CSE-571
AI-based Mobile Robotics

Bayes Filter Implementations

Particle filters

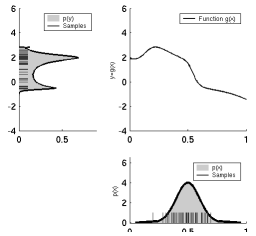
Particle Filters

§ Represent belief by random samples
§ Estimation of non-Gaussian, nonlinear processes

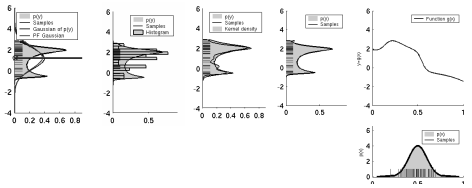
§ Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

§ Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
§ Computer vision: [Isard and Blake 96, 98]
§ Dynamic Bayesian Networks: [Kanazawa et al., 95]d

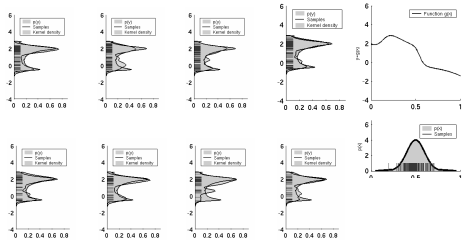
Particle Filter Projection



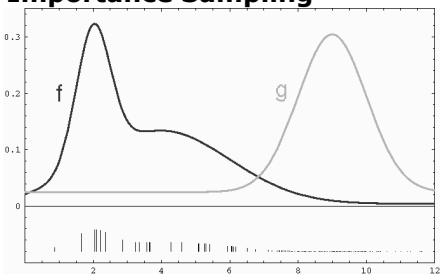
Density Extraction



Sampling Variance

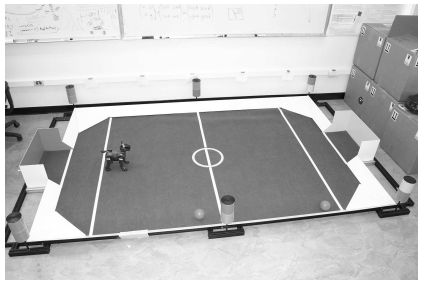


Importance Sampling

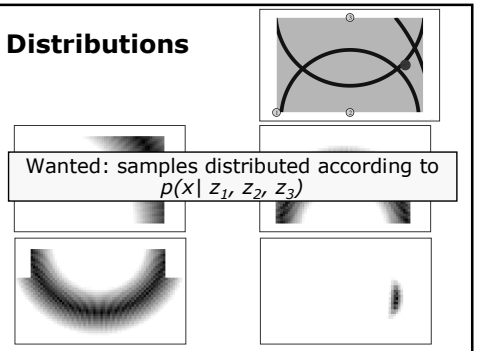


Weight samples: $w = f / g$

Importance Sampling with Resampling: Landmark Detection Example



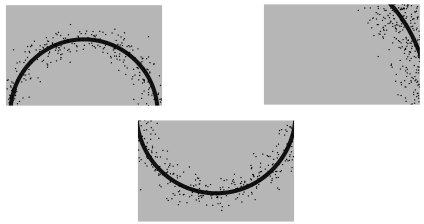
Distributions



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.

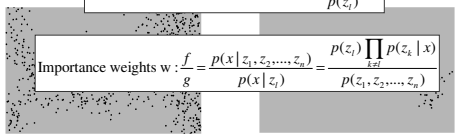


Importance Sampling with Resampling

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

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

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



Weighted samples

After resampling

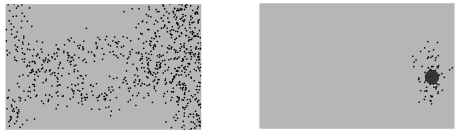
Importance Sampling with Resampling

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

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

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

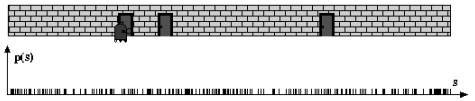
Importance Sampling with Resampling



Weighted samples

After resampling

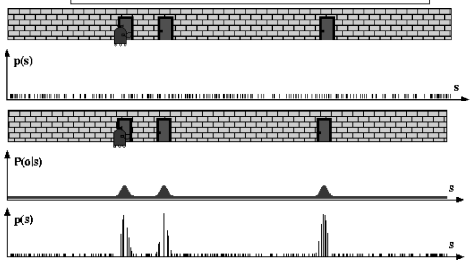
Particle Filters



Sensor Information: Importance Sampling

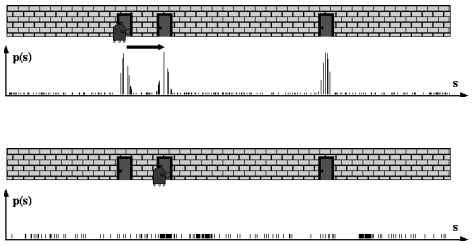
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

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



Robot Motion

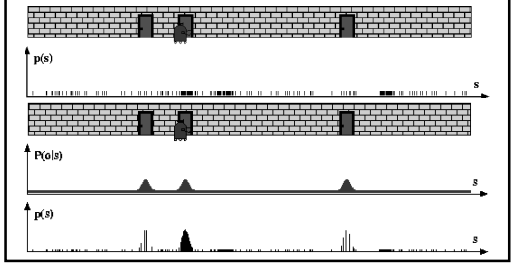
$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



Sensor Information: Importance Sampling

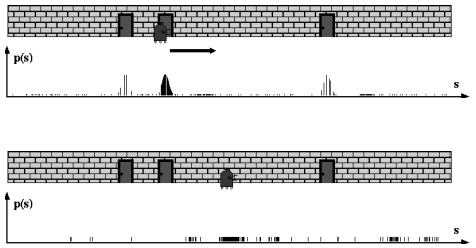
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

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



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

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

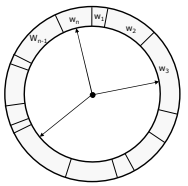
- draw x^i_{t-1} from $Bel(x_{t-1})$
- draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$
- Importance factor for x^i_t:

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

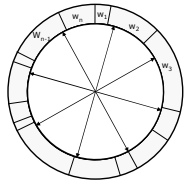
Resampling

- **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



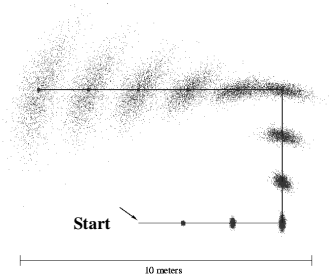
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Resampling Algorithm

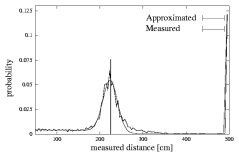
1. Algorithm **systematic_resampling**(S,n):
2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ — *Generate cdf*
4. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ — *Initialize threshold*
6. **For** $j = 1 \ldots n$ — *Draw samples …*
7. $\quad$ **While** ($u_j > c_i$) — *Skip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

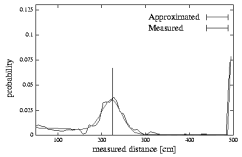
Motion Model Reminder



Proximity Sensor Model Reminder

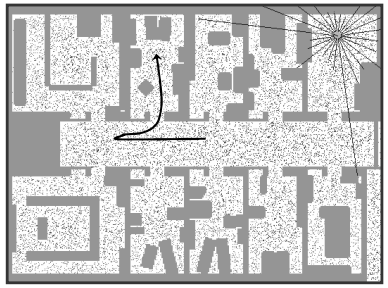


Laser sensor

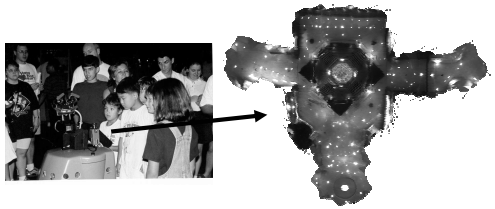


Sonar sensor

Sample-based Localization (sonar)

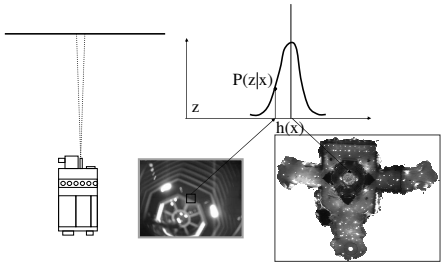


Using Ceiling Maps for Localization



[Dellaert et al. 99]

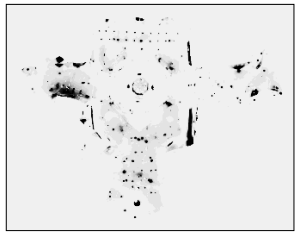
Vision-based Localization



Under a Light

Measurement z:

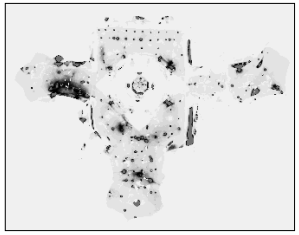
P(z|*x*):



Next to a Light

Measurement z:

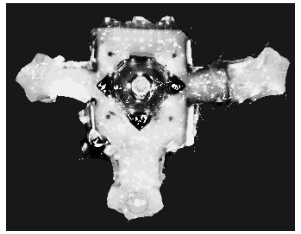
P(z|*x*):



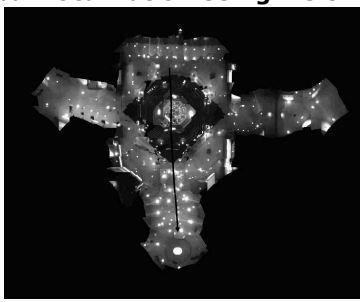
Elsewhere

Measurement z:

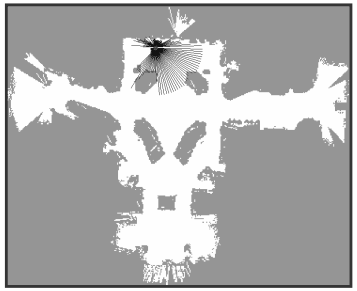
P(z|*x*):



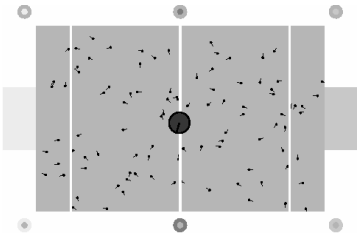
Global Localization Using Vision



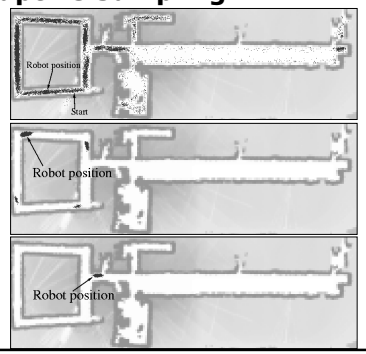
Recovery from Failure



Localization for AIBO robots



Adaptive Sampling



KLD-sampling

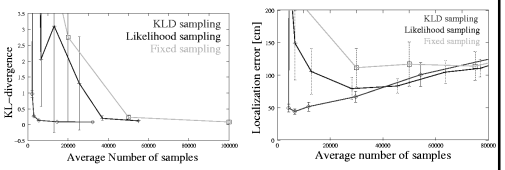
- **Idea**:
  - Assume we know the true belief.
  - Represent this belief as a multinomial distribution.
  - Determine number of samples such that we can guarantee that, with probability (1- δ), the KL-distance between the true posterior and the sample-based approximation is less than ε.
- **Observation**:
  - For fixed δ and ε, number of samples only depends on number k of bins with support:

$$n = \frac{1}{2\varepsilon}\chi^2(k-1,1-\delta) \cong \frac{k-1}{2\varepsilon}\left\{1-\frac{2}{9(k-1)}+\sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3$$

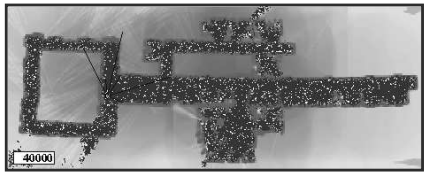
Adaptive Particle Filter Algorithm

1. Algorithm **adaptive_particle_filter**(S_{t-1}, u_{t-1} z_t, $\Delta, \varepsilon, \delta$):
2. $S_t = \varnothing$, $\alpha = 0$, $n = 0$, $k = 0$, $b = \varnothing$
3. **Do** *Generate new samples*
4. Sample index *j(n)* from the discrete distribution given by w_{t-1}
5. Sample x_t^n from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(n)}$ and u_{t-1}
6. $w_t^n = p(z_t \mid x_t^n)$ *Compute importance weight*
7. $\eta = \eta + w_t^n$ *Update normalization factor*
8. $S_t = S_t \cup \{< x_t^n, w_t^n >\}$ *Insert*
9. **If** (x_t^n falls into an empty bin *b*) *Update bins with support*
10. *k=k+1, b = non-empty*
11. *n=n+1*
12. **While** ($n < \frac{1}{2\varepsilon}\chi^2(k-1,1-\delta)$)
13. **For** $i = 1 \ldots n$
14. $w_t^i = w_t^i / \eta$ *Normalize weights*

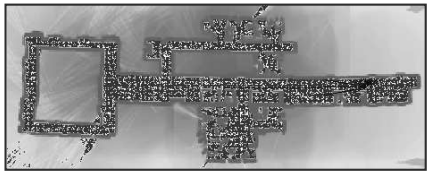
Evaluation



Example Run Sonar



Example Run Laser



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