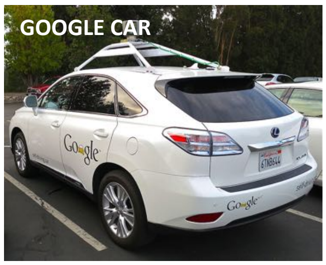
CS 189: Autonomous Robot Systems

Spring 2018, Fridays 1-4pm, Pierce 301

Agenda

- **Lecture: Robot Navigation -> MAPPING!**
- Demo Time:
  - LAB4 (Extended Kalman Filter*)
  - Then help TFs take robots down to MD B127 (Pset 5 test arena)
- Upcoming:
  - **Pset 5: Autonomous Mapper due next week**
  - **Start ASAP! (uses Lab 4, in MD B127)**
- References:
  - This lecture is partially based on “Introduction to AI Robotics”, chapter 11, Robin Murphy, 2000,
  - For SLAM, see online theory tutorial paper “SLAM: Part 1 The Essential Algorithms”, by Durrant-Whyte et al, 2006 and online practical tutorial paper “SLAM for Dummies” S. Riisgaard, and M. Blas. (2005)

Today: Robots Navigating the World



Scenarios

- *Hospital Helper (e.g. Diligent, Tugs)*
- *Office security or mail-delivery (e.g. Cobal, Savioke)*
- *Tour Guide robot in a museum (Minerva)*
- *Autonomous Car with GPS and Nav system*

***Biological analogies**: Humans, bees and ants, migrating birds, herds*

Today: Robots Navigating the World

Second Part of CS189: High-level reasoning

From finite state machines to complex representation and memory

- Path Planning: *How to I get to my Goal?*
- Localization: *Where am I?*
- Mapping: *Where have I been?*
- Exploration: *Where haven't I been?*

Mapping and Exploration

- **Question:**
  You are roaming around in an unknown space, what can you learn about it?
- Two parts of the problem:
  - Mapping: As you roam around the world, how do you build a memory of the shape of the space you have moved through?
  - Exploration: (coverage of unknown space) Given that you don't know the shape or size of the environment, how do make sure you covered all of it?
- Both have many uses:
  - Searching for objects, Mapping a collapsed mine or building.
  - Mowing a golf course or cleaning a room efficiently.
- Mapping and Exploration are also "collections of algorithms"
  - E.g. Many representations of a "map" can be
  - E.g. Random walks are a form of exploration that does come with guarantees
  - We will focus on "Occupancy Grid" algorithms

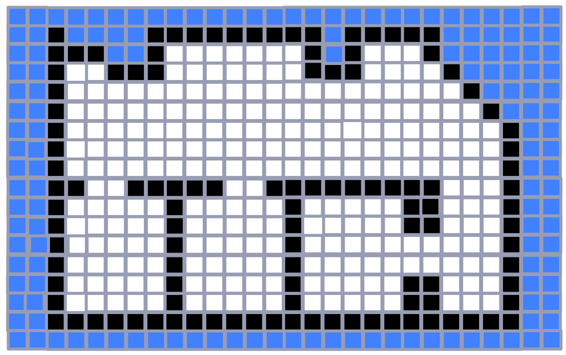
Today's topics

- Mapping and Exploration Algorithms
  - Occupancy Grids and Sensor Models
  - A First-cut Simple Mapping Algorithm
- Three Improvements
  - Exploration strategies
    - Frontier based exploration (guaranteed coverage)
  - Managing sensor uncertainty
    - Probabilistic algorithms for Occupancy Grid mapping (Bayes Rule)
  - Managing motion uncertainty
    - Briefly: Simultaneous Localization and Mapping (SLAM)
- Pset 5: Your Autonomous OG Mapper!

What is an Occupancy Grid?

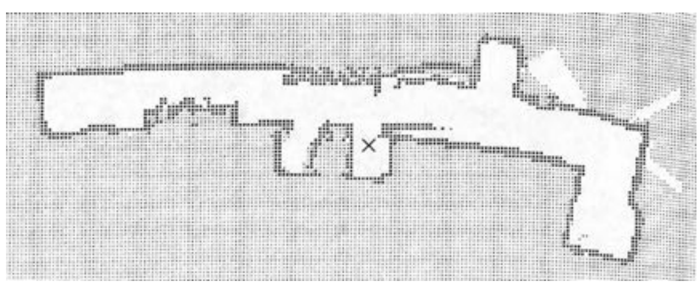
- A way of representing a map as a gridded world where each cell is either “occupied” or “empty” or “unknown”.

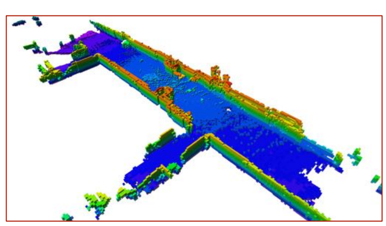
Your World

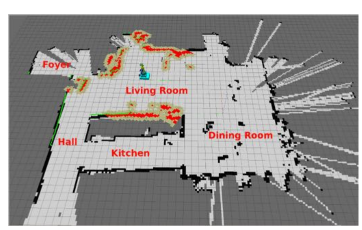


Grid generated by a Robot => boundary shape

Examples







What is a Sensor Model?

- Step1: Constructing a Sensor Model
  - A sensor measures *raw values* in an environment
  - You have to map that into a Grid Cell Value.
  - Robots can have very different sensors and configurations
  - Examples:
    - Think about LIDAR/Depth Camera
    - Vs. a 360 degree vision/ranging system

Constructing a Sensor Model

Example: Depth Sensor Model

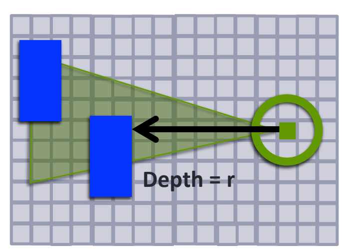
R = maximum range, B = maximum angle

Let say the sensor at point p returns **depth = "r"**

Region 1 (dist < r, grid cell probably empty)

Region 2 (dist = r, grid cell probably obstacle)

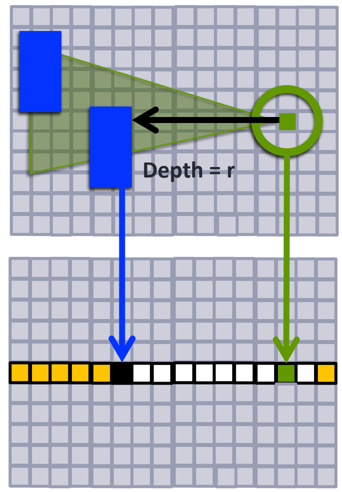
Region 3 (dist > r, grid cell unknown/obscured)



Constructing a Sensor Model

Example: Depth Sensor Model
R = maximum range, B = maximum angle
Let say the sensor at point p returns **depth = “r”**

Region 1 (dist < r, grid cell probably empty)
Region 2 (dist = r, grid cell probably obstacle)
Region 3 (dist > r, grid cell unknown/obscured)



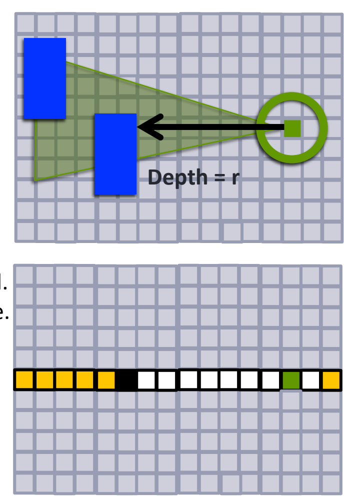
Constructing a Sensor Model

Example: Depth Sensor Model
R = maximum range, B = maximum angle
Let say the sensor at point p returns **depth = “r”**

Region 1 (dist < r, grid cell probably empty)
Region 2 (dist = r, grid cell probably obstacle)
Region 3 (dist > r, grid cell unknown/obscured)

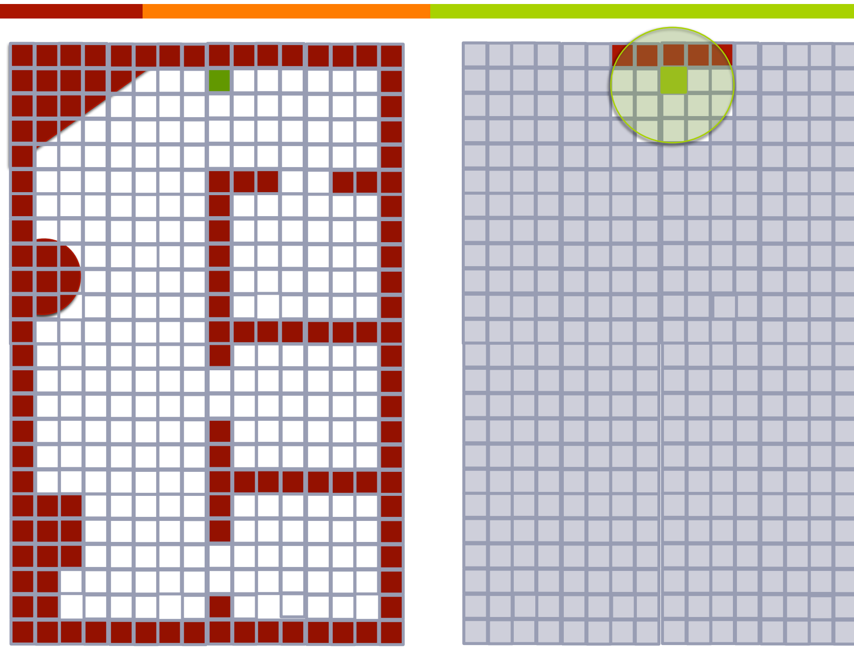
Simple Model:
Set region 1 cells as empty, region 2 cells as occupied.
Pick a Maximum Range/Angle where depth is reliable.

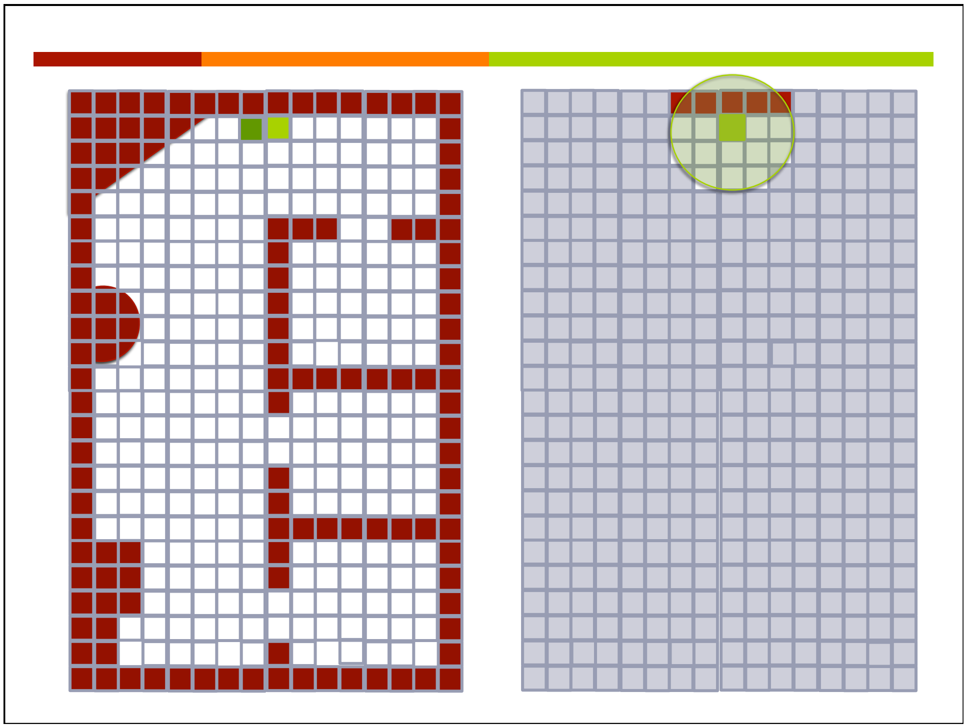
More Complex Model:
For a cell at distance r and angle a
P(correctness) = [(R-r/R) + (B-a/B)]/2
i.e. Uncertainty in my assessment grows with distance and angle from the centerline

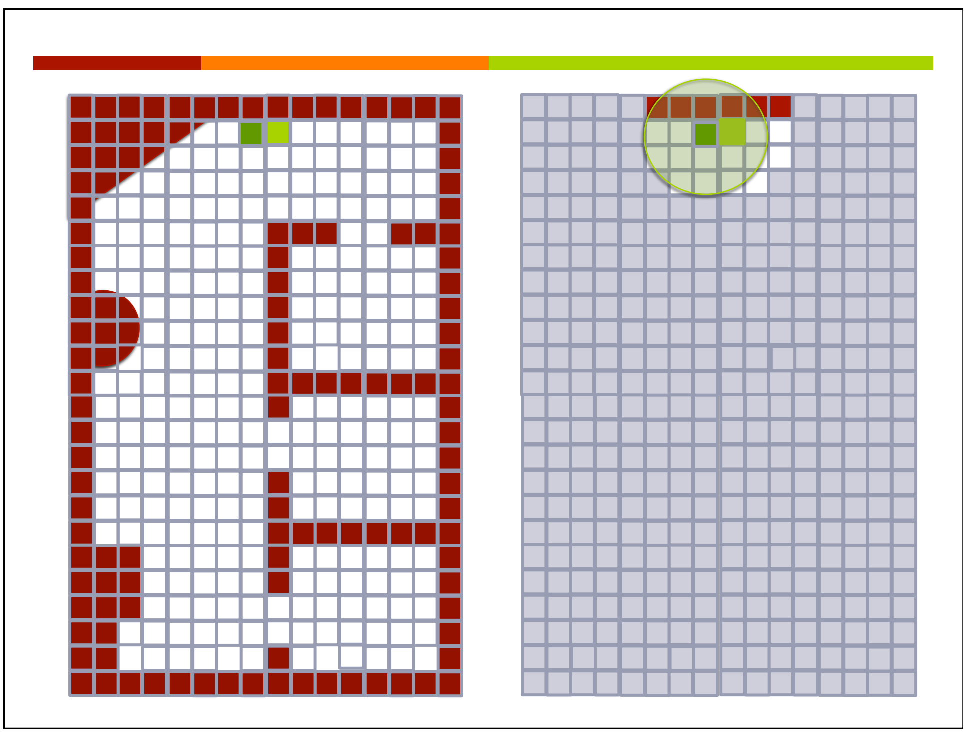


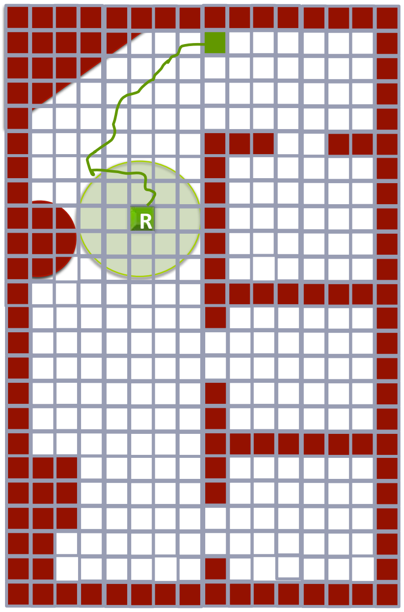
A Simple OG Mapping Algorithm

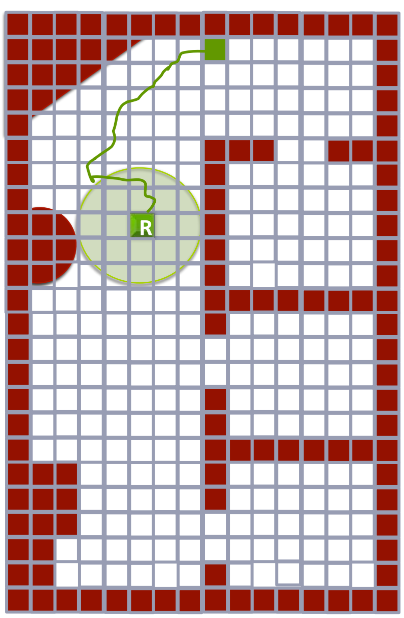
1. Initialize a Grid
   - Set all locations as "unknown", pick a start location and orientation
2. Update the Grid
   - *Mark your current grid position as "empty"*
   - Using your simple sensor model,
     Mark all visible grid locations as "empty" or "occupied"
3. Pick a Next Move
   - Look at neighboring grid positions in your map
   - Pick a neighboring grid location that is empty (randomly)
   - Move to it and update your current position in the Grid
4. Loop forever
   Keep moving and updating the grid (unless you are "done")

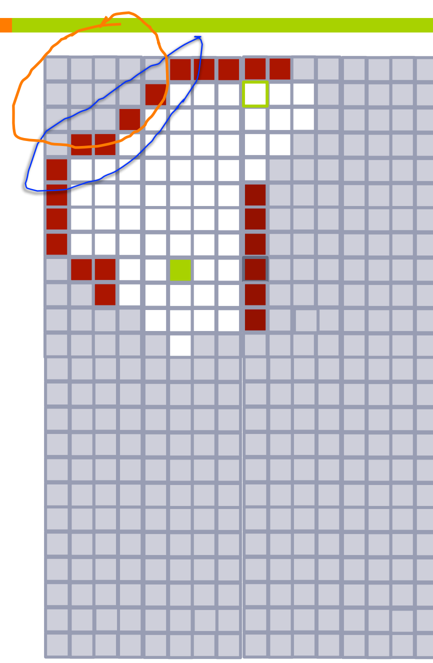







R


R



A Simple Mapping Algorithm

1. Initialize Grid
2. Update the Grid
   - Mark your current position as "empty"
   - Mark sensed nearby grid locations
     As "empty" or "occupied"
3. Pick a Next Move
   - Look at neighboring grid positions
   - Choose a random empty direction
   - Move and update your position in the Grid
4. Loop forever

Improvement 1: Exploration Strategy

Better to systematically and (hopefully) efficiently cover the space.

Also would be good to know when you are done.

Exploration

- Basic Concept in Math: Random Walks in bounded 2D
  - With Probability=1 you will *eventually* visit every spot
- Basic Concept in CS: Systematic Graph Coverage
  - You are given a "graph" with V nodes
    Write an algorithm that visits all of the nodes
  - BFS, DFS, Time Complexity: O(V+E)
- Basic Concept in Robotics: Traversing a GRID Graph is different
  - DFS works, but will still make a robot retrace steps
  - Ideal: Visit every node exactly once (Hamiltonian path, NP-complete!)
  - **Better choice: Frontier Based Exploration**

Exploration in Grid Worlds

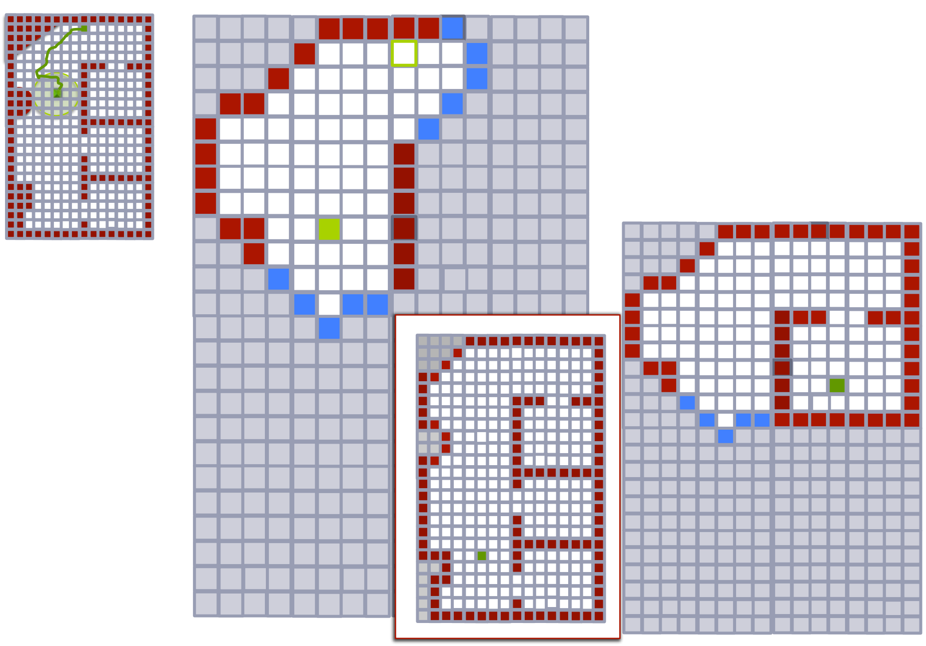
- **Frontier Based Exploration**
  - A common technique for building maps
  - Key Idea:
    - Identify the “frontiers” between known and unknown

      Frontier cell = a unknown cell with at least one empty cell nbr
    - Pick a frontier cell (e.g. the closest)
    - Plan a path to go explore it.

  - Done Condition:
    - No more frontier nodes left!

    If finite world, then any algorithm that systematically explores frontier nodes is guaranteed to cover the whole world.

    Can use this condition by itself to determine if your map is complete.



A Less Simple Mapping Algorithm

1. Initialize Grid
2. Update the Grid
   - Mark your current position as "empty"
   - Mark sensed nearby grid locations
     As "empty" or "occupied"
3. **Pick a Next Move**
   - Identify ***frontier cells***
   - Pick one (e.g. maybe the closest)
   - Plan a path* to the ***nbr empty cell***.
   - Go to that location using this path
     (and keep track of your position as you move)
4. Loop until no frontier nodes are left

Improvement 2:
Sensors aren't perfect

Take advantage of the fact that you are often retracing steps

And taking measurements multiple times of the same location

* We covered path planning two lectures ago

Bayesian Mapping

For every grid location (i,j), store a probability value

P(Occupied) = Probability this grid location is Occupied 0 <= P(Occupied) <=1

P(Empty) = 1 - P(Occupied)

A More Compex Sensor Model
P(s|Occupied)
Probability that you sense value **s** given that a grid location is occupied.
Determine this experimentally

Mapping
P(Occupied|s)
Probability that a grid location is occupied given that you sensed value **s**
We can compute this!

Bayes Rule

$$P(Occupied|s) = \frac{P(s|Occupied)\,P(Occupied)}{P(s|Occupied)P(Occupied) + P(s|Empty)\,P(Empty)}$$

Bayes Update Rule

$$P(Occupied|s_n) = \frac{P(s_n|Occupied)\,P(Occupied|s_{n-1})}{P(s_n|Occupied)P(Occupied|s_{n-1}) + P(s_n|Empty)\,P(Empty|s_{n-1})}$$

Bayesian Mapping

- In the beginning of time,
  - P(Occupied) = P(Empty) = 0.5

Bayes Update Rule:

P(Occupied|sn) = P(sn|Occupied) P(Occupied|sn-1) / [P(sn|Occupied) P(Occupied|sn-1) + P(sn|Empty) P(Empty|sn-1)]

- For grid(i,j), lets say s=6 (depth sensor value)
  - P(s=6|Occupied) = 0.62
    P(s=6|Empty)=0.38
    P(Occupied)=P(Empty)=0.5
  - **P(Occupied|s=6) = (0.62*0.5) / (0.62*0.5) + (0.38*0.5) = 0.62**
    Which is what you'd expect because we have no better knowledge
- Later if we observe location grid (i,j) again, we have *prior* knowledge
  - We now think P(Occupied)=0.62 P(empty)=0.38
  - New sensor reading P(s=s'|Occupied) = x
  - **P(Occupied|s=s') = (x*0.62) / (x*0.62)+(1-x)*0.38 = new confidence**

Probabilistic Mapping

- Overarching idea
  - Store *probabilities* of occupancy rather than binary values.
- But you periodically must turn probability into Occupied/Empty!
  - Otherwise, how do you move?
    - Use some threshold to decide
      - P(occupied) > 0.7 and P(empty) < 0.3, rest is "unknown".
    - Then do frontier exploration and path planning on your deterministic map.

A Probabilistic OG Mapping Algorithm

1. Initialize Grid to 0.5
2. Update the Grid
   - Mark your current position as high probability "empty"
   - Use your sensor model and Bayes rule to update grid
3. Pick a Next Move
   - Threshold your map into empty, occupied, unknown
   - Identify frontier nodes, and pick one
   - Plan a path to the clear node nearest frontier
   - Go to that location and update position
4. Loop until no frontier nodes are left

Probabilistic Mapping

- Overarching idea
  - Store *probabilities* of occupancy rather than binary values.
- This is great! So what can go wrong?
  - Motion uncertainty!!!
  - (1 Lecture back: Kalman Filter and Particle Filter)

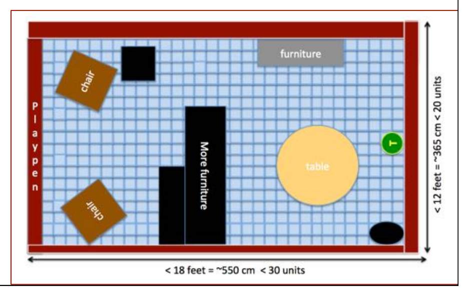
Pset 5: The Autonomous OG Mapper

Digression ---- Mapping A Fake Office!

Generate a map for navigation in the office (B127)

- Setup an Occupancy Grid (deterministic)
- Construct a Depth Sensor Model for the Turtlebot
- Use a Simple Exploration strategy (random)
- Use EKF (lab 4) for localization.
- Output the map.

Optional: If you get the above working really well, then take a video and map of your work, and try more complex ideas from this lecture.



Probabilistic Localization and Mapping

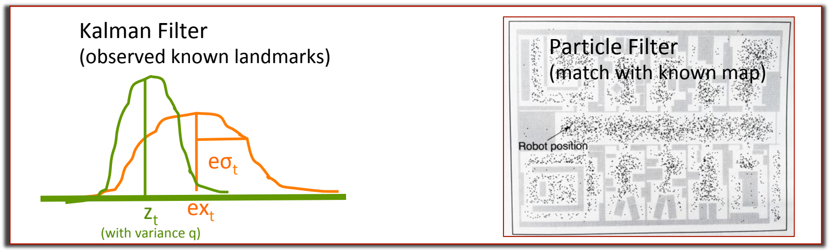
- **Probablistic Localization**
  - **$P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}\ \text{map})$**
  - Where am I? Given that I took the noisy actions U and noisy observations Z of things in my perfect map/landmarks.

1 lecture ago:
Kalman Filters
Particle Filters

Probabilistic Localization and Mapping

- **Probablistic Localization**
  - **$P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}$ map)**
  - Where am I? Given that I took the noisy actions U and noisy observations Z of things in my perfect map/landmarks.

1 lecture ago:
Kalman Filters
Particle Filters



Probabilistic Localization and Mapping

- **Probablistic Localization**
  - **$P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}$ map)**
  - Where am I? Given that I took the noisy actions U and noisy observations Z of things in my perfect map/landmarks.

1 lecture ago:
Kalman Filters
Particle Filters

- **Probablistic Mapping**
  - **P(map I $Z_{0\text{-}t}$, $U_{0\text{-}t}$)**
  - What is my map like? Given that I made noisy observations Z as I walked along my perfect path dictated by U

Today:
Bayesian Occupancy Grids

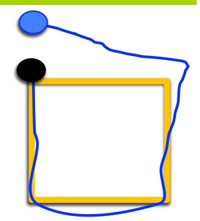
Probabilistic Localization and Mapping

- **Probablistic Localization: $P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}$ map)**
- **Probablistic Mapping: P(map I $Z_{0\text{-}t}\ U_{0\text{-}t}$)**
- **Probablistic SLAM ("Simultaneous")**
  - **$P(x_t$, map| $Z_{0\text{-}t}\ U_{0\text{-}t})$**
  - Where am I and what is my map?
  - Given noisy actions U and made noisy observations Z
  - Distribution of a huge space! (all possible positions and maps)
- **Many Methods**
  - EKF-SLAM (Kalman Filter) and Fast-SLAM (Particle Filters)

Extended Kalman Filter SLAM

- In original EKF, robot position is a Gaussian ($x_t\ \sigma_t$)
- In EKF-SLAM, robot and all landmark positions are Gaussians
  - **State = {x_t, m1, m2, m3 ……mn}** (number of landmarks grows!)
  - **Co-variance = (n+1)x(n+1) matrix** (uncertainty in correlated!)
  - *Good news!: Correlations can help you converge faster and better!*
- Basic Procedure is similar (w much more math! Often offline)
  - **Alternate Three Steps**
    - Motion Step: Update $P(x_t$, map | $Z_{0\text{-}(t\text{-}1)}\ U_{0\text{-}t})$ based on action U_t
    - Observation Step: Update $P(x_t$, map | $Z_{0\text{-}t}\ U_{0\text{-}t})$ based on Z_t
    - Add New landmarks to the State
  - Important – these steps update the Whole Map (not just what you see)
    - Captures correlations between landmarks, with high certainty!
    - Many advances (FastSLAM) to make the process real-time.

More About SLAM



- Data Association and Loop Closure
  - We don't really have landmarks
    - Instead we have depth camera "features"
    - Local matching is easier than long term matching
- Practical Implementations
  - These algorithms are theoretically well-grounded
  - But practical implementation still requires significant work, including understanding environment (constructing sensor/motion models)
- References (online)
  - SLAM Part 1: The Essential Algorithms, Durrant et al, 2006 (in theory)
  - SLAM for Dummies, Riisgaard et al 2005 (in practice)
  - Gmapping in ROS! (PRR chapter 9 = offline map making)

Conclude: Robots Navigating the World

Second Part of CS189: High-level reasoning

From finite state machines to complex representation and memory

- PathPlanning: *How to I get there?*
- Localization: *Where am I?*
- Mapping: *Where have I been?*
- Exploration: *Where haven't I been?*

Preview of Rest of Term

Lab 4 and Pset 5
Mapping
Final Project
Warehouse
Upcoming lectures
Automation Ethics
Darpa Challenge
Multi-Robot systems