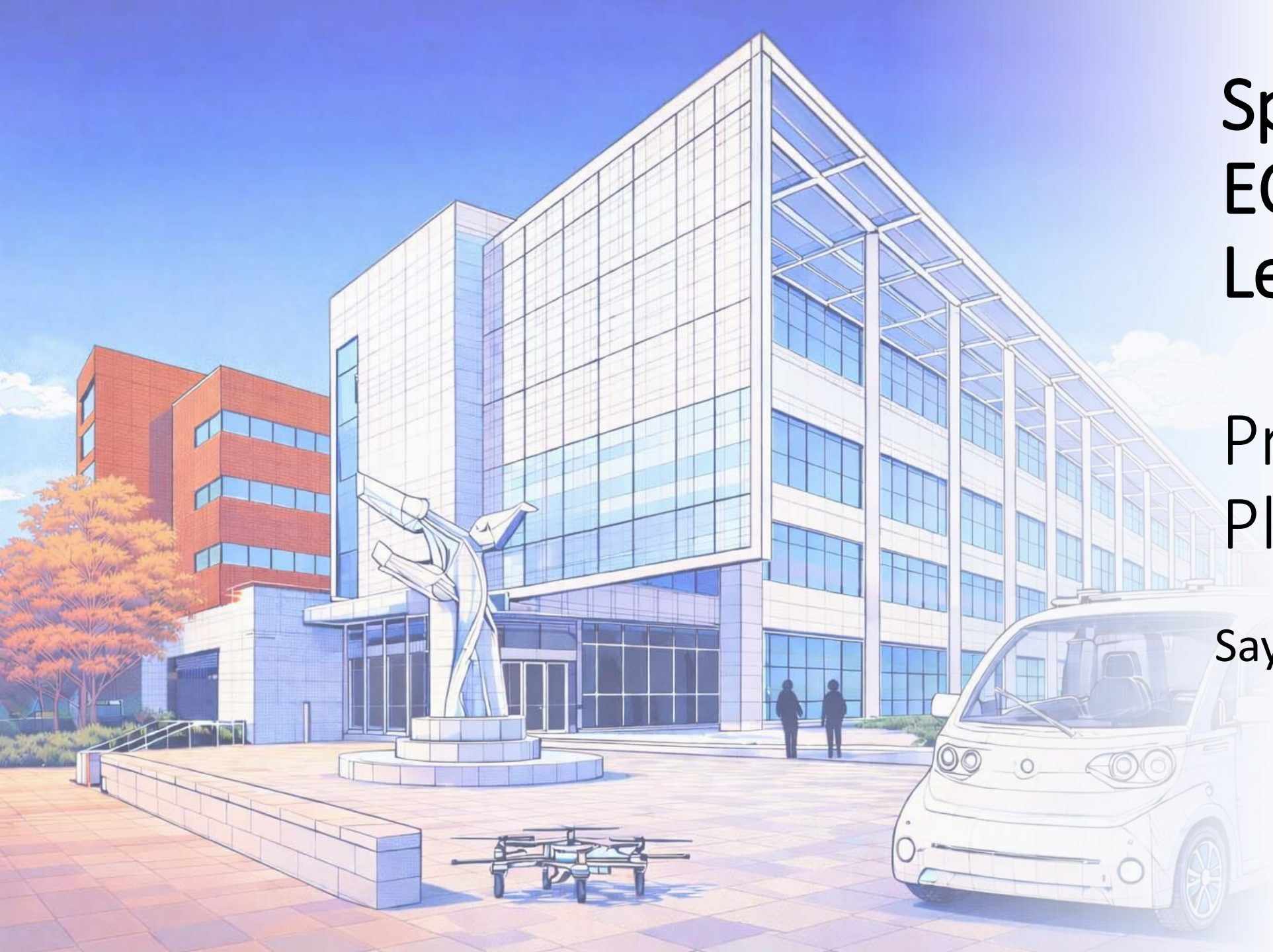


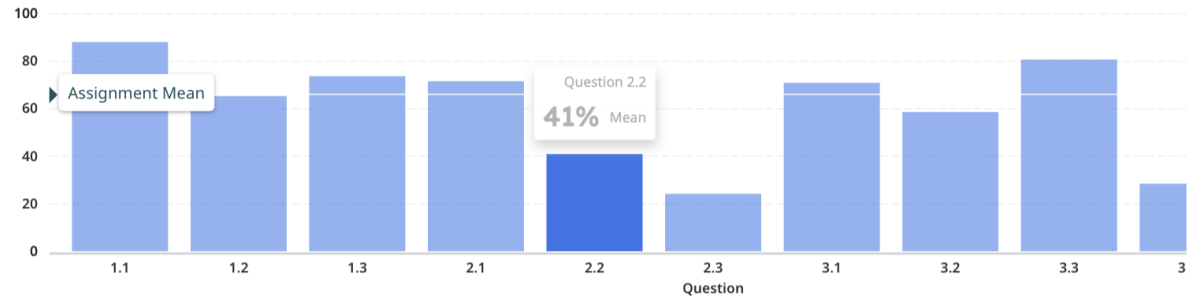
Spring 26
ECE484
Lecture 16

Probabilistic Planning

Sayan Mitra

Announcements

- Project Midpoint Check-in: April 21st ECEB 5072 and 1015 Control
- Midterm 3 Review: April 28
- Midterm 3: May 5th
- Final demo for F1tenth: May 7th (all day)
- Final project presentation: May 14th 7-10 pm (More details to follow)

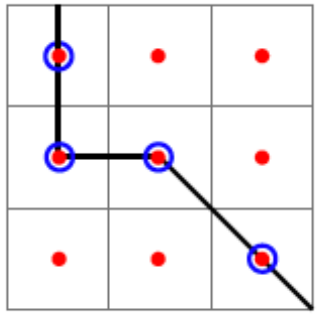


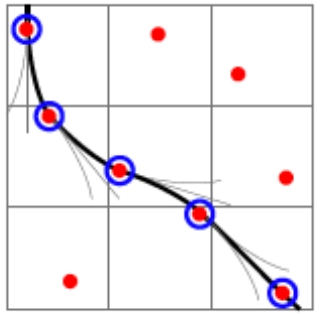
Midterm 2 80.0 points

Minimum	Median	Maximum	Mean	Std Dev
26.25%	67.81%	97.5%	65.72%	16.88%

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 and dynamic programming
- A*, dynamic programming on graph: Each vertex represents a region of the gridded state space; edges between centers
  - Paths may not be dynamically feasible
  - Grid/discretization does not scale to high-dimensional state spaces
- LQR (dynamic programming for LTI systems)
  - Scalable and (LTI) dynamics-aware
  - Does not directly support obstacles

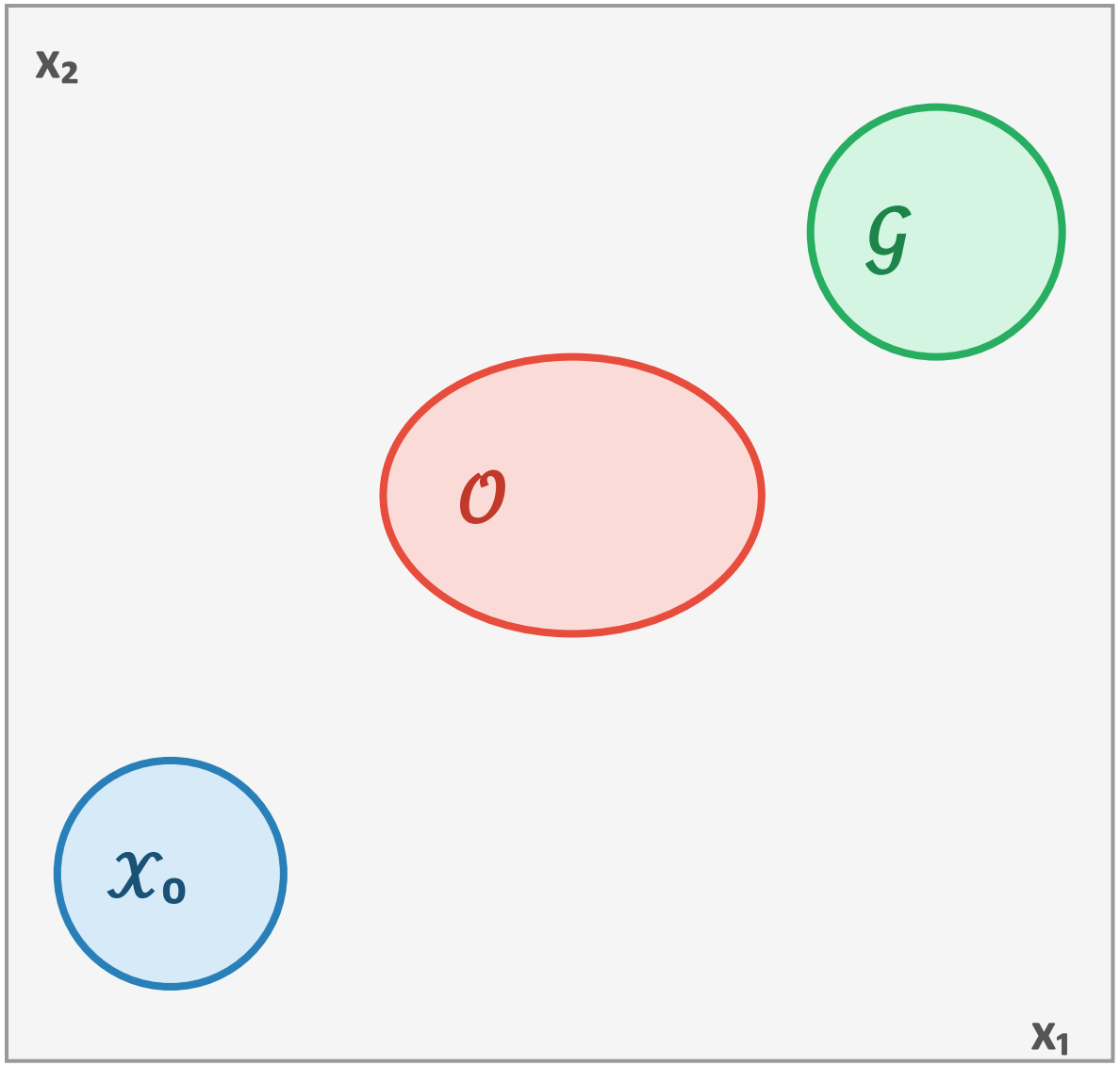




Sampling-based motion planning

- Can directly incorporate dynamical constraints
- Can handle obstacles
- Scales to higher dimensions
- Cons: Probabilistic completeness

LQR Review: Reach-Avoid



Dynamics

$x_{t+1} = A\, x_t + B\, u_t$

Goal Set

$\mathcal{G} = \{ x : (x - x_g)^T M_g (x - x_g) \le r_g \}$

Initial Set

$\mathcal{X}_0 = \{ x : \|x - x_0\| \le r_0 \}$

Obstacle Set (ellipsoidal)

$\mathcal{O} = \{ x : (x - x_o)^T M (x - x_o) \le 1 \}$

Objective: reach $\mathcal{G}$ by horizon T, minimizing J, penalizing proximity to $\mathcal{O}$

LQR Reach-Avoid: Closed-Form Solution

$$J = \Sigma_{t=0}^{T-1}[\, x_t^{\mathrm{T}}(Q + \lambda Q_\varphi)x_t + u_t^{\mathrm{T}} R\, u_t\,] + (x_T - x_g)^{\mathrm{T}} P_T\, (x_T - x_q)$$

Stage cost $\ell(x_t, u_t)$

$x_t^T Q\, x_t + u_t^T R\, u_t$
$Q \geq 0,\ R > 0$

Avoid cost $\lambda \cdot \phi(x_t)$

$\phi(x_t) = x_t^T Q_\phi\, x_t$
$Q_\phi \geq 0$: designer-chosen

Terminal cost V_T

$(x_T - x_g)^T P_T\, (x_T - x_g)$
$P_T > 0$ (reach penalty)

Key: total cost quadratic

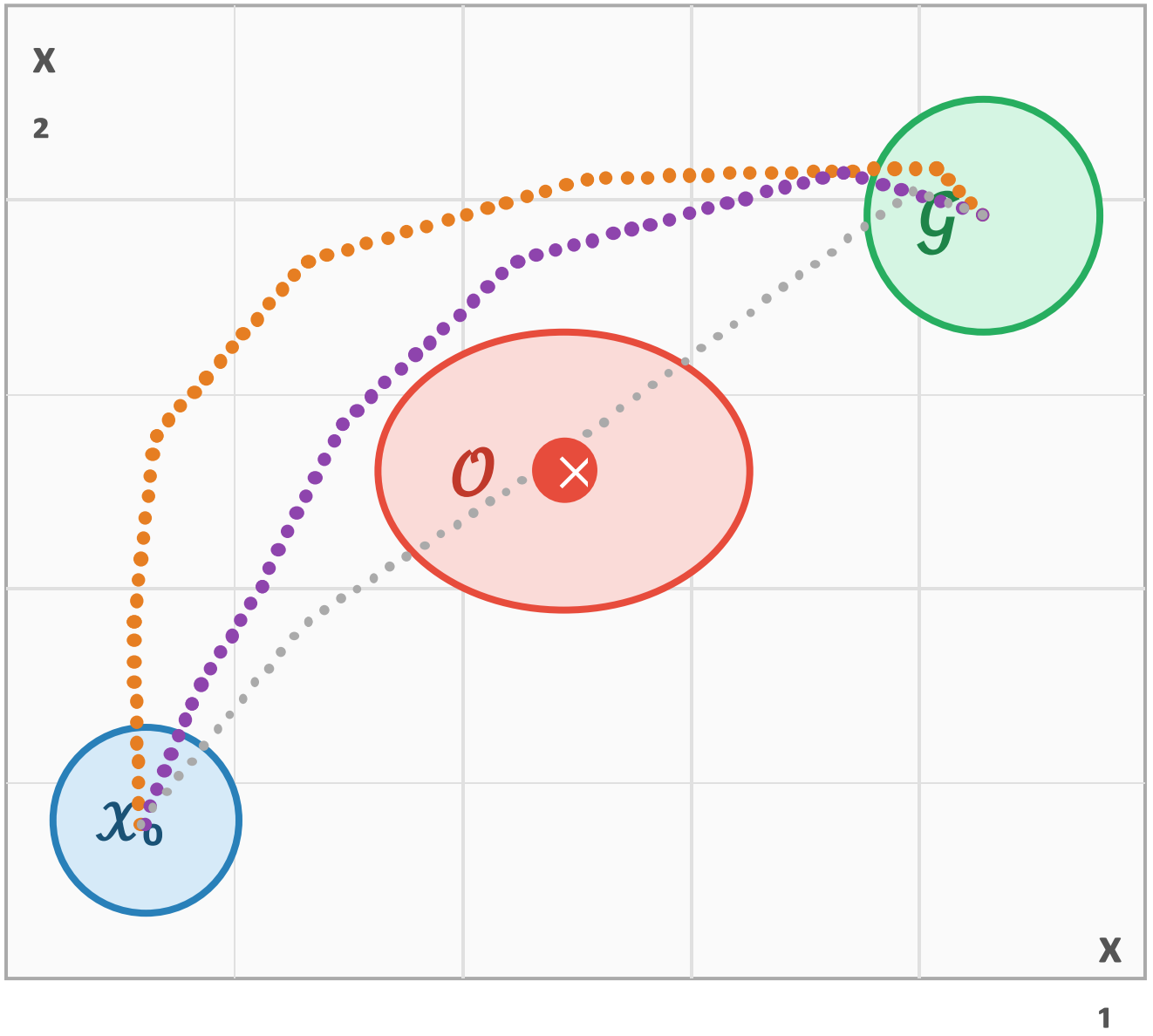
Riccati: $P_t = (Q + \lambda Q_\phi) + A^T P_{t+1} A - A^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$

Gain: $K_t = (R + B^T P_{t+1} B)^{-1}\, B^T P_{t+1} A$

Control: $u_t^* = -K_t\, x_t$ (closed-form linear state-feedback)

Clear choice $\phi(x_t) = 1 / [(x_t - x_o)^T M (x_t - x_o)]$ but not quadratic. Alternative: $Q_\phi = \alpha\, (x_o x_o^T / \|x_o\|^2)$ — penalizes the direction toward the obstacle center x_o.

Trajectories & Why Avoidance Is Not Guaranteed



- λ large — detours around $\mathcal{O}$
- λ medium — skirts $\mathcal{O}$ edge
- λ small — passes through $\mathcal{O}$ ×

⚠ Soft constraint — avoidance NOT

λQ_ϕ penalizes state magnitude in directions encoded by Q_ϕ. This is a quadratic surrogate — it does not measure proximity to $\mathcal{O}$ and does not guarantee or even systematically reduce obstacle entry. The trajectories shown are illustrative;

Hard guarantees need:
Reachability · MPC w/ hard constraints

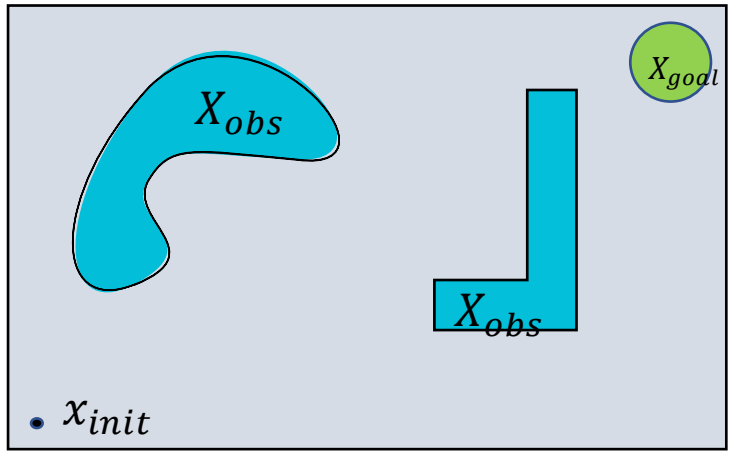
Motion planning problem

Given a dynamical system:

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

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.

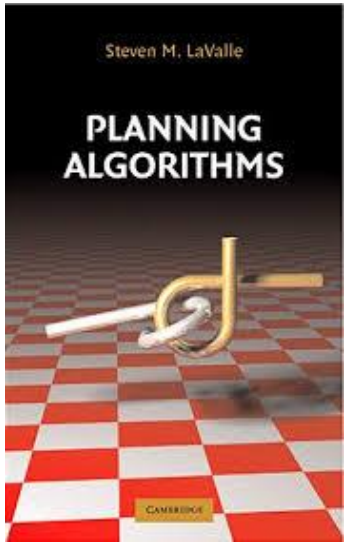


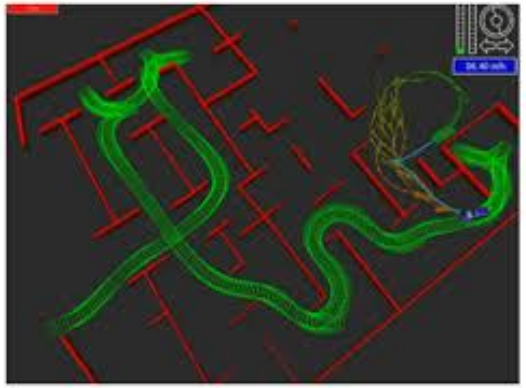
Motion planning is a central problem in robotics

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].





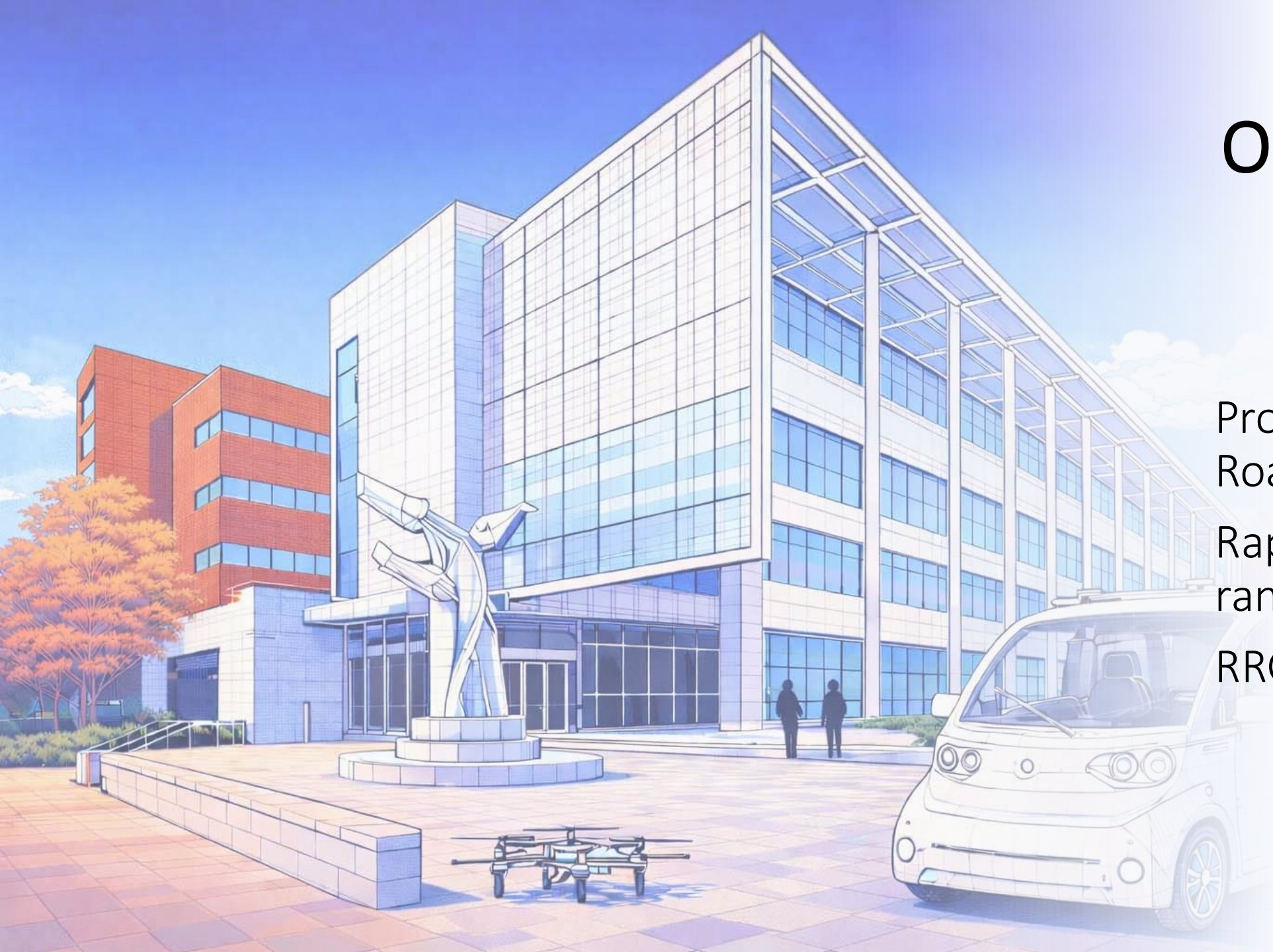
Sampling-based algorithms

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

Rapidly expanding random trees (RRT)

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.

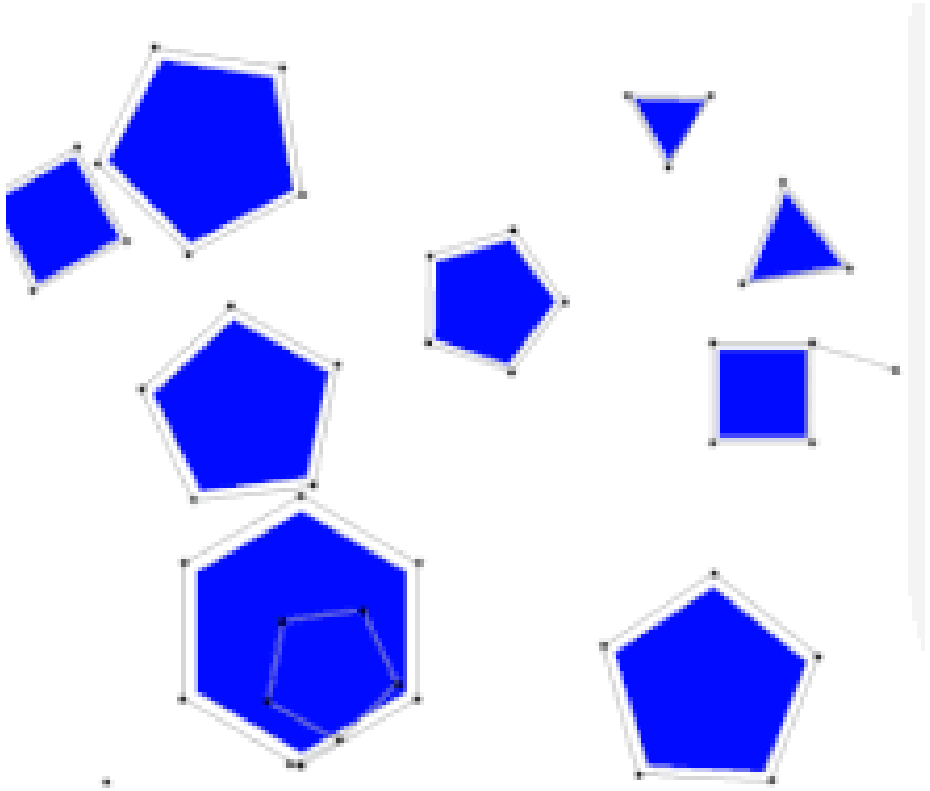
At run time:

- 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 construction

$V \leftarrow \{x_{init}\} \cup \{v_i \sim X_{free}\}_{i=1,\ldots,N-1}$

$E \leftarrow \emptyset$

foreach $v \in V$ **do**

  $U \leftarrow$ Near(G = (V, E), v, r) \ {v}

    foreach $u \in U$ **do**

        if CollisionFree(v, u) **then**

          $E \leftarrow E \cup \{(v, u),(u, v)\}$

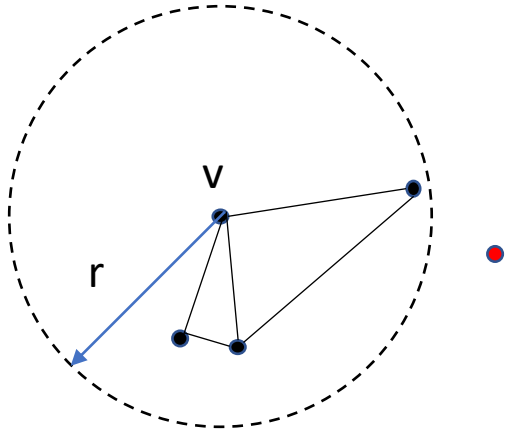
return G = (V, E)

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



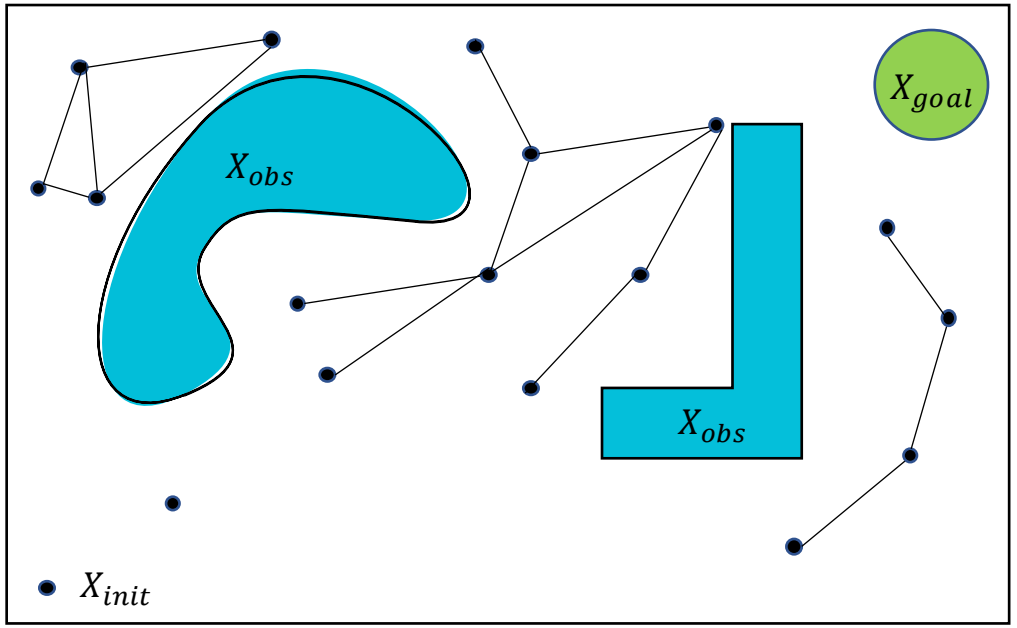
Probabilistic RoadMap

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?



Simple PRM construction

$V \leftarrow \{x_{init}\} \cup \{v_i \sim X_{free}\}_{i=1,\ldots,N-1}$ O(N)

$E \leftarrow \emptyset$

foreach $v \in V$ **do**

  $U \leftarrow$ Near(G = (V, E), v, r) \ {v} O(log N)

    foreach $u \in U$ **do**

      if CollisionFree(v, u) **then**

        $E \leftarrow E \cup \{(v, u),(u, v)\}$

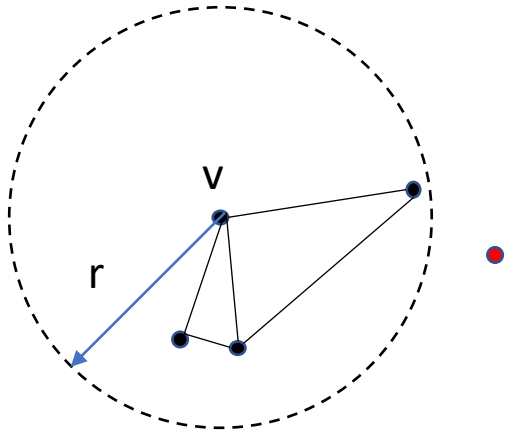
return G = (V, E) Total O(N log N + N k T_lp)

path = shortest_path(x_{init}, x_{goal}, V, E) O((N+E)log N)

// Dijkstra's or A*

Near(G, v, r): Finds the subset of vertices in G that are within r distance of v [O(N log N) with KD tree]

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



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 $P = (X_{free}, x_{init}, X_{goal})$, $\lim_{N\to\infty} \Pr(ALG\ returns\ a\ solution\ to\ P) = 1$.

- Applicable to motion planning problems with a robust solution.

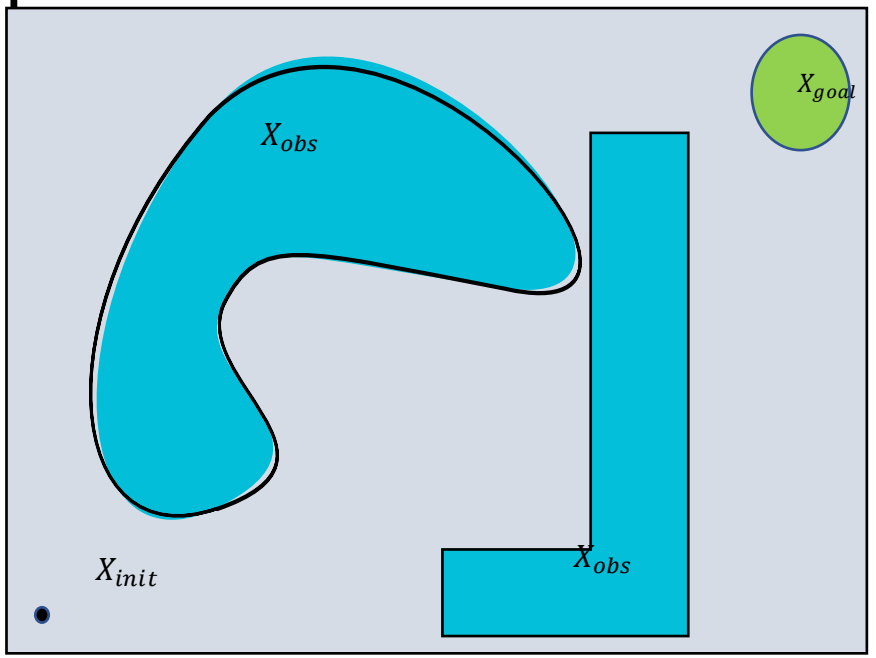


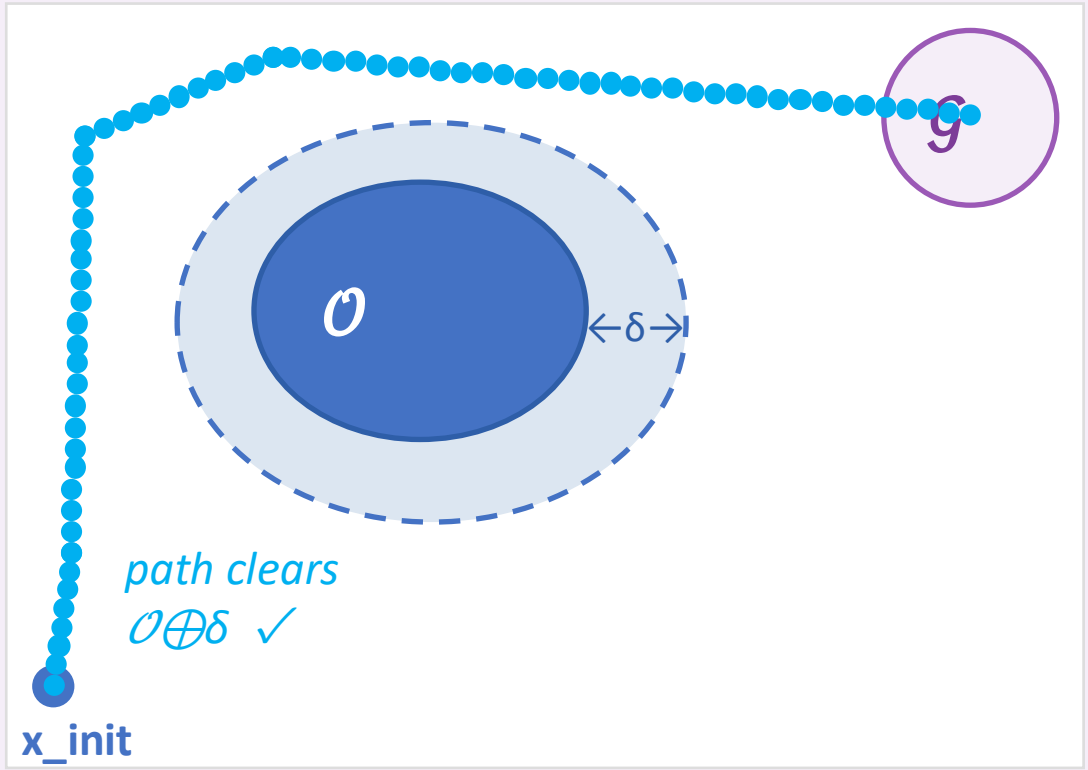
Fig. not robustly feasible.

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

Robust feasibility

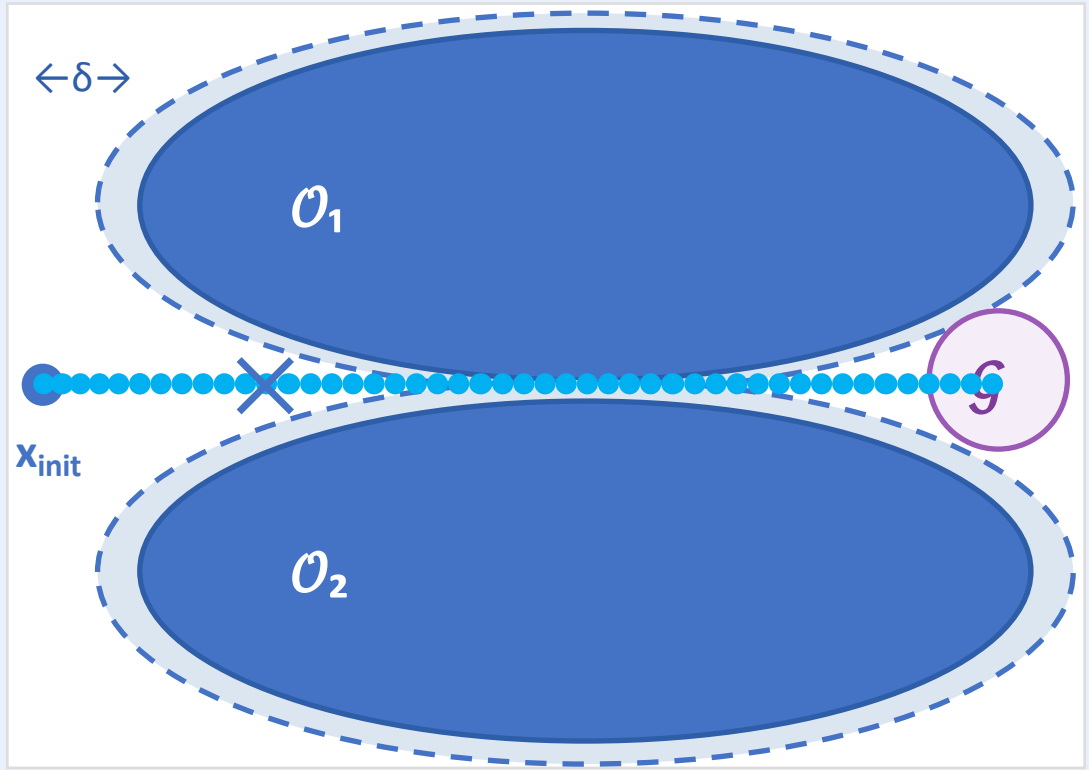
Def. P robustly feasible $\iff$ $\exists\ \delta>0$: solution to P remains a solution when all obstacles dilated by δ

✓ Robustly Feasible



$\exists\ \delta>0$: path avoids $\mathcal{O} \oplus \delta$

✗ Not Robustly Feasible



$\forall\ \delta>0$: $\mathcal{O} \oplus \delta$ blocks passage

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 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

New key concept: combinatorial complexity vs. “visibility”

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 $O(N \log N + N k T_{lp})$

Practical complexity reduction tricks

- k-nearest neighbors: connect to the k nearest neighbors. Complexity $\Theta(N \log N)$. (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.

Rapidly Exploring Random Trees (RRT)

Introduced by LaValle and Kuffner in 1998

Appropriate for single-query planning problems

Idea: build (online) a tree, exploring the region of the state space that can be reached from the initial condition.

At each step: sample one point from X_{free}, and try to connect it to the closest vertex in the tree.

Very effective in practice

Rapidly expanding Random Trees

LaValle, Steven M.; Kuffner Jr., James J. (2001). "Randomized Kinodynamic Planning". *The International Journal of Robotics Research*. **20** (5): 378-400.

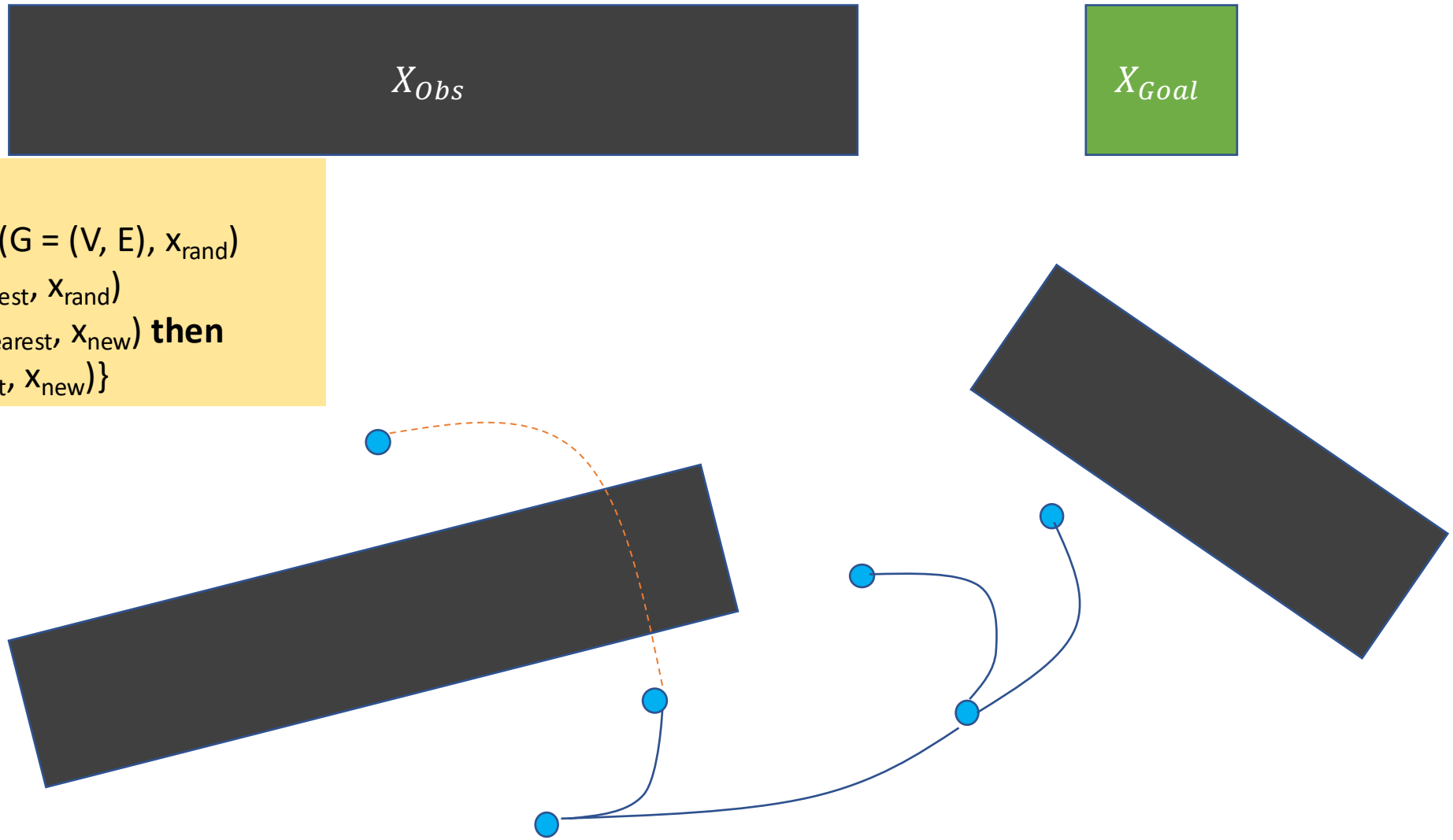
RRT Construction

$V \leftarrow \{x_{init}\}$
$E \leftarrow \emptyset$
for i = 1, . . . , N **do**
  $x_{rand} \sim X_{free}$
  $x_{nearest} \leftarrow$ Nearest(G = (V, E), x_{rand})
  $x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand})
  if ObtacleFree($x_{nearest}$, x_{new}) **then**
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
return G = (V, E)

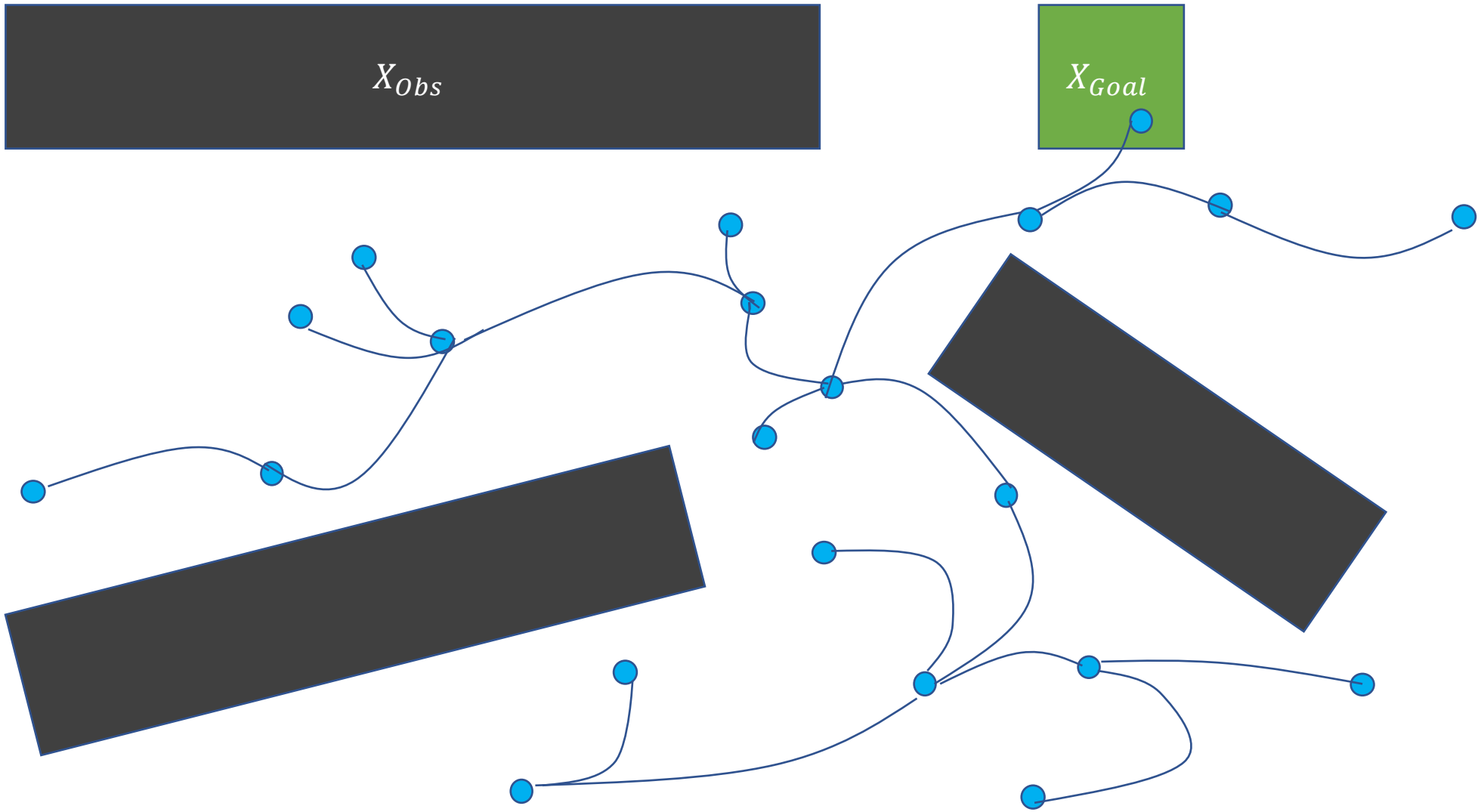
Nearest(G, x_{rand}): Finds the nearest vertex in G from x_{rand}

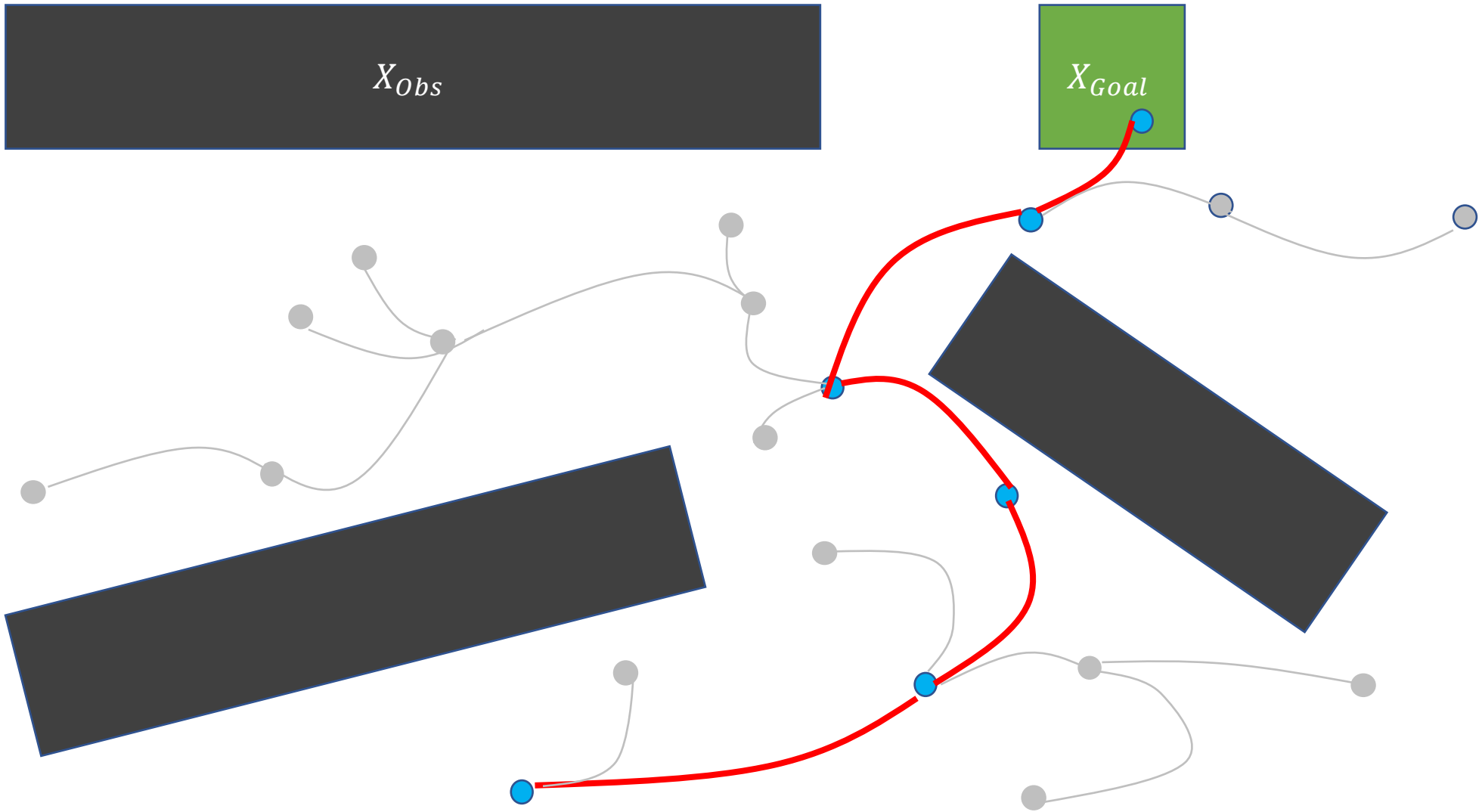
Steer(u, v): Tries to drive the robot from u to v and returns the point nearest to v that it could reach

ObstacleFree(x_0,x_g): Checks whether the path from x_0 to x_g is obstacle free

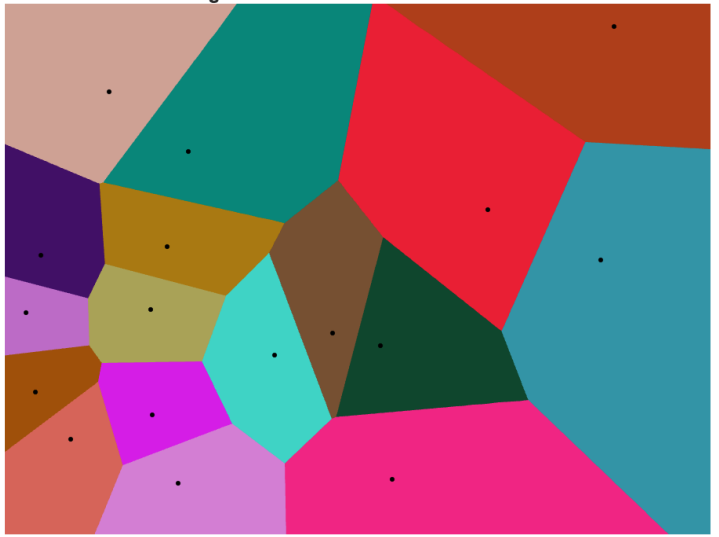


$x_{rand} \sim X_{free}$
$x_{nearest} \leftarrow$ Nearest(G = (V, E), x_{rand})
$x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand})
if ObtacleFree($x_{nearest}$, x_{new}) **then**
  E $\leftarrow$ E $\cup$ {($x_{nearest}$, x_{new})}


X_{Obs}
X_{Goal}


X_{Obs}
X_{Goal}

RRT sampling enjoys Voronoi bias



Given n points in d dimensions, the ***Voronoi diagram*** of the points is a partition of R^d into regions, one region per point, such that all points in the interior of each region lie closer to that region's center than to any other center.

Try it: http://alexbeutel.com/webgl/voronoi.html

RRT enjoys Voronoi bias.

Vertices of the RRT that are more likely to be in isolated or unexplored parts of X_{free} because those regions have larger Voronoi regions

Therefore, RRT grows in unexplored parts

RRT in action [Frazzoli]

- Talos, the MIT entry to the 2007 DARPA Urban Challenge, relied on an“RRT-like” algorithm for real-time motion planning and control.
- Detailed engineering needed to make RRTs work in practice
  - Real-time, on-line planning for a safety-critical vehicle with substantial momentum.
  - Uncertain, dynamic environment with limited/faulty sensors.
- Main innovations [Kuwata, et al. ’09]
  - Closed-loop planning: plan reference trajectories for a closed-loop model of the vehicle under a stabilizing feedback
  - Safety invariance: Always maintain the ability to stop safely within the sensing region.
  - Lazy evaluation: the actual trajectory may deviate from the planned one, need to efficiently re-check the tree for feasibility.
- The RRT-based P+C system performed flawlessly throughout the race.
- https://journals.sagepub.com/doi/abs/10.1177/0278364911406761

Limitations

The MIT DARPA Urban Challenge code, as well as other incremental sampling methods, suffer from the following limitations:

- No characterization of the quality (e.g., “cost”) of the trajectories returned by the algorithm.
- Keep running the RRT even after the first solution has been obtained, for as long as possible (given the real-time constraints), hoping to find a better path than that already available.
- No systematic method for imposing temporal/logical constraints, such as, e.g., the rules of the road, complicated mission objectives, ethical/deontic code.
- In the DARPA Urban Challenge, all logics for, e.g., intersection handling, had to be hand-coded, at a huge cost in terms of debugging effort/reliability of the code.

RRTs and Asymptotic Optimality

- RRTs are great at finding feasible trajectories quickly, however, RRTs are apparently terrible at finding good trajectories. Why?
- Let Y^{RRT}_n be the cost of the best path in the RRT at the end of iteration n.
- It is easy to show that Y^{RRT}_n converges (to a random variable), $\lim_{n\to\infty} Y_n^{RRT} = Y_\infty^{RRT}$

  where Y_∞^{RRT} is sampled from a distribution with zero mass at the optimum

Theorem [Karaman & Frizzoli`10] (Almost sure *sub-optimality*) If the set of optimal paths has measure zero, the sampling distribution is absolutely continuous with positive density in X_{free}, and d ≥ 2, then best RRT path converges to a sub-optimal solution almost surely, i.e.,

$$\Pr[Y_\infty^{RRT} > c^*] = 1.$$

Why is RRT not asymptotically optimal?

Root node has infinitely many subtrees that extend at least a distance ϵ away from x_{init}.

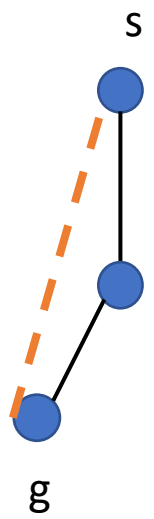
The RRT algorithm traps itself by disallowing new better paths to emerge (unlike hybrid A*)

Why?

When is suboptimality pronounced?

Heuristics such as running the RRT multiple times, running multiple trees concurrently etc., work better than the standard RRT, but also result in almost-sure sub-optimality.

Careful rethinking of RRT required for (asymptotic) optimality.



RRT will not connect s to g

Rapidly Exploring Random Graphs (possibly cyclic)

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;

for i = 1, . . . , N **do**

  $x_{rand} \sim X_{free}$

  $x_{nearest} \leftarrow$ Nearest(G = (V, E), x_{rand});

  $x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand}) ;

  if ObtacleFree($x_{nearest}$, x_{new}) **then**

    $V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{(x_{nearest}, x_{new}),(x_{new}, x_{nearest})\}$;

    $X_{near} \leftarrow$ Near(G = (V, E), x_{new}, $r(|V|)$) ;

    foreach $x_{near} \in X_{near}$ **do**

      if ObtacleFree(x_{near}, x_{new}) **then** $E \leftarrow E \cup \{(x_{near}, x_{new}),(x_{new}, x_{near})\}$

return G = (V, E);

RRG tries to connect the new sample x_{new} to all vertices in a ball of radius r centered at it. (Or just default to the nearest one if such ball is empty.)

$$r(|V|) = \min\{\gamma_{RRG}\left(\frac{\log(|V|)}{|V|}\right)^{\frac{1}{d}}, \eta\}$$

The RRT graph is a subgraph of the RRG graph (which may have cycles)

Theorems [not required for exam]

Probabilistic completeness. Since $V_n^{RRG} = V_n^{RRT}$, for all n RRG has the same completeness properties as RRT, i.e.,

$$Pr\left[V_n^{RRG} \cap X_{goal} = \emptyset\right] = O\left(e^{-bn}\right).$$

Asymptotic optimality. If the **Near** procedure returns all nodes in V within a ball of volume $Vol = \frac{\gamma \log n}{n}$, $\gamma > 2^d\left(1 + \frac{1}{d}\right)$, under some technical assumptions (e.g., sampling distribution, ϵ-robustness of optimal path, and continuity c function), the best RRG path converges to an optimal solution almost surely, i.e.,

$$\Pr[Y_\infty^{RRG} = c^*] = 1.$$

Remarks on RRG

- What is the additional computational load?
  - O(log n) extra calls to ObstacleFree compared to RRT
- Key idea in RRG/RRT*:
  - Combine optimality and computational efficiency, it is necessary to attempt connection to Θ(log N) nodes at each iteration.
  - Reduce volume of the “connection ball” as log(N)/N;
  - Increase the number of connections as log(N).
- These principles can be used to obtain “optimal” versions of PRM, etc.

Summary and future directions

- RRT converges to a NON-optimal solution almost-surely
- RRG and RRT$*$ almost-surely converge to optimal solutions while incurring no significant cost overhead
- Reference: S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. Int. Journal of Robotics Research, 2011. available at http://arxiv.org/abs/1105.1186.
- research directions:
  - Optimal motion planning with temporal/logic constraints
  - Anytime solution of differential games
  - Stochastic optimal motion planning (process + sensor noise)
  - Multi-agent problems.

Algorithm	Prob. Completeness	Asymptotic Optimality	Complexity
sPRM	Yes	Yes	O(N)
k-nearest sPRM	No	No	O(log N)
RRT	Yes	No	O(log N)
PRM*	Yes	Yes	O(log N)
k-nearest PRM*	Yes	Yes	O(log N)
RRG	Yes	Yes	O(log N)
k-nearest RRG	Yes	Yes	O(log N)
RRT*	Yes	Yes	O(log N)
k-nearest RRT*	Yes	Yes	O(log N)