Spring 26
ECE484
Lecture 17

Rapidly Expanding Random Trees

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)

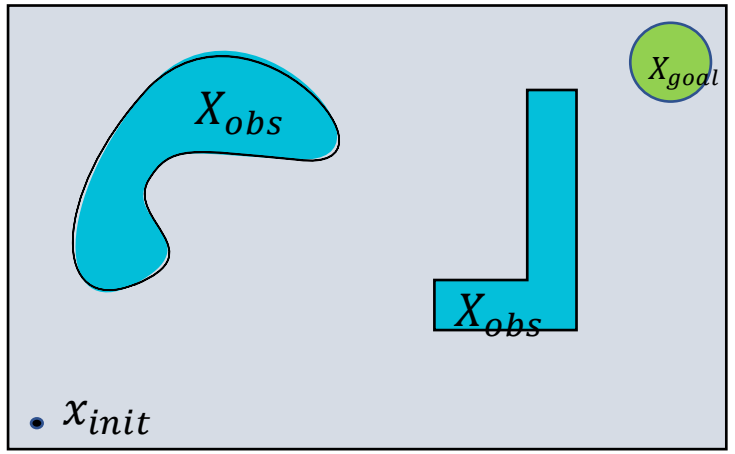
Motion planning problem

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.

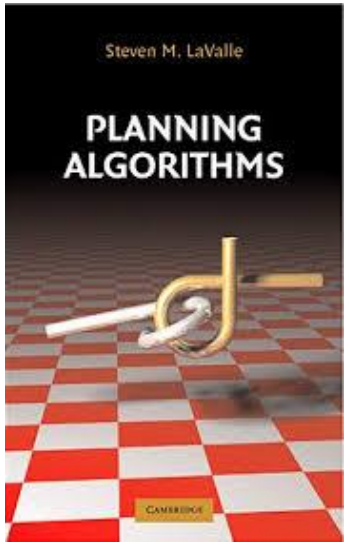


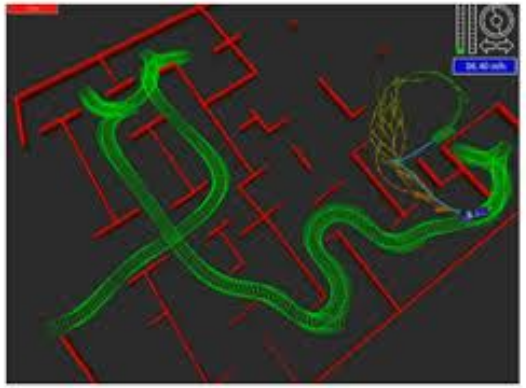
Motion planning is a central problem in robotics

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.

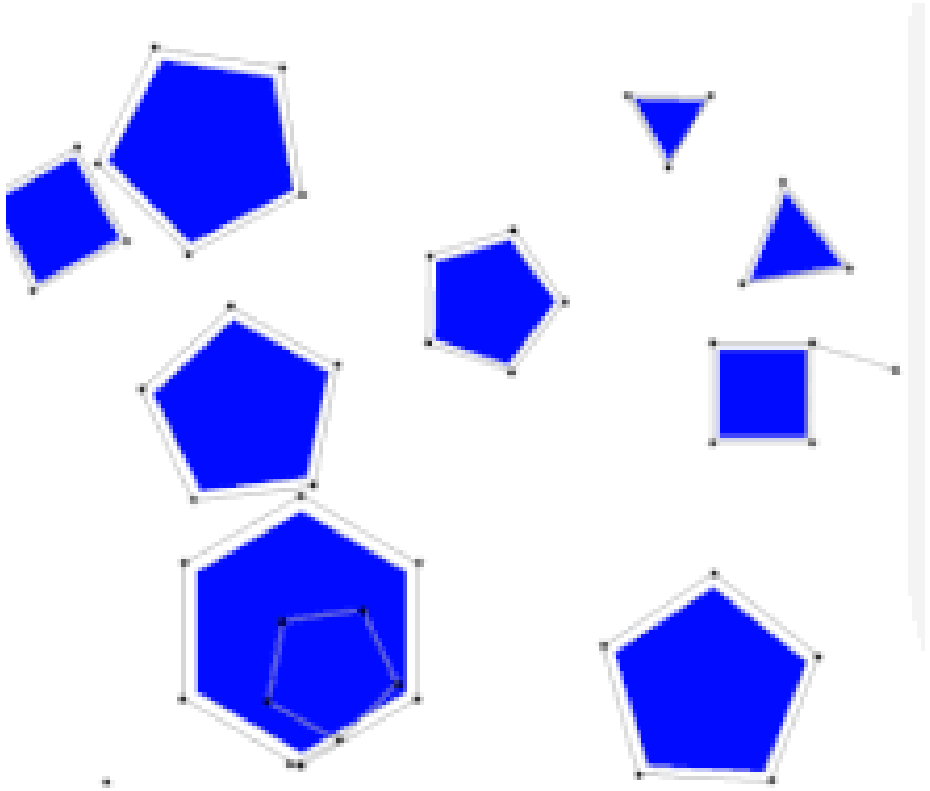
Retain some form of completeness, e.g., probabilistic completeness

Incremental sampling methods

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

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

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.

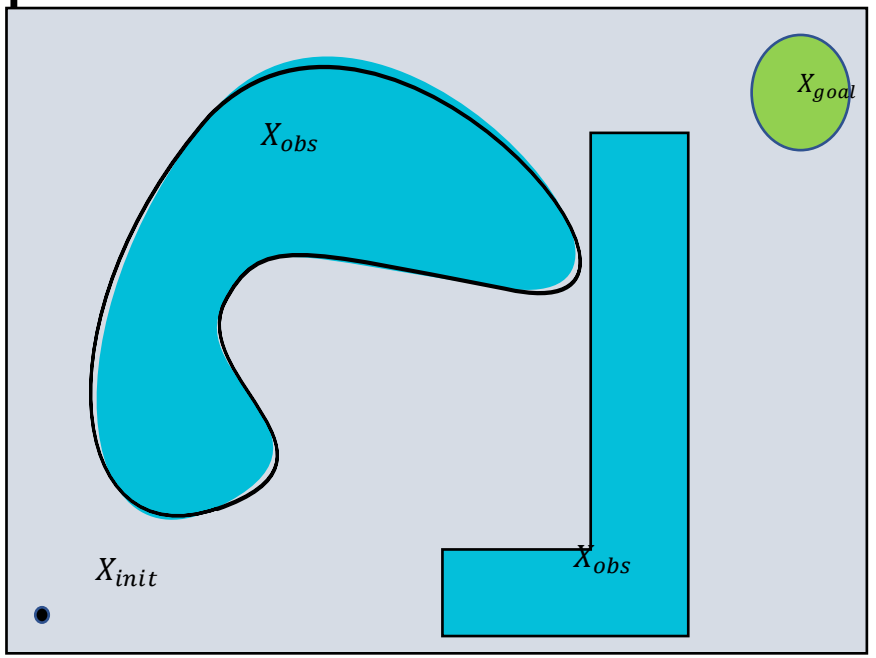


Fig. not robustly feasible.

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

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

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

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.

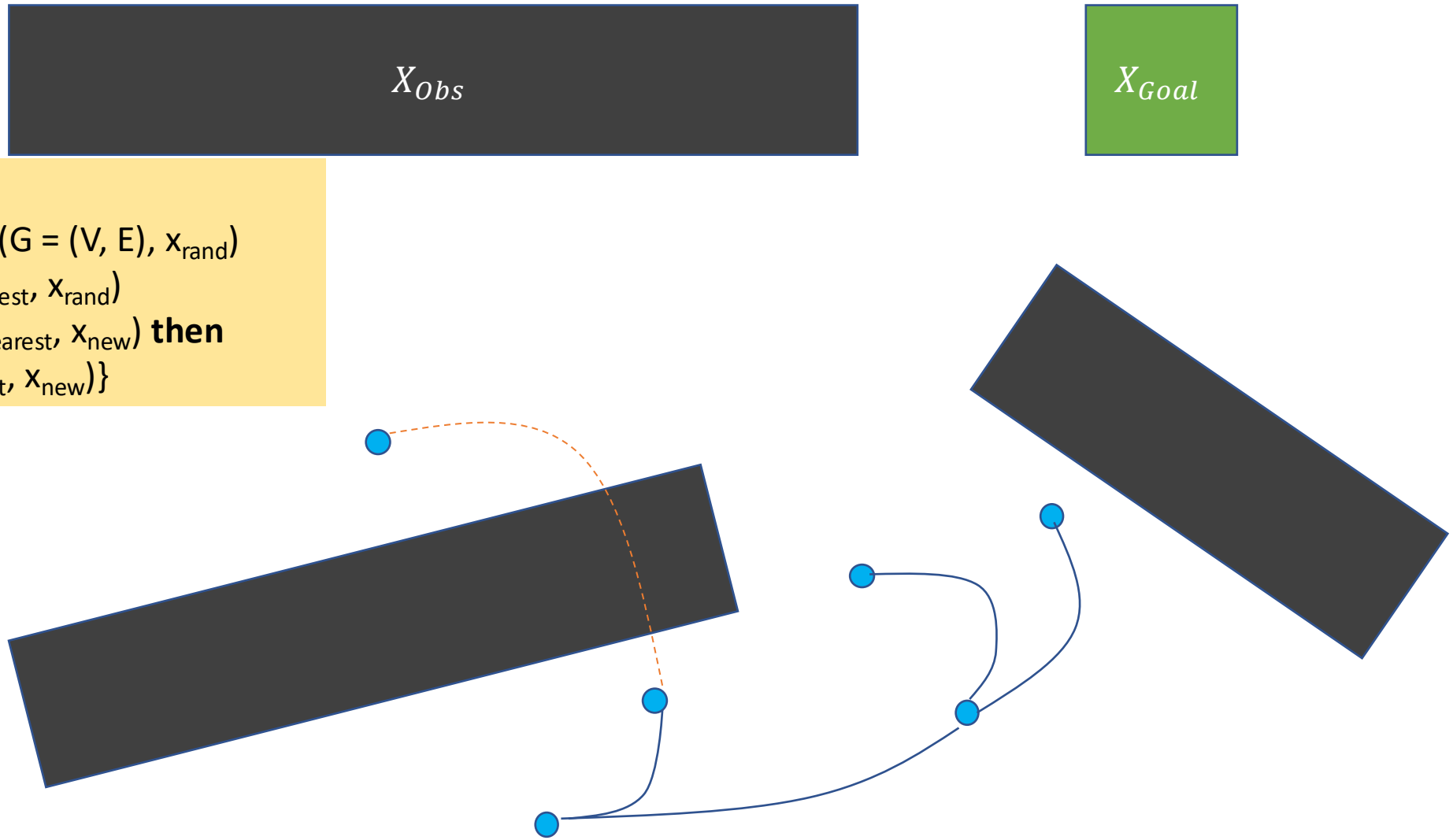
RRT Construction

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

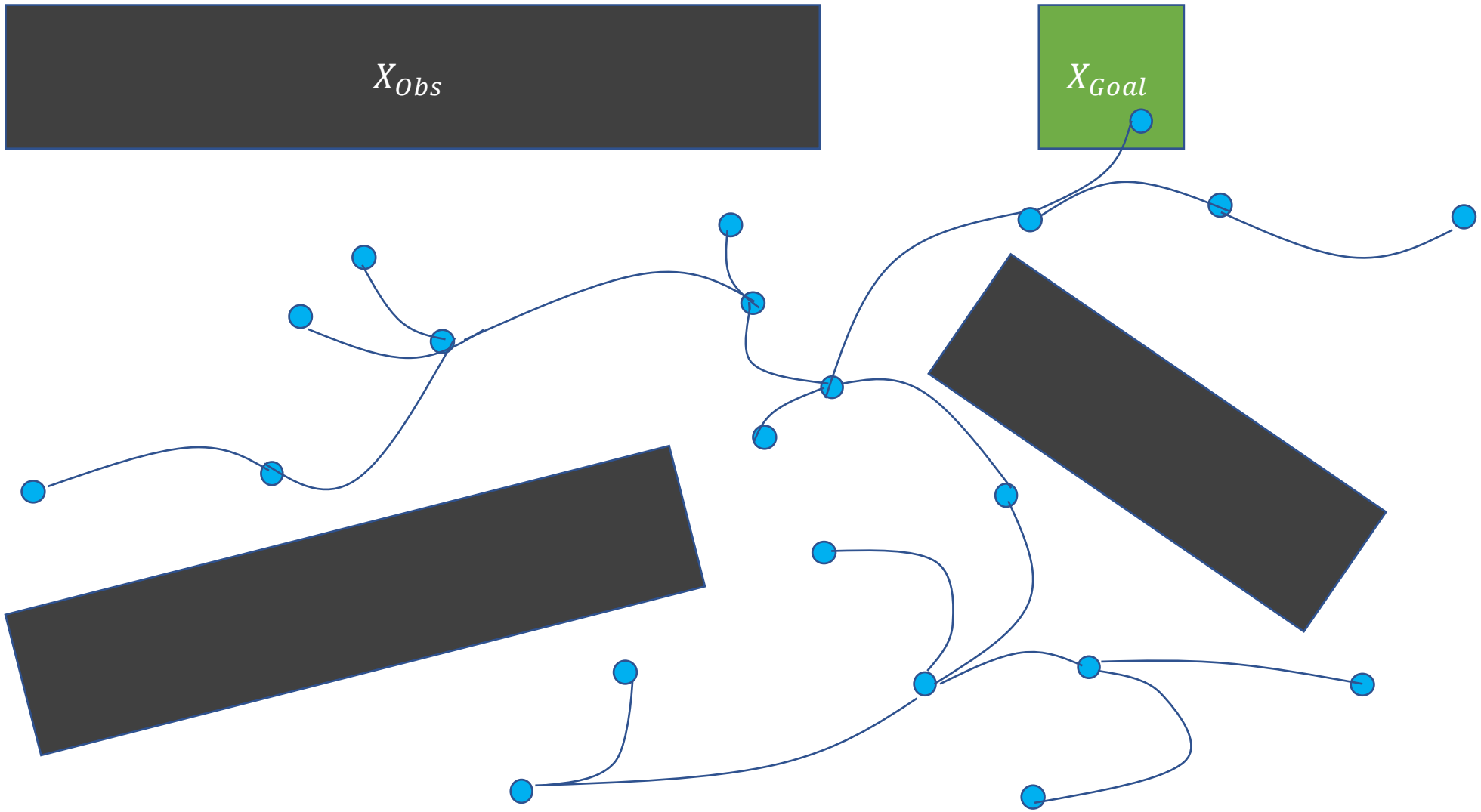
Nearest(G, x_{rand}): Finds the nearest vertex in G from x_{rand}

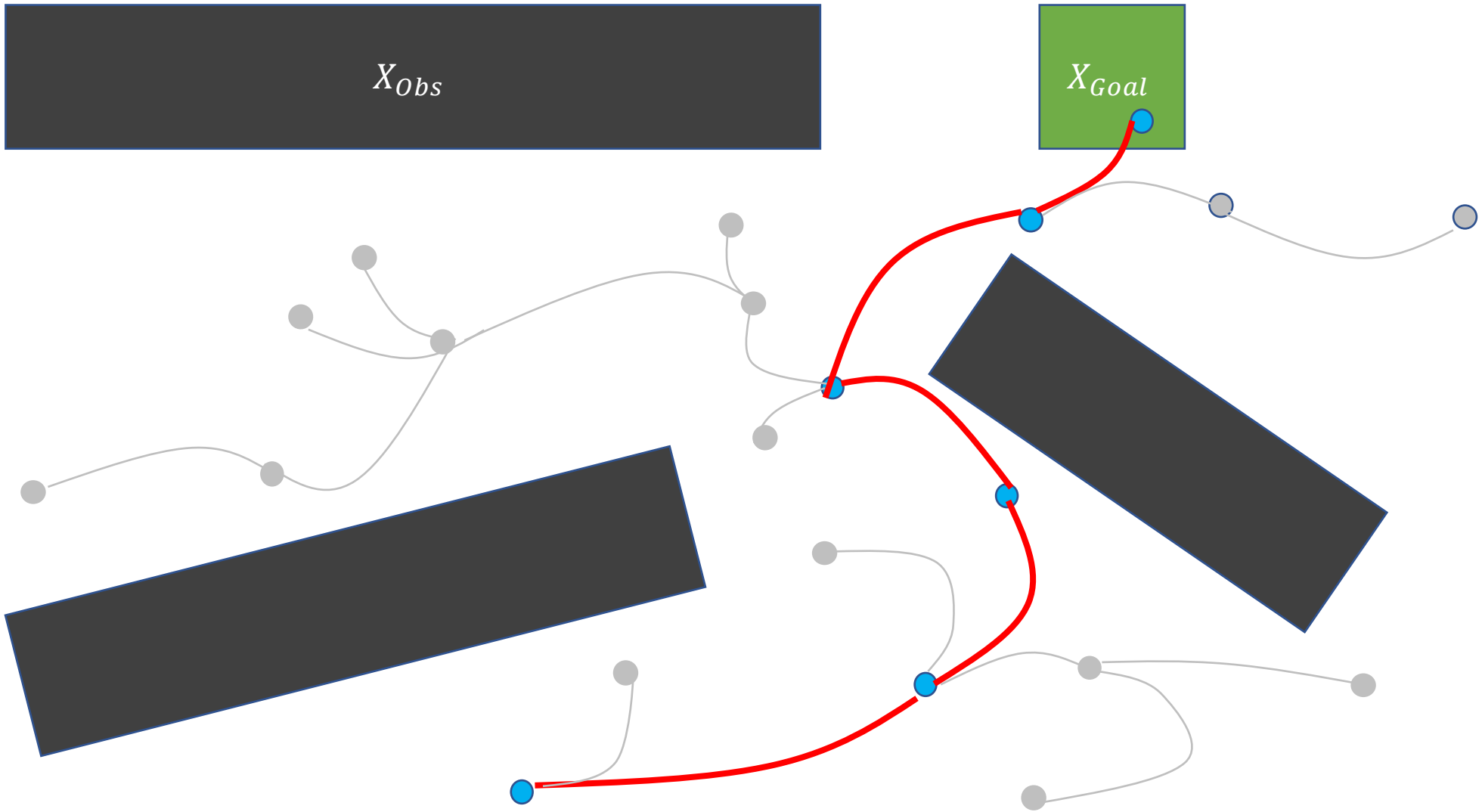
Steer(u, v): Tries to drive the robot from u to v and returns the point nearest to v that it could reach

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

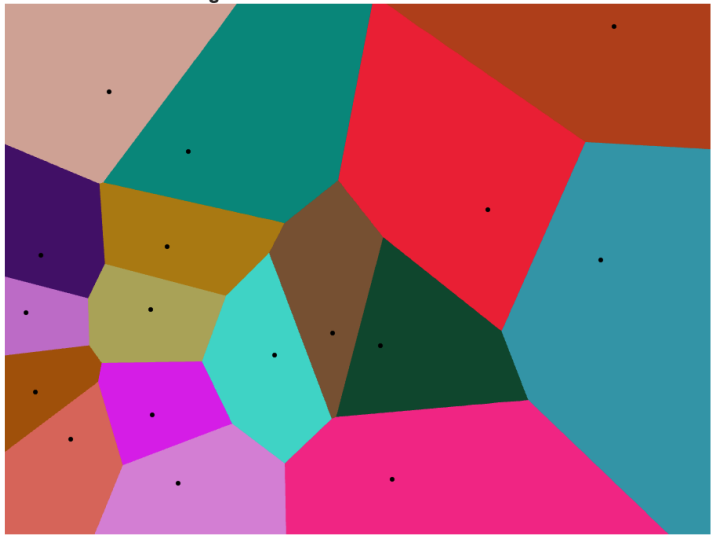


$x_{rand} \sim X_{free}$
$x_{nearest} \leftarrow$ Nearest(G = (V, E), x_{rand})
$x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand})
if ObtacleFree($x_{nearest}$, x_{new}) **then**
  E $\leftarrow$ E $\cup$ {($x_{nearest}$, x_{new})}


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}.

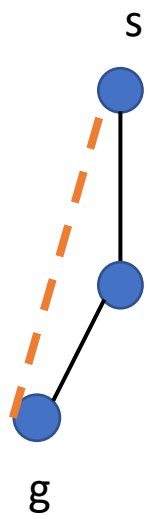
The RRT algorithm traps itself by disallowing new better paths to emerge (unlike hybrid A*)

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 G 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.

RRT vs. RRG: Algorithm Comparison

RRT — Rapidly-exploring Random Tree

1. xrand ~ Xfree

2. xnearest ← Nearest(G, xrand)

3. xnew ← Steer(xnearest, xrand)

4. If ObstacleFree(xnearest, xnew):
   V ← V ∪ {xnew}
   E ← E ∪ {(xnearest, xnew)}
   — one directed parent edge only —

Data structure: Tree (acyclic)
Edges per step: exactly 1
Connection: fixed step size η

RRG — Rapidly-exploring Random Graph

1. xrand ~ Xfree [identical]

2. xnearest ← Nearest(G, xrand) [identical]

3. xnew ← Steer(xnearest, xrand) [identical]

4. If ObstacleFree(xnearest, xnew):
   V ← V ∪ {xnew}
   E ← E ∪ {(xnearest,xnew),(xnew,xnearest)}
   Xnear ← Near(G, xnew, r(|V|))
   ∀ x ∈ Xnear : add edge if obstacle-free

Data structure: Graph (may have cycles)
Edges per step: 1 + |Xnear| = 1 + Θ(log n)
Connection: shrinking radius $r(|V|) = \min\{\gamma(\log|V|/|V|)^{(1/d)}, \eta\}$

Key difference: RRG rewires to ALL obstacle-free neighbors within ball r(|V|) — the RRT tree is a subgraph of RRG.

RRT vs. RRG: Graph Structure & Connectivity

RRT — Tree

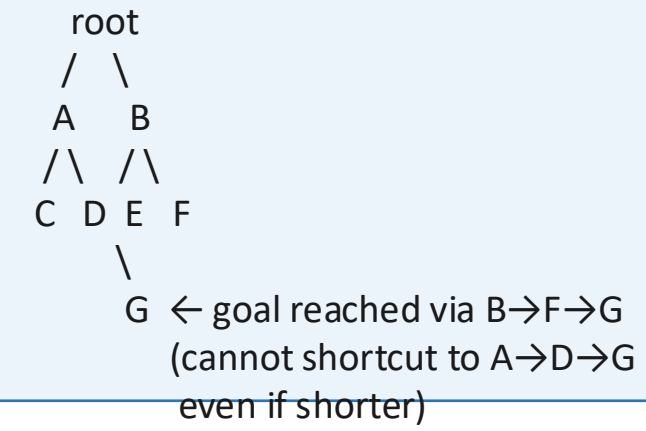
Structure: Directed acyclic graph (tree)

Edges: Each node has exactly one parent.
No cross-edges between branches.

Path: Exactly one path from root to
any node — no alternatives.

Optimality: Once a path to the goal is found,
the tree is locked in.
No mechanism to rewire or improve.

Example tree (schematic):



RRG — Graph (possibly cyclic)

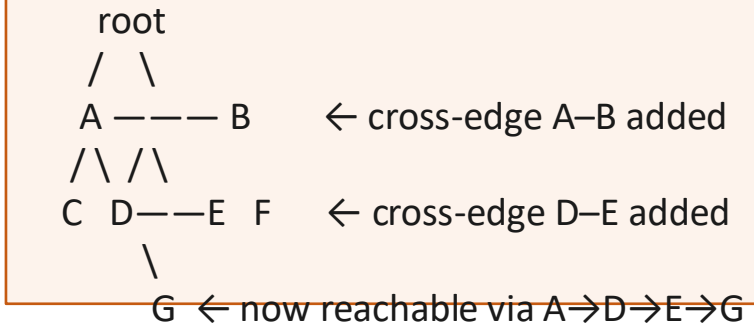
Structure: Undirected graph (may have cycles)

Edges: Each node connects to ALL
obstacle-free neighbors within
ball of radius r(|V|).

Path: Multiple paths between nodes.
Shorter alternatives can always
emerge as more nodes are added.

Optimality: Cross-edges enable rewiring:
if a shorter route to any node
is found, it is added to the graph.

Example graph (same nodes, more edges):



Vertex sets are identical: V_n^RRT = V_n^RRG. RRG adds O(log n) extra edges per step — no asymptotic overhead.

RRT vs. RRG: Optimality & Complexity

Property	RRT	RRG
Completeness	Probabilistically complete $P(V_n \cap X_{goal} = \emptyset) = O(e^{-bn})$	Probabilistically complete (identical — same vertex set V_n)
Asymptotic optimality	NOT optimal Almost surely converges to a suboptimal solution	OPTIMAL ✓ Almost surely: cost → c^* as $n \to \infty$ (Karaman & Frazzoli, 2011)
Edges / step	Exactly 1 (parent → new node)	$1 + \lvert X_{near} \rvert = 1 + \Theta(\log n)$ (all near neighbors within $r(\lvert V \rvert)$)
Time / step	$O(\log n)$ (k-d tree nearest query)	$O(\log n)$ (same class — no significant overhead!)
Memory	$O(n)$ nodes $O(n)$ edges	$O(n)$ nodes $O(n \log n)$ edges
Rewiring	None — tree is fixed once a node is placed	Yes — adds edges to all obstacle-free neighbors
Practical use	Fast, real-time planning Simpler to implement	Asymptotically optimal Slightly higher memory

Takeaway: RRG is asymptotically optimal with $O(\log n)$ overhead per step — the same complexity class as RRT.

RRT vs. RRG: Worked Example — Narrow Passage

Setup: 2D state space. An obstacle wall splits the space into left (start s) and right (goal g) chambers, connected by a narrow corridor.

RRT result

State space schematic:

[s] ——— ——— ——— ——— [g]
↑ narrow gap

RRT samples nodes in both chambers.
Eventually a node passes through the
corridor and connects s to g.

Tree locks in that path immediately.
New nodes keep connecting to the
nearest existing node — reinforcing
the same route, even if it goes
through a wide detour.

Result after n → ∞ samples:
Path length → constant $c_{RRT} > c^*$

Why? Tree structure prevents rewiring.

RRG result

Same state space, same samples.

[s] ——— ——— ——— ——— [g]
↑ narrow gap

RRG finds the same initial path.
But every new node also connects to
ALL obstacle-free neighbors within
ball of radius $r(|V|) = \Theta(\log|V|/|V|)^{(1/d)}$.

As r shrinks and more nodes land near
the corridor, shorter cross-connections
form, progressively shortening the path.

Result after n → ∞ samples:
Path length → c^* (optimal!)

Why? Graph edges allow alternative
routes to emerge and be retained.

Theorem (Karaman & Frazzoli, 2011): RRT is NOT asymptotically optimal. RRG IS asymptotically optimal.