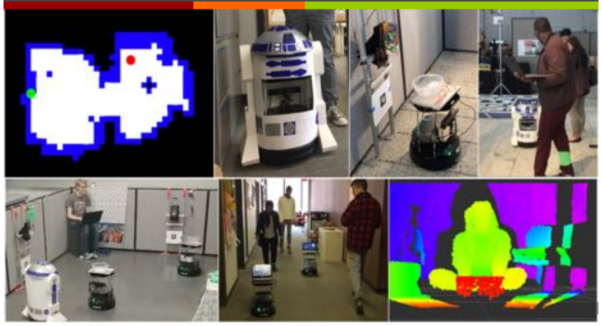
CS 189: Autonomous Robot Systems

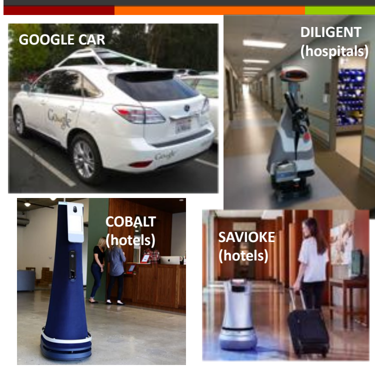
Spring 2020, Fridays 9-11:45am, Pierce 301



Agenda

- **Lecture: Robot Navigation -> MAPPING!**
- **TheConstructSim**:
  - Try out the ROS/Gazebo/Rviz simulators for Turtlebot3
  - More information on assignments will be posted to Piazza (ignore the schedule online)
- Upcoming:
  - Lecture next week: Ethics of Robotics and Automation
- 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?* — **Last-1 Lecture**
- Localization: *Where am I?* — **Last Lecture**
- Mapping: *Where have I been?*
- Exploration: *Where haven't I been?* — **Today!**

Mapping and Exploration

You are roaming around in an unknown space, what can you learn about it?

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 OR 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

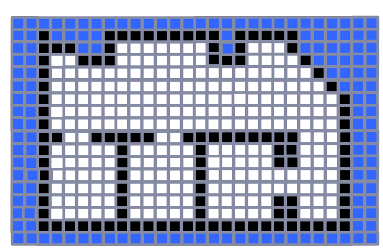
Today's topics

- 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 and sensor uncertainty together
    - Simultaneous Localization and Mapping (SLAM)
- Maybe? Pset 4: Your Autonomous OG Mapper!*
  * uses material from all 3 navigation lectures

What is an Occupancy Grid?

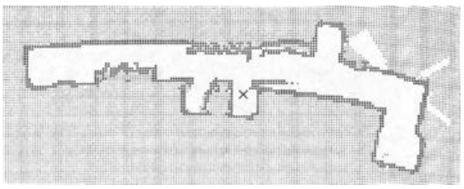
- A way of representing a map as a gridded world where each cell is either "occupied" or "empty" or "unknown".

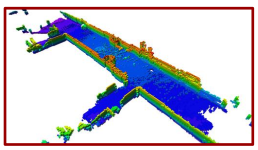
Your World

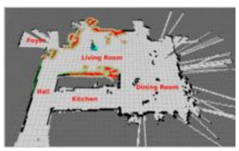


Grid generated by a Robot => boundary shape

Examples







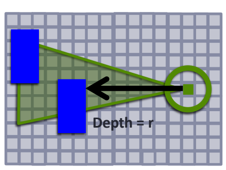
What is a Sensor Model?

- 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)

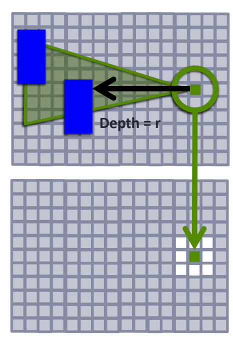


Constructing a Sensor Model

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)



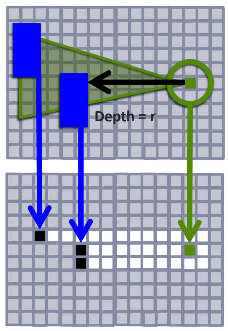
Constructing a Sensor Model

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)

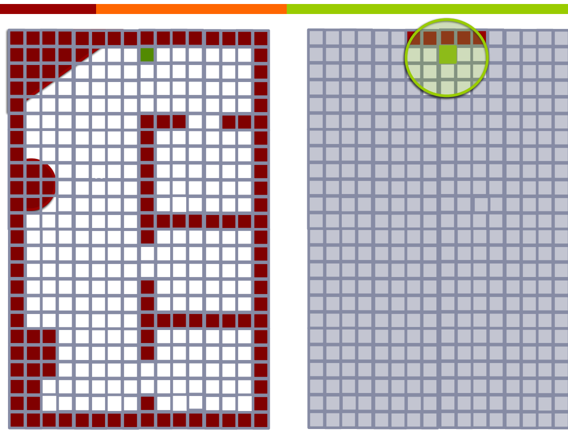
Simplest Sensor Model
Where I stand is Empty (white)

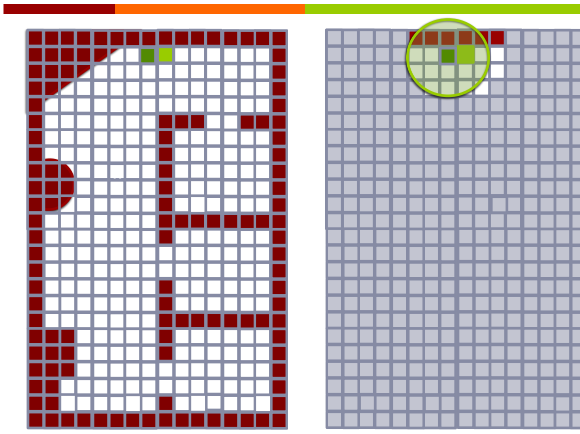
A Better Model
Set Region 1 cells as Empty (white)
Set Region 2 cells as Occupied (black).
Pick a max range/angle where data is reliable
Rest is still Unknown (gray)

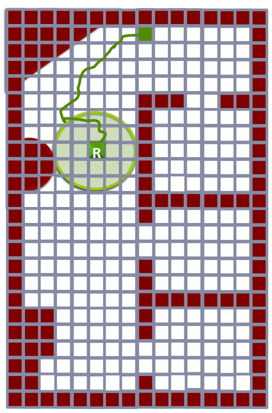


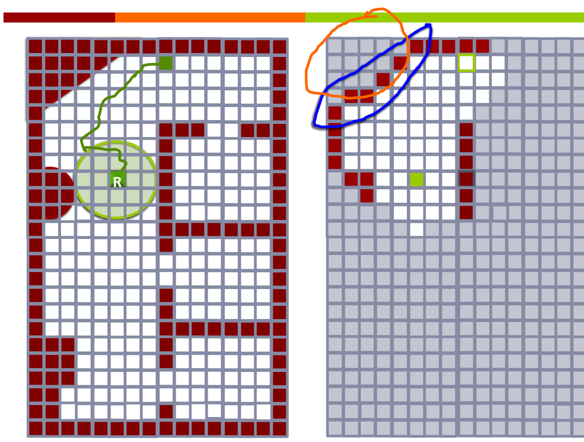
A Simple OG Mapping Algorithm

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")









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

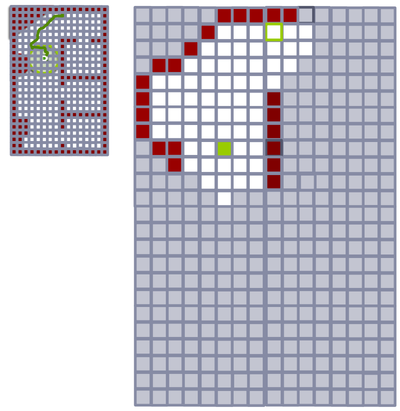
- Basic Concept in Math: Random Walks in bounded 2D
  - With Probability=1 you will *eventually* visit every spot
- Basic Concept in CS: Systematic Graph Coverage
  - You are given a "graph" with V nodes
    Write an algorithm that visits all of the nodes
    Breath-First Search and Depth-First Search; 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

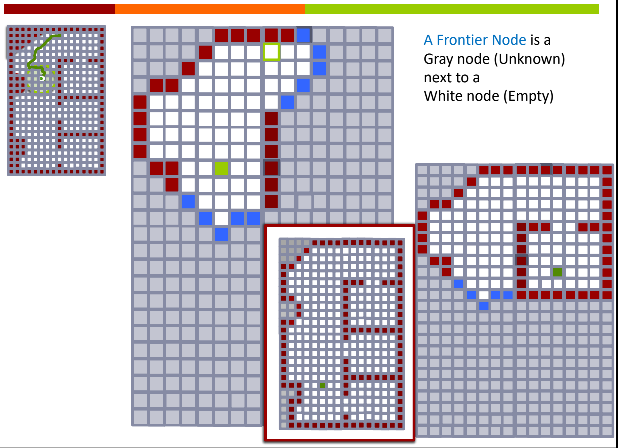
- **Frontier Based Exploration**
  - A common technique for building maps
  - Key Idea:
    - Identify the "frontiers" between known and unknown

      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)

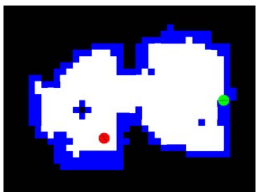
A Less Simple Mapping Algorithm

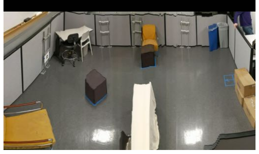
1. Initialize Grid
2. Update the Grid
   - Mark your current position as "empty"
   - Mark sensed nearby grid locations
     As "empty" or "occupied"
3. **Pick a Next Move**
   - Identify ***frontier cells***
   - Pick one (e.g. maybe the closest)
   - Plan a path* to the ***nbr empty cell***.
   - Go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

The smart sensor model and smart exploration strategy make for much faster mapping!

* We covered path planning two lectures ago

Turtlebots can do this!





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 and sensor uncertainty together
    - Simultaneous Localization and Mapping (SLAM)

Questions?

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 and sensor uncertainty together
    - Simultaneous Localization and Mapping (SLAM)

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

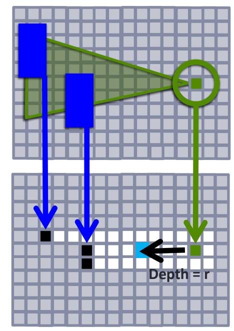
And taking measurements multiple times of the same location

Part1: 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



Part2: A Bayesian OG Map

For every grid location (i,j), store a probability value
P(Occupied) = Probability this grid location is Occupied
P(Empty) = 1 - P(Occupied)

1	0	0
1	-1	0
1	-1	-1

Before

1.0	0.1	0.1
0.8	0.5	0.2
0.8	0.5	0.5

After

Bayesian OG Mapping

For every grid location (i,j), store current probability value
P(Occupied|s_n) = Probability this grid location is Occupied ("n" timestep)

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

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

Bayes Rule
$$P(A|s) = \frac{P(s|A)\,P(A)}{P(s|A)P(A) + P(s|(1-A))\,P(1-A)}$$

Bayes Map 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 OG Mapping

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

Bayes Map Update Rule:

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

- Lets say I observe grid(5,6) for the first time,
  and lets say my sensor reading s="obstacle" (but its far away, i.e. less sure)
  - New Reading: P(s|Occupied) = 0.62, P(s|Empty)=0.38
  - Old Map Estimate P(Occupied)=P(Empty)=0.5
  - **P(Occupied|s="obs") = (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 (5,6) again, we have *prior* knowledge
  - We now think P(Occupied)=0.62 P(empty)=0.38
  - New sensor reading P(s="obstacle"|Occupied) = .80 (we are closer & surer)
  - **P(Occupied|s="obs") = (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)

Improvement 2: Probabilistic Mapping

- Overarching idea
  - Store *probabilities* of occupancy rather than 3 values.
  - Caveat: We treat each grid cell as independent even though its not.
- But how do you move in this probabilistic map?
  - You periodically must turn probability into Occupied/Empty!
  - Use some threshold to decide,
    e.g. P(occupied) > 0.8 and P(empty) < 0.2, rest is "unknown".
  - Then do frontier exploration and path planning as before 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,
   - 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!

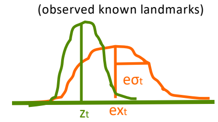
And you are just messing up your grid over time due to drift

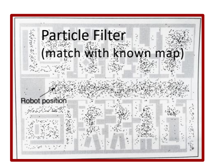
Recall: Probabilistic Localization…

- **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.

1 lecture ago:
Kalman Filters
Particle Filters

Kalman Filter
(observed known landmarks)





Probabilistic Localization and Mapping

- **Probablistic Localization**
  - **$P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}\ \text{map})$**
  - Where am I? Given that I took the noisy actions U and noisy observations Z of things in my perfect map.
- **Probablistic Mapping**
  - **$P(\text{map} \mid Z_{0\text{-}t}, U_{0\text{-}t})$**
  - 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**
  - **$P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}\ \text{map})$**
  - Where am I? Given that I took the noisy actions U and noisy observations Z of things in my perfect map.
- **Probablistic Mapping**
  - **$P(\text{map} \mid Z_{0\text{-}t}, U_{0\text{-}t})$**
  - What is my **map** like? Given that I made noisy observations Z as I walked along my perfect path dictated by U.

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

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: $P(x_t \mid Z_{0\text{-}t}\ U_{0\text{-}t}\ \text{map})$**
  - **Probablistic Mapping: $P(\text{map} \mid Z_{0\text{-}t}\ U_{0\text{-}t})$**
- **Probablistic SLAM ("Simultaneous")**
  - **$P(x_t, \text{map} \mid Z_{0\text{-}t}\ U_{0\text{-}t})$**
  - 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/OG)

Extended Kalman Filter SLAM

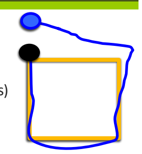
- In original EKF,
  - State == robot position, represented as a Gaussian (x_t σ_t)
- In EKF-SLAM,
  - State = [robot and all landmark] positions as Gaussians
  - **Position X_t = {x_t, m1, m2, m3 … mn}** (number of landmarks grows!)
  - **Co-variance σ_t = (n+1)x(n+1) matrix** (uncertainty is correlated!)
  - 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: Four Steps (Repeat)
  1. Motion Step: Update $P(x_t, \text{map} \mid Z_{0\text{-}(t\text{-}1)}\ U_{0\text{-}t})$ based on action U_t
  2. Observation Step: Update $P(x_t, \text{map} \mid Z_{0\text{-}t}\ U_{0\text{-}t})$ based on Z_t
  3. Combine into Single Estimate

     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
     Then update the whole map (m1-m4), thanks to co-variance matrix
  4. 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 machines to complex representation and memory

PathPlanning	Localization	Mapping/Exploration
Visual Homing Forward Kinematics (direct methods) Bug Algorithms (obstacles) A* Algorithm (maps)	Dead-Reckoning (using internal motion) Landmarks (using external sensing)) Kalman Filter Particle Filters (combine uncertain motion and sensing)	Occupancy Grid Mapping Sensor models Frontier Exploration (faster mapping) Bayesian Mapping SLAM (uncertainty in maps and location)

Conclude: Robots Navigating the World

Second Part of CS189: High-level reasoning

From finite state machines to complex representation and memory

PathPlanning	Localization	Mapping/Exploration
Visual Homing Forward Kinematics (direct methods) Bug Algorithms (obstacles) A* Algorithm (maps)	Dead-Reckoning (using internal motion) Landmarks (using external sensing)) Kalman Filter (ROS pkg) Particle Filters (combine uncertain motion and sensing)	Occupancy Grid Mapping Sensor models Frontier Exploration (faster mapping) Bayesian Mapping SLAM (ROSpkg: Gmapping) (uncertainty in maps and location)