CSE 461

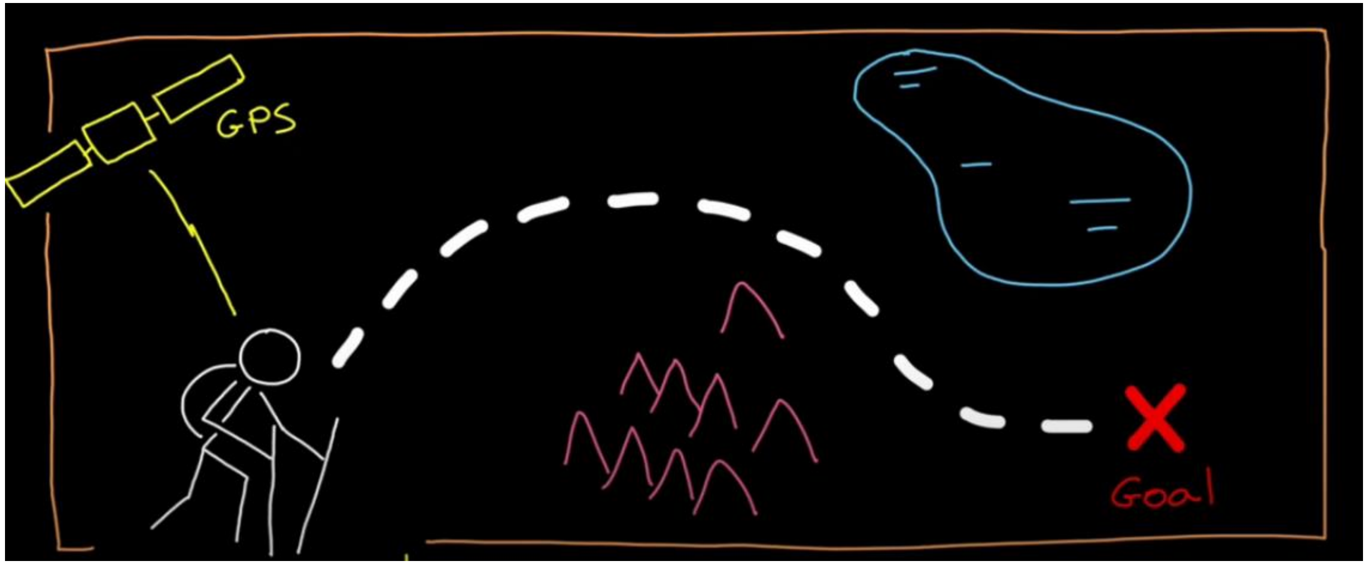
Introduction to Robotics

Navigation : Path planning, Localization, Mapping ,Exploration

Abdulla Hil Kafi
Research Associate,
School of Engineering,
Brac University

Navigation

- The act, activity, or process of finding the way to get to a place
- Navigation is the ability to determine your location and plan a path to some goal



Robot Navigation

For autonomous behavior, robots need the ability to navigate:

- Learn the environment->“Model”
- Estimate where it is in the environment->“Localize”
- Move to desired locations in the environment

Navigation Example

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

Navigation Problem

Problem Characteristics

- Environments are Known versus Unknown
- Environments are Static versus Dynamic
- Environments are Structured versus Unstructured (Indoors versus Outdoors?)

Robots Navigating

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

What is Path Planning?

- Simple Question: How do I get to my Goal?
- Not a simple answer!
  - Can you see your goal?
    - Do you have a map?
    - Are obstacles unknown or dynamic?
  - Does it matter how fast you get there?
  - Does it matter how smooth the path is ?
  - How much compute power do you have?
  - How precise is your motion control?
- Path Planning is best thought of as a Collection of Algorithms
- 3 Things need to consider: Environment, Success metrics, Robot capability.

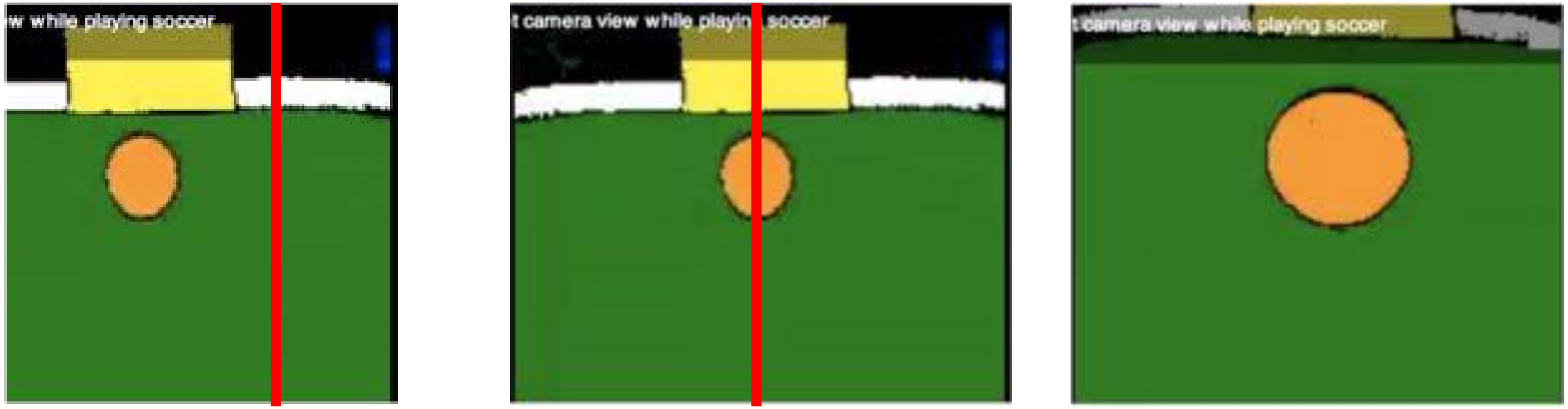
Types of Path Planning Approaches

- Basics
  - Visual homing (Purely local sensing and feedback control)
  - Inverse Kinematics (Turn-move-turn to get from A to B)
- Bug-based Path Planning (mostly-local without a map)
  - Robots can see the Goal (direction and distance)
  - But there are unknown obstacles in the way (No map)
- Metric (A*) Path Planning (global with a map)
  - Assumes that you have a map (distance or graph)and you know where you and the goal are located in it.
  - Path is represented as a of series of waypoints(directions)

Basics: Visual Homing

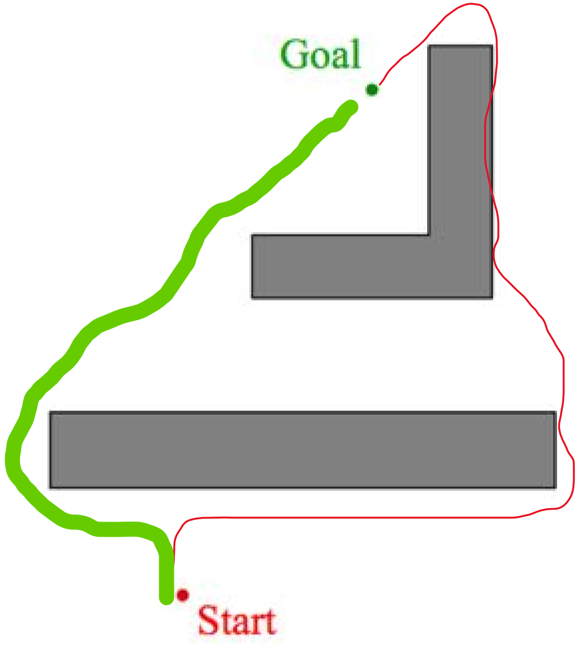
Purely Reactive Navigation

- Measure Visual (x,y) Position of Goal
- Move to bring goal to Visual Center
- Proportional Control (if you see the goal), Random walk (if you don't)



Bug-based Path Planning

- What if the Robot has obstacles in the way?
  - Always have Goal direction and/or distance (Global)
  - But No Map: Only local knowledge of environment (Local)
  - *Example Scenario:*
    Outdoor robot knows GPS location of goal, but building in the way.
    Indoor robot see goal location, but furniture in the way.
  - "Bug" Algorithms depend on simple but provable behaviors!
    - Don't need to build a map
    - *Simple Computation: Visual Homing + Wall-following + Odometry*



- Very intuitive class of algorithms – but surprisingly powerful

Metric/Global Path Planning

- What if the Robot has Full Knowledge
  - A map of the environment and robot + goal's locations
  - Goal: Find a "optimal" path (typically distance)

- Two Components
  - Map Representation ("graph"):
    - Feature based maps (office numbers, landmarks)
    - Grid based maps (cartesian, quadtrees)
    - Polygonal maps (geometric decompositions)
  - Path Finding Algorithms:
    - Shortest-Path Graph Algorithms (Breadth-First-Search, A* Algorithm)

Map Representation: Feature based

- Also known as a Topological or Landmark-based Map
  - Features your robot can recognize:
    - Includes **both natural** landmarks (corner, doorway, hallways) and **artificial** ones (office door numbers; or robot-friendly tags)
  - Gateways are landmarks that represent decisions (e.g. intersection)
  - Distinguishable places are unique landmarks
- World is a graph that connects landmarks
  - Edges represent **actual motion:** how to get from landmark A to landmark B
  Usually visual/reactive navigation is possible along an edge
  - Edges can also keep **extra attributes**: distance, time it takes, etc.
- Google Maps are topological maps for humans (e.g. turn at intersection)

Path Finding Algorithms

- All Map Representations are a weighted “graph”
  - Nice part is that you only need to do this once
- Algorithm: Compute shortest paths in the graph
  - Path is represented by a series of waypoints
  - Single Path Search Algorithms: Find shortest path A to B
    - Breadth-First-Search (simple graphs);
    - A* search for large graphs (BFS + Heuristic)
  - Gradient Path Algorithms: Find *all paths* towards B
    - E.g. Fixed Base station: BFS, Dijkstra’s, Wavefront algorithms, etc.

Robots Navigating

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

Localization

- **Simple Question:** *Where am I?*
- **Not a simple answer:**
  - Do you have a map?

    Yes => a global position in the world

    No => position in reference to other objects? Or your own past?
  - What can you sense?

    Can you sense and record your own self-movement?

    Can you sense external things like landmarks?

    How certain are you about what you sense?
- **Localization is a "collection of algorithms"**

Localization Techniques

- **Dead-reckoning (motion)**
  - Keep track of where you are without a map, by recording the series of actions that you made, using internal sensors. (also called Odometry, Path Integration)
- **Landmarks (sensing)**
  - Triangulate your position geometrically, by measuring distance to one or more known landmarks
    E.g. Visual beacons or features, Radio/Cell towers and signal strength, GPS!
- **State Estimation (uncertainty in motion & sensing)**
  - *Probabilistic Reasoning*
    - **Kalman Filters** (combine both motion and sensing)
    - Particle Filters (also known as Monte Carlo Localization)

Dead-Reckoning

- Keep track of initial position and the series of movements/actions that you made.
- Method: Take a “step”, compute new position.
- Also called odometry or path integration

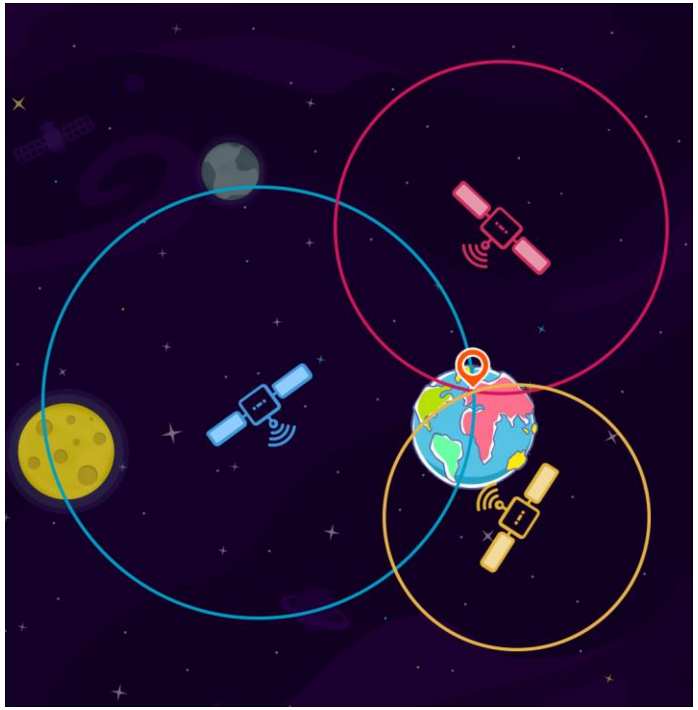
Example: **Inertial navigation systems** (INS)

- Complex motion (momentum, external effects)
- Include **accelerometers** and **gyroscopes** to provide better measurements of instantaneous velocity.
- Expensive systems very good
  - satellites, submarines
- But, low-cost IMU’s increasingly available

Land Mark

- **How it works**
  - *Opposite of dead-reckoning!*
  - Use measurements to external landmarks of known position
  - Examples: visual landmarks, radio towers, GPS!

- Example
  - GPS Satellites are your “landmarks”
  - Continually transmits a message
  - Message includes both time of transmission, and satellite position
  - GPS Receiver
  - Compute distance by measuring signal transmission time (speed of light)
  - 3D: Lie on the intersection of 4 spheres!



State Estimation (uncertainty in motion & sensing)

- **Key Idea: Combine Motion and Sensing**
  - **(Dead-reckoning + uncertainty)** + **(Landmarks + uncertainty)**
  - Each has error, but the error can be complementary
- **Kalman Filters**
  - Take advantage of mathematics of Gaussians to model uncertainty
  - General method for state estimation (not just localization)
  - Applications: Car + GPS, Lawnmower + beacons, warehouse robots
- **Particle Filters (Monte Carlo Localization)**
  - Use a discrete distribution of “Particles” to represent uncertainty think of sampling or histograms)
  - Useful when environment is complex and ambiguous
  - Application: A robot wandering in a building with a map

Robots Navigating

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

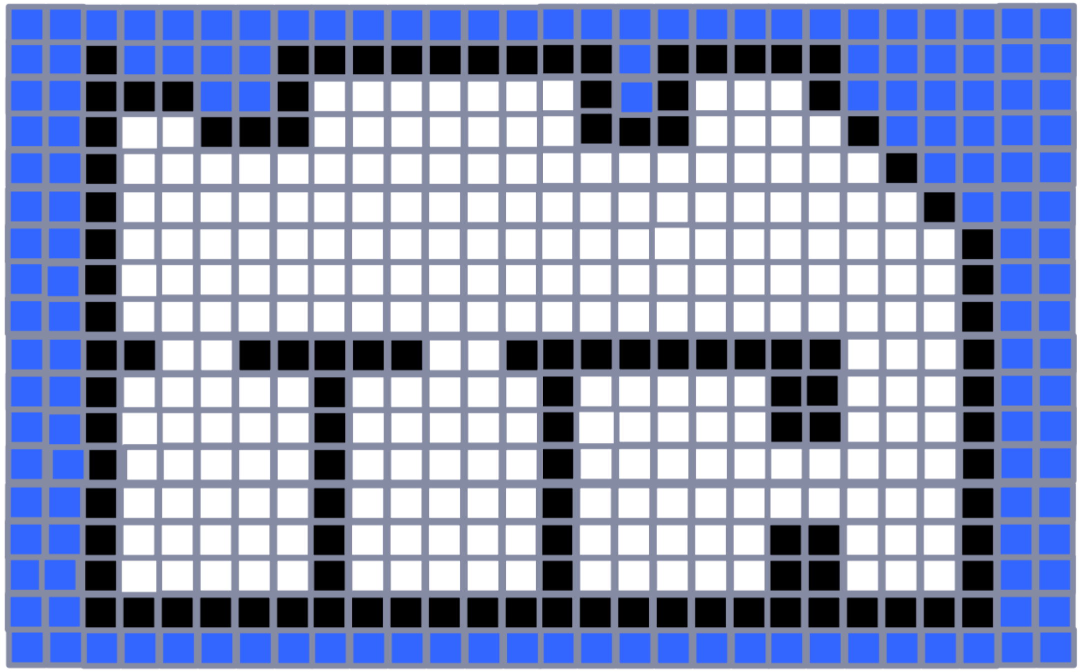
Mapping and Exploration

- **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 to 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"
  - We will focus on "Occupancy Grid" algorithms

What is an Occupancy Grid?

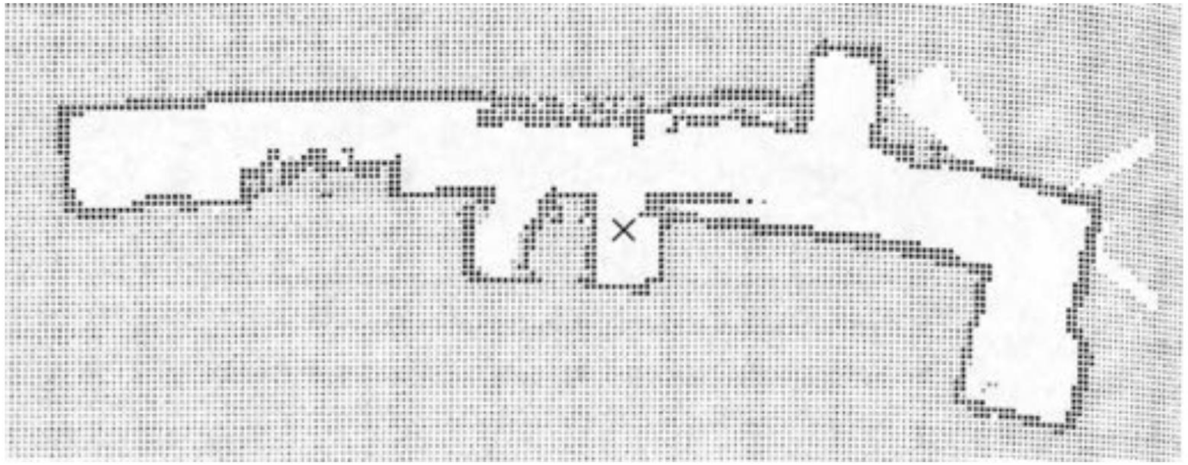
- A way of representing a map as a gridded world where each cell is either "occupied" or "empty" or "unknown".

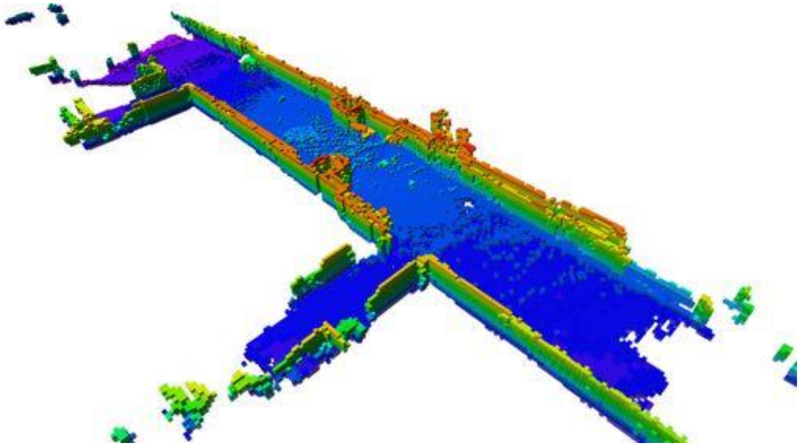
Your World

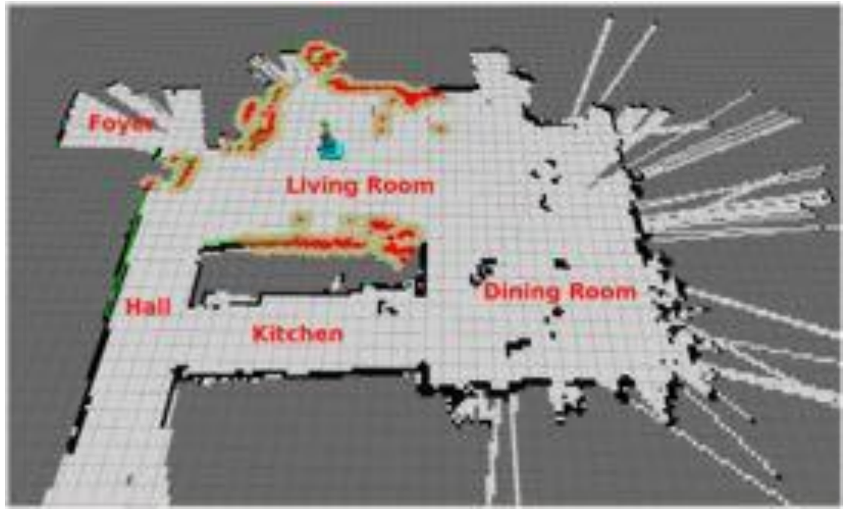


Grid generated by a Robot => boundary shape

Examples







What is a Sensor Model?

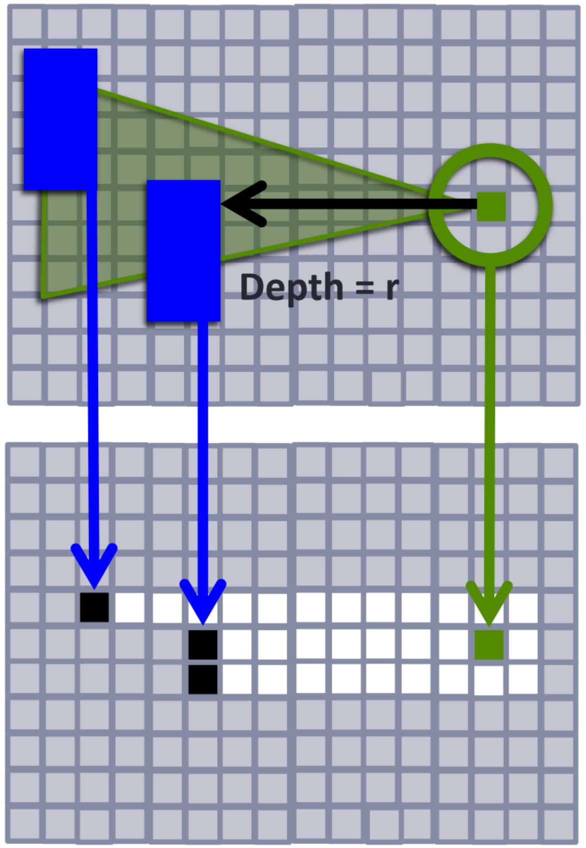
- Constructing a Sensor Model
  - A sensor measures *raw values* in an environment
  - You have to map that into a Grid Cell Value.
  - Robots can have very different sensors and configurations
  - Examples:
    LIDAR/Depth Camera
    Vs. a 360 degree vision/ranging system

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)

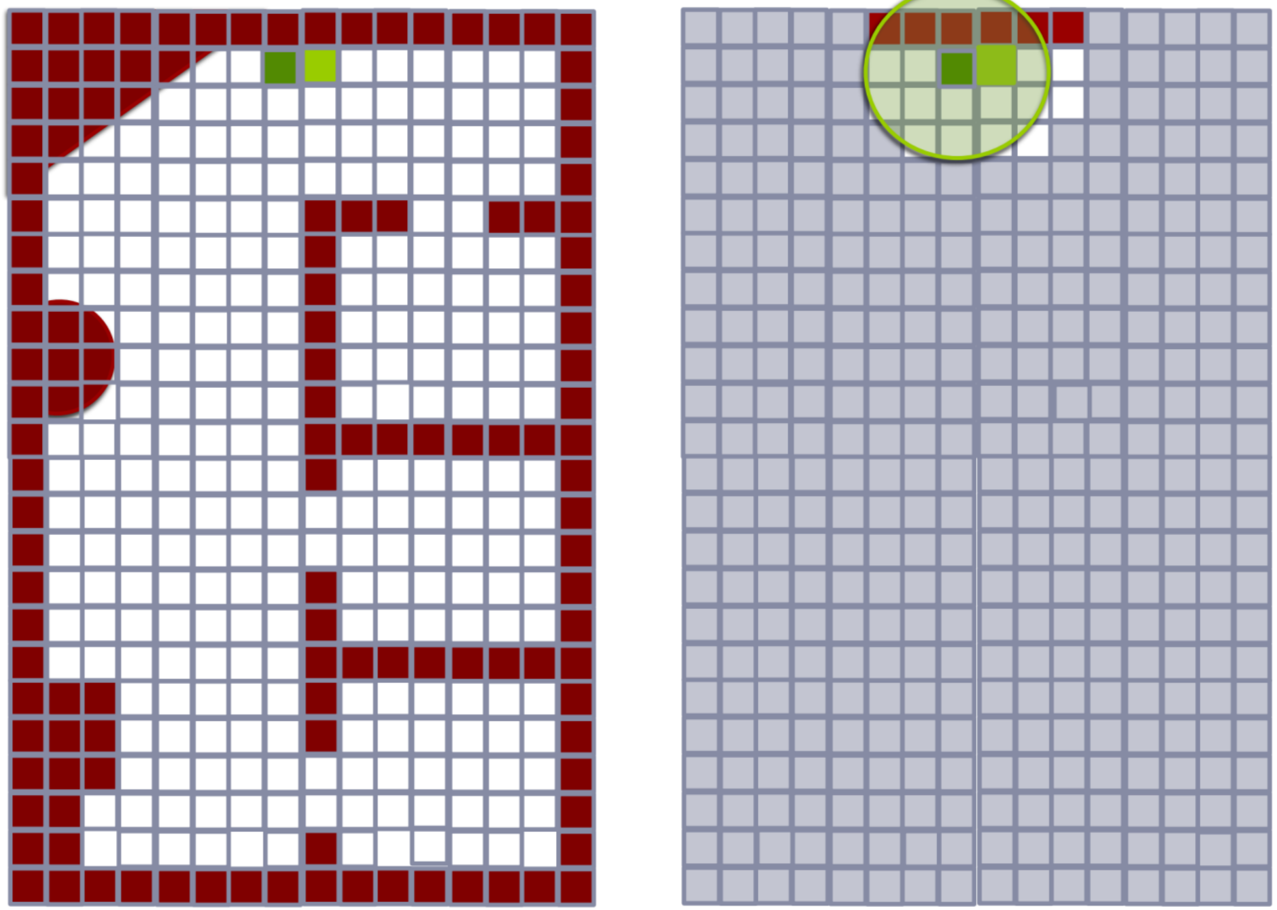
- Simplest Sensor Model
  Where I stand is Empty (white)

- 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”)

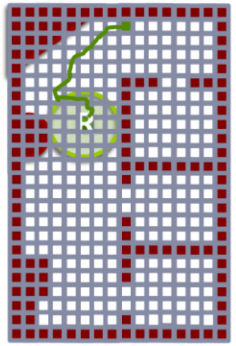


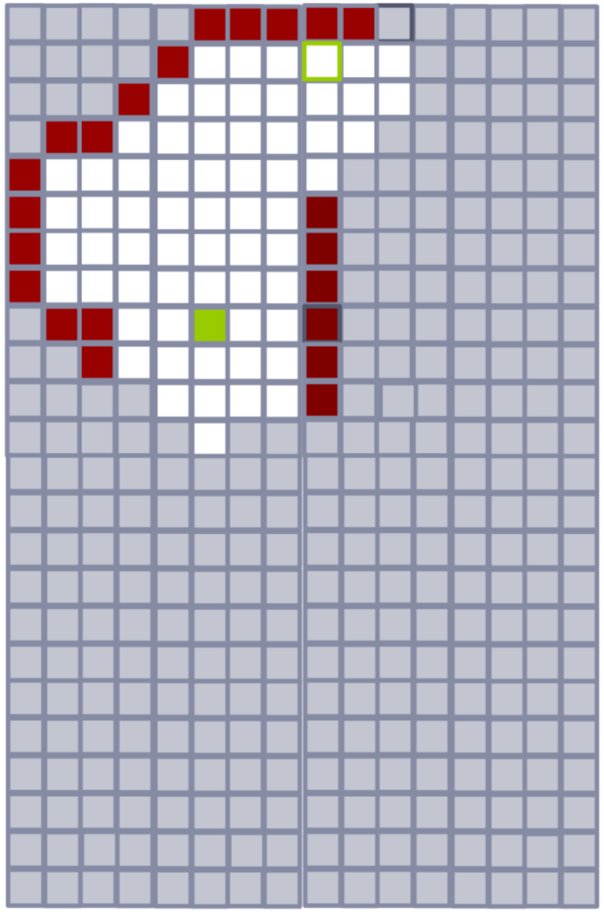
Exploration

- Basic Concept in Robotics: Navigating a GRID Graph is different
  - DFS works, but will still make a robot retrace steps
  - **Better choice: Frontier Based Exploration**

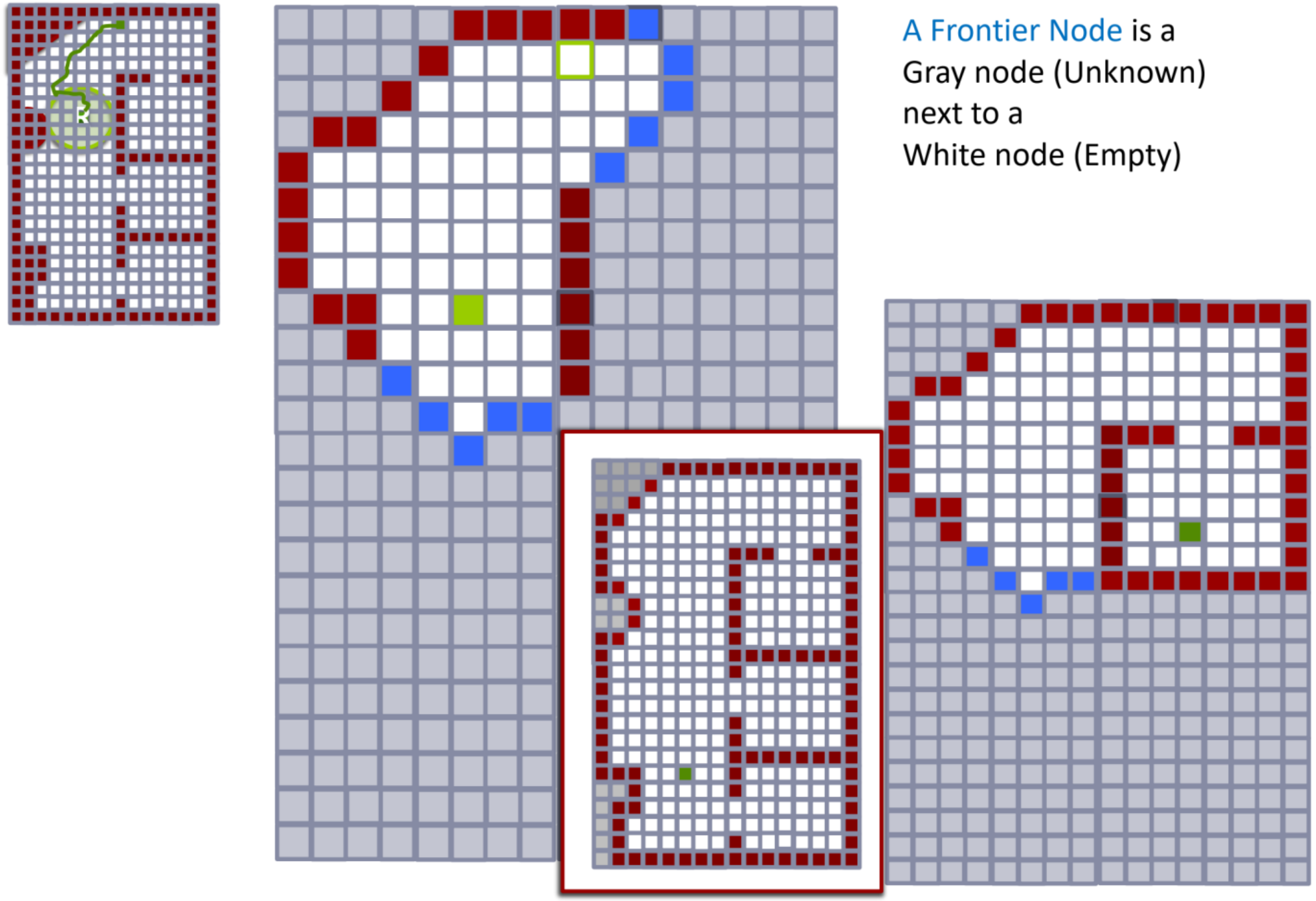
Exploration in Grid Worlds

- **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
    - 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.



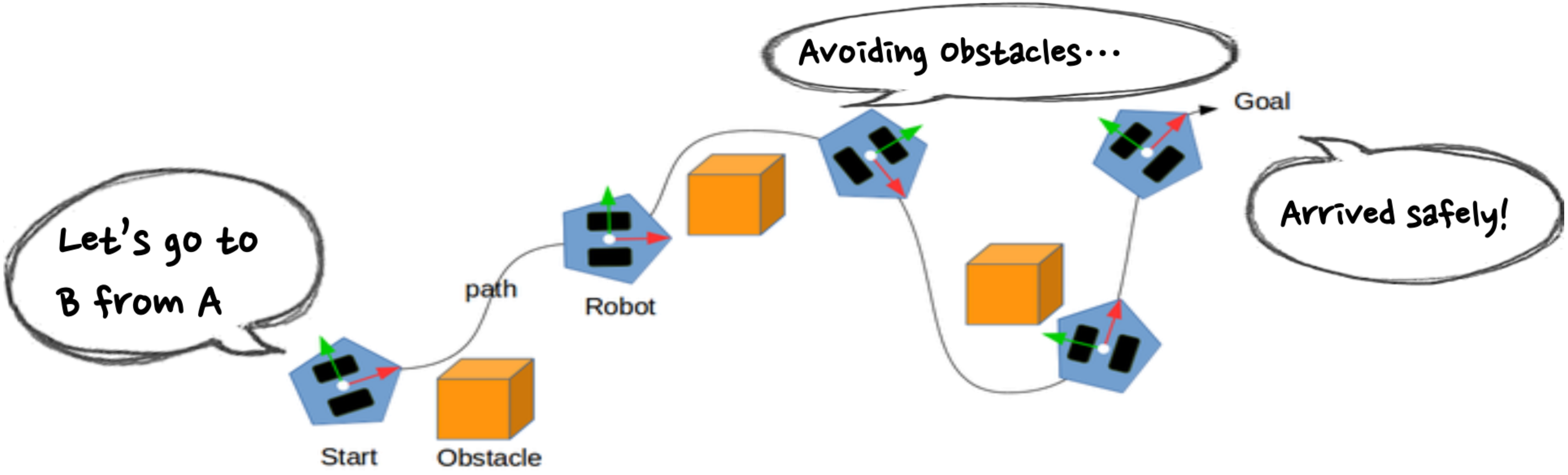


A Frontier Node is a
Gray node (Unknown)
next to a
White node (Empty)



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

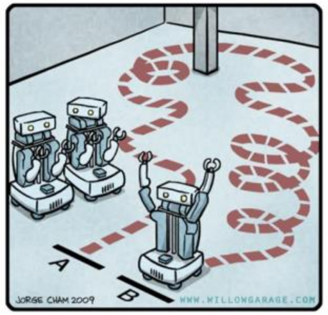
Summary



① **Position**: Measuring/estimating the robot's position
② **Sensing**: Measuring obstacles such as walls and objects
③ **Map**: Maps with road and obstacle information
④ **Path**: Calculate optimal path to the destination and follow the path

Summary

① **Position** ② **Sensing** ③ **Map** ④ **Path**



Position+Sensing → **Map**

SLAM

Simultaneous Localization And Mapping

Position+Sensing+Map → **Path**

Navigation