ECE 484: Principles of Safe Autonomy (Fall 2025)
Lecture 17: Search and Planning: Hybrid A*, PRM

Professor: Huan Zhang

https://publish.illinois.edu/safe-autonomy/

https://huan-zhang.com

huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025

Announcements

- Project checkpoint: 11/13 or 11/14 (check CampusWire)
  - No regular lecture on 11/13 (next Thursday)
- Midterm 2: 11/20
  - Filtering/Localization + Planning
- Midterm 2 Review: 11/18 during lecture
  - Will be very helpful for passing the exam! Please be sure to attend
- Guest Lectures (on Zoom): link posted on course schedule website
  - Hongge Chen (Cruise): 12/2
  - Zhouxing Shi (UC Riverside): 12/9
  - If you attend both guest lectures fully, I will give you extra 1 point for your final class grade (attendence will be tracked on Zoom)

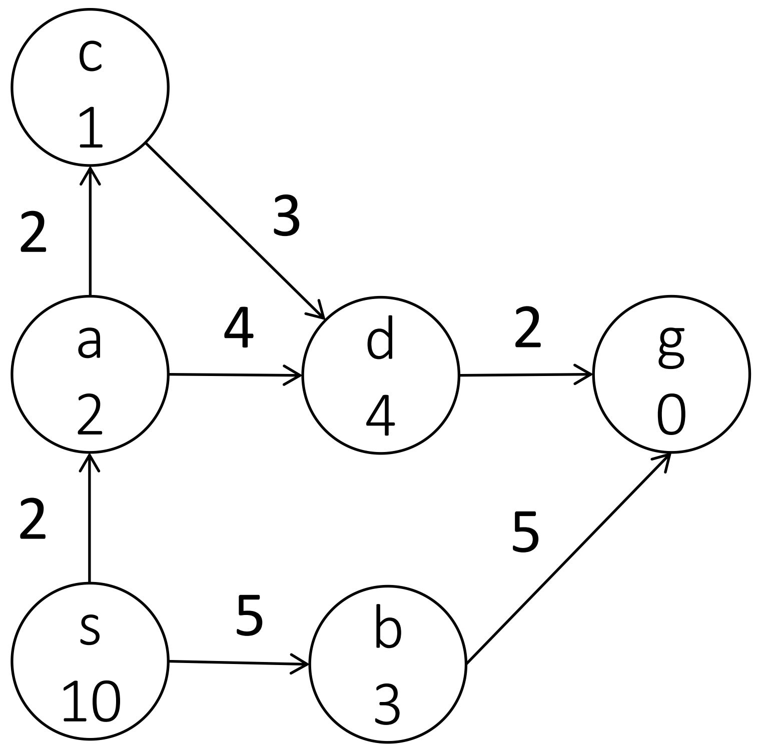
Review: deterministic search

- Deterministic search
  - Uniform Cost Search (uninformed search)
  - Informed search
    - Greedy/Best-first search
    - **A, A* search**
      - Admissible heuristics
      - Design of heuristics
- How to transform a planning problem in to a search problem?
- Know how to apply each search algorithm
- Soundness, completeness, optimality
- Design of heuristics

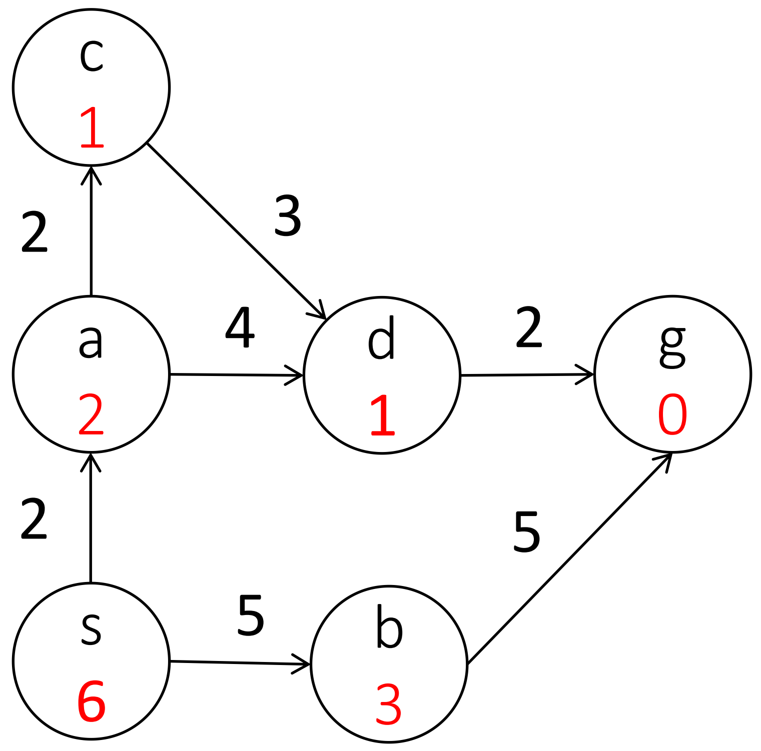
A* search (heuristics must be admissible)

```
$Q \leftarrow \langle start \rangle$                                                    // initialize queue with start
while $Q \neq \emptyset$:
    pick $(and\ remove)$  $path\ P\ with\ lowest$ $estimated\ cost\ f(P) = g(P) + h\big(head(P)\big)\ from\ Q$
    if $head(P) = goal$ then return $P$                          // Reached the goal
    foreach $vertex\ v\ such\ that\ (head(P), v) \in E$, do      // for all neighbors
        add $\langle v, P \rangle\ to\ Q$ ;                                   // Add expanded paths
return $FAILURE$ ;                                                  // nothing left to consider
```

Review: Example of A search



These are the heuristic values we used for greedy search. Are they admissible?

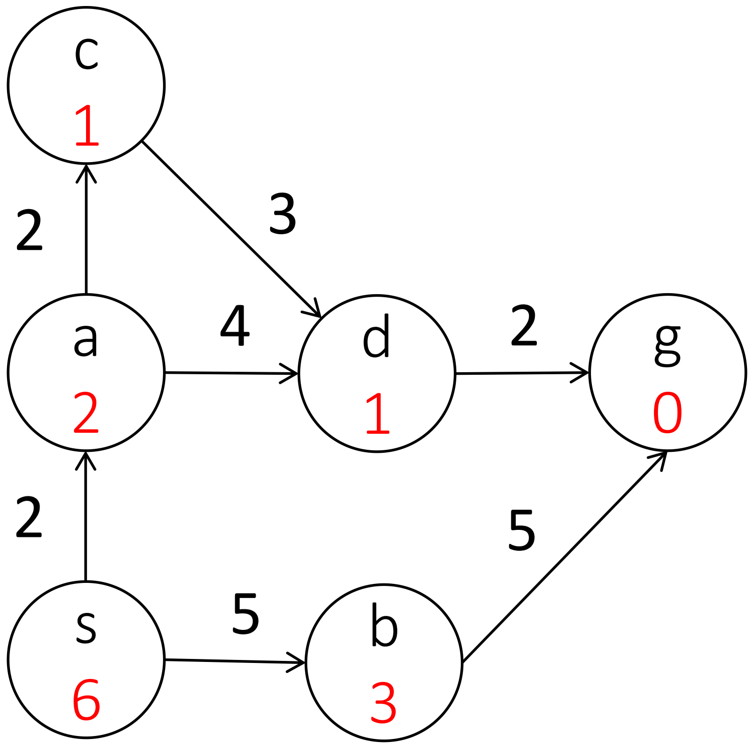


Updated heuristic values

Example of A search

Q:

Path	g	h	f
$\langle s \rangle$	0	6	0+6

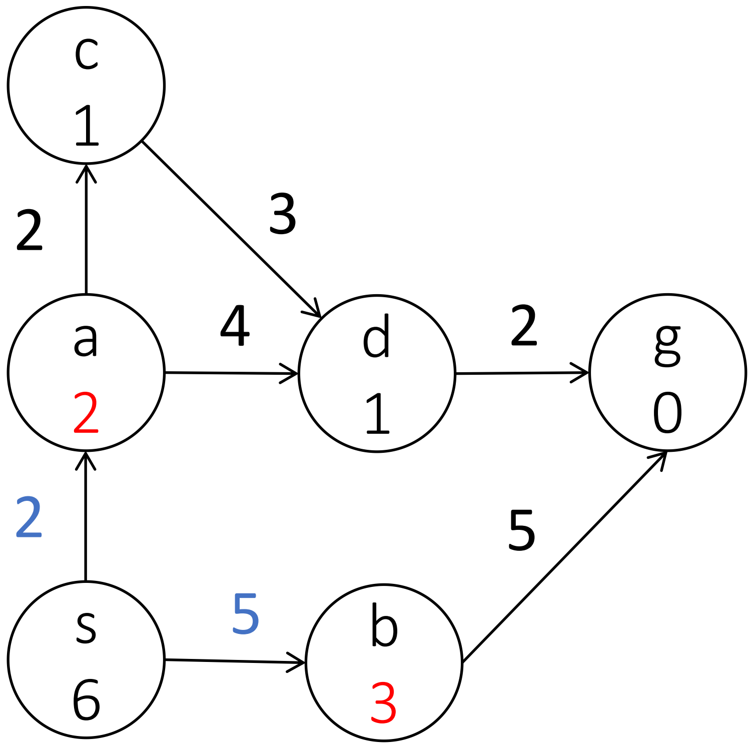


Admissible
heristic values

Example of A search

Q:

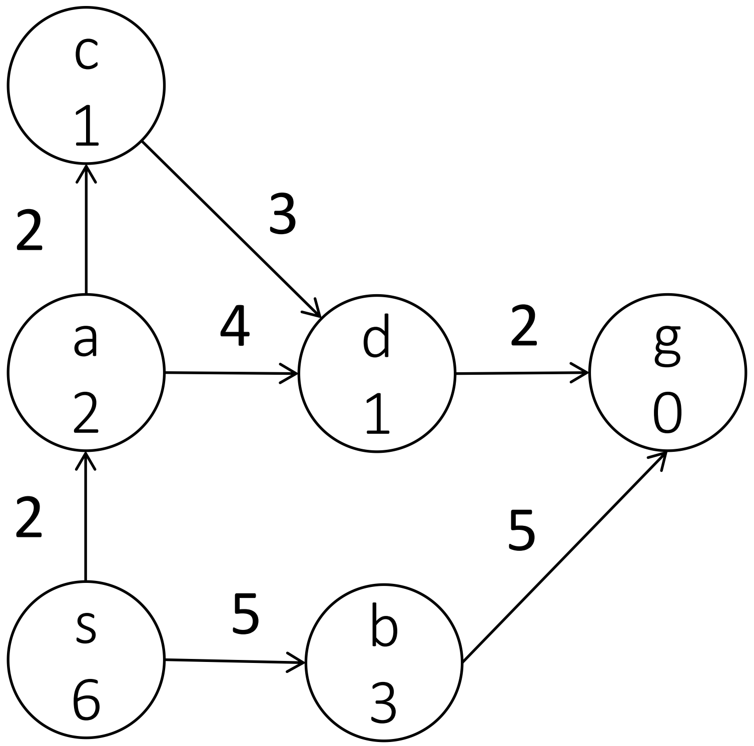
Path	g	h	f
$\langle a, s\rangle$	2	2	2+2=4
$\langle b, s\rangle$	5	3	5+3=8



Example of A search

Q:

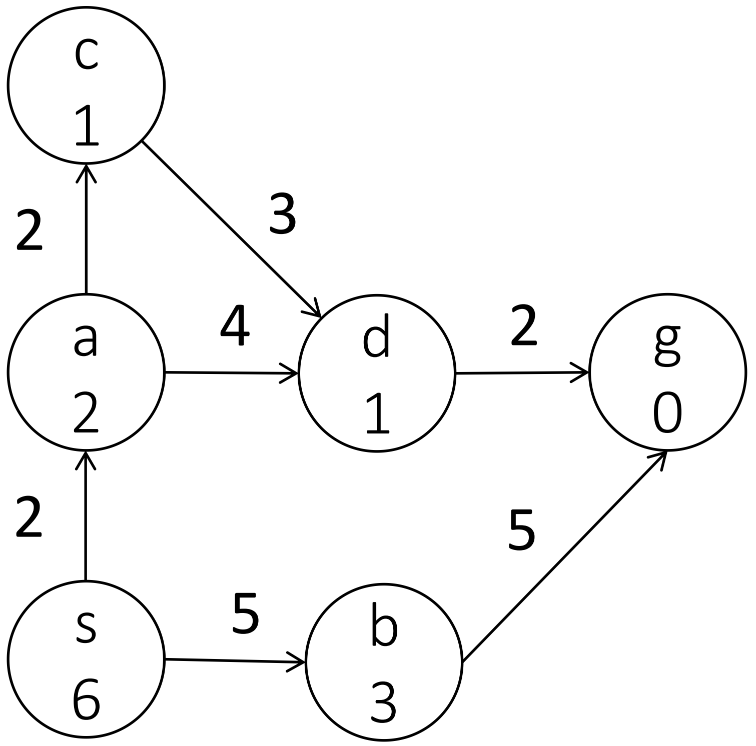
Path	g	h	f
$\langle c, a, s\rangle$	4	1	5
$\langle d, a, s\rangle$	6	1	7
$\langle b, s\rangle$	5	3	8



Example of A search

Q:

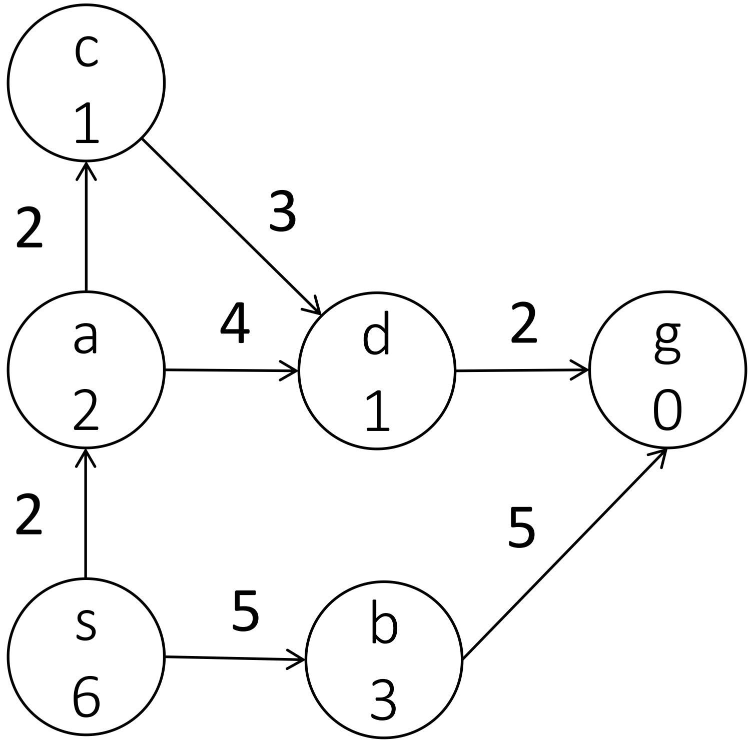
Path	g	h	f
$\langle d, c, a, s\rangle$	7	1	8
$\langle d, a, s\rangle$	6	1	7
$\langle b, s\rangle$	5	3	8



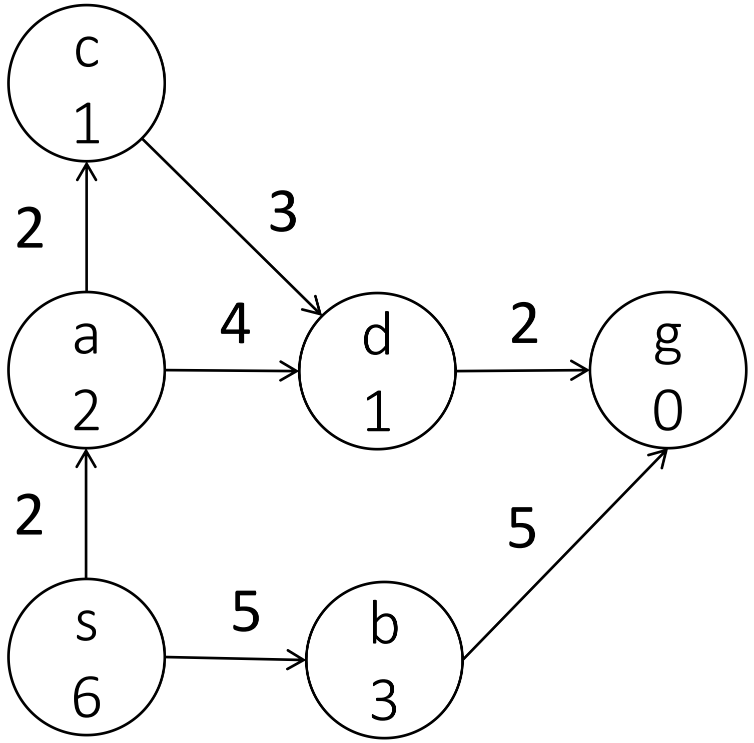
Example of A search

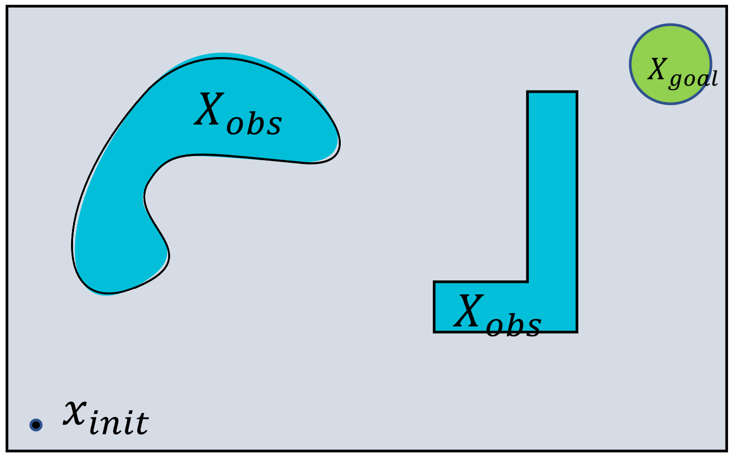
Q:

Path	g	h	f
$\langle g, d, a, s\rangle$	8	0	8
$\langle d, c, a, s\rangle$	7	1	8
$\langle b, s\rangle$	5	3	8



This class: Search in Continuous State Spaces





First approach: *Hybrid* A*

Vehicle models have **continuous state** spaces and kinematic constraints

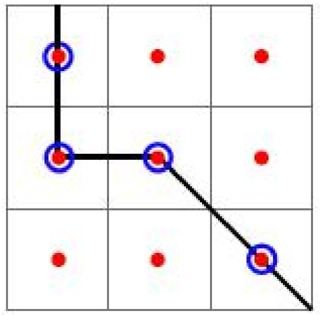
We could represent vehicle state in a *uniform* discrete grid, for example, a 3D grid: $x, y, \theta\ (heading)$

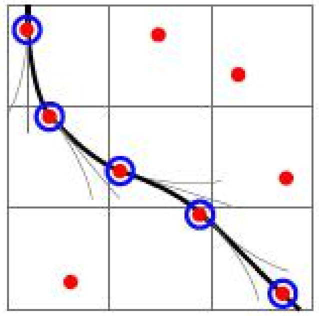
A path (a) over this discrete grid can be a starting plan

But, the discrete path (a) may not be executable by the vehicle dynamics

Hybrid A:* An extension of the traditional A* algorithm that operates on a continuous state space an incorporates **kinematic constraints** (e.g., cars cannot move sideways instantaneously)

(a)



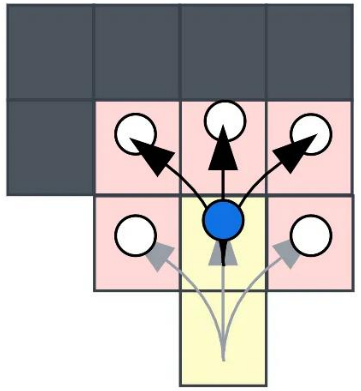


Montemerlo, Michael, et al. "Junior: The stanford entry in the urban challenge." *Journal of field Robotics* 25.9 (2008): 569-597.

Hybrid A* Search (high-level)

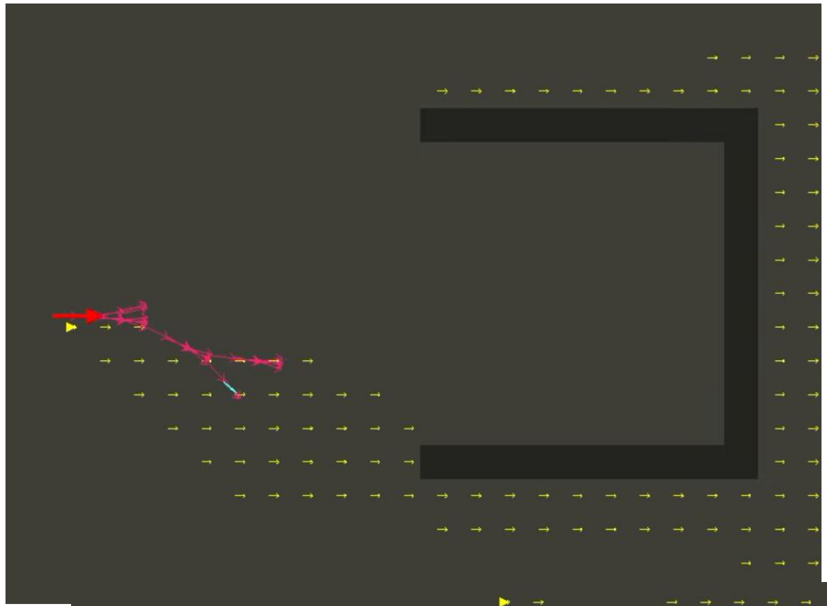
```
O = {($x_0$, $y_0$, $\theta_0$, g=0, h=0)}     // open list
C = {}                                          // closed list, tracking visited states
While O ≠ {}:
   n = (x, y, $\theta$) node with min cost f(n) = g(n) + h(n) in O
   If n = goal, return the path.
   C = C U {n}
   For each motion primitive {m}:
        s = (x', y', $\theta$') = motion(x, y, $\theta$, m)
        g(s) = g(n) + cost(n, s)              // Calculate cost
        h(s) = heuristic(s, goal)
        f(s) = g(s) + h(s)
        If s ∉ C or g(s) is lower than previously recorded cost:
              Add/update s in O with g(s), h(s)
Return failure if O is empty
```

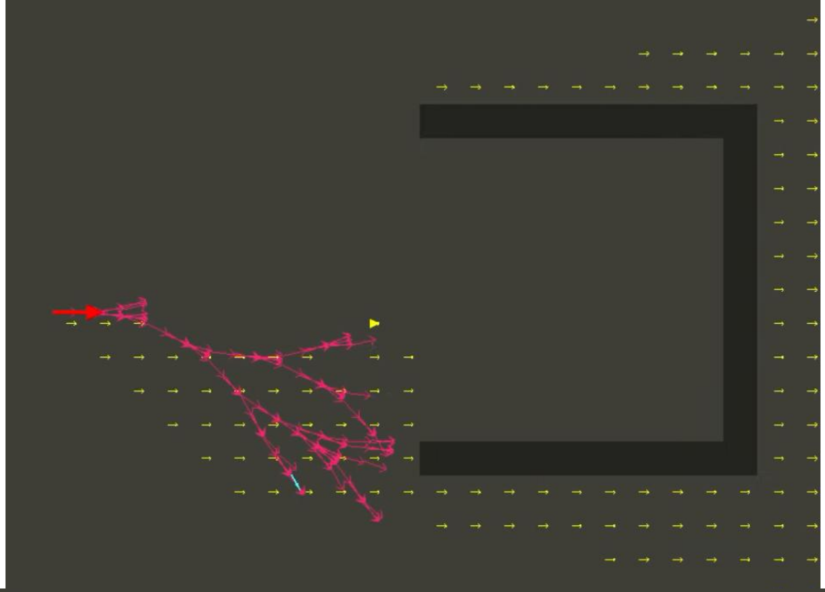
Input: Source node x_0, y_0, θ_0, goal node, Motion primitives

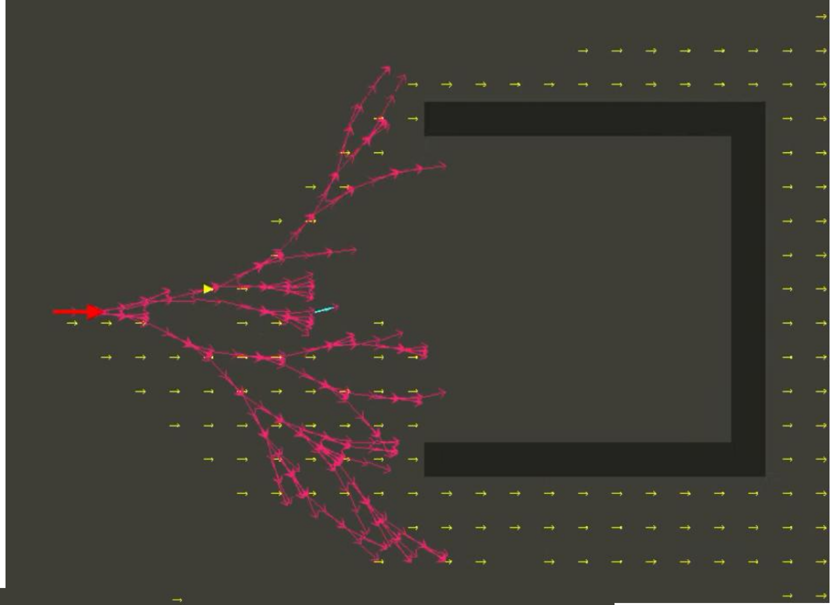


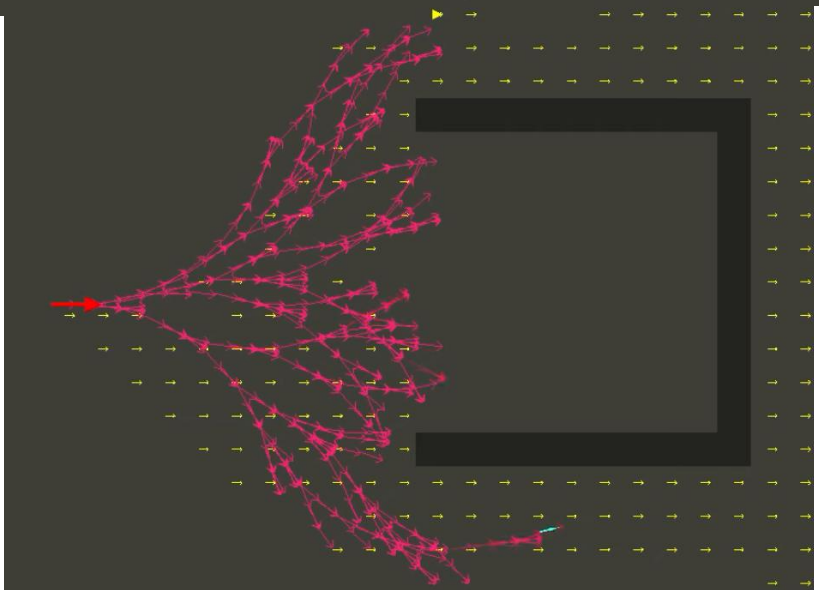
three motion primitives

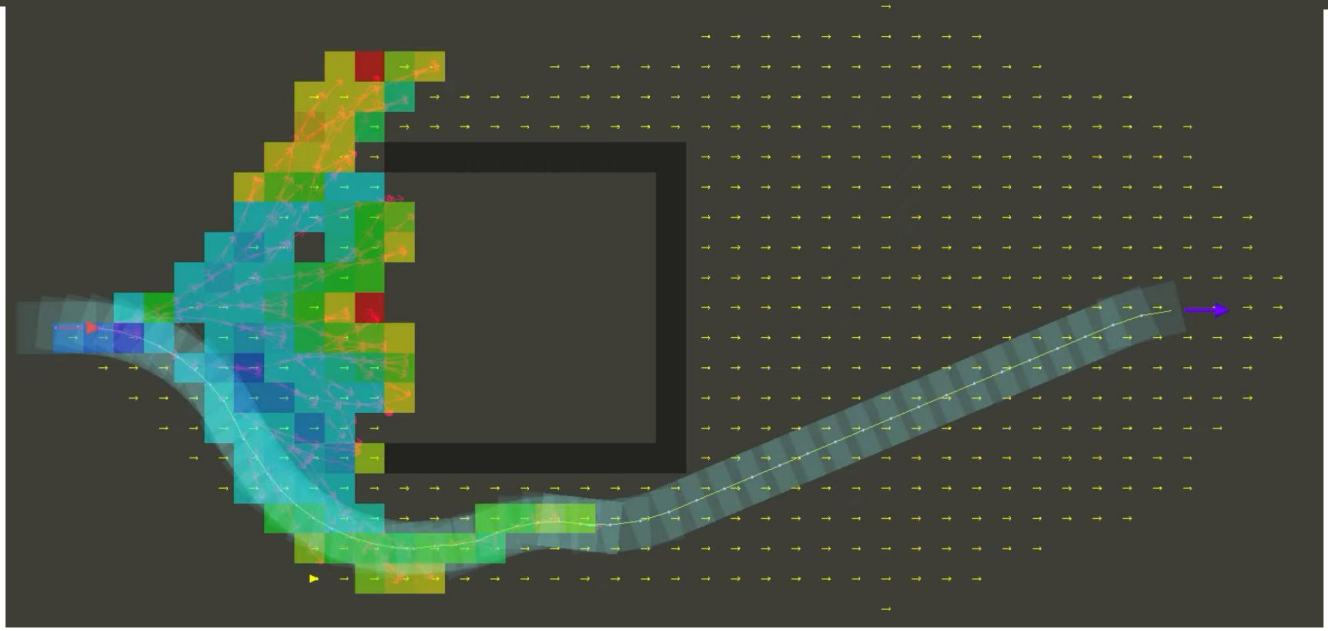
In practice, more tricks can be added, e.g., directly expand a analytical path when getting very close to the goal or in an open space (no obstacle)











https://www.youtube.com/watch?v=Ip2iUrVoFXc

Junior: The Stanford Entry in the Urban Challenge

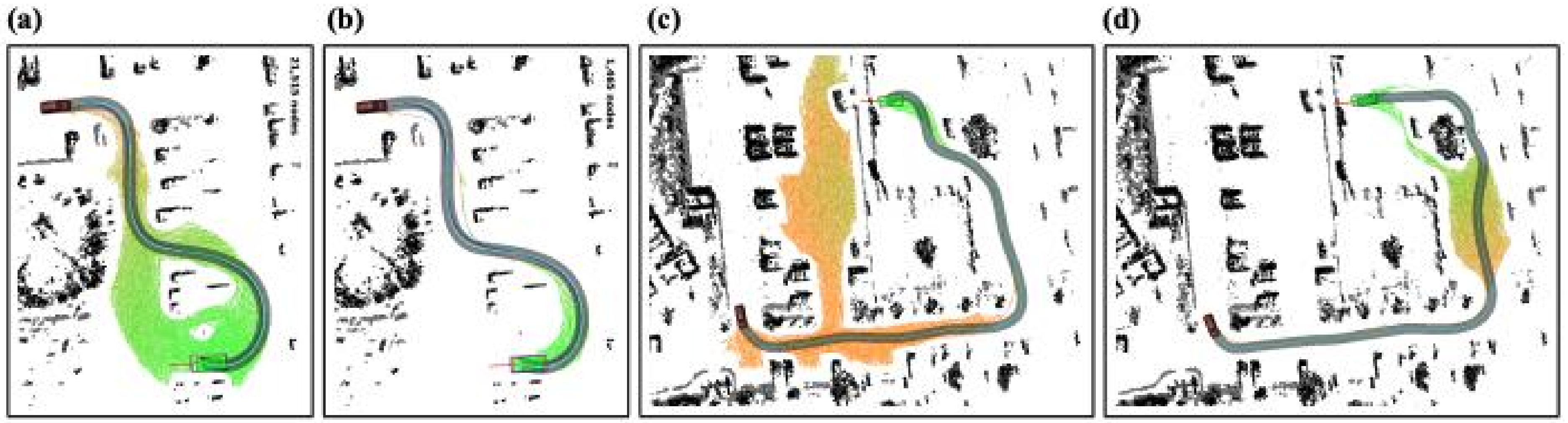


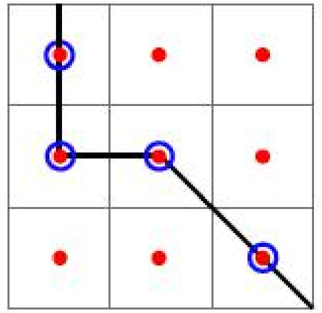
Figure 16: Hybrid-state A* heuristics. (a) Euclidean distance in 2-D expands 21, 515 nodes. (b) The non-holonomic-without-obstacles heuristic is a significant improvement, as it expands 1, 465 nodes, but as shown in (c), it can lead to wasteful exploration of dead-ends in more complex settings (68, 730 nodes). (d) This is rectified by using the latter in conjunction with the holonomic-with-obstacles heuristic (10, 588 nodes).

http://robots.stanford.edu/papers/junior08.pdf

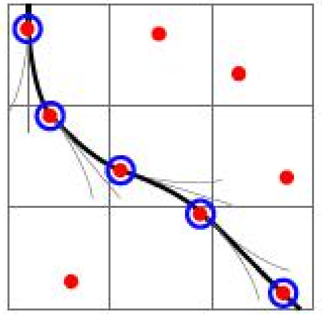
Properties of Hybrid A*

- **Soundness**
  - Unlike A*, soundness in hybrid A* may be compromised if motion primitives do not accurately represent feasible trajectories.
- **Completeness:**
  - Hybrid A* is not strictly complete due to the continuous state space.
  - Completeness can be approximated with dense discretization of the state space.
- **Optimality:**
  - Hybrid A* is not guaranteed to find the optimal path since it approximates the continuous space with motion primitives.
  - The quality of the solution depends on the choice of motion primitives and heuristics.
- **Feasibility:**
  - The generated path is kinematically feasible, respecting vehicle constraints (e.g., turning radius)

The motion planning problem



- Get from point A to point B avoiding obstacles
- Last 2 lectures we saw how to search for collision free trajectories can be converted to graph search
  - Each vertex represents a region of the gridded state space; edges between centers
    - Paths may not be realizable
  - Hybrid A* constructs dynamically feasible paths
    - edges between arbitrary points in grid regions
    - Not guaranteed to be complete
  - Grid/discretization does not scale to high-dimensional state spaces



- Today: **sampling-based motion planning**
  - Can directly incorporate dynamical constraints
  - **Scales to higher dimensions**
  - Cons: Probabilistic completeness

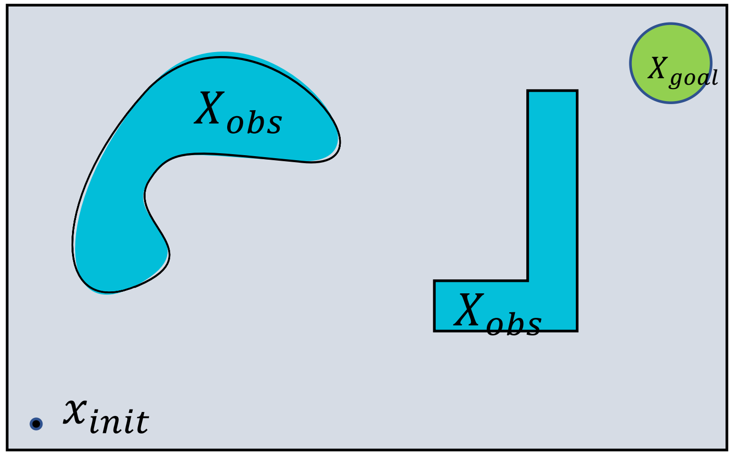
Motion planning problem

Given a dynamical system:

$$\frac{dx(t)}{dt} = f(x(t), u(t)), \; x(0) = x_{init}. \quad (1)$$

an obstacle set $X_{obs} \subset \mathbb{R}^d$, and a goal set $X_{goal} \subset \mathbb{R}^d$, the objective is to find (if it exists) a control signal $u()$ such that the solution of (1) satisfies

- for all $t \in \mathbb{R}_{\geq 0}$, $x(t) \notin X_{obs}$ and
- for some finite T ≥ 0, $x(T) \in X_{goal}$
- Return failure if no such control signal exists.

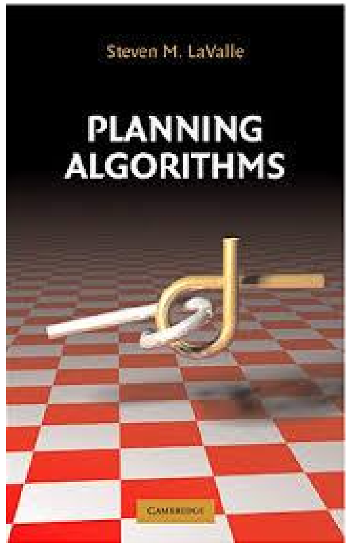


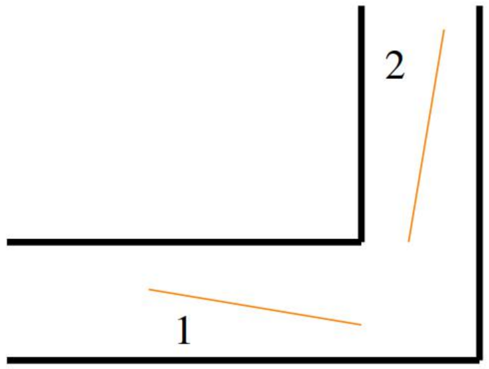
Motion planning is a central problem in robotics

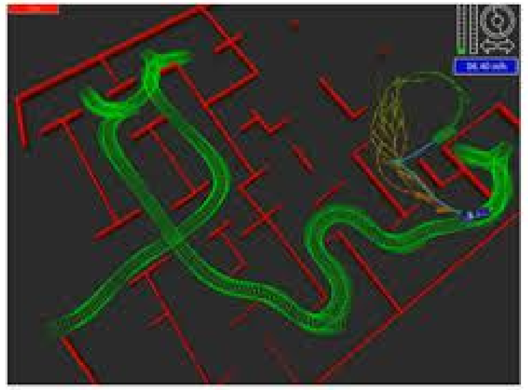
Basic problem in robotics

- Autonomous vehicles
- Puzzles

Provably computationally hard: a basic version (the Generalized Piano Mover's problem) is known to be PSPACE-hard [Reif, '79].







Types of planners

Discretization + graph search: Analytic/grid-based methods do not scale well to high dimensions.

- A* , D* , etc. can be sensitive to graph size. Resolution complete.

Algebraic planners: Explicit representation of obstacles.

- Use complicated algebra (visibility computations/projections) to find the path. Complete, but often impractical.

Potential fields/navigation functions: Virtual attractive forces towards the goal, repulsive forces away from the obstacles.

- No completeness guarantees, unless "navigation functions" are available—very hard to compute in general.

Our focus: **Sampling**-based algorithms

Solutions are computed based on samples from some distribution.

Retain some form of completeness, e.g., probabilistic completeness

Incremental sampling methods

- Lend themselves to real-time, on-line implementations
- Can work with very general dynamics
- Do not require explicit constraints

Outline

Probabilistic Roadmaps (this lecture)

Rapidly expanding random trees (RRT)

Rapidly-exploring Random Graph (RRG)

Probabilistic RoadMaps (PRM)

Introduced by Kavraki and Latombe in 1994

Mainly geared towards "multi-query" motion planning problems

Idea: build (offline) a graph (i.e., the roadmap) representing the "connectivity" of the environment; use this roadmap to figure out paths quickly at run time.

Learning/pre-processing phase:

- Sample n points from $X_{free} = [0, 1]^d \setminus X_{obs}$
- Try to connect these points using a fast "local planner"
- If connection is successful (i.e., no collisions), add an edge between the points.

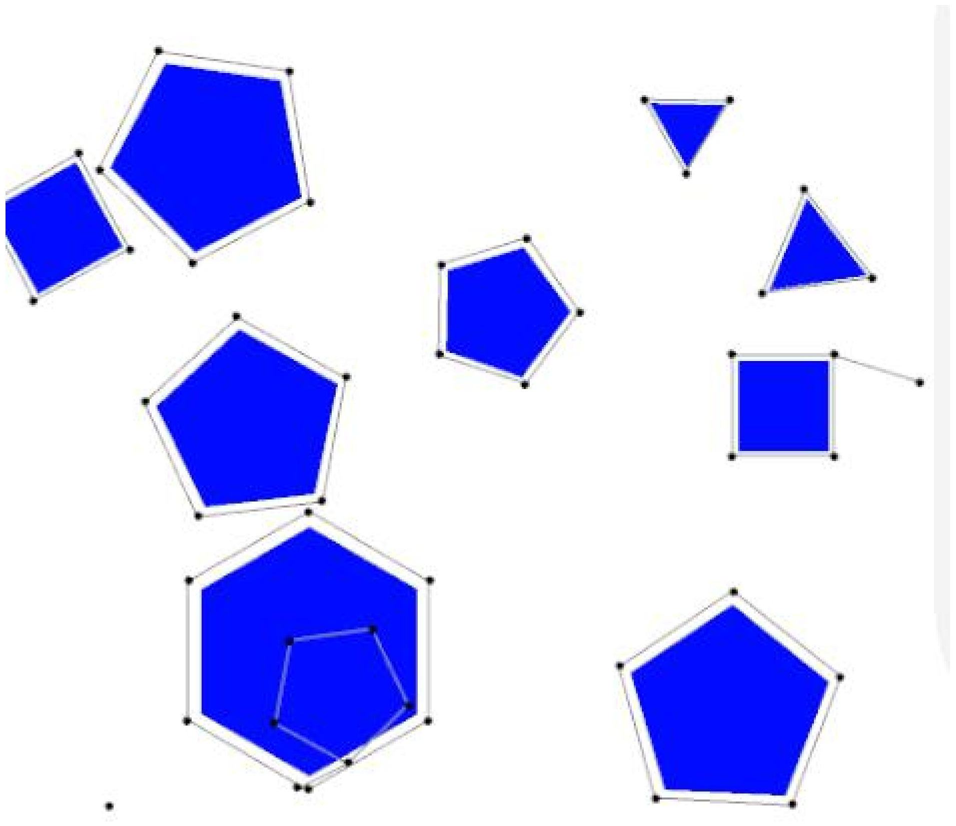
At run time:

- Connect the start and end goal to the closest nodes in the roadmap
- Find a path on the roadmap, e.g., using BFS, DFS, A*

First planner ever to demonstrate the ability to solve general planning problems in > 4-5 dimensions!

PRM in action

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; Overmars, M. H. (1996), "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, ***12*** *(4): 566–580*



Picture from Wikipedia.org
https://en.wikipedia.org/wiki/Probabilistic_roadmap

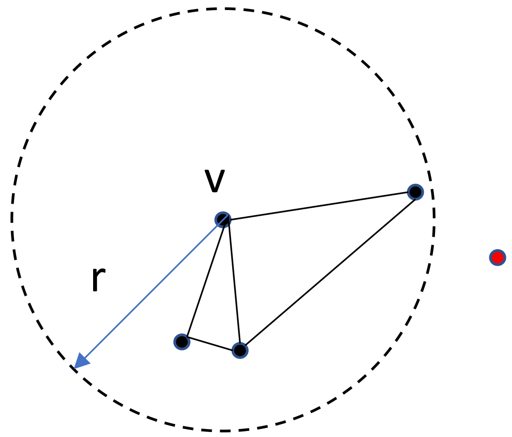
Simple PRM (sPRM) construction

```
V ← $\{x_{init}\} \cup \{v_i \sim X_{free}\}_{i=1,\dots,N-1}$
E ← ∅
foreach v ∈ V do
  U ← Near(G = (V, E), v, r) \ {v}
    foreach u ∈ U do
        if CollisionFree(v, u) then
          E ← E ∪ {(v, u),(u, v)}
return G = (V, E)

path = shortest_path($x_{init}$, $x_{goal}$, V, E)
// Dijkstra's or A*
```

Near(G, v, r): Finds the subset of vertices in G that are within r distance of v

CollisionFree(v, u): checks whether there is a path from u to v that does not collide with the obstacles



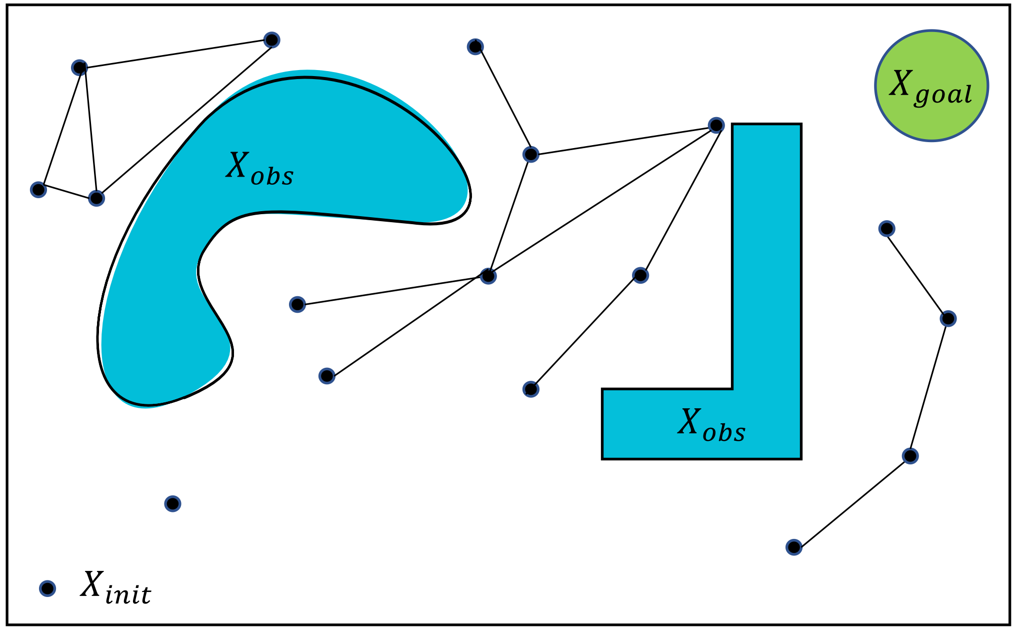
Probabilistic RoadMap

Connect points within a radius r, starting from “closest” ones

Do not attempt to connect points already on the same connected component of PRM

What properties does this algorithm have?

- Will it find a solution if one exists?
- Is this an optimal solution?
- What is the complexity?



Robustness and Probabilistic completeness

Definition. A motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ is *robustly feasible* if there exists some small δ>0 such that a solution remains a solution if obstacles are "**dilated**" by δ.

Definition. An algorithm ALG is *probabilistically complete* if, for any *robustly feasible* motion planning problem defined by $P = (X_{free}, x_{init}, X_{goal}), \lim_{N\to\infty} \Pr(\text{ALG returns a solution to } P) = 1.$

- N is the number of samples
- Applicable to motion planning problems with a robust solution.

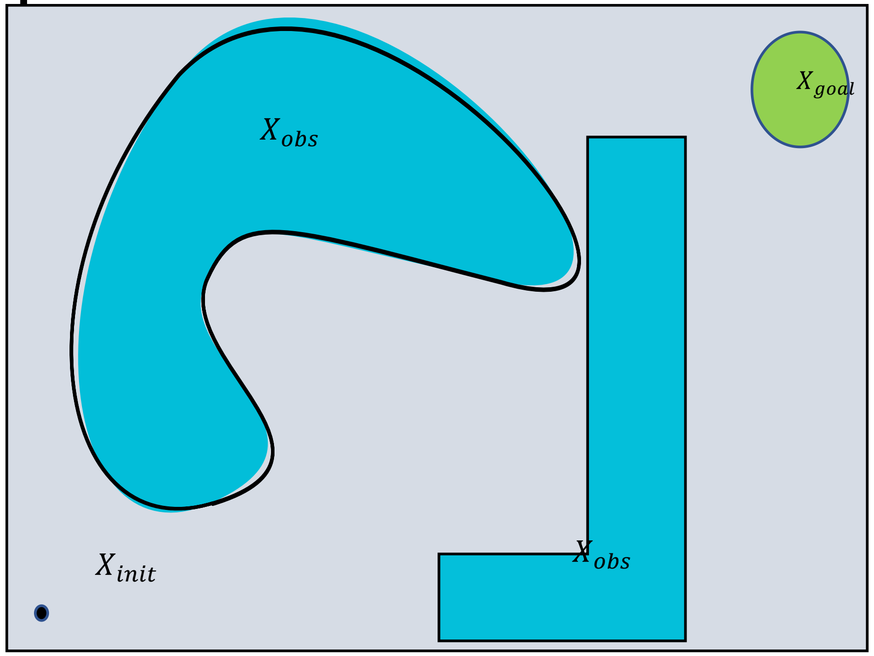


Fig. not robustly feasible.

Paper: Sampling-based Algorithms for Optimal Motion Planning, Sertac Karaman Emilio Frazzoli

Asymptotic optimality of sampling-based algorithms

Suppose we have a cost function c that associates to each path σ a non-negative cost $c(\sigma)$, e.g., $c(\sigma) = \int_\sigma \chi(s)\, ds$.

$Y_i^{ALG} = c(\sigma_i)$ *Cost of the output path* σ_i *from ALG with* $\boldsymbol{i}$ ***samples***

Definition. An algorithm ALG is *asymptotically optimal* if, for any motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ and cost function c that admits a robust optimal solution with finite cost c^*,

$$\boldsymbol{P}\left(\left\{\lim_{i\to\infty} Y_i^{ALG} = c^*\right\}\right) = 1$$

Properties of PRM

The simplified version of the PRM (sPRM) algorithm has been shown to be probabilistically complete. (No proofs available for the "real" PRM!)

Moreover, the probability of success goes to 1 exponentially fast, if the environment satisfies **visibility** conditions.

Two nodes satisfies visibility if they have a straight line collision-free path between them

But, NOT *asymptotically* optimal

- Edges make unnecessary connections in a connected component
- Set of optimal paths has measure 0

Improved version of PRM (PRM*) exists to achieve asymptotical optimality

Complexity of Sampling-based Algorithms

How can we measure complexity for an algorithm that does not necessarily terminate?

- Treat the number of samples as “the size of the input.” (Everything else stays the same)
- Complexity per sample: how much work (time/memory) is needed to process one sample.
- Useful for comparison of sampling-based algorithms. Not for deterministic, complete algorithms.

Complexity of PRM for N samples $\Theta(N^2)$

Practical complexity reduction tricks

- k-nearest neighbors: connect to the k nearest neighbors. Complexity Θ(N log N). (Finding nearest neighbors takes log N time.)
- Bounded degree: connect at most k neighbors among those within radius r.
- Variable radius: change the connection radius r as a function of N. How?