ECE 484: Principles of Safe Autonomy (Fall 2025)
Lecture 18: Search and Planning: RRT and RRG

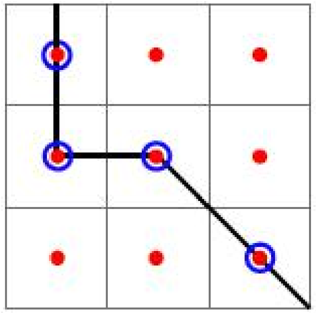
Professor: Huan Zhang
https://publish.illinois.edu/safe-autonomy/
https://huan-zhang.com
huanz@illinois.edu

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

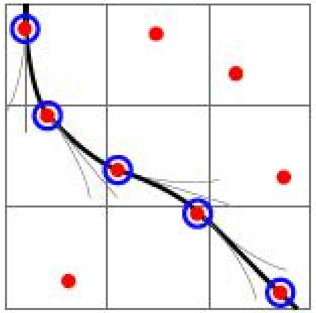
Announcements

- 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: The motion planning problem



- Get from point A to point B avoiding obstacles
- 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

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

Sampling-based algorithms

- Probabilistic Roadmaps (last lecture)
- Rapidly expanding random trees (RRT)
- Rapidly-exploring Random Graph (RRG)

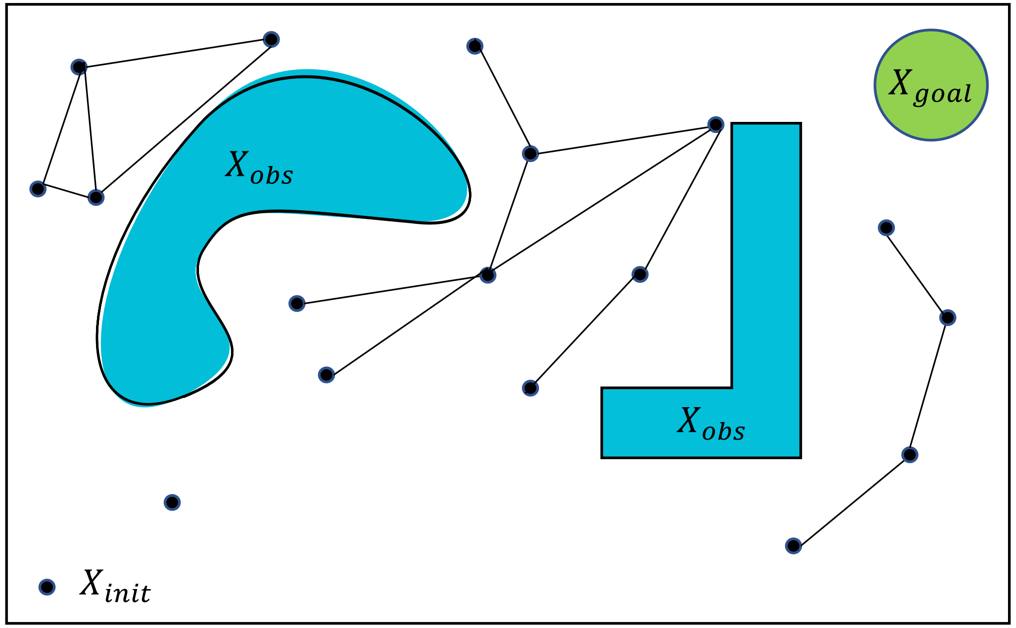
Review: Probabilistic RoadMap

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?



Review: Simple PRM (sPRM) construction

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

$E \leftarrow \varnothing$

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)}

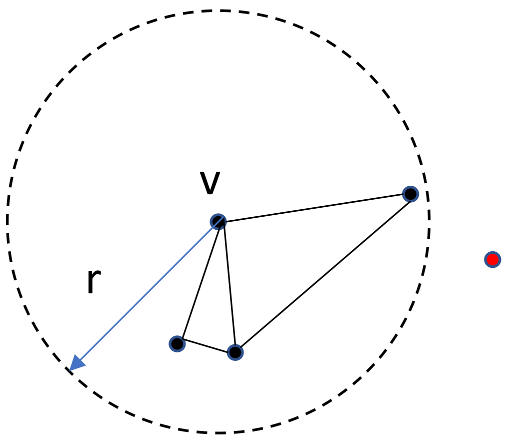
return G = (V, E)

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



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.

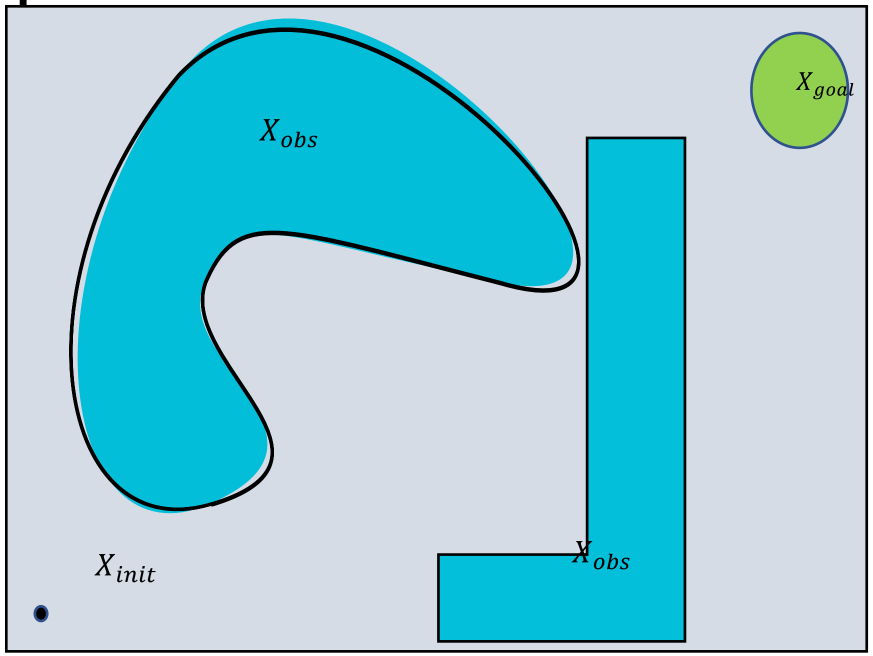


Fig. not robustly feasible.

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.

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 **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^*,

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

PRM (sPRM) algorithm is probabilistically complete, but not asymptotically optimal

"as the number of samples increases, the path found by the algorithm does not necessarily converge to the true shortest possible path"

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

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

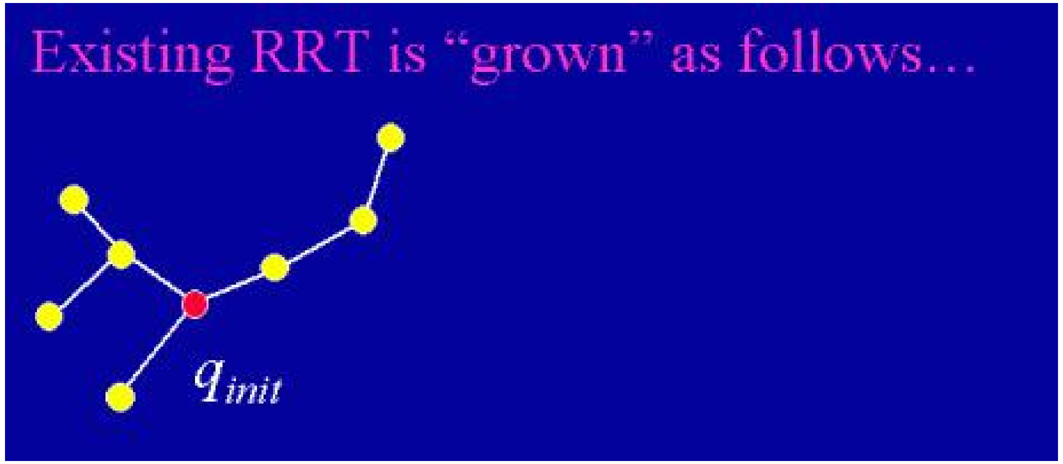
RRT Construction

$V \leftarrow \{x_{init}\}$
$E \leftarrow \varnothing$
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



http://www.kuffner.org/james/plan/algorithm.php

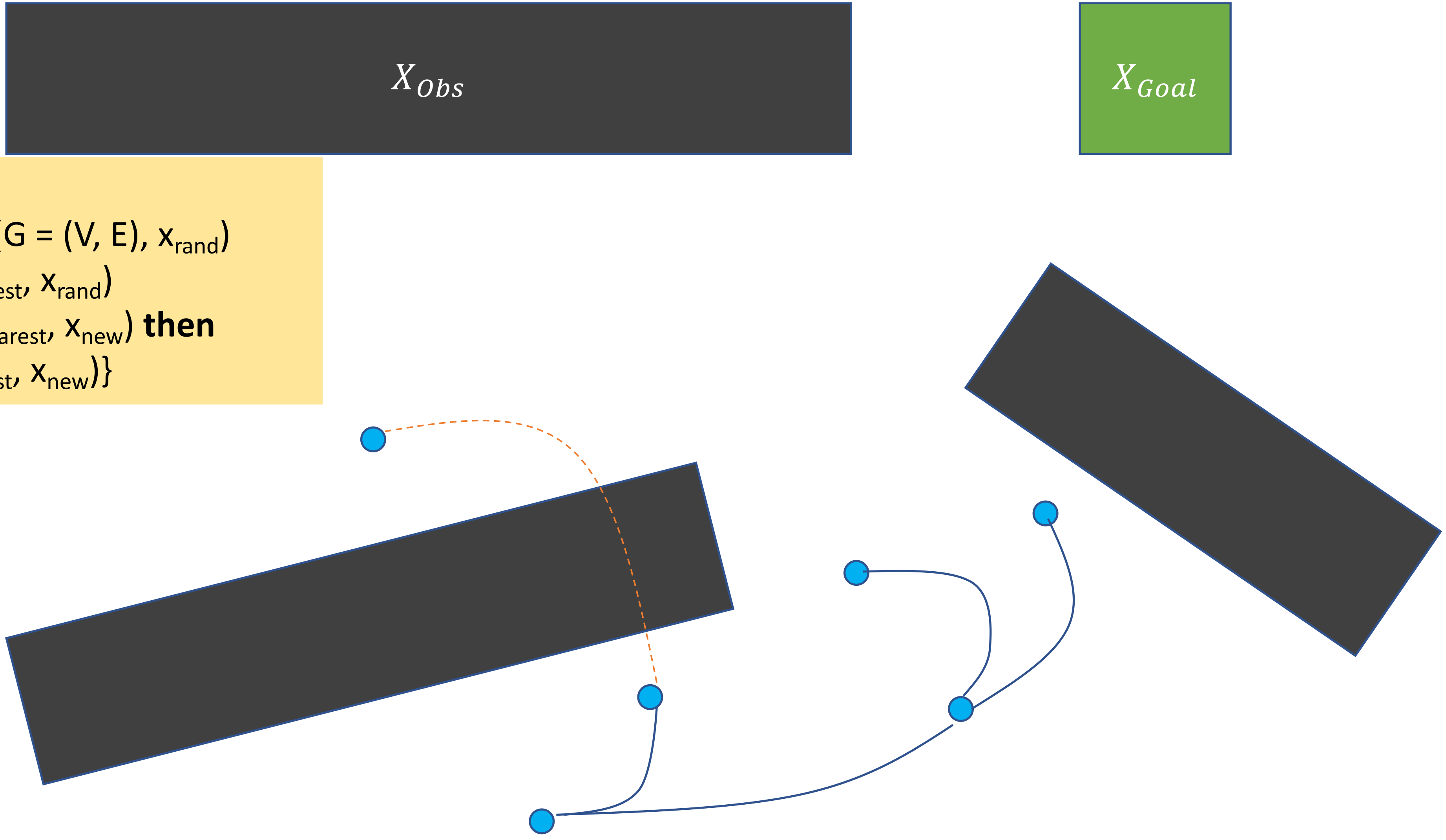
$x_{rand} \sim X_{free}$

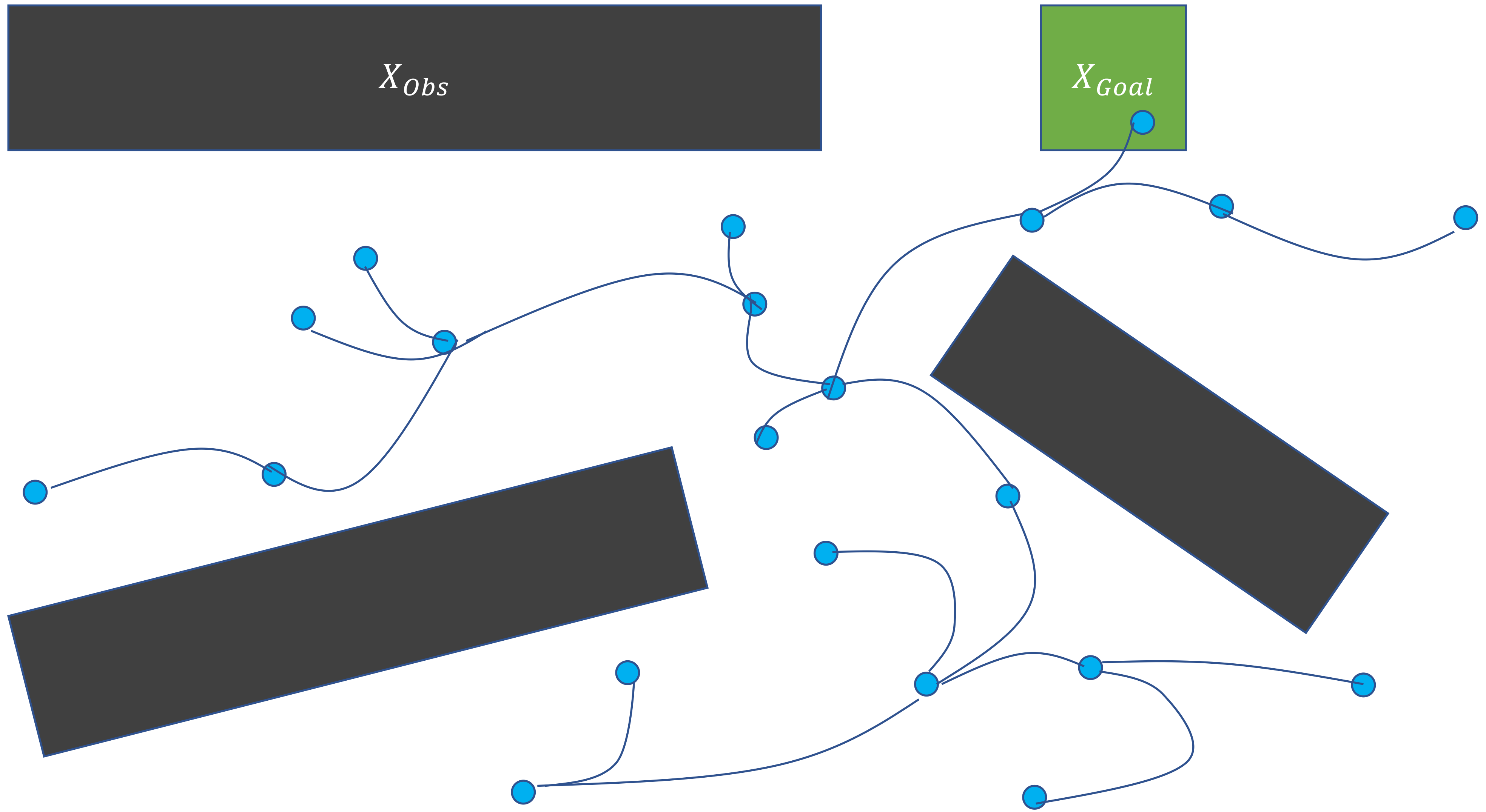
$x_{nearest} \leftarrow$ Nearest(G = (V, E), x_{rand})

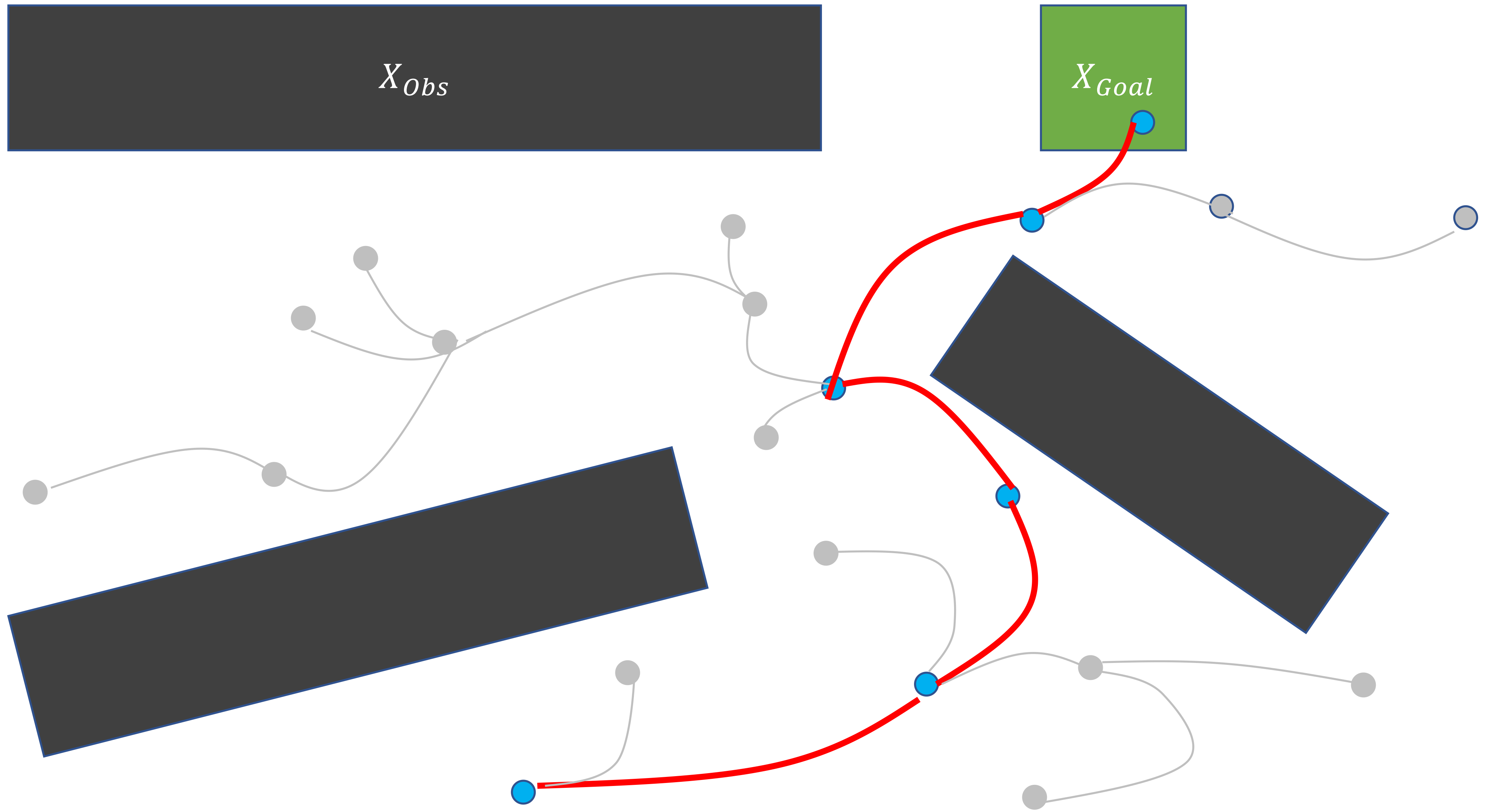
$x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand})

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

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

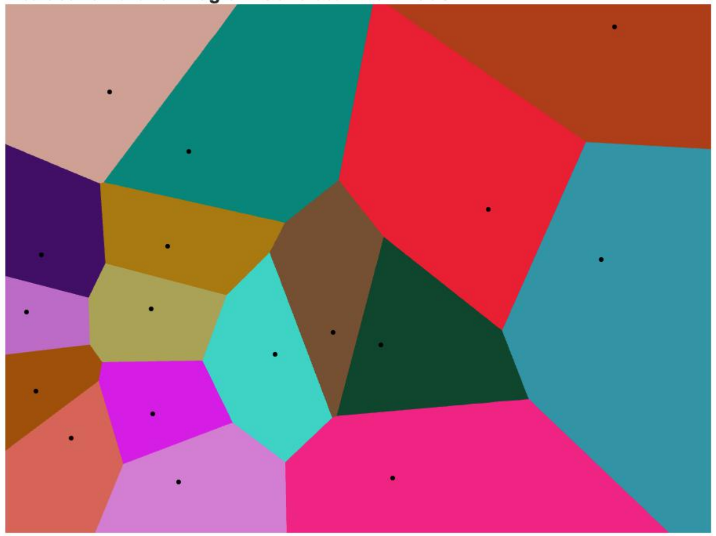



X_{Obs}
X_{Goal}



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:

- A Voronoi diagram partitions the space into Voronoi cells, where all points within a specific cell are closer to that cell's associated tree node than to any other tree node.
- The probability of a given node **being selected for expansion is proportional to the volume of its Voronoi region**.
- Larger Voronoi regions correspond to areas of the configuration space that have been sampled less and are, therefore, less explored. Therefore, RRT tend to grow 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.
  - https://www.youtube.com/watch?v=F_tk6C9KGL4
- 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.

RRT: probabilistic completeness with no asymptotic optimality

- RRTs are **probabilistic complete (guaranteed to find a solution if one exists, given a sufficient number of random samples)** Intuition: As more samples are taken, the algorithm is increasingly likely to sample points that can bridge gaps and connect branches of the tree across narrow passages
- 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-surely *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}.

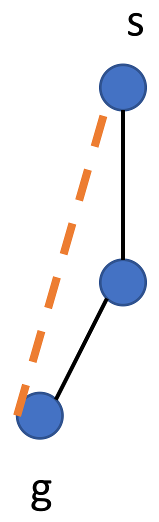
The RRT algorithm traps itself by disallowing new better paths to emerge (unlike hybrid A*)

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.)

$r(|V|)$ is chosen to roughly contains $\log(|V|)$ neighbours

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

Karaman S and Frazzoli E. Sampling-based motion planning with deterministic-calculus specifications.

Theorems [not required for exam]

Probabilistic completeness. Since $V_n^{RRG} = V_n^{RRT}$ (RRG has more edges but same vertices), for all n RRG has the same completeness properties as RRT, i.e.,

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

Asymptotic optimality. If the **Near** procedure returns all nodes in V within a ball of volume $Vol = \frac{\gamma \log \mathrm{N}}{N}$, $\gamma > 2^d\left(1 + \frac{1}{d}\right)$, where N=|V|, 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:
  - Combine optimality and computational efficiency, it is necessary to attempt connection to Θ(log N) nodes at each iteration.
  - Increase the number of connections as log(N).
  - Other similar algorithm: RRT*
- 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: 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.

Comparisons: PRM, RRT, RRG

- PRM: prebuilt map can be used on **multiple queries** (different pairs of source+goal nodes)
- RRT/RRG: **single query** (need to rerun when the source/goal changes)

- Is PRM, RRT, or RRG:
  - Sound?
  - Probabilistically complete?
    - guaranteed to find a solution if one exists, given a sufficient number of random samples
  - Asymptotically optimal?
    - as the number of samples increases, the path found by the algorithm necessarily converge to the true shortest possible path