Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots

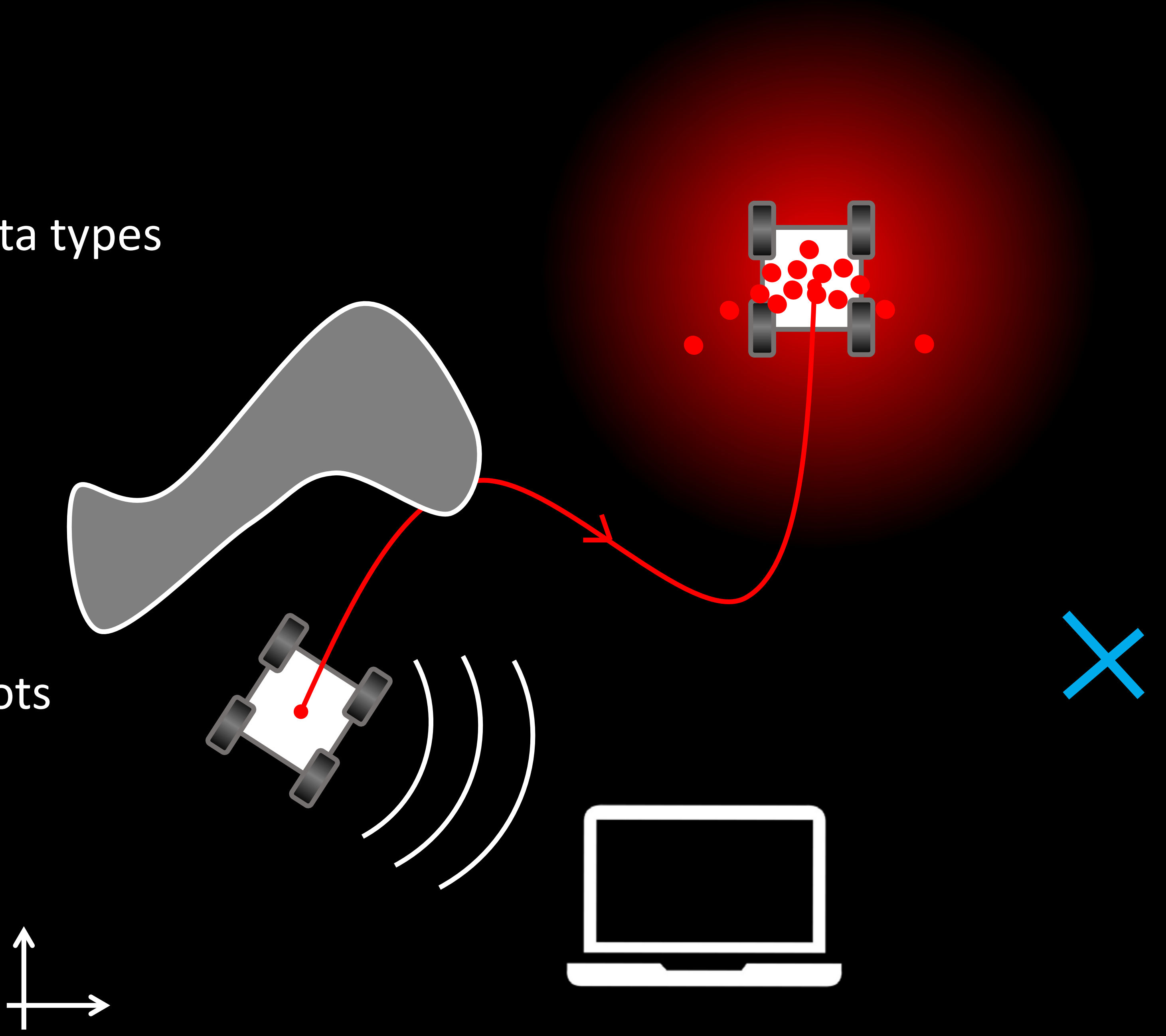
Slides adapted from Vivek Thangavelu

Progress: Sanity check!

- Ideas for how to break it up?
  - Divide up IMU lab and PID lab?
  - Divide up IMU+bluetooth+ramp lab and PID lab?
  - Make it a two-week lab
- What now?
  - We cut lab 7 in half!
  - Lab 7: Map your room
  - Lab 8: Localization on the simulator
  - Lab 9: Localization in your room
  - Lab 10: Path planning on your robot / simulator
    - Anything goes!
    - Global planning – local planning – open loop control
    - On-board – Off-board control

What we covered so far…

- Transformation matrices
- Bluetooth communication and data types
- Distance Sensors
- Odometry and IMU
- Robot/sensor characterization
- Noise and errors
- PID control
- Deterministic → Probabilistic robots
    - Odometry and sensor models
    - Bayes theorem, Bayes filter

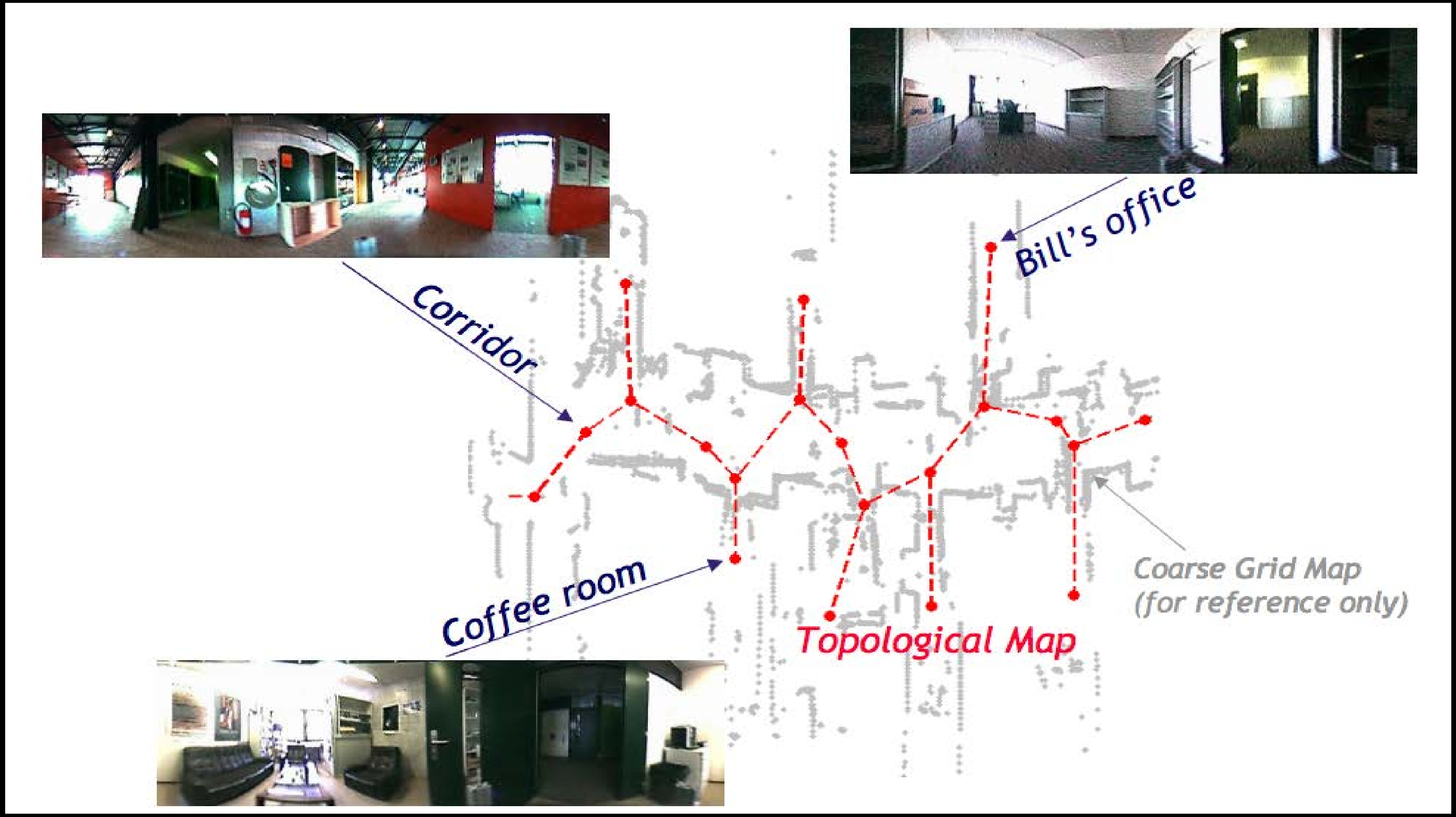


Navigation and Path Planning

- 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

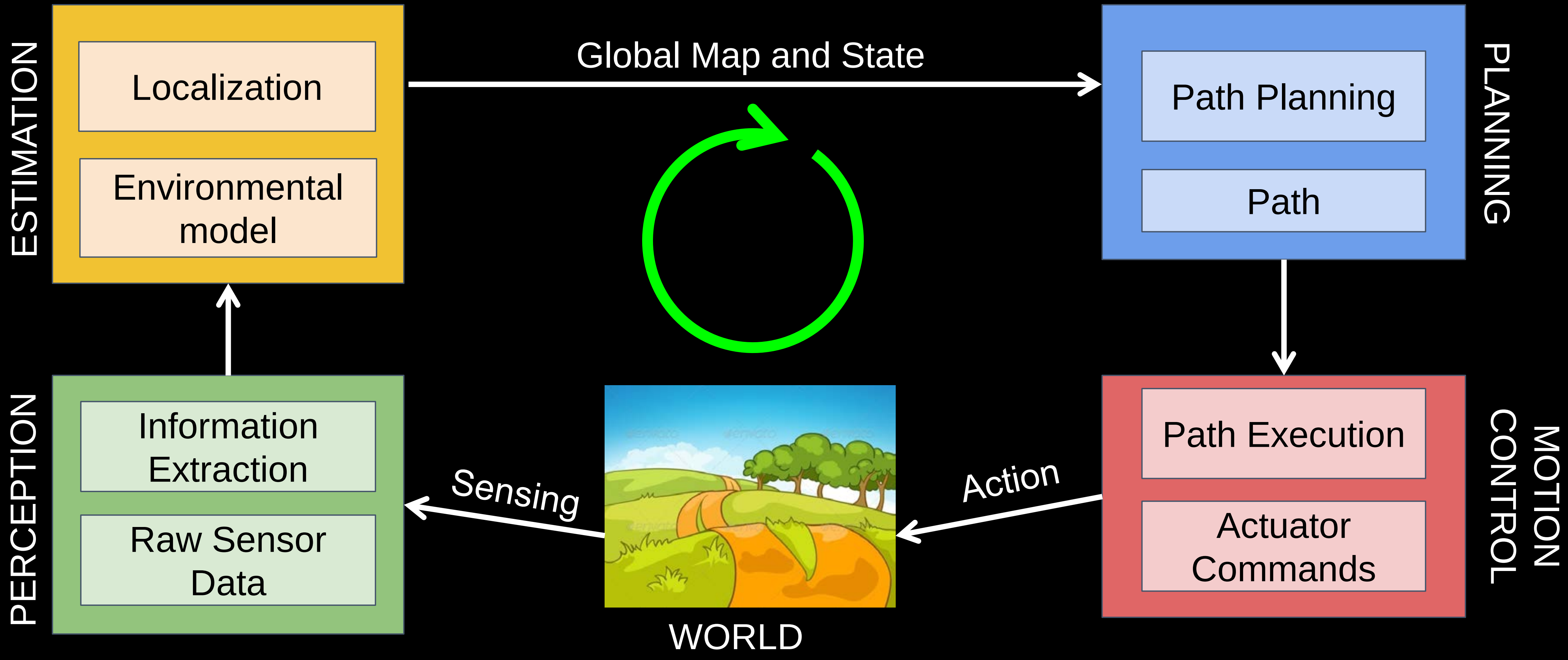
- **Problem:** Find the path in the workspace from an initial location to a goal location, while avoiding collisions
- **Assumption:** There exists a good map of the environment for navigation



- 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)

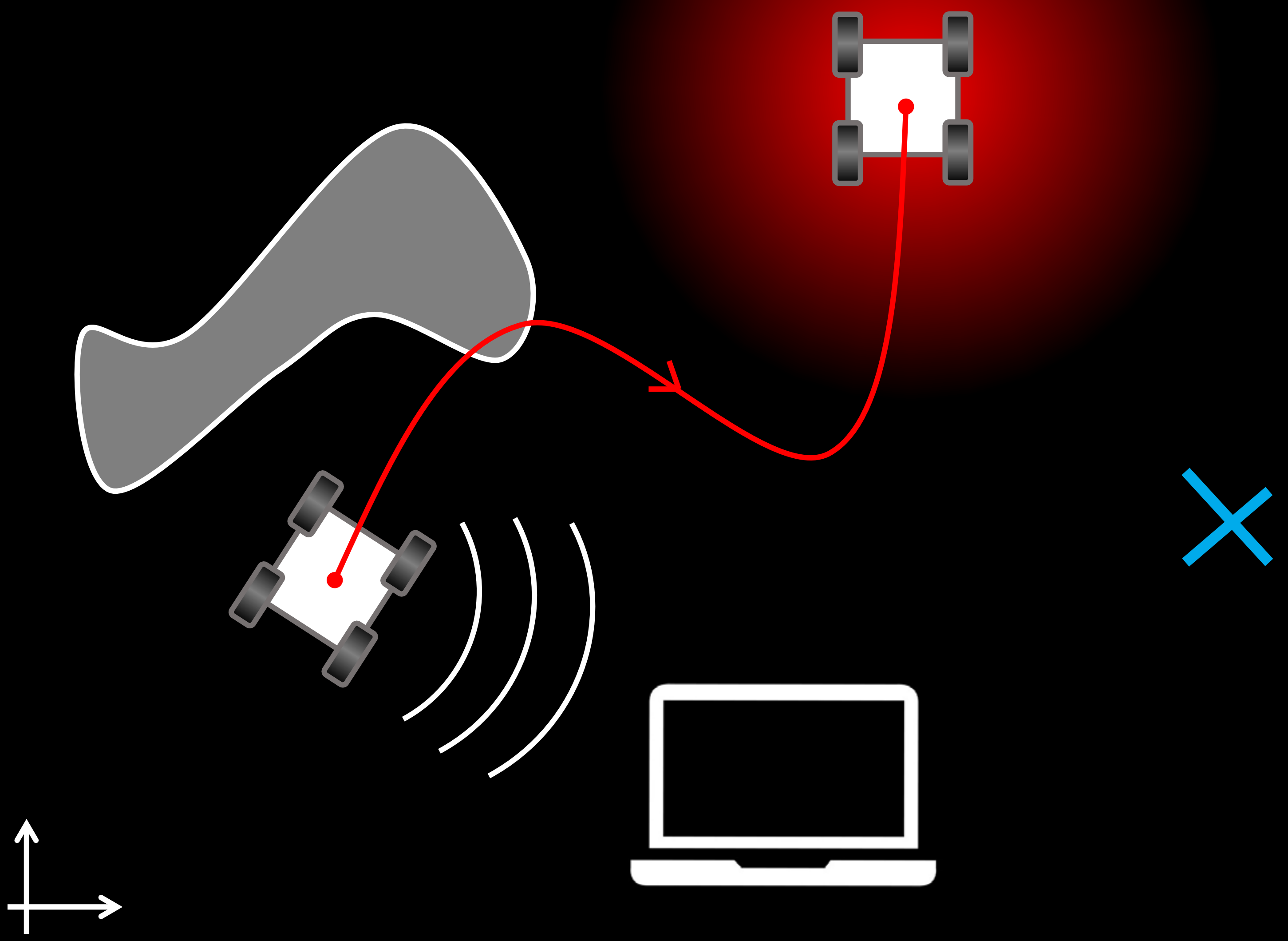
Navigation and Path Planning

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



Outline of the next module on Navigation

- Local planners
- Global localization and planning
  - Configuration space
  - 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
  - Can be based on a local map
  - Implemented as a separate task most of the times
  - Runs at a much faster rate than the global planning

  - BUG Algorithms
  - Vector Field Histogram (VFH)
  - Dynamic Window Approach (DWA)

Wagner, ITS 2015

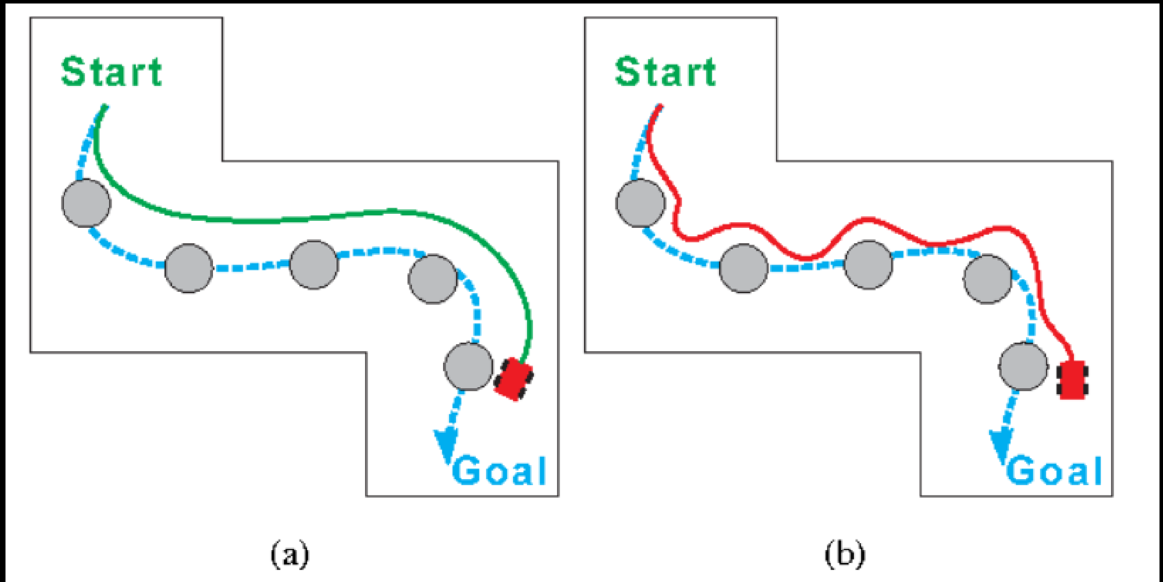
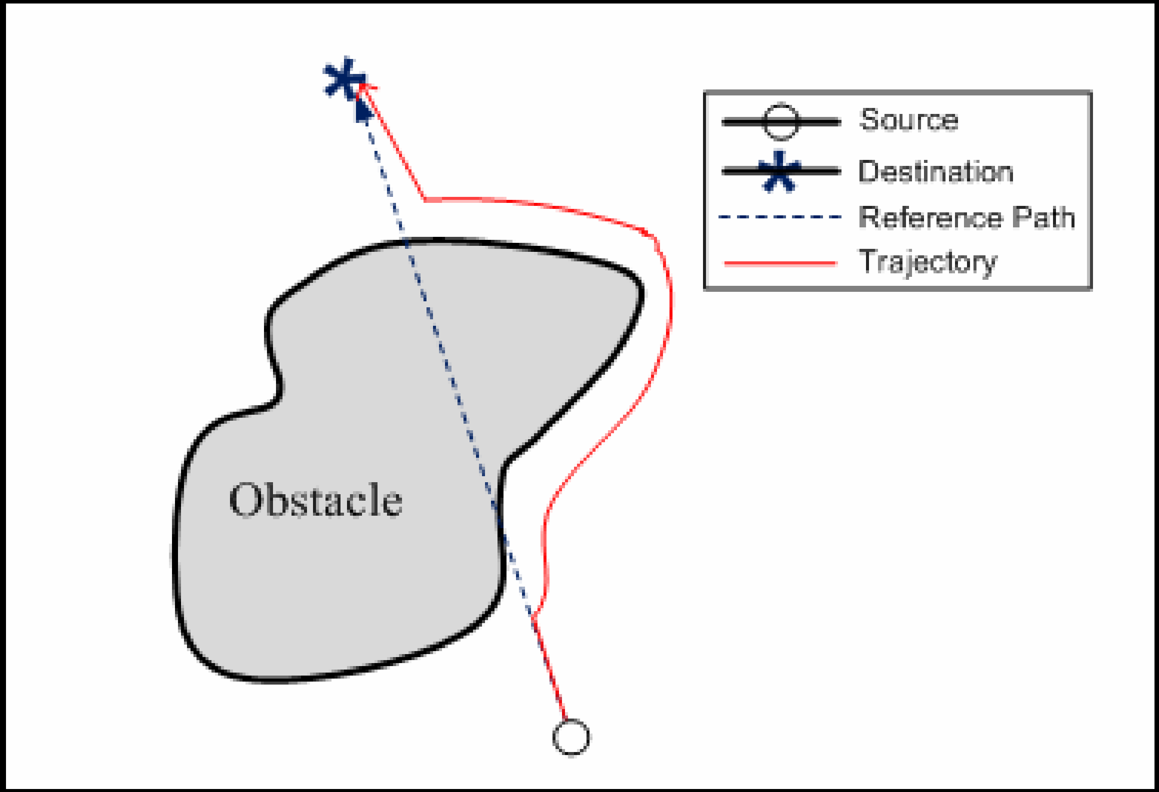


Fig. 1. Dashed blue spline is global path: a) Green spline is ideal local path; b) Red spline is actual local path

Bug Algorithms

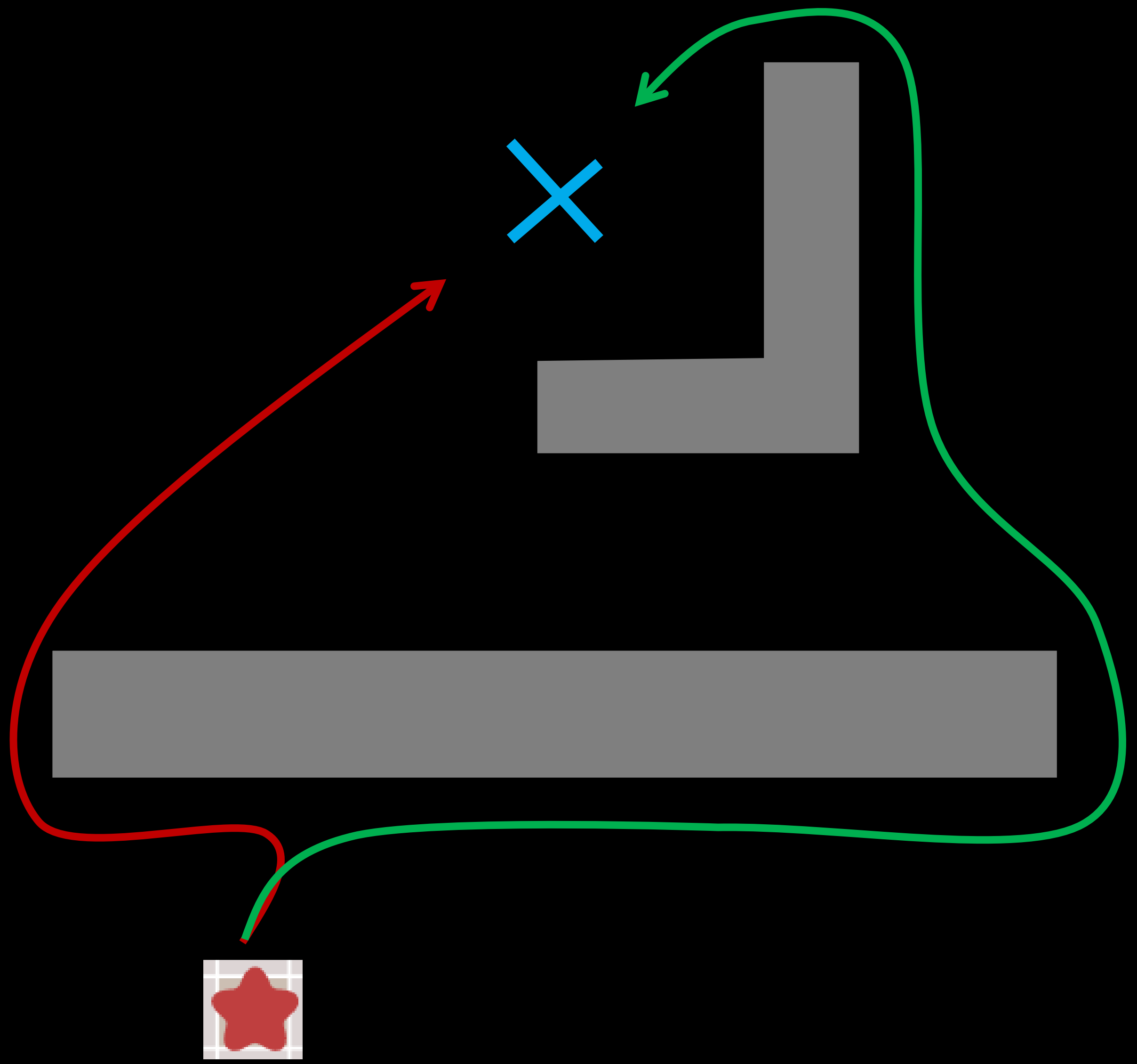
- Uses local knowledge, and the direction and distance to the goal
- Basic idea
  - Follow the contour of obstacles until you see the goal
  - State 1: Seek goal
  - State 2: follow wall
- Different variants: Bug0, Bug1, Bug2
- Advantages
  - Super simple
  - No global map
  - Completeness
- Disadvantages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



Bug 0

Sensor Assumptions

- Direction to the goal
- Detect walls



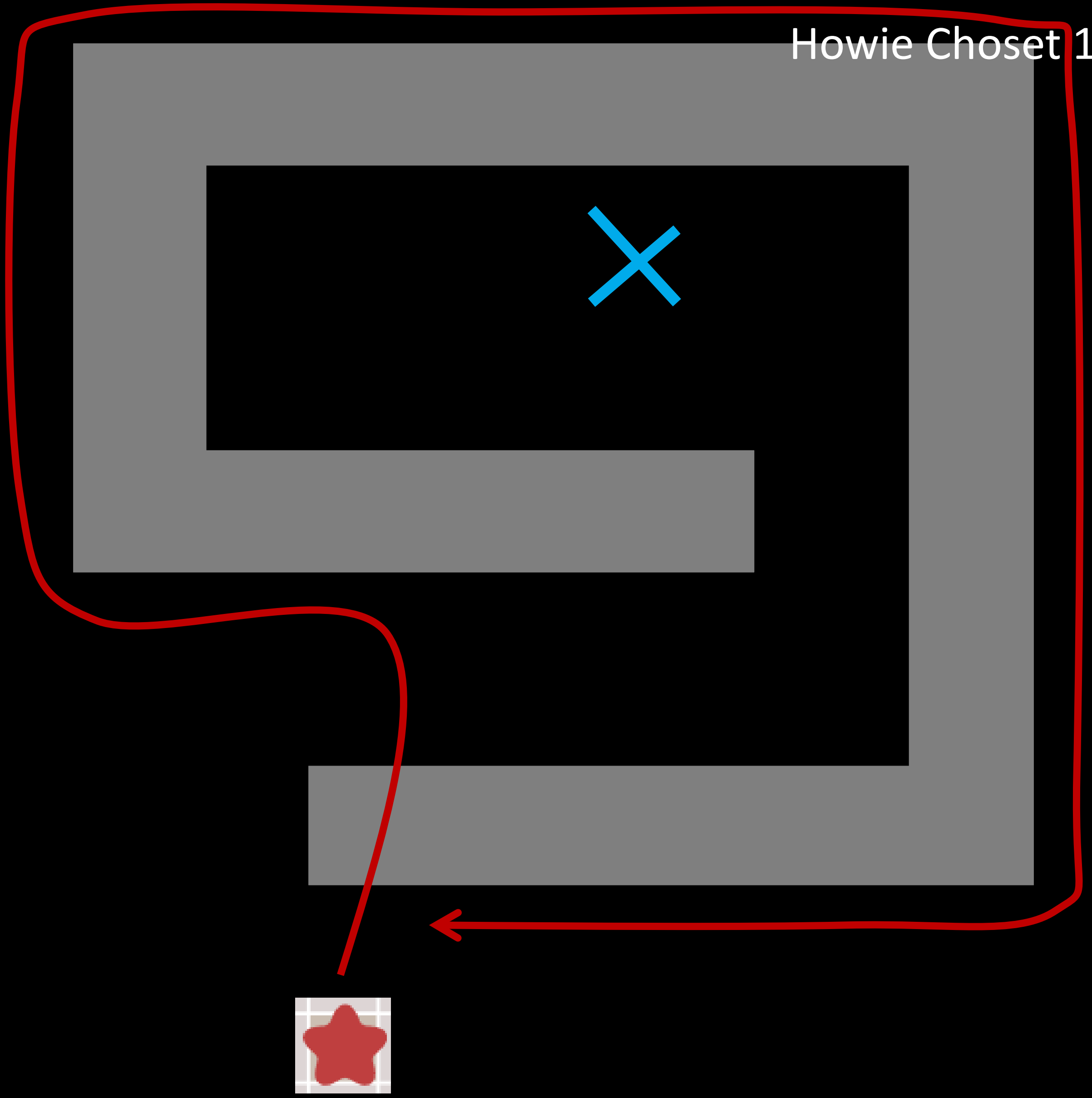
Algorithm

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



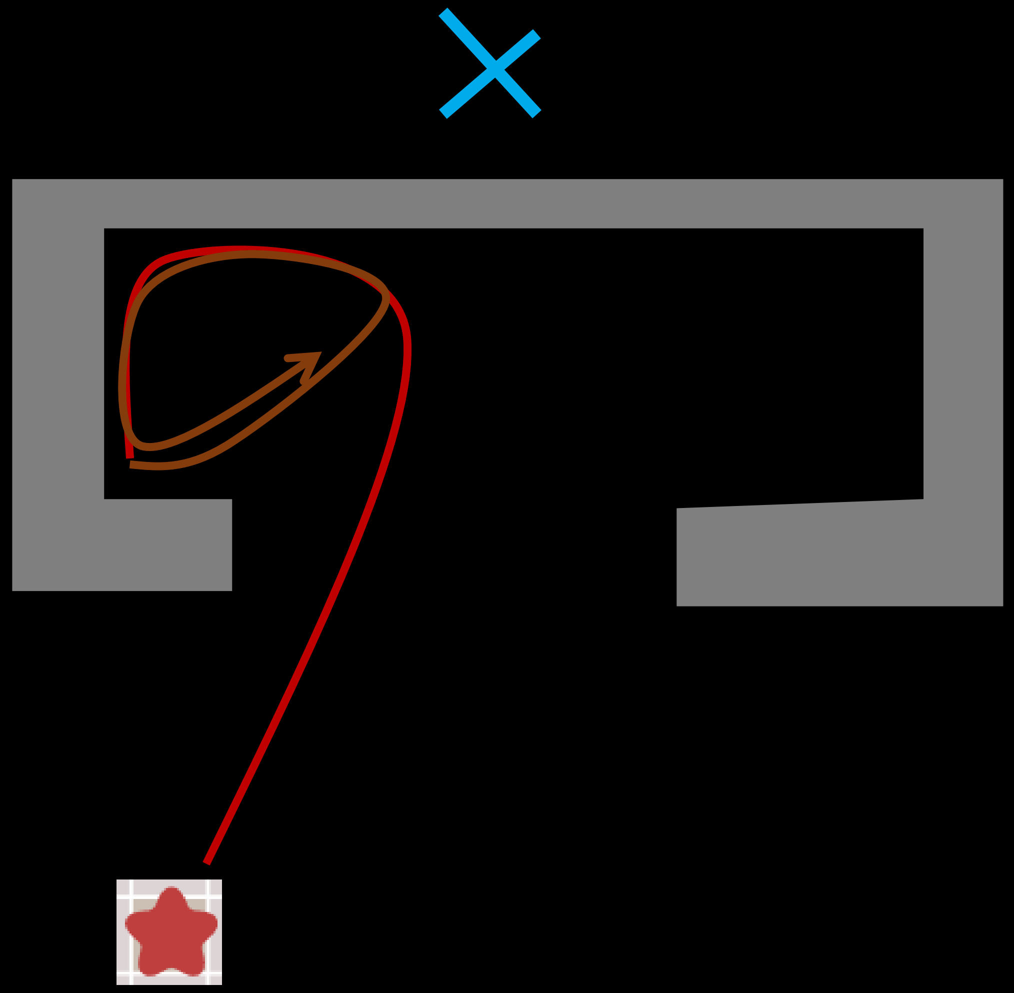
Algorithm

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

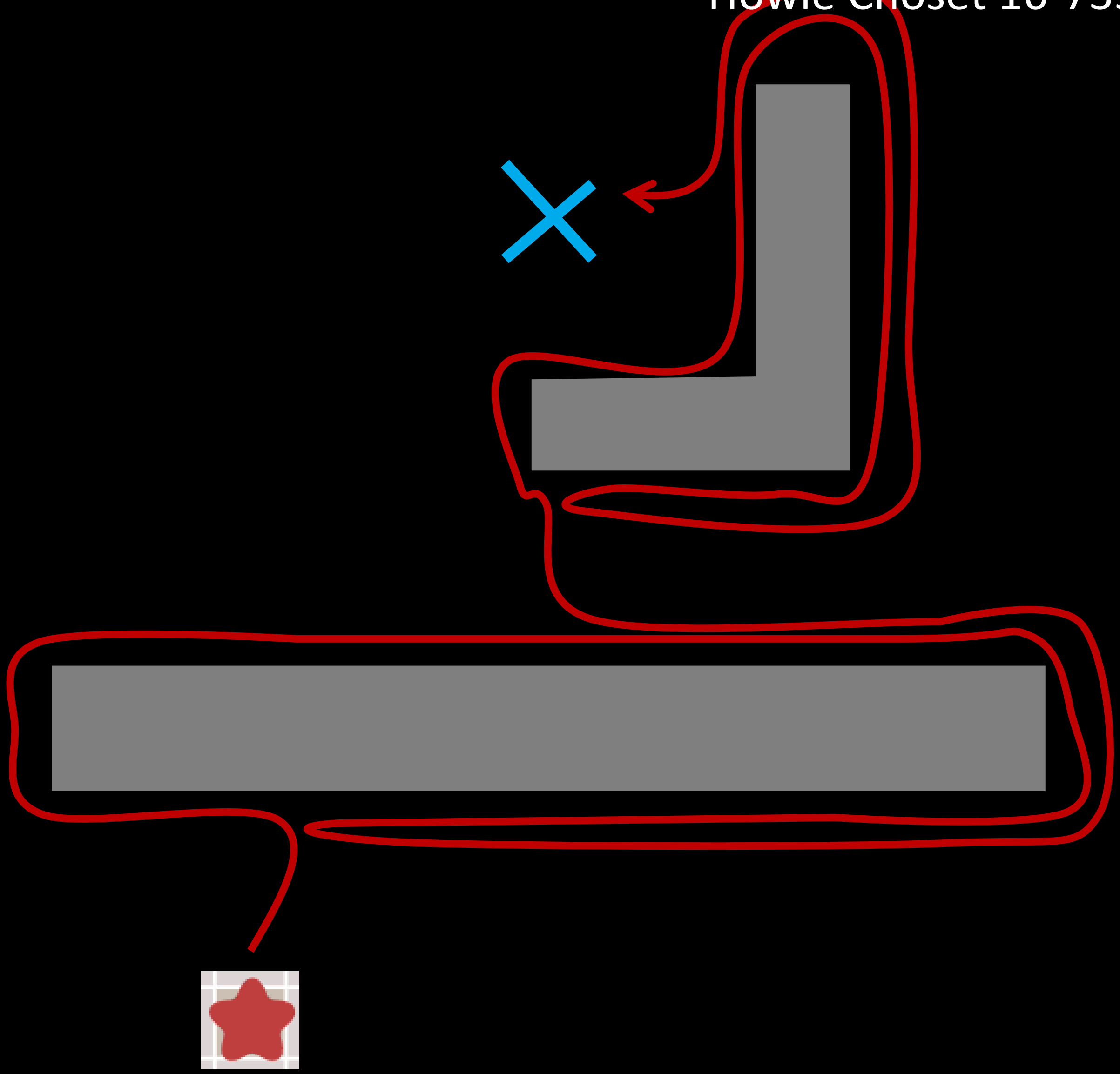
Bug 1

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



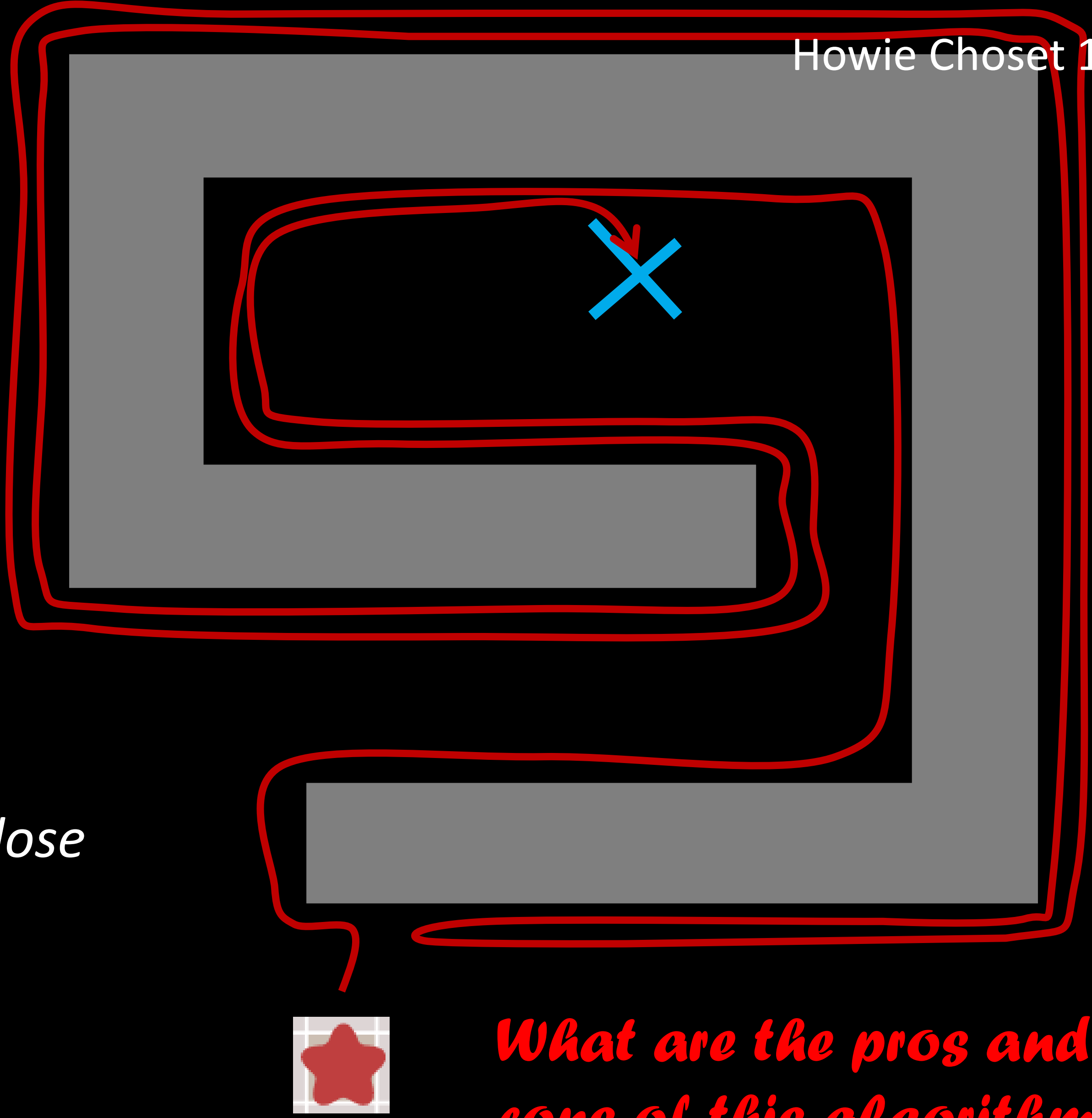
Bug 1

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



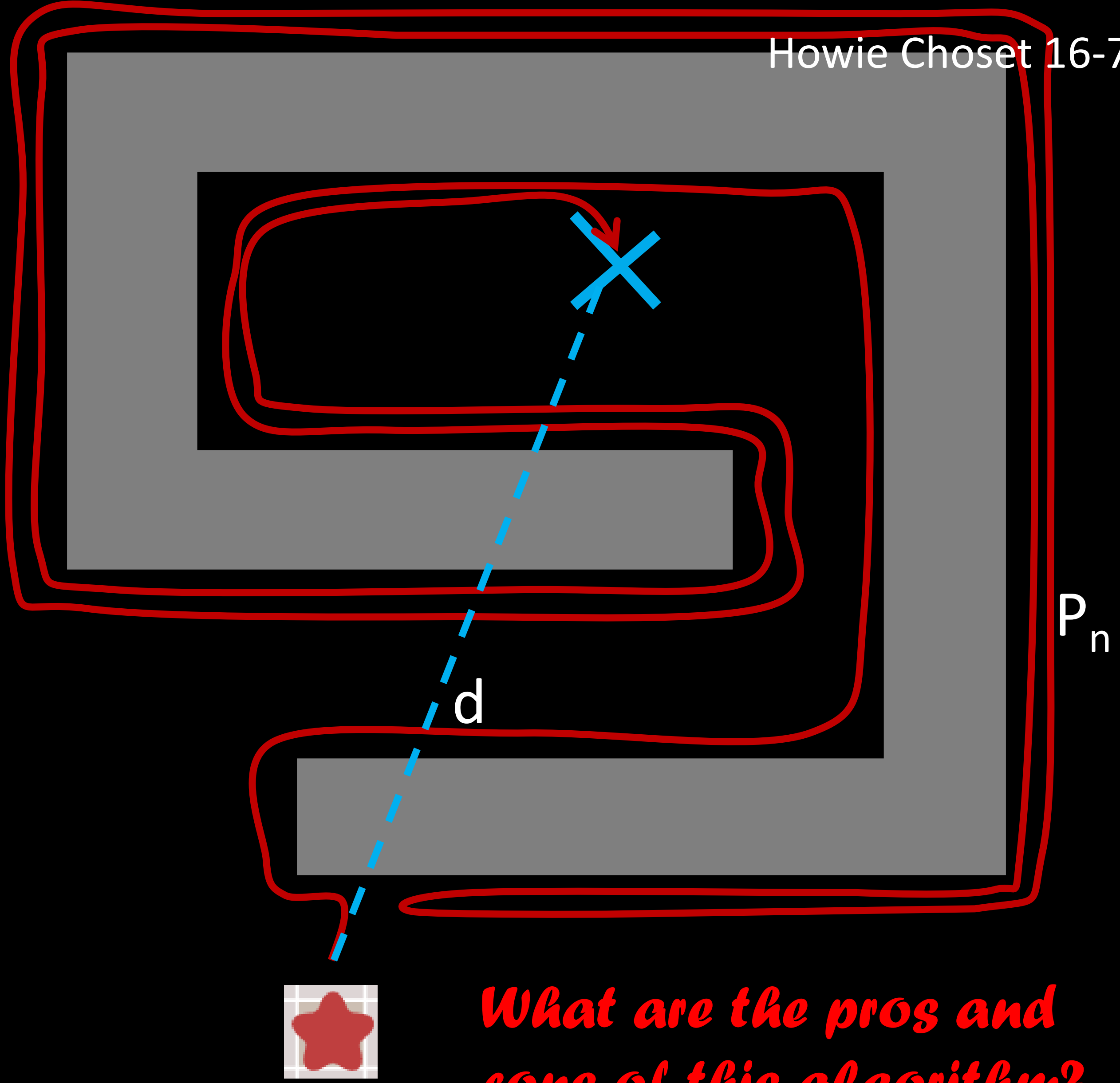
What are the pros and cons of this algorithm?

Bug 1 - formally

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry

- Lower bound traversal?
  - d
- Upper bound traversal?
  - d + 1.5 · Sum(P)

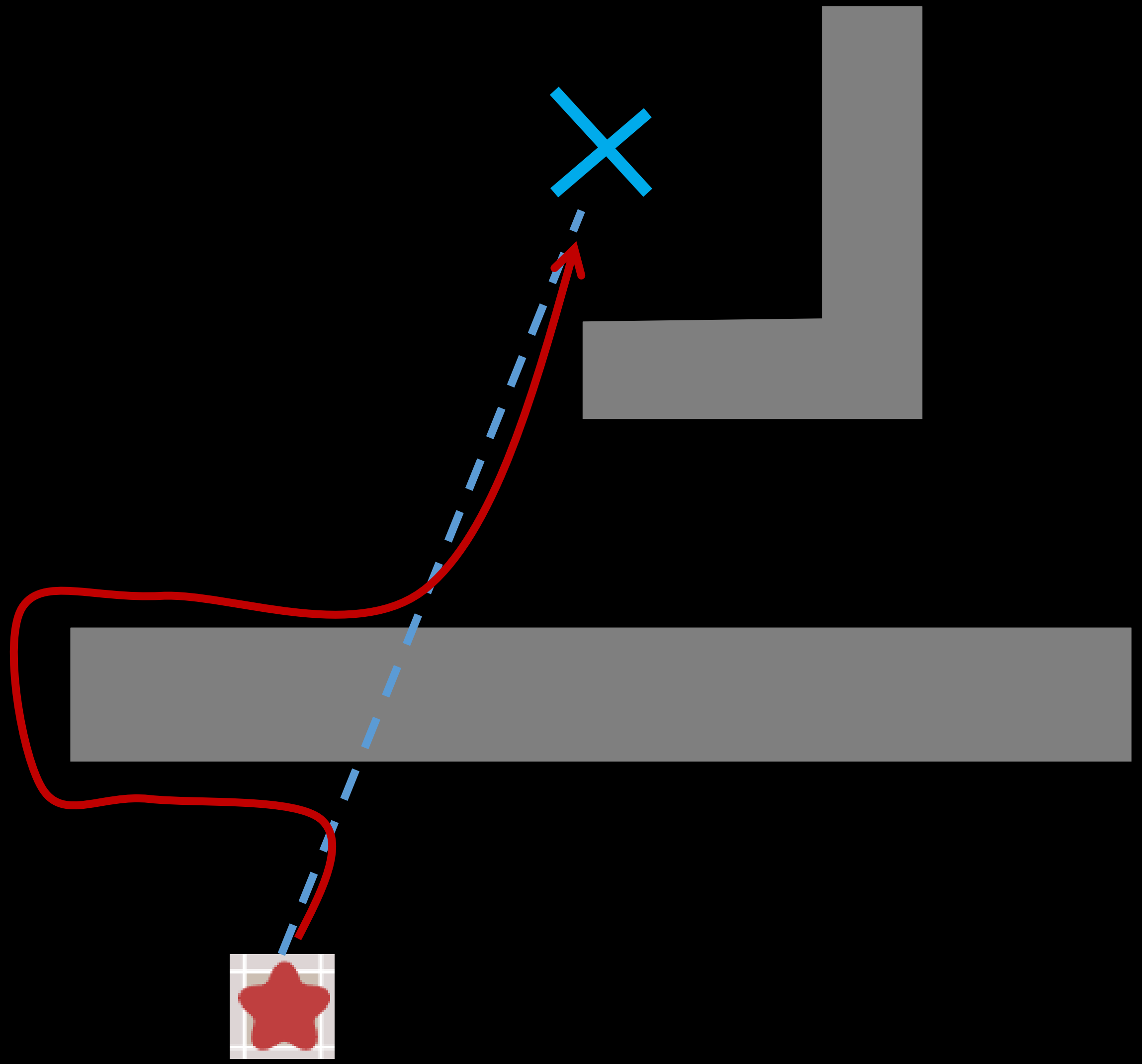


What are the pros and cons of this algorithm?

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

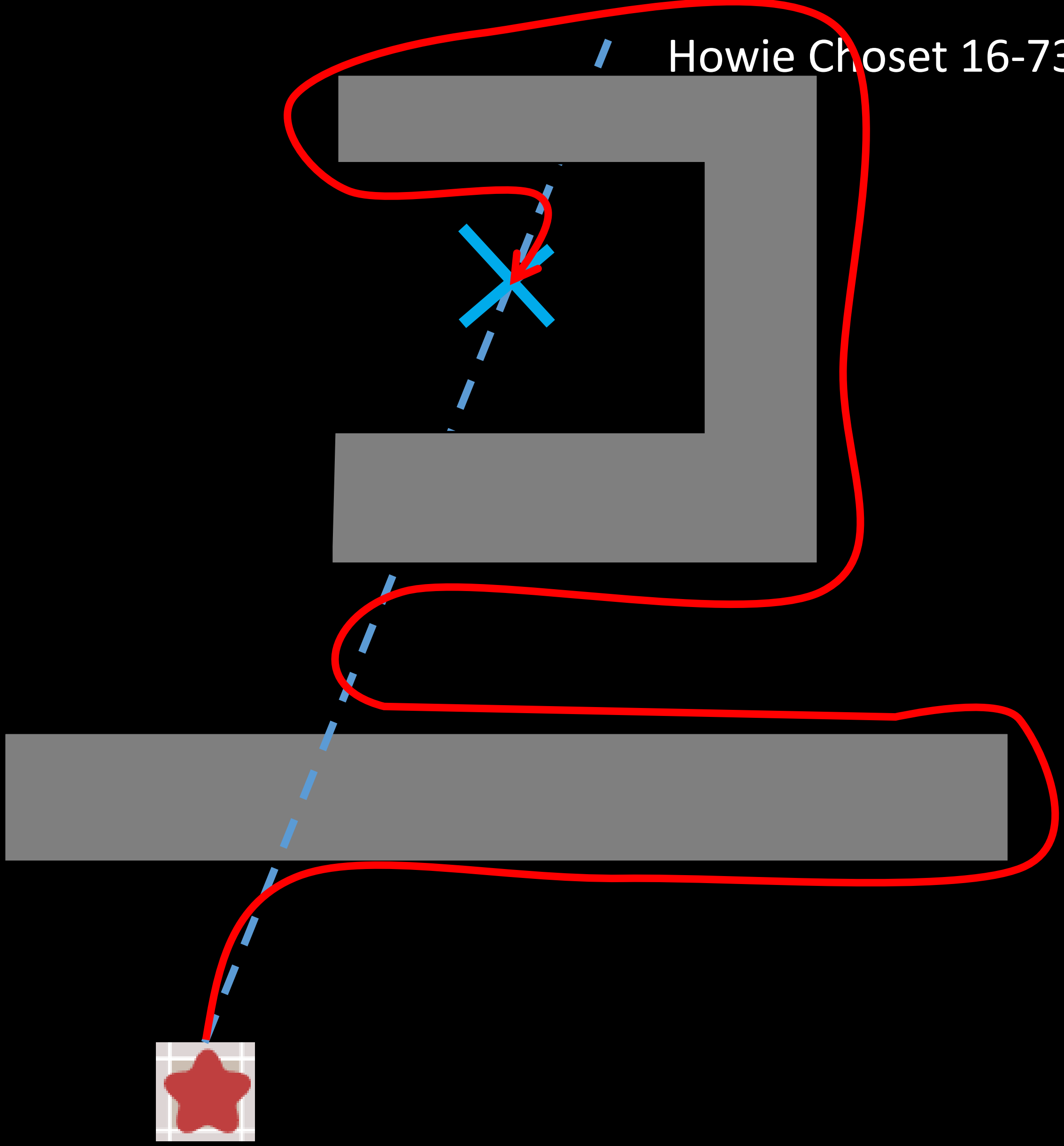
Bug 2

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



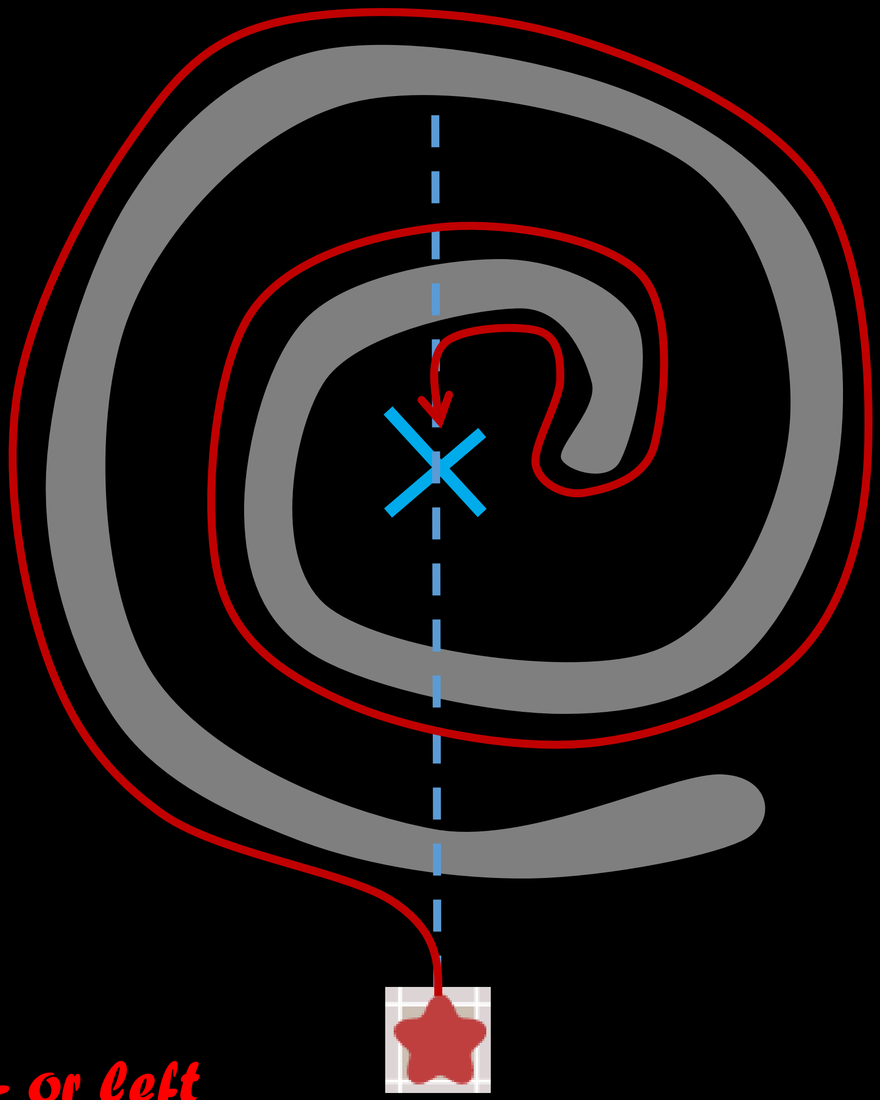
Bug 2

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?

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

Battle of the Bugs (1 vs 2)

Bug 1
Layout 1

Bug 2
Layout 1

Battle of the Bugs (1 vs 2)

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

Exhaustive Search

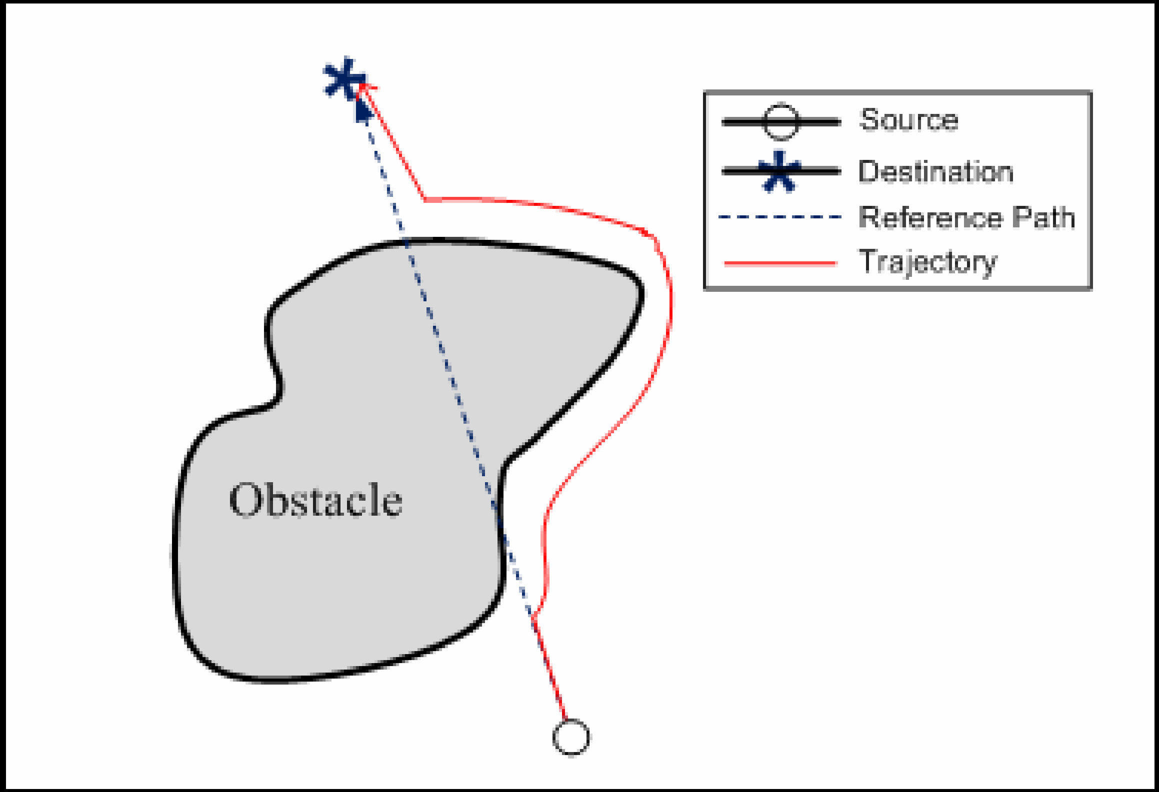
Bug 1
Layout 2

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

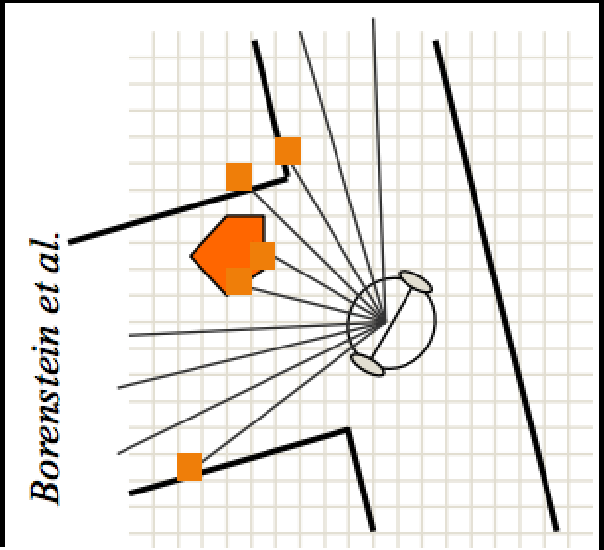


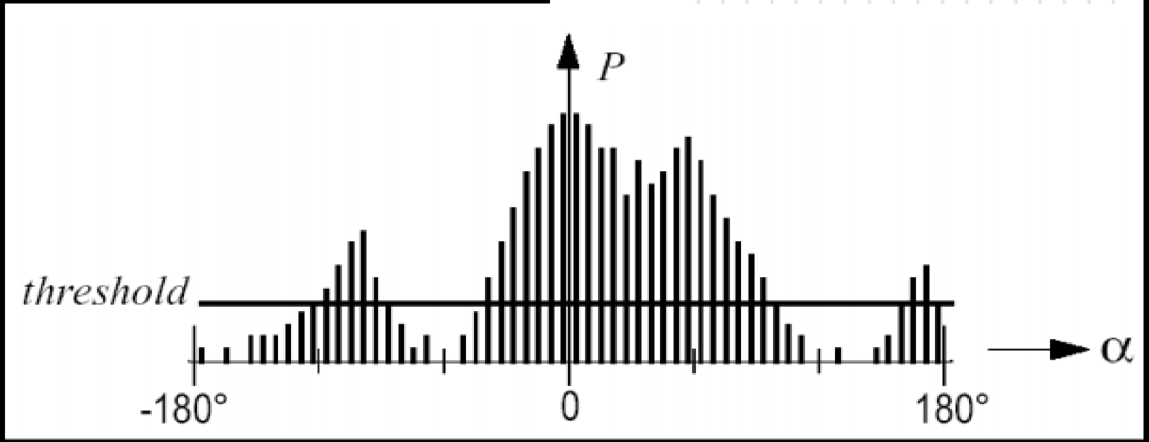
- The robot motion behavior is reactive
- Issues if the instantaneous sensor readings do not provide enough information or are noisy

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

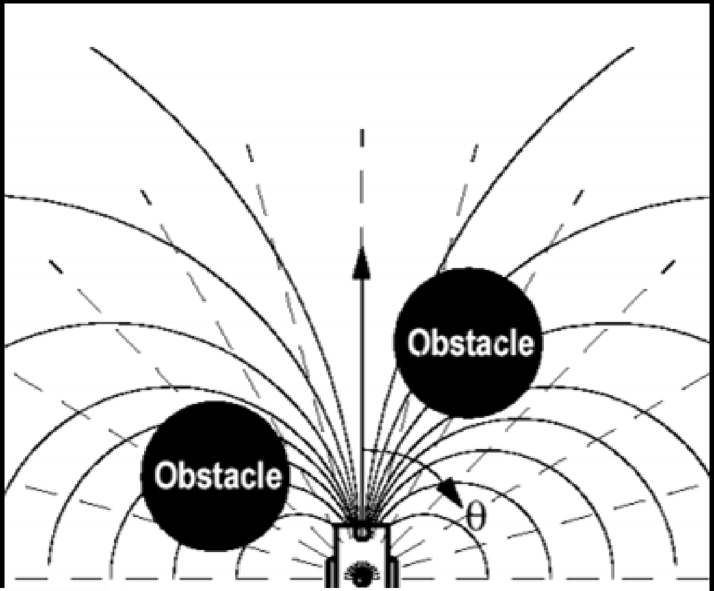


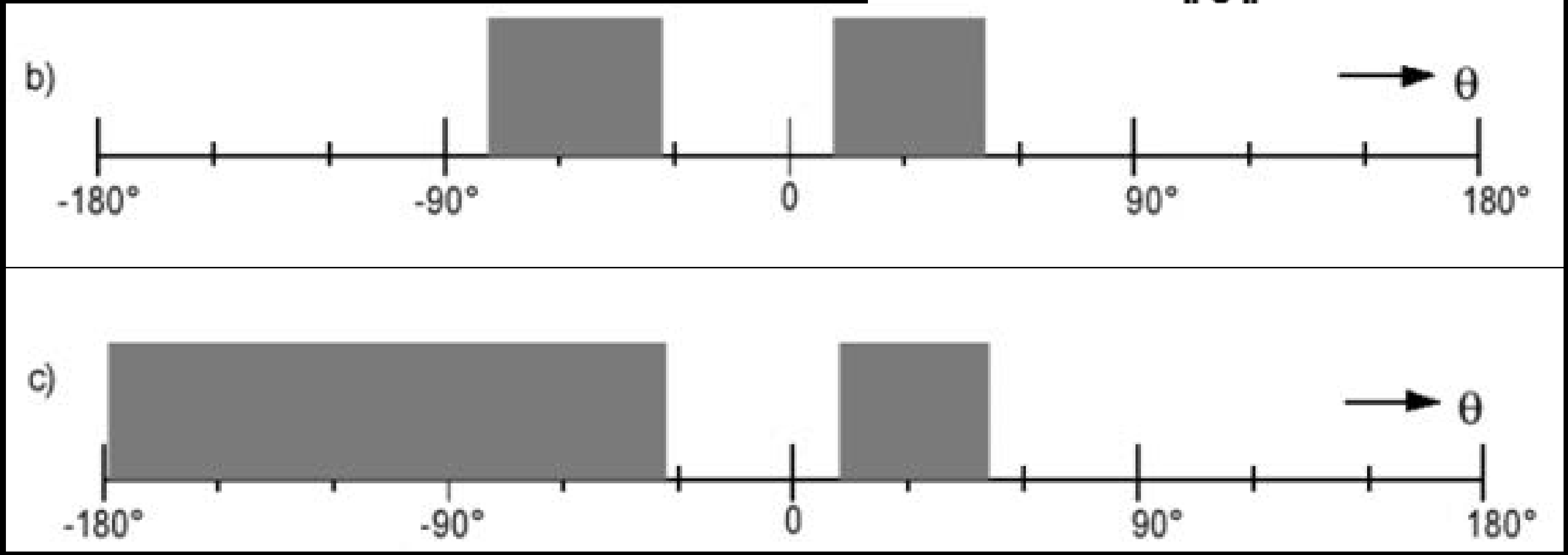


Vector Field Histograms

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: $G(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega))$

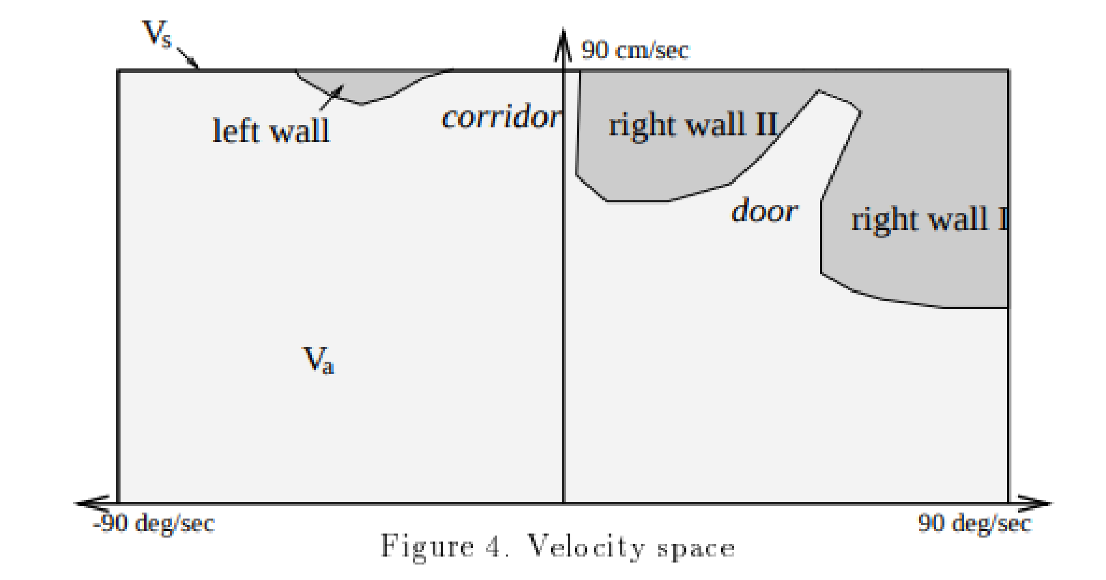
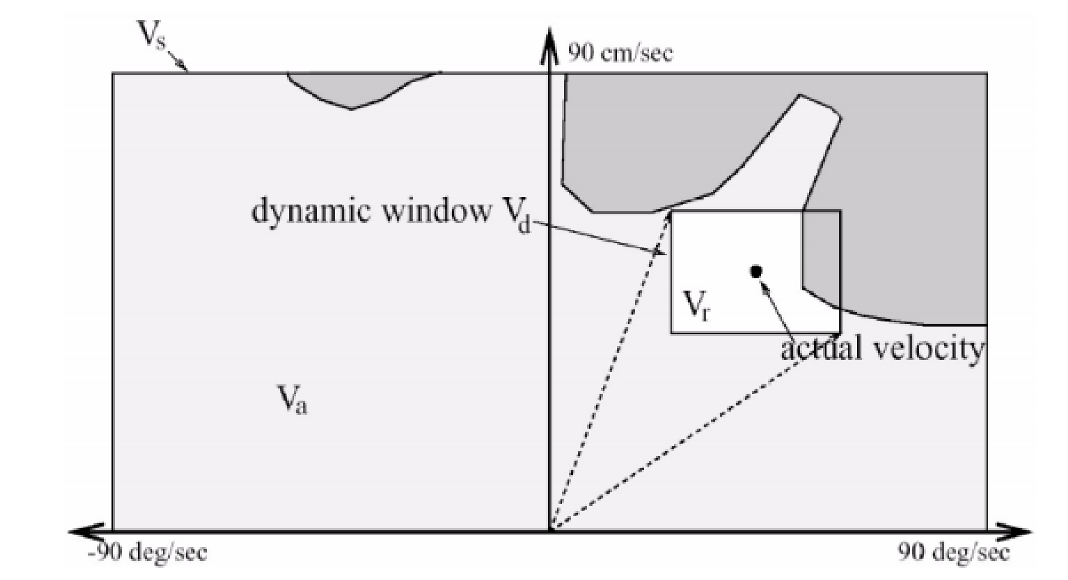


Figure 4. Velocity space

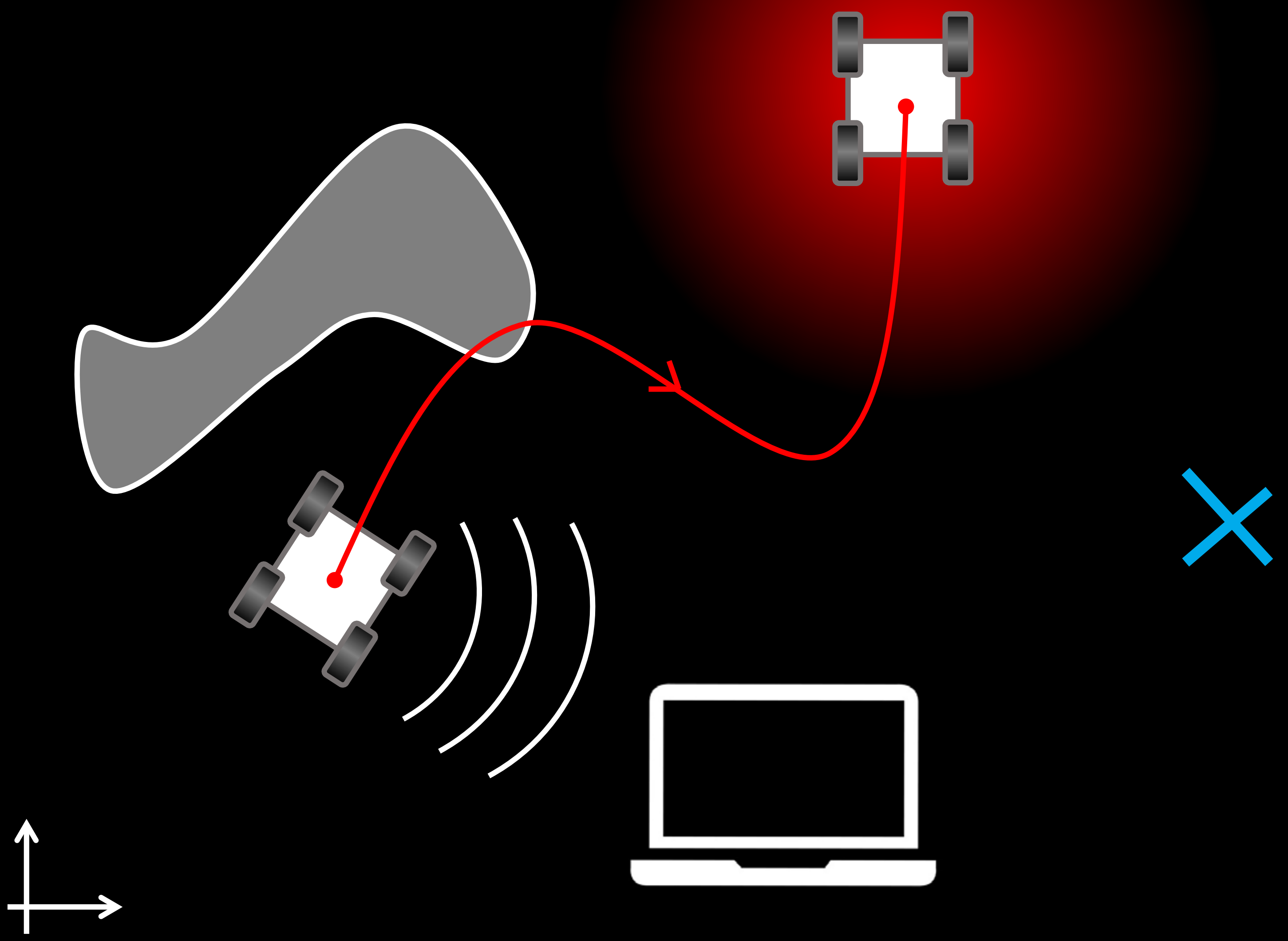


Local Planning Algorithms, Summary

- 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

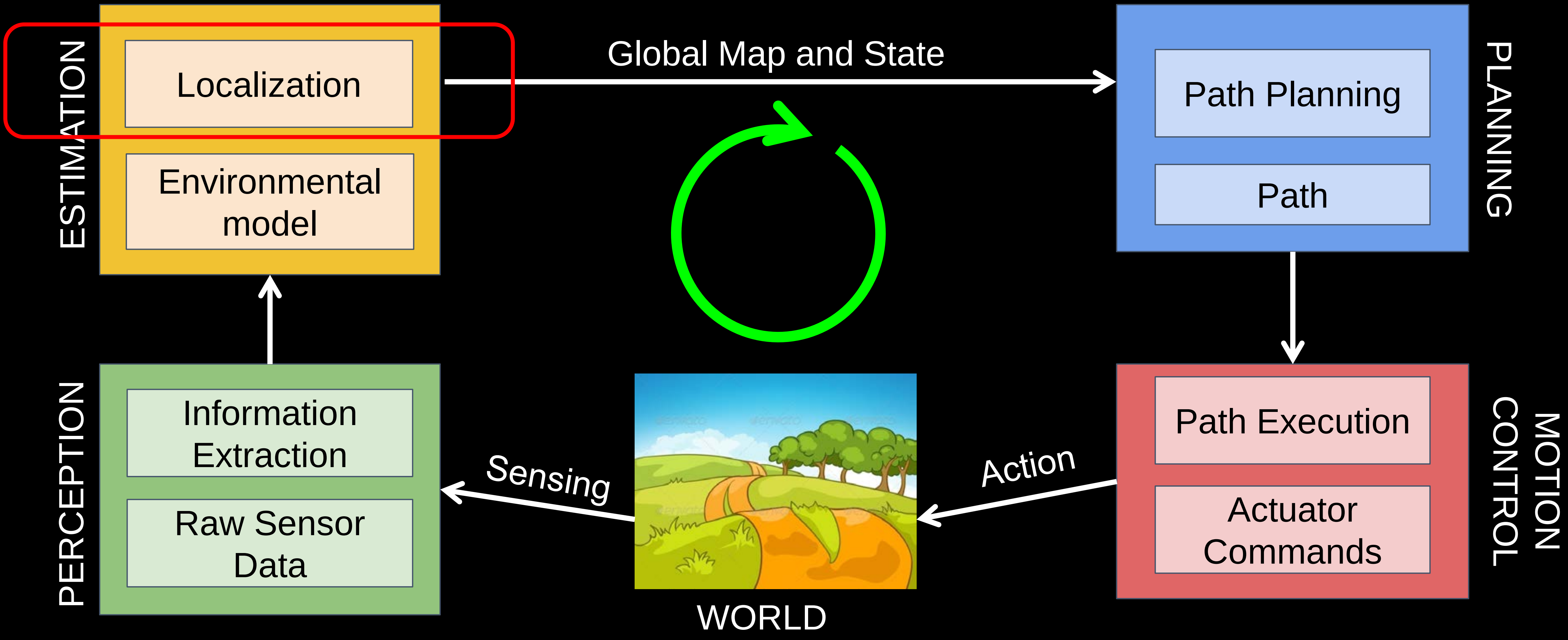
Outline of the next module on Navigation

- 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

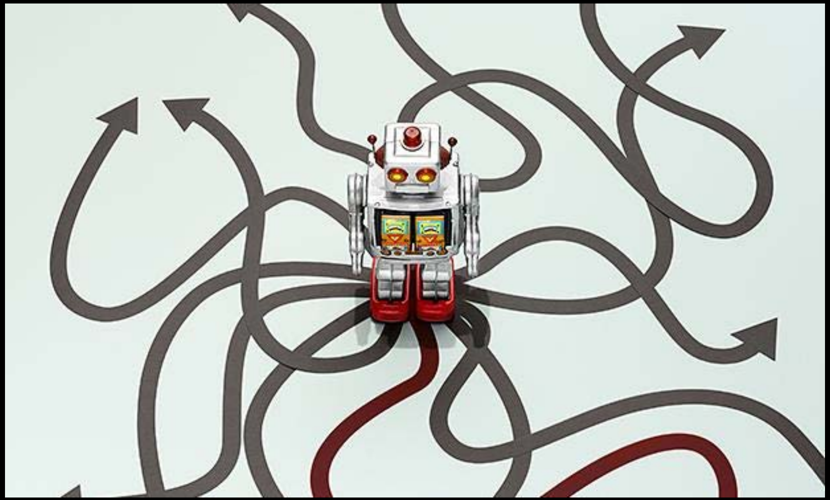
Position Tracking

- 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

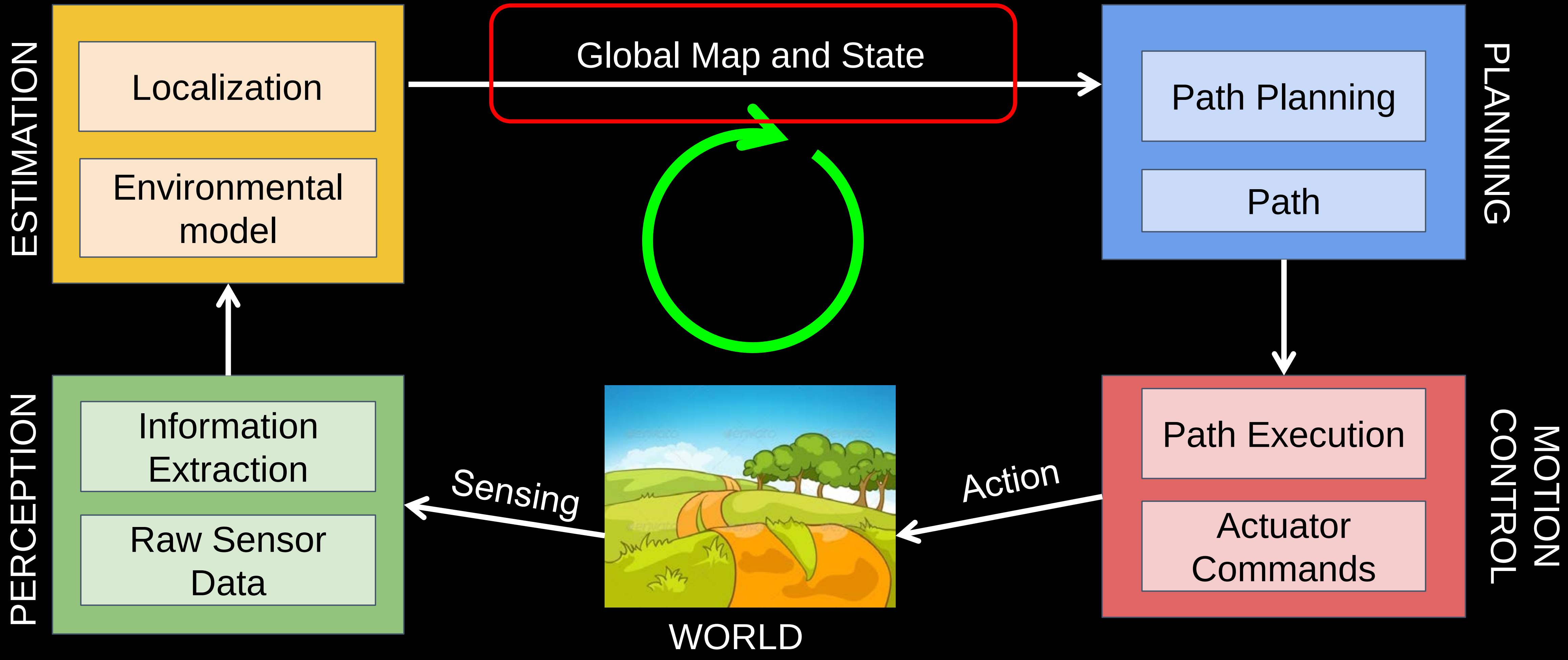
- Initial robot pose is unknown
- Need to estimate position from scratch
- A more difficult "global" problem, where you cannot assume boundedness in pose error

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

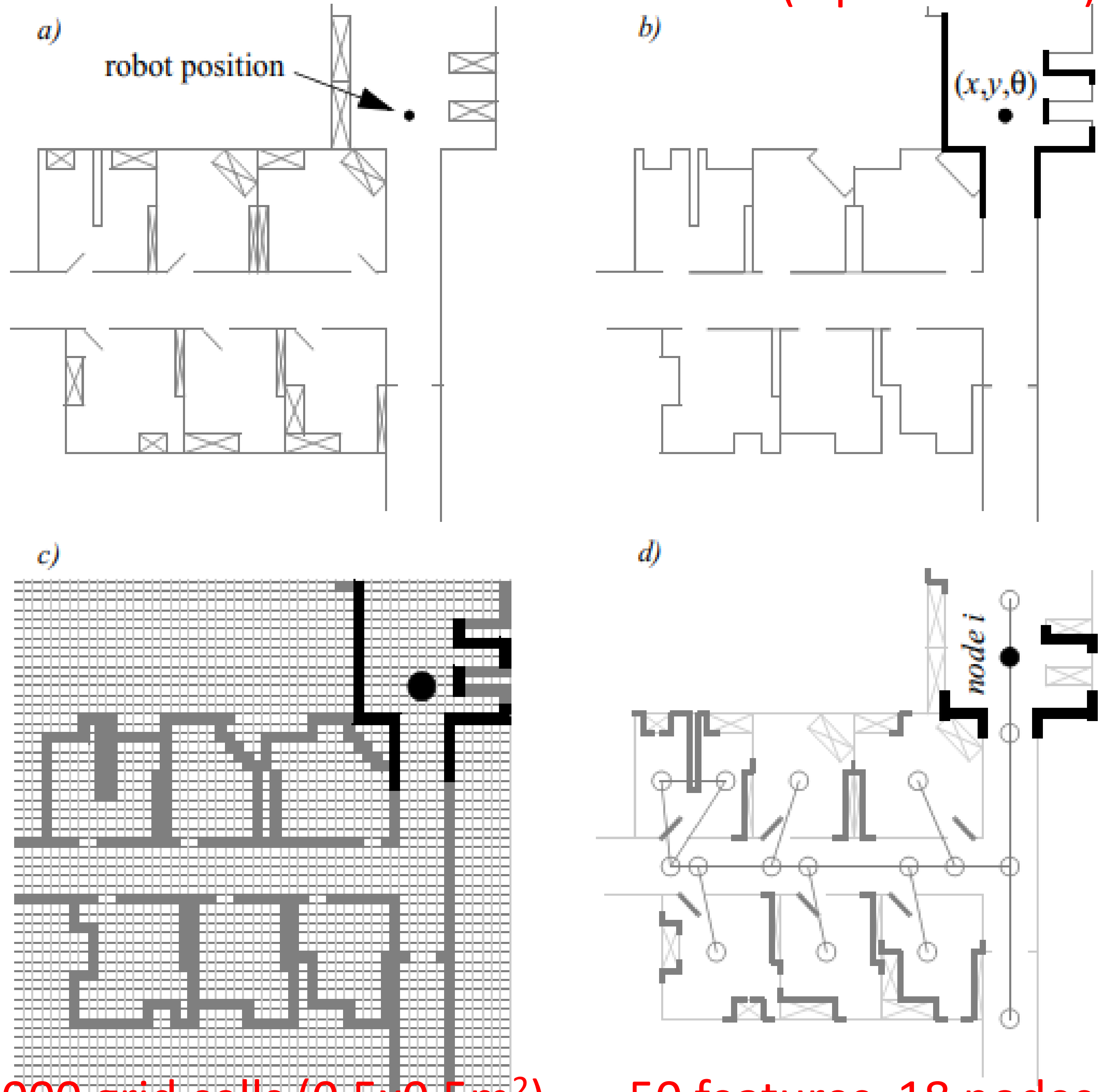
(c) occupancy grid-based map

(d) topological map

Important properties

- Memory allocation
- Computation
- Robot pose

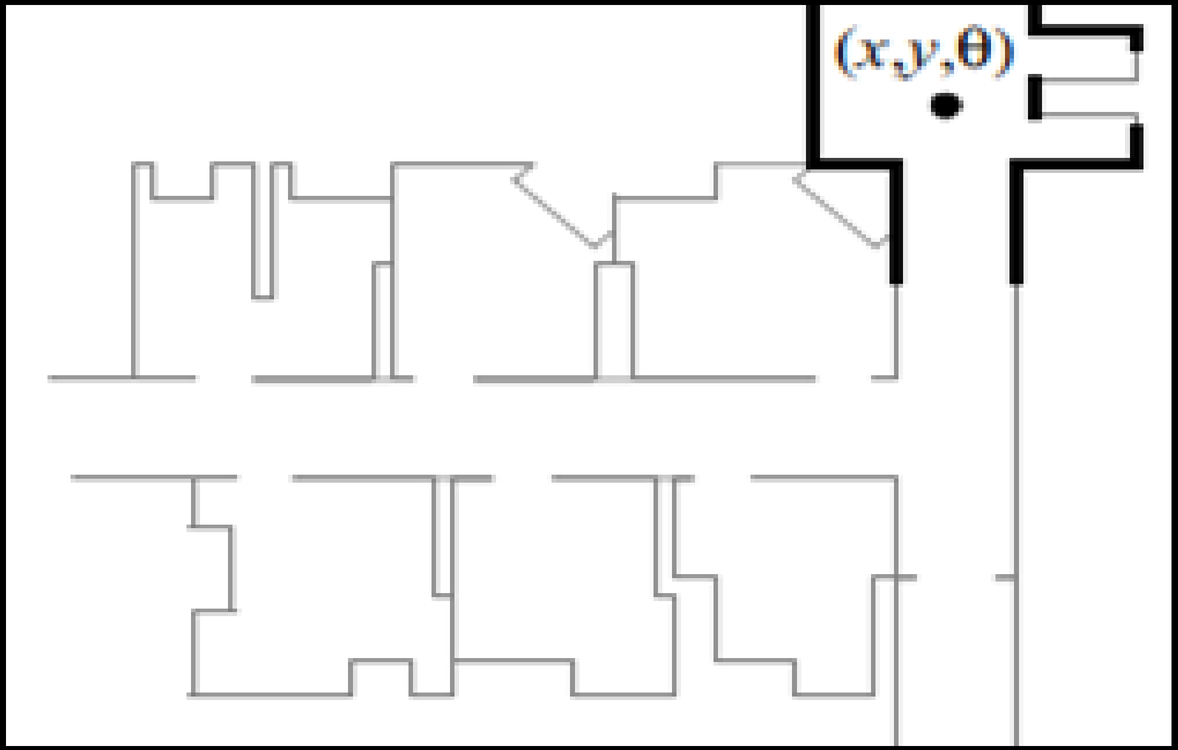
100 lines (2 parameters)

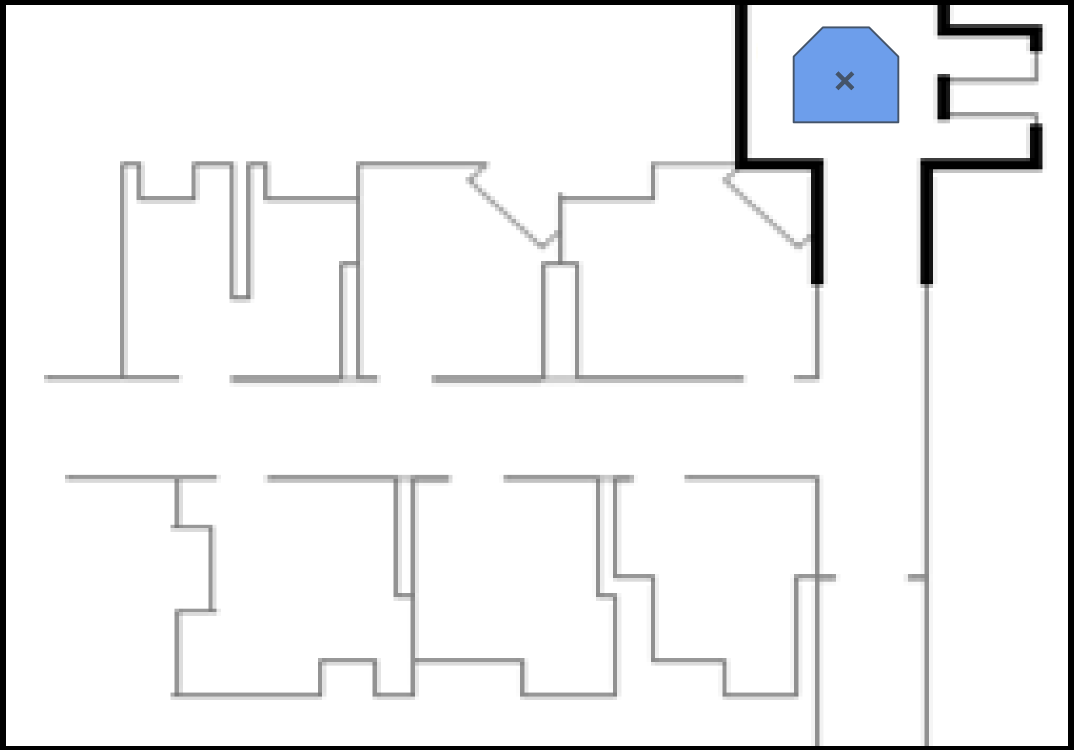


3000 grid cells ($0.5x0.5m^2$)

50 features, 18 nodes

What if the robot is not a point?

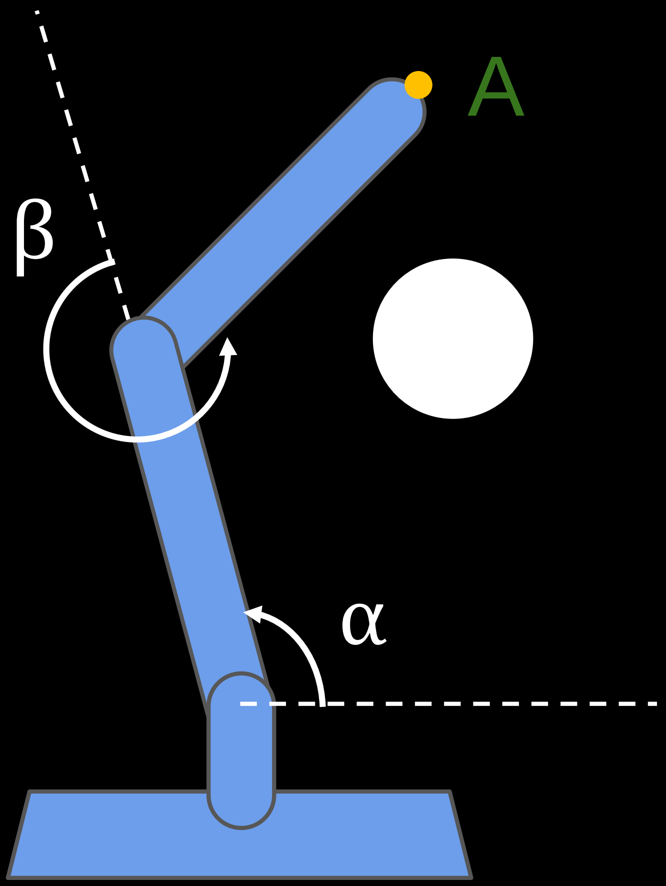


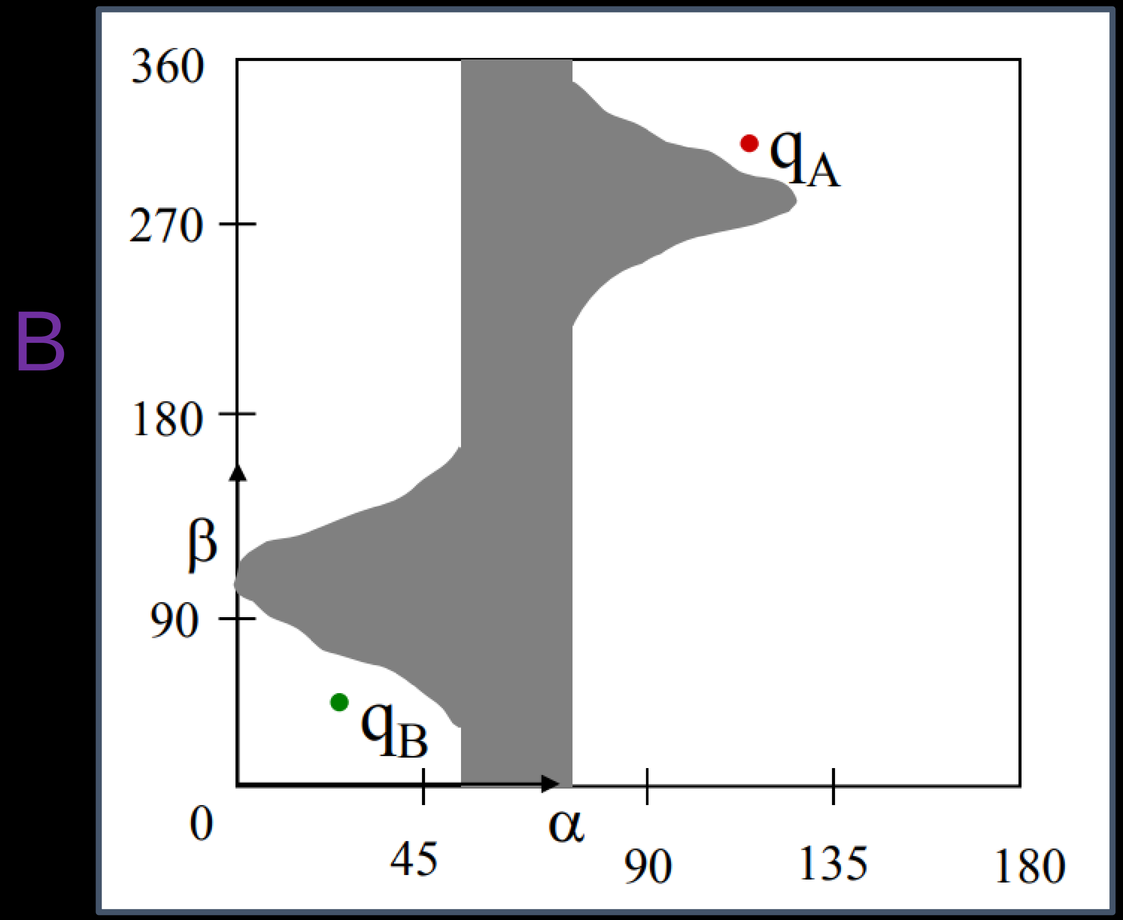


Configuration Space

- 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

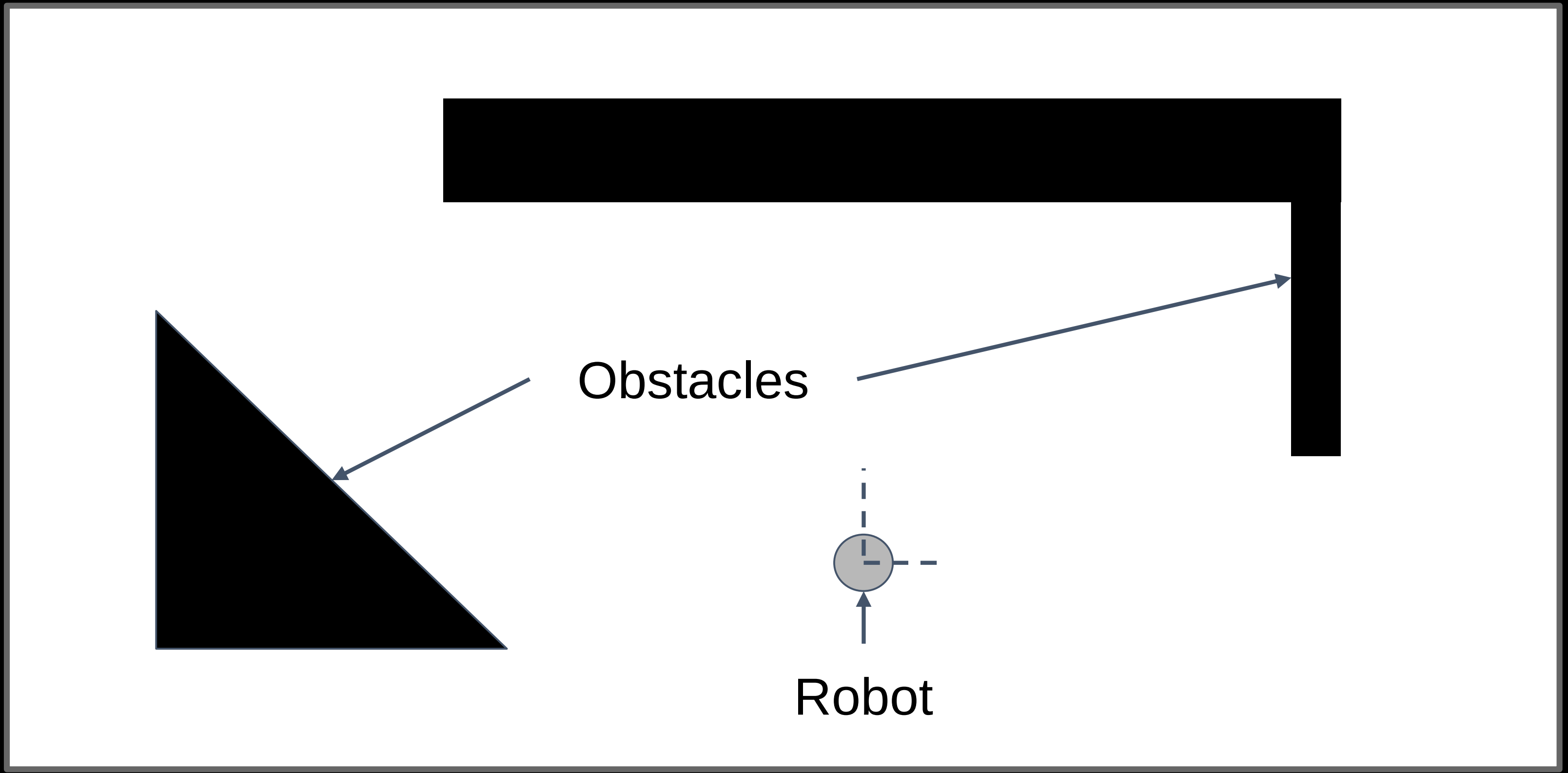




Configuration Space

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

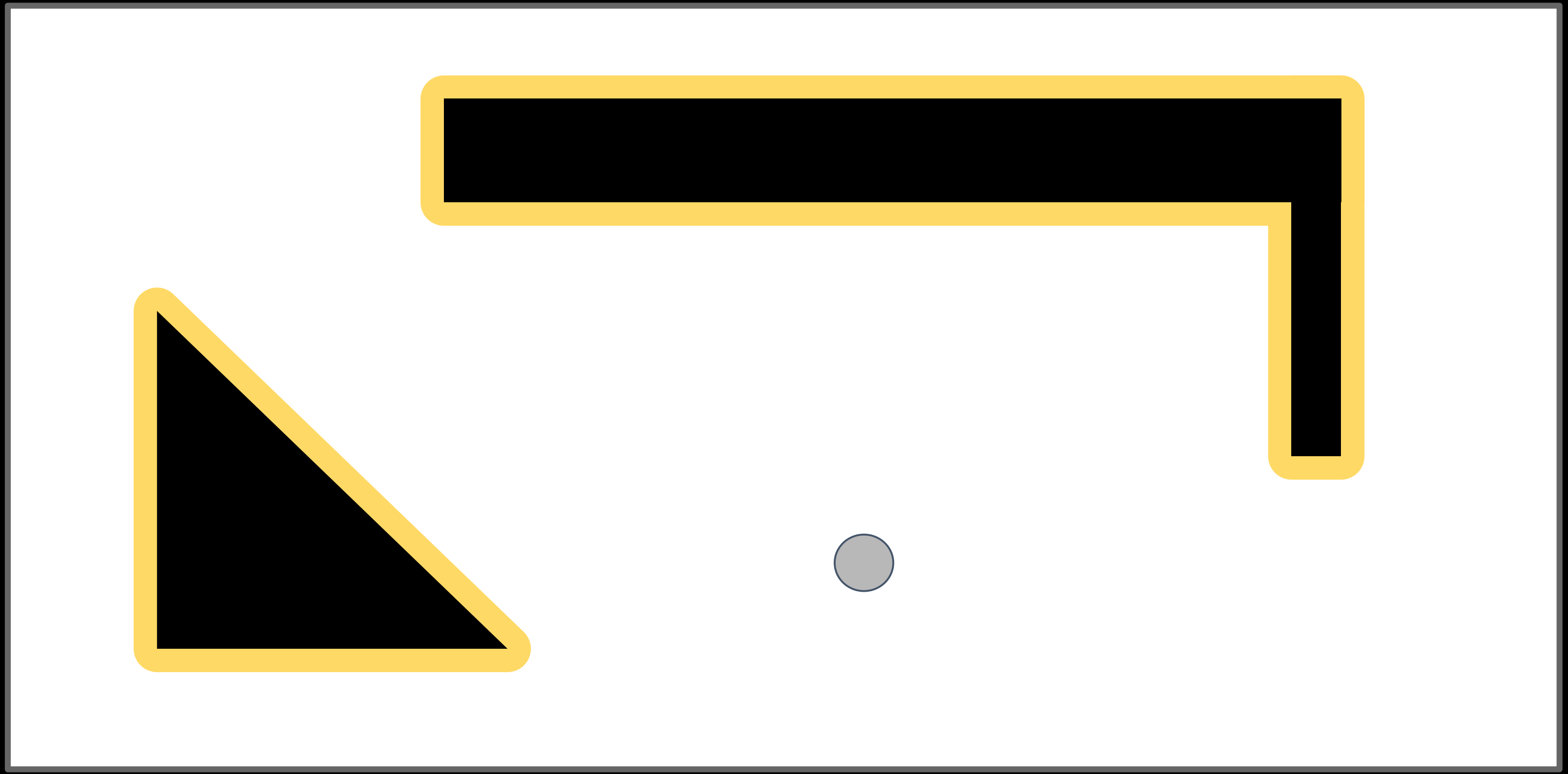
Ex 2: Circular root in 2D world



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 root 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 root in 2D world

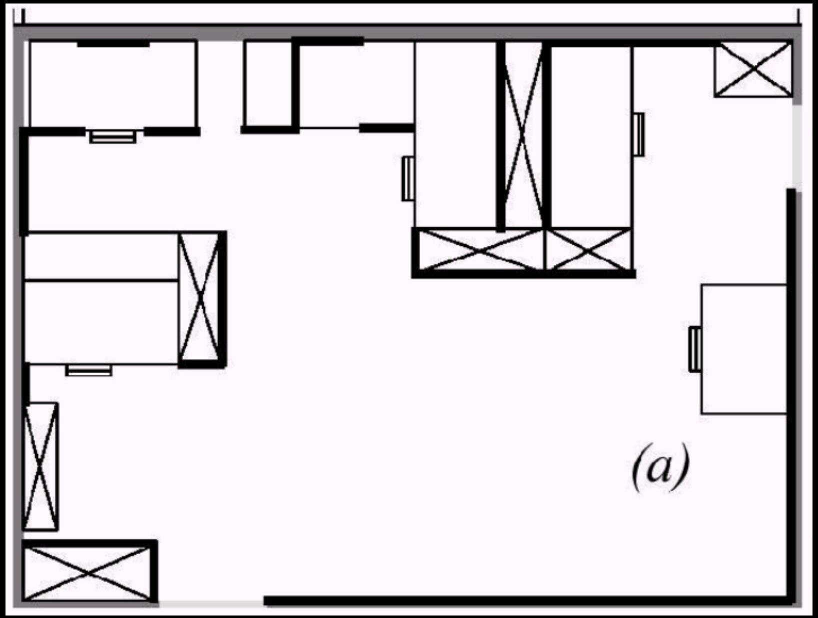
Map Representation Considerations

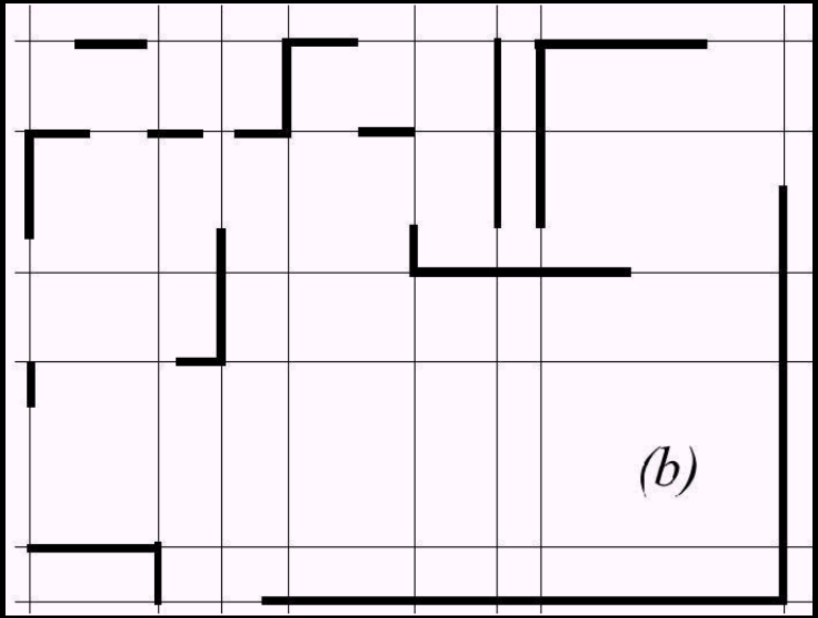
"Vivek's three rules for map representation"

- 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

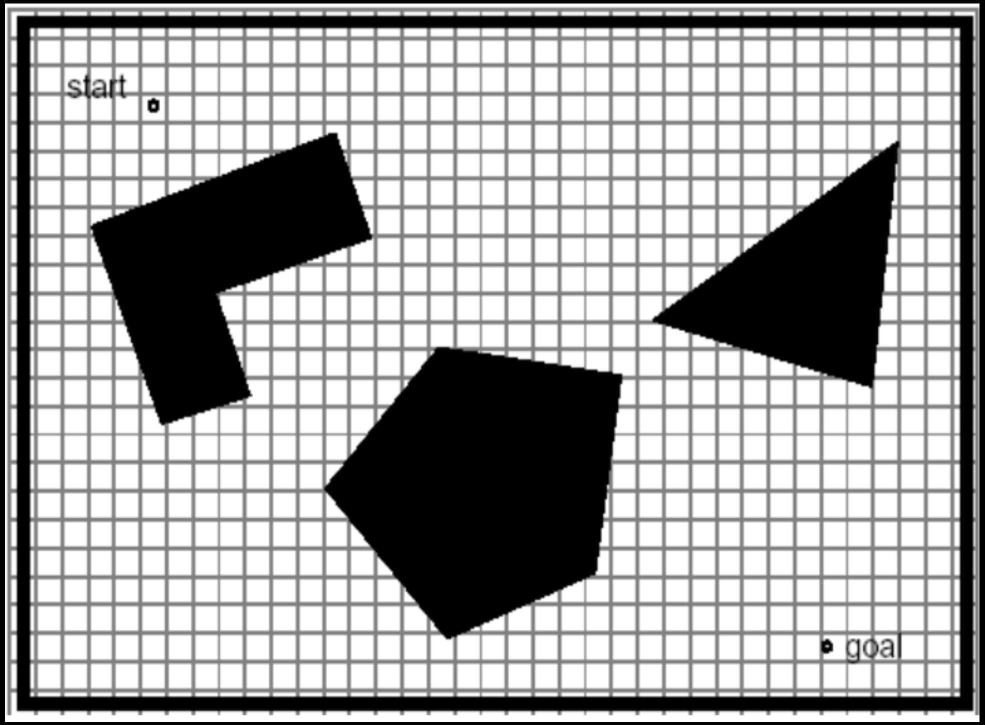
- Exact decomposition of the environment
- Used mainly in 2D representations
- Closed-world assumption
- Storage proportional to object density
- Example: Continuous line representations
  - Using range finders, we can extract lines/line segments in the environment

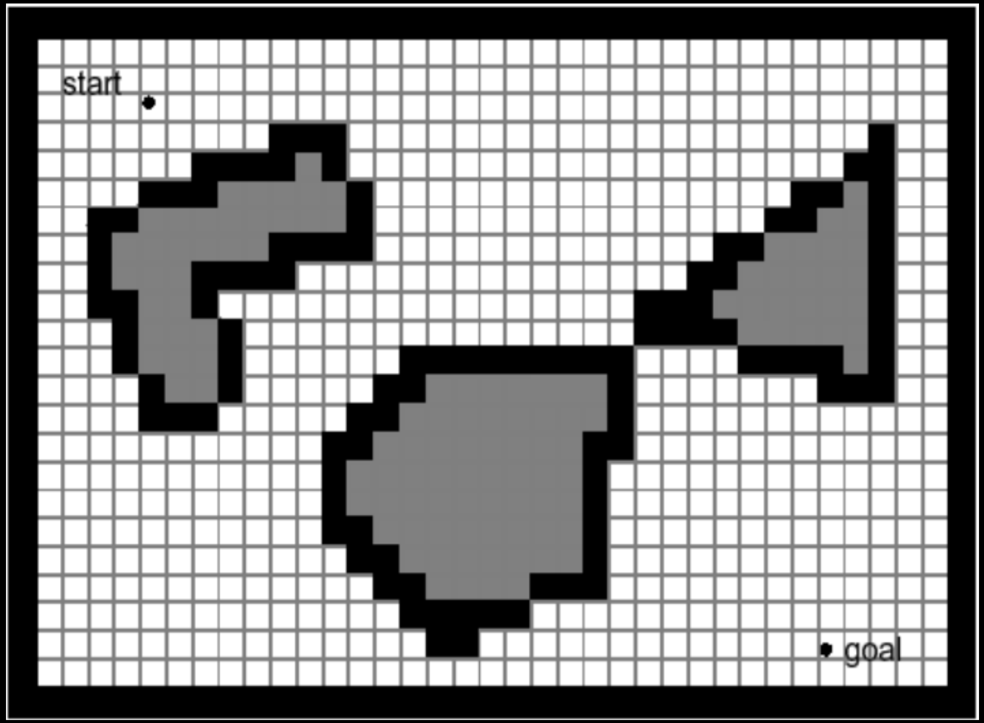




Fixed Decomposition

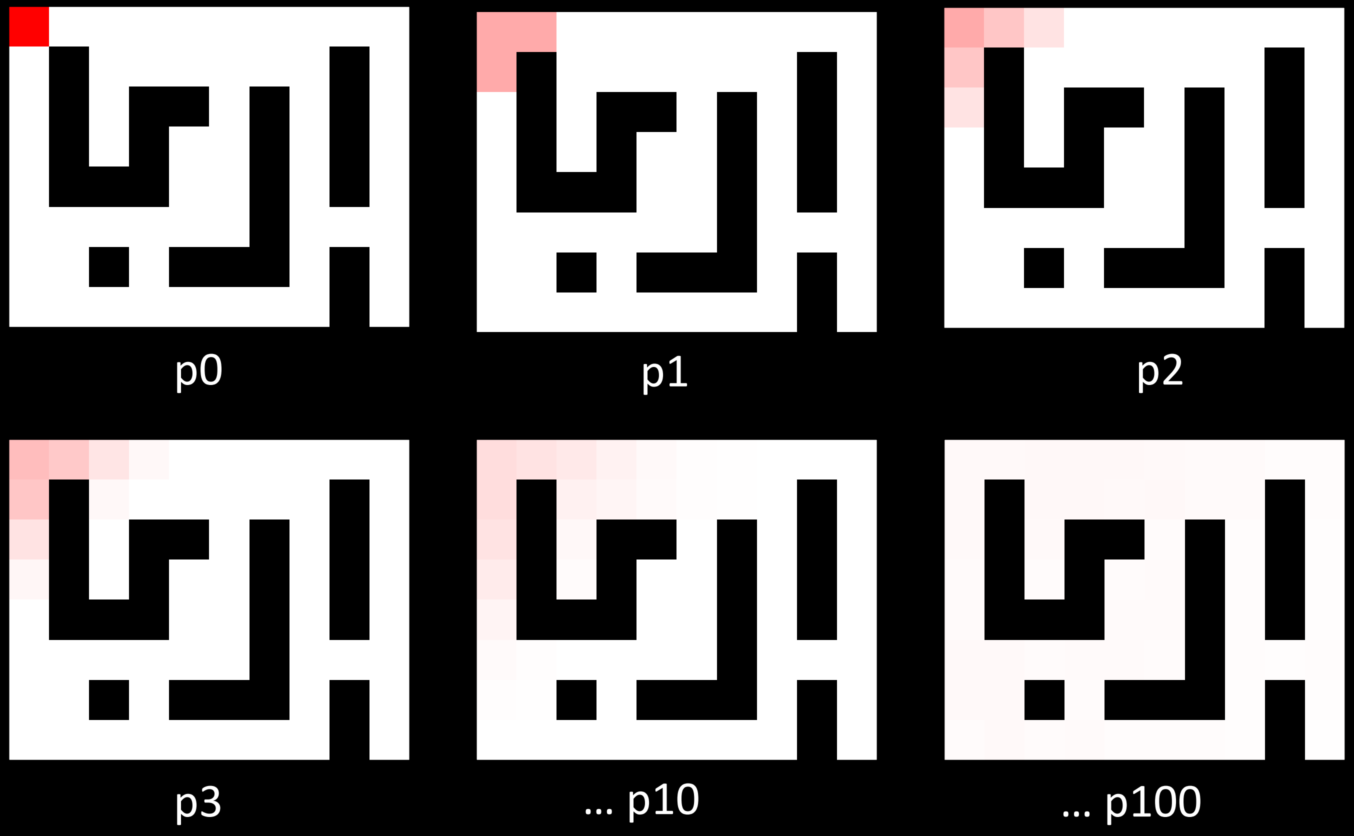
- Tessellate the world at a fixed resolution
- Approximate features given the resolution
- Most commonly used: Occupancy grid





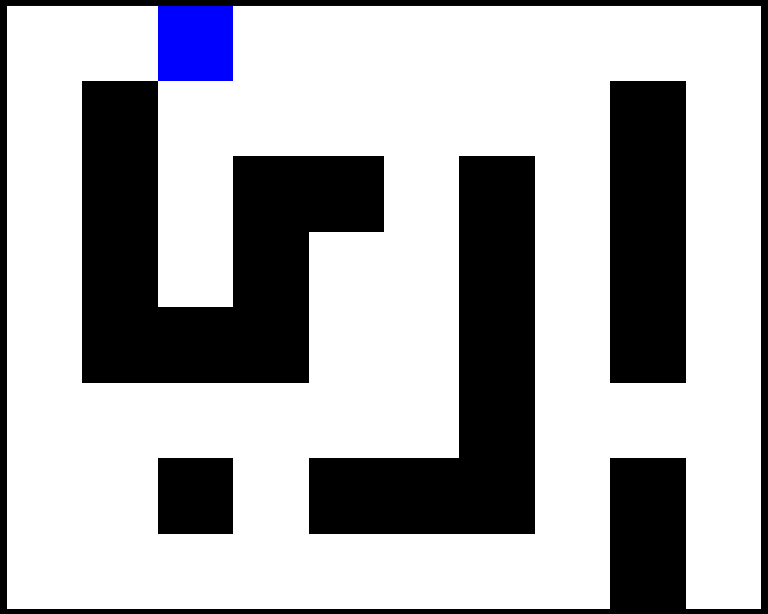
Fixed Decomposition

- Matlab example
    - Transition model (Random movement)

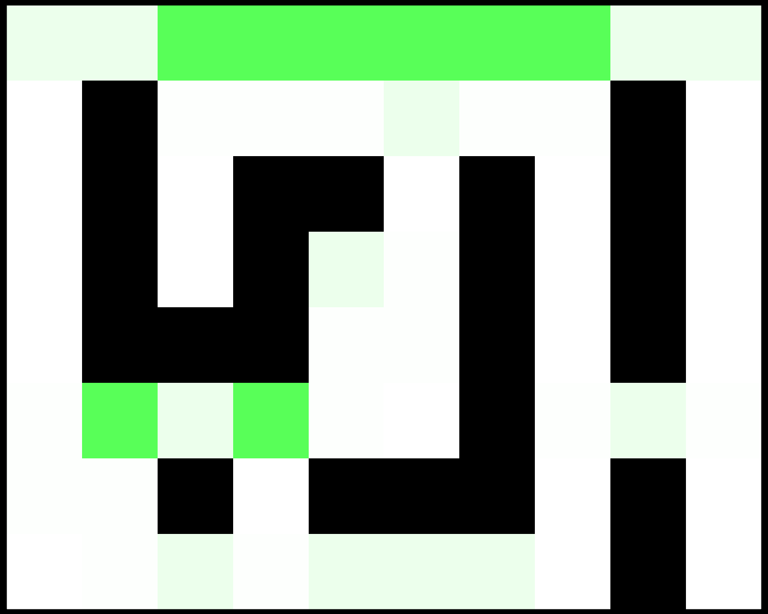


Fixed Decomposition

- Matlab example
  - Transition model (Random movement)
  - Sensor model (90% correct)



a robot state (x)

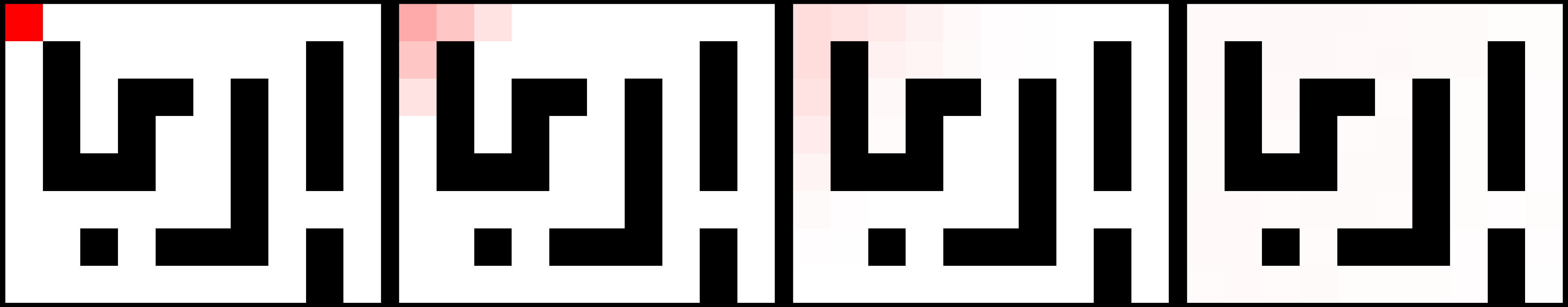


P(y|X)

Fixed Decomposition

- Matlab example
  - Transition model (Random movement)
  - Sensor model (90% correct)

$$P(X_{t+1} | y_{1:t+1}) = \eta \, P(y_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) P(x_t | y_{1:t})$$

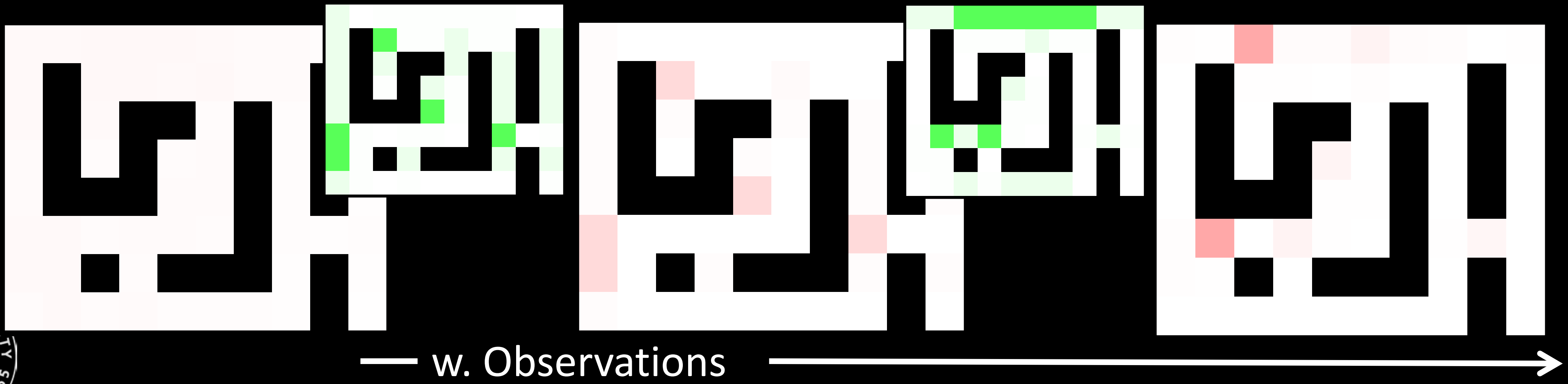


— No Observations ⟶

Fixed Decomposition

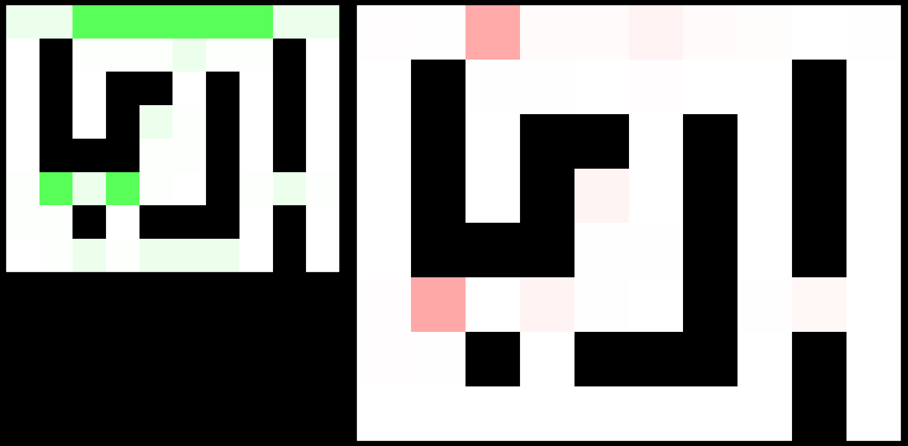
- Matlab example
  - Transition model (Random movement)
  - Sensor model (90% correct)

$$P(X_{t+1} | y_{1:t+1}) = \eta \, P(y_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) P(x_t | y_{1:t})$$



Fixed Decomposition

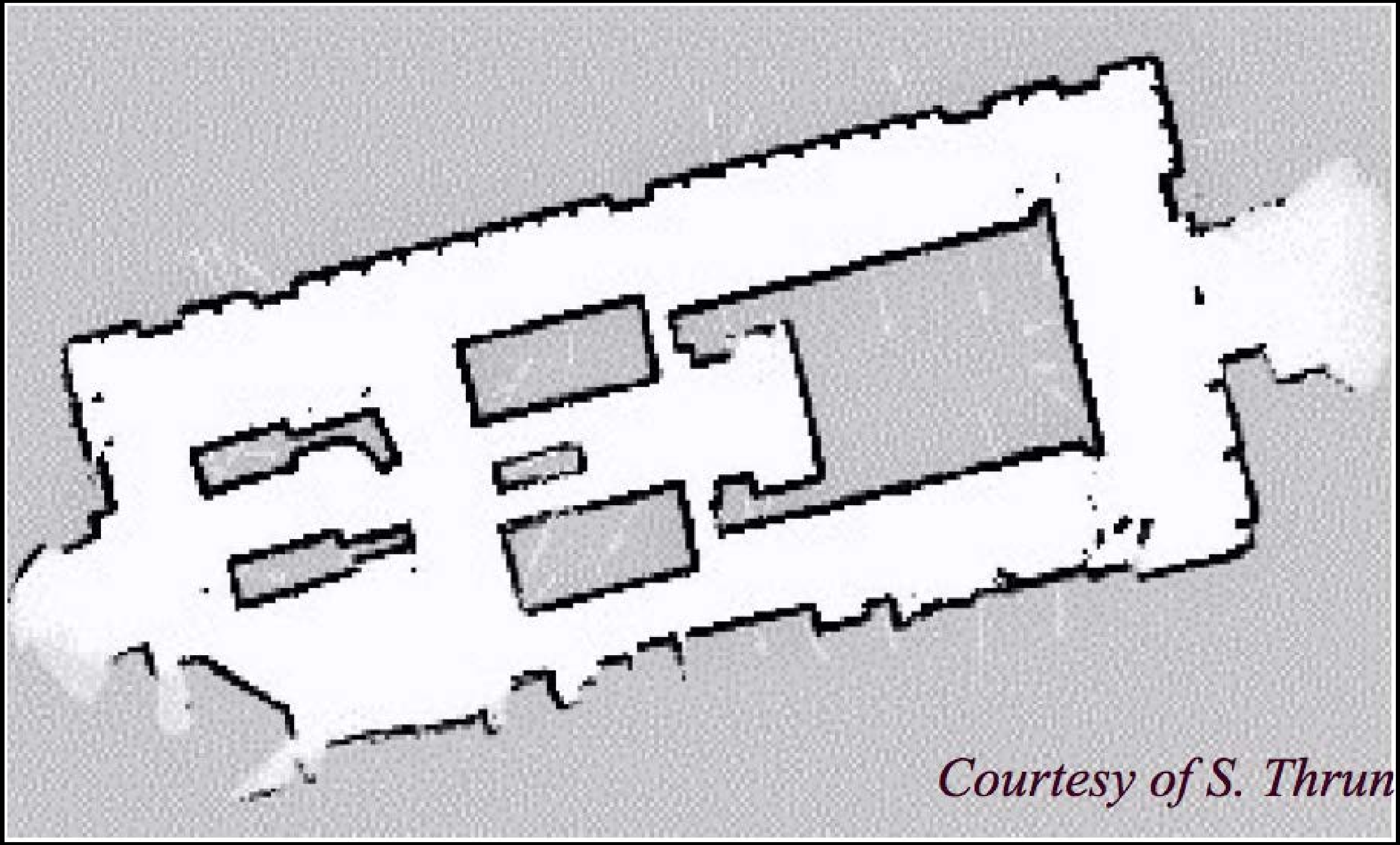
- Matlab example
  - Transition model (Random movement)
  - Sensor model (90% correct)
  - Factor in the input



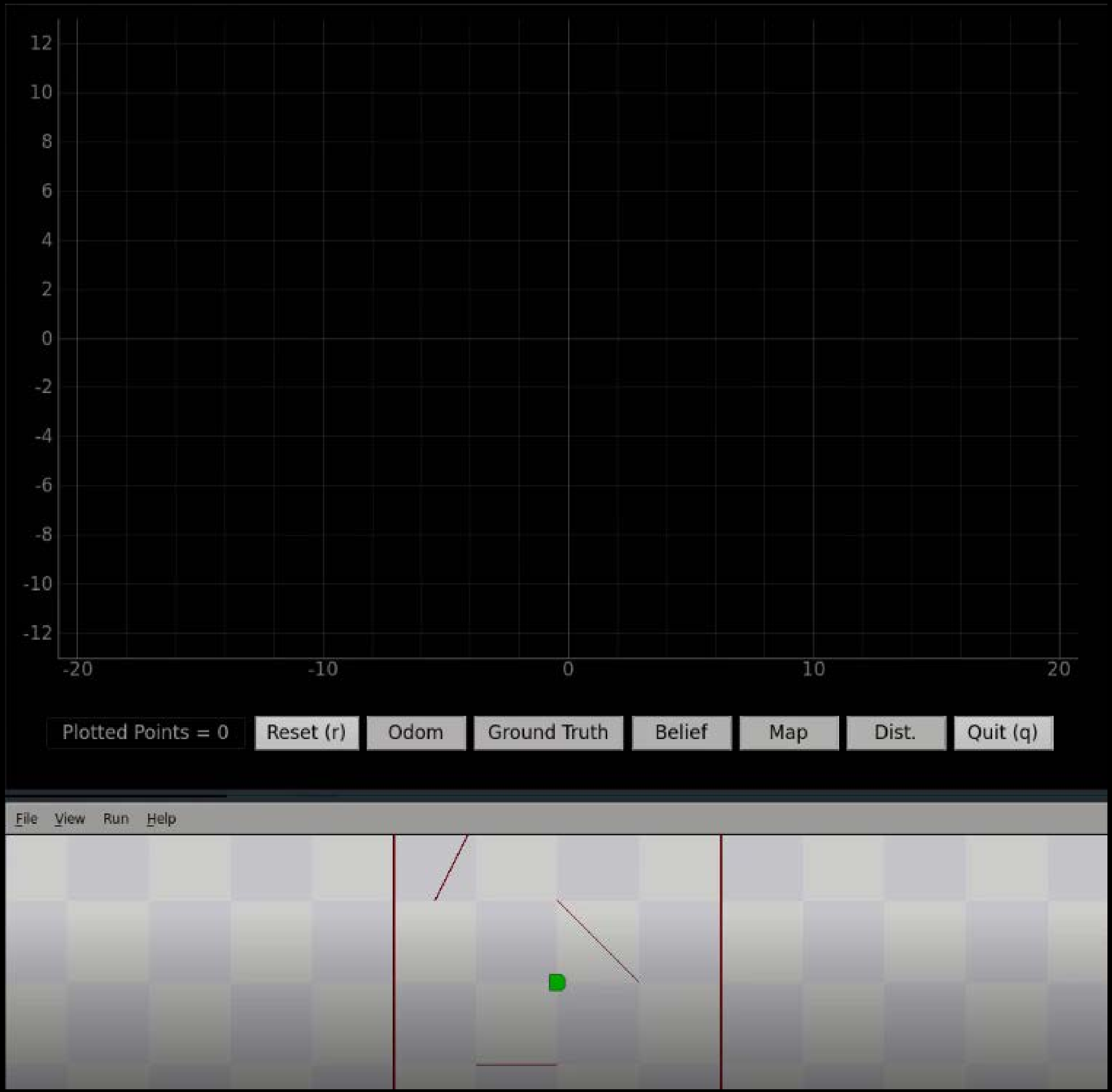
$$P(X_{t+1} | \, y_{1:t+1}) = \eta \; P(y_{t+1} | X_{t+1}) \sum_{xt} P(X_{t+1} \, | x_t, u_{t-1}) \, P(x_t \, | y_{1:t})$$

- It is easy to represent obstacles
- It is easy to compute probabilities
- But memory and computation is costly
- Resolution is critical
  - Must be high enough to capture motion and noise
  - Small features in the map

Fixed Decomposition



Lab 7-10: Combo of Linear Representation and Fixed Decomposition



Courtesy of Vivek Thangavelu