Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ECE 4160/5160
MAE 4910/5910

Fast Robots
Kalman Filter (recap)

ic: March 14th: Snow Day!

canvas.cornell.edu/courses/51518/discussion_topics/565888

Cornell | Passkey | CEI-Lab Slack | ECE Robotics @ Cor... | Slack | STC CROPPS | Slack | ECE 4960: Fa... | Slack | Packard Fell... | Other bookmarks

ECE4160/ECE516... › Announcements › March 14th: Snow...

Student View

Spring 2023

- Home
- Announcements
- Syllabus
- Modules
- Grades
- Zoom
- People
- Assignments
- Ed Discussion
- Rubrics
- Collaborations
- BigBlueButton

Edit

March 14th: Snow Day!

Kirstin Hagelskjaer Petersen

All Sections

Mar 13 at 6:47pm

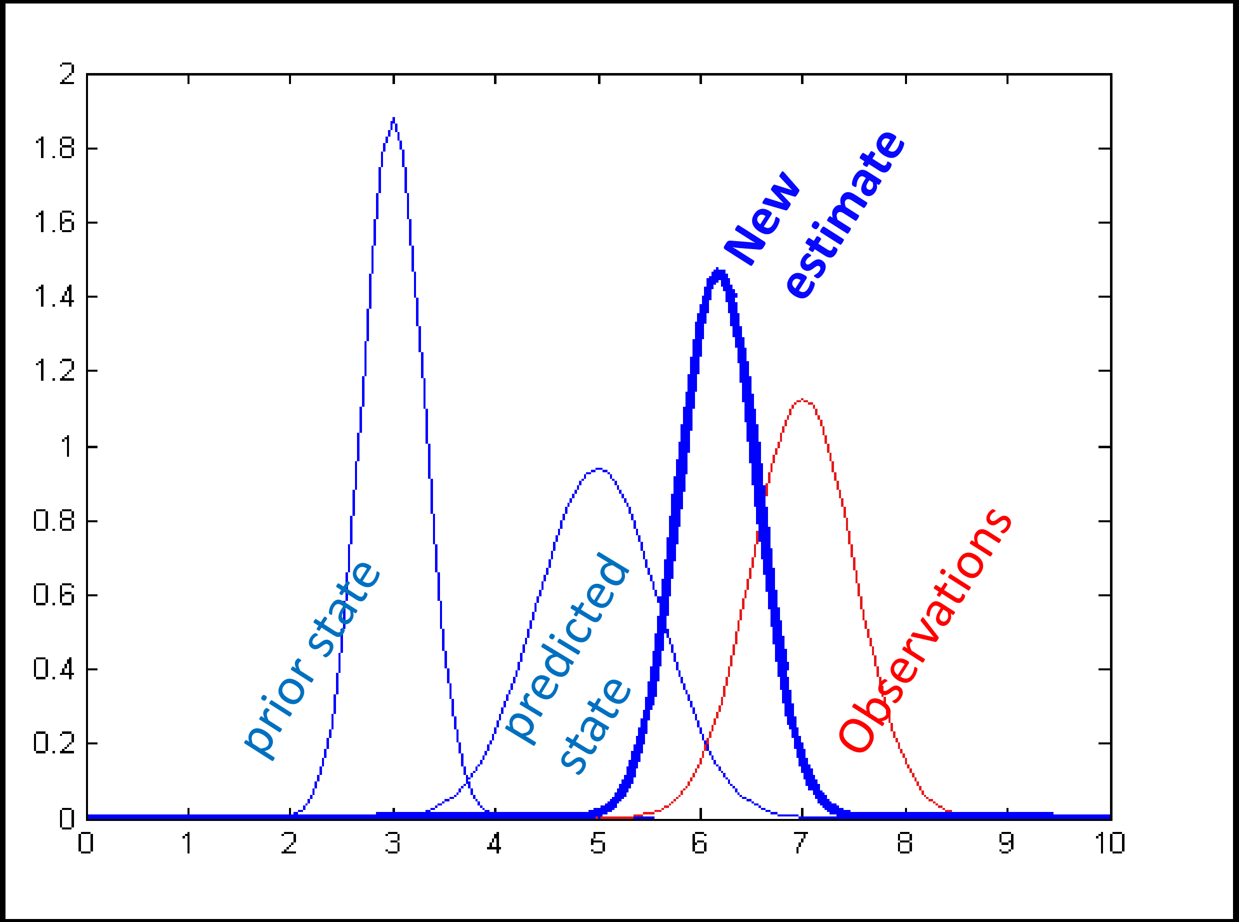
Today's class and lab are cancelled per university regulations!

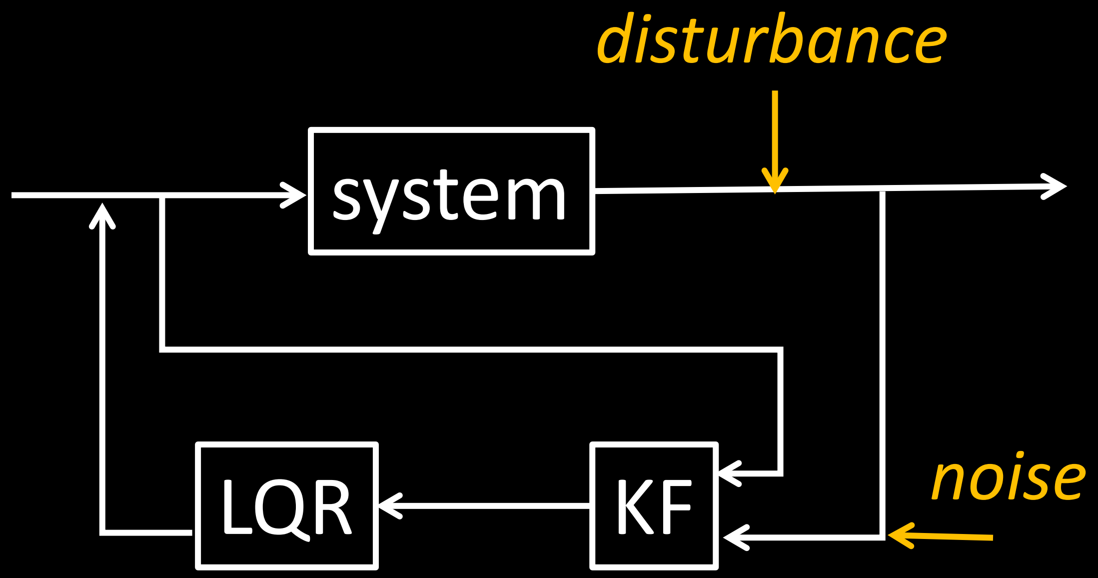
I have discussed with the TAs, and here's the new plan -- in part to make up for the snow day, and in part to help students who are struggling to keep up with the deadlines:

- Lab 6 deadlines have been postponed a full week (dates in Canvas are up-to-date). If you have already requested slip days, these will be cancelled.
- Lab 7 deadlines have also been postponed a full week (dates in Canvas are up-to-date), and we've decided to give you full points for completing task 1-3 as well as the new task 4.a in which you simply do extrapolation on the TOF data to get a better estimate of the distance to the wall. If you have time, consider doing task 4.b (implementing the Kalman Filter on your robot instead) for up to 4 bonus points!
- Lab 8 deadlines remain the same.

Hopefully, this news will make your snow day more enjoyable!

Kalman Filter





Kalman Filter

Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, u(t), z(t))

1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
5. $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

prediction (steps 1–2)

update (steps 3–6)

Example process and measurement noise covariance matrices

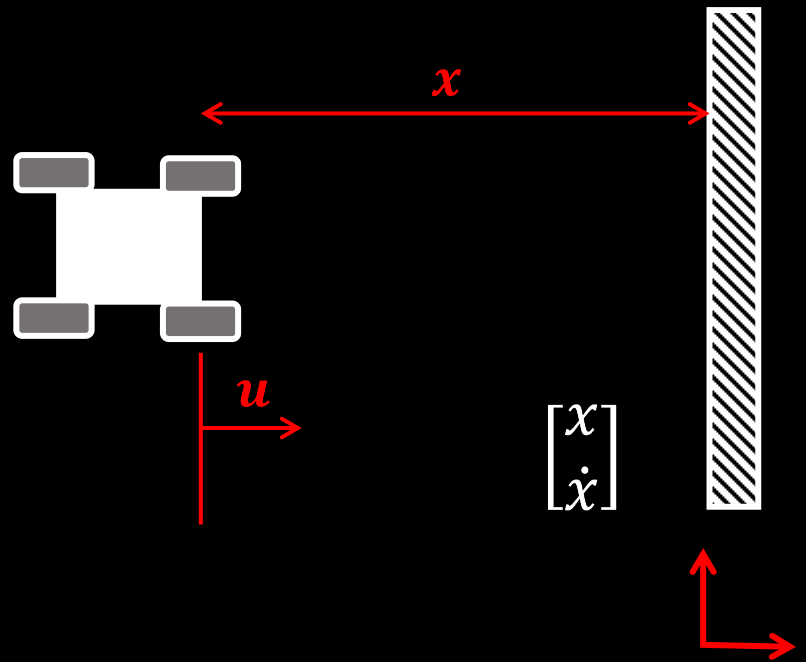
$$\Sigma_u = \begin{bmatrix} \sigma_1{}^2 & 0 \\ 0 & \sigma_2{}^2 \end{bmatrix}, \Sigma_z = \sigma_3{}^2$$

Example Lab 7

- Define A, B, C matrices
  - Using system ID on a step response

Example Lab 7

$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$



What is d and m?

- At steady state (cst speed), we can find *d*
  - $d = \frac{u}{\dot{x}} \approx 0.0005$ *(Assume u=1 for now)*
- We can use the rise time to find *m*
  - $m = \frac{-dt_{0.9}}{\ln(0.1)} \approx 4.1258 \cdot 10^{-4}$

State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Example Lab 7

- Define A, B, C matrices
  - Using system ID on a step response
- Sanity check
  - Run virtual Kalman Filter on data from Lab 6 PID
    - What is your initial state, and how confident in it are you?
    - How much trust do you put in your model versus your sensor values?
    - Experiment
      - E.g. put less trust in the model
      - E.g. put less trust in the sensors
      - Start with a bad initial estimate
        - Recall, our dynamic model is a bad estimate for the static robot

Linear Systems Control – "review of review"

- Linear system: $\dot{x} = Ax$
- Solution: $x(t) = e^{At}x(0)$
- Eigenvectors: $T = [\xi_1 \quad \xi_2 \quad \dots \quad \xi_n]$
- Eigenvalues: $D = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \dots & \\ 0 & & & \lambda_n \end{bmatrix}$

```
>>[T,D] = eig(A)
```

- Linear transform: $AT = TD$
- Solution: $e^{At} = Te^{Dt}T^{-1}$
- Mapping from x to z to x: $x(t) = Te^{Dt}T^{-1}x(0)$
- Stability in continuous time: $\lambda = a + ib$, stable iff a<0
  - Discrete time: $x(k+1) = \tilde{A}x(k), \tilde{A} = e^{A\Delta t}$
    - Stability in discrete time: $\tilde{\lambda}^n = R^n e^{in\theta}$, stable iff R<1

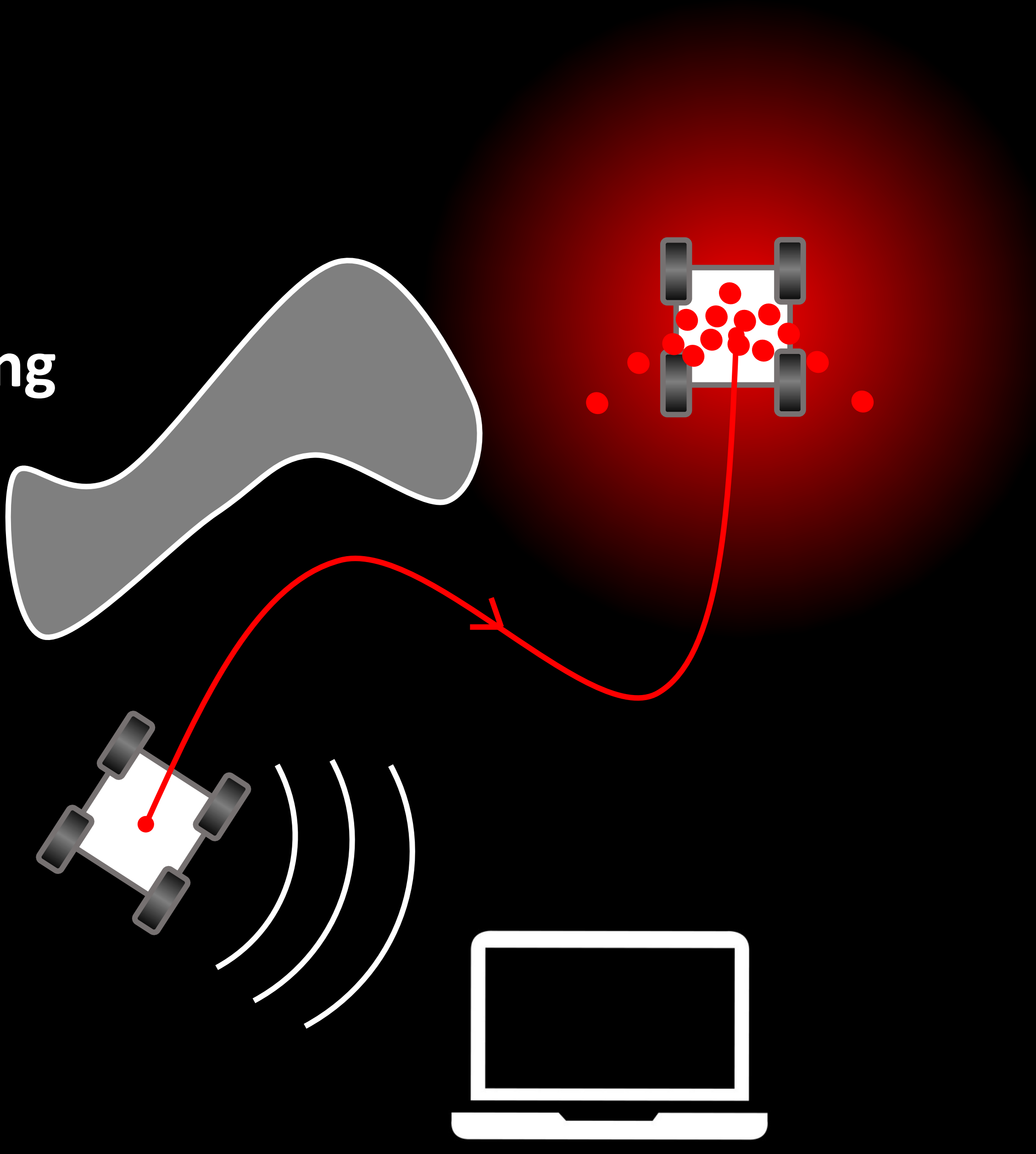
- Linearizing non-linear systems
  - Fixed points
  - Jacobian
- Controllability
  - $\dot{x} = (A - BK)x$
  - `>>rank(ctrb(A,B))`
- Reachability
- Controllability Gramian
- Pole placement
  - `>>K=place(A,B,p)`
- Optimal control (LQR)
  - `>>K=lqr(A,B,Q,R)`
- Observability
  - `>>rank(obsv(A,C))`
- Optimal observer (KF)
  - Sensor/model noise

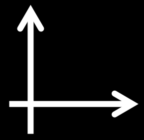
What we covered so far…

- Configuration space and transformations
- Data types
- Sensors
  - Distance Sensors
  - Odometry and IMU
  - Characterization
- Actuators/Motors
- Wiring/EMI
- Control
  - State space models
  - PID/LQR control
  - Observers
  - Deterministic -> Probabilistic Robots
    - Bayes theorem

Next up….

Navigation and Planning





Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

ECE 4160/5160
MAE 4910/5910

Fast Robots
Navigation and Planning

Slides adapted from Vivek Thangavelu

Navigation

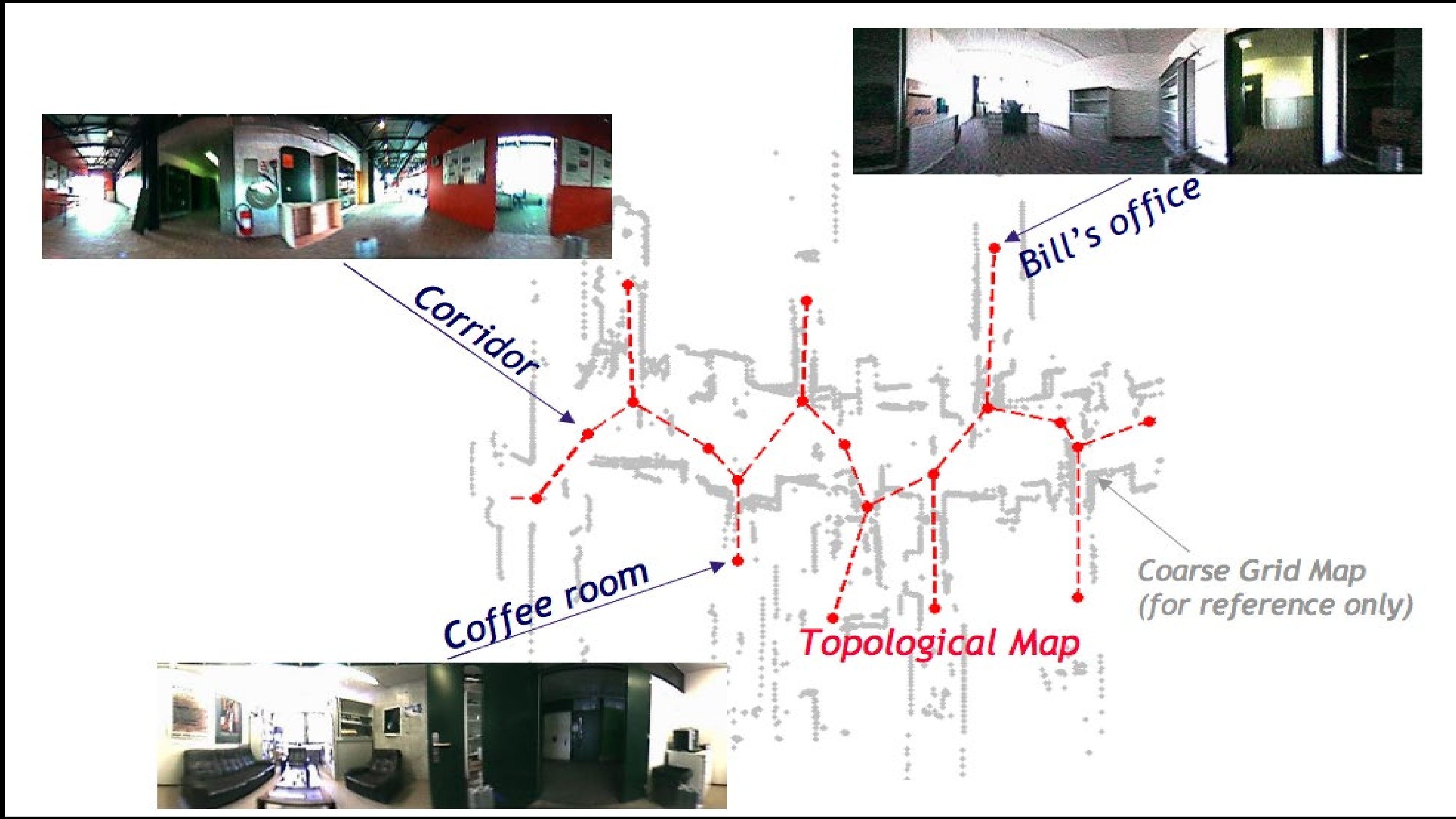
- **Problem:** Find the path in the workspace from an initial location to a goal location, while avoiding collisions
- How do you get to your goal?
  - Can you see your goal?
  - Do you have a map?
  - Are obstacles unknown or dynamic?
  - Does it matter how fast you get there?
  - Does it matter how smooth the path is?
  - How much compute power do you have?
  - How precise and accurate is your motion control?
  - What sensors do you have available?
  - etc.

Navigation

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

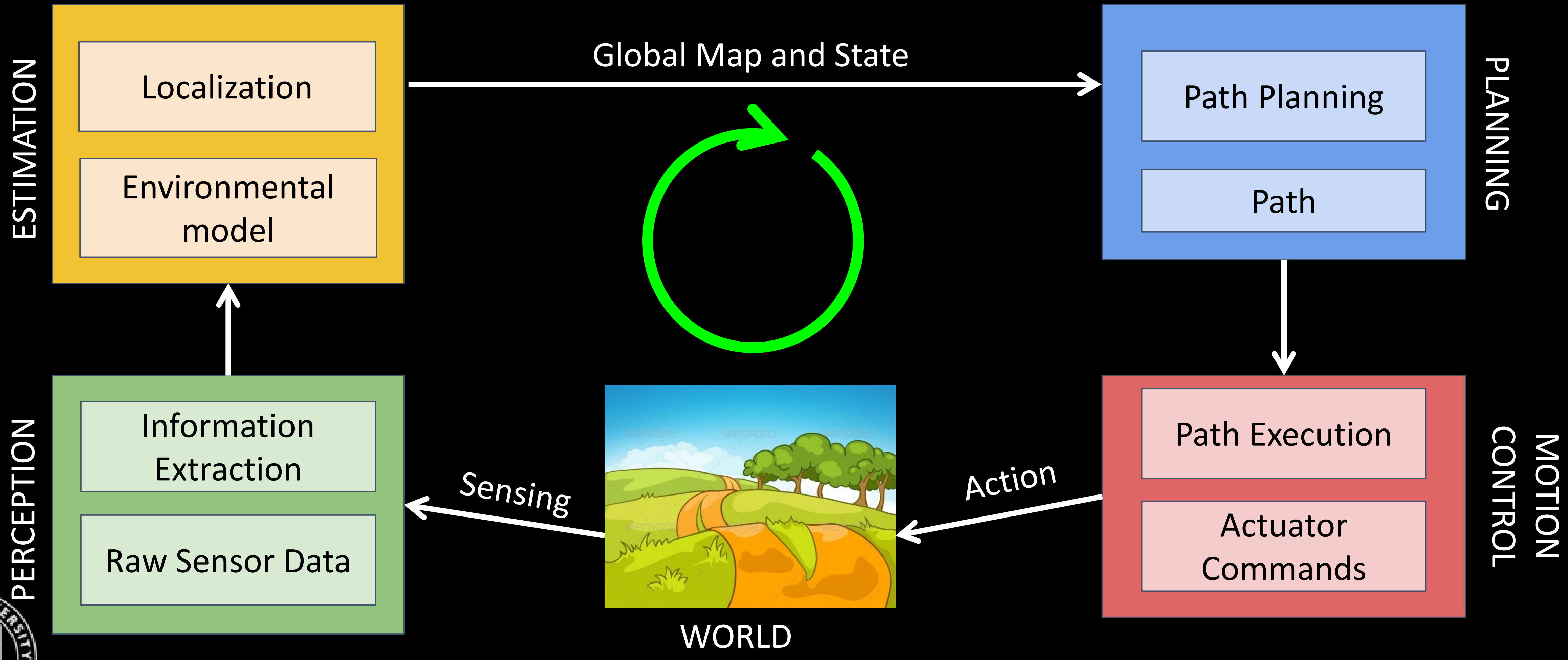
  - **Global navigation**
    - Given a map and a goal location, find and execute a trajectory that brings the robot to the goal
    - (Long term plan)

  - **Local navigation**
    - Given real-time sensor readings, modulate the robot trajectory to avoid collisions
    - (Short term plan)



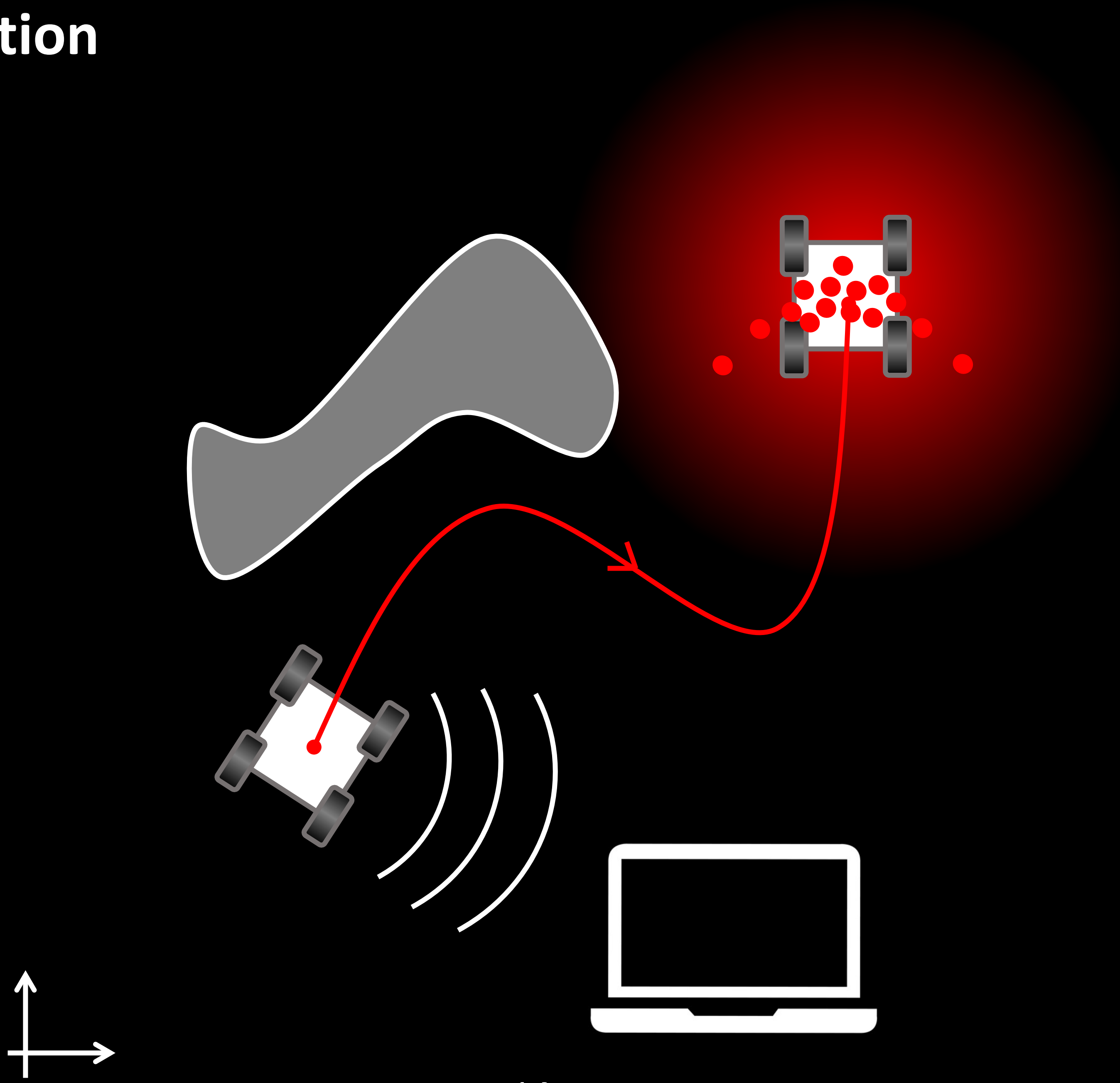
Navigation

- 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

- Use goal position, recent sensor readings, and relative position of robot to goal
  - Can be based on a local map
  - Often implemented as a separate task
  - 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

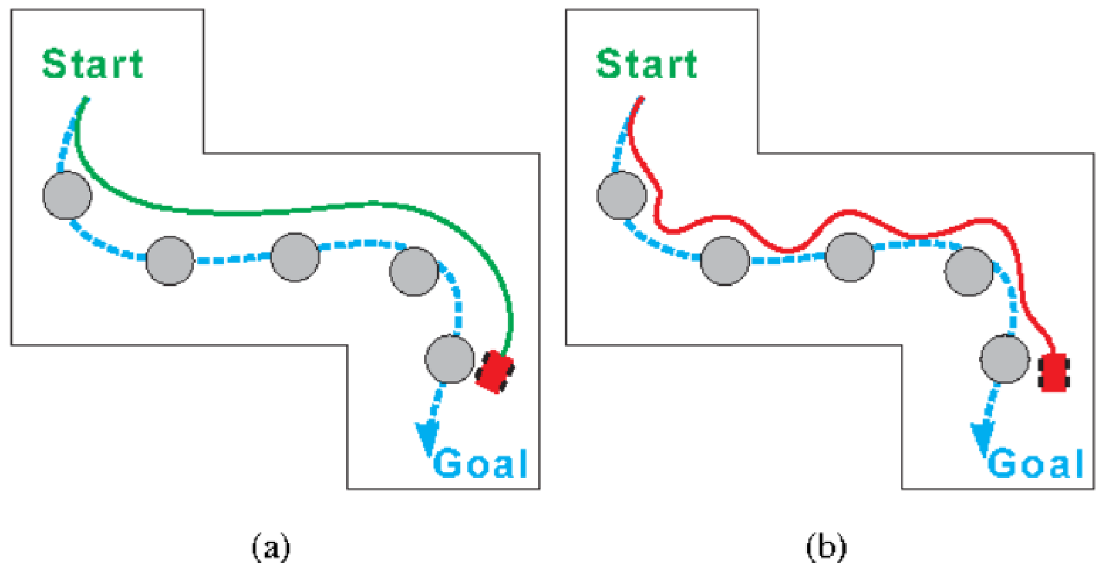
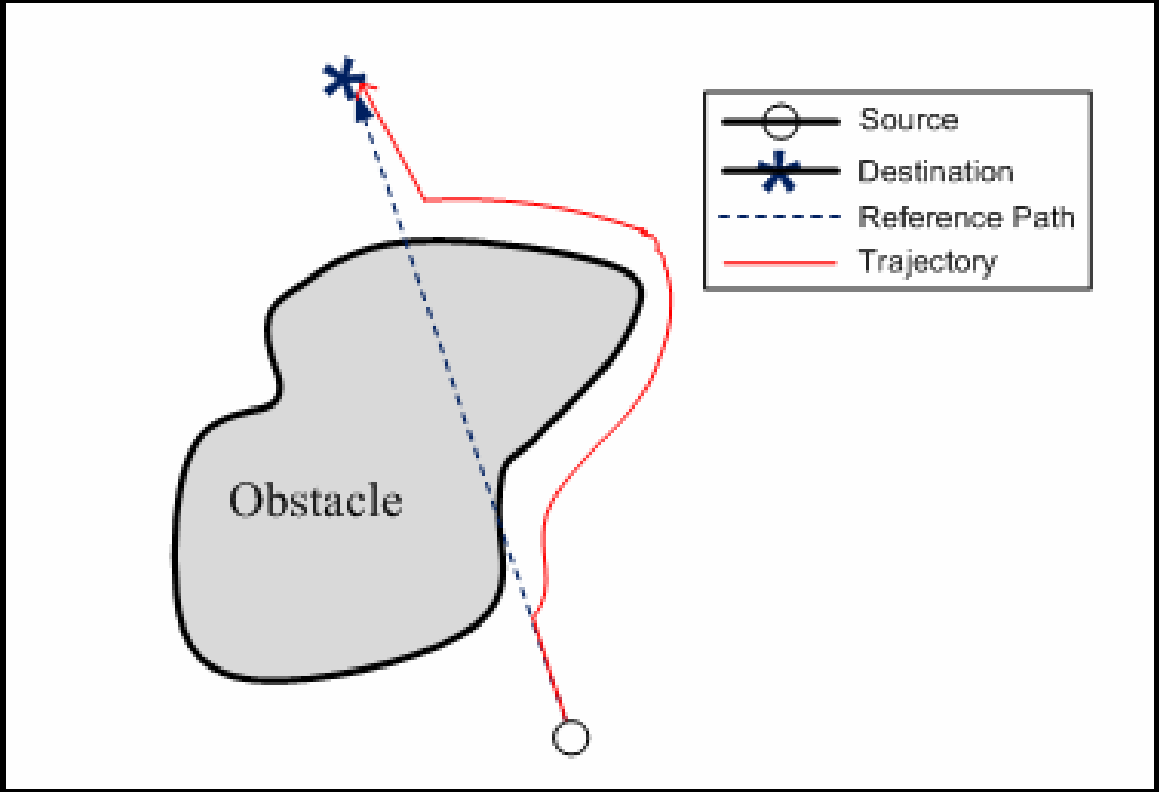


Fig. 1. Dashed blue spline is global path: a) Green spline is ideal local path; b) Red spline is actual local path

Bug Algorithms

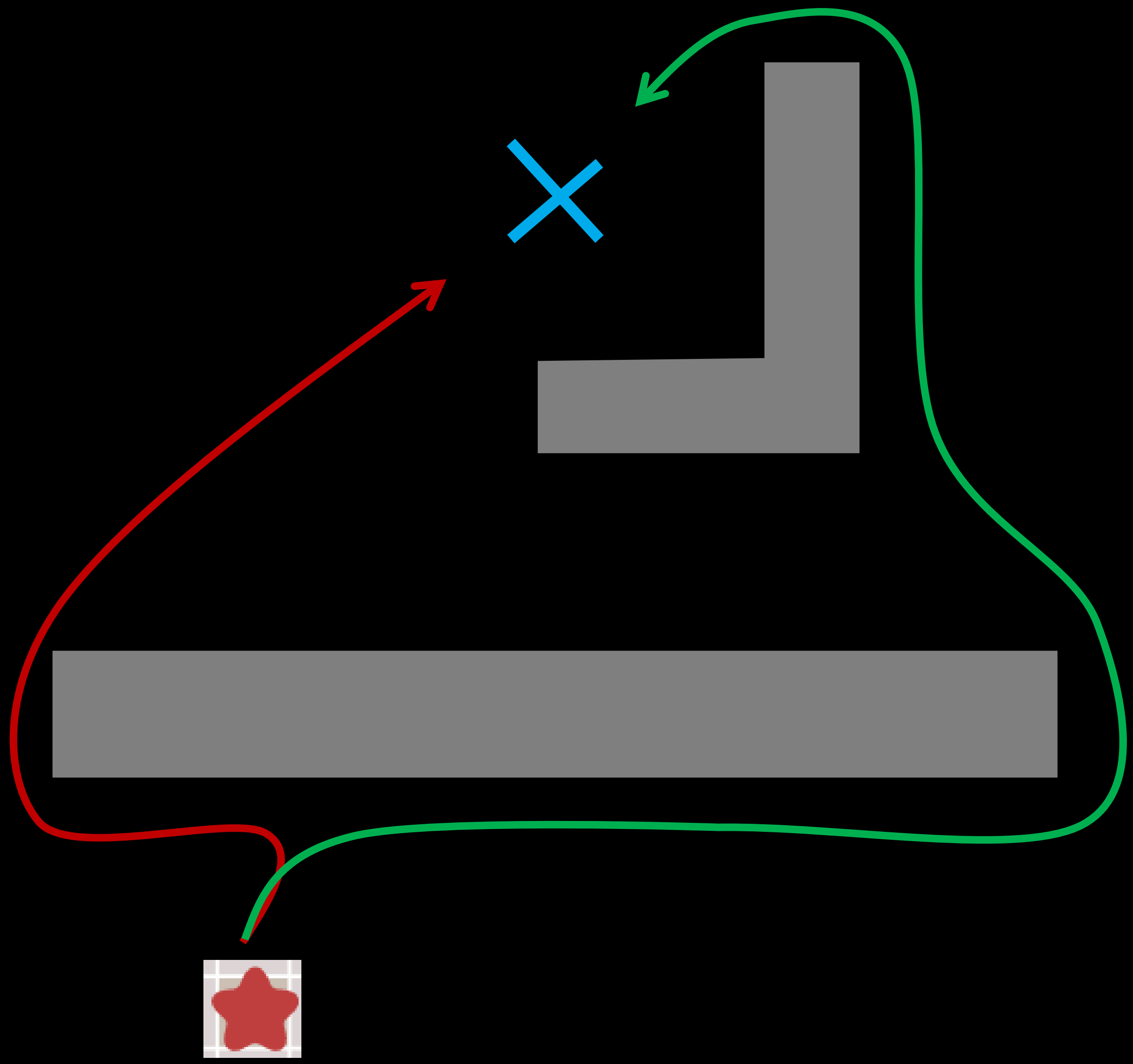
- Uses local knowledge, and the direction and distance to the goal
- Basic idea
  - Follow the contour of obstacles until you see the goal
  - State 1: Seek goal
  - State 2: follow wall
- Different variants: Bug0, Bug1, Bug2
- Advantages
  - Super simple
  - No global map
  - Completeness
- Disadvantages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

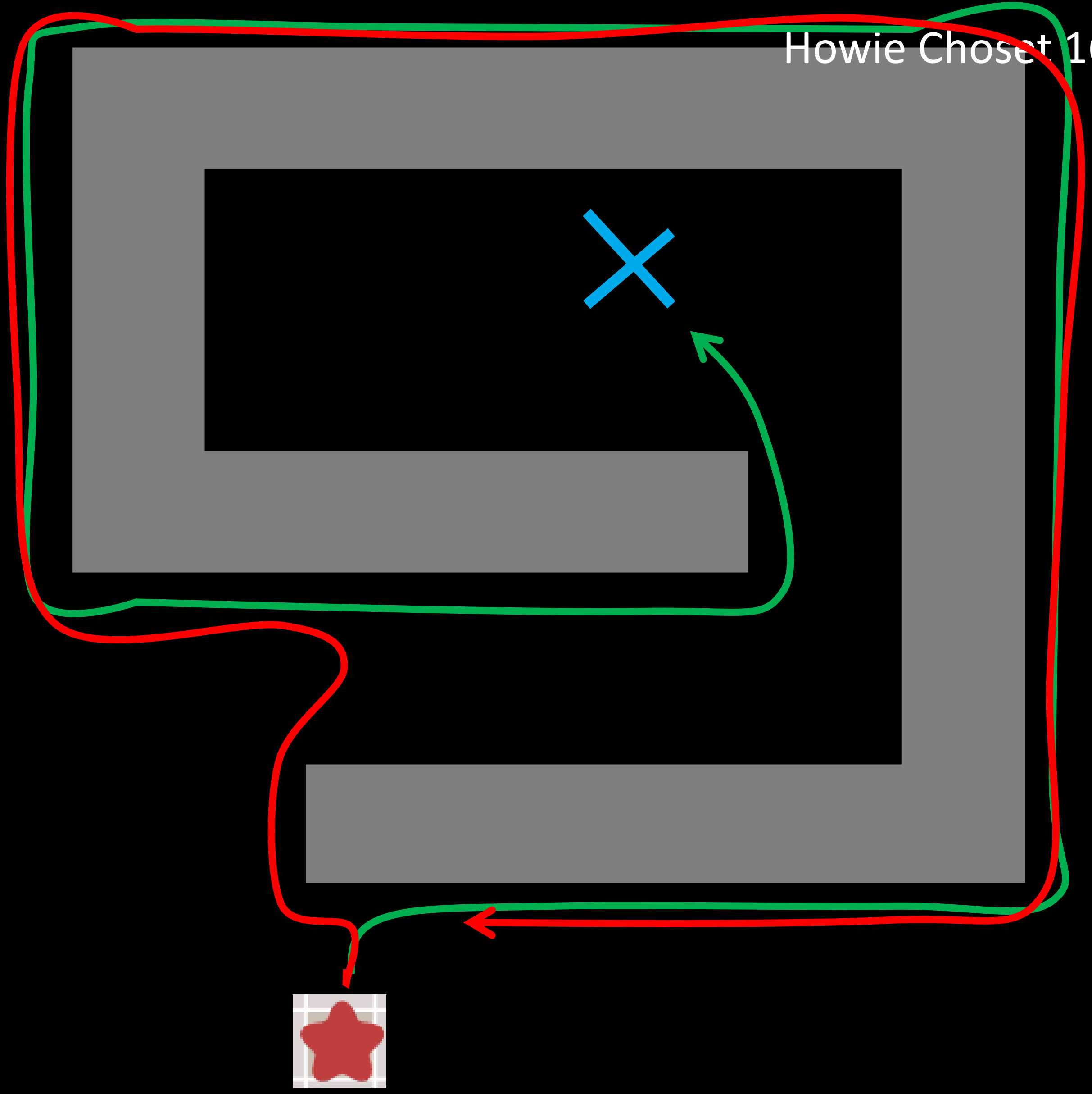
Howie Choset 16-735

Sensor Assumptions

- Direction to the goal
- Detect walls

Algorithm

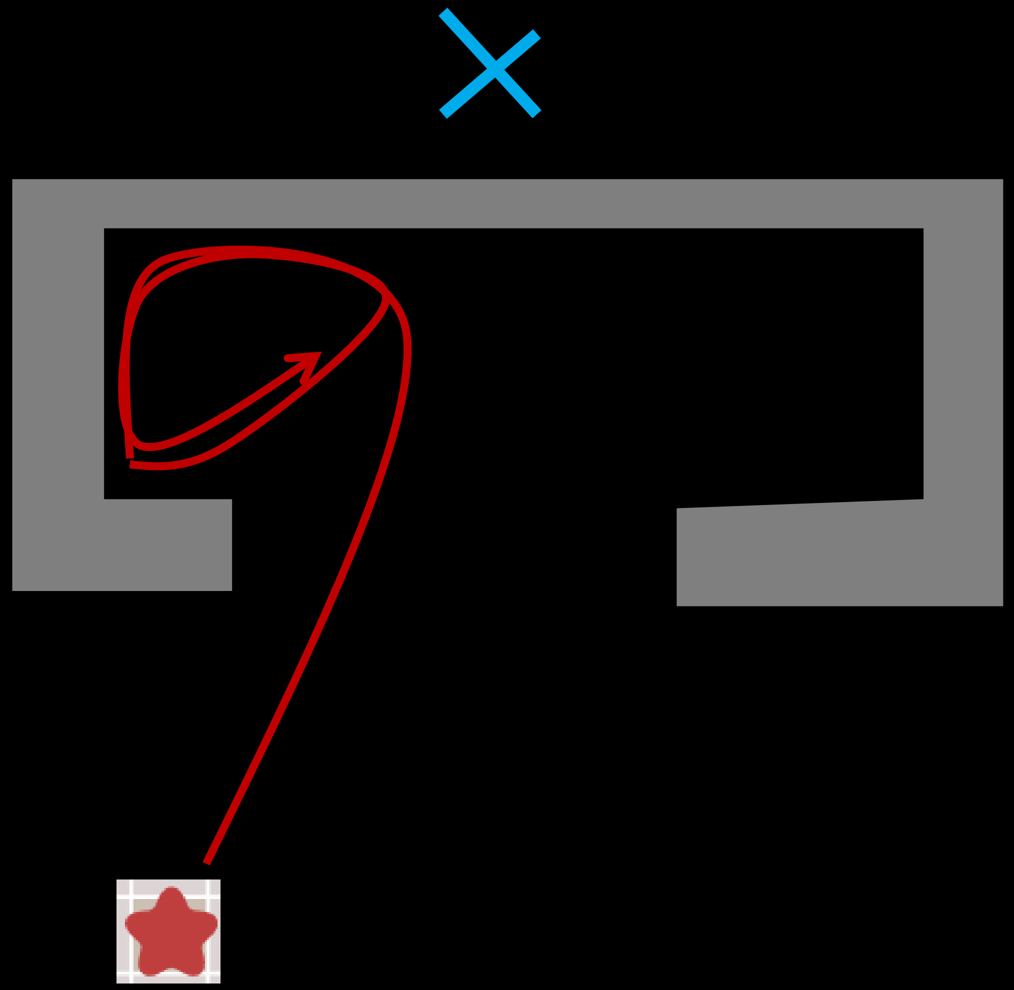
1. Go towards goal
2. Follow obstacles until you can go towards goal again
3. Loop



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

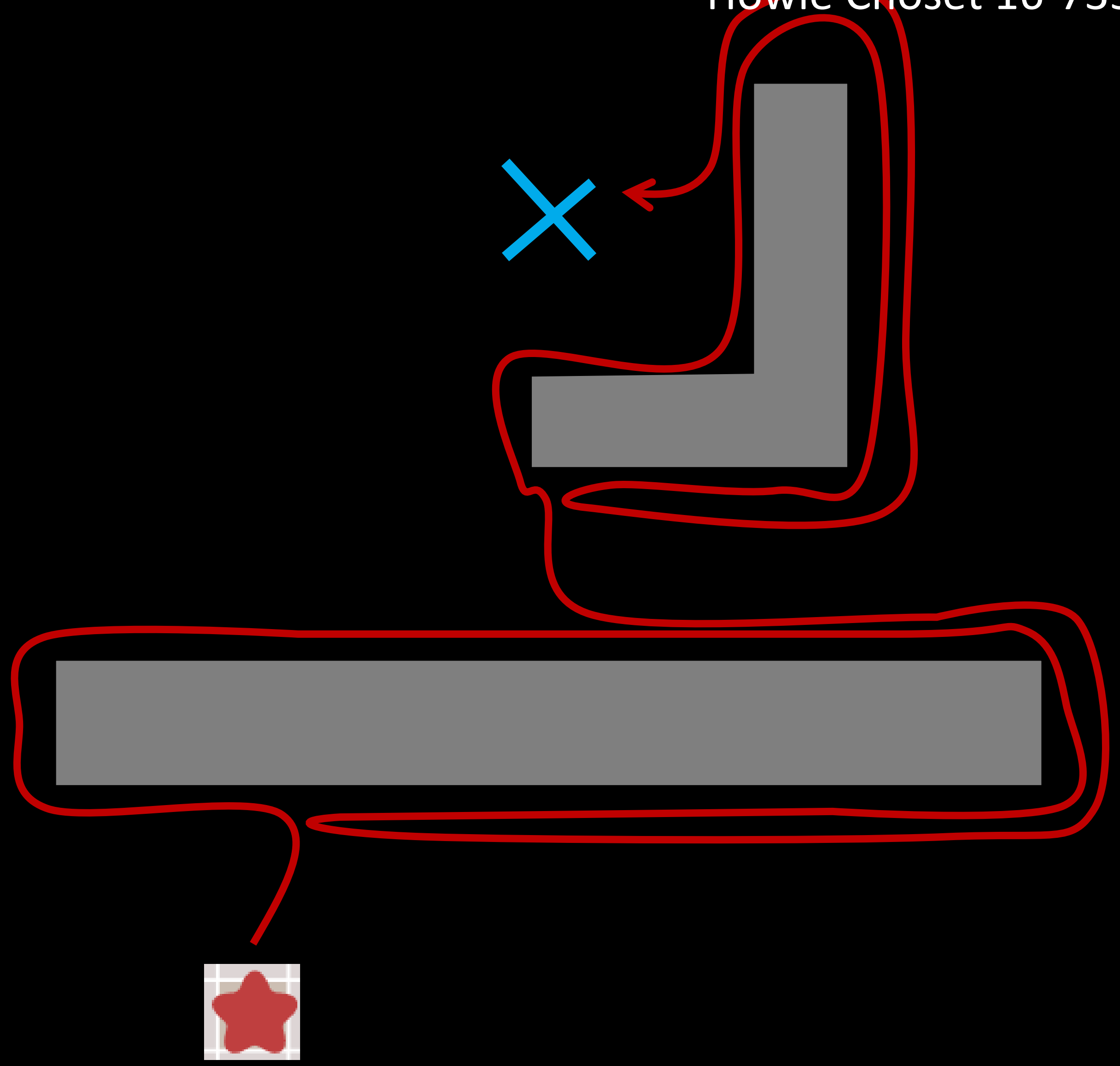
Bug 1

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



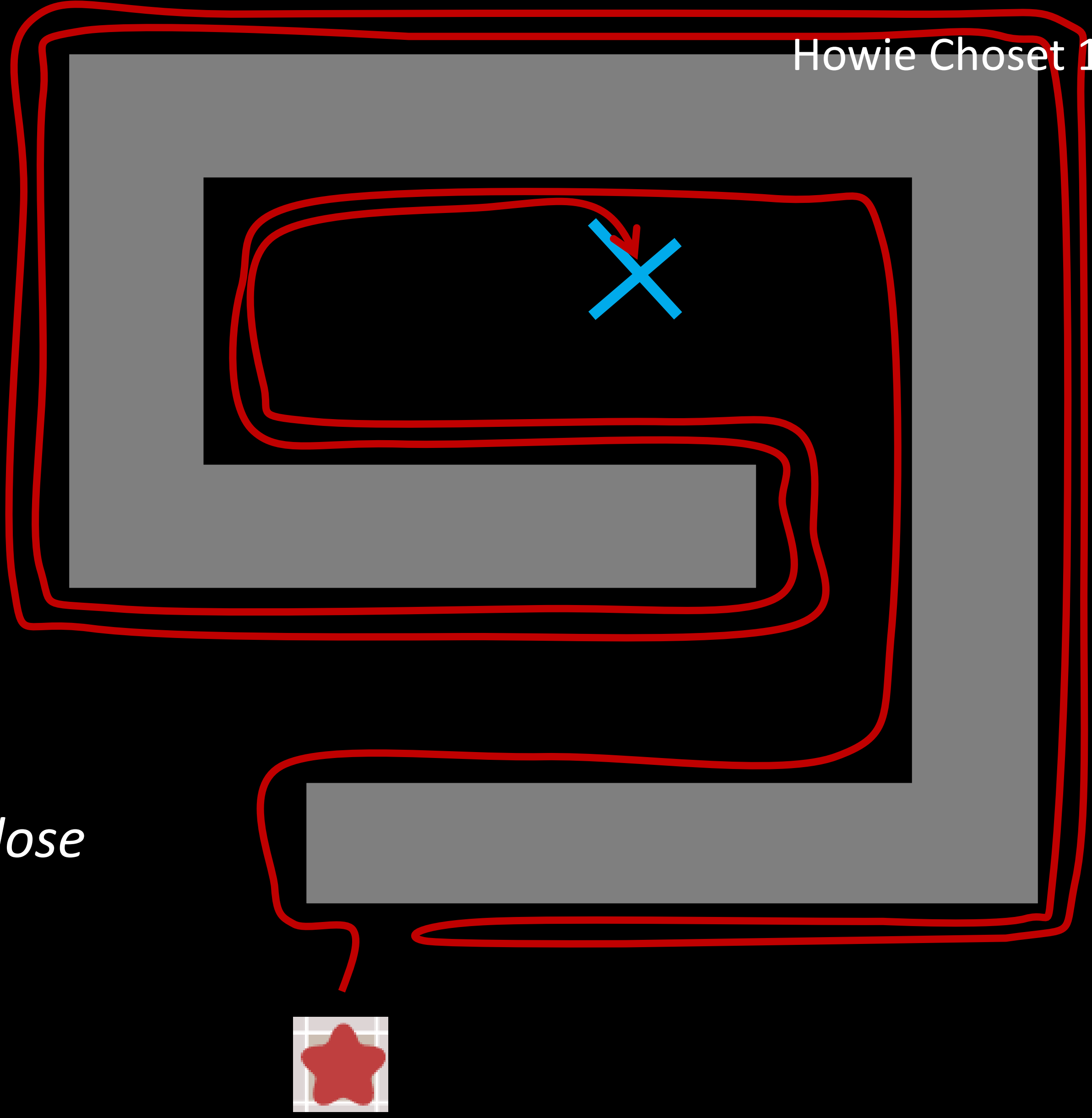
Bug 1

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry

Algorithm

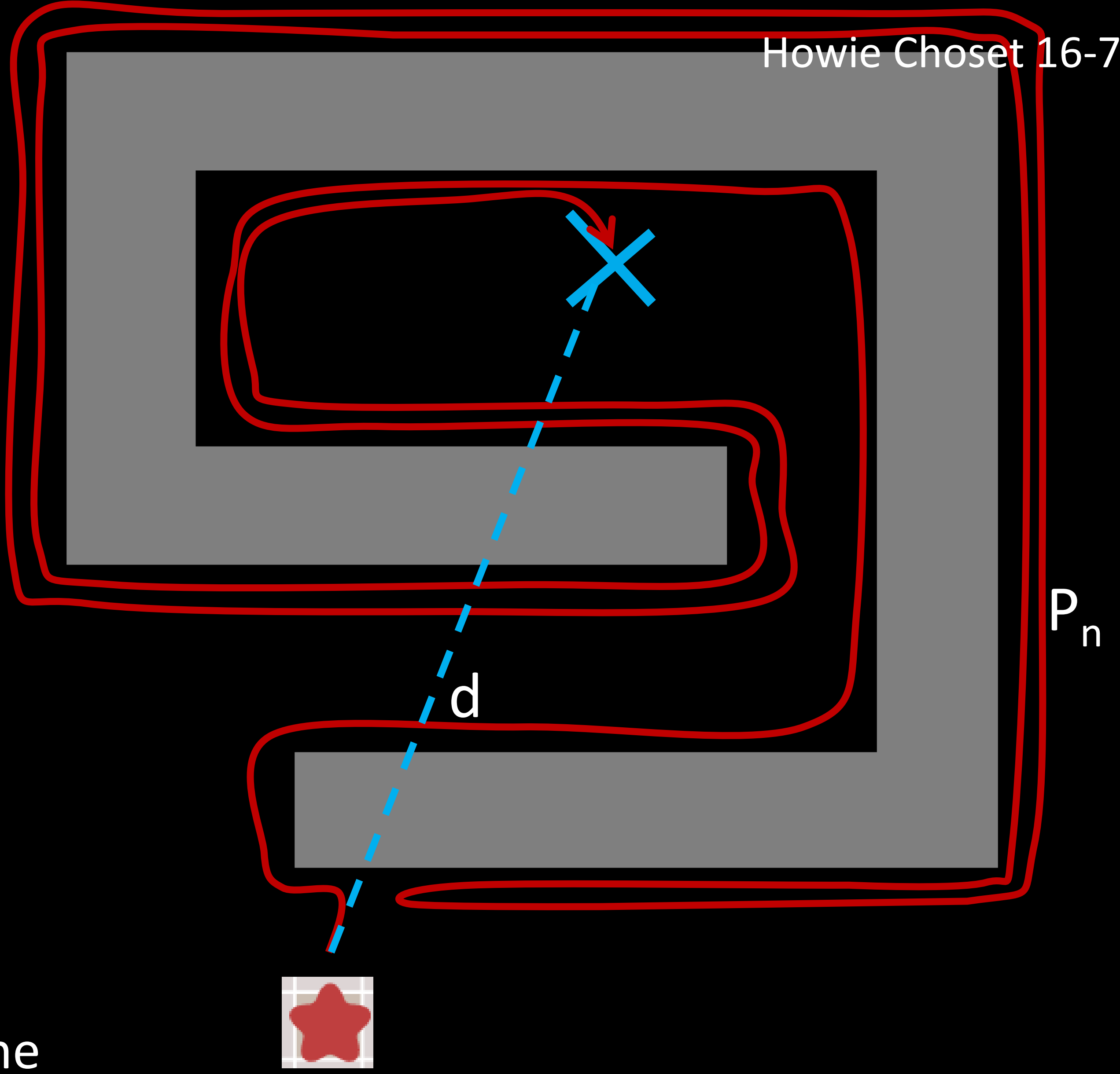
1. Go towards goal
2. Follow obstacles *and remember how close you got to the goal*
3. Return to the closest point, and loop



Bug 1 - formally

Sensor Assumptions

- Direction to the goal
- Detect walls
- Odometry



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

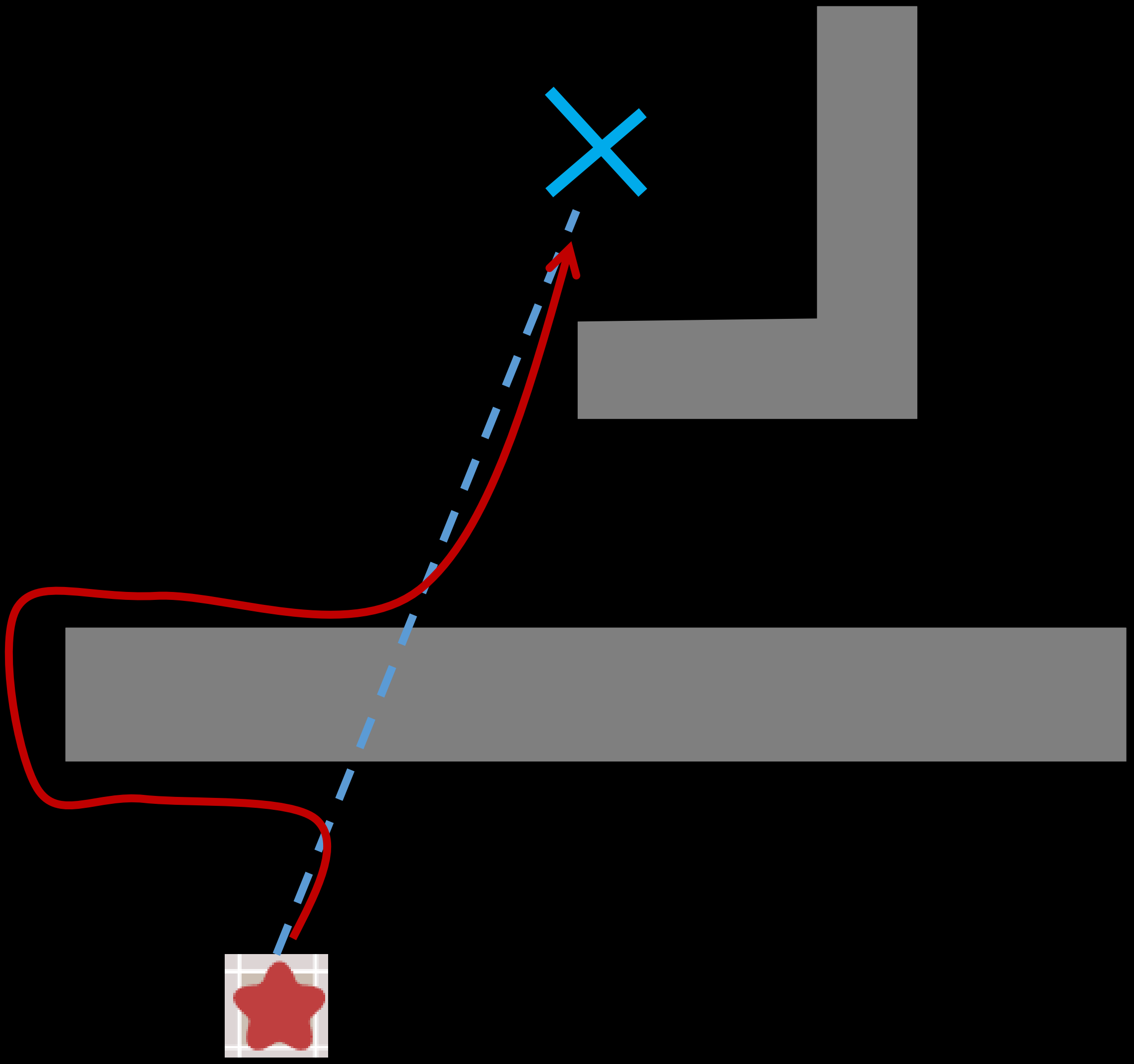
Bug 2

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



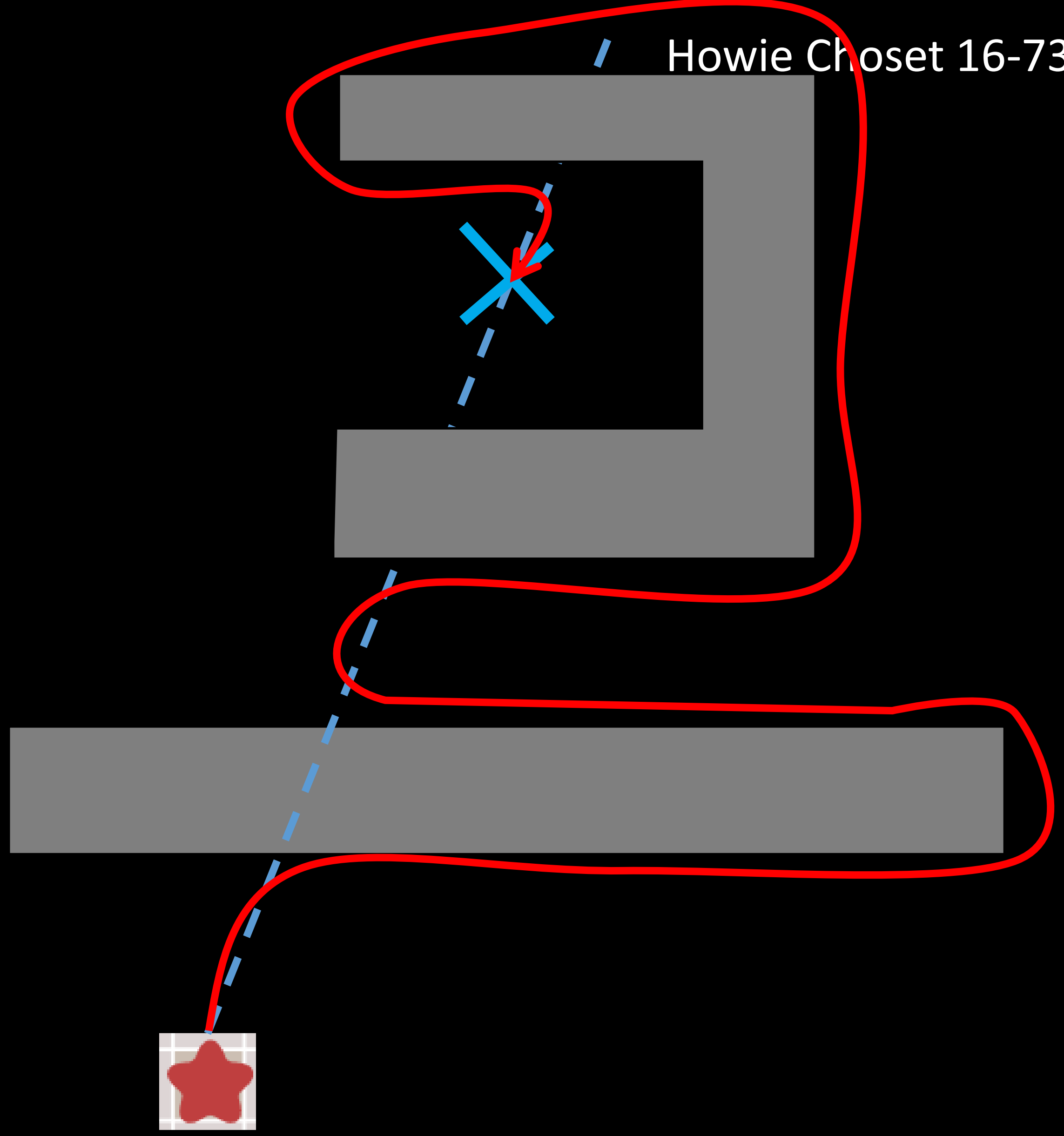
Bug 2

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



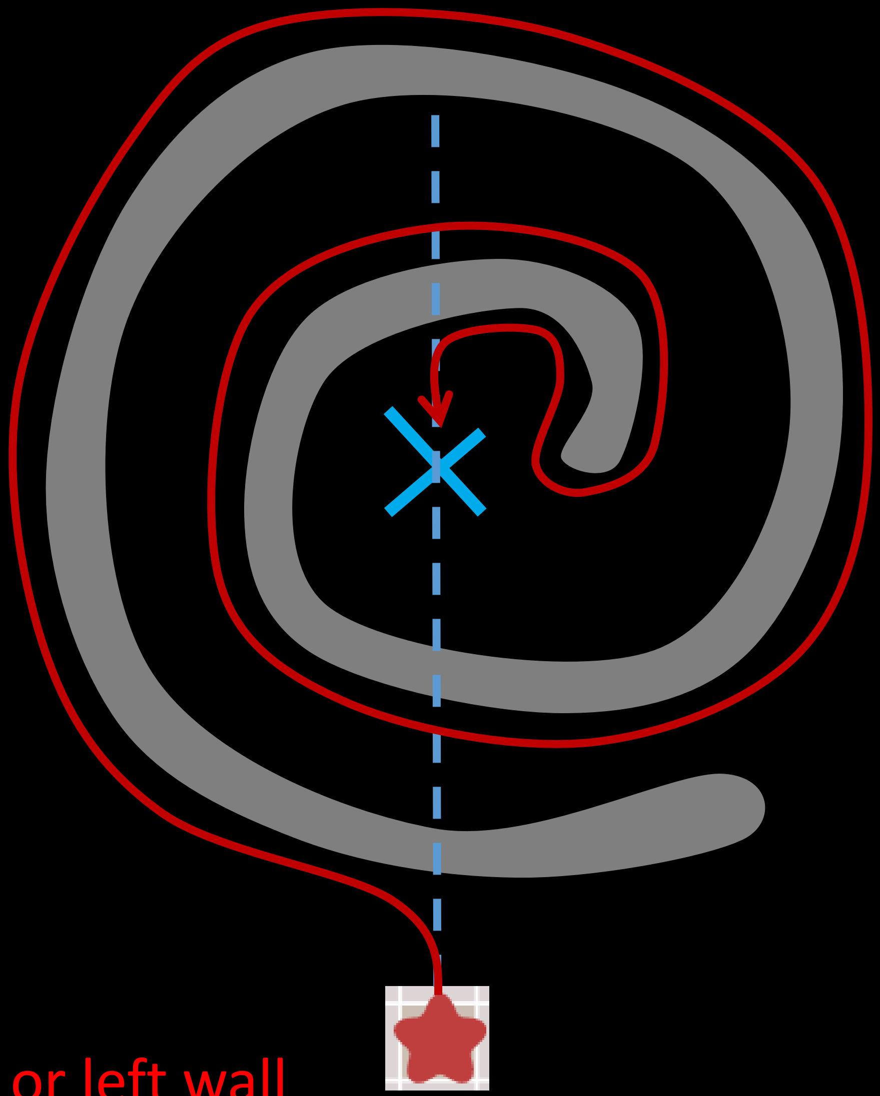
Bug 2

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

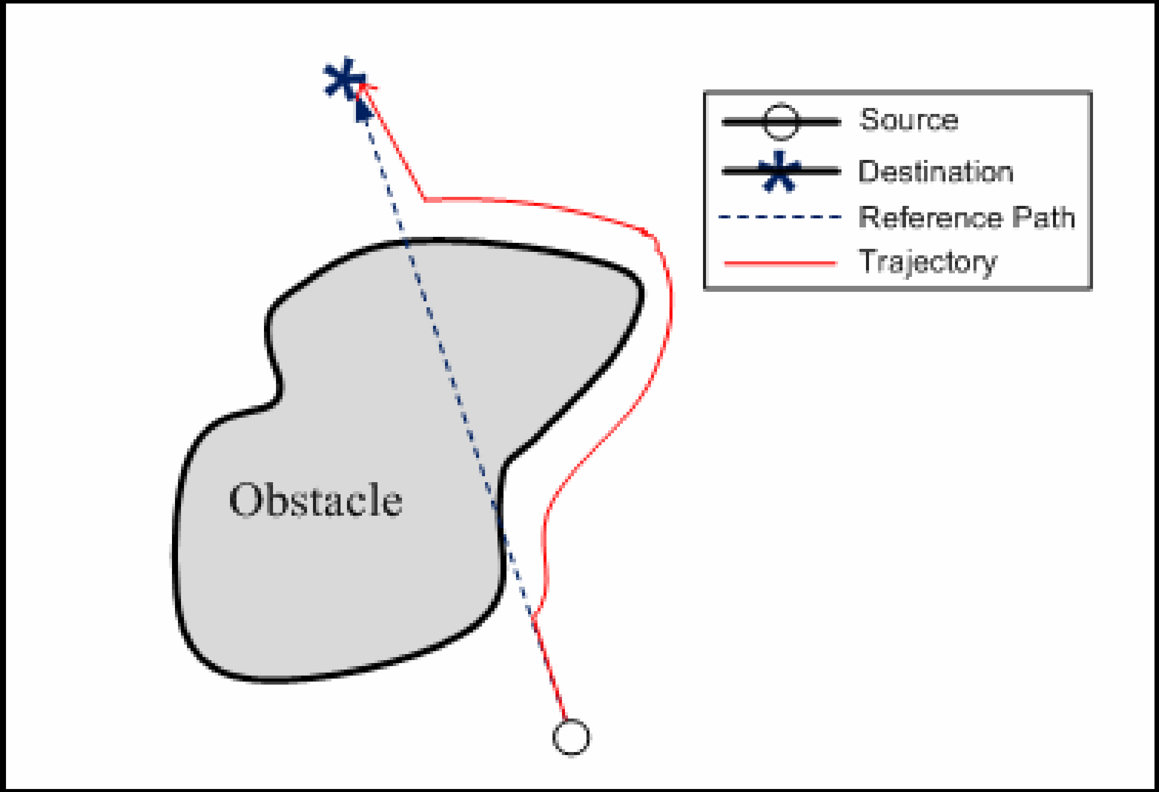
Bug 1
Layout 2

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

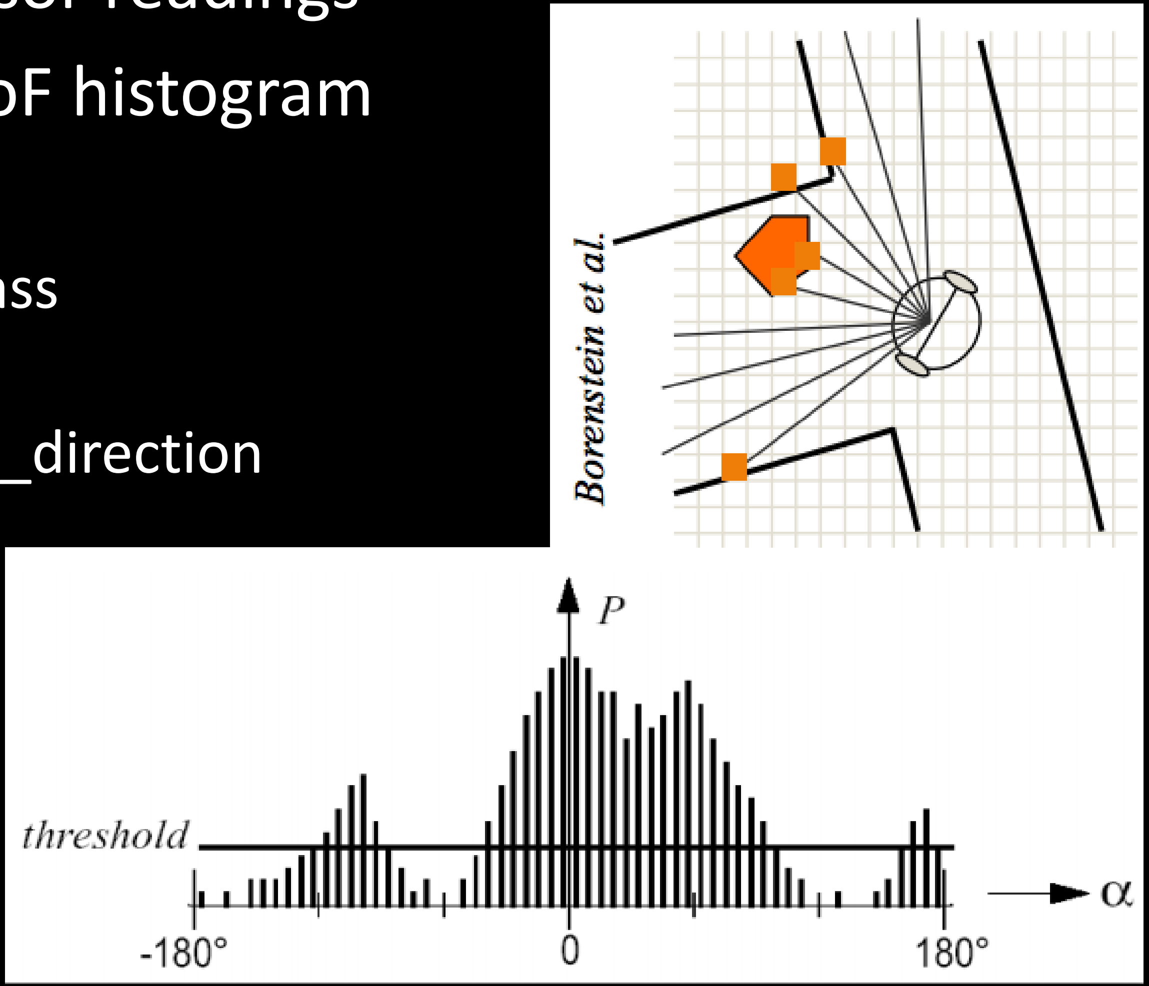


- The robot motion behavior is reactive
- Issues if the instantaneous sensor readings do not provide enough information or are noisy

Vector Field Histograms

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

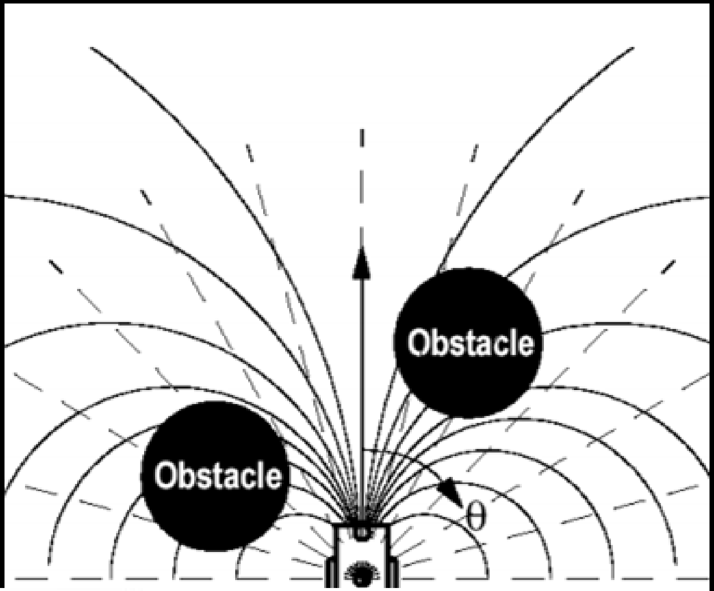
- VFH creates a local map of the environment around the robot populated by “relatively” recent sensor readings
- Build a local 2D grid map → reduce to 1-DoF histogram
- Planning
    - Find all openings large enough for robot to pass
    - Choose the one with the lowest cost, G
    - G = a*goal_direction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

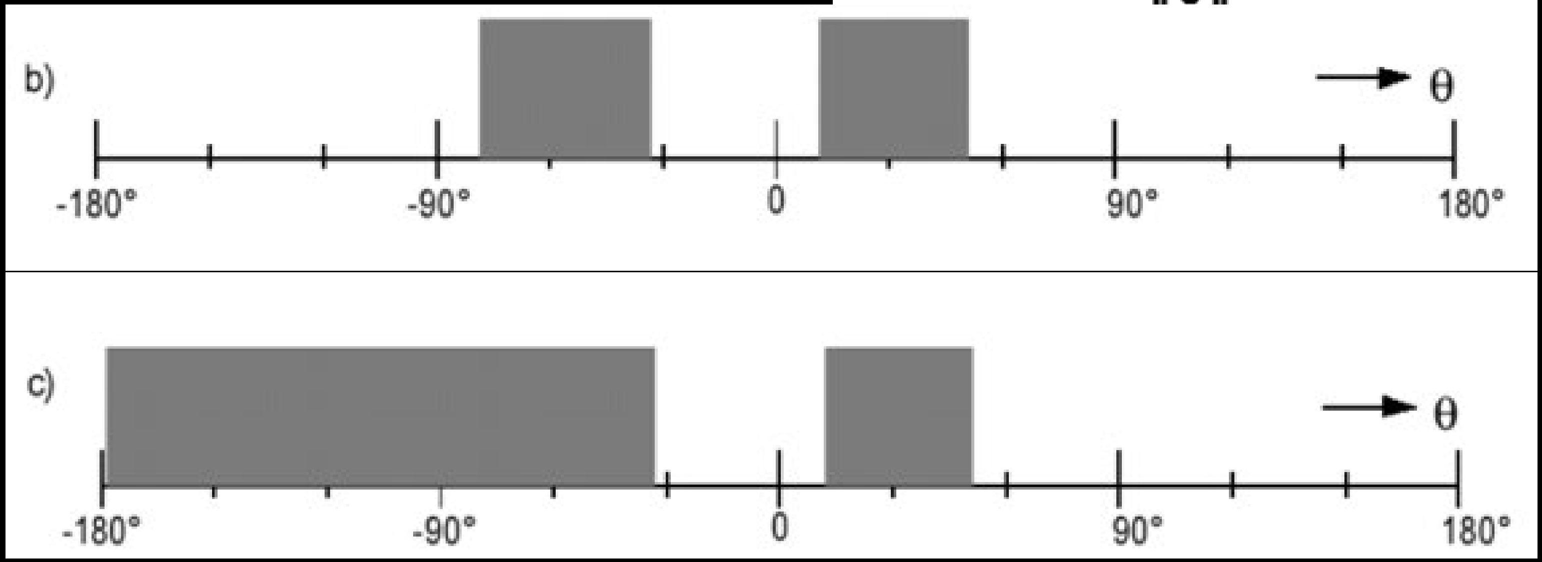


http://www.personal.umich.edu/~johannb/Papers/paper16.pdf

Vector Field Histograms

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

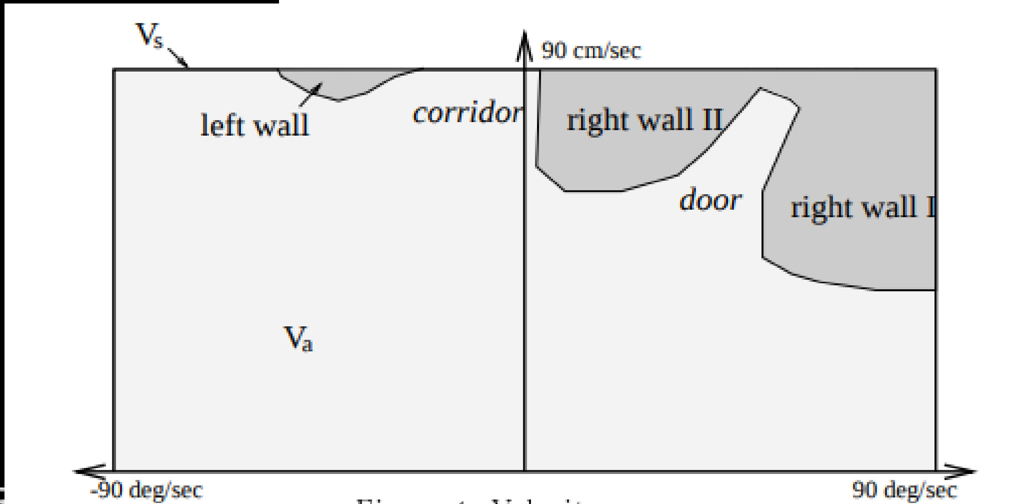
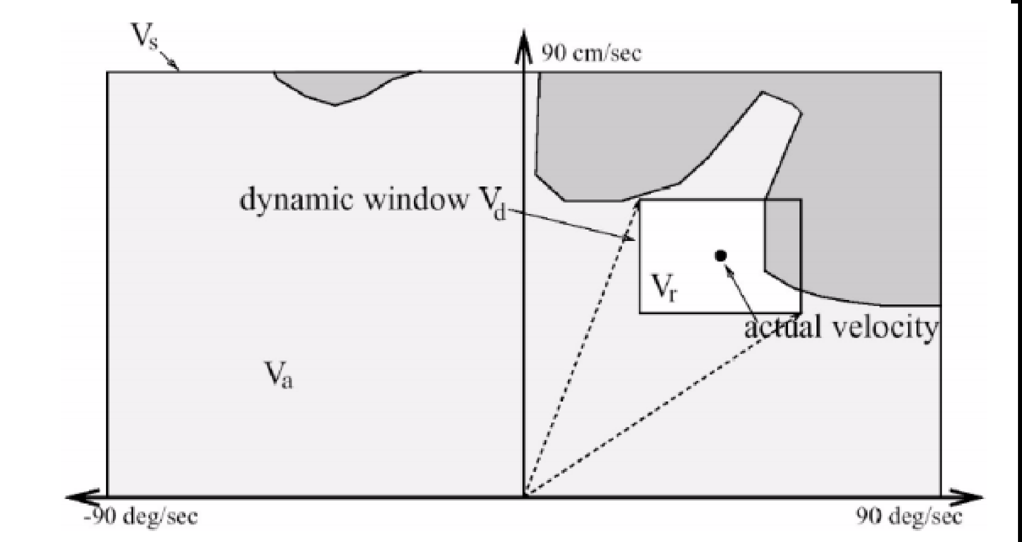


Figure 4. Velocity space



Local Planning Algorithms, Summary

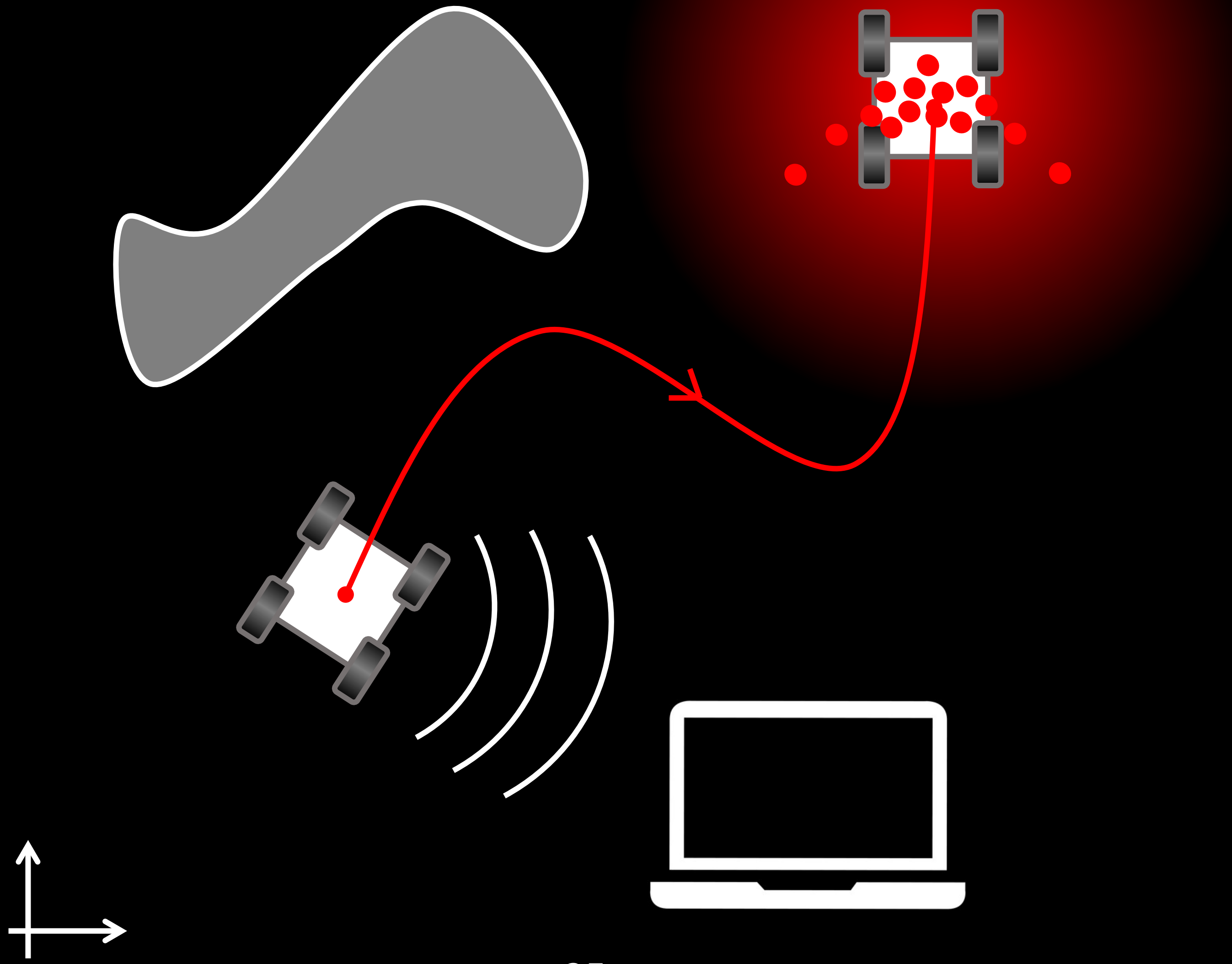
- Bug Algorithms
  - Inefficient, but can be exhaustive
- Vector Field Histograms
  - Takes into account probabilistic sensor measurements
- Vector Field Histograms +
  - Takes into account probabilistic sensor measurements and robot kinematics
- Dynamic Window Approach
  - Takes into account robot dynamics

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Global Localization

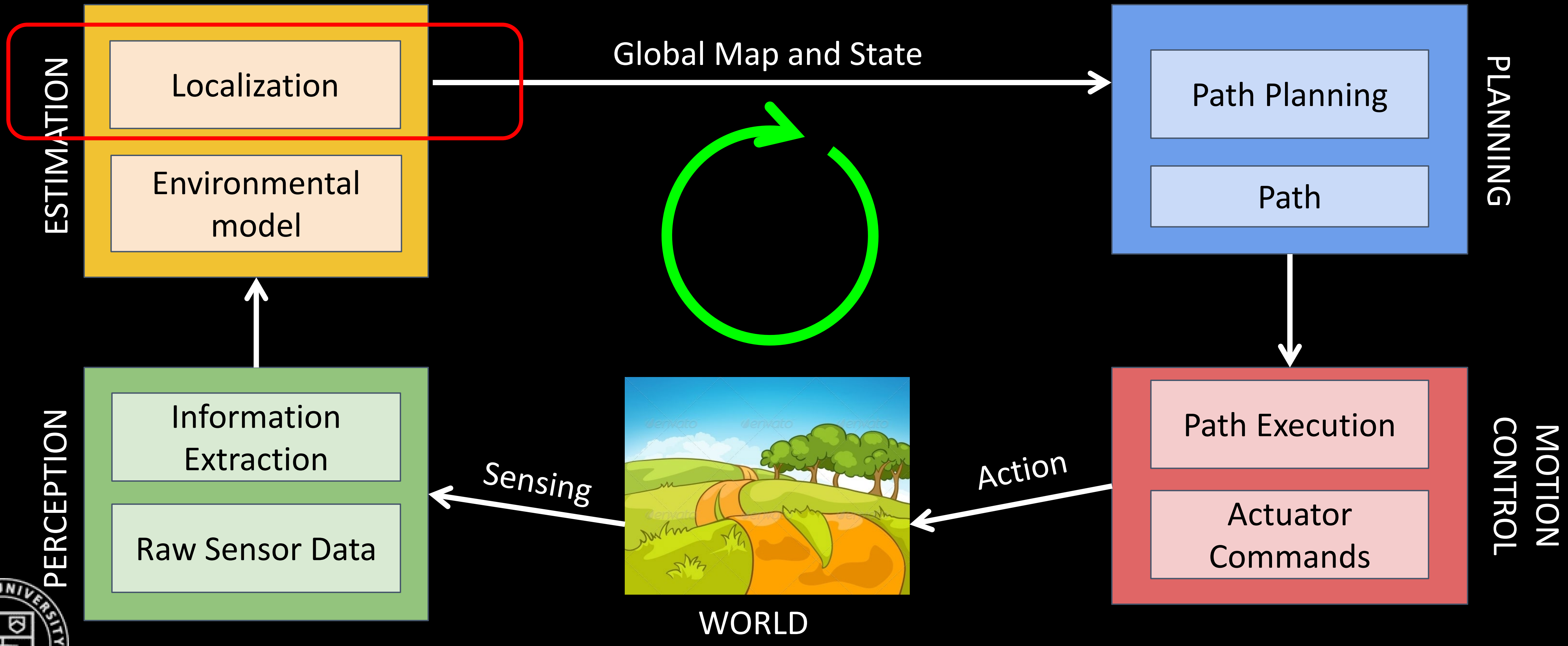
Outline of the next module on Navigation

- 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*



Navigation

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



Localization Problem

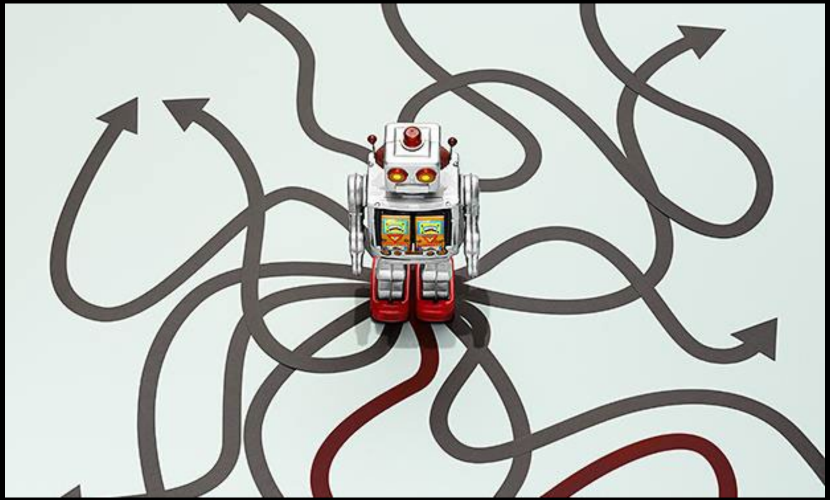
Position Tracking

- Initial robot pose is known
- Either deterministically (odometry) or through Bayesian statistic (motion and sensor models)
- 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

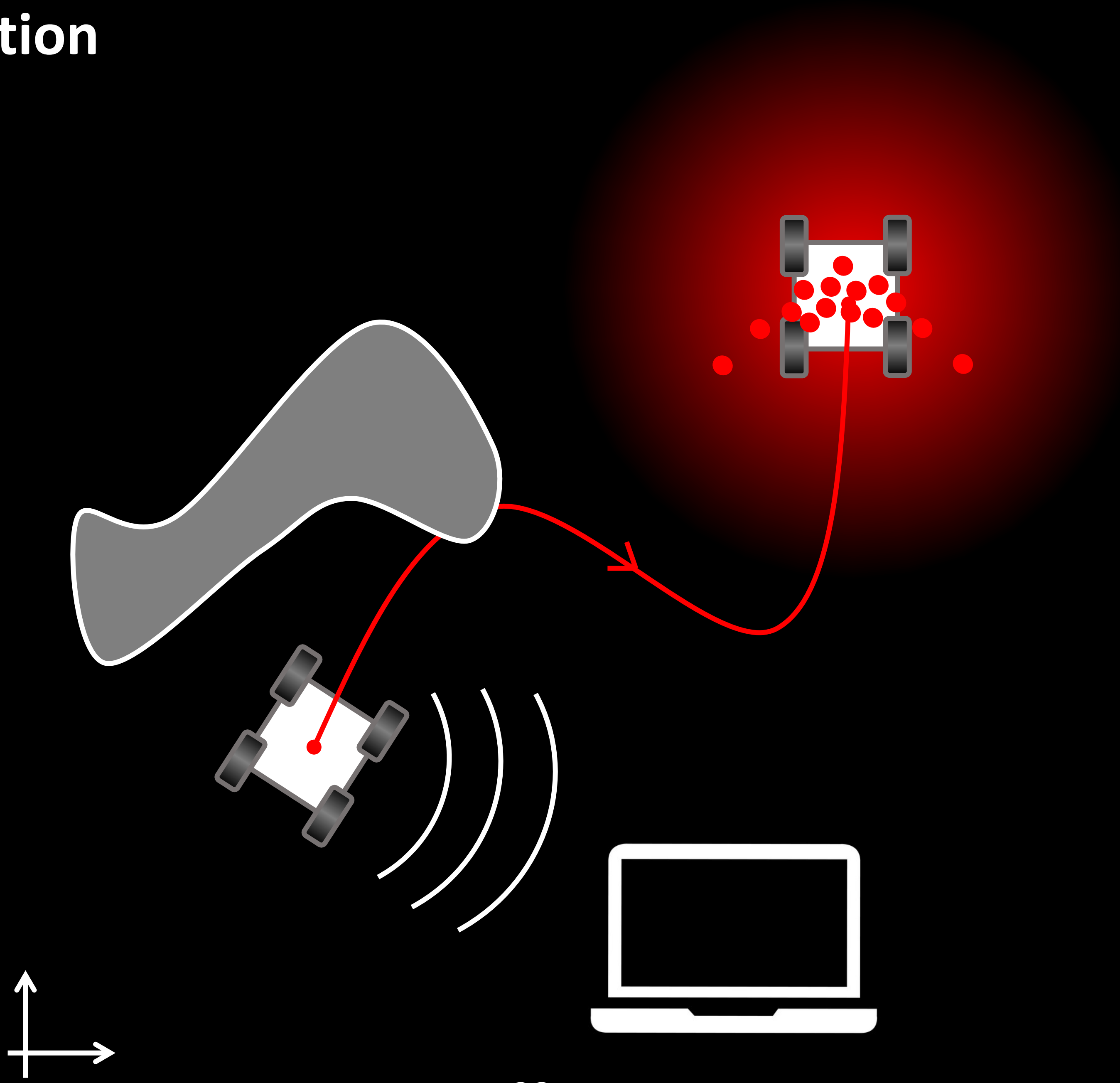
- Initial robot pose is unknown
- Need to estimate position from scratch
- A more difficult "global" problem, where you cannot assume boundedness in pose error

kidnapped robot problem



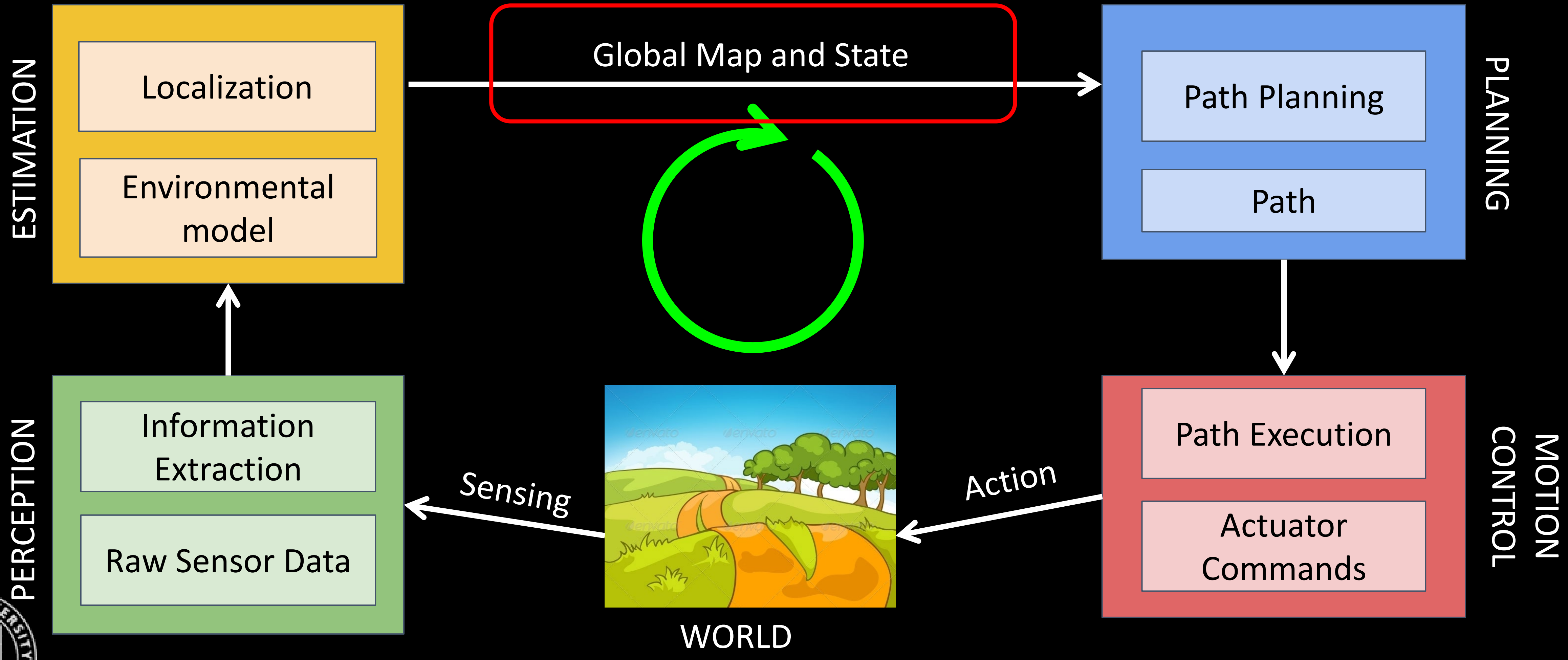
Outline of the next module on Navigation

- 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*



Navigation

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



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Map Representations

Map Representation

(a) Building plan

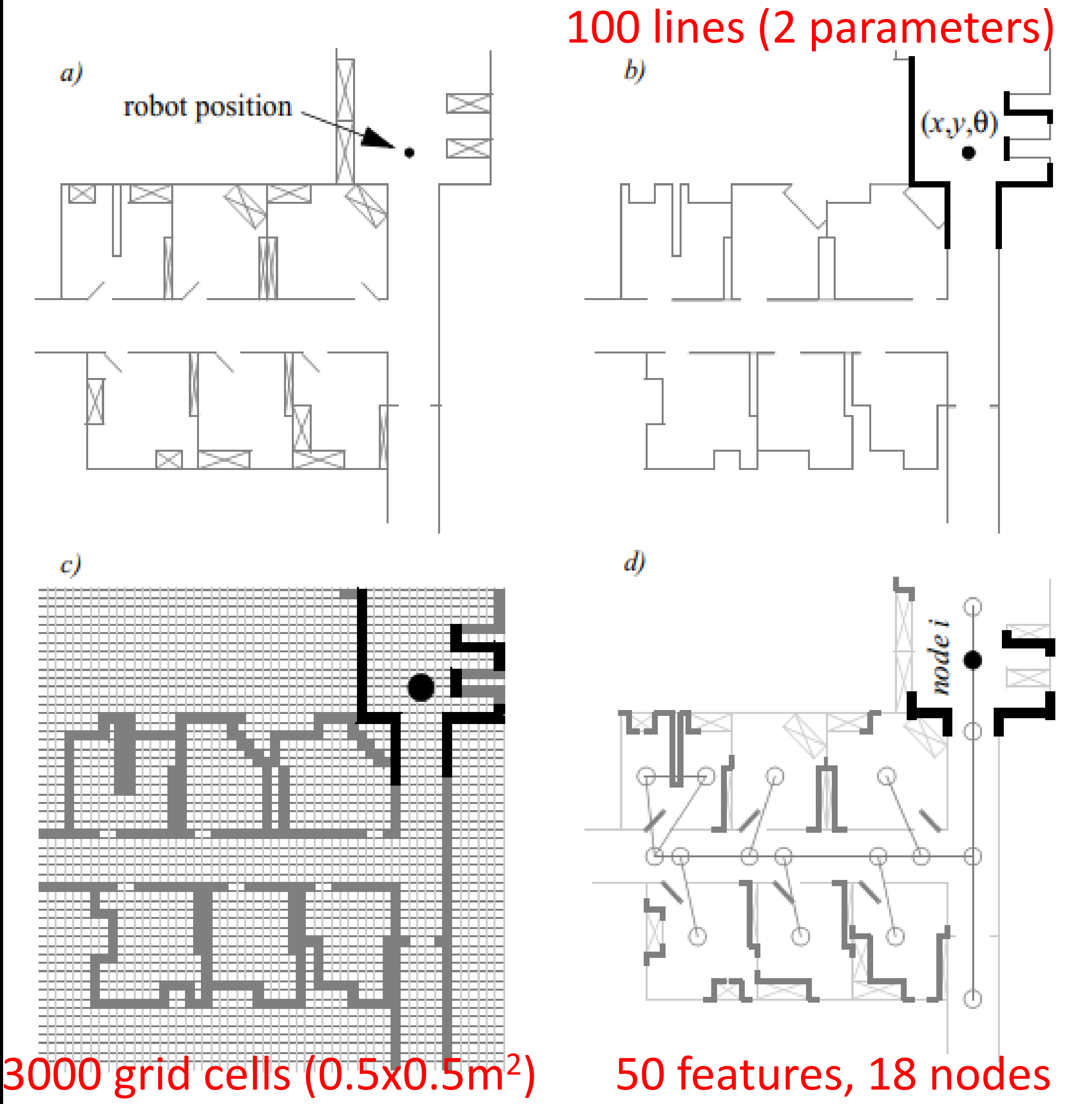
(b) line-based map

(c) occupancy grid-based map

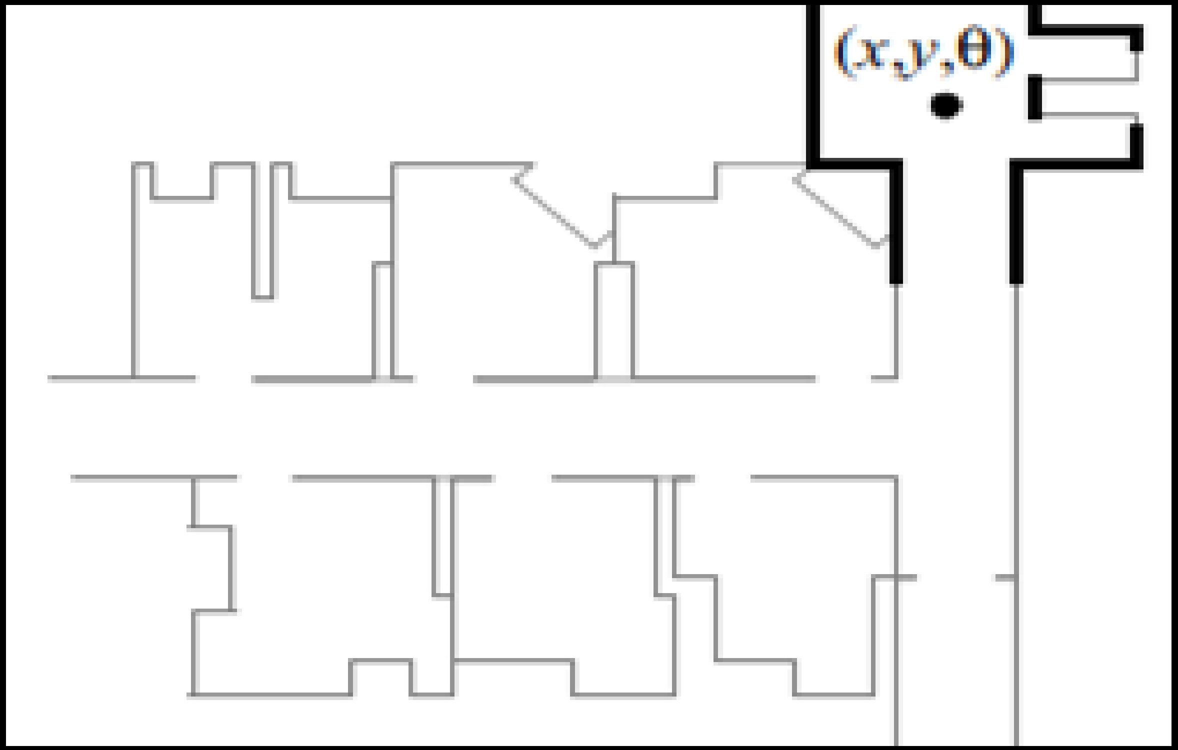
(d) topological map

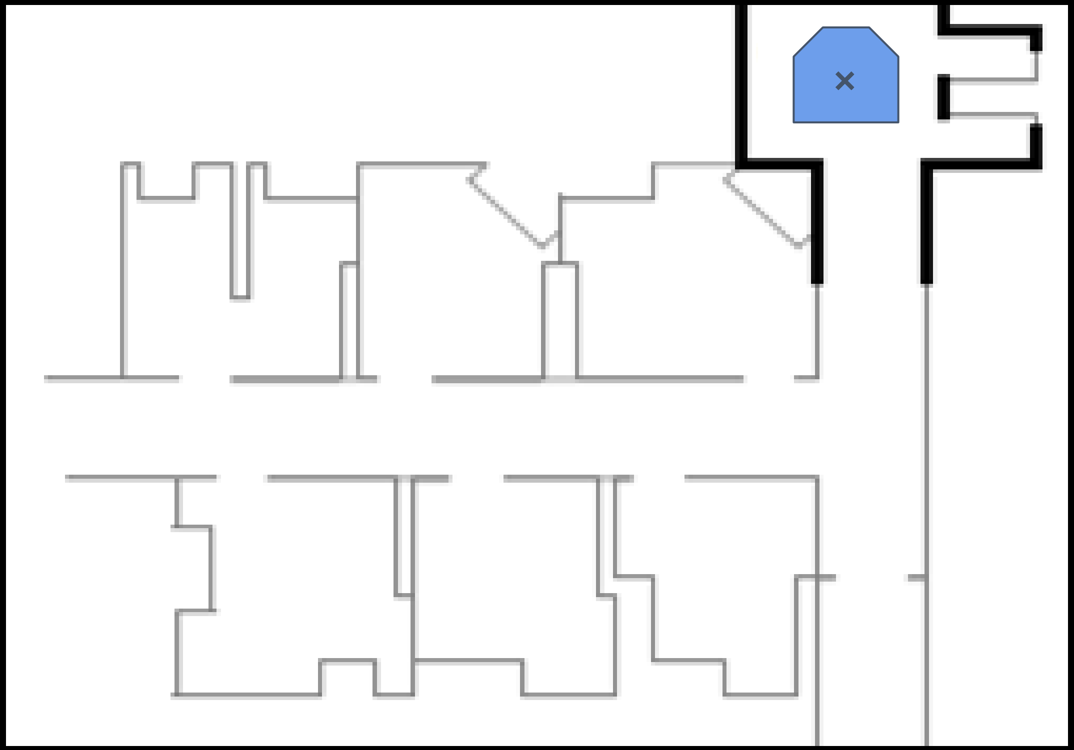
Important properties

- Memory allocation
- Computation
- Robot pose



What if the robot is not a point?

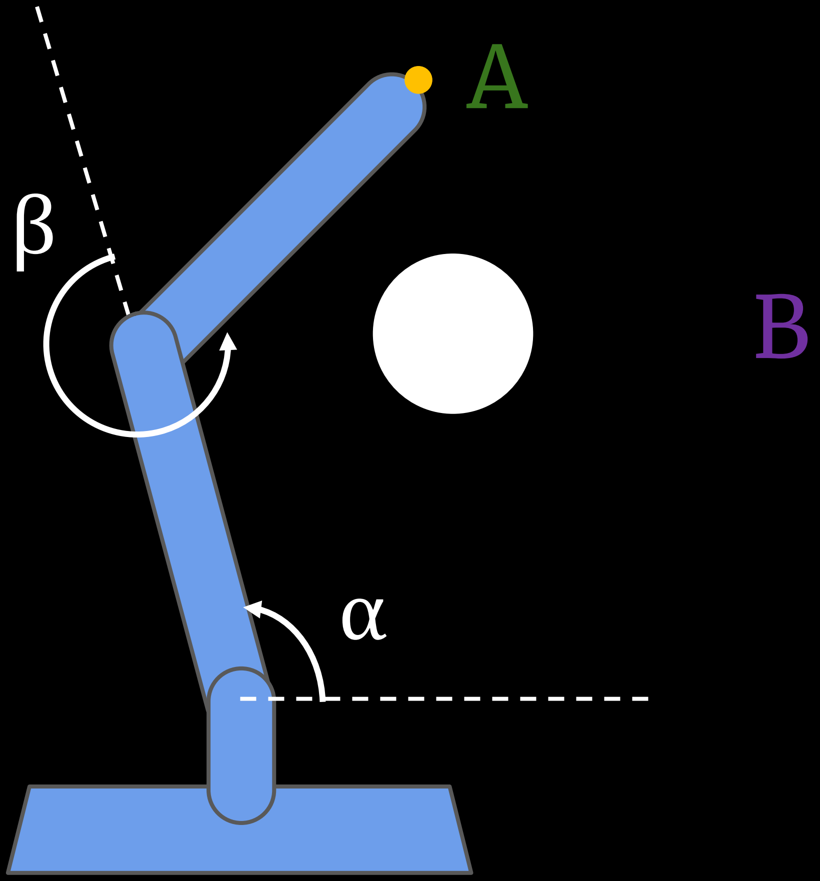


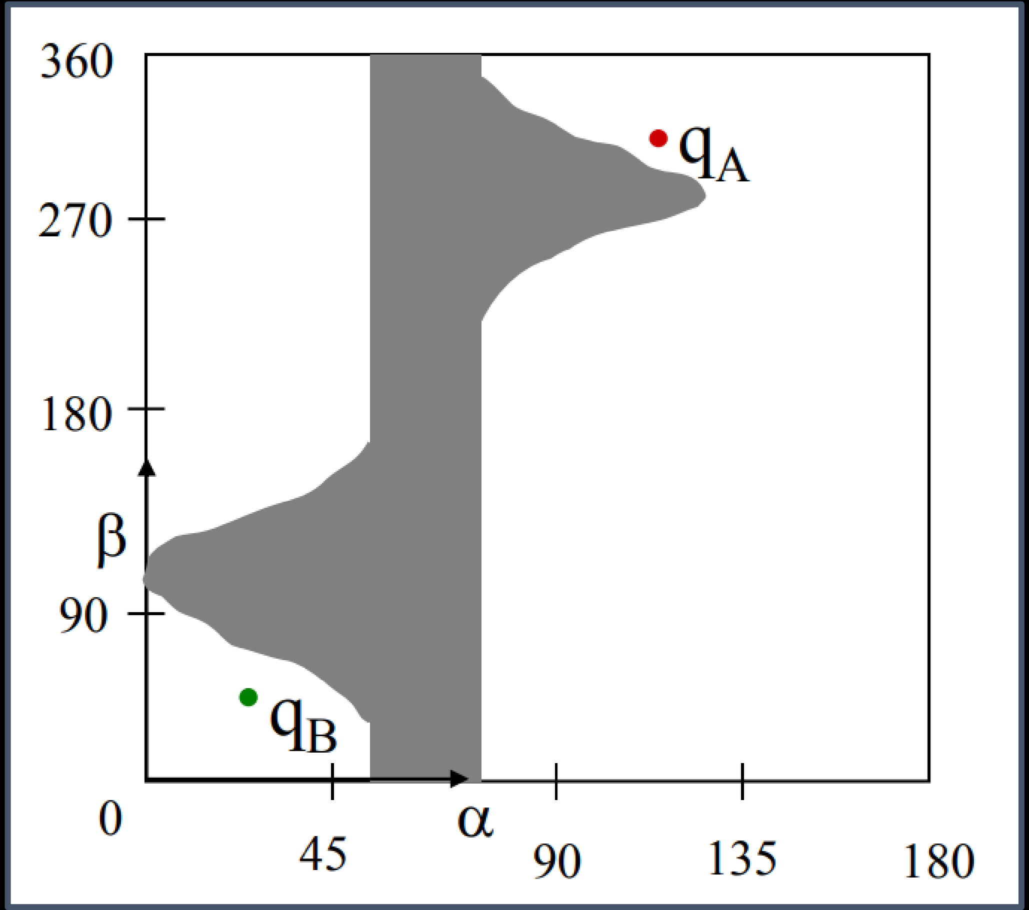


Configuration Space

- 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

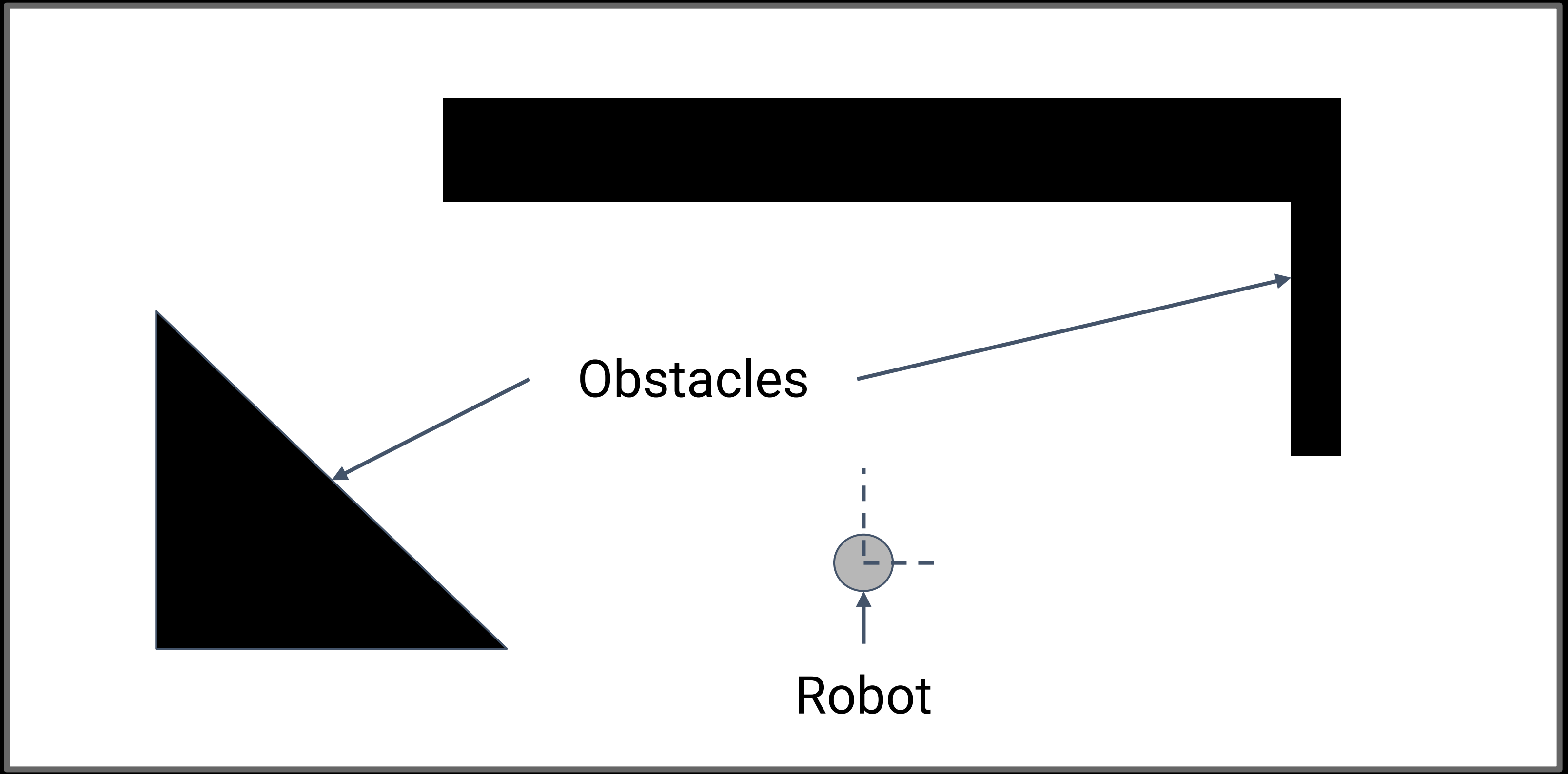




Configuration Space

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

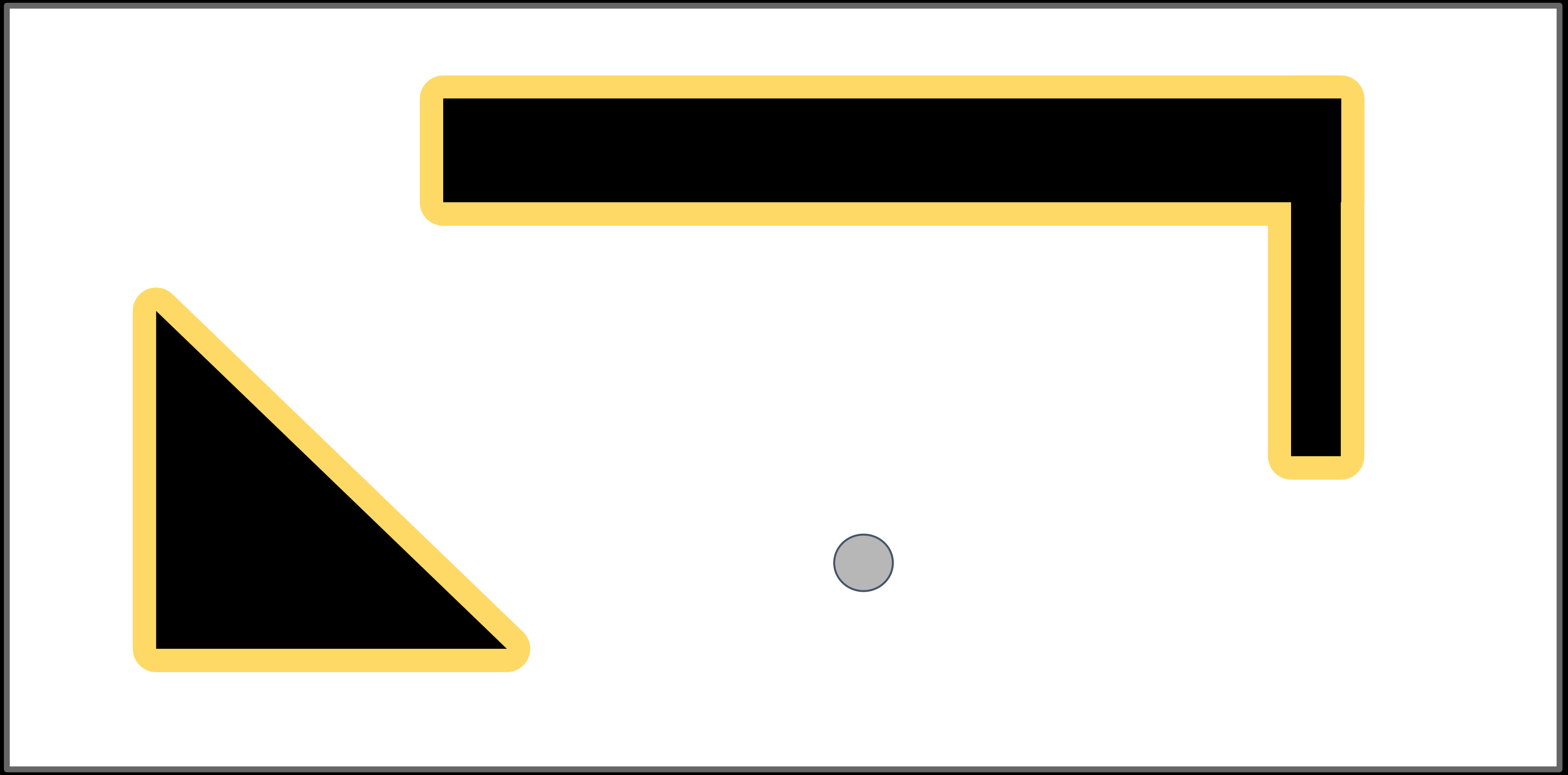
Ex 2: Circular root in 2D world



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

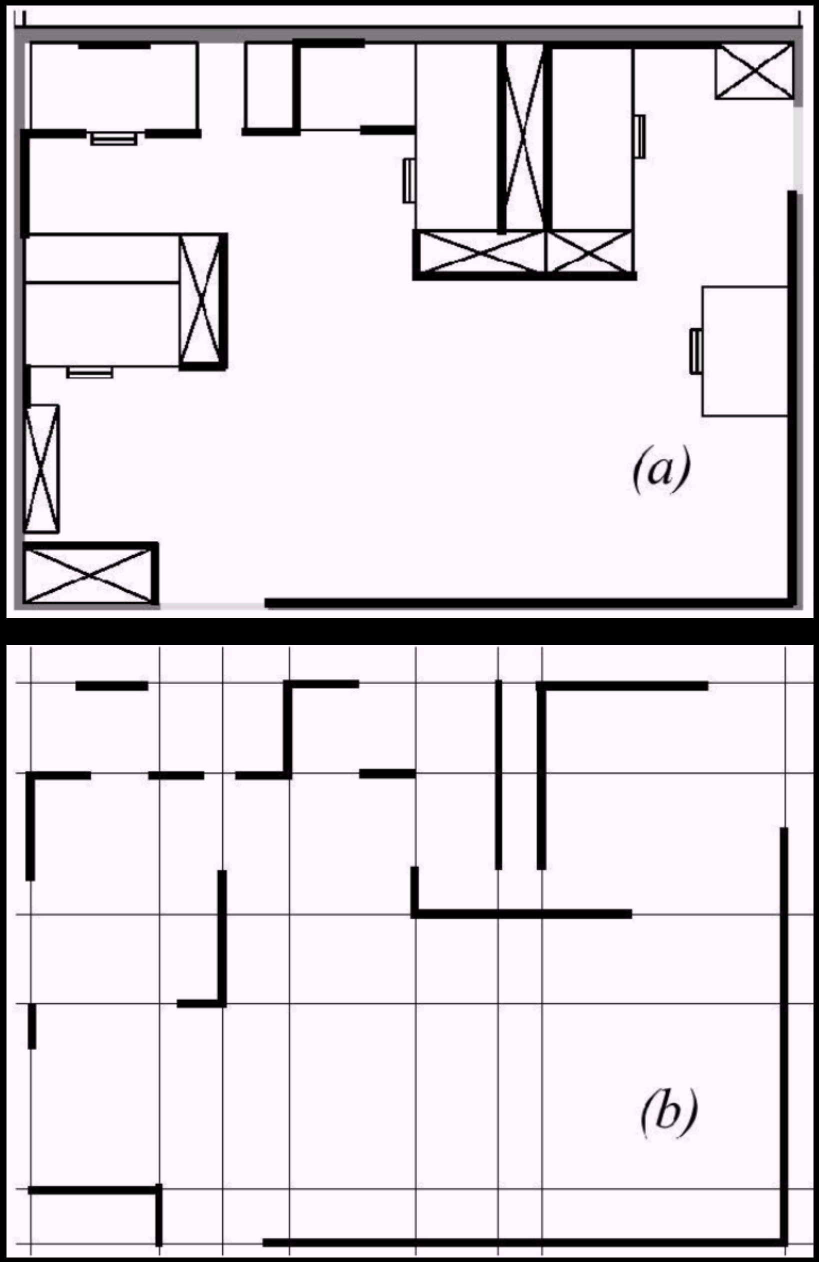
Map Representation Considerations

Summary

- 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

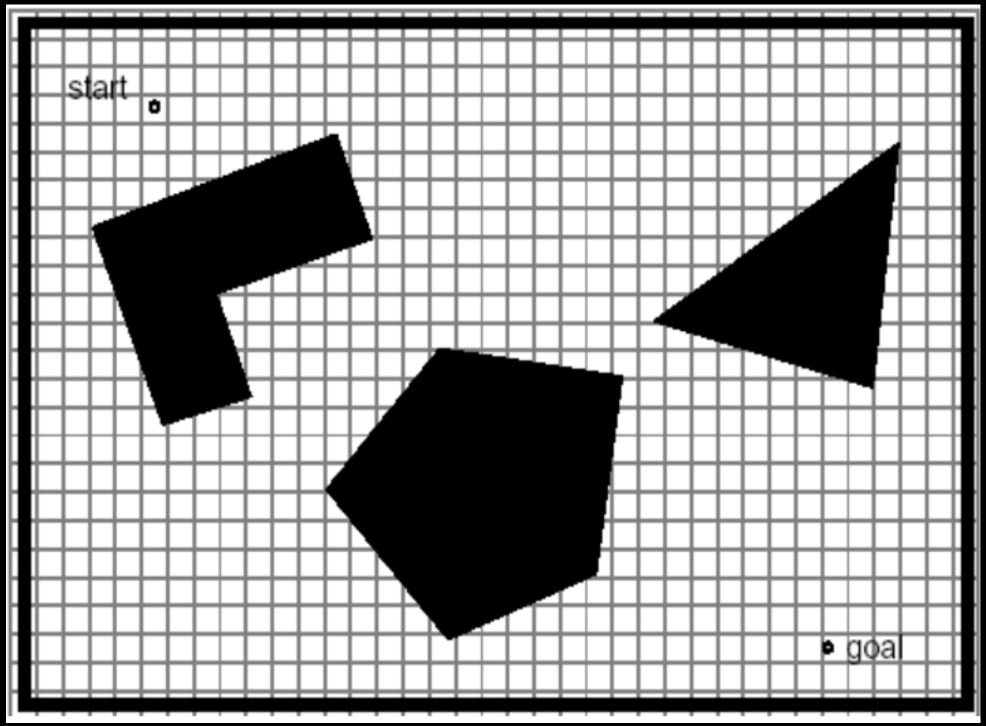
Continuous Representations

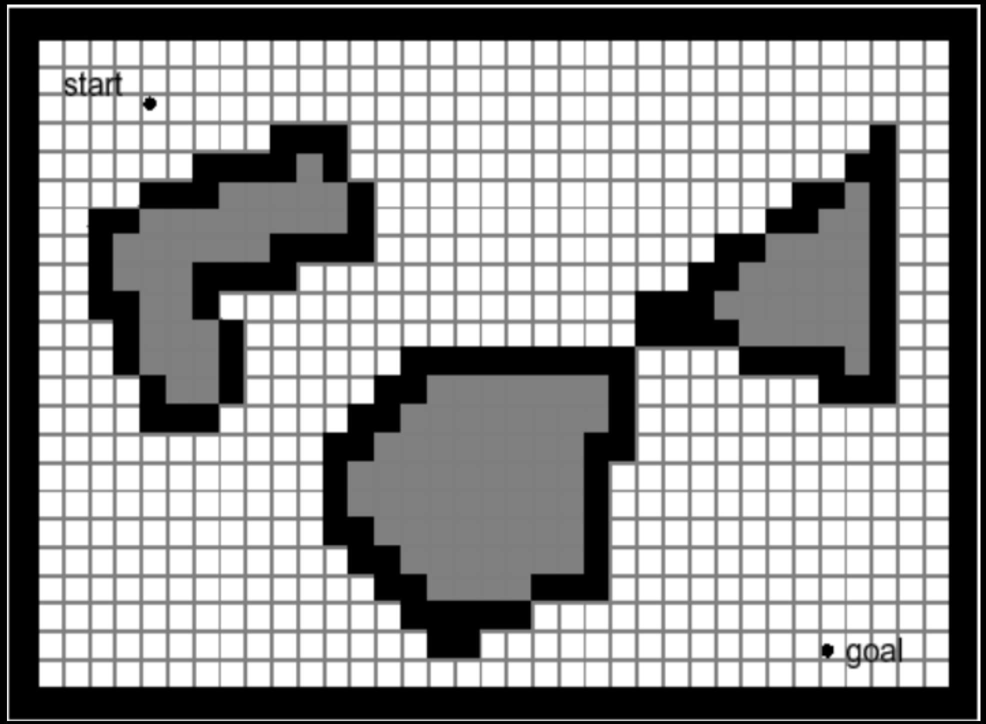
- Exact decomposition of the environment
- Used mainly in 2D representations
- Closed-world assumption
- Storage proportional to object density
- Example: Continuous line representations
  - Using range finders, we can extract lines/line segments in the environment



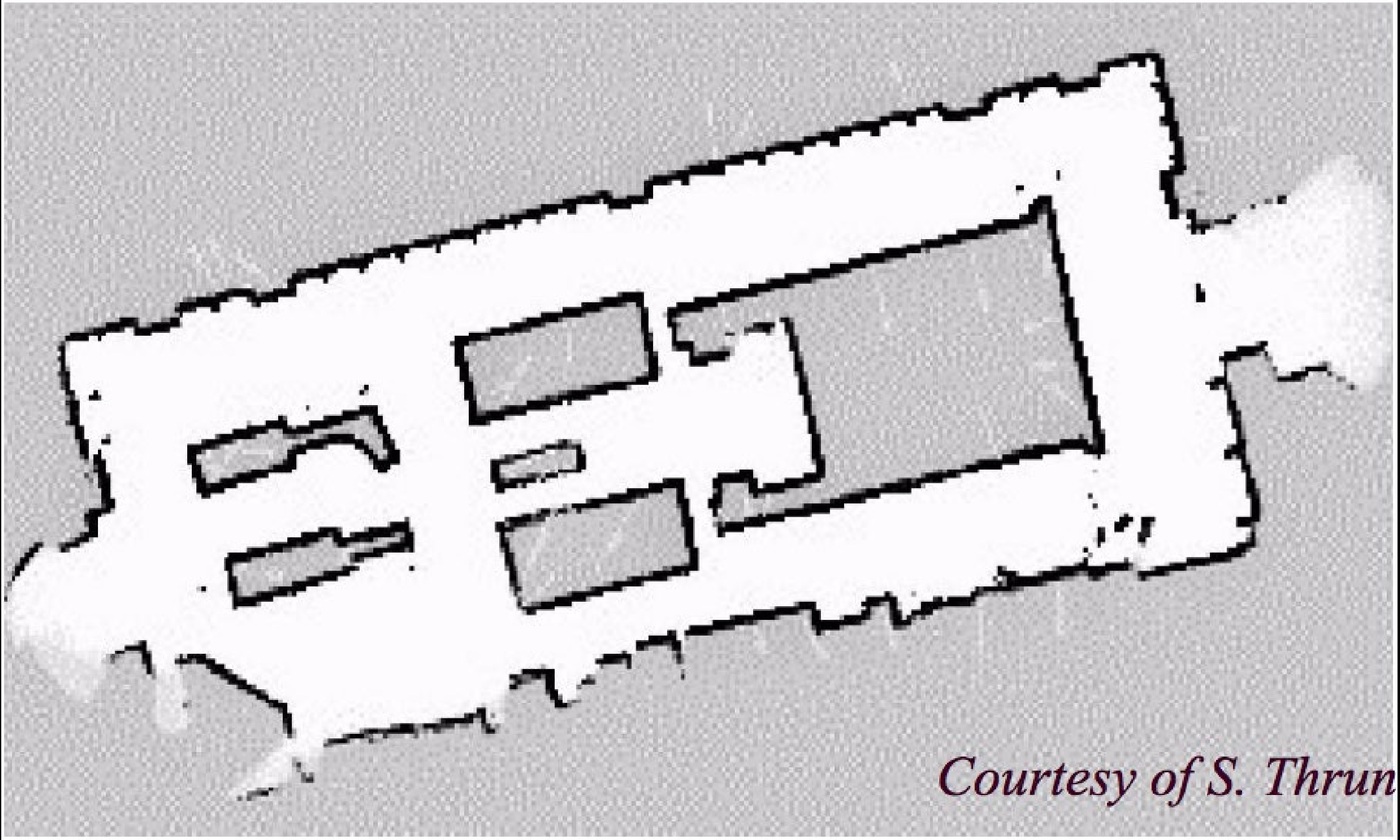
Fixed Decomposition

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



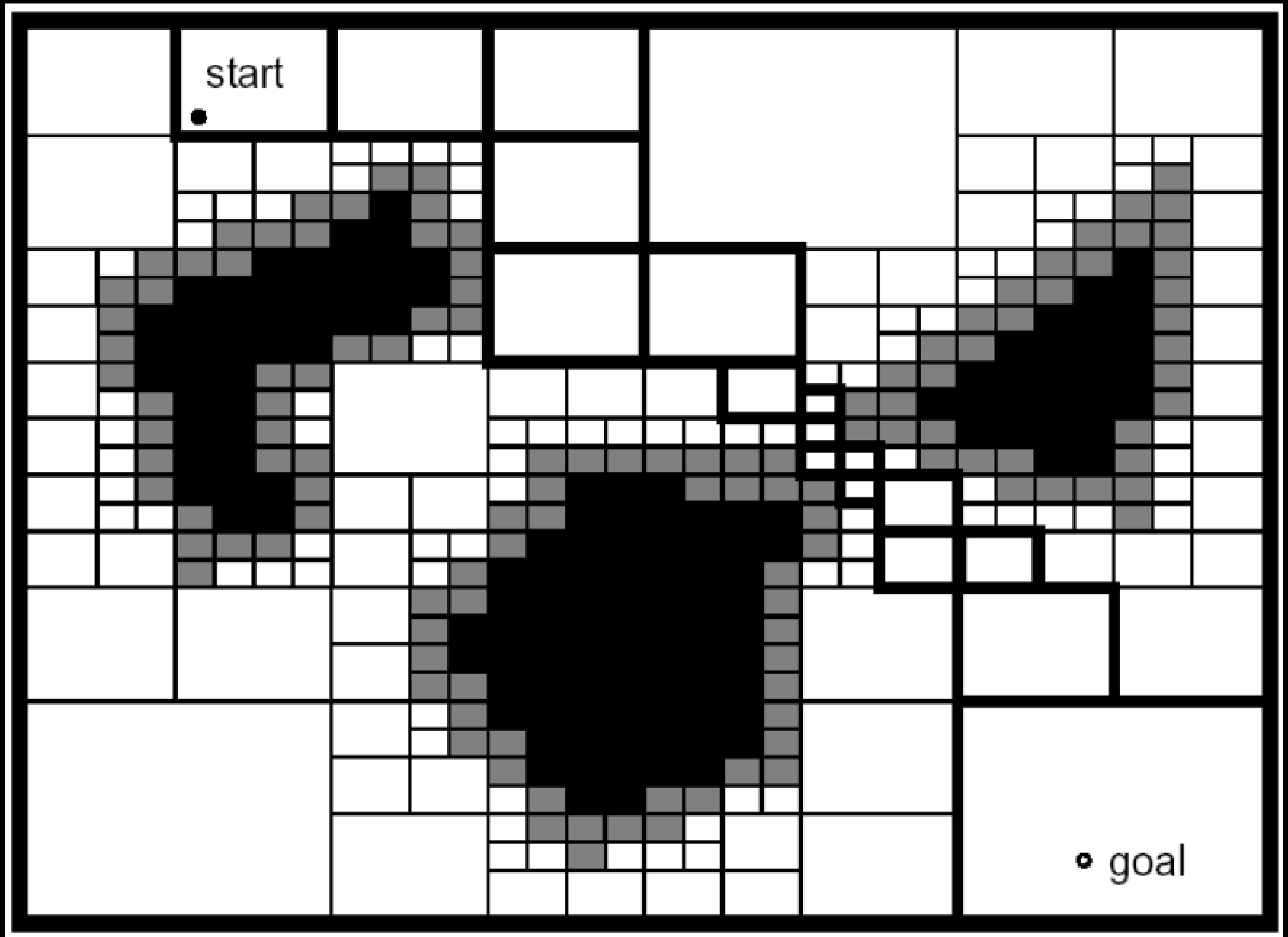


Fixed Decomposition



Adaptive Cell Decomposition

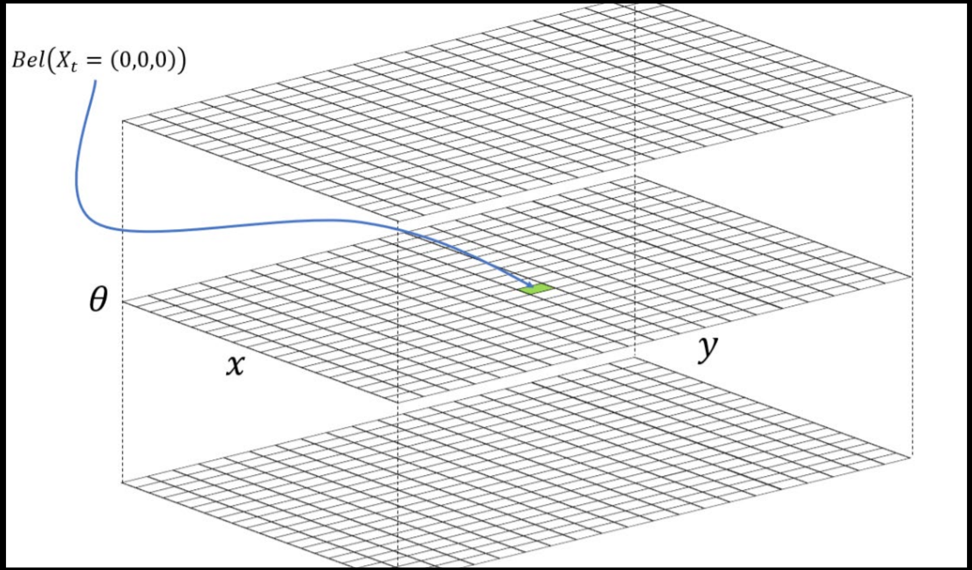
- Adapt cell size to features

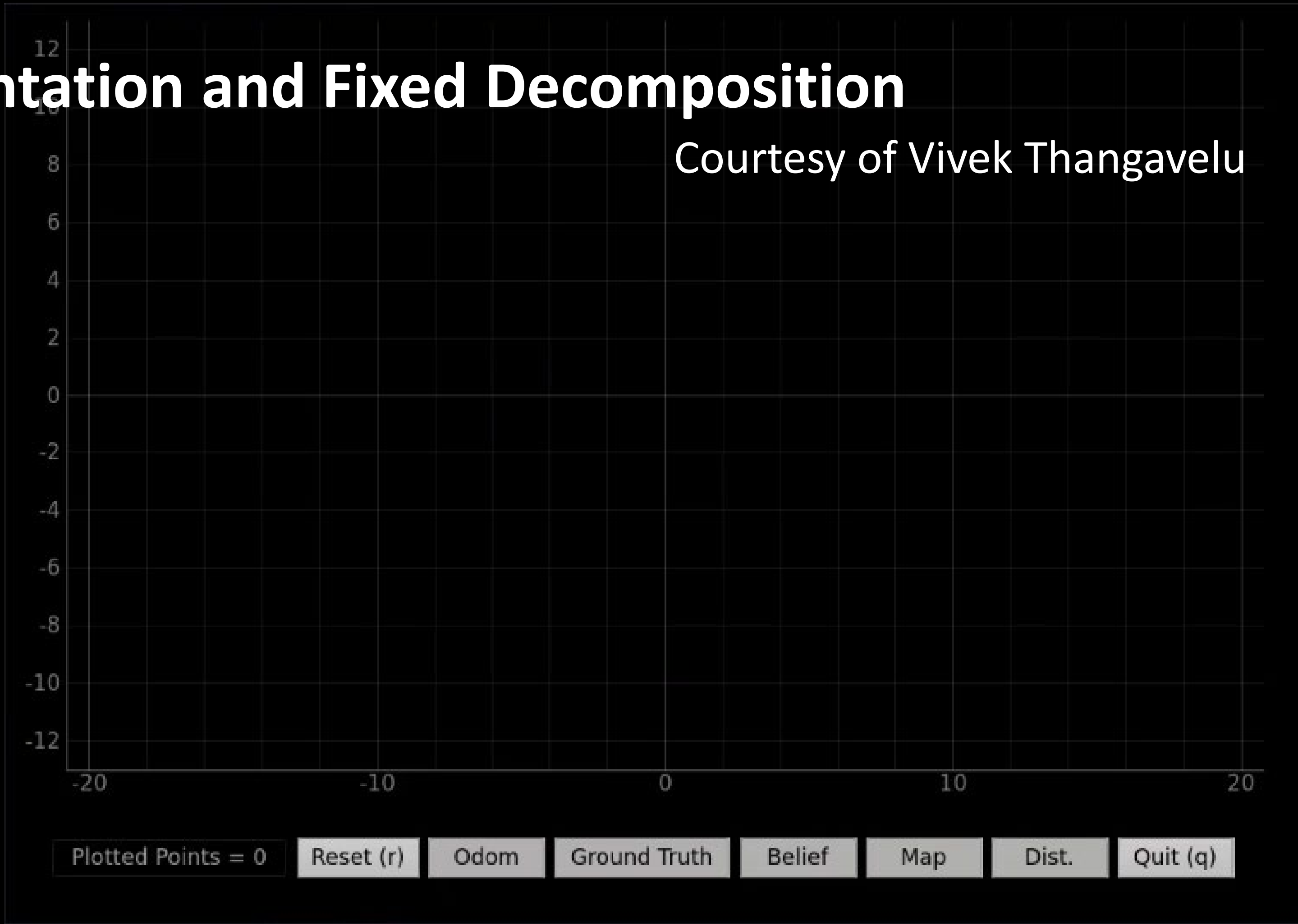


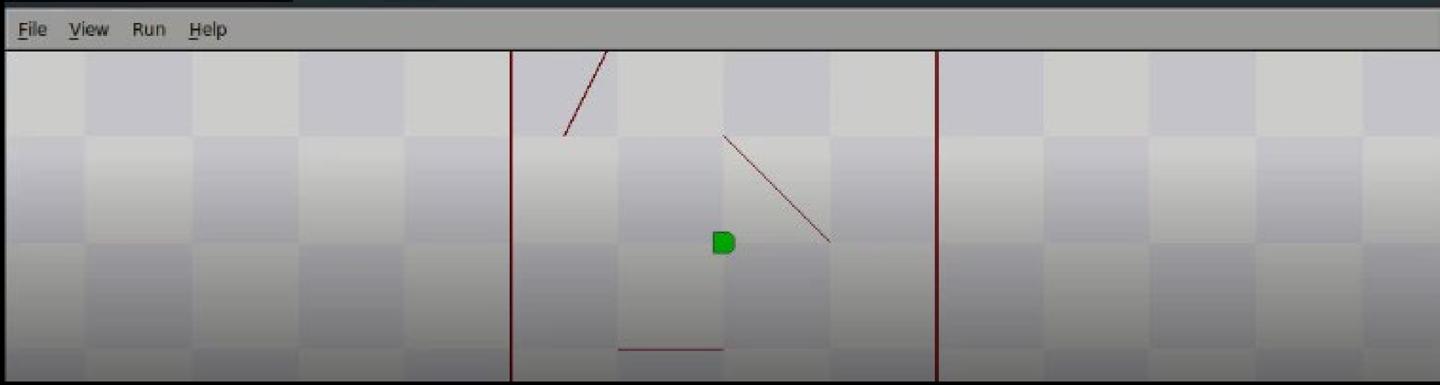
Lab 9-12: Combo of Linear Representation and Fixed Decomposition

Courtesy of Vivek Thangavelu

- Map is represented by lines
- Robot pose is represented by a fixed decomposition of (x,y,theta)





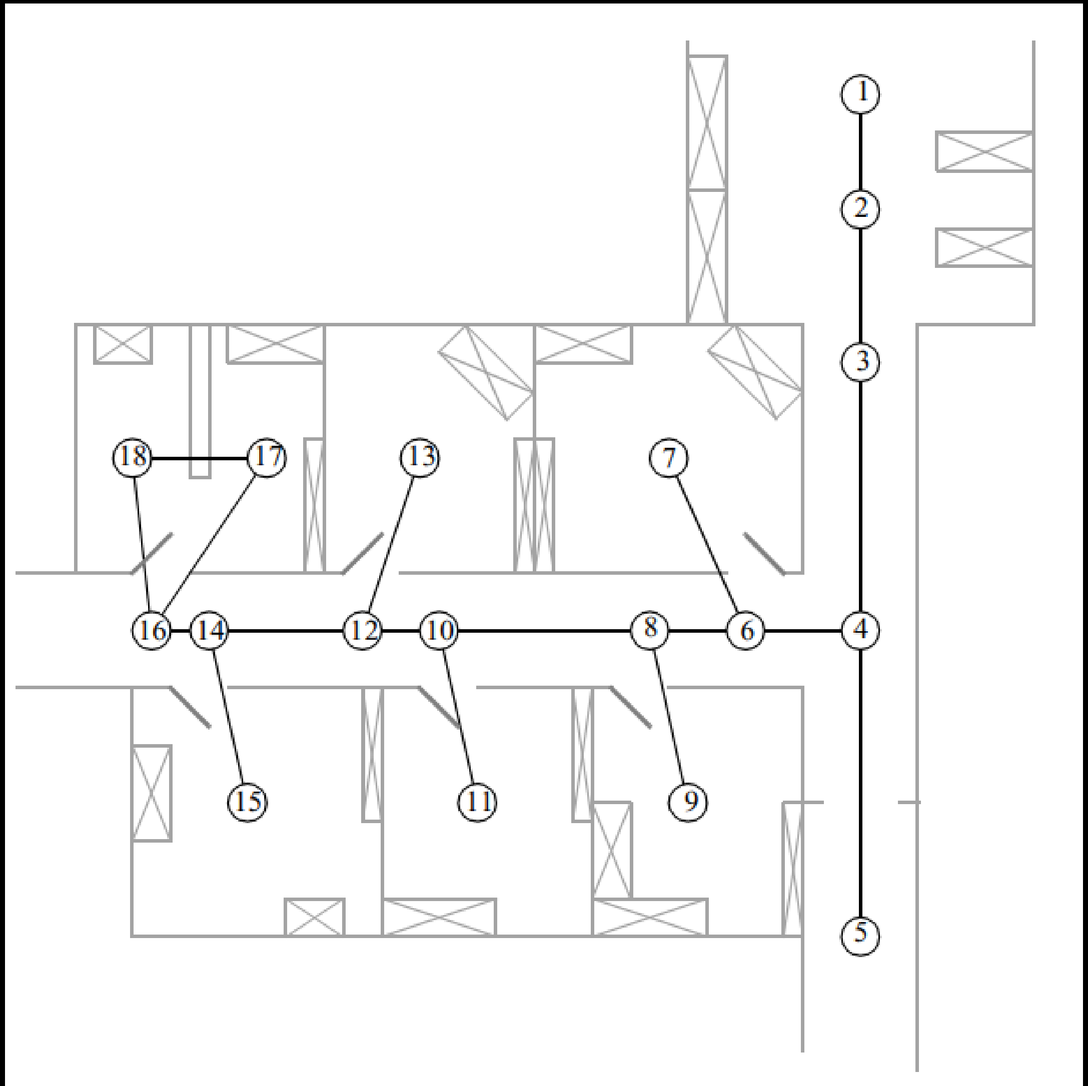


Robots in 3D Environment

- 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
  - Map representations
    - Continuous
    - Discrete
    - Topological
- Maps as graphs
  - Graph Search Algorithms
    - Breadth First Search
    - Depth First Search
    - Dijkstras
    - A*