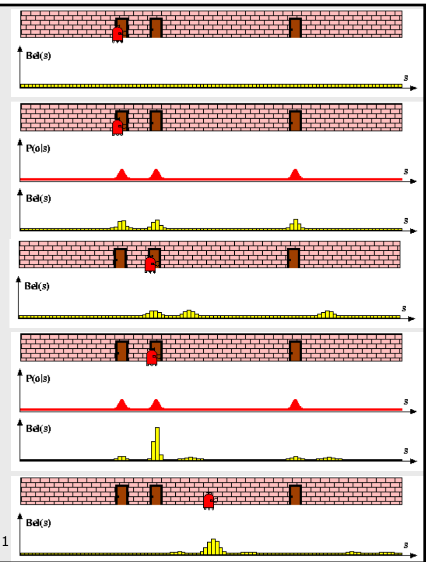


CSE-571
Robotics

Bayes Filter Implementations

Discrete filters

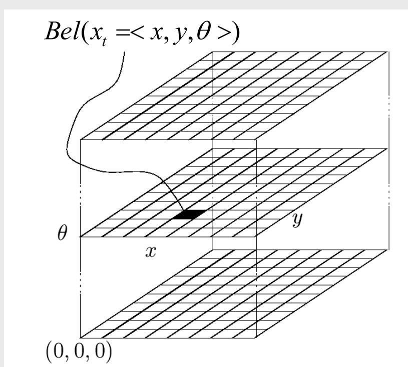
Piecewise Constant



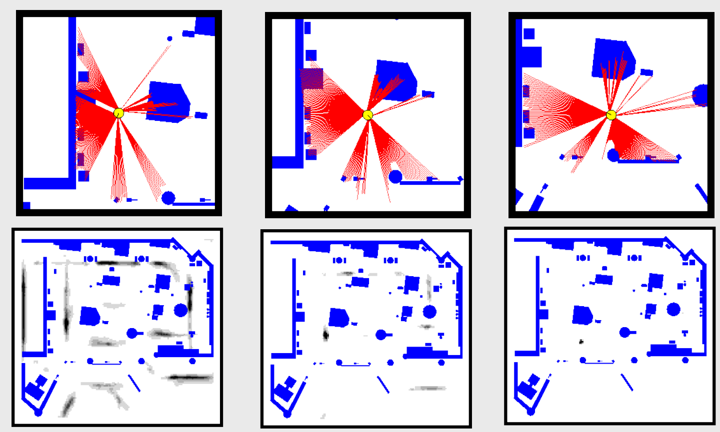
Discrete Bayes Filter Algorithm

1. Algorithm **Discrete_Bayes_filter**(*Bel(x),d*):
2. η=0
3. If *d* is a perceptual data item *z* then
4. For all *x* do
5. $Bel'(x) = P(z \mid x) Bel(x)$
6. $\eta = \eta + Bel'(x)$
7. For all *x* do
8. $Bel'(x) = \eta^{-1} Bel'(x)$
9. Else if *d* is an action data item *u* then
10. For all *x* do
11. $Bel'(x) = \sum_{x'} P(x \mid u, x')\, Bel(x')$
12. Return *Bel'(x)*

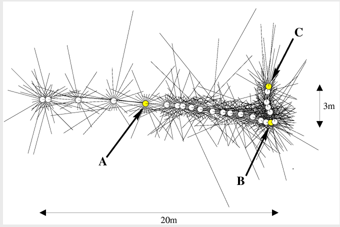
Piecewise Constant Representation

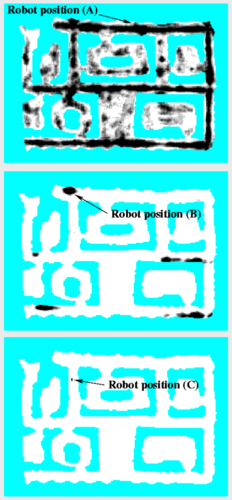


Grid-based Localization



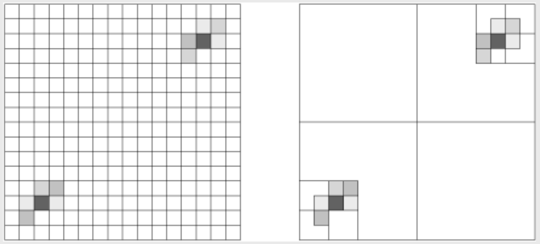
Sonars and Occupancy Grid Map





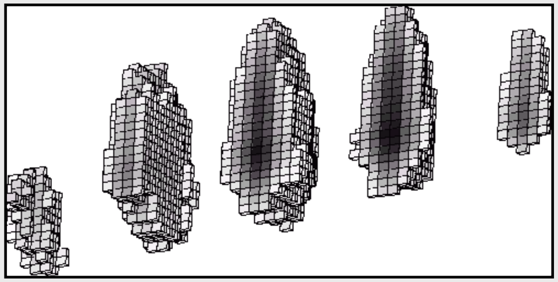
Tree-based Representation

Idea: Represent density using a variant of Octrees



Tree-based Representations

- Efficient in space and time
- Multi-resolution



Localization Algorithms - Comparison

	Kalman filter	Multi-hypothesis tracking	Topological maps	Grid-based (fixed/variable)	Particle filter
Sensors	Gaussian	Gaussian	Features	Non-Gaussian	Non-Gaussian
Posterior	Gaussian	Multi-modal	Piecewise constant	Piecewise constant	Samples
Efficiency (memory)	++	++	++	-/o	+/++
Efficiency (time)	++	++	++	o/+	+/++
Implementation	+	o	+	+/o	++
Accuracy	++	++	-	+/++	++
Robustness	-	+	+	++	+/++
Global localization	No	Yes	Yes	Yes	Yes