Lecture 17:
FastSLAM

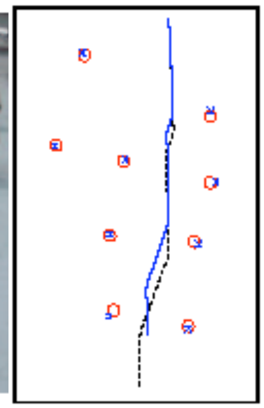
CS 344R/393R: Robotics

Benjamin Kuipers

Landmark-Based Mapping

- Suppose the environment consists of a set of isolated landmarks:
  - Trees in the forest
  - Rocks in the Martian desert
- Treat a landmark as a point location (x_k, y_k).
- SLAM: the robot learns the locations of the landmarks while localizing itself.

“Martian” Rocky Desert



The *real* Martian desert

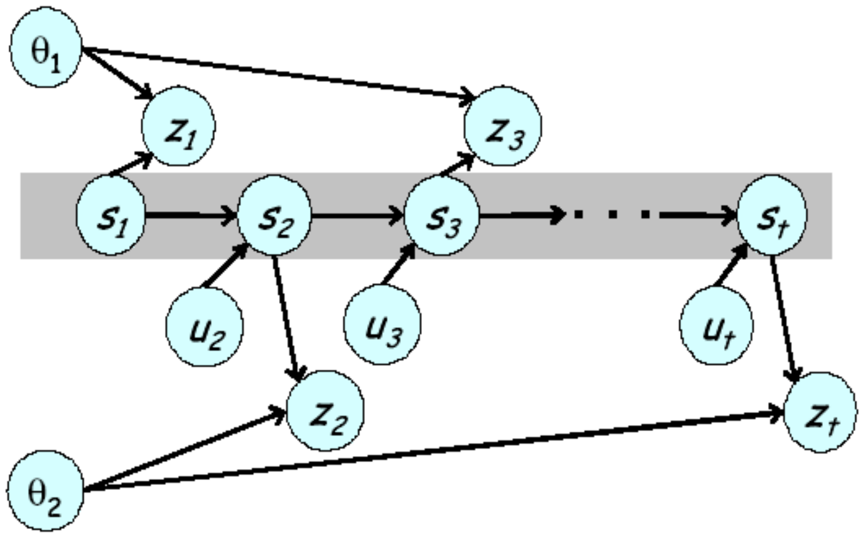
Landmarks vs Occupancy Grids

- An occupancy grid makes no assumption about types of features.
  - Now we assume point landmarks, but walls and other types of features are also possible.
- An occupancy grid (typically) has fixed resolution.
  - Feature models can be arbitrarily precise.
- An occupancy grid takes space and time the size of the environment to be mapped.
  - A feature-based map takes space and time reflecting the contents of the environment.

Kalman Filters for Features

- A landmark feature has parameters (x_i,y_i).
  - The robot's pose has parameters (x,y,φ).
  - Robot plus K landmarks needs $2K+3$ parameters.
- To estimate these with a Kalman filter:
  - The means require $2K+3$ parameters.
  - The covariance matrix needs $(2K+3)^2$.
- FastSLAM observes that the landmarks are independent, given the robot's pose.
  - A 2×2 covariance matrix for each landmark.
  - Total parameters: $K(2 + 2\times2) + 3$

Landmark Poses are Independent Given the Robot's Pose



Bayesian Model

- Robot poses: $s^t = s_1, s_2, \ldots s_t$.
  - (Slightly different from our usual notation.)
- Robot actions: $u^t = u_1, u_2, \ldots u_t$
- Observations: $z^t = z_1, z_2, \ldots z_t$
- Landmarks: $\Theta = \theta_1, \ldots \theta_k$
- Correspondence: $n^t = n_1, n_2, \ldots n_t$
  - $n_t = k$ means z_t is an observation of θ_k
  - Assume (without loss of generality) that landmarks are observed one at a time.

The SLAM Problem

- Estimate $p(s^t, \Theta \mid z^t, u^t, n^t)$ using
  - action model: $p(s_t \mid u_t, s_{t-1})$
  - sensor model: $p(z_t \mid s_t, \Theta, n_t)$
- Independence lets us factor

$$p(s^t, \Theta \mid z^t, u^t, n^t)$$

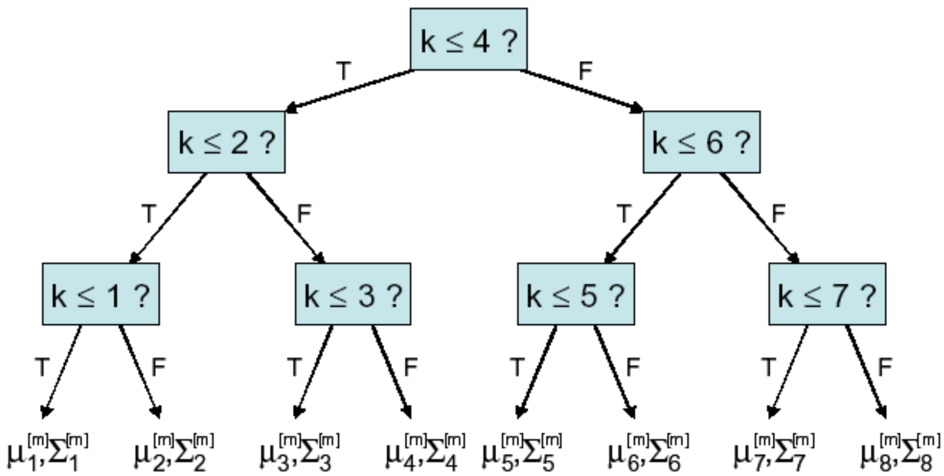
$$= p(s^t \mid z^t, u^t, n^t) \prod_k p(\theta_k \mid s^t, z^t, u^t, n^t)$$

  - trajectory estimation $p(s^t \mid z^t, u^t, n^t)$
  - from landmark estimation $p(\theta_k \mid s^t, z^t, u^t, n^t)$

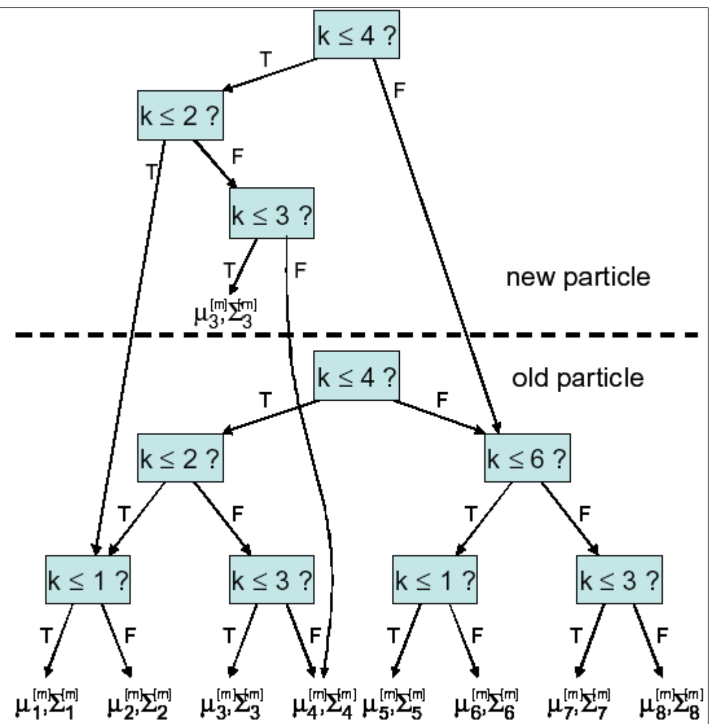
Factor the Uncertainty

- *Rao-Blackwellized particle filters*.
- Use particle filters to represent the distribution over trajectories $p(s^t \mid z^t, u^t, n^t)$
  - M particles
- Within each particle, use Kalman filters to represent distribution for each landmark pose $p(\theta_k \mid s^t, z^t, u^t, n^t)$
  - K Kalman filters per particle
- Each update requires $O(MK)$ time.
  - Easy to improve to $O(M \log K)$.

Balanced Tree of Gaussians In Each Particle

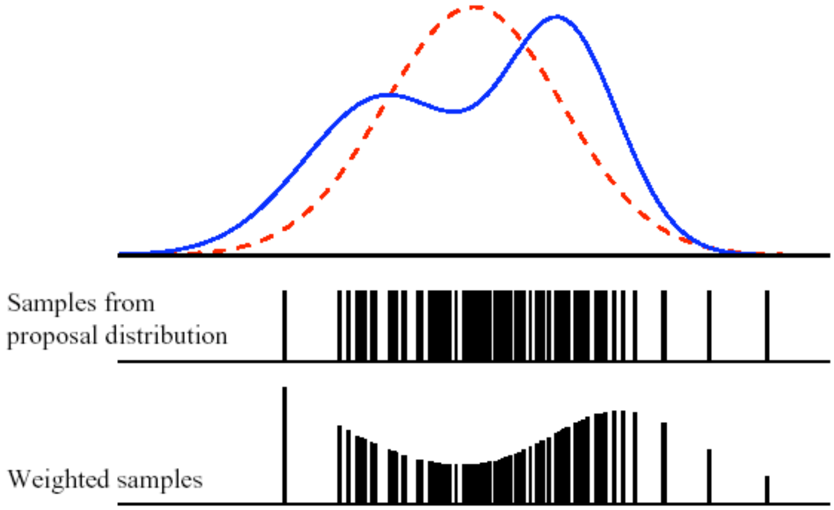


Insertions Are Also Cheap: $O(\log K)$



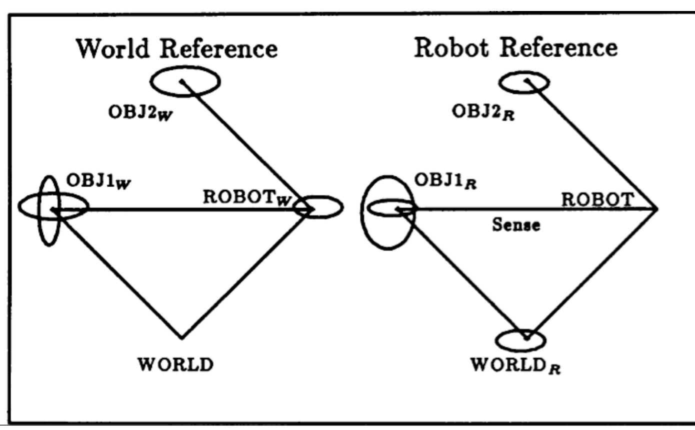
Importance Sampling

- Sample from one distribution.
  - Correct to approximate a target distribution.



Kalman Filters to Estimate Locations of Fixed Landmarks

- Within the context of each particle, the pose $ROBOT_W$ is known perfectly.

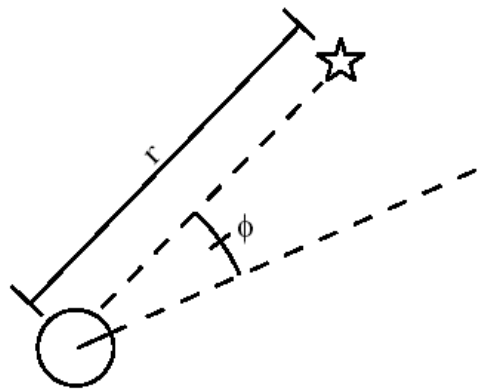


Updating One Landmark $p(\theta_k \mid s^t, z^t, u^t, n^t)$

- $p(\theta_k \mid s^t, z^t, u^t, n^t) =$
  $p(\theta_k \mid s_t, z_t, n_t{=}k)$
- $z_t = z = (r, \phi)^{\mathrm{T}}$
- $s_t = s = (x, y, \varphi)^{\mathrm{T}}$

Kalman filter model:

- $\theta_{k,t} = \theta_{k,t\text{-}1}$
- $z_t = g(s_t, \theta_{k,t})$



Kalman Measurement Function

- The measurement function $z = g(s,\theta) = (r,\phi)^{\mathrm{T}}$
  - where $s = (x, y, \varphi)^{\mathrm{T}}$ and $\theta = (u, v)^{\mathrm{T}}$

$$g(s,\theta) = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{(u-x)^2 + (v-y)^2} \\ \operatorname{atan2}(v-y, u-x) - \varphi \end{bmatrix}$$

- The Jacobian of g with respect to $\theta=(u,v)^T$ is:

$$G_\theta = \begin{bmatrix} (u-x)/r & (v-y)/r \\ -(v-y)/r^2 & (u-x)/r^2 \end{bmatrix}$$

Kalman Filter Update Step

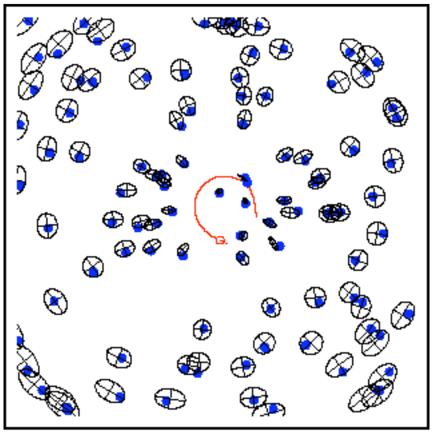
- Let (μ_t, Σ_t) be the mean and covariance of θ_k at time t.
- Landmarks are fixed, so prediction step is trivial.
  - $\theta_{k,t} = \theta_{k,t\text{-}1}$
- The KF correction step:
  - given ($\mu_{t\text{-}1}$, $\Sigma_{t\text{-}1}$)
  - and pose s_t
  - and observation z_t
  - update (μ_t, Σ_t)
- for each landmark θ_k
- in each particle $s_t^{(m)}$.

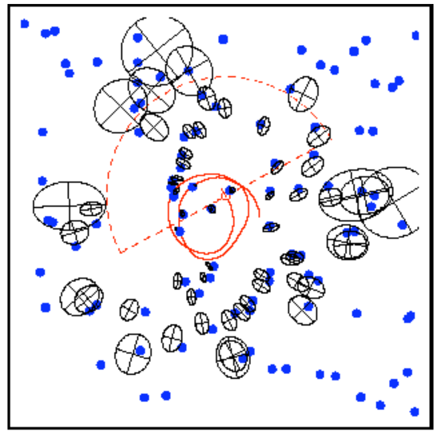
$$\hat{z}_t = g(s_t, \mu_{t-1})$$
$$G = \nabla_\theta g(s_t, \mu_{t-1})$$
$$Z = G\Sigma_{t-1}G^T + R$$
$$K = \Sigma_{t-1}G^T Z^{-1}$$
$$\mu_t = \mu_{t-1} + K(z_t - \hat{z}_t)$$
$$\Sigma_t = (I - KG)\Sigma_{t-1}$$

Implementation Hint

- Alan Oursland has a Java implementation
  - http://www.oursland.net/projects/fastslam/
- He reports having a hard time getting it to work, until Dieter Fox helped him tune the Kalman Filter.
  - The observation covariance R must be very large, so observations can match far-away landmarks.
- This is an example of how personal experience is important to replicating ideas.

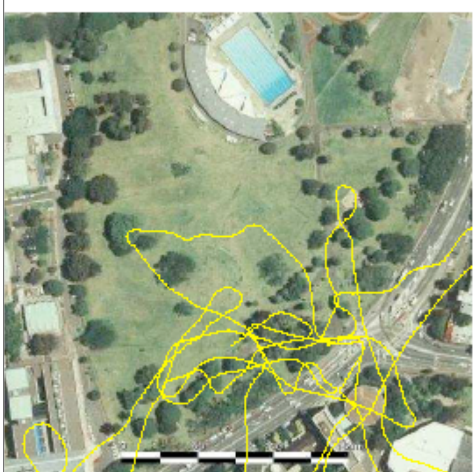
Maps and Robot Paths



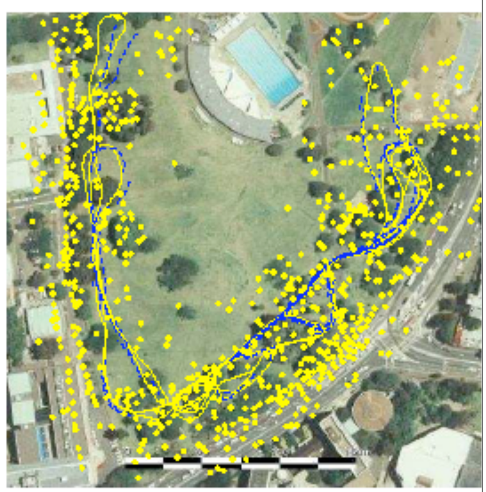


Tested successfully with up to 50,000 landmarks.

FastSLAM in Victoria Park

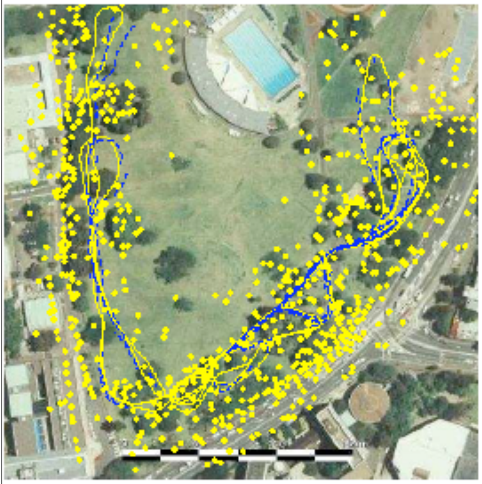


with raw odometry

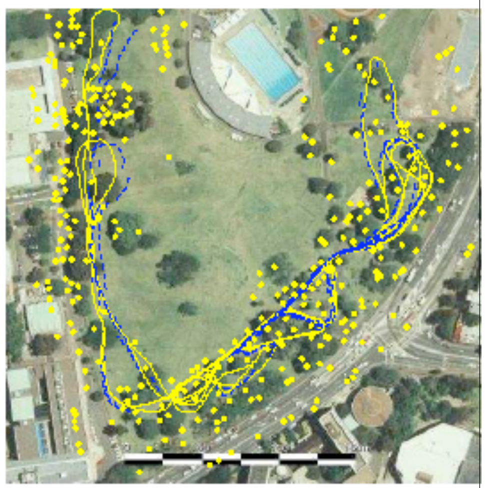


FastSLAM 2.0

FastSLAM in Victoria Park



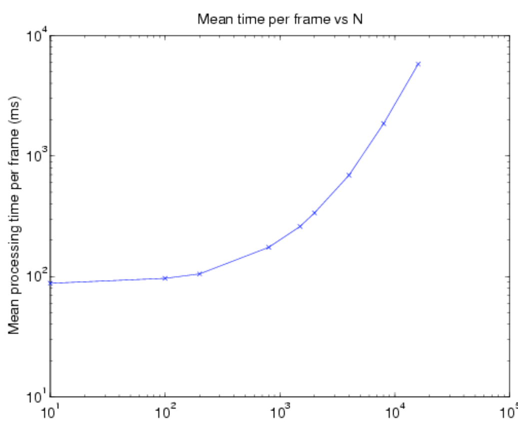
FastSLAM 2.0



Pruning bad landmarks

Another Practical Note

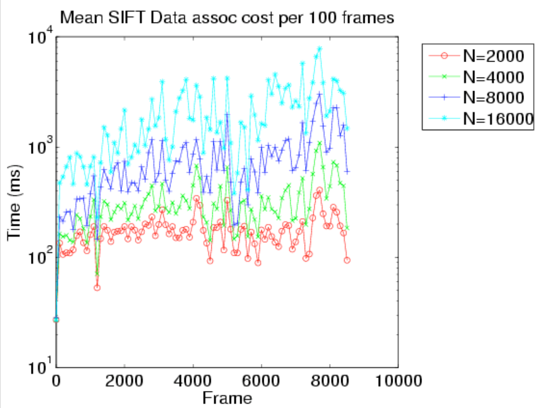
- In theory, FastSLAM should scale well: $O(KN \log M)$, where
  - N is the number of particles
  - K is the number of landmarks observed
  - M is the number of landmarks in the map
- But, in practice . . .

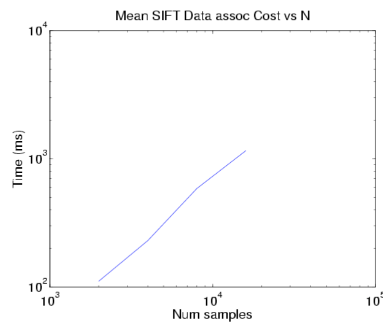


Robert Sim, http://www.cs.ubc.ca/~simra/lci/fastslam/nonlinear.html

A Complexity Experiment

- Measure only operations independent of N.
  - Take an image: $O(1)$.
  - Extract and match SIFT features: $O(K \log S)$
    - K features, matched against S features in kd-tree.
  - Update N particles, but don't count that cost.





Why doesn't FastSLAM scale?

- At each frame:
  - K SIFT features are added to the kd-tree, and
  - NK landmarks are added to the FastSLAM tree.
- Memory fragmentation:
  - In time, nearby SIFT features are separated in memory, so CPU cache miss rate goes up.
  - For large maps, page fault rate will also increase.
- So the problem is the memory hierarchy, due to failure of locality.

Coming Attractions

- Topological mapping (3)
- Social and ethical implications
  - *What if we succeed?*