Two Applications of Probabilistic Reasoning

- Object tracking in video
- Robot localization and mapping

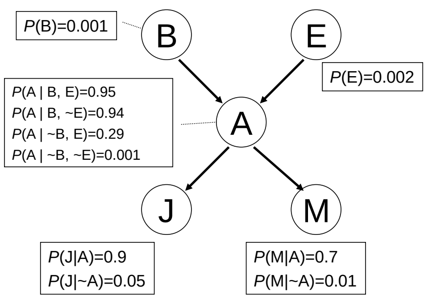
Approximate Inference by Sampling

- Inference can be done ***approximately*** by sampling
- General sampling approach:
  – Generate many, many samples (each sample is a complete assignment of all variables)
  – Count the fraction of samples matching query and evidence
  – As the number of samples approaches ∞, the fraction converges to the desired posterior:

$$P(query \mid evidence)$$

Simple Sampling

- This BN defines a joint distribution
- Can you generate a set of samples that have the same underlying joint distribution?

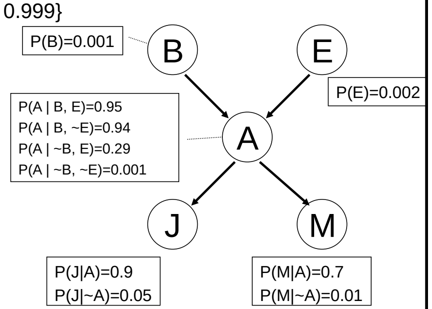


Simple Sampling

1. Sample B: x=rand(0,1). If (x<0.001) B=true else B=false
2. Sample E: x=rand(0,1). If (x<0.002) E=true else E=false
3. If (B==true and E==true) sample A ~ {0.95, 0.05}
   elseif (B==true and E==false) sample A ~ {0.94, 0.06}
   elseif (B==false and E==false) sample A ~ {0.29, 0.71}
   else sample A ~ {0.001, 0.999}
4. Similarly sample J
5. Similarly sample M

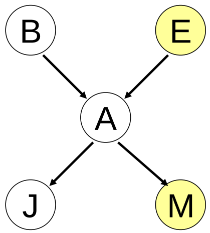
This generates one sample.

Repeat to generate more samples



Inference with Simple Sampling

- Say we want to infer B, given E, M, i.e., $P(B \mid E, M)$
- We generate tons of samples
- Keep those samples with E=true and M=true, throw away the others
- In the ones we keep (n of them), count the ones with B=true, i.e., those that fit our query (n_1)
- We return an estimate of
  $P(B \mid E, M) \approx n_1 / n$
- The quality of this estimate improves as we sample more
- Can generalize the method to an arbitrary BN



Problem

- Track the (hidden) state, **X**, of a system as it changes over time
- Given: A sequence of noisy observations (i.e., features), **Z**
- Goal: Compute best estimate of the hidden variables specifying the state, **X**, of the system. That is, compute

$$\underset{\mathbf{X}}{\operatorname{argmax}}\ P(\mathbf{X} \mid \mathbf{Z})$$

Particle Filters

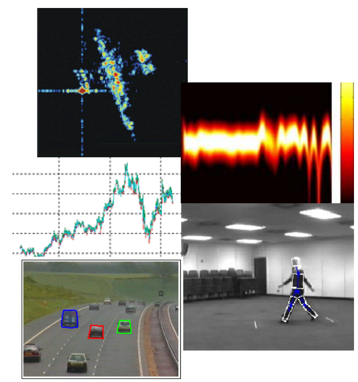
Also known as:

- Sequential Monte Carlo filter
- Bootstrap filter
- Condensation
- Survival of the fittest

- Sampling-based approach rather than trying to calculate the exact posterior
- Represent belief by a set of random samples
- Represent posterior *distribution*
- Approximate solution to an exact model (rather than an optimal solution to an approximate model)

Applications

- Tracking of aircraft positions from radar
- Estimating communications signals from noisy measurements
- Predicting financial data
- Tracking of people or cars in surveillance videos



Application

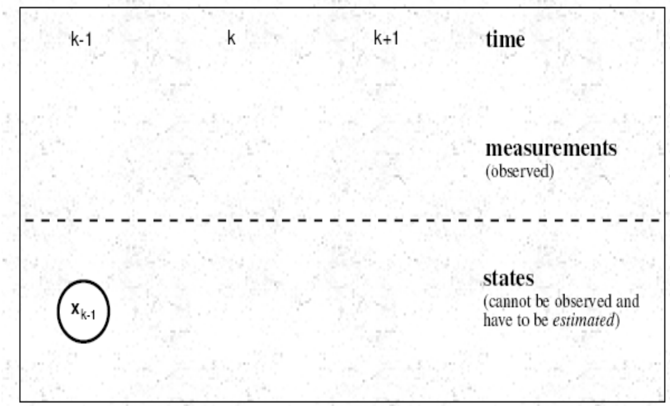
- Model-based visual tracking in dense clutter at near video frame rates

Tracking using Particle Filters: CONDENSATION (Conditional Density Propagation)

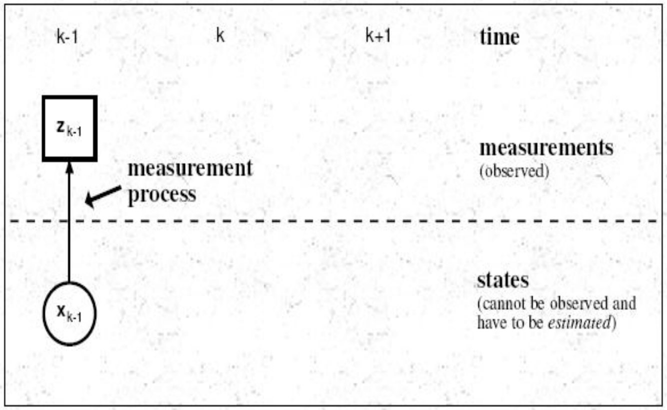
M. Isard and A. Blake, CONDENSATION – Conditional density propagation for visual tracking, *Int. J. Computer Vision* **29**(1), 1998, pp. 4-28

Example of CONDENSATION Algorithm

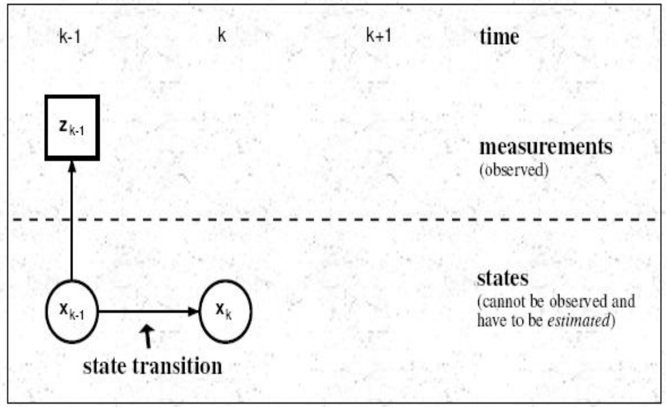
Particle Filtering Algorithm (1)



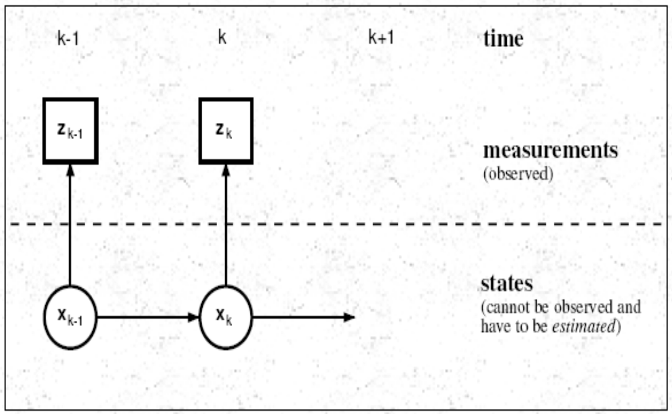
Particle Filtering (2)



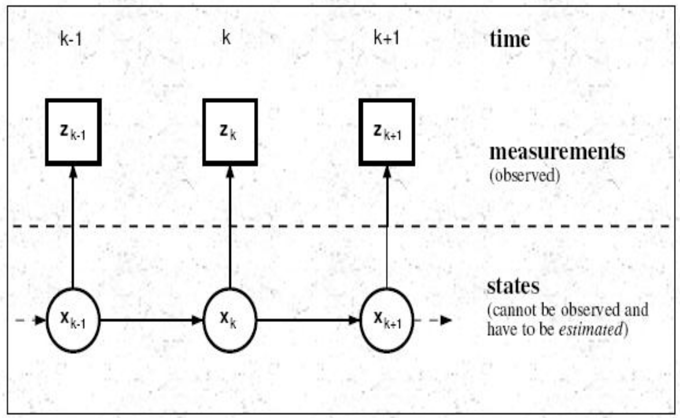
Particle Filtering (3)



Particle Filtering (4)



Particle Filtering (5)

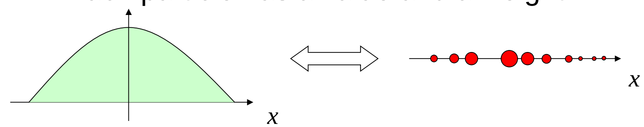


Approach

- Probabilistic (Bayesian) framework for tracking objects such as curves in clutter using an iterative Monte Carlo sampling algorithm
- Model motion and shape of target
- Top-down approach
- Simulation using discrete samples instead of analytic solution

Monte Carlo Samples (Particles)

- The posterior distribution $P(x|z)$ may be difficult or impossible to compute in closed form
- An alternative is to represent $P(x|z)$ using Monte Carlo ***samples*** (particles):
  - Each particle has a value and a weight



[http://www.fulton.asu.edu/~morrell/581/]

Monte Carlo Methods

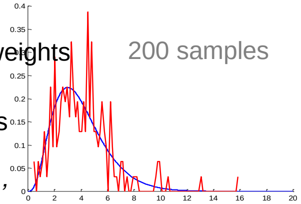
- A class of numerical methods involving statistical sampling processes for finding approximate solutions to quantitative problems
- In 1864 O. Fox estimated π by dropping a needle onto a ruled board
- S. Ulam used computers in 1940s to automate the statistical sampling process for developing nuclear weapons at Los Alamos
- S. Ulam, J. von Neuman, and N. Metropolis developed algorithms to convert non-random problems into random forms so that statistical sampling can be used for their solution; named them “Monte Carlo” methods after the casino

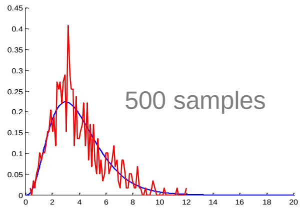
Samples $\Leftrightarrow$ Densities

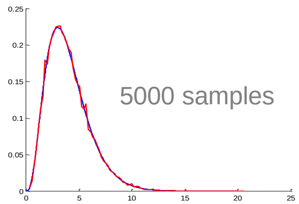
- Density $\Rightarrow$ samples
  Obvious
- Samples $\Rightarrow$ density
  Histogram, Kernel Density Estimation

Drawing Samples from a Probability Distribution Function

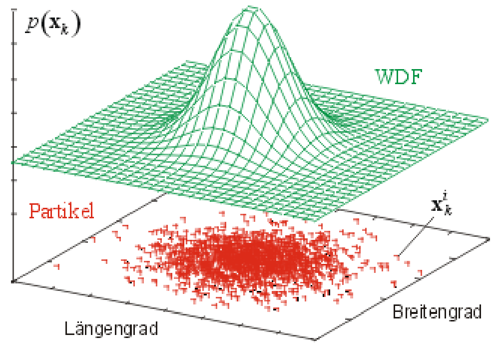
- Concept of samples and their weights
- Take P(x)=Gamma(4,1)
- Generate some random samples
- Plot basic approximation to pdf
- Each sample is called a ‘*particle*’







In 2D it looks like this



[http://www.ite.uni-karlsruhe.de/METZGER/DIPLOMARBEITEN/dipl2.html]

Obtaining State Estimates from Samples

- Any estimate of a function f(x_t) can be calculated by discrete PDF-approximation

$$E[f(x_t)] = \frac{1}{N}\sum_{j=1}^{N} w_t^{(j)} f(x_t^{(j)})$$

- Mean: $E[x_t] = \frac{1}{N}\sum_{j=1}^{N} w_t^{(j)} x_t^{(j)}$
- MAP-estimate: particle with largest weight
- Robust mean: mean within window around MAP-estimate

Probabilistic Framework

- Object dynamics form a temporal (first-order) Markov chain $P(x_t \mid \mathrm{X}_{t-1}) = P(x_t \mid x_{t-1})$
- Observations, z_t, are independent (mutually and wrt process)

$$P(Z_{t-1}, x_t \mid X_{t-1}) = P(x_t \mid X_{t-1}) \prod_{i=1}^{t-1} P(z_i \mid x_i)$$

- Uses Bayes’s rule

Tracking as Estimation

- Compute state posterior, $P(\mathbf{X}|\mathbf{Z})$, and select next state to be the one that maximizes this maximum *a posteriori* (MAP) estimate
- Measurements are complex and noisy, so posterior cannot be evaluated in closed form
- Particle filter (iterative sampling) idea: **Stochastically approximate the state posterior with a set of *N* weighted particles, (*s*, π), where *s* is a sample state and π is its weight**
- Use Bayes's rule to compute $P(\mathbf{X} \mid \mathbf{Z})$

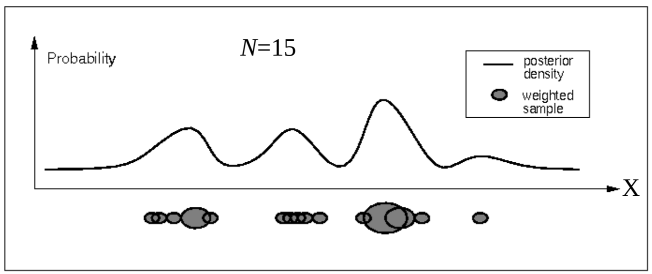
Factored Sampling

- Generate a set of samples that *approximates* the posterior, $P(\mathbf{X}|\mathbf{Z})$
- Sample set $\mathbf{s} = \{s^{(1)}, ..., s^{(N)}\}$ generated from $P(\mathbf{X})$; each sample has a **weight** ("probability")

$$\pi_i = \frac{P_z(s^{(i)})}{\sum_{j=1}^{N} P_z(s^{(j)})}$$

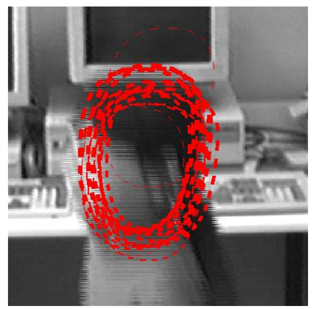
$$P_z(x) = P(z \mid x)$$

Factored Sampling

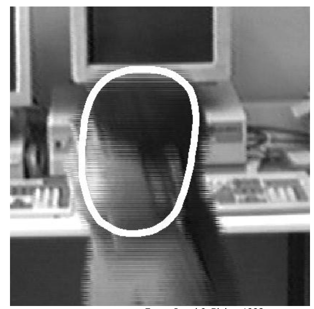


- CONDENSATION for one image

Estimating Target State



State samples



From *Isard & Blake, 1998*

Mean of weighted state samples

Bayes's Rule

This is what you can evaluate

This is what you may know *a priori*, or what you can ***predict***

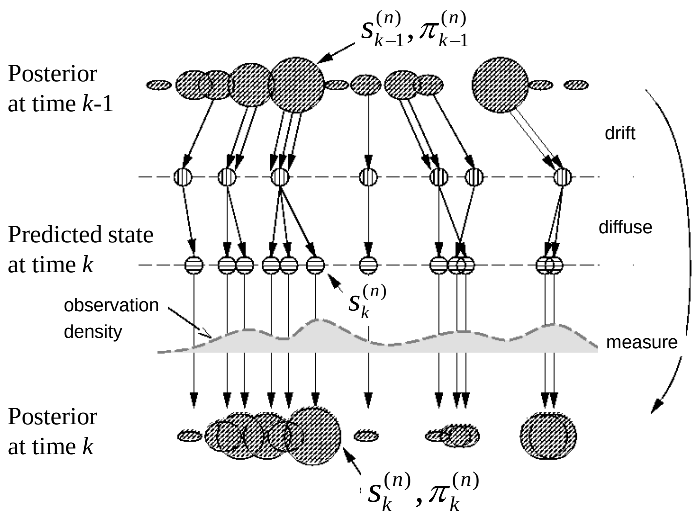
$$P(\mathbf{X} \mid \mathbf{Z}) = \frac{P(\mathbf{Z} \mid \mathbf{X})\, P(\mathbf{X})}{\cancel{P(\mathbf{Z})}}$$

This is what you want. Knowing *P*(**X**|**Z**) will tell us the most likely state **X**

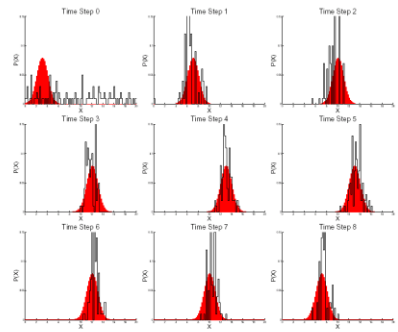
This is a constant for a given image

CONDENSATION Algorithm

1. **Select**: Randomly select *N* particles from $\{s_{t-1}^{(n)}\}$ based on weights $\pi_{t-1}^{(n)}$; same particle may be picked multiple times (*factored sampling*)
2. **Predict**: Move particles according to deterministic dynamics of motion model (*drift*), then perturb individually (*diffuse*)
3. **Measure**: Get a likelihood for each new sample by comparing it with the image's local appearance, i.e., based on $P(z_t \mid x_t)$; then update weight accordingly to obtain $\{(s_t^{(n)}, \pi_t^{(n)})\}$

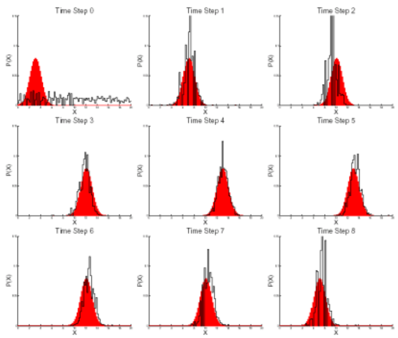


Particle Filter Demo 1



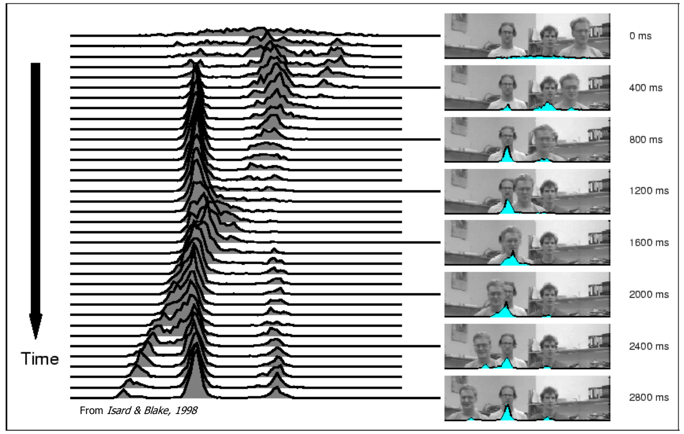
moving Gaussian + uniform, N=100 particles

Particle Filter Demo 2



moving Gaussian + uniform, N=1000 particles

State Posterior



Object Motion Model

- For video tracking we need a way to propagate probability densities, so we need a “motion model” such as
  - $\mathbf{X}_{t+1} = \mathbf{A}\ \mathbf{X}_t + \mathbf{B}\ \mathbf{W}_t$ where **W** is a noise term and **A** and **B** are state transition matrices that can be learned from training sequences
- The state, **X**, of an object, e.g., a B-spline curve, can be represented as a point in a 6D state space of possible 2D affine transformations of the object

Dancing Example

Hand Example

Pointing Hand Example

Glasses Example

- 6D state space of affine transformations of a spline curve
- Edge detector applied along normals to the spline
- Autoregressive motion model

2D Articulated Models for Tracking

(Bregler '93)

3D Models are More Accurate…

- … when they are right
- [BTW, why is she wearing a black shirt?]

(Isard & Blake '99)

3D Model-based Example

- 3D state space: image position + angle
- Polyhedral model of object

Probabilistic Robotics: SLAM
(Simultaneous Localization and Mapping)

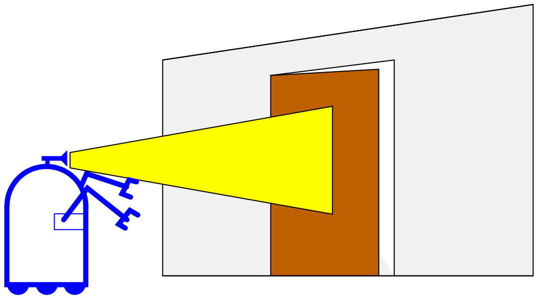
- Given no map
- No independent means of localization
- Unknown location and environment
- Robot must build a map *and* localize itself on this map

SLAM Problems

- The robot cannot (completely) trust its observations to build an accurate map
- Without an accurate map, it cannot localize itself accurately
- Without accurate localization, how can the robot build the map?

Simple Example of State Estimation

- Suppose a robot obtains measurement *z*
- What is *P*(*open* | *z*)?



Actions

- Often the world is **dynamic** since
  - **actions carried out by the robot**,
  - **actions carried out by other agents**,
  - or just the **time** passing by

  change the world

- How can we **incorporate** such **actions**?

Typical Actions

- The robot **turns its wheels** to move
- The robot **uses its manipulator** to grasp an object
- Etc.
- Actions are **never carried out with absolute certainty**
- In contrast to measurements, **actions generally increase uncertainty**

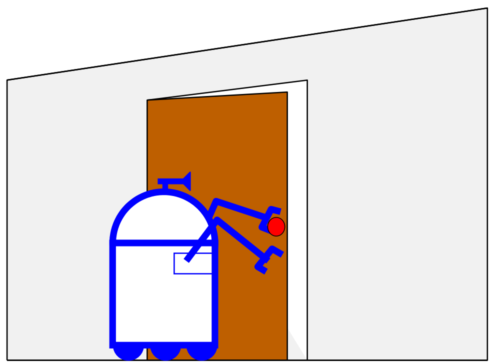
Modeling Actions

- To incorporate the outcome of an action, *u,* into the current "belief," we compute the conditional probability:

$$P(x \mid u, x')$$

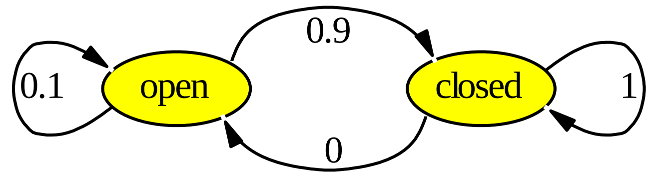
- Specifies that **executing *u* changes the state from *x′* to *x***

Example: Closing the Door



State Transitions

P(x|u,x') for *u* = "close door":



If the door is open, the action "close door" succeeds in 90% of all cases

Bayesian Framework

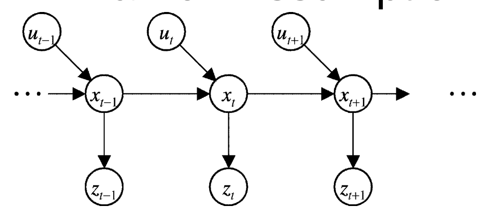
- **Given:**
  - Stream of observations, *z,* and action data, *u:*

$$d_t = \{u_1, z_1 \ldots, u_t, z_t\}$$

  - Sensor model: $P(z \mid x)$
  - Action model: $P(x \mid u, x')$
  - Prior: probability of the system state $P(x)$
- **Wanted:**
  - Estimate the state, *x,* of a dynamical system
  - The posterior of the state is also called **Belief**:

$$Bel(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

Markov Assumption



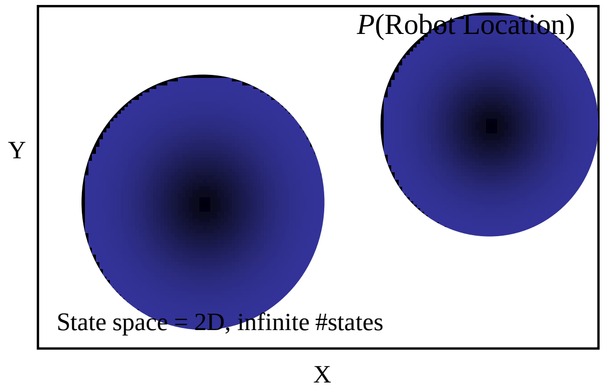
$$p(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t \mid x_t)$$

$$p(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

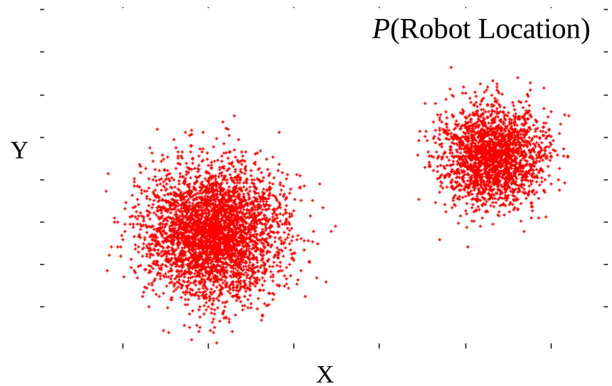
Assumptions:

- Static world
- Independent noise
- Perfect model, no approximation errors

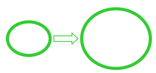
Probability of Robot Location

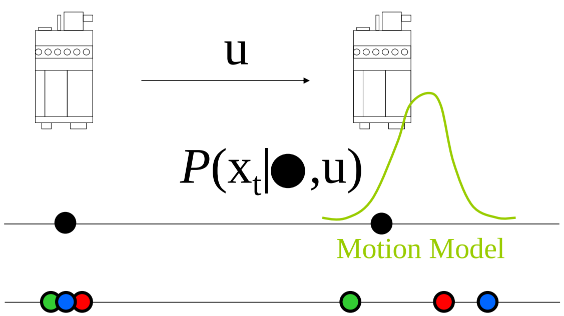


Sampling as Representation

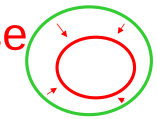


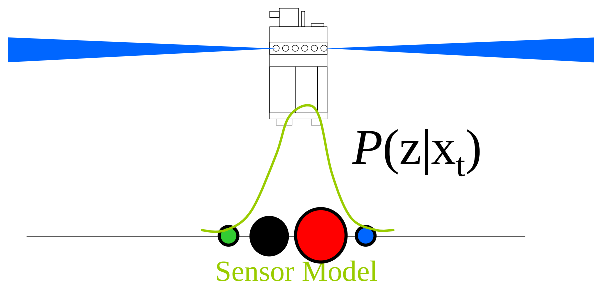
1. Prediction Phase



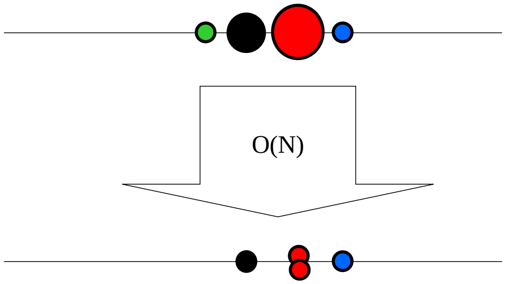


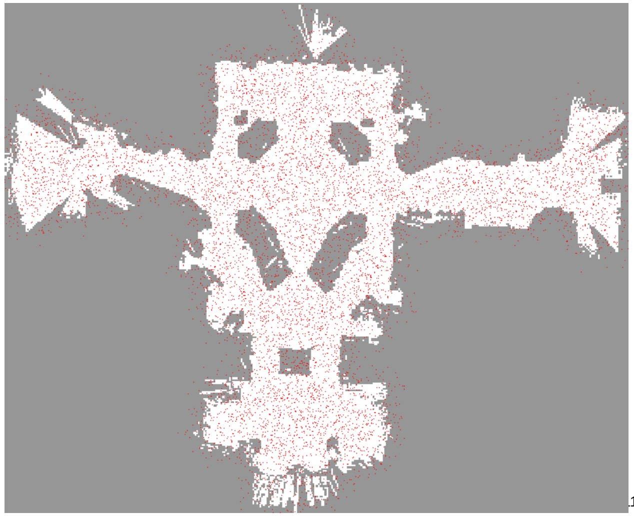
2. Measurement Phase

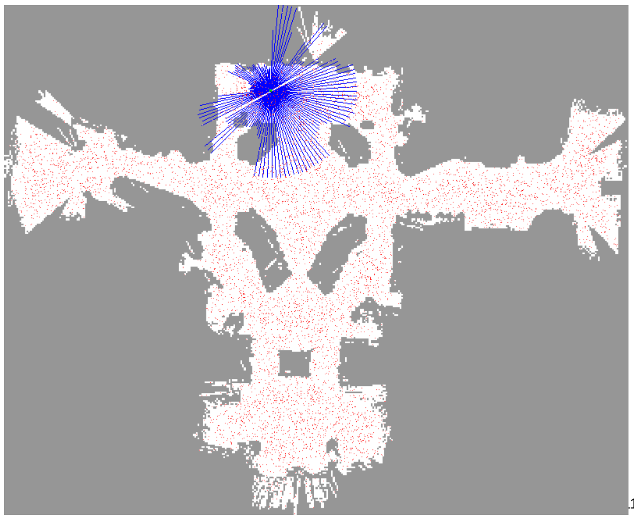


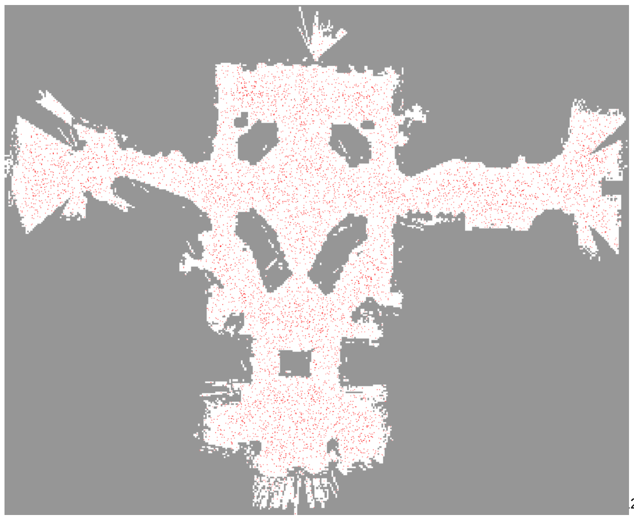


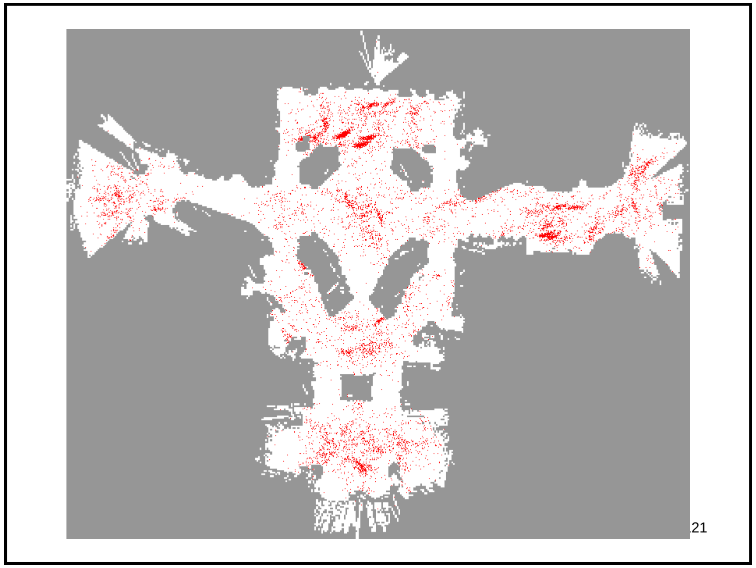
3. Resampling Step

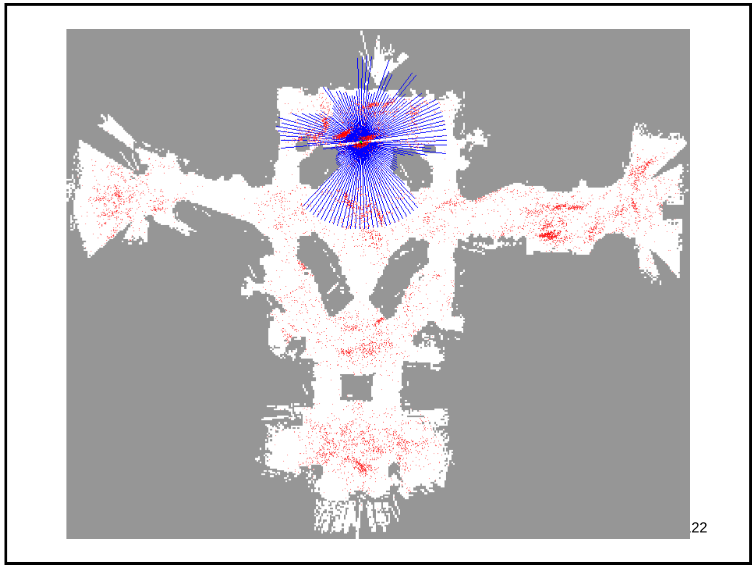


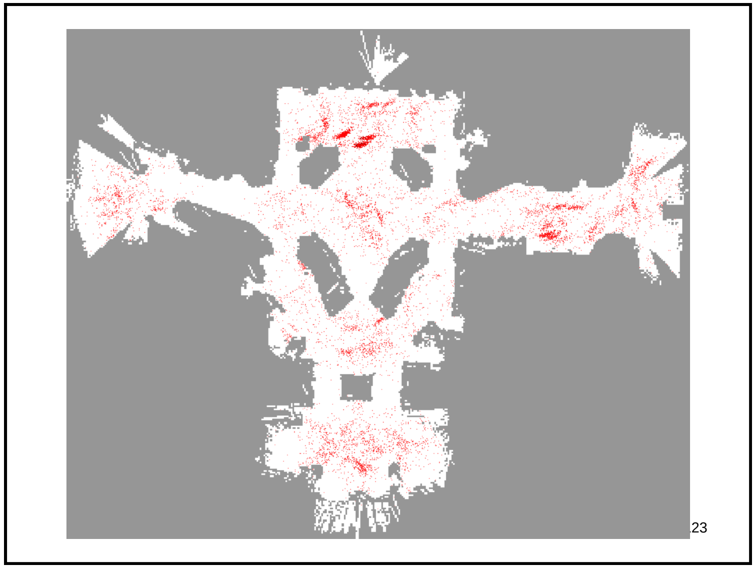


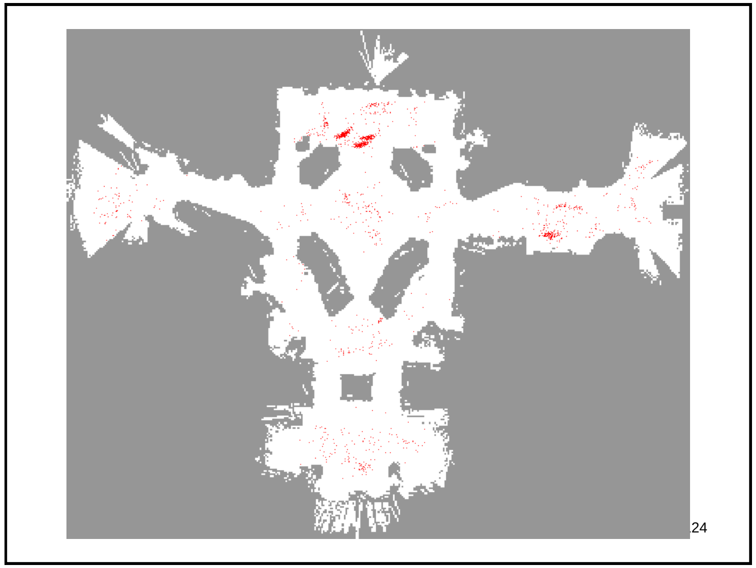




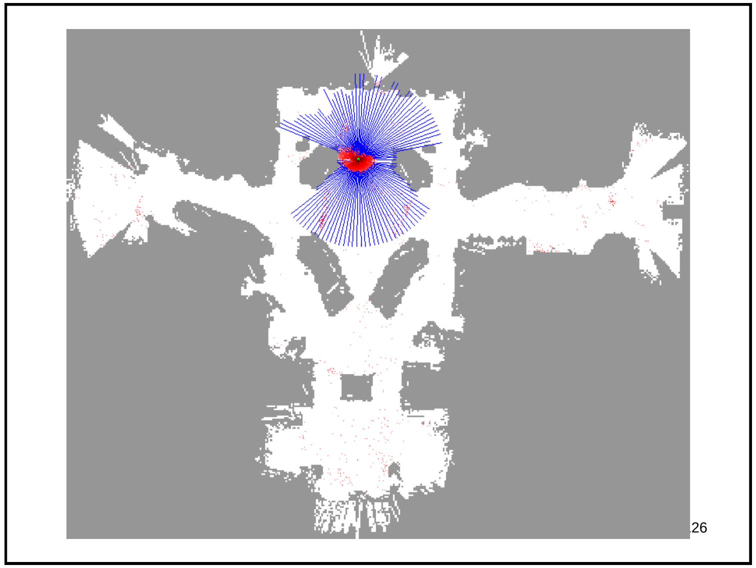

21

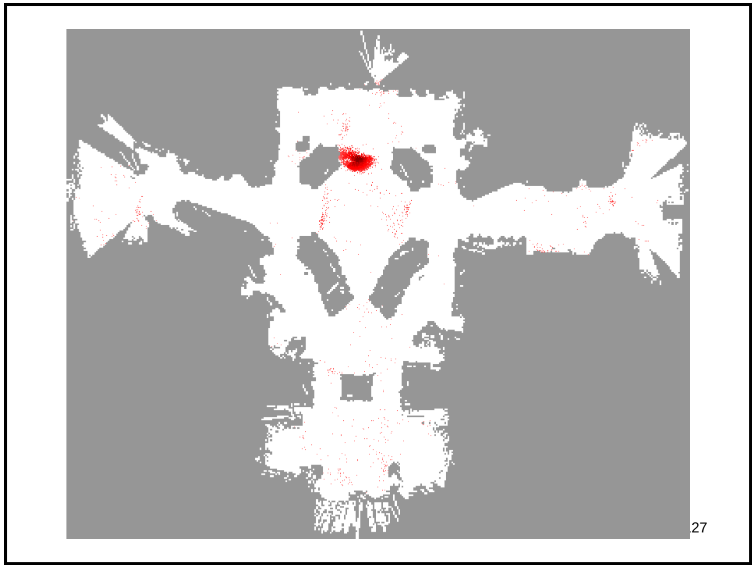

22

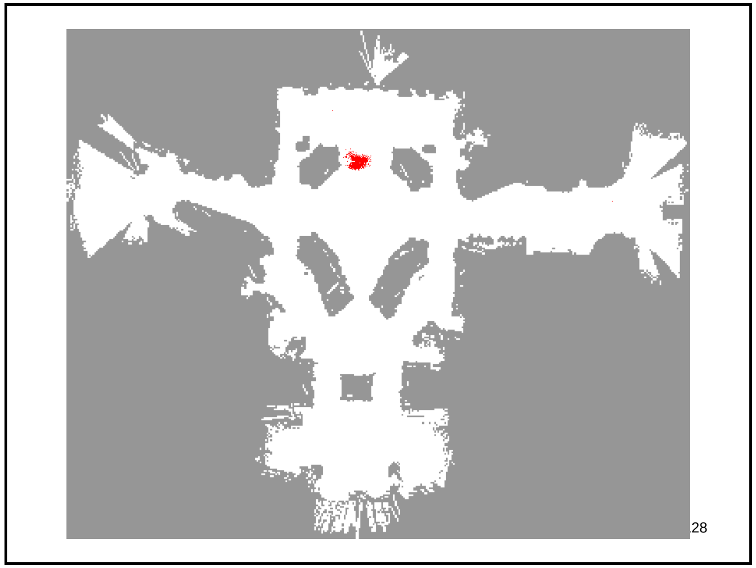

23

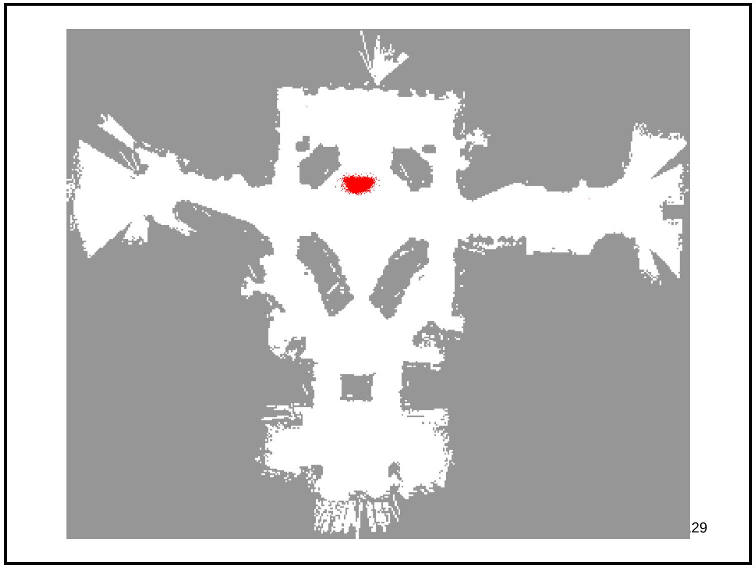

24

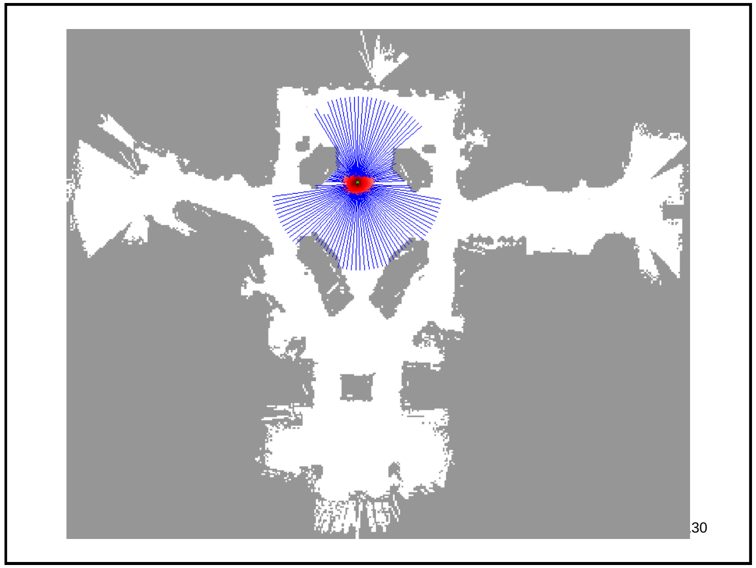

25

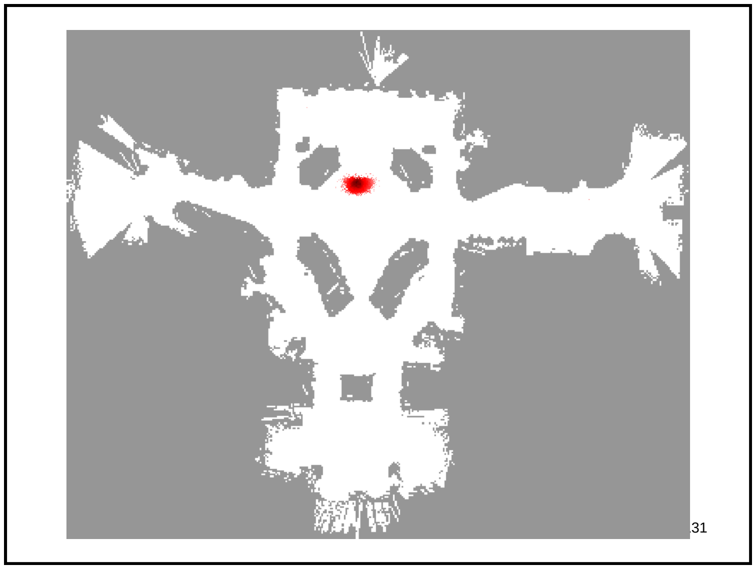

26

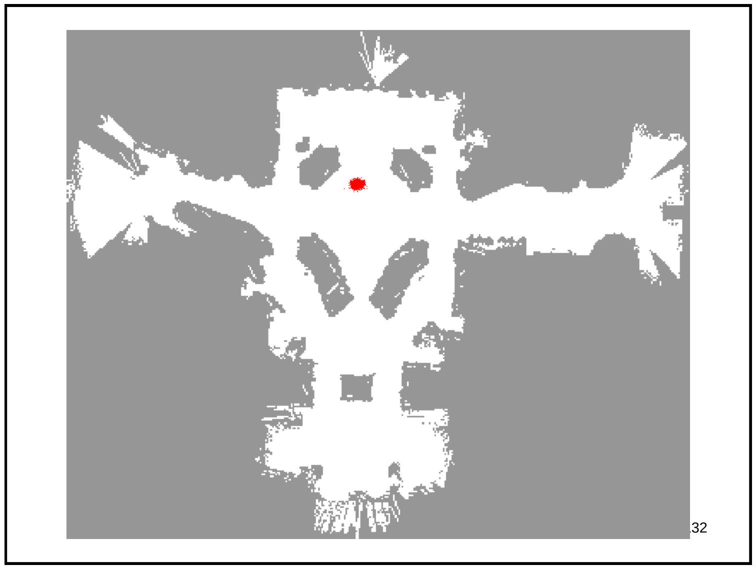

27

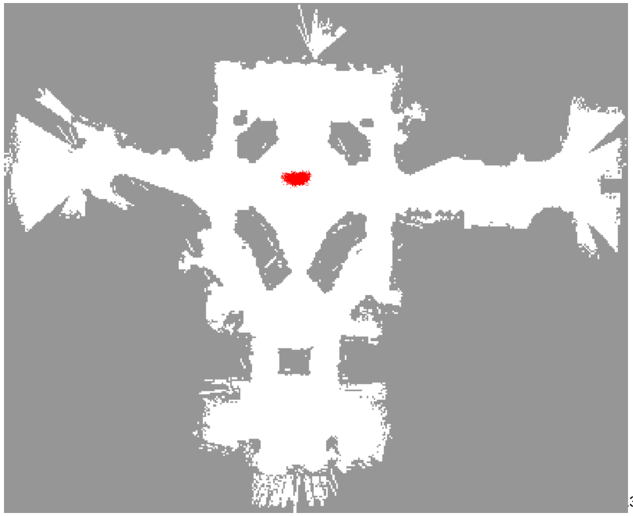

28


29

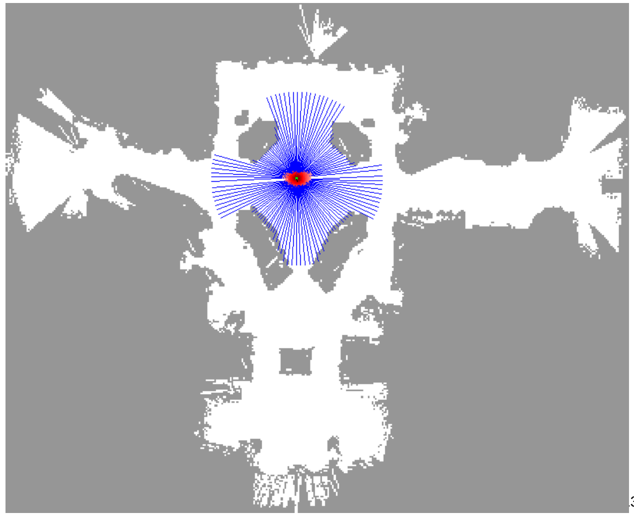

30


31

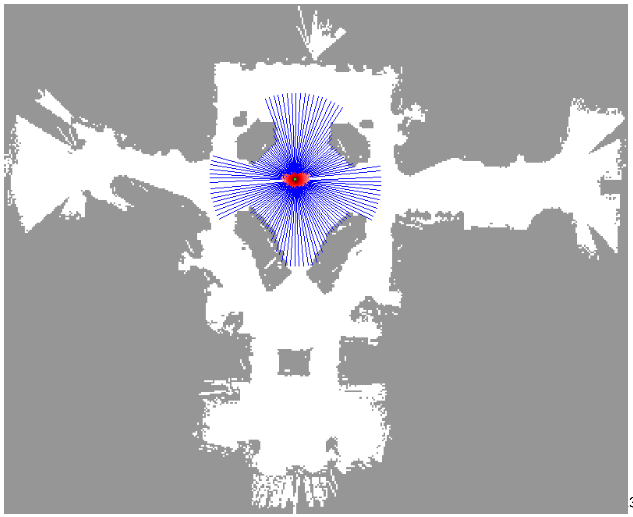

32



33

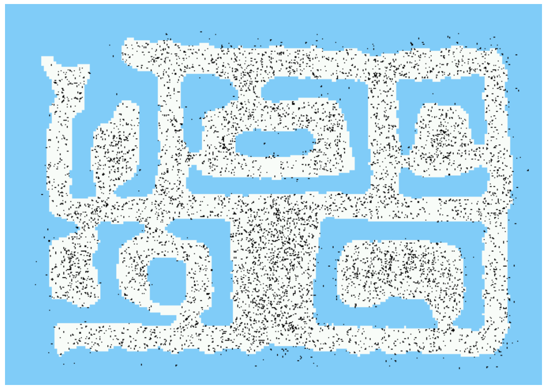


34



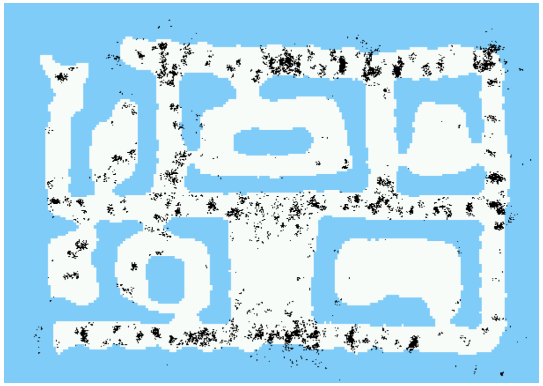
35

Initial Distribution



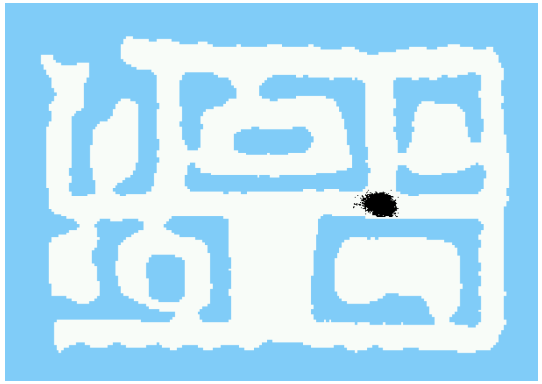
137

After Incorporating Ten Ultrasound Scans



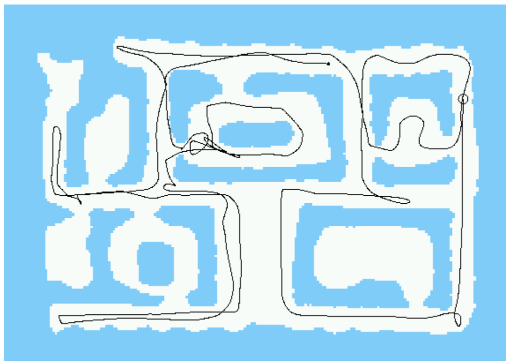
138

After Incorporating 65 Ultrasound Scans



139

Estimated Path



140

Advantages of Particle Filtering

- Nonlinear dynamics, measurement model easily incorporated
- Copes with lots of false positives
- Multi-modal posterior okay (unlike Kalman filter)
- Multiple samples provides multiple hypotheses
- Fast and simple to implement
- **Representing uncertainty using samples is powerful, fast, and simple**