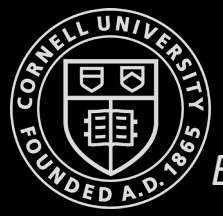
ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots



Storage (SRAM)

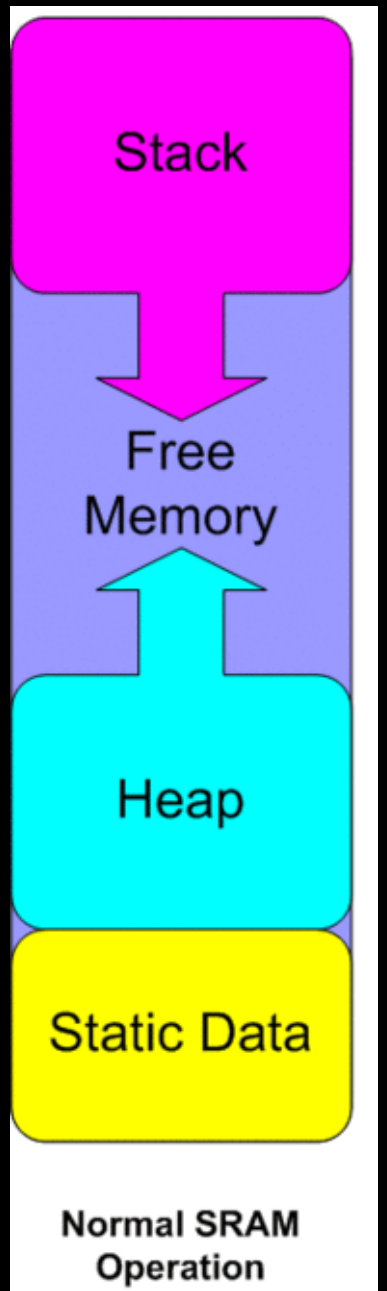
- Artemis Nano
  - 384kB RAM

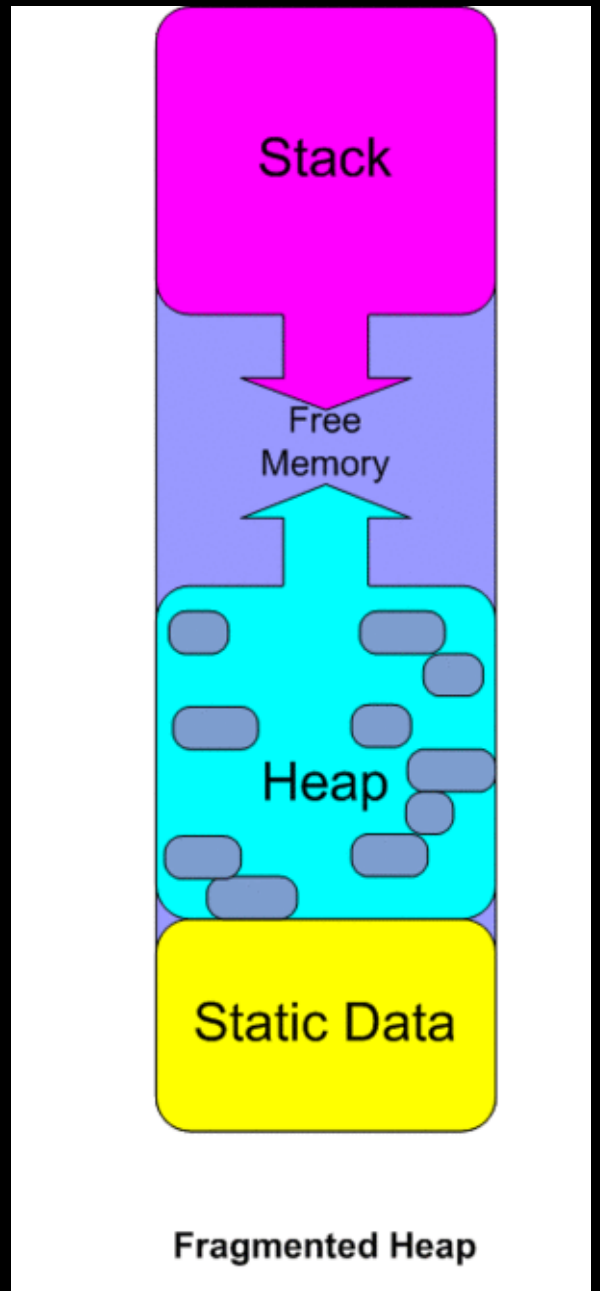
Function calls
- reclaimable!
- recursive fcts

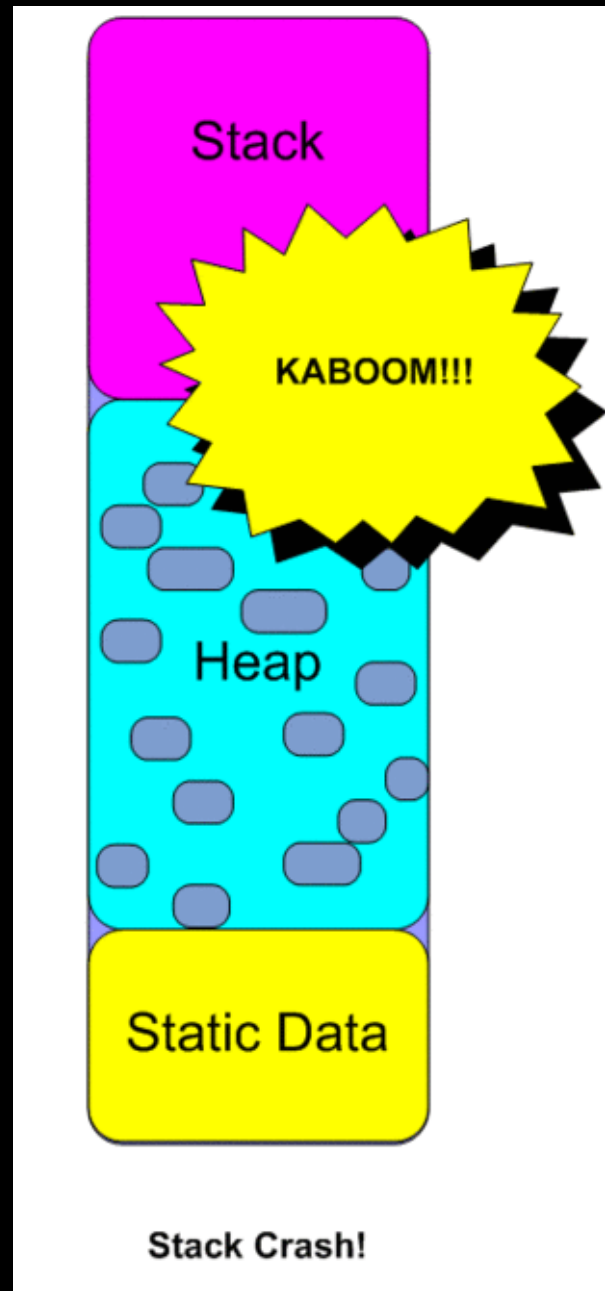
Local variables
- Reclaimable
- Use when possible

Global/static variables

Dynamically allocated objects
- Malloc

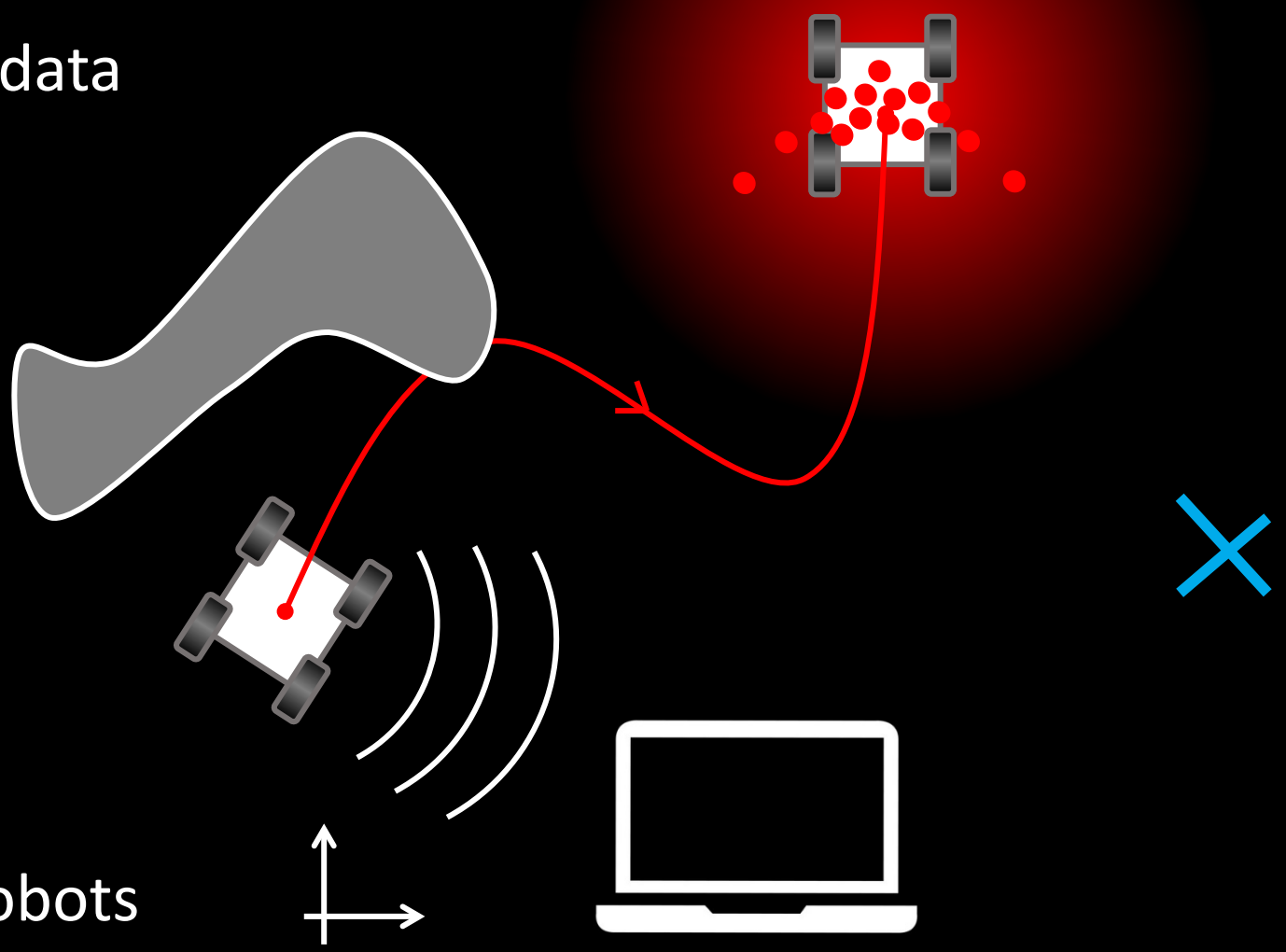






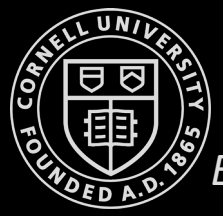
What we covered in class so far…

- Transformation matrices
- Bluetooth communication and data types
- Distance Sensors
- Odometry and IMU
- Actuators
- Controllers
  - PID control
  - LQR
- Observers
- Deterministic → Probabilistic robots
- Localization and Navigation



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Navigation

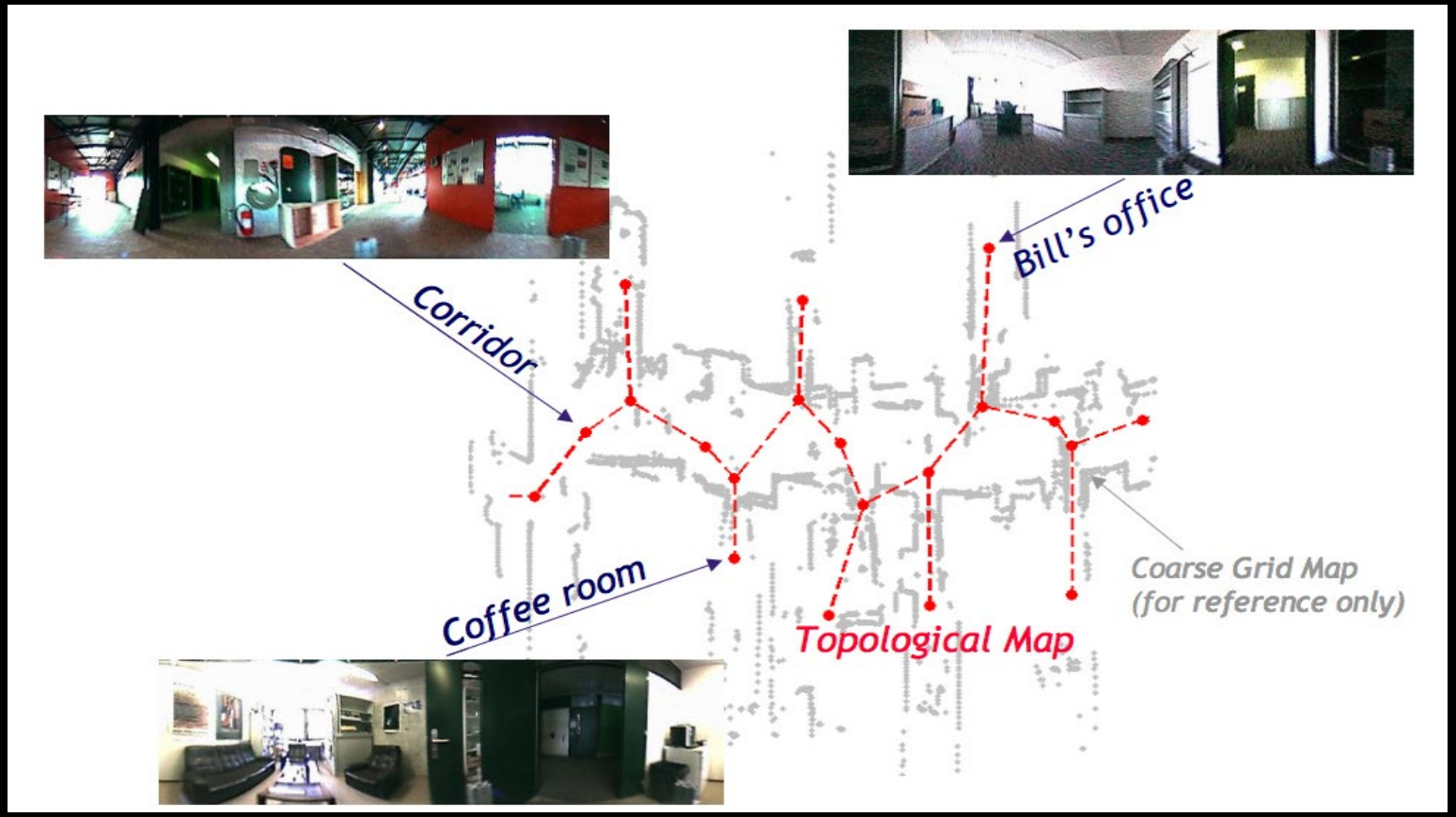


Navigation and Path Planning

- How do you get to your goal?
- No simple answers…
  - 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 computing power do you have?
  - How precise and accurate is your motion control?

Navigation and Path Planning

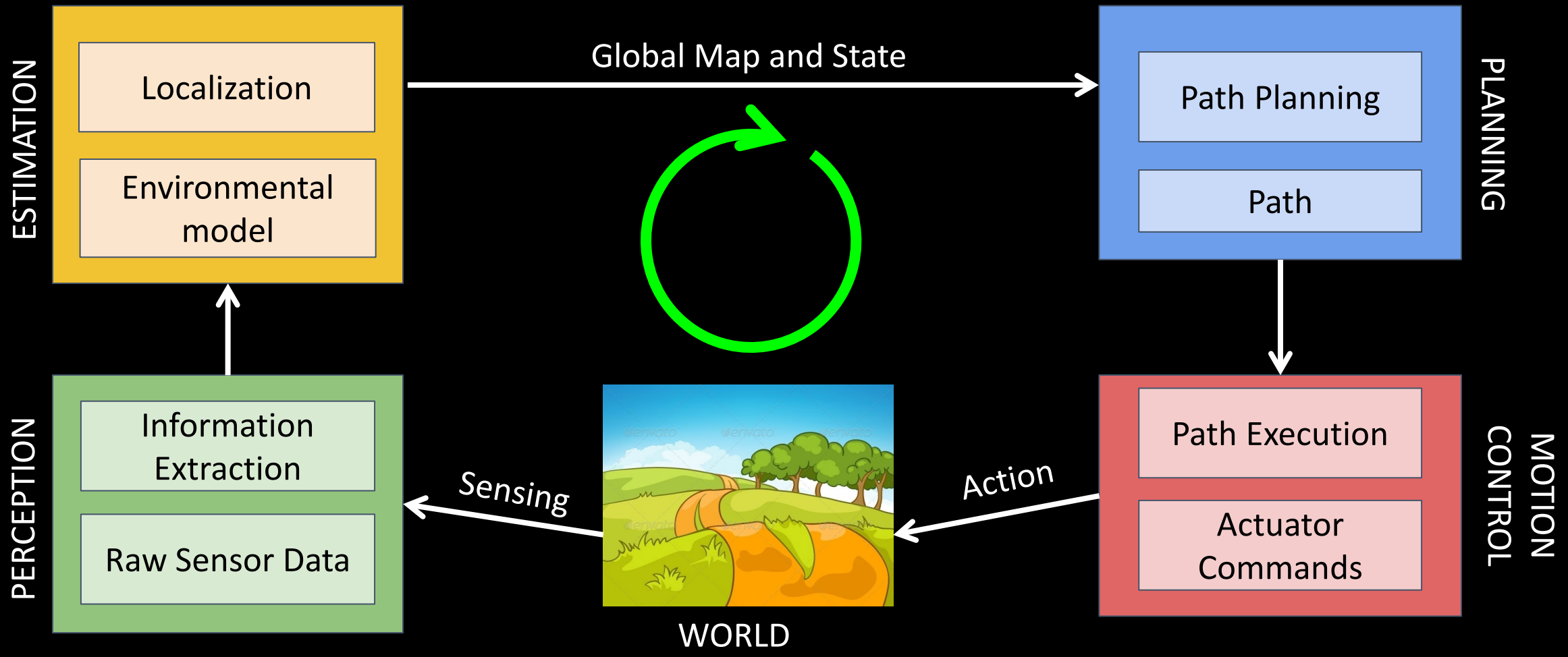
- **Problem:** Find the path in the workspace from an initial location to a goal location, while avoiding collisions
- **Assumption:** There exists a good map of the environment for navigation



- Global navigation
  - Given a map and a goal location, find and execute a trajectory that brings the robot to the goal
  - (Long term plan)
- Local navigation
  - Given real-time sensor readings, modulate the robot trajectory to avoid collisions
  - (Short term plan)

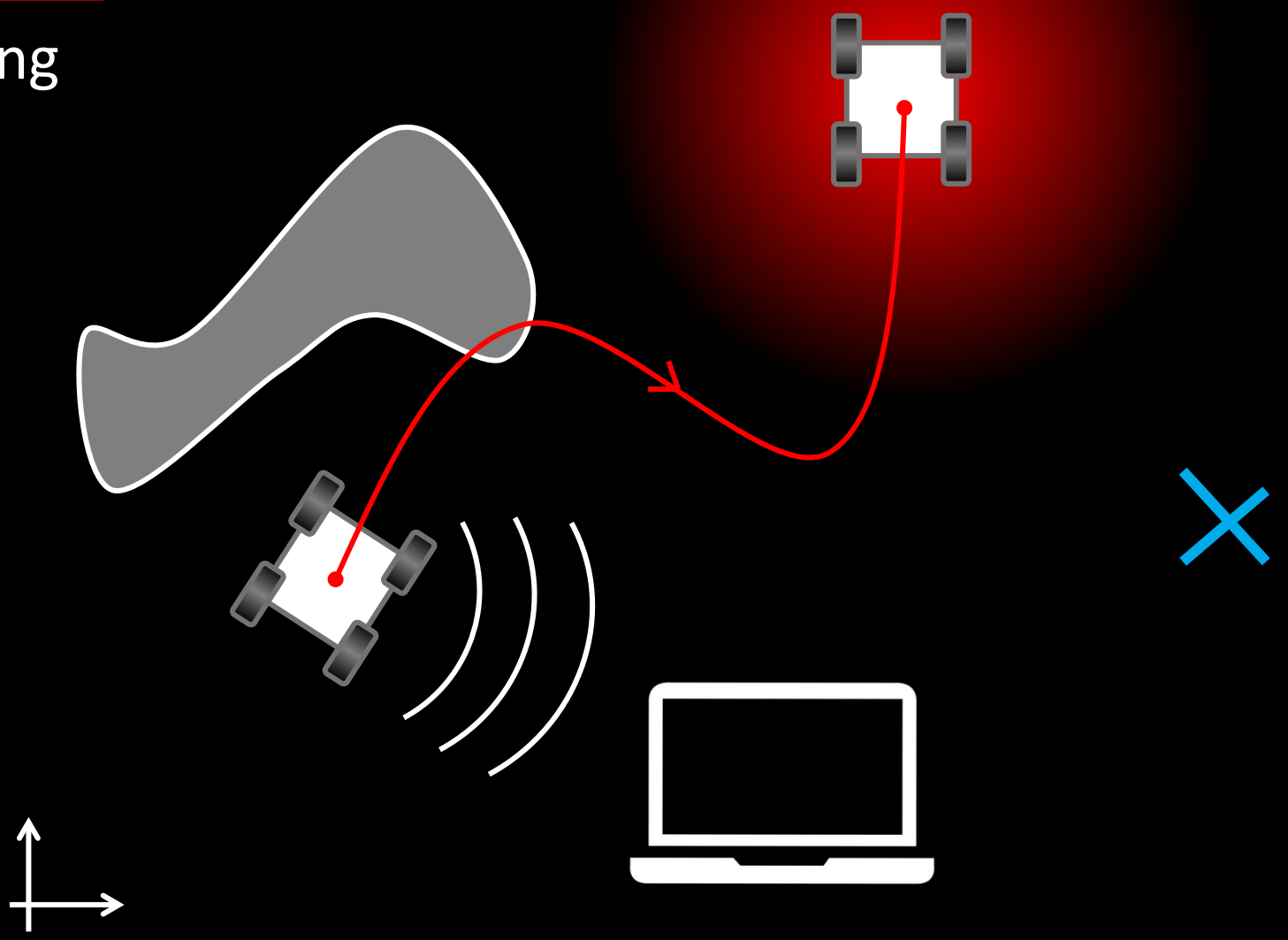
Navigation and Path Planning

- Navigation breaks down to: Localization, Map Building, Path Planning



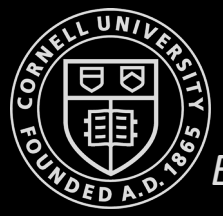
Outline of the next module on Navigation

- Local planners
- Global localization and planning
  - Map representations
    - Continuous
    - Discrete
    - Topological
  - Maps as graphs
  - Graph Search Algorithms
    - Breadth First Search
    - Depth First Search
    - Dijkstras
    - A*



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Local Planners



Local Path Planning / Obstacle Avoidance

- Utilize goal position, recent sensor readings, and relative position of robot to goal
  - Implemented as a separate task most of the times
  - Runs at a much faster rate than the global planning
- 3 examples
  - BUG Algorithms
  - Vector Field Histogram (VFH)
  - Dynamic Window Approach (DWA)

Wagner, ITS 2015

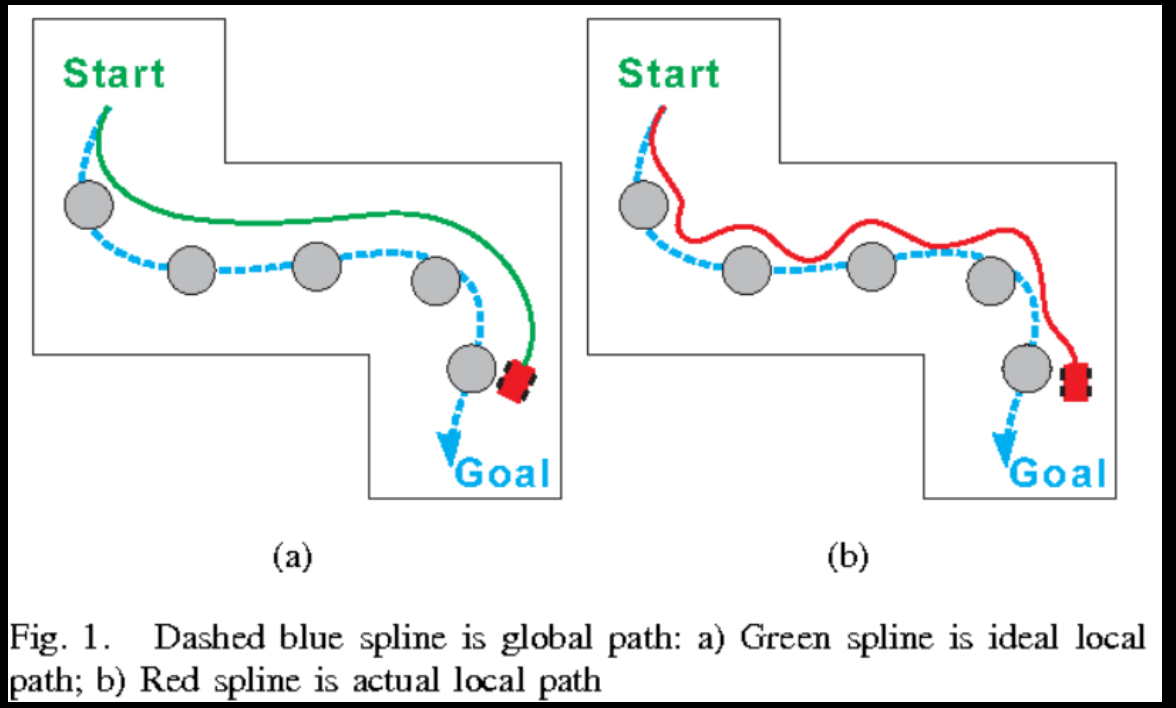
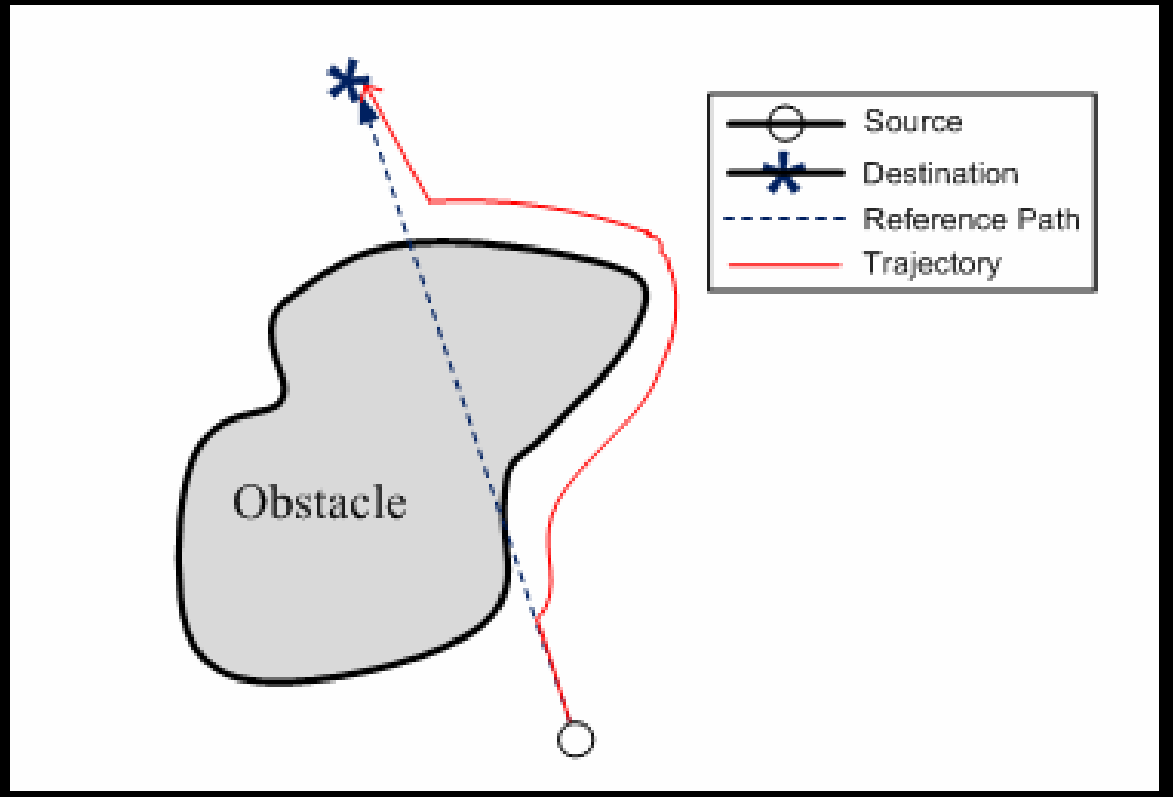


Fig. 1. Dashed blue spline is global path: a) Green spline is ideal local path; b) Red spline is actual local path

Bug Algorithms

- Uses local knowledge, and the direction and distance to the goal
- Basic idea
  - Follow the contour of obstacles until you see the goal
  - State 1: Seek goal
  - State 2: follow wall
- Different variants: Bug0, Bug1, Bug2
- Advantages
  - Super simple
  - No global map
  - Completeness
- Disadvantages
  - Suboptimal



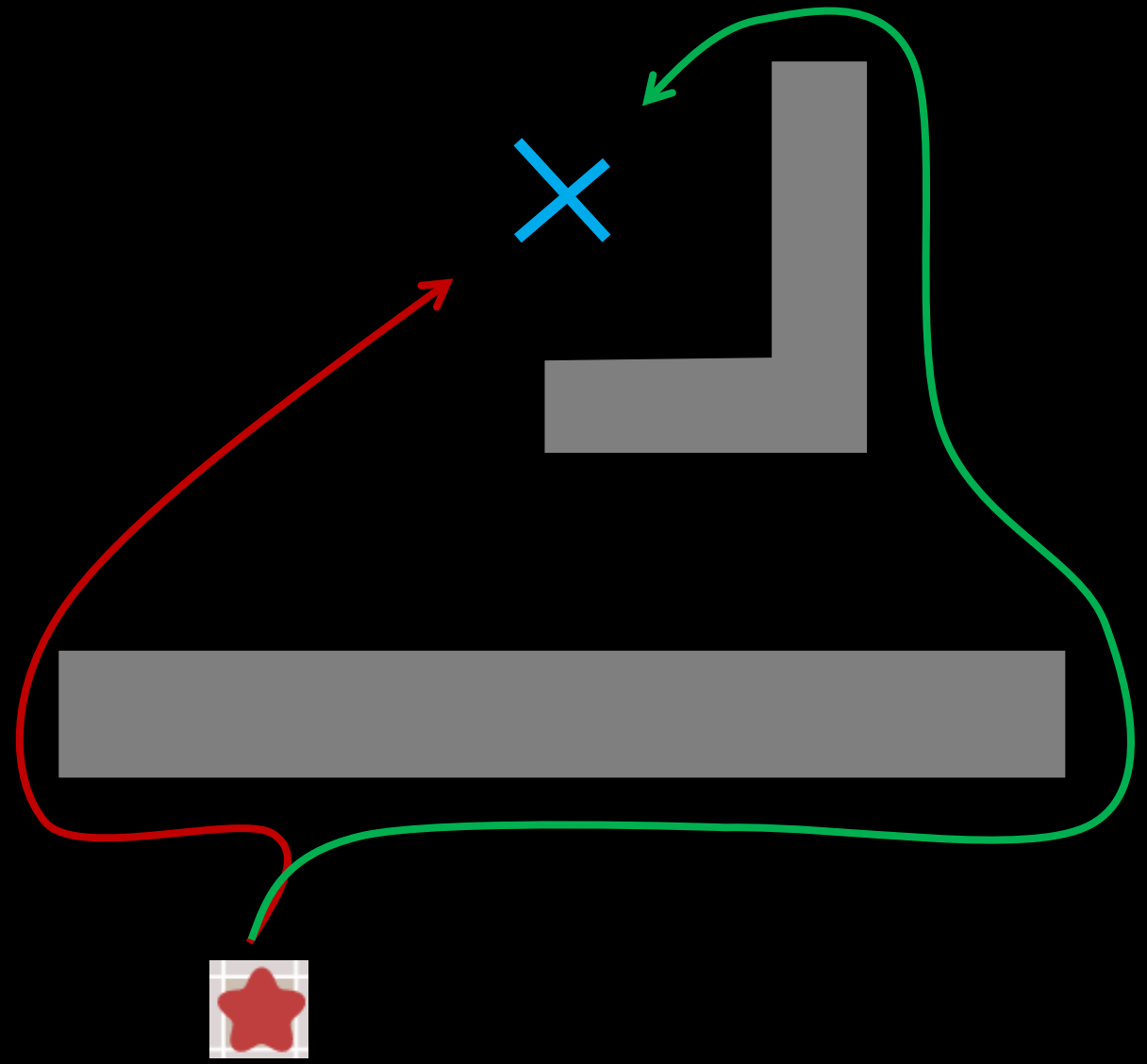
Bug 0

Sensor Assumptions

- Direction to the goal
- Detect walls

Algorithm

1. Go towards goal
2. Follow obstacles until you can go towards goal again
3. Loop



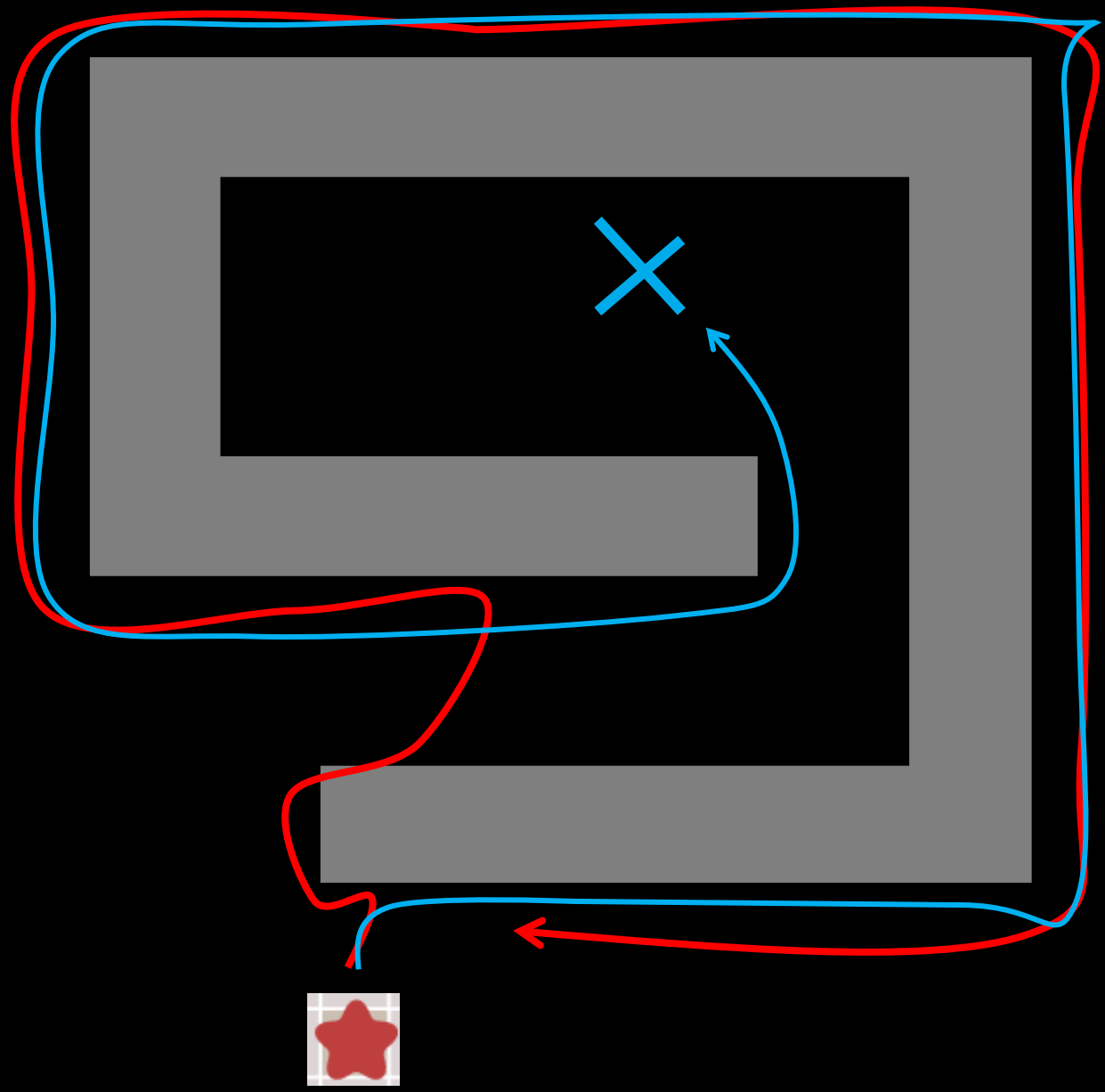
Bug 0

Sensor Assumptions

- Direction to the goal
- Detect walls

Algorithm

1. Go towards goal
2. Follow obstacles until you can go towards goal again
3. Loop



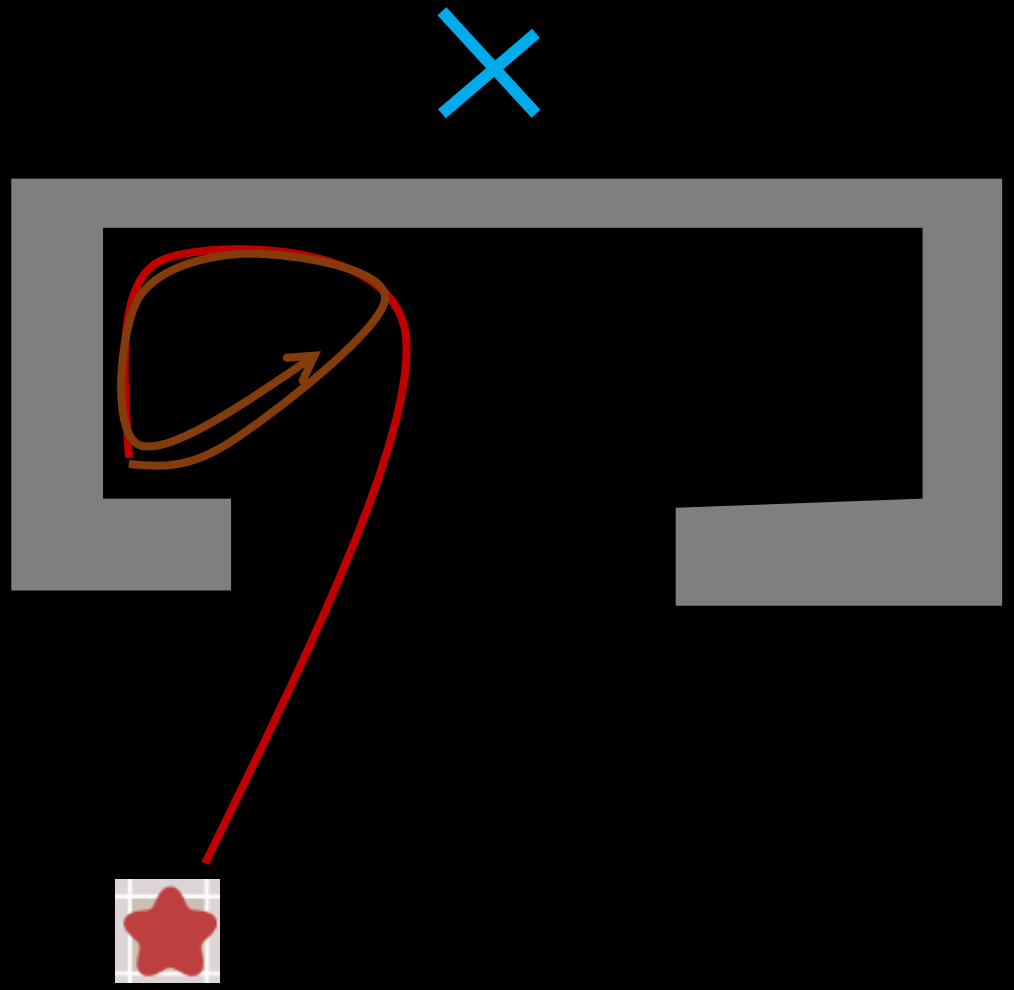
Bug 0

Sensor Assumptions

- Direction to the goal
- Detect walls

Algorithm

1. Go towards goal
2. Follow obstacles until you can go towards goal again
3. Loop



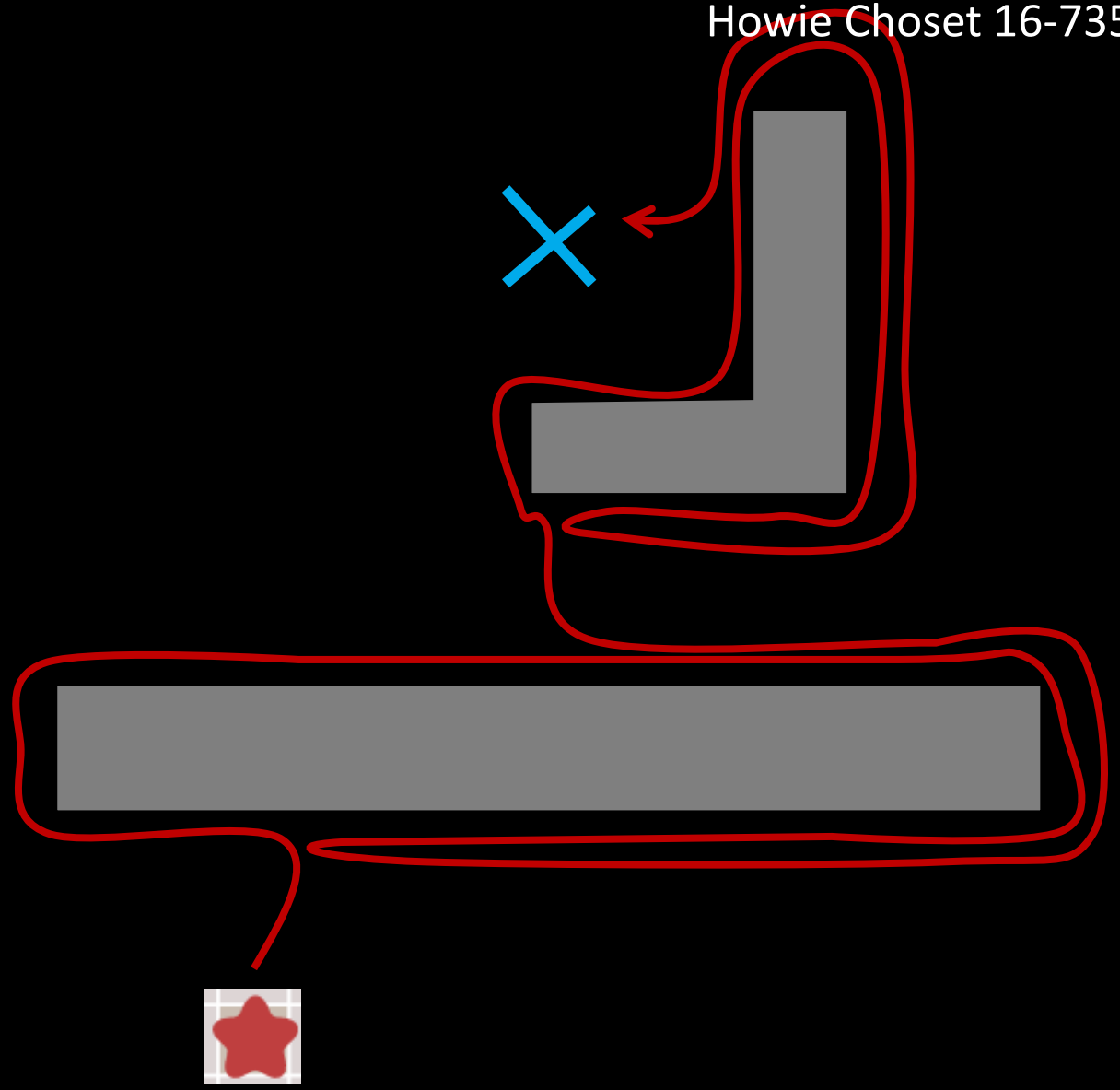
Bug 1

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry

Algorithm

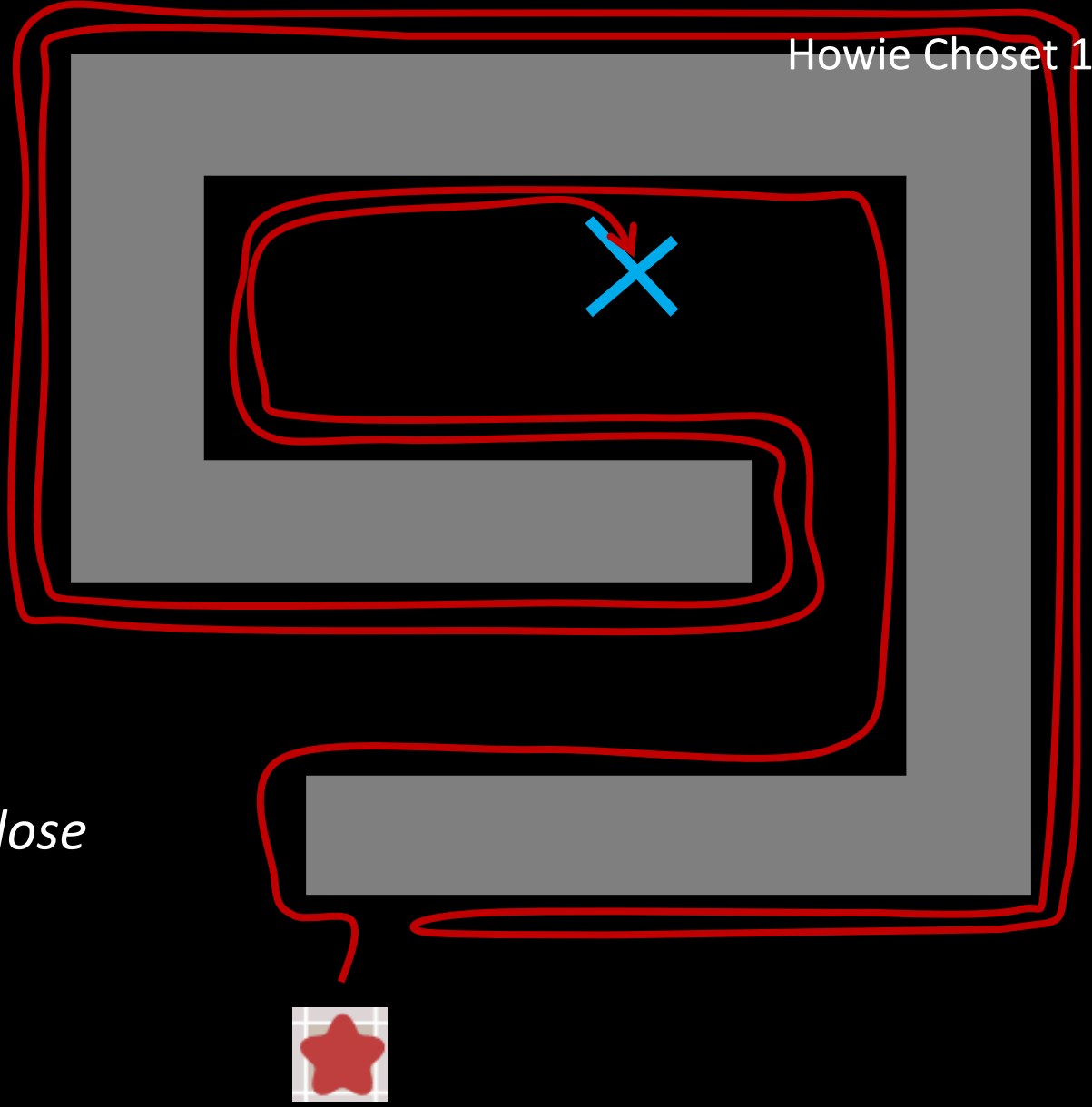
1. Go towards goal
2. Follow obstacles *and remember how close you got to the goal*
3. Return to the closest point, and loop



Bug 1

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry



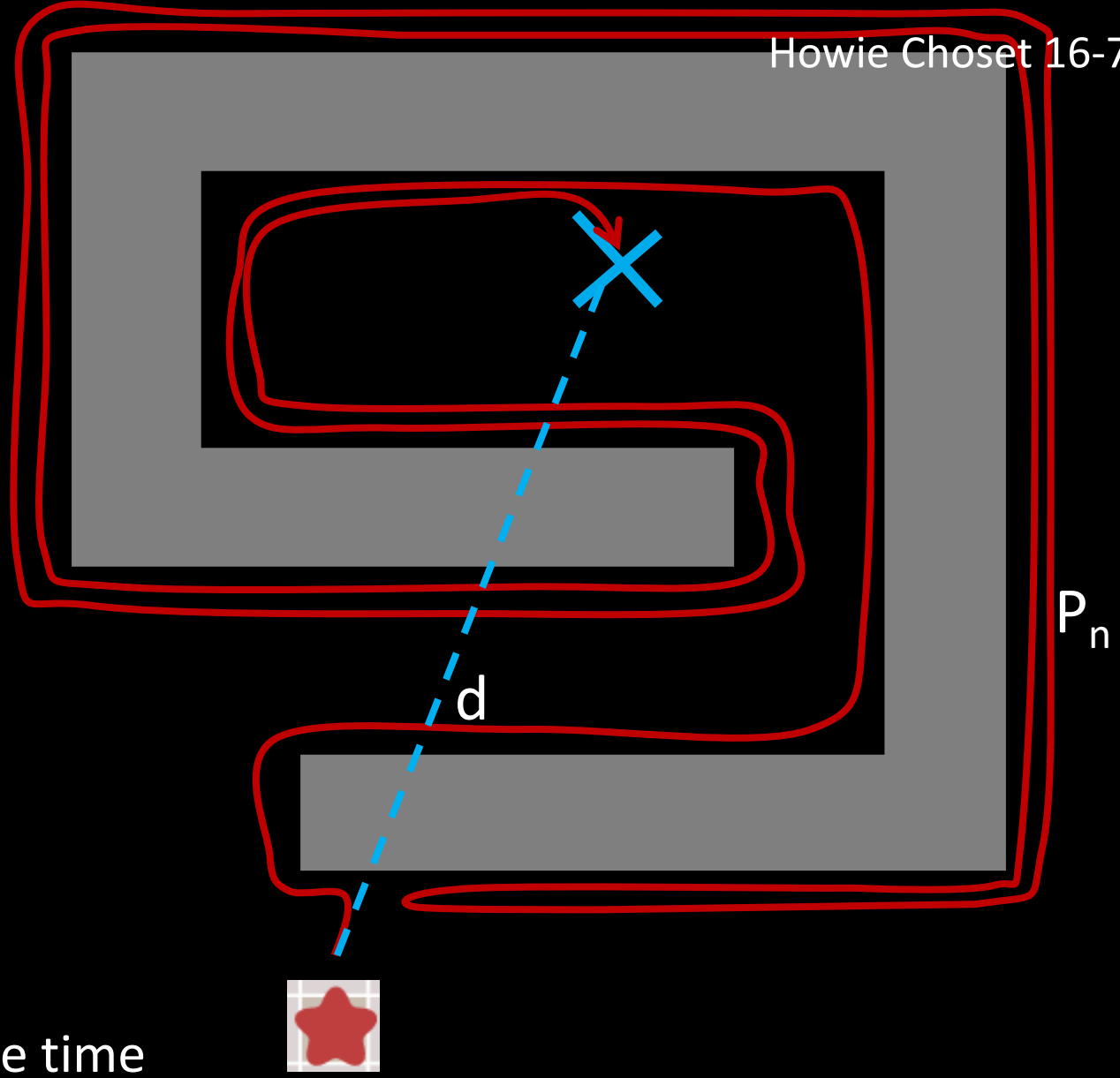
Algorithm

1. Go towards goal
2. Follow obstacles *and remember how close you got to the goal*
3. Return to the closest point, and loop

Bug 1 - formally

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry



- ***Lower bound traversal?***
  - d
- ***Upper bound traversal?***
  - $d + 1.5 \cdot \text{Sum}(P_n)$
- ***Pros?***
  - If a path exist, it returns one in finite time
  - AND it knows if *none* exist!

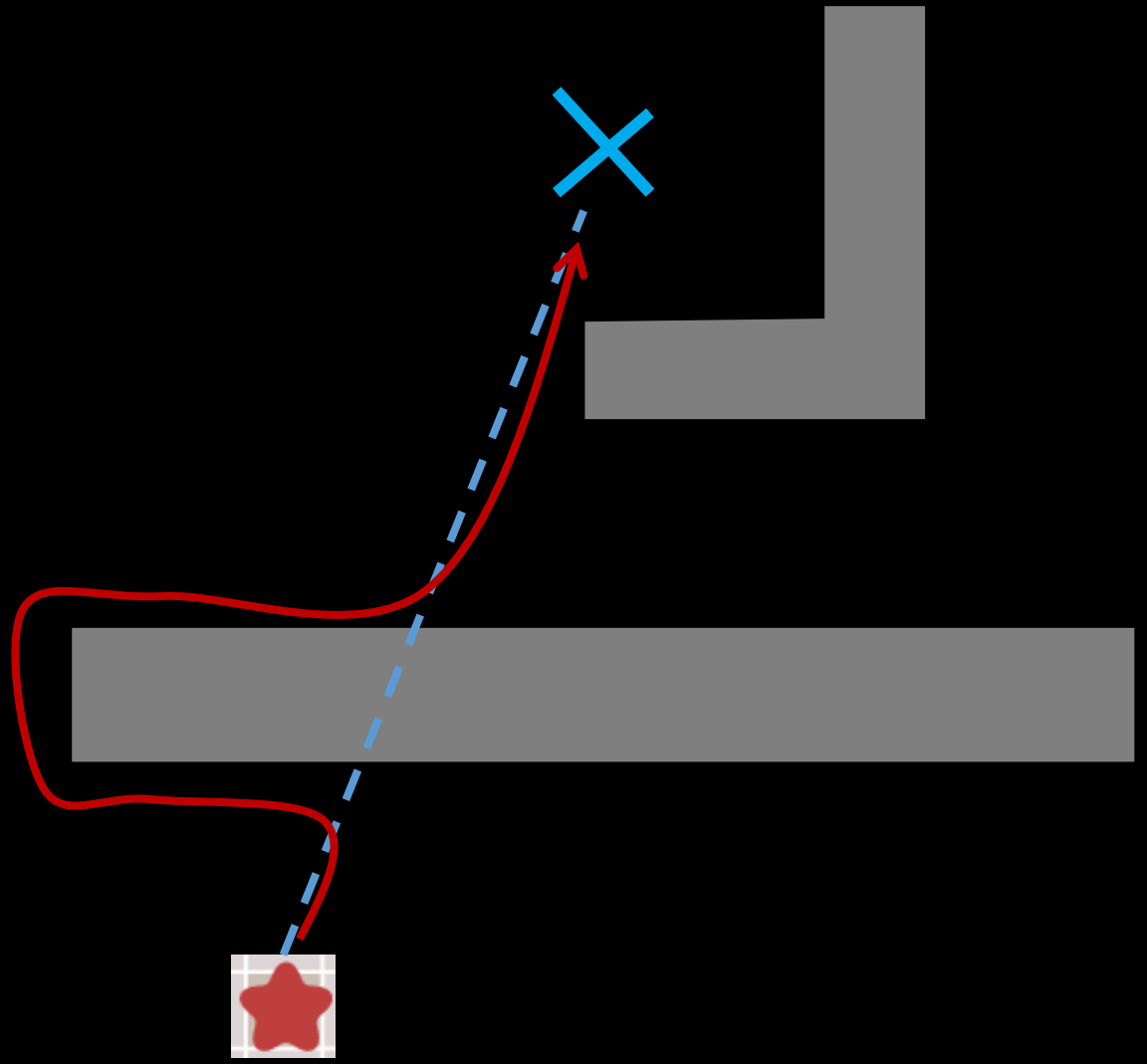
Bug 2

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry
- Original vector to the goal

Algorithm

1. Go towards goal on the vector
2. Follow obstacles *until you are back on the vector (and closer to the obstacle)*
3. Loop



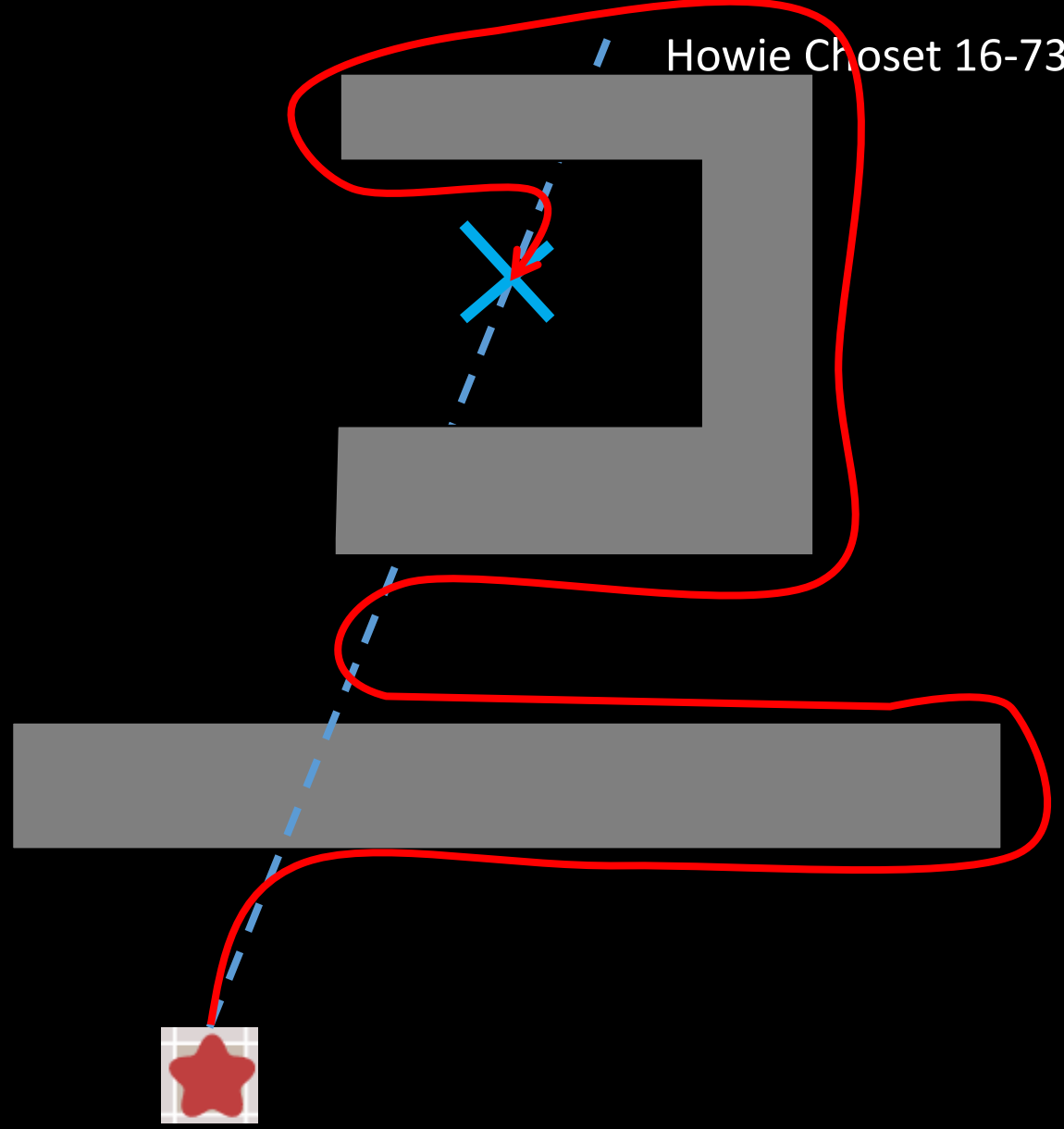
Bug 2

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry
- Original vector to the goal

Algorithm

1. Go towards goal on the vector
2. Follow obstacles *until you are back on the vector (and closer to the obstacle)*
3. Loop



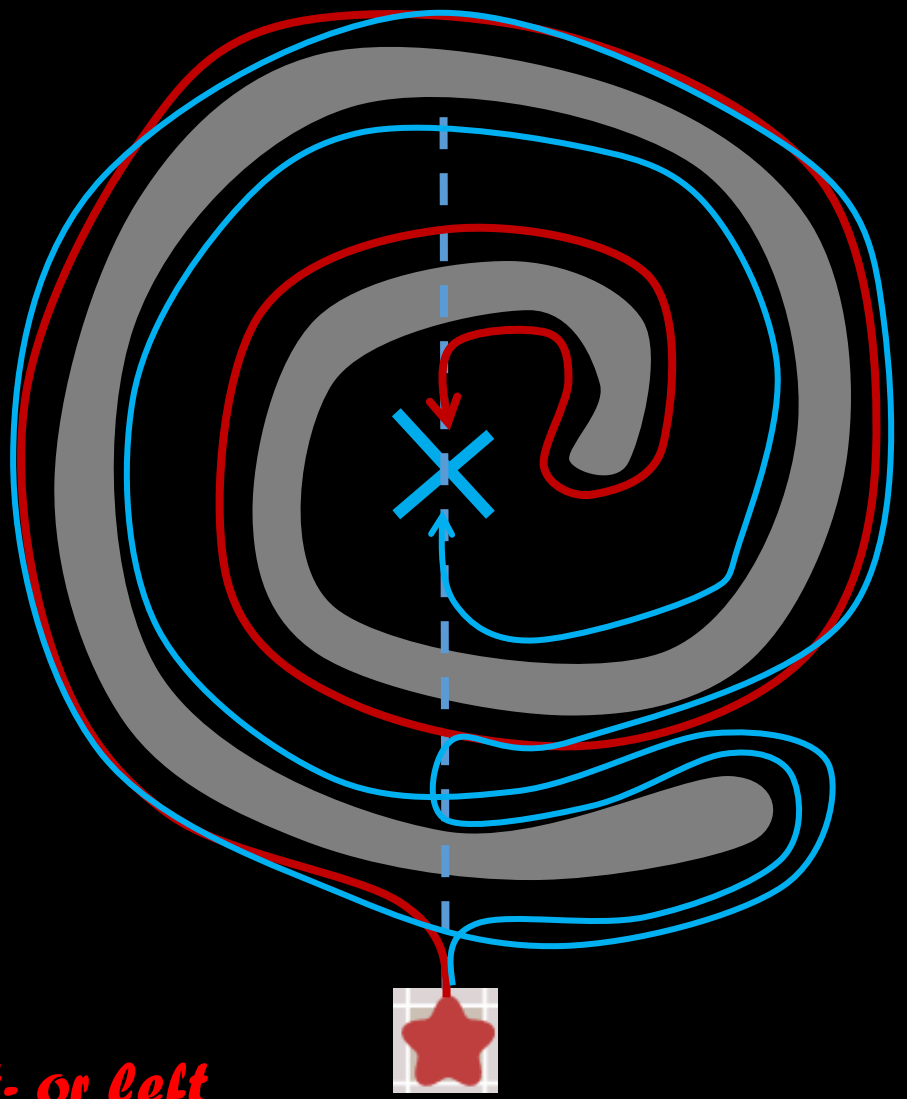
Bug 2

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry
- Original vector to the goal

Algorithm

1. Go towards goal on the vector
2. Follow obstacles *until you are back on the vector (and closer to the obstacle)*
3. Loop



What is faster, right- or left wall following?

Battle of the Bugs (1 vs 2)

https://www.youtube.com/watch?v=T2PVaKyxMmY

Bug 1
Layout 1

Bug 2
Layout 1

Battle of the Bugs (1 vs 2)

https://www.youtube.com/watch?v=T2PVaKyxMmY

Exhaustive Search

Bug 1
Layout 2

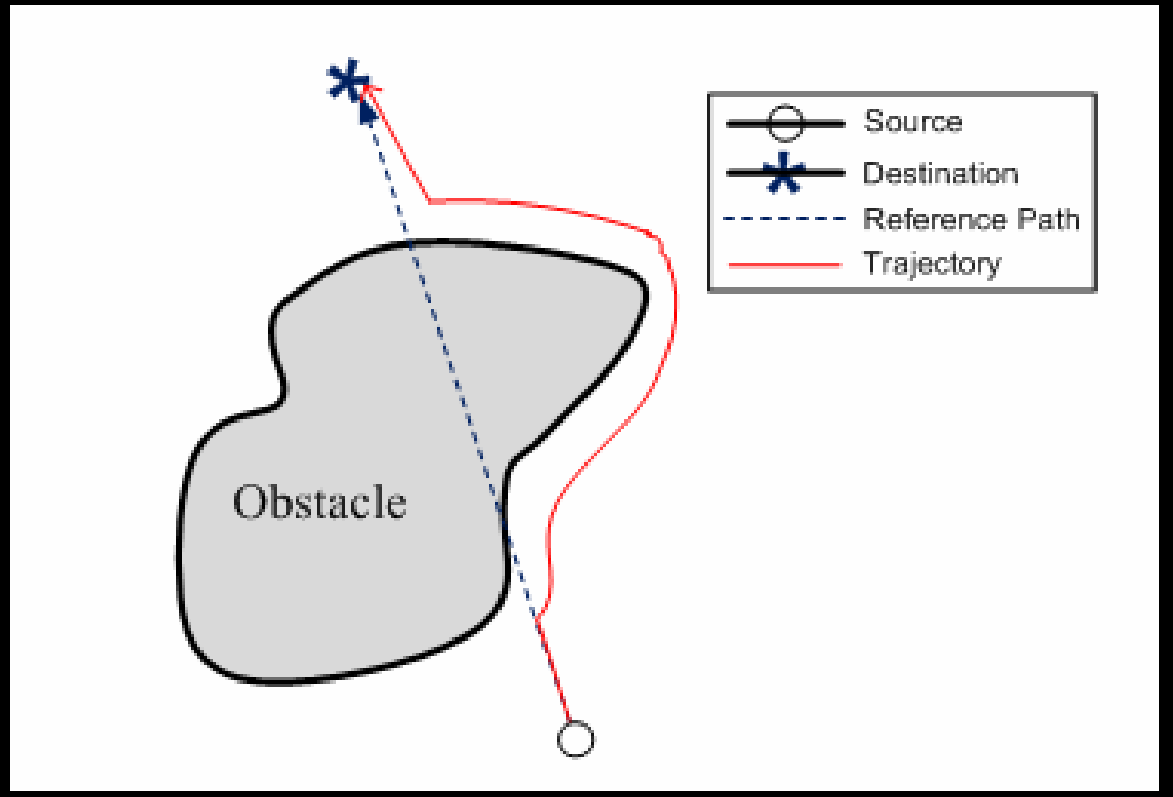
Greedy Search

Bug 2
Layout 2

Bug Algorithms

- Uses local knowledge, and the direction and distance to the goal
- Basic idea
  - Follow the contour of obstacles until you see the goal
  - State 1: Seek goal
  - State 2: follow wall
- Different variants: Bug0, Bug1, Bug2

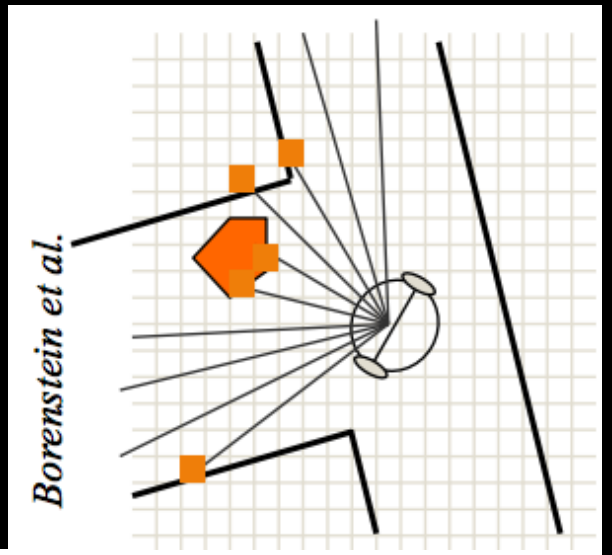
- The robot motion behavior is *reactive*
- Issues if the instantaneous sensor readings do not provide enough information or are noisy

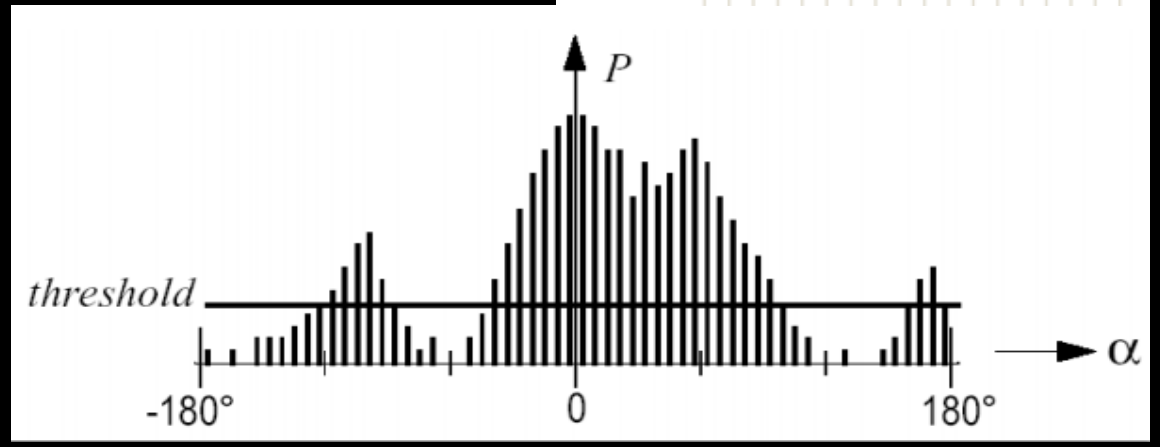


Vector Field Histograms

http://www.personal.umich.edu/~johannb/Papers/paper16.pdf

- VFH creates a local map of the environment around the robot populated by “relatively” recent sensor readings
- Build a local 2D grid map → reduce to 1-DoF histogram
- Planning
  - Find all openings large enough for robot to pass
  - Choose the one with the lowest cost, G
  - G = a*goal_direction + b*orientation + c*prev_direction

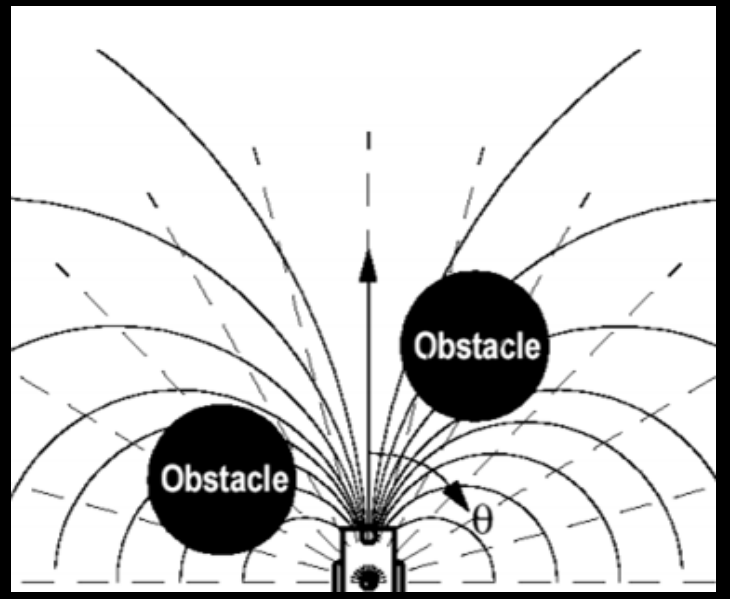


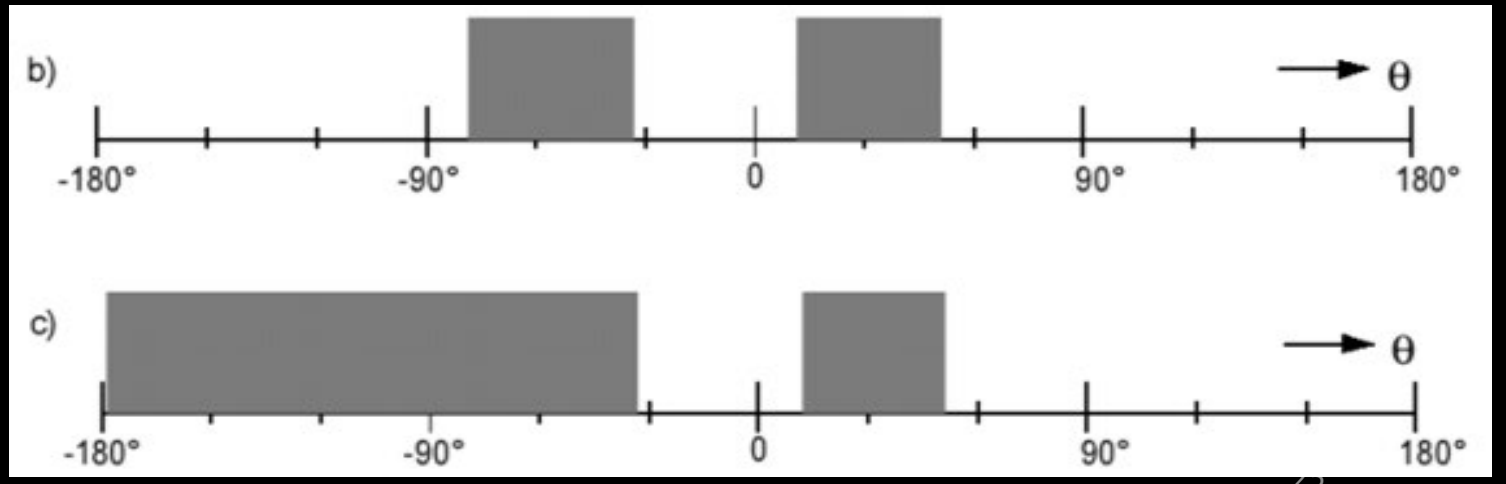


Vector Field Histograms

http://www.personal.umich.edu/~johannb/Papers/paper16.pdf

- VFH creates a local map of the environment around the robot populated by “relatively” recent sensor readings
- Build a local 2D grid map → reduce to 1-DoF histogram
- Planning
  - Find all openings large enough for robot to pass
  - Choose the one with the lowest cost, G
  - G = a*goal_direction + b*orientation + c*prev_direction
  - VHF+: Incorporate kinematics
- Limitations
  - Does not avoid local minima
  - Not guaranteed to reach goal





Dynamic Window Approach

http://www4.cs.umanitoba.ca/~jacky/Teaching/Courses/74.795-LocalVision/ReadingList/fox97dynamic.pdf

- Search in the velocity space (robot moves in circular arcs)
  - Takes into account robot acceleration capabilities and update rate
- A dynamic window, V_d, is the set of all tuples (v_d, ω_d) that can be reached
- Admissible velocities, V_a, include those where the robot can stop before collision
- The search space is then $V_r = V_s \cap V_a \cap V_d$
- Cost function: $G(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega))$

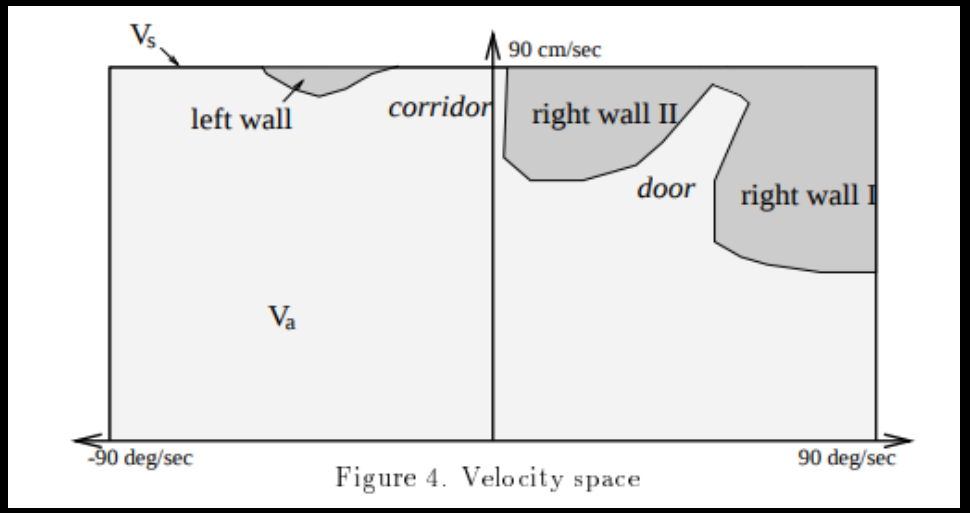
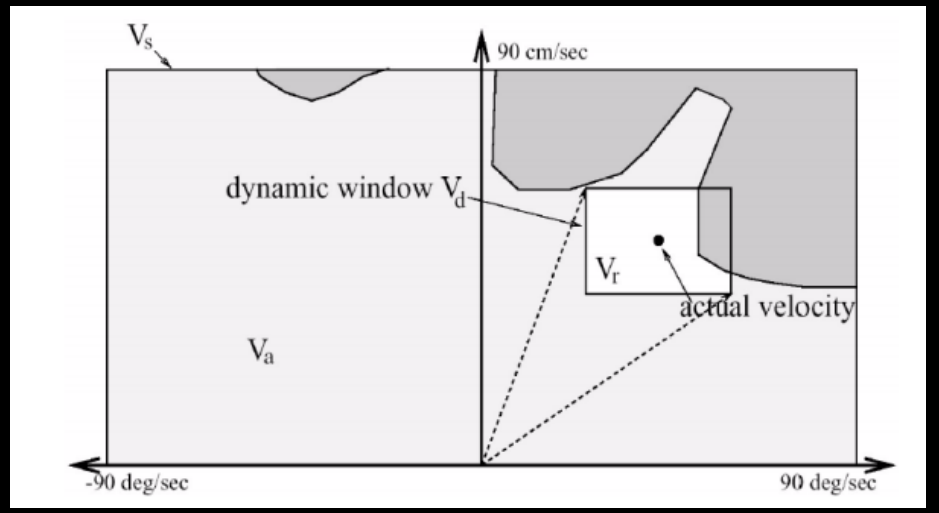


Figure 4. Velocity space

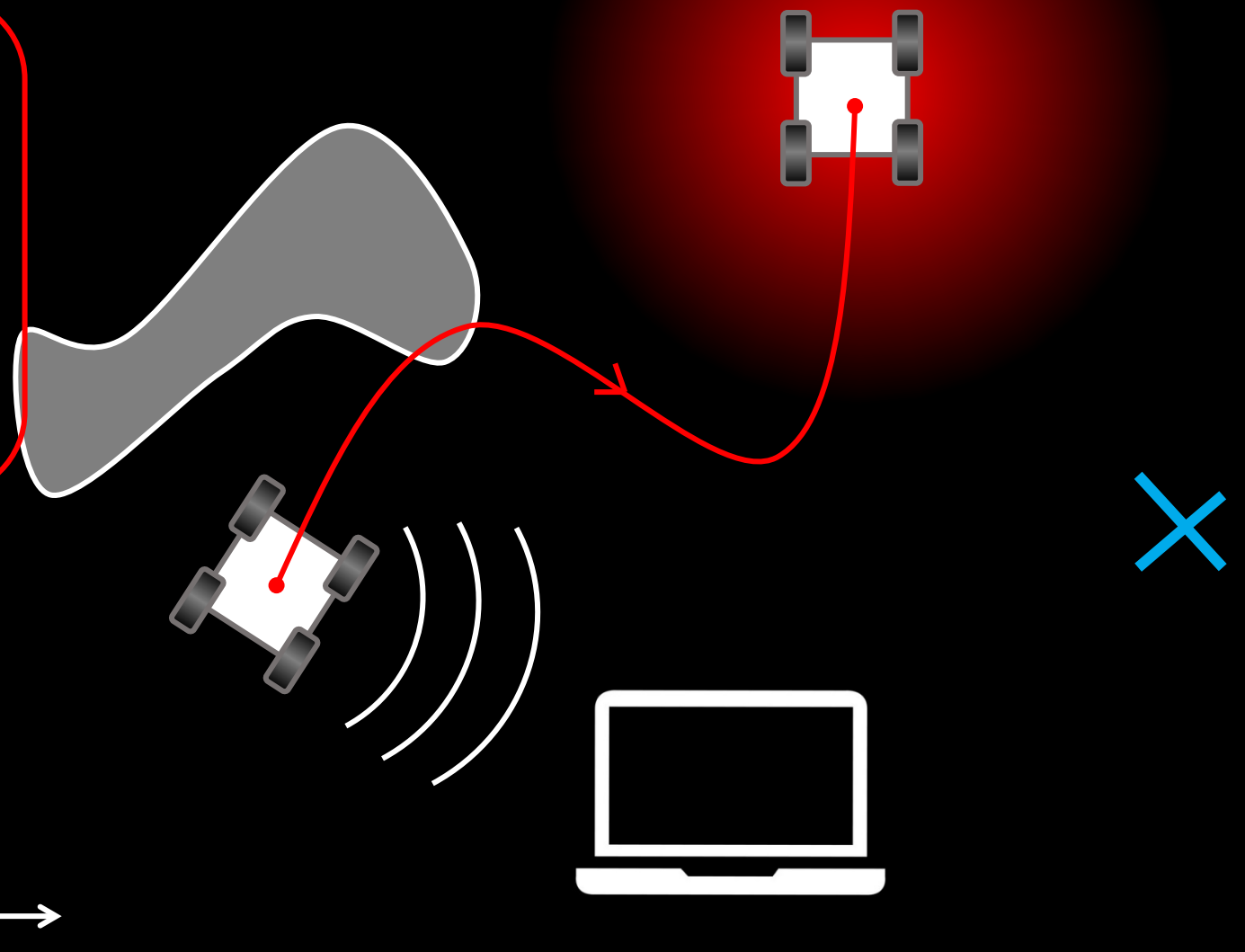


Local Planning Algorithms, Summary

- Bug Algorithms
  - Inefficient, but can be exhaustive
- Vector Field Histograms
  - Takes into account probabilistic sensor measurements
- Vector Field Histograms +
  - Takes into account probabilistic sensor measurements and robot kinematics
- Dynamic Window Approach
  - Takes into account robot dynamics

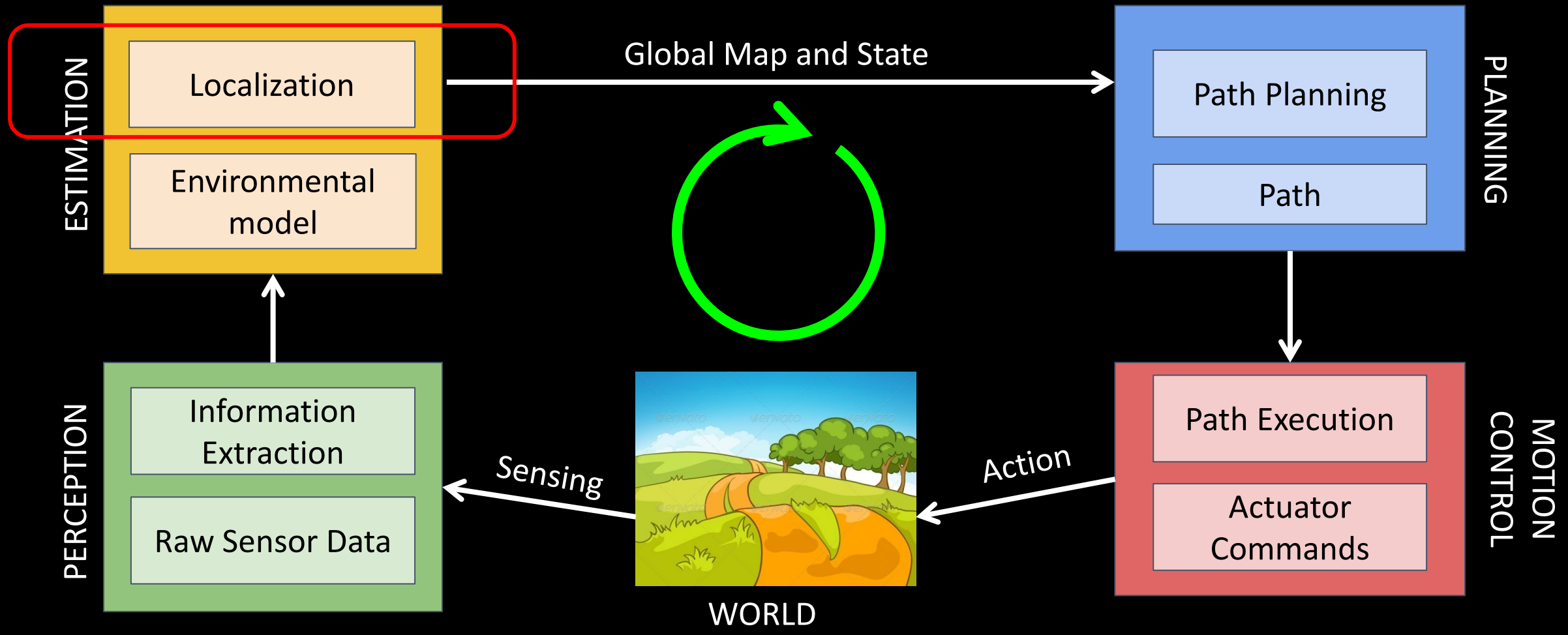
Outline of the next module on Navigation

- Local planners
- Global localization and planning
  - Configuration space
  - Map representations
    - Continuous
    - Discrete
    - Topological
  - Graph Search Algorithms
    - Breadth First Search
    - Depth First Search
    - Dijkstras
    - A*



Navigation and Path Planning

- Navigation breaks down to: Localization, Map Building, Path Planning



Localization Problem

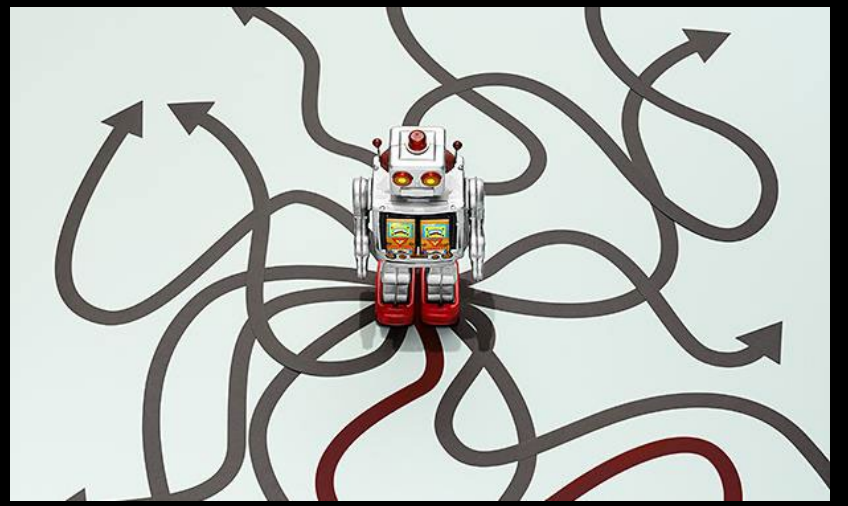
Position Tracking

- Initial robot pose is known
- Achieved by accommodating the noise in robot motion
- It is a “local” problem, as the uncertainty is local (often small) and confined to a region near the robot’s true pose

Global Localization

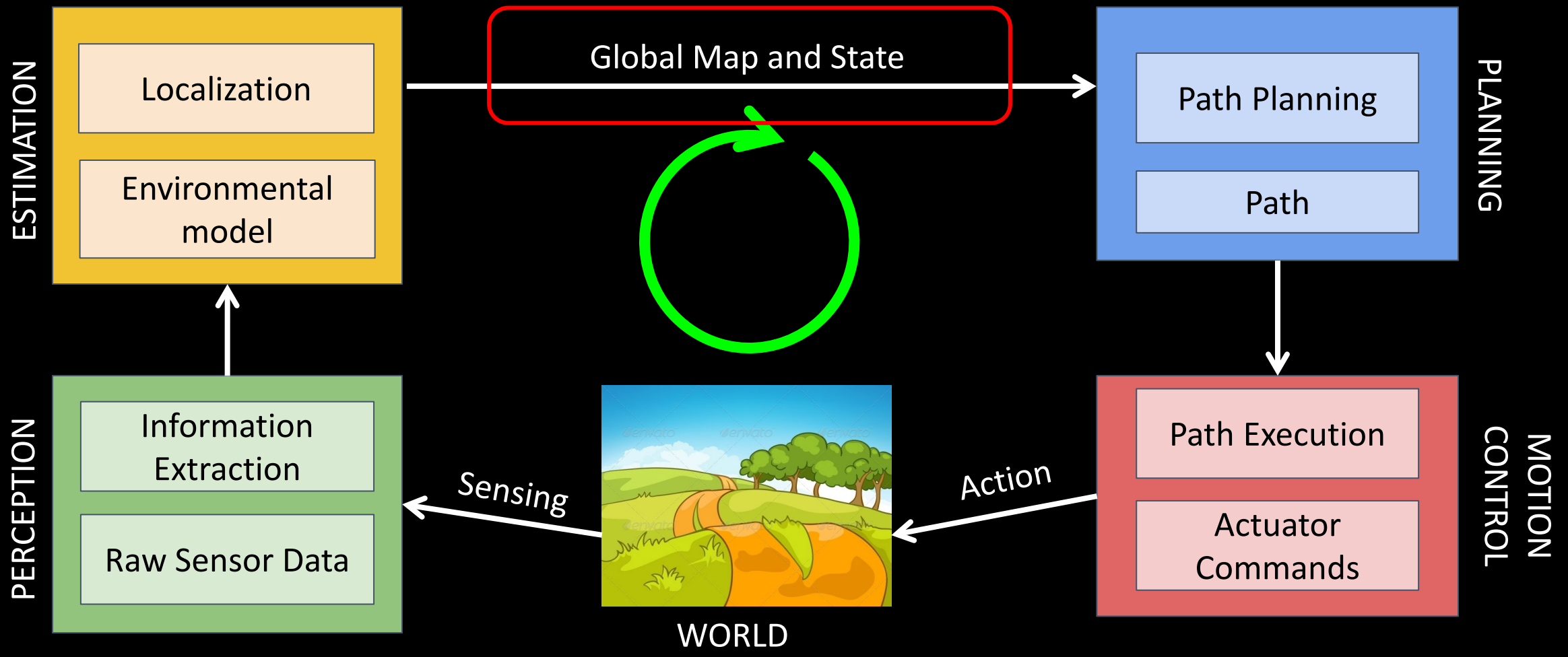
- Initial robot pose is unknown
- Need to estimate position from scratch
- A more difficult “global” problem, where you cannot assume boundedness in pose error

kidnapped robot problem



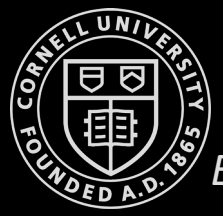
Navigation and Path Planning

- Navigation breaks down to: Localization, Map Building, Path Planning



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Map Representations



Map Representation

(a) Building plan

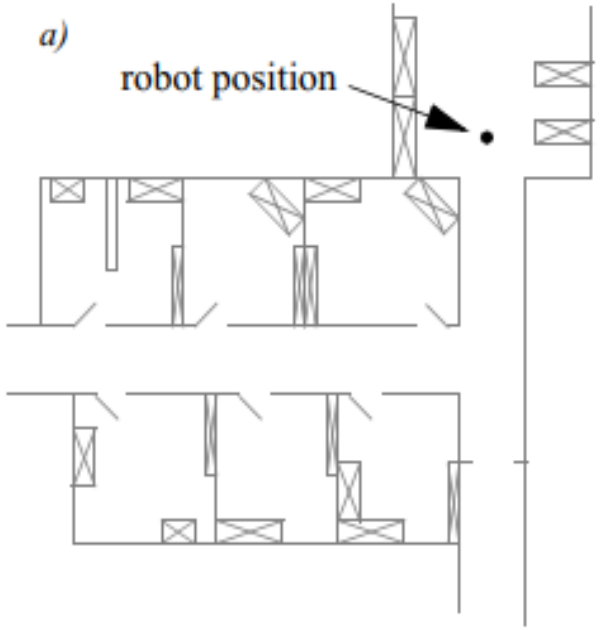
(b) line-based map

(c) occupancy grid-based map

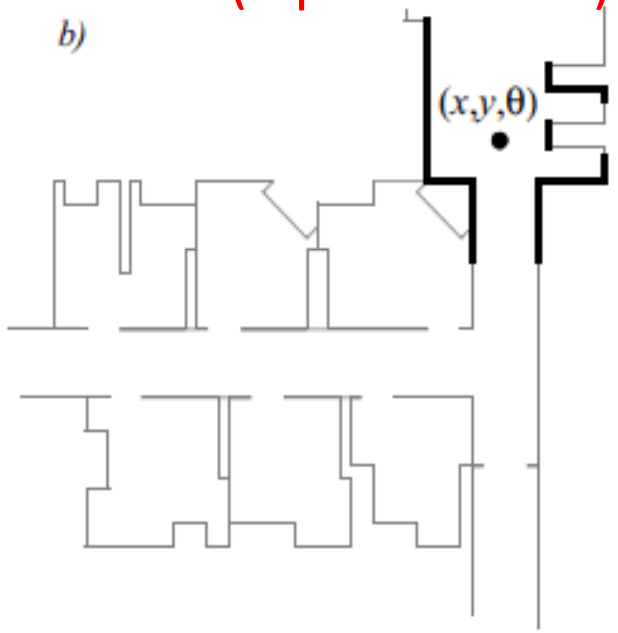
(d) topological map

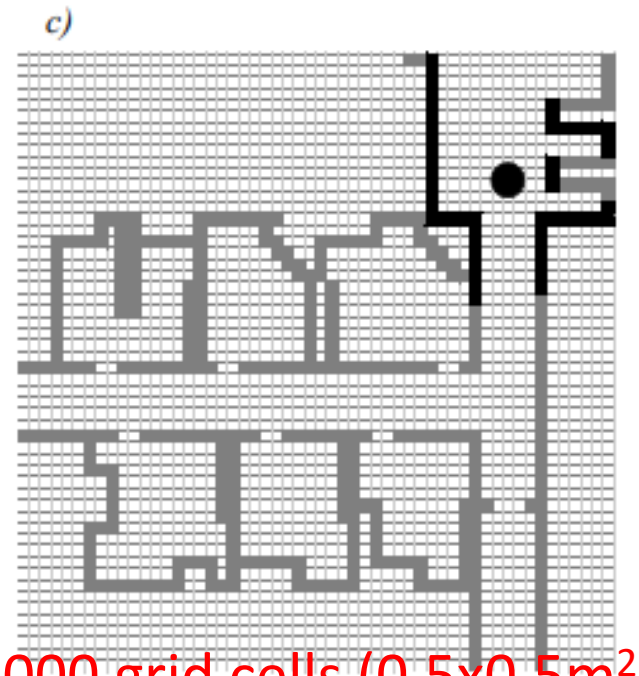
Important properties

- Memory allocation
- Computation
- Robot pose

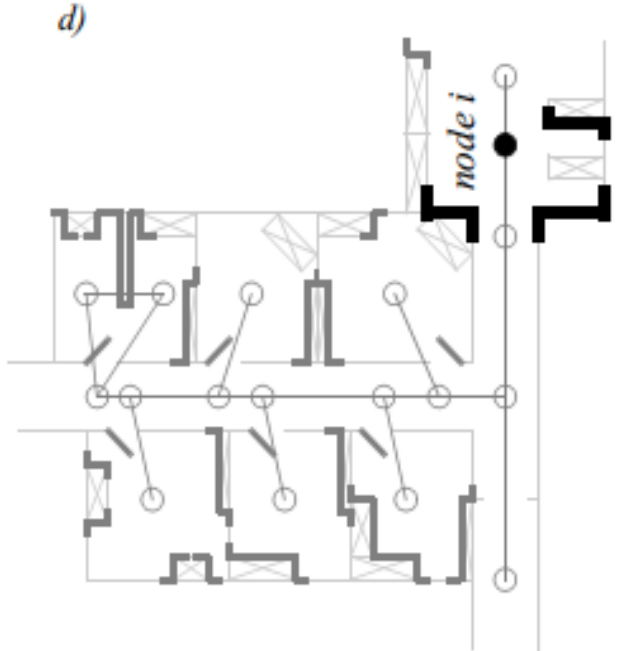


100 lines (2 parameters)



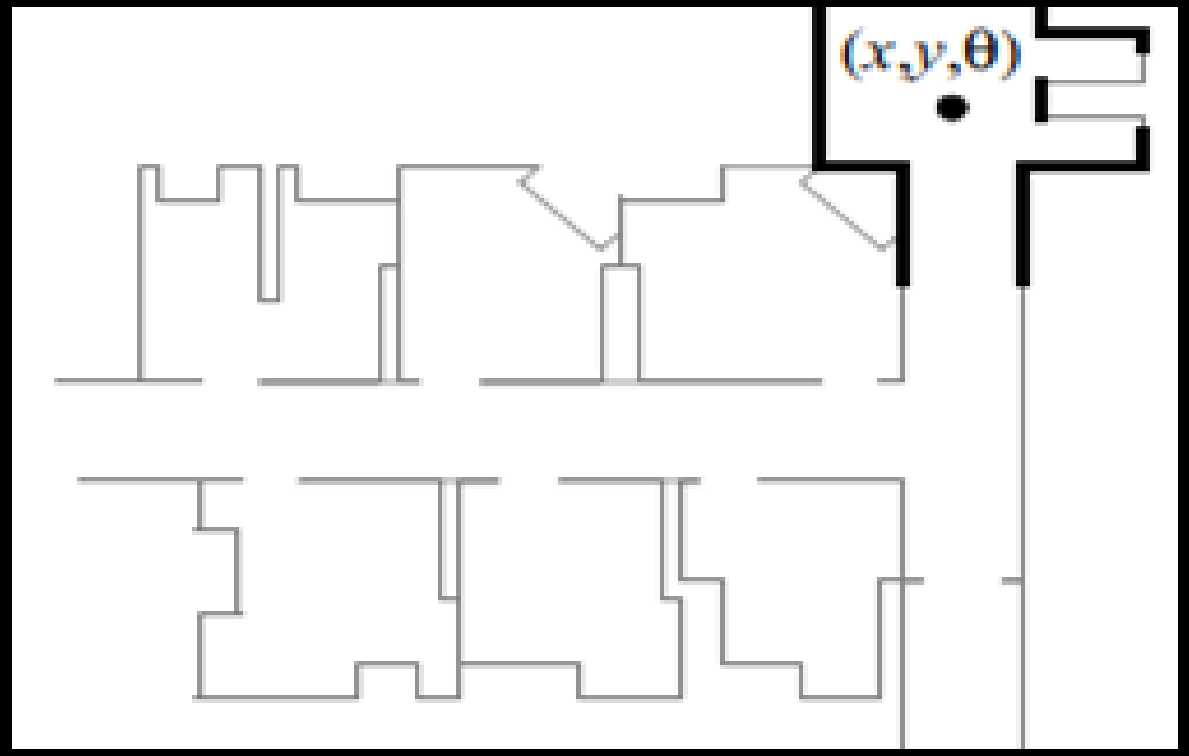


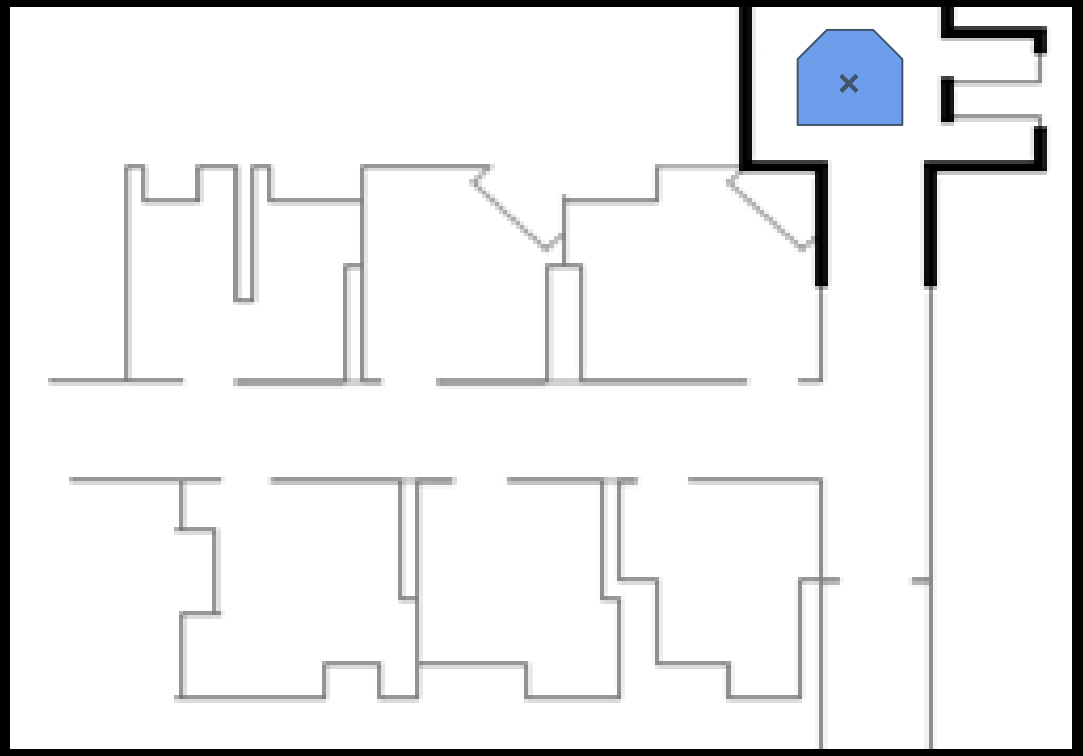
3000 grid cells (0.5x0.5m^2)



50 features, 18 nodes

What if the robot is not a point?

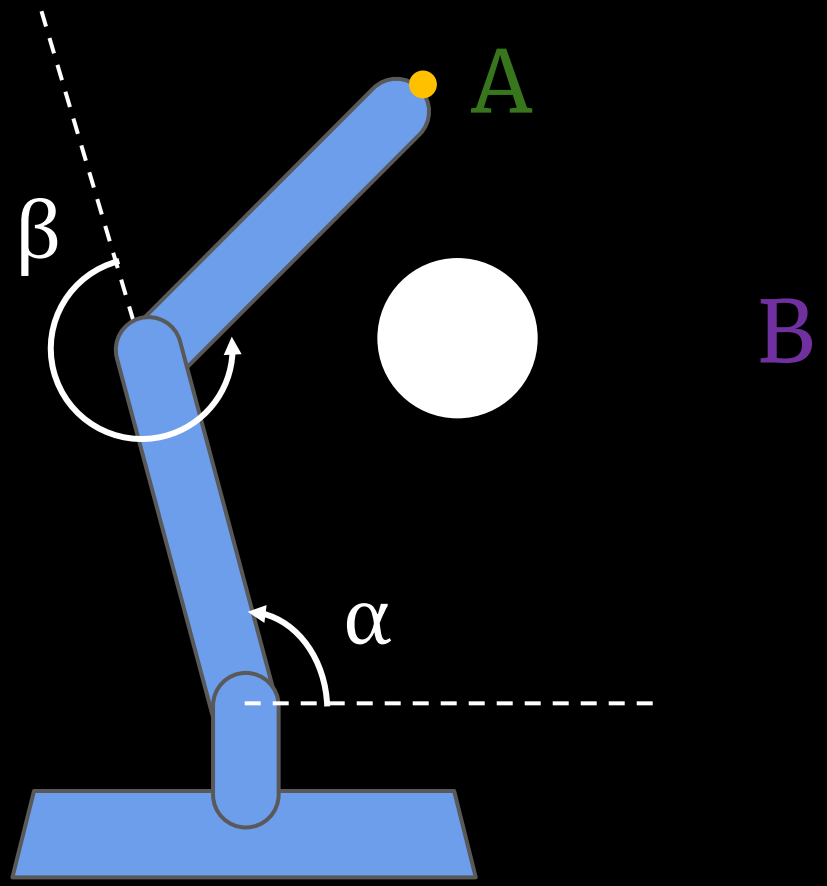




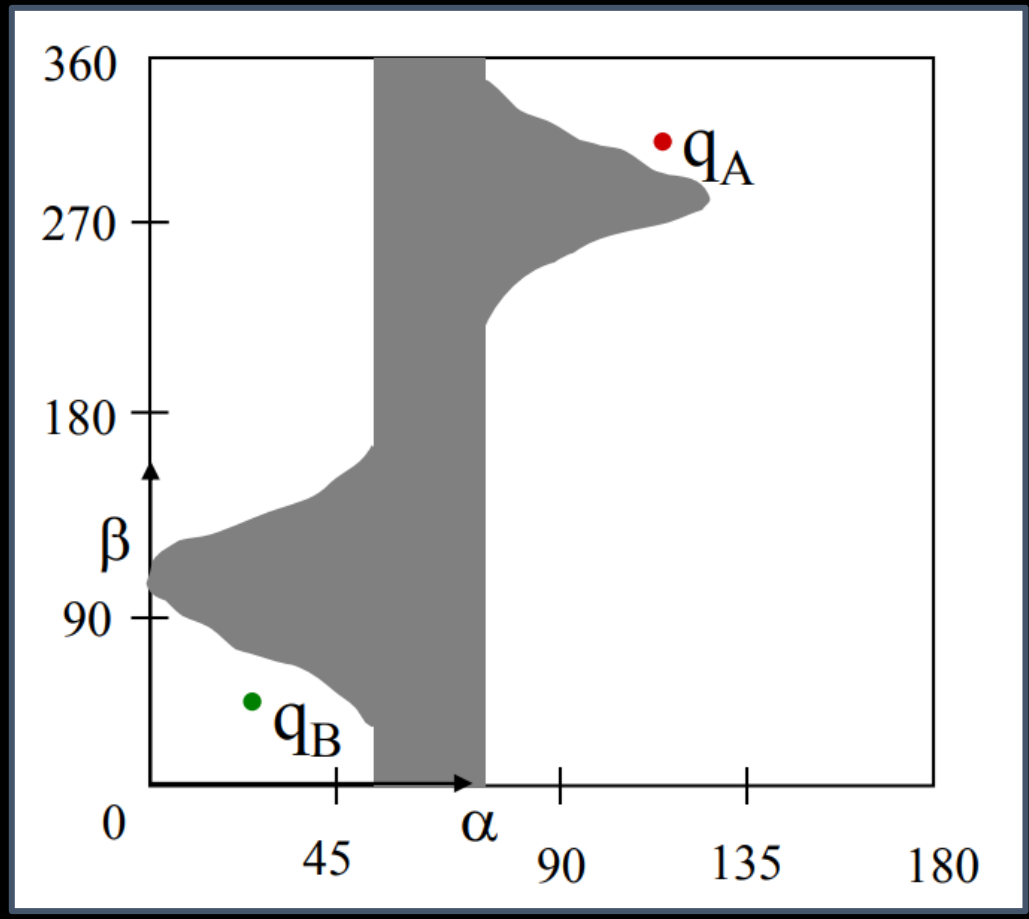
Configuration Space

- Each coordinate in the configuration space represents a robot degree of freedom
  - Global motion planning normally takes place in the configuration space

Ex 1: Planar arm



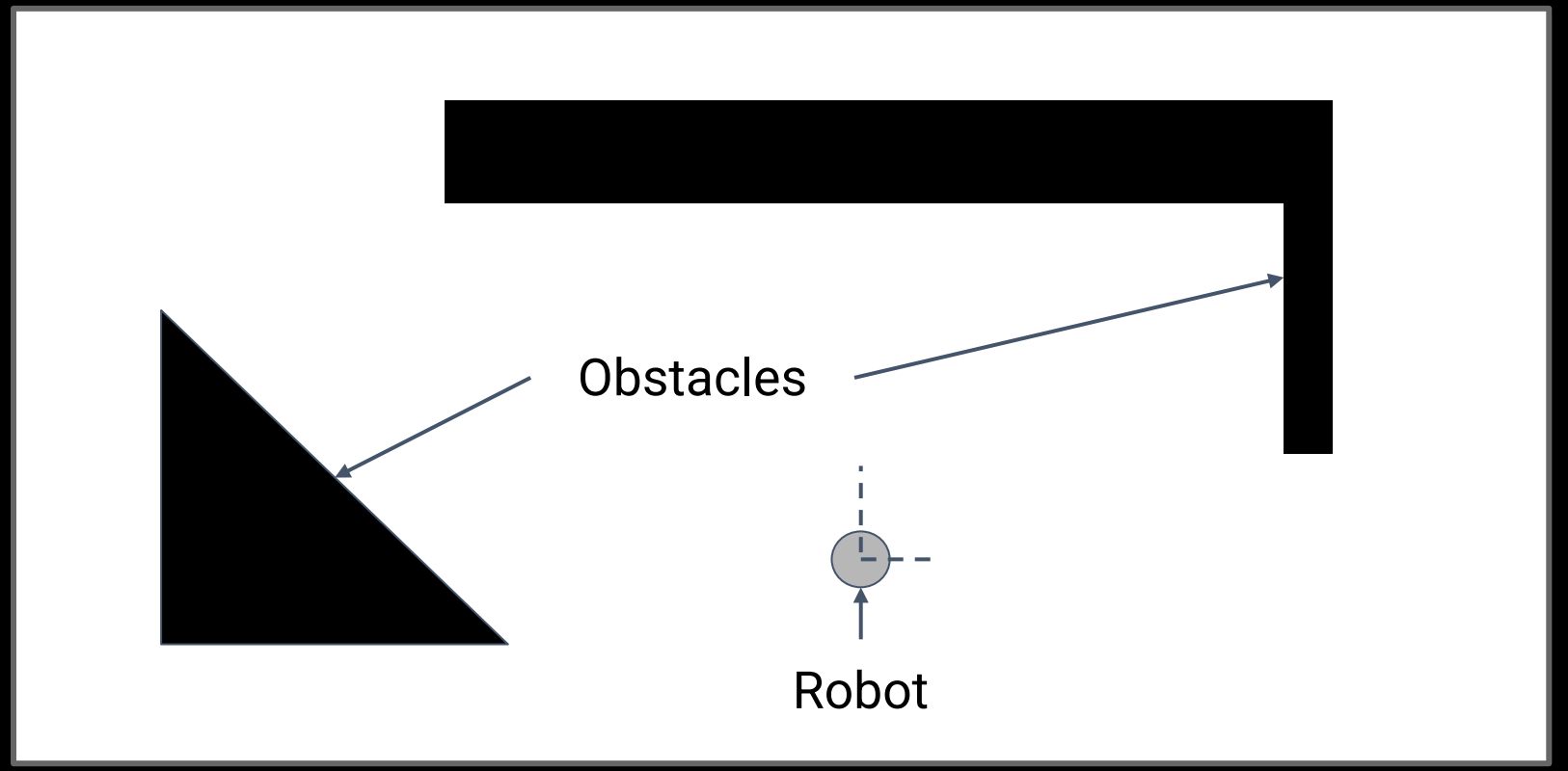
B



Configuration Space

- Each coordinate in the configuration space represents a robot degree of freedom
  - Global motion planning normally takes place in the configuration space

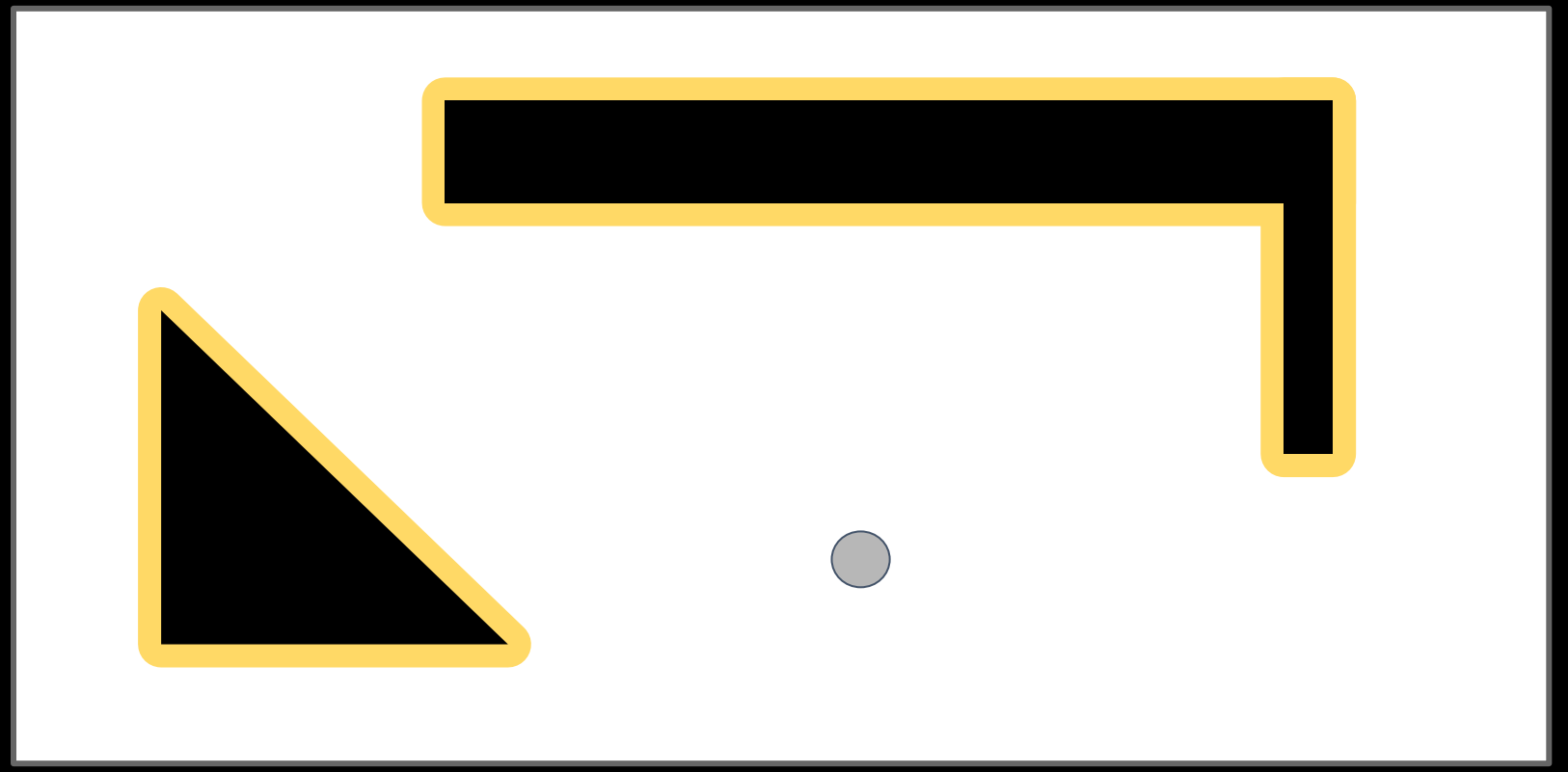
Ex 2: Circular robot in 2D world



Configuration Space

- Each coordinate in the configuration space represents a robot degree of freedom
  - Global motion planning normally takes place in the configuration space

Ex 2: Circular robot in 2D world



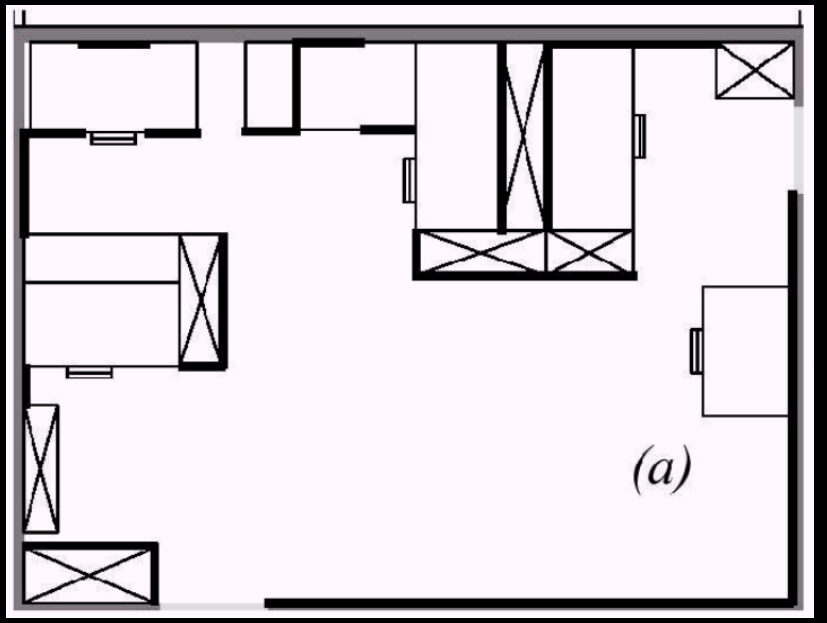
Map Representation Considerations

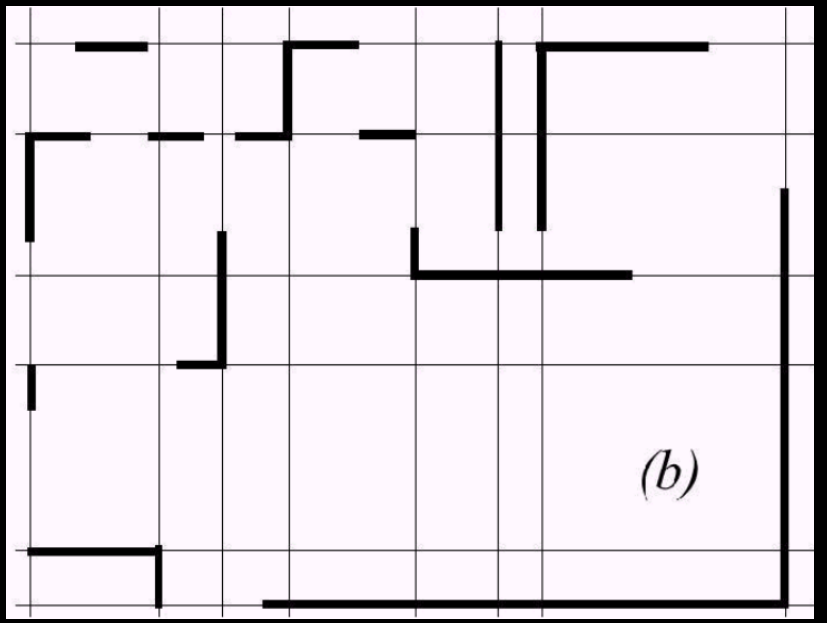
"Vivek's three rules for map representation"

- The precision of the map must appropriately match the precision with which the robot needs to achieve its goals
- The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors
- The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization, and navigation

Continuous Representations

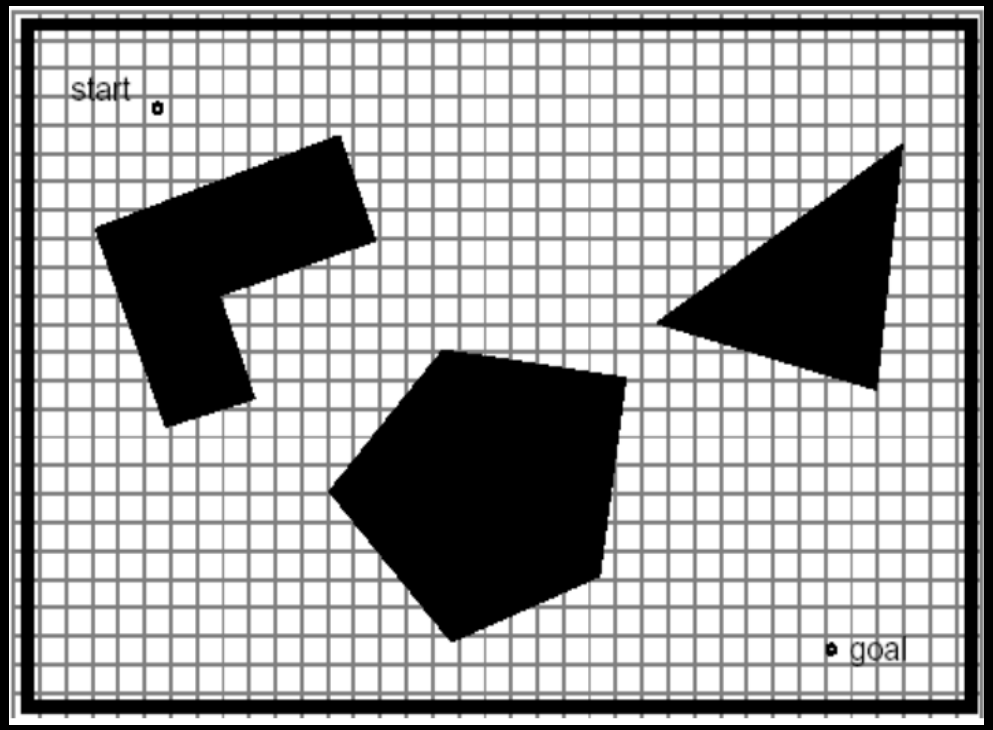
- Exact decomposition of the environment
- Used mainly in 2D representations
- Closed-world assumption
- Storage proportional to object density
- Example: Continuous line representations
  - Using range finders, we can extract lines/line segments in the environment

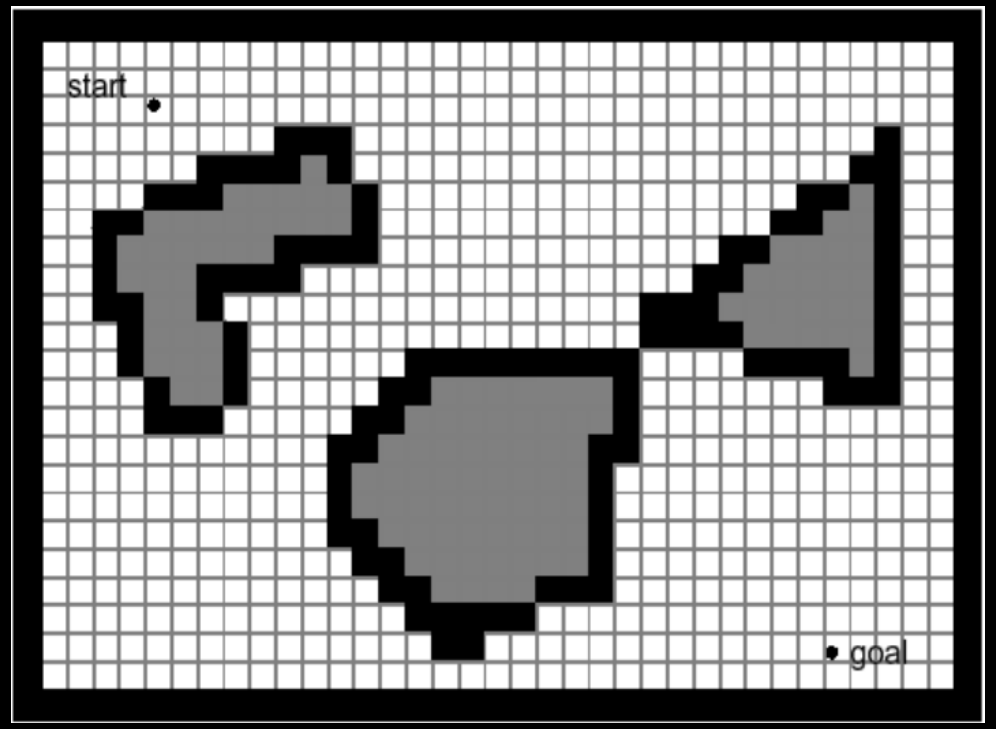




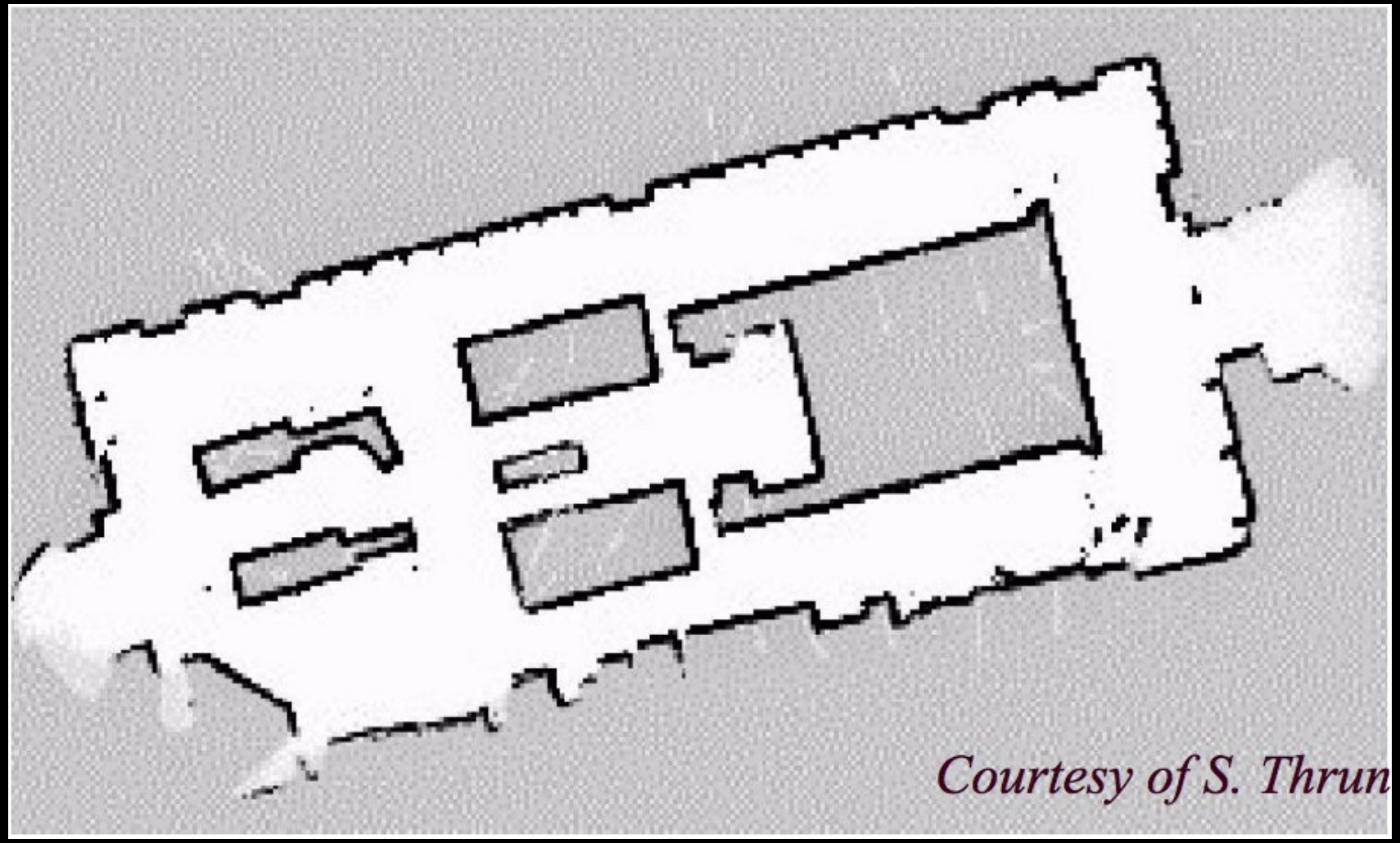
Fixed Decomposition

- Tessellate the world at a fixed resolution
- Approximate features given the resolution
- Most commonly used: Occupancy grid



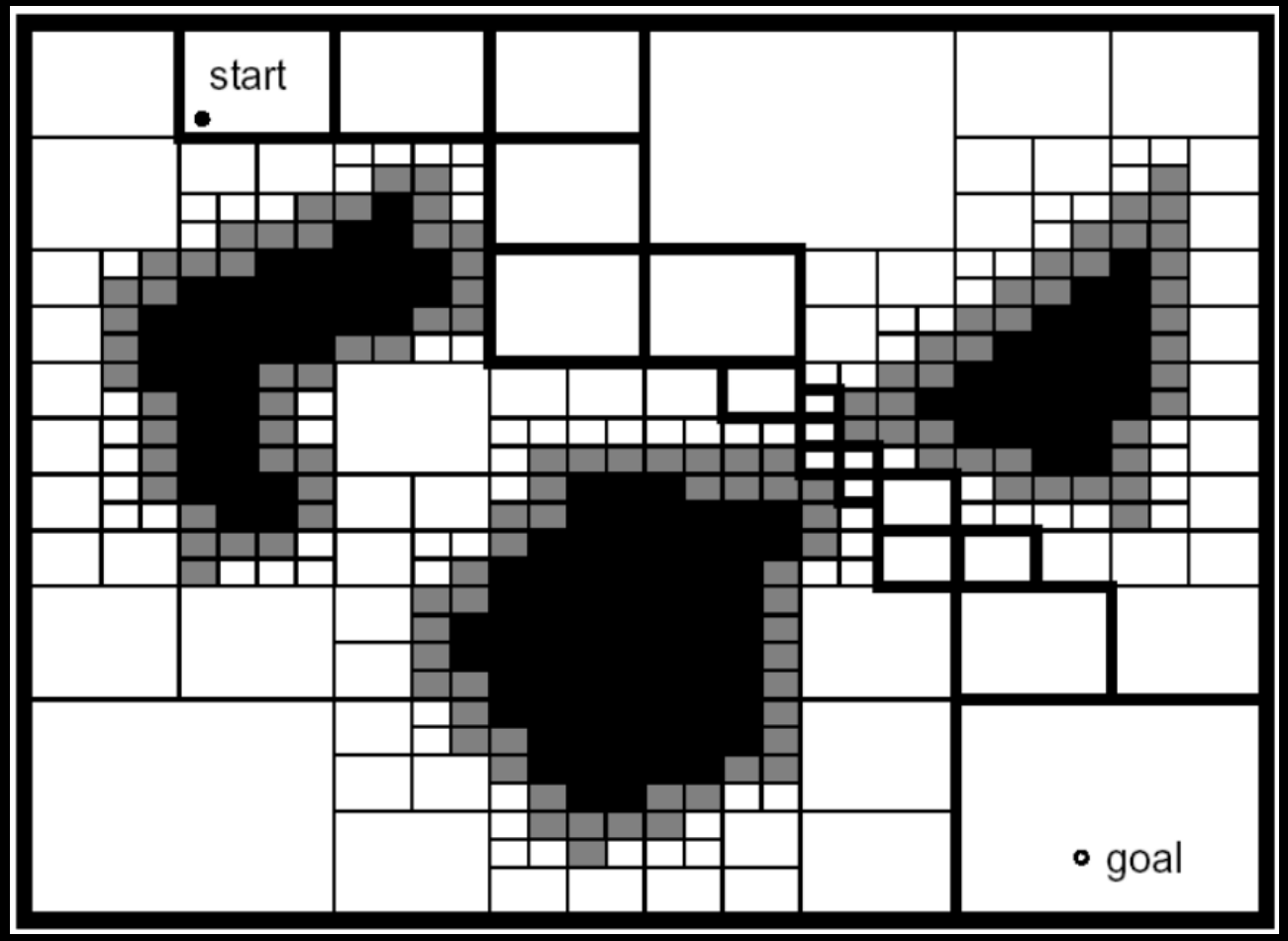


Fixed Decomposition

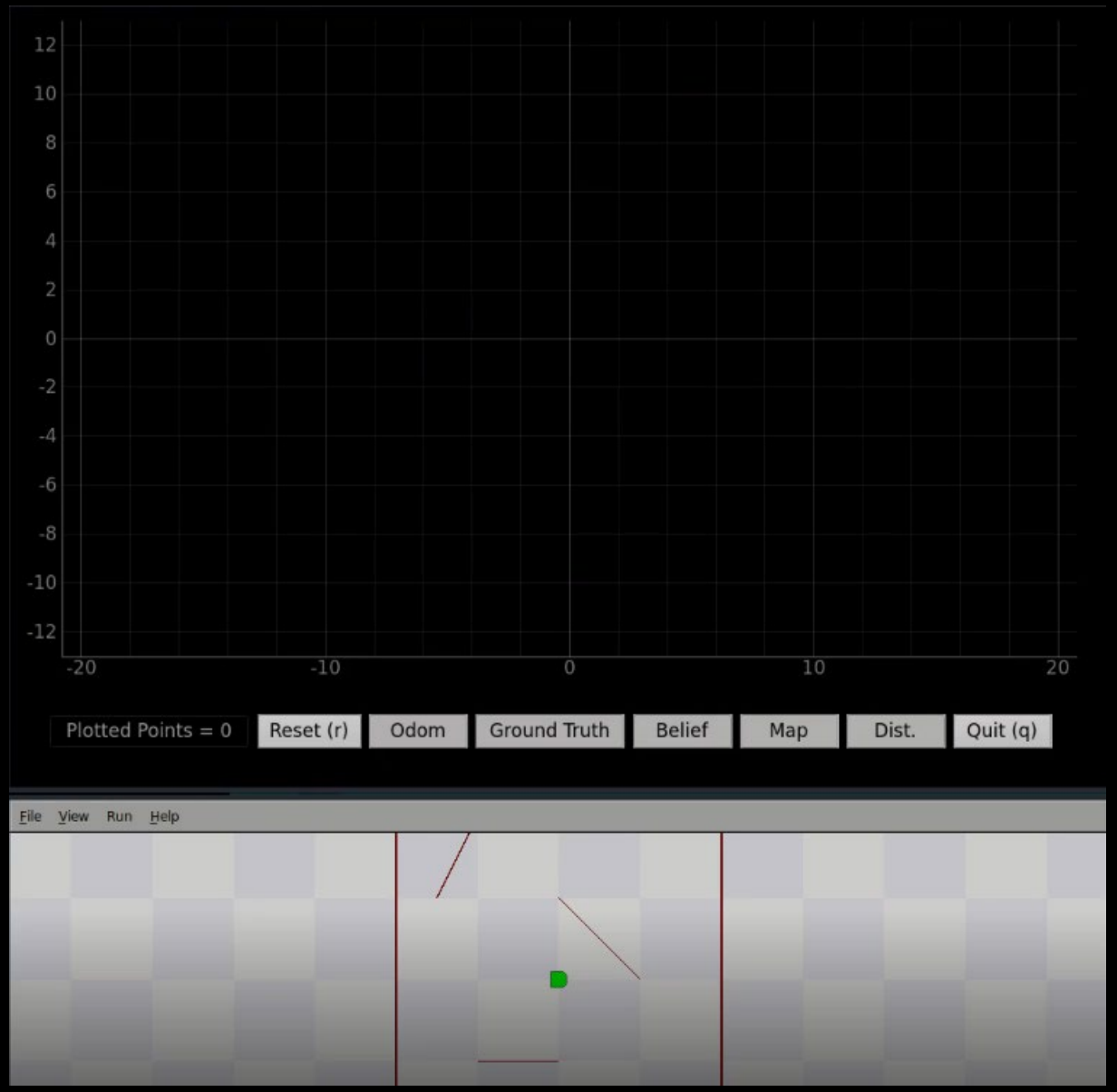


Adaptive Cell Decomposition

- Adapt cell size to features



Lab 9-13: Combo of Linear Representation and Fixed Decomposition



Courtesy of Vivek Thangavelu

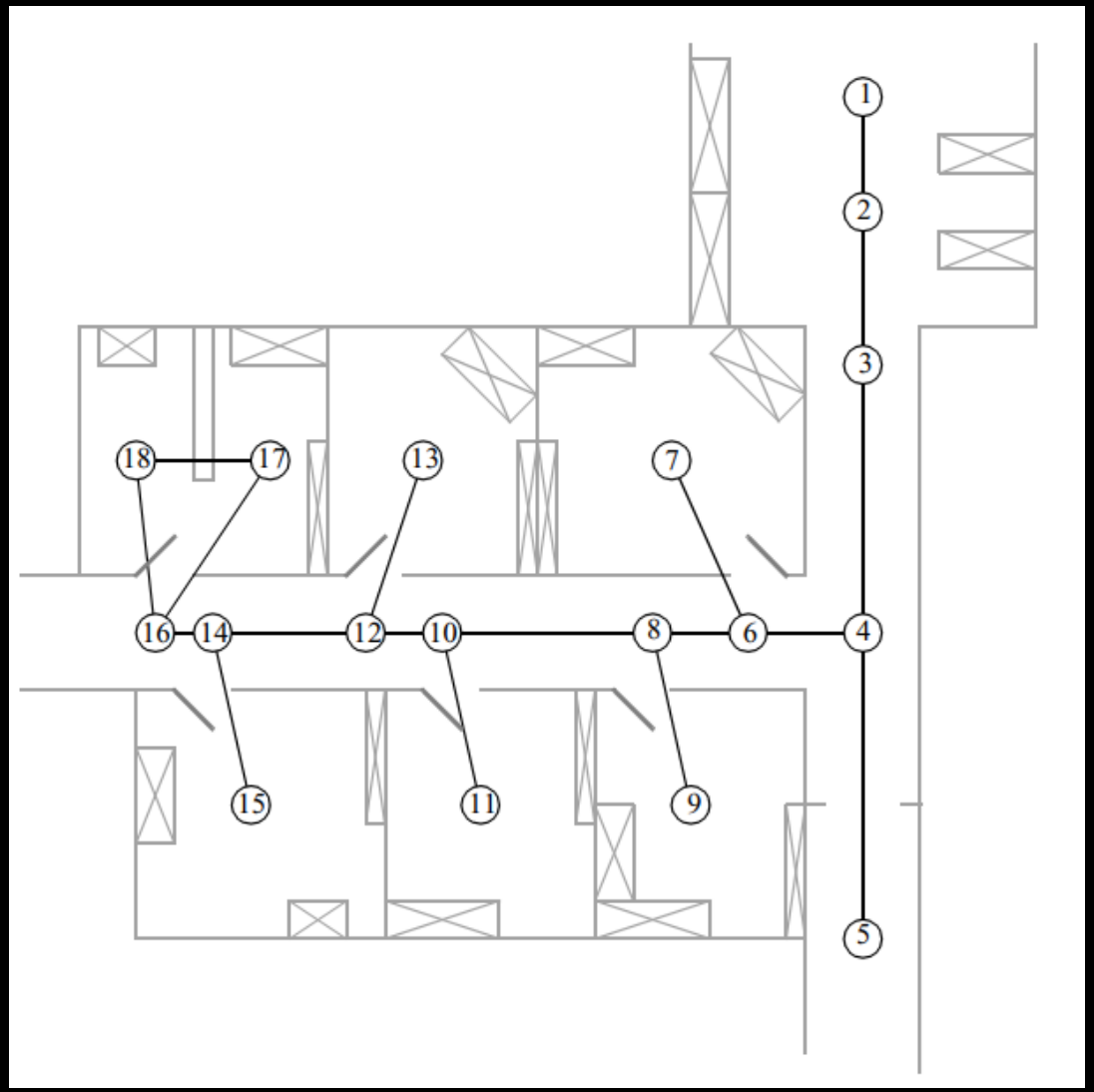
Robots in 3D Environment

- How many coordinates are needed now?
  - 6 DOF
- Representation requirements
  - Compact in memory
  - Efficient access and queries
  - Enables sensor fusion

- Solution
  - Topological Representation

Topological Decomposition

- A topological representation is a graph that specifies nodes and edges
  - Nodes denote areas in the environment
  - Edges describe environment connectivity
- Robots can…
  - …detect their current position in terms of the nodes of the topological graph
  - …travel between nodes using robot motion



Outline of the next module on Navigation

- Local planners
- Global localization and planning
  - Configuration space
  - Map representations
    - Continuous
    - Discrete
    - Topological
  - Graph Search Algorithms
    - Breadth First Search
    - Depth First Search
    - Dijkstras
    - A*