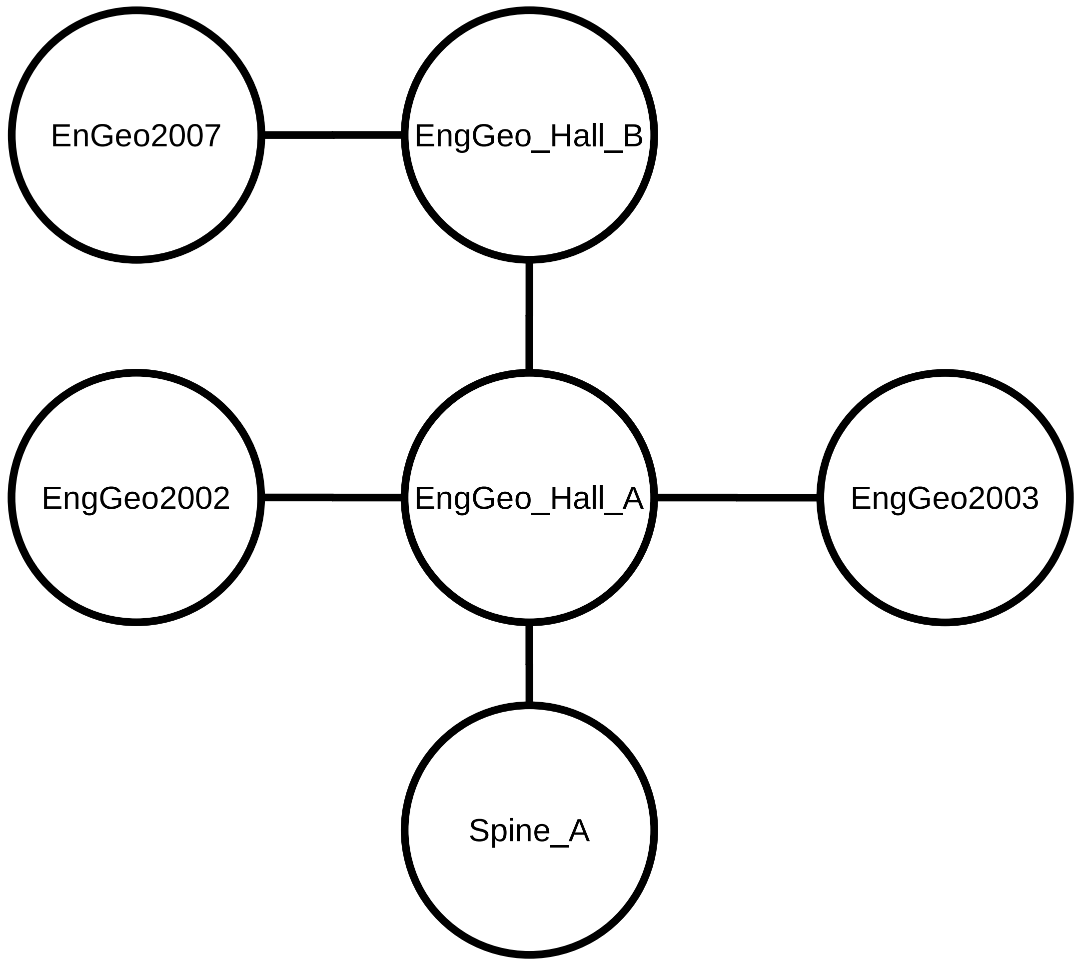
CS354

JAMES MADISON UNIVERSITY®

Representing Maps: Topological

- Represent relative locations using a graph structure.
  - \+ Good for high level navigation
  - \- Difficult to build autonomously
  - \- Not good for low-level localization and navigation



Representing Maps: Geometric Landmark Based

- Store the geometric location of recognizable landmarks.
  - Maybe artificial beacons or markers.
  - Maybe distinctive environmental features.

+ Memory-efficient

+ Allows precise localization

- Landmark mis-identification can cause problems

- May not be ideal for navigation: only landmark positions are stored, not necessarily the positions of all obstacles

Representing Maps: Occupancy Grid

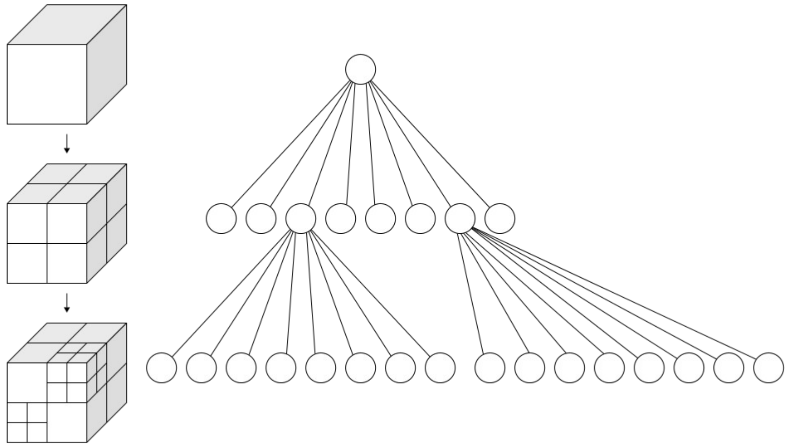
- Divide the environment into grid cells, maintain an "occupied" probability for each cell.
  - \- Memory intensive (particularly in 3D)
  - \+ Good for navigation
  - \+ Good for localization
  - \+ Relatively simple to create autonomously

Quadtrees/Octrees

- Large occupancy grids can be expensive to store:
  - 100m x 100m map, 1cm resolution
  - 100,000,000 cells
- Quadtree is a more space-efficient alternative...

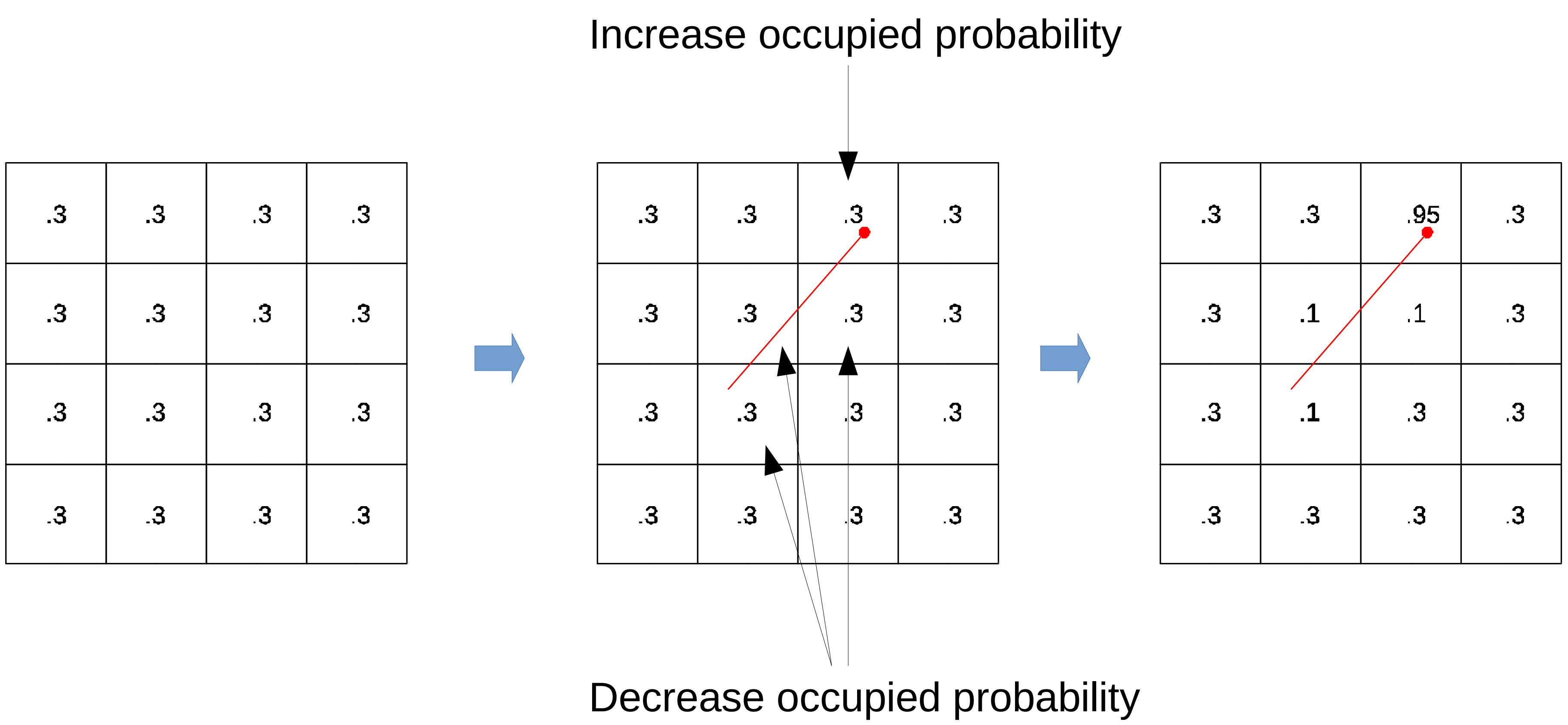
Quadtrees/Octrees

- Large occupancy grids can be expensive to store:
  - 100m x 100m map, 1cm resolution
  - 100,000,000 cells
- Quadtree is a more space-efficient alternative...
- Octree is the 3d generalization:



Mapping w/ Occupancy Grids

- Relatively easy if we know the robot pose:



H.P. Moravec. Sensor fusion in certainty grids for mobile robots. AIMagazine, pages 61–74, Summer 1988.

SLAM – Simultaneous Localization and Mapping

- Recall the localization problem:

$$P(\mathbf{x}_t | \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

- The SLAM problem is reassuringly familiar:

$$P(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t})$$

- Where $\mathbf{m}$ represents the map.
- Before we wanted a probability distribution over all possible robot poses.
- Now we want a joint probability distribution over all possible robot poses and all possible maps.

“Distribution over possible maps” is not as manageable as “distribution over poses”

SLAM "Solution"

- Prediction:

$$Bel^{-}(\mathbf{x}_t, \mathbf{m}) = \int P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) Bel(\mathbf{x}_{t-1}, \mathbf{m}) d\mathbf{x}_{t-1}$$

- Correction:

$$Bel(\mathbf{x}_t, \mathbf{m}) = \eta P(\mathbf{z}_k \mid \mathbf{x}_t, \mathbf{m}) Bel^{-}(\mathbf{x}_t, \mathbf{m})$$

Adapted from Simultaneous Localisation and Mapping (SLAM): Part 1 The Essential Algorithms, Hugh Durrant-Whyte, 2006

SLAM Solutions

- Solutions fall into three families (in roughly historical order)
  - EFK SLAM
  - Particle-Filter SLAM
  - GraphSLAM

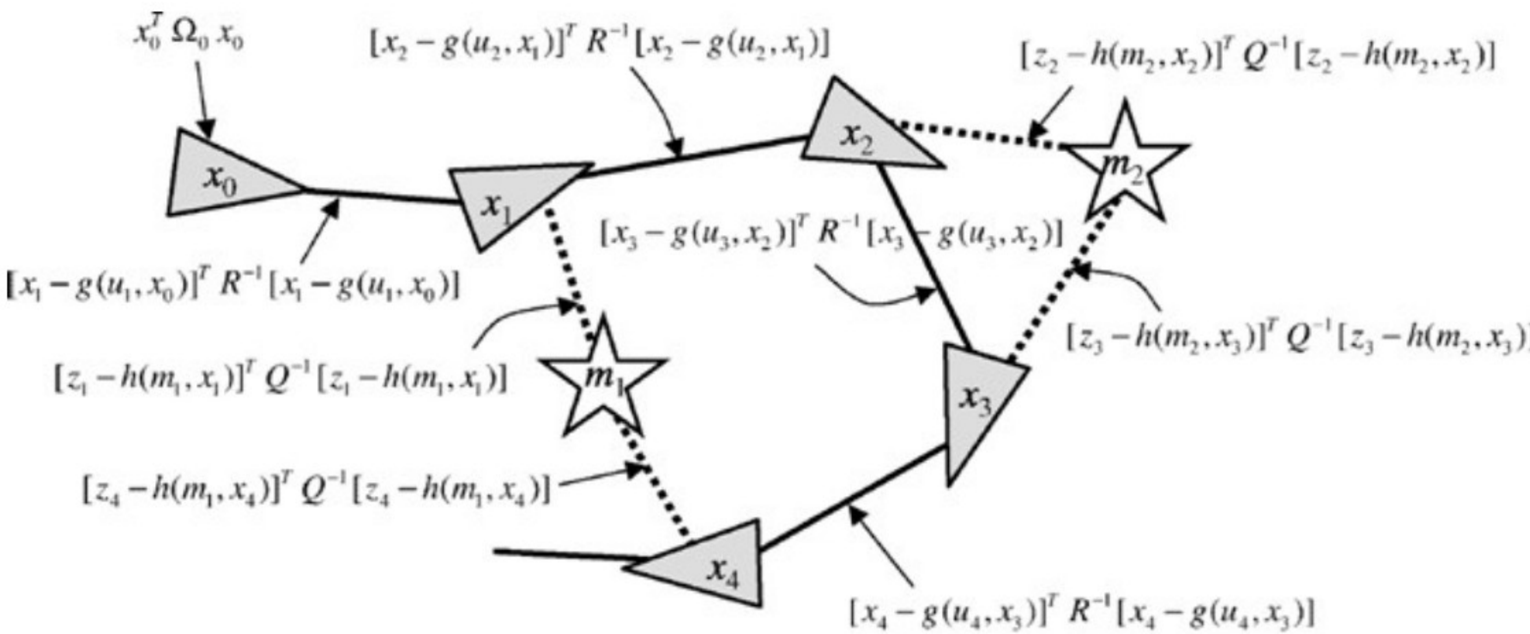
(Extended) Kalman Filter SLAM

- Most appropriate for landmark-based maps.
- Problems:
  - Not clear how to use this for occupancy grids
  - The covariance matrix gets big as the number of landmarks grows
  - Video Example

Particle Filter SLAM: Problems

- Covering the space of possible poses and maps with particles is not practical:
  - “Pose particle”: 3-6 dimensions
  - “Map particle” for a (tiny) 10x10 grid: 100 dimensions
  - Joint map x pose particle: 300-600 dimensions

Graph SLAM



Sum of all constraints:

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

Thrun, Sebastian, and Michael Montemerlo. "The graph SLAM algorithm with applications to large-scale mapping of urban structures." The International Journal of Robotics Research 25.5-6 (2006): 403-429.

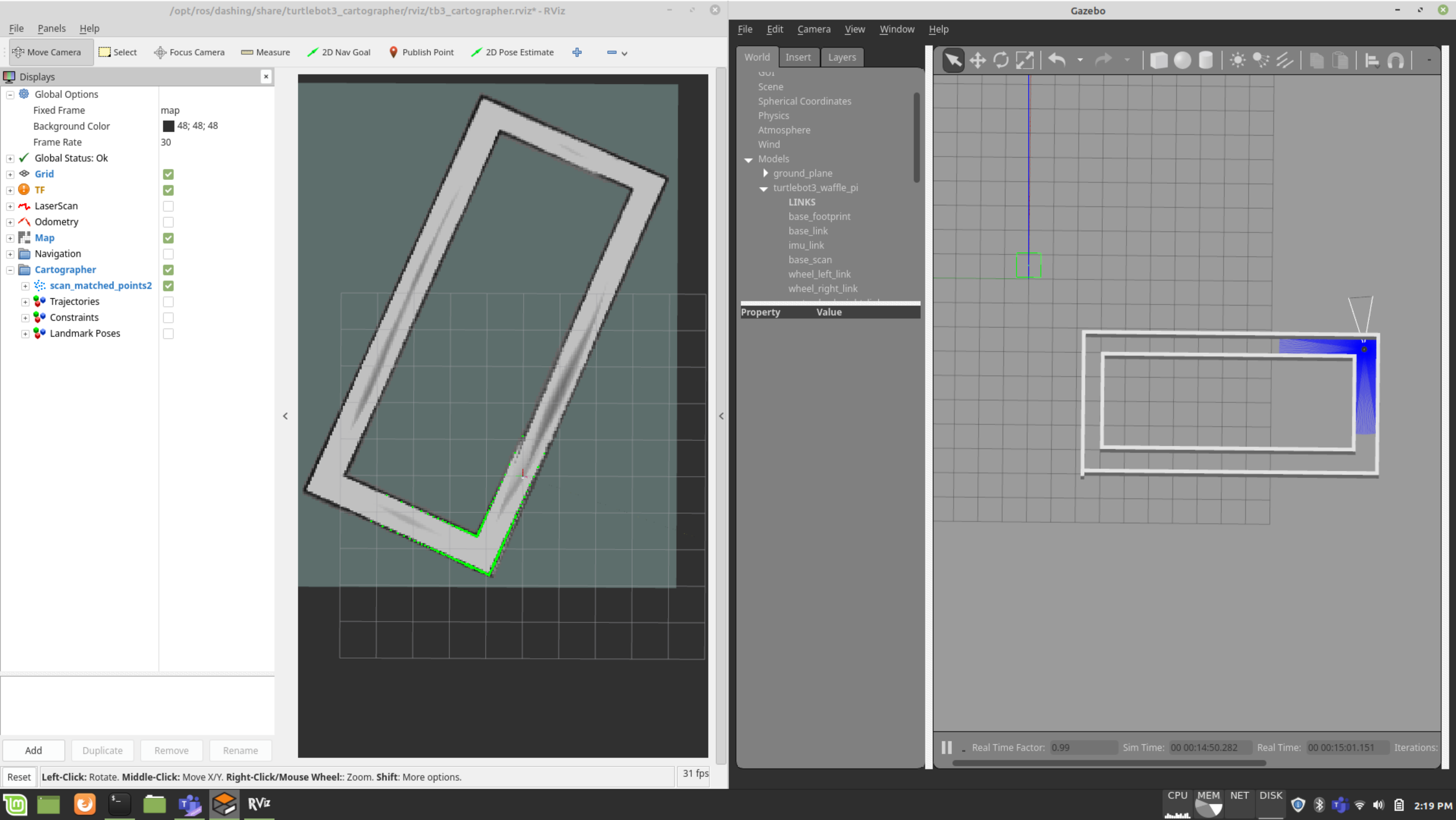
Rao-Blackwellized Particle Filter for SLAM

- Solution/Approximation:
  - Each pose particle has an associated map.
  - Each map is updated under the assumption that its particle represents the correct pose.
  - The map may be landmark-based or occupancy grid-based.

Challenges

- LOOP CLOSURES!!!!

Successful Loop Closure



Incorrect Loop Closure