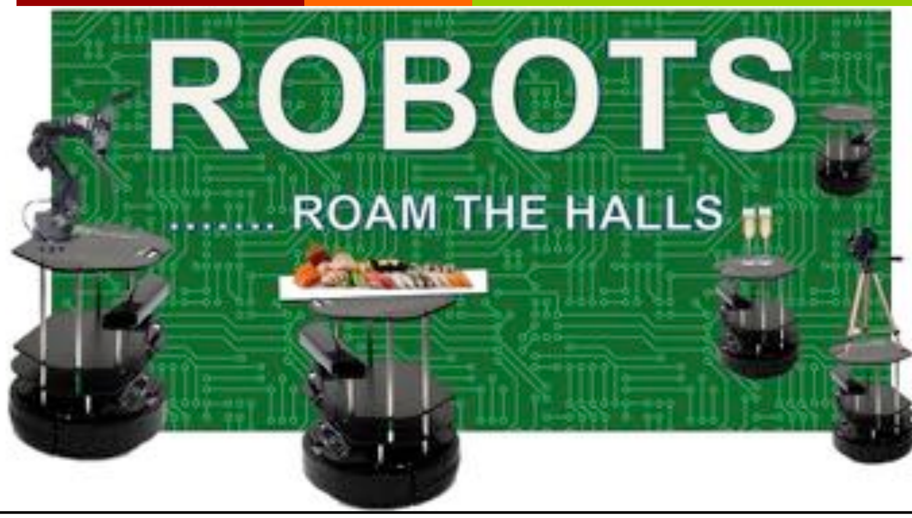
CS 189: Autonomous Robot Systems

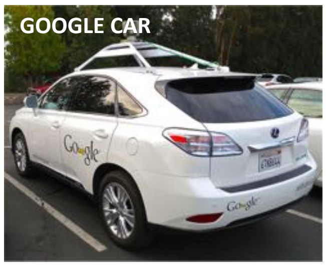
Spring 2019



Agenda

- **Lecture: Robot Navigation -> MAPPING!**
- Demo Time:
  - Visit B127 and Talk about Pset 4 and Final project
- Upcoming:
  - **Pset 4a: Autonomous Mapper due next week (map B127, start ASAP!)**
  - **Lecture next week: Automation Ethics**
    - **Meet in Pierce 301 at 9am (after will go to B127 for Pset4a)**
    - **Videos to watch ahead of time (posted on Piazza)**
- 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: 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:
  - Returning back to home/charger after some task.
  - Cleaning a new room efficiently; Systematic search for survivors
  - Mapping a collapsed mine or building.
- Mapping and Exploration are also "collections of algorithms"
  - E.g. Many representations of a "map"; random walks are exploration
  - We will focus on "Occupancy Grid" algorithms

Today's topics

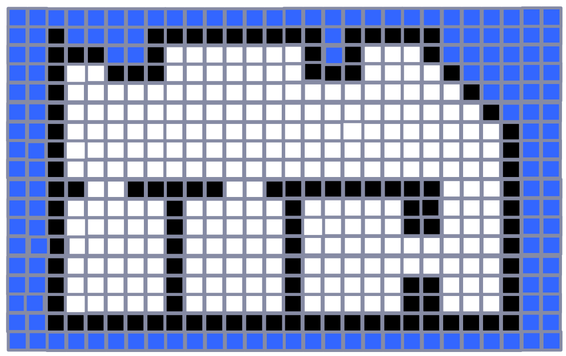
- Mapping and Exploration Algorithms
  - Occupancy Grids and Sensor Models
  - A First-cut Simple Mapping Algorithm
- Three Improvements
  - Exploration strategies
    - Frontier based exploration (guaranteed coverage)
  - Managing sensor uncertainty
    - Probabilistic algorithms for Occupancy Grid Mapping (Bayes Rule)
  - Managing motion uncertainty
    - *Briefly:* Simultaneous Localization and Mapping (SLAM)
- Pset 4: Your Autonomous OG Mapper!*

  * uses material from all 3 navigation lectures

What is an Occupancy Grid?

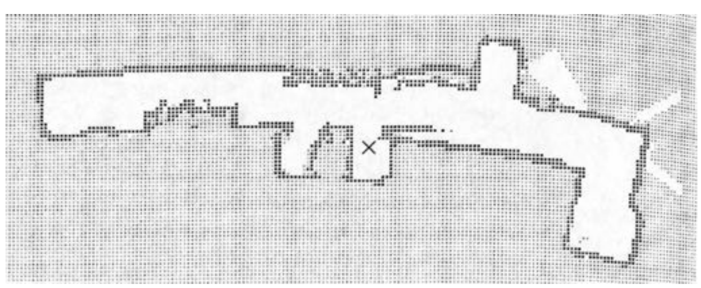
- A way of representing a map as a gridded world where each cell is either “occupied” or “empty” or “unknown”.

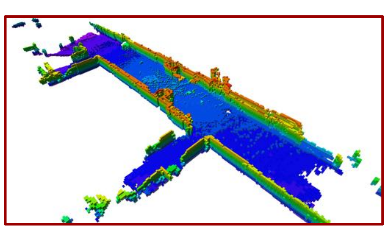
Your World

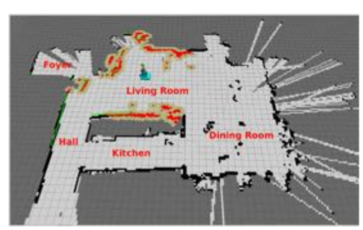


Grid generated by a Robot => boundary shape

Examples







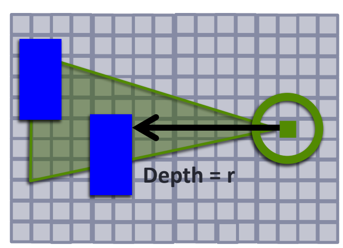
What is a Sensor Model?

- 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 **distance = "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

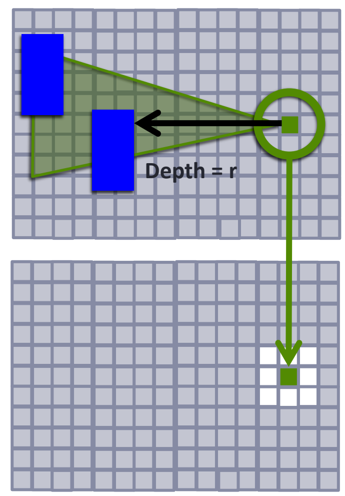
Example: Depth Sensor Model

R = maximum range, B = maximum angle

Let say the sensor at point p returns **distance = "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)

Simplest Sensor Model
Where I stand is Empty (white)



Constructing a Sensor Model

Example: Depth Sensor Model

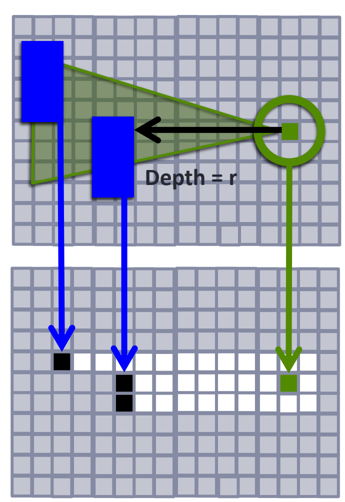
R = maximum range, B = maximum angle

Let say the sensor at point p returns **distance = "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)

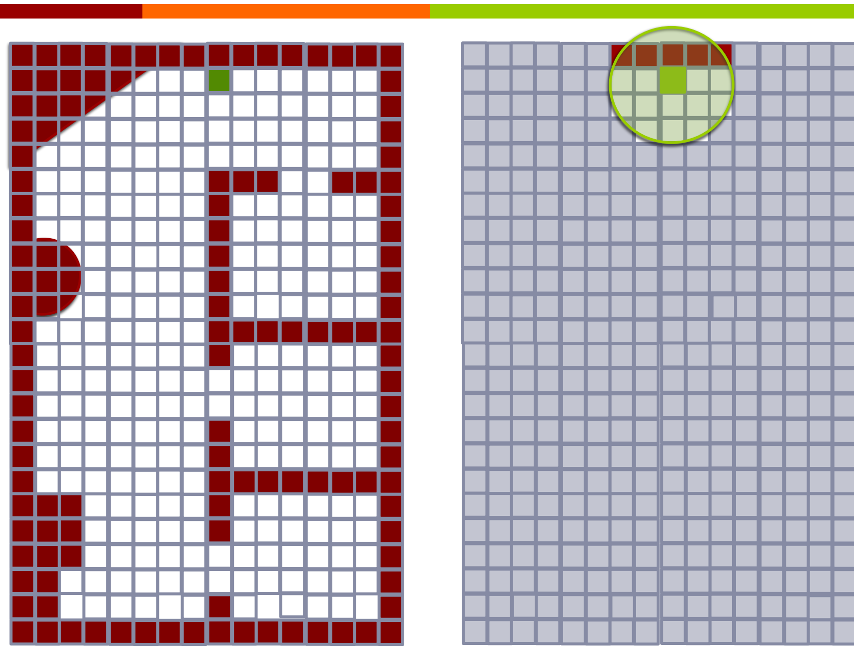
Simplest Sensor Model
Where I stand is Empty (white)

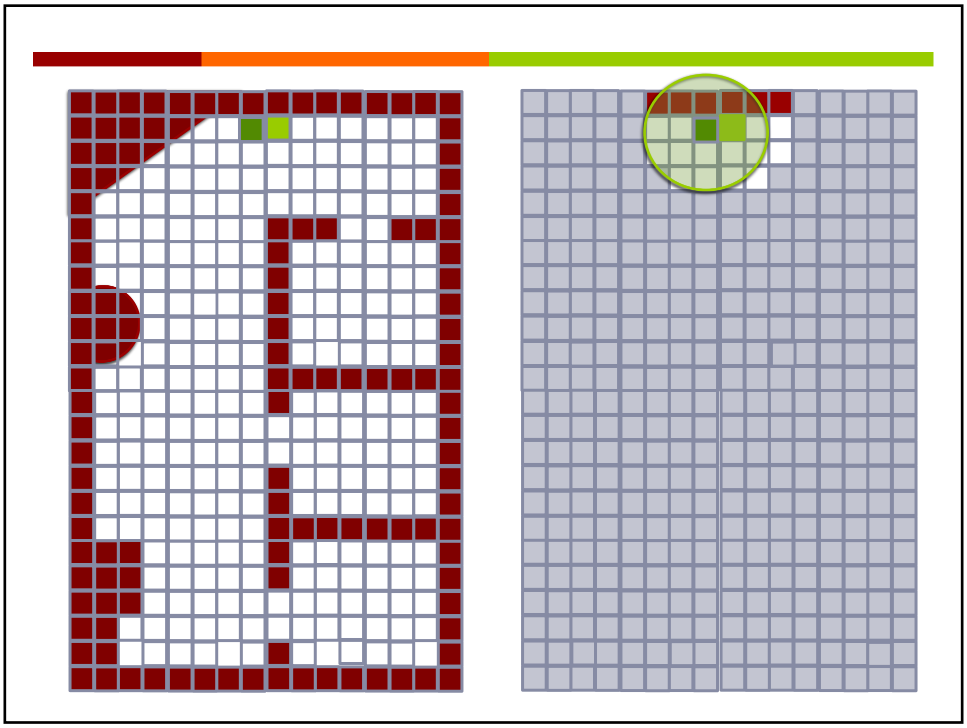
A Better Model
Set Region 1 cells as Empty (white)
Set Region 2 cells as Occupied (black).
Pick a max range/angle where data is reliable
Rest is still Unknown (gray)

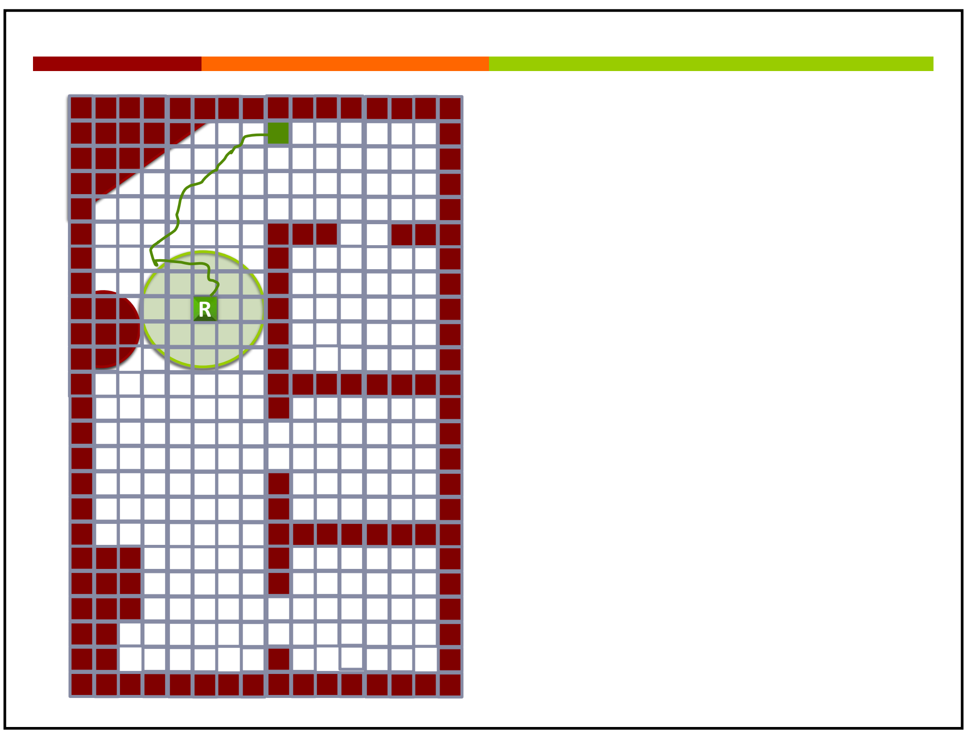


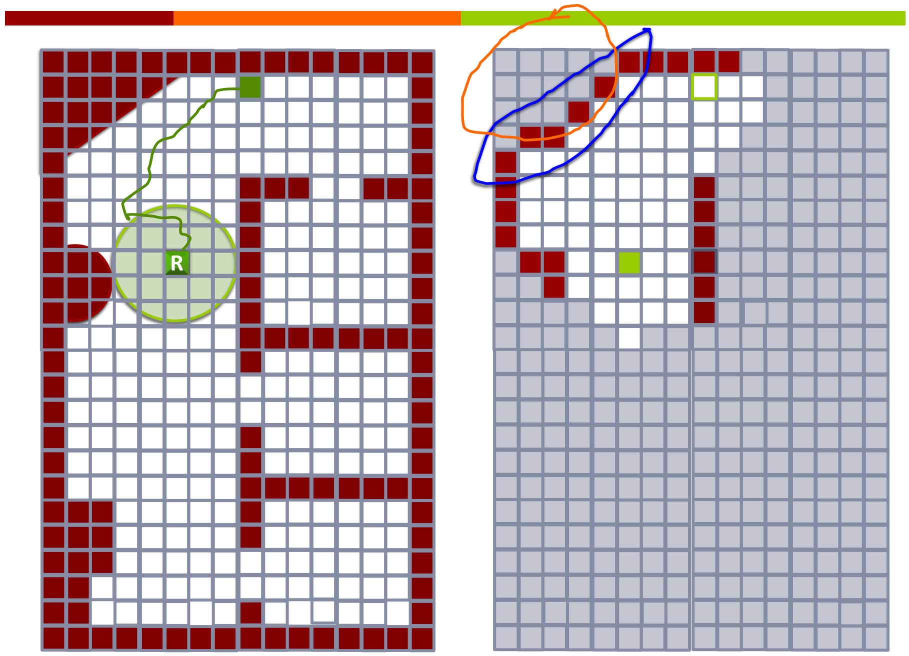
A Simple OG Mapping Algorithm

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 better 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



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

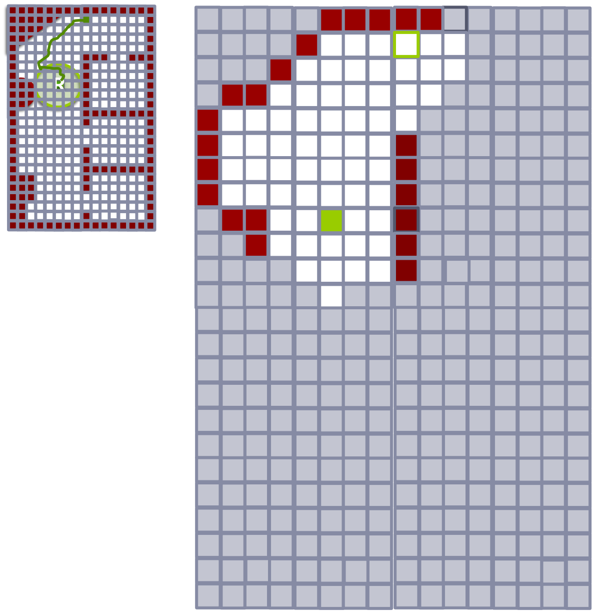
    Breath-First Search and Depth-First Search; Time Complexity: O(V+E)
- Basic Concept in Robotics: Traversing a GRID Graph is different
  - DFS works, but will still make a robot retrace steps
  - **Better choice: Frontier Based Exploration**

Exploration in Grid Worlds

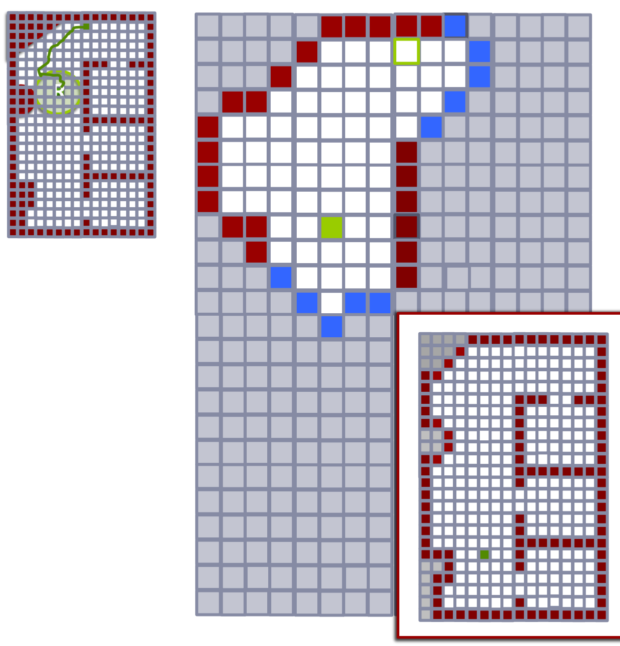
- **Frontier Based Exploration**
  - A common technique for building maps
  - Key Idea:
    - Identify the "frontiers" between known and unknown

      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 => your map is Complete!

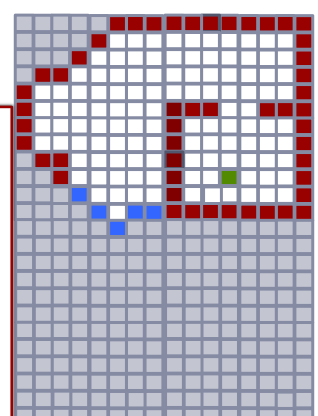
    If finite world, then any algorithm that systematically explores frontier nodes is guaranteed to cover the whole world.



A Frontier Node is a
Gray node (Unknown)
next to a
White node (Empty)



A Frontier Node is a
Gray node (Unknown)
next to a
White node (Empty)



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

* We covered path planning two lectures ago

Pset 4: The Autonomous OG Mapper

Digression ---- Mapping A Fake Office! *(B127)*

- Part 4a
  - Setup an Occupancy Grid Map (dimensions given)
  - Use Lab4 EKF package to keep track of where you are (aka localization)
  - Use a very simple wandering strategy (aka exploration)
  - Use a very simple sensor model to mark OGMap
  - Display your OGMap (display code provided)
- Part 4b
  - Modify above to use Better Sensor Model
  - Modify above to use Frontier Exploration

Part a is due next Friday! Start early!

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 4: Your Autonomous OG Mapper!*

* uses material from all 3 navigation lectures

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 nbring 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

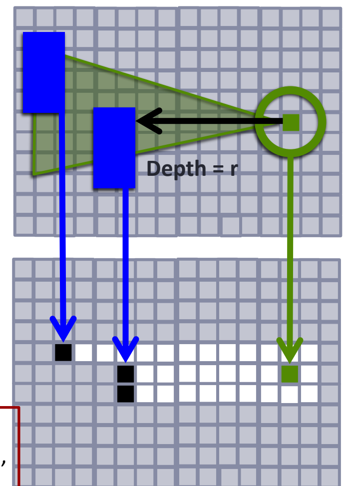
Take advantage of the fact that you are often retracing steps

And taking measurements multiple times of the same location

A Probabilistic Sensor Model

Example: Depth Sensor Model
R = maximum range, B = maximum angle
Let say the sensor at point p returns **distance = "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)



A More Complex Sensor Model: Probabilistic
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

Intuition: Build confidence!
The more times I independently visit/sense a grid cell, *the more certain I am* about my conclusion.

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 Complex Sensor Model
P(s|Occupied)
Probability that you sense value **s** given that a grid location is occupied.
Your sensor error model

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:

$$P(Occupied|sn) = \frac{P(sn|Occupied)\ P(Occupied|sn\text{-}1)}{P(sn|Occupied)\ P(Occupied|sn\text{-}1) + P(sn|Empty)\ P(Empty|sn\text{-}1)}$$

- 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=2|Occupied) = .80
  - **P(Occupied|s=2) = (0.8*0.62) / (0.8*0.62)+(0.2*0.38) = 0. 87**
    (my new confidence is higher, that this grid cell is occupied)

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 probabil
   - Use your sensor model and Bayes rule to up
3. Pick a Next Move
   - Threshold your map into empty, occupied, u
   - Identify frontier nodes, and pick one
   - Plan a path to the clear node nearest
   - Go to that location and update position
4. Loop until no frontier nodes are left

Improvement 3:
Motion isn't perfect either!

Maybe you are not where you think you are!

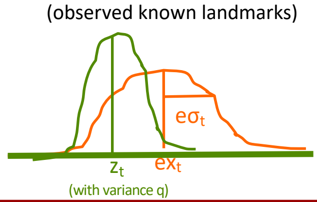
And you are just messing up your grid over time due to drift

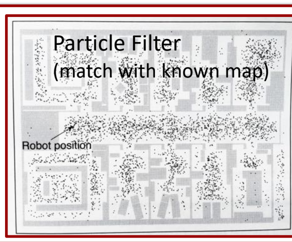
Probabilistic Localization and Mapping

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

1 lecture ago:
Kalman Filters
Particle Filters

Kalman Filter
(observed known landmarks)





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

1 lecture ago:
Kalman Filters
Particle Filters

Today:
Bayesian Occupancy Grids

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.

My **autonomous mini-rover** keeps track of its position using its wheel encoders, IMU, and occasionally gets GPS signals

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

Its goal is to **construct a map of the disaster area** obstructions, so that other vehicles can find safe paths

Probabilistic Localization and Mapping

- You took a time series of Actions U and Observations Z
  - **Probablistic Localization: P(x_t | Z_{0-t} U_{0-t} map)**
  - **Probablistic Mapping: P(map I Z_{0-t} U_{0-t})**
- **Probablistic SLAM ("Simultaneous")**
  - **P(x_t, map| Z_{0-t} U_{0-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/OG)

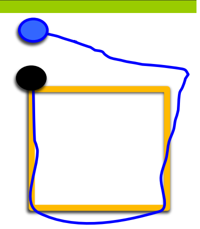
Extended Kalman Filter SLAM

- In original EKF,
  - State == robot position, represented as a Gaussian (x_t σ_t)
- In EKF-SLAM,
  - State = [robot and all landmark] positions as Gaussians
  - **Position X_t = {x_t, m1, m2, m3 … mn}** (number of landmarks grows!)
  - **Co-variance σ_t = (n+1)x(n+1) matrix** (uncertainty is correlated!)
  - Supply a motion model and observation model as before (Gaussian)
- Interesting factors
  - Number of landmarks (n) grows with time (i.e. you build a map).
  - But good news: Landmark correlations can help you converge faster and better.

Extended Kalman Filter SLAM

- Lets say EKF-SLAM State at time t is
  - **Position X = {x, m1, m2, m3, m4}** (robot + landmarks-so-far)
  - **Co-variance σ = 5x5 matrix** (uncertainty and correlations)
- Basic Procedure: Three Steps (Repeat)
  1. Motion Step: Update P(x_t, map | $Z_{0-(t-1)}$ U_{0-t}) based on action U_t
  2. Observation Step: Update P(x_t, map | Z_{0-t} U_{0-t}) based on Z_t
     Data Association: Determine which landmarks are re-observed* (lets say m2 m3)
     Your motion state estimate = xt, m2' m3' (where you expect to see these landmarks)
     Your observation estimate = xt'' m2'' m3'' (where you see landmarks & think you are)
     Kalman Gain = Compute relative confidence and combine estimates
     NOTE: The whole map gets updated! (m1-m4), thanks to co-variance matrix
  3. Add Landmarks: Add New landmarks to the State (say m5)
- Important – implementing Data Association and landmark choice!

More About SLAM



- Data Association and Loop Closure
  - We don't really have perfect landmarks
    - Instead we have laserscan "features" (e.g. major corners)
    - Tradeoff: Uniqueness and frequency
    - *Local matching is easier than long term matching*
    - *Can do loop closure with human assistance.*
- Practical Implementations
  - These algorithms are theoretically well-grounded
  - But practical implementation still requires significant work (e.g. constructing sensor/motion models, choosing landmarks.)
- References (online)
  - SLAM Part 1: The Essential Algorithms, Durrant et al, 2006 (theory)
  - SLAM for Dummies, Riisgaard et al 2005 (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
representati

- PathPlanning: *How*
- Localization: *Where*
- Mapping: *Where ha*
- Exploration: *Where*

Brief Preview of Rest of Term

Pset4
Mapping (2 phases)
Final Project
Robot Candy Store ("Warehouse")
[Lecture in Pierce 301; rest in B127]

Upcoming "Applications" lectures
Robot Ethics about Automation
Humanoid Robots & Darpa Challenge
Human Robot Interaction
Multi-Robot systems