Sampling-Based Motion Planning: From PRMs to RRTs

Instructor: Chris Mavrogiannis

TAs: Kay Ke, Gilwoo Lee, Matt Schmittle

*Slides based on or adapted from Sanjiban Choudhury, Steve Lavalle, Peter Allen, Pieter Abbeel

Logistics

- Lab 3
  - Deadline Friday March 6th
  - Demo Thursday March 5th (recitation slots)
  - **Extra Credit important for final project**
- Final Project
  - Out today!
  - Demo Thursday March 12th
  - Short writeup due Monday 16th
- Special Topics

Probabilistic Roadmap Path Planning

- Explicit Geometry based planners impractical in high dimensional spaces.
- Exact solutions with complex geometries provably exponential
- Sampling-based planners can often create plans in high-dimensional spaces efficiently
- Rather than Compute the C-Space explicitly, we Sample it

Slide credit: Peter Allen

Completeness in Motion Planning

- **Complete Planner**: always answers a path planning query correctly in bounded time (return a path if one exists, otherwise report failure)
- ***Probabilistically* Complete Planner**: Probability of returning a path approaches 1 as more samples are generated.
- ***Resolution*-Complete Planner**: Probability of finding a path approaches 1 as the resolution becomes finer.

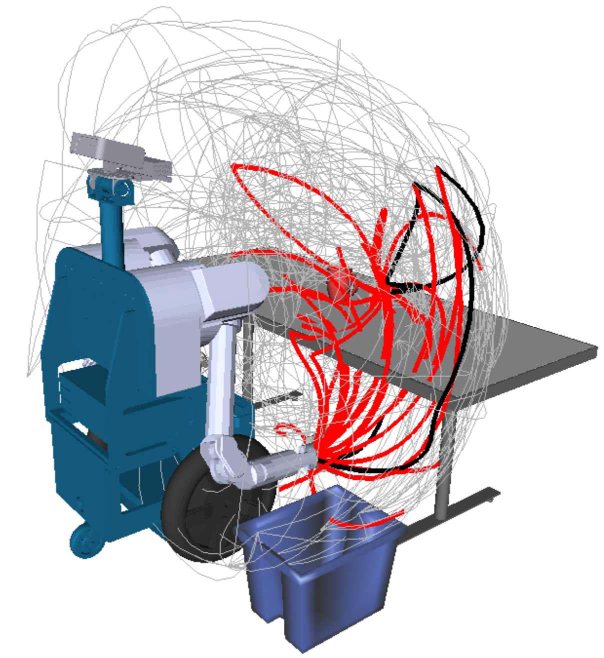
Slide credit: Peter Allen

Sampling-Based Planners

- Do not attempt to explicitly construct the C-Space and its boundaries.
- Simply need to know if a single robot configuration is in collision
- Exploits simple tests for collision with full knowledge of the space
- Collision detection is a separate module- can be tailored to the application
- As collision detection improves, so do these algorithms
- Different approaches for single-query and multi-query requests

Slide credit: Peter Allen

This is the PRM Algorithm!



PRM = Probabilistic Roadmap

1. Sample vertices randomly and collision check

2. Try to connect vertices within radius (or k NN)
   - collision check potential edges

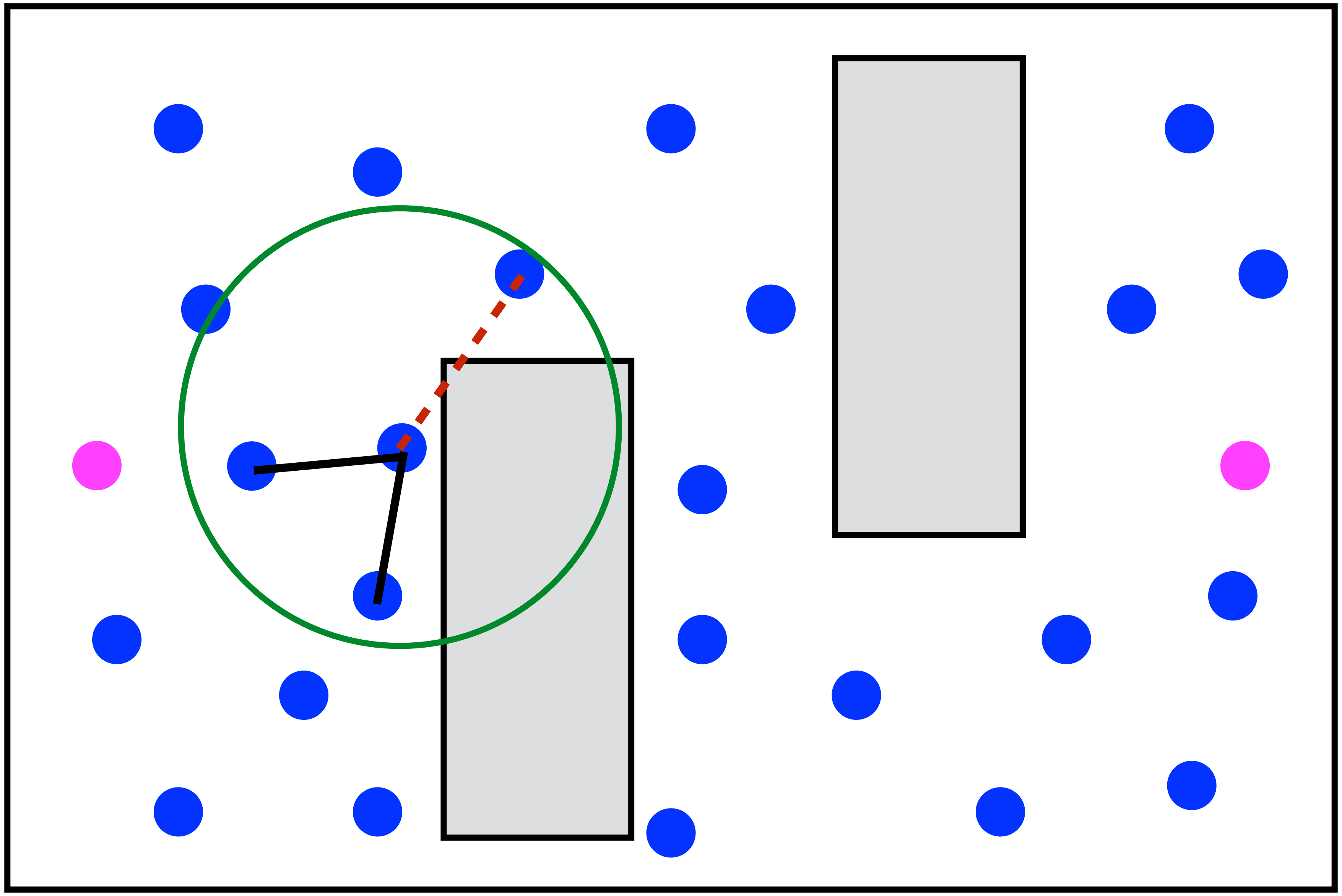
3. Search graph to find a solution
   - can reuse graph across multiple queries

Probabilistically Complete

L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE TRO, 12(4):566–580, June 1996.

What is the optimal radius?

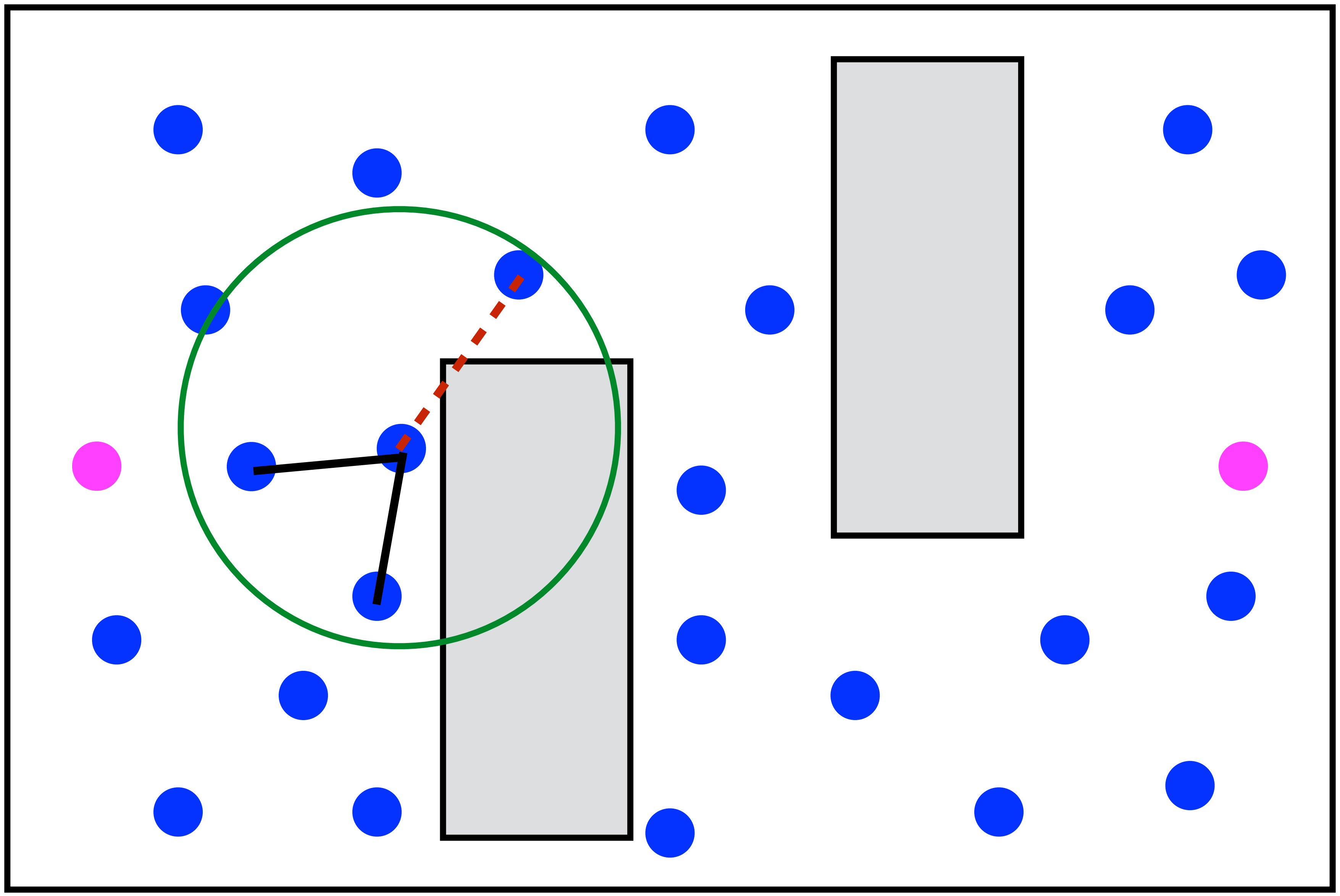
What happens if radius too large? too small?



What is the optimal radius?

Set the radius to $r = \gamma \left(\frac{\log |V|}{|V|} \right)^{1/d}$ where magic constant!

$$\gamma \geq 2(1 + 1/d)^{1/d} \frac{\mu(\mathcal{C}_{free})}{\zeta_d}$$



Also known as a Random Geometric Graph (RGG)

This is the PRM* Algorithm!

1. Sample vertices randomly

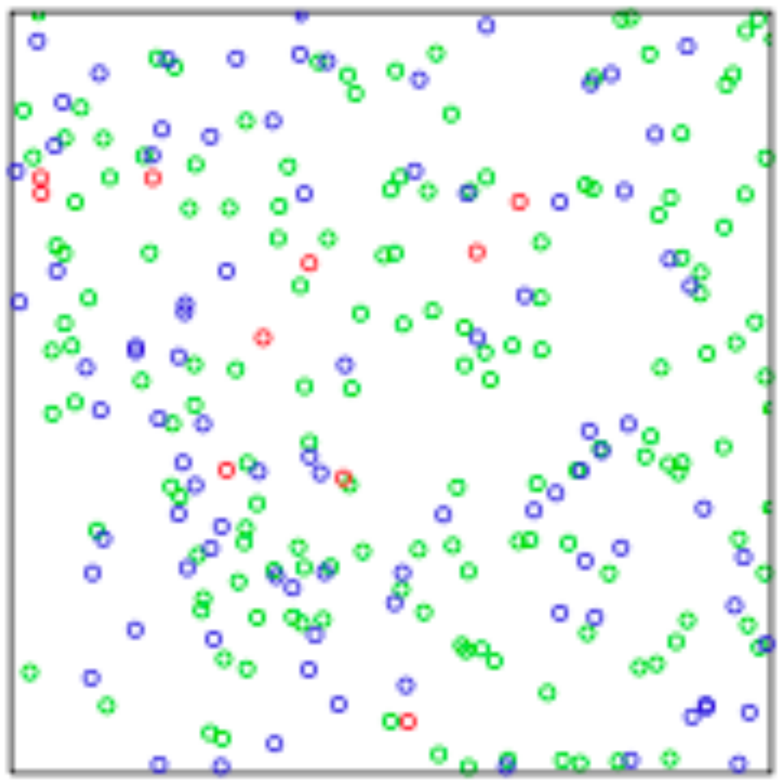
2. Use optimal radius formula to connect vertices

3. Search graph to find a solution

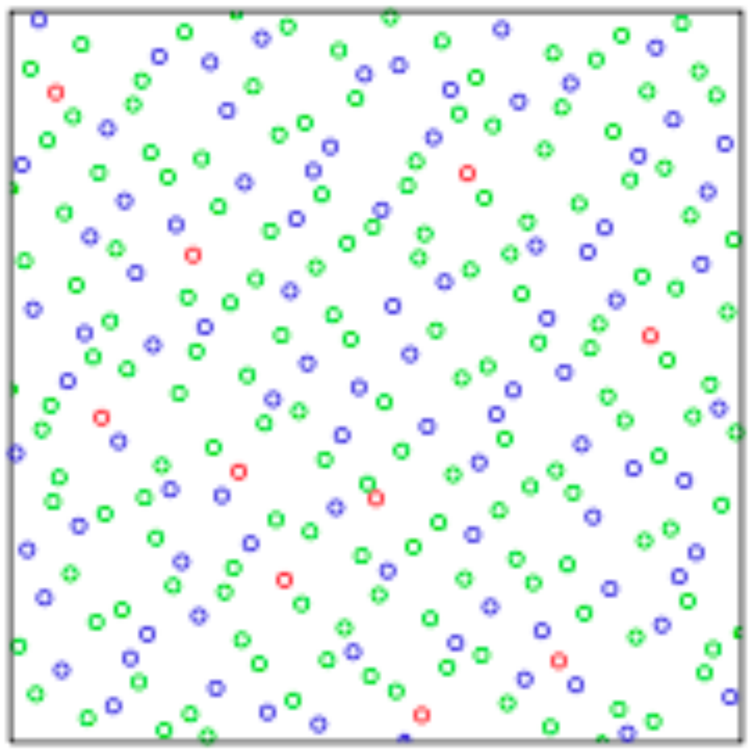
Theorem: Probabilistically complete AND Asymptotically optimal

"Sampling-based Algorithms for Optimal Motion Planning"
Sertac Karaman and Emilio Frazzoli, IJRR 2011

Can we do better than random?



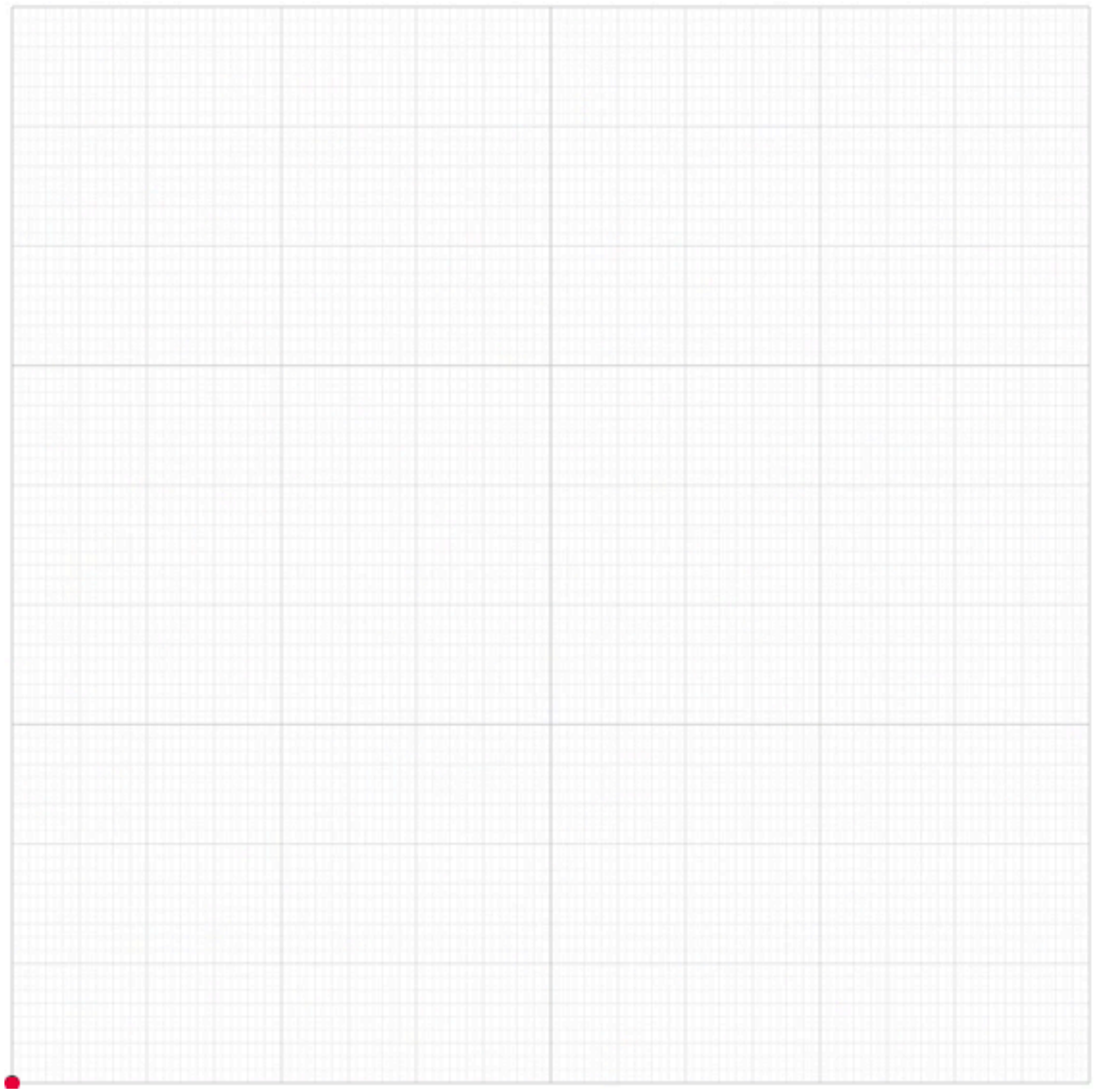
Uniform random sampling tends to clump



Ideally we would want points to be spread out evenly

Question: How do we do this without discretization?

Halton Sequence

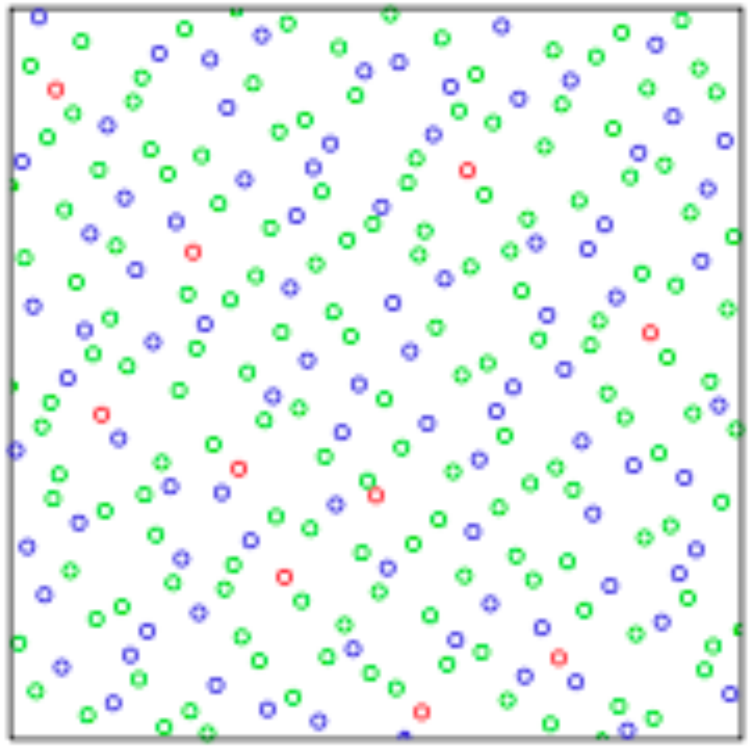


Generalization of
Van de Coruput Sequence

Intuition: Create a sequence using prime numbers that uniformly densify space

Link for exact algorithm:
https://observablehq.com/@jrus/halton

How do we connect vertices?



Halton sequences have much better coverage

(i.e. they are low dispersion)

Connect vertices that are within a radius of

$$r = \gamma \left(\frac{1}{|V|} \right)^{1/d}$$

(as opposed to:)

$$r = \gamma \left(\frac{\log |V|}{|V|} \right)^{1/d}$$

This is the gPRM Algorithm!

1. Sample vertices randomly

2. Use optimal radius formula to connect vertices

3. Search graph to find a solution

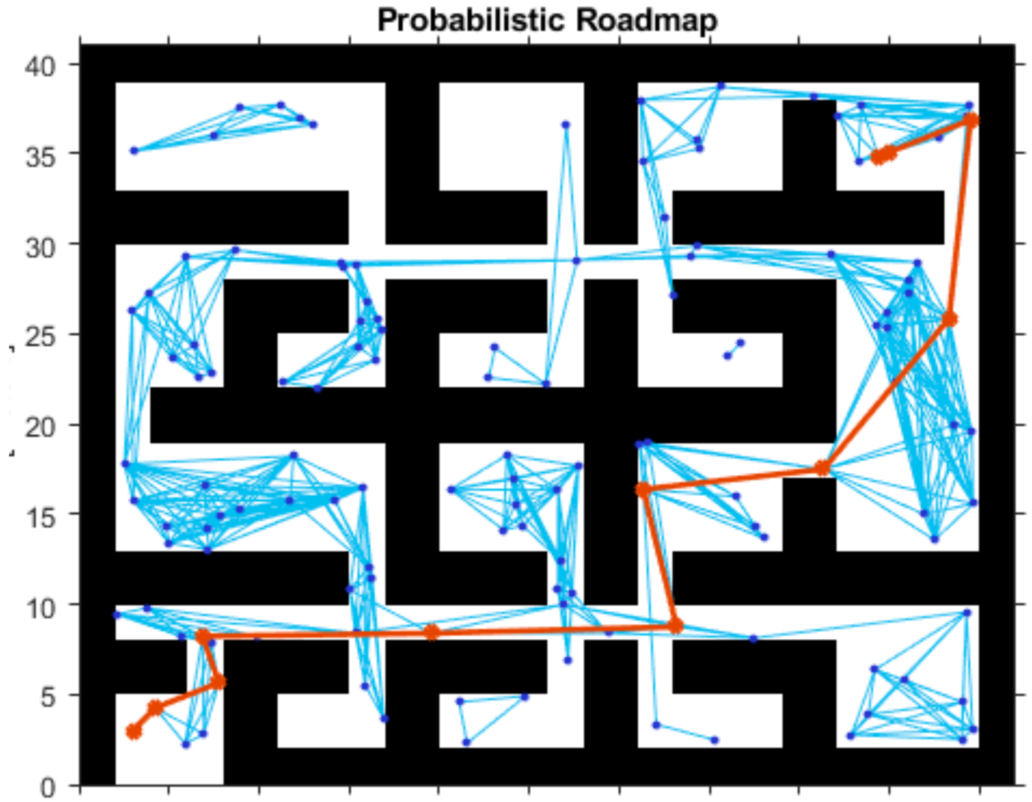
Theorem: Probabilistically complete AND Asymptotically optimal AND Asymptotic rate of convergence

"Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance"
Lucas Janson, Brian Ichter, Marco Pavone, IJRR 2017

What makes a good graph?

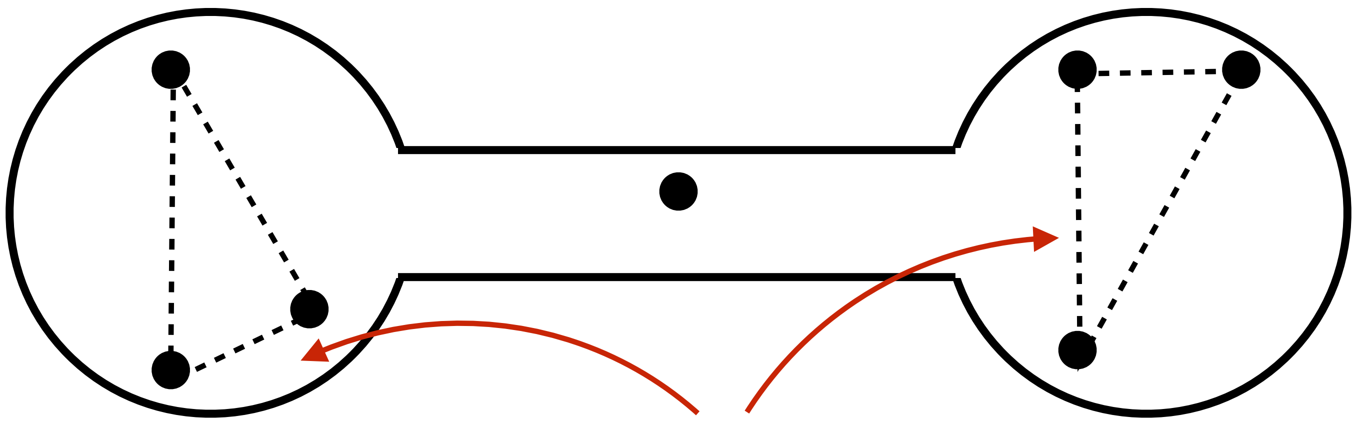
1. A good graph must be sparse (both in vertices and edges)

2. A good graph must have good coverage



3. A good graph must have the same connectivity of free space

The Narrow Passage: Planning's boogie man!



Why is narrow passage mathematically hard to plan in?

Mathematical Question:

How many samples do we need to connect the space (with high probability)?

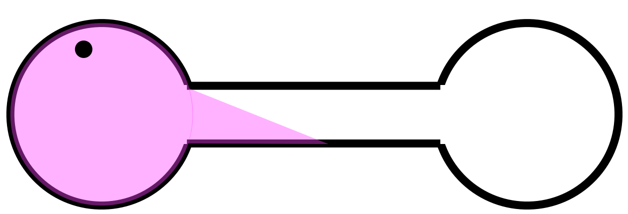
How many samples do we need?

Theorem [Hsu et al., 1999] Let $2n$ vertices be sampled from X_{free}.
Then the roadmap is connected with probability at least $1-\gamma$ if:

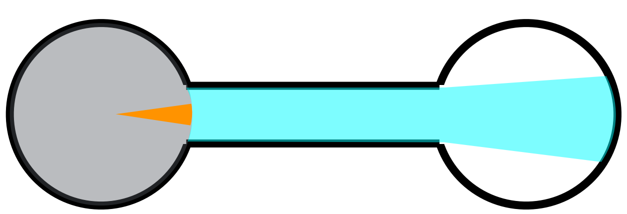
$$n \geq \left\lceil 8 \frac{\log\left(\frac{8}{\epsilon\alpha\gamma}\right)}{\epsilon\alpha} + \frac{3}{\beta} + 2 \right\rceil$$

The shape of free C-space is dictated by α, β, $\epsilon \in [0,1]$

Visibility of free space (ϵ)



Expansion of visibility (α, β)

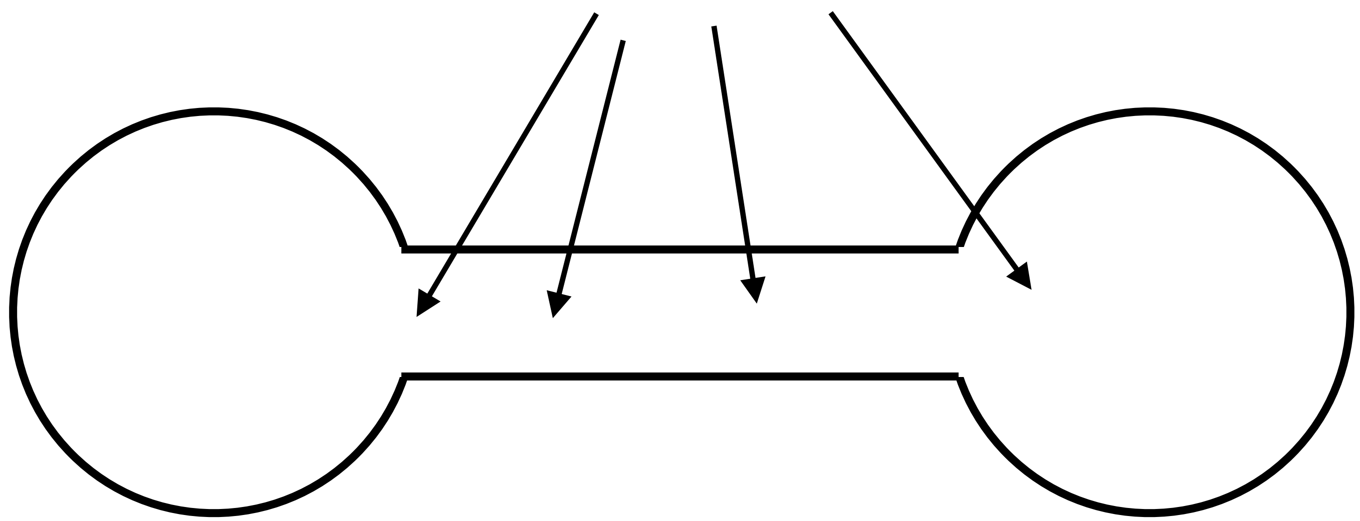


Narrow passage has small values of α, β, ϵ

Hence, needs more samples to find a path

How do we bias sampling?

We somehow need more samples here



1. Sample near obstacle surface?

   V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. 1999

2. Add samples that are in between two obstacles?

   D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners.2003.

3. Train a learner to detect the narrow passages?

   B. Ichter, J. Harrison, M. Pavone. Learning Sampling Distributions for Robot Motion Planning, 2018

Summary of ways to create graphs

Algorithms	**How to sample vertices?**	**How to connect vertices?**
Lattice	Discretize	connectivity rule
PRM	Uniform random	r-disc, k-nn
PRM*	Uniform random	optimal r-disc, k-nn
gPRM	Halton sequence	optimal r-disc, k-nn
Bridge	Sample with bridge test	any visible points
Gaussian	Sample near obstacles	r-disc, k-nn
MAPRM	Sample along medial axis	r-disc, k-nn
Approx. Visibility Graph	Sample on surface of obstacles	any visible points
Learnt Sampler	Use CVAE to approximate free space	optimal r-disc, k-nn

What graph should I use?

Low dim (2-3): Discretize evenly

Higher dim(>=4): Halton sequence

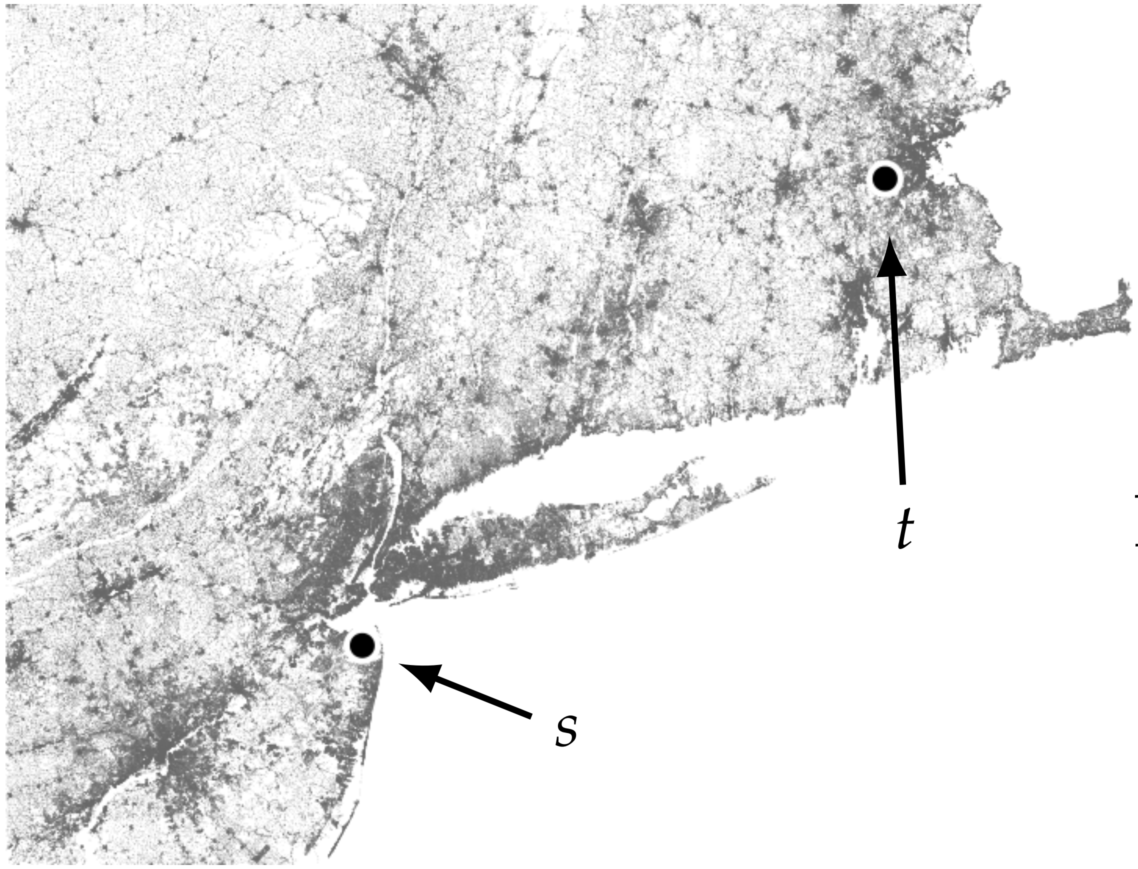
Narrow passage: Bias sampling

So far we have looked at an Explicit Evaluated Graph

Is it a good idea to evaluate every edge that we discover?

Is it a good idea to explicitly store the entire graph in memory?

Do we need to store this whole graph?



DIMACS dataset

1.5 million vertices, 3.67 million edges

A* search touches a small fraction

Bidirectional A*

Key Idea:

Use implicit unevaluated graphs

Implicit graph

1. Initialize with at least one seed vertex

   E.g. The start state, the goal state, both (for bi-directional search)

2. Provide a generative function for producing successors

$$\texttt{succ}(u) = \{(u, v) | (u, v) \in E\}$$

function returns both

- new vertices to add to V
- new edges to add to E

Searching with implicit graph

1. Start the search with initial vertex (start, goal, or both)

2. Whenever search chooses to "expand a vertex"

   - call $\texttt{succ}(u)$

   - get successor vertices

   - evaluate the edges

   - add these vertices to the search queue

Explicit vs Implicit Graphs

- Explicit Graph:

– All **vertices** are identified individually and represented separately.
– All **edges** are identified individually and represented separately.

- Implicit Graph:

– Only a subset, possibly only one, of the vertices is given an explicit representation. (The others are implied.)
– Only a subset, and possibly zero, of the edges is given an explicit representation.
– A set of "operators" is provided that can be used to construct "new" edges and vertices.

Slide credit: Steve Tanimoto

Planning with differential constraints

Differential constraints

So far we assumed only kinematic constraints

$$q \notin \mathcal{C}_{obs}$$

When is this assumption true?

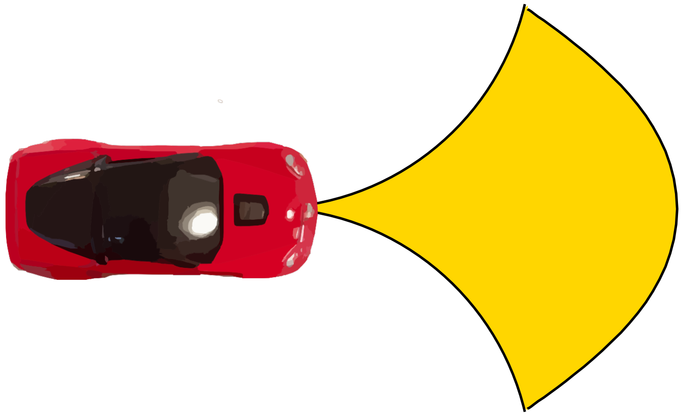
- when controller can track any path

- when robots move very slowly (stop and turn)

Differential constraints

We now introduce differential constraints

$$\dot{q} = f(q, u)$$



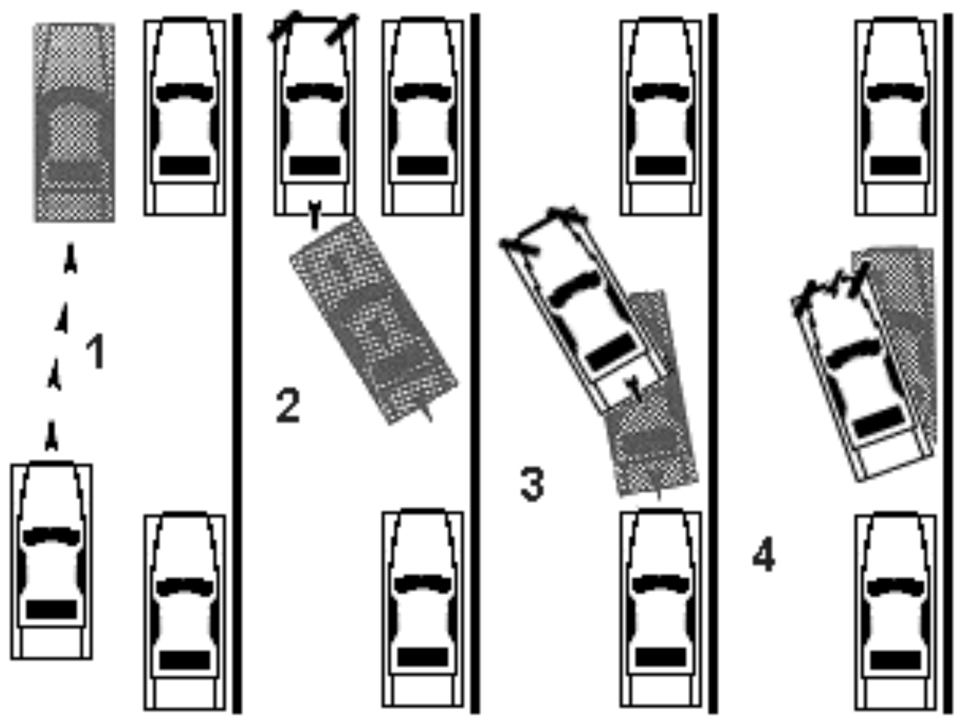
Two new terms:

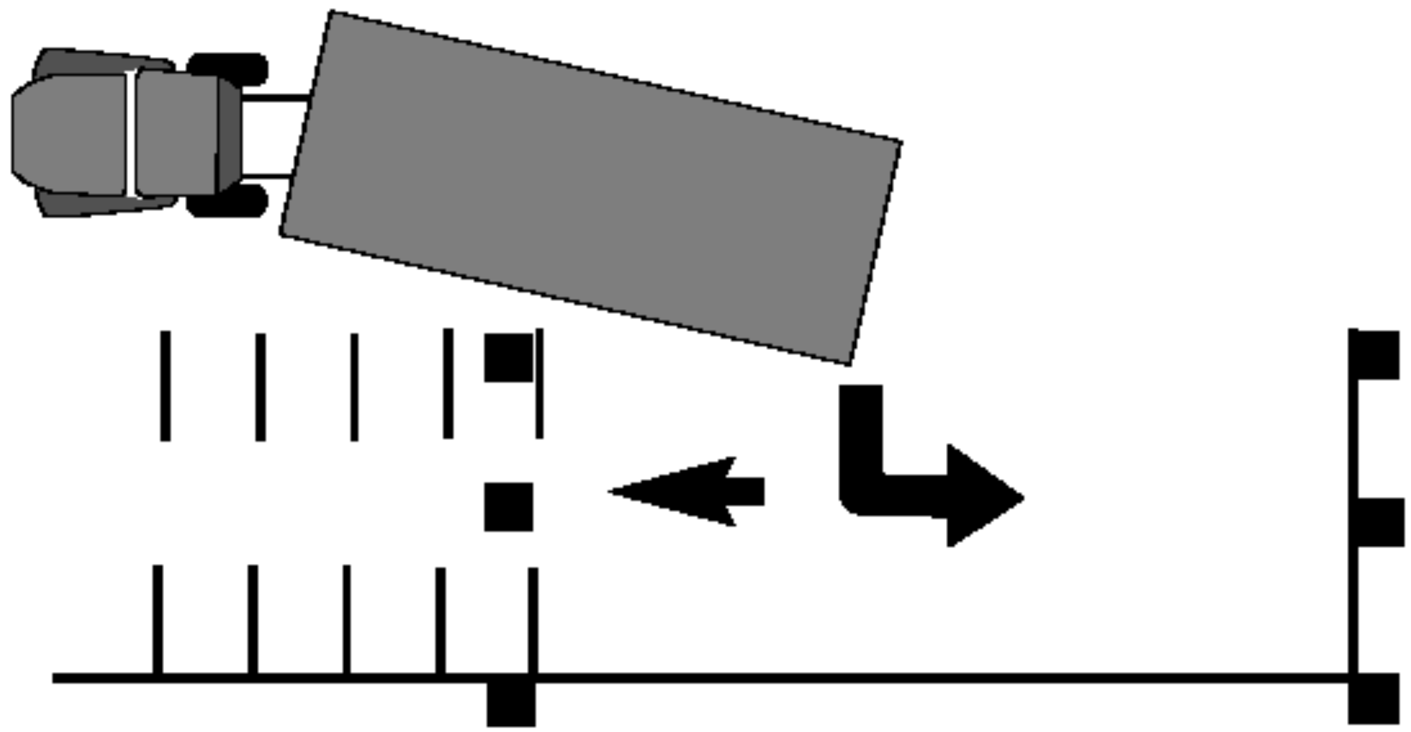
1. Introduction of control space
2. Introduction of an equality constraint

Differential constraints make things even harder

Holonomic constraints: Constraints that can be integrated, i.e.

$$\dot{q} = f(q,u) \longrightarrow g(q,u) = 0$$

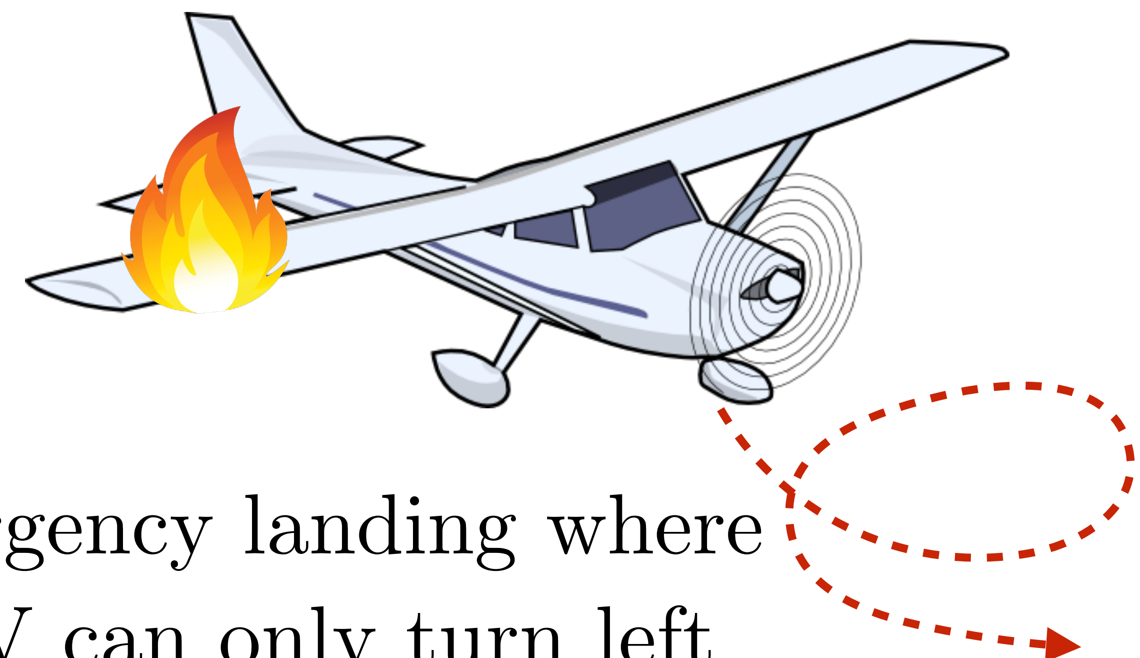




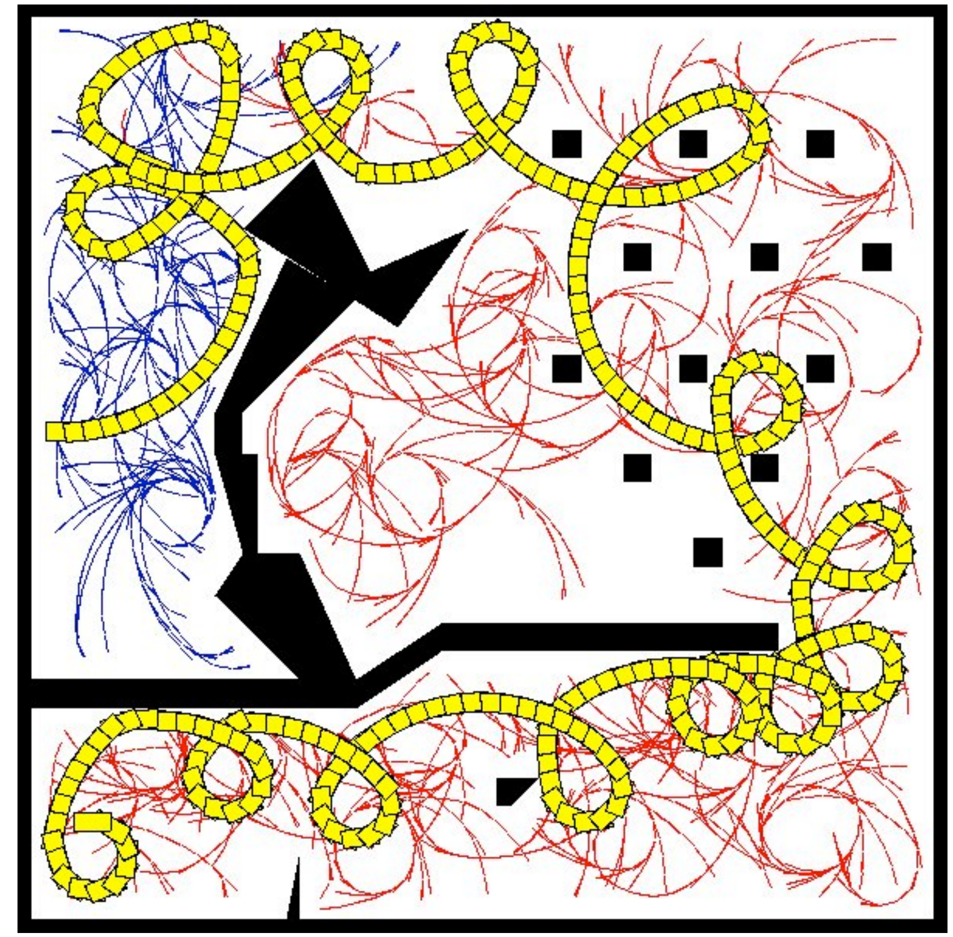
Non-holonomic constraints: Constraints that can't be integrated, i.e.

i.e. the system is trapped in some sub-manifold of the config space

Differential constraints make things even harder



Emergency landing where UAV can only turn left



"Left-turning-car"

Non-holonomic constraints: Constraints that can't be integrated, i.e.

i.e. the system is trapped in some sub-manifold of the config space

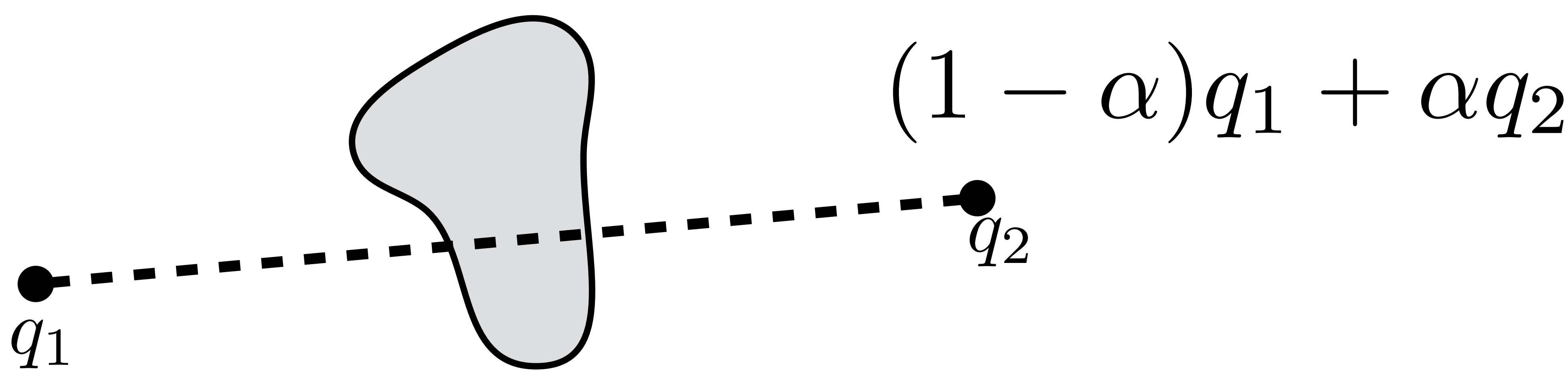
How do we incorporate differential constraints in our framework?

Only change the STEER function!

Recap: STEER function for geometric

$$\texttt{steer}(q_1, q_2)$$

A steer function tries to join two configurations with a feasible path

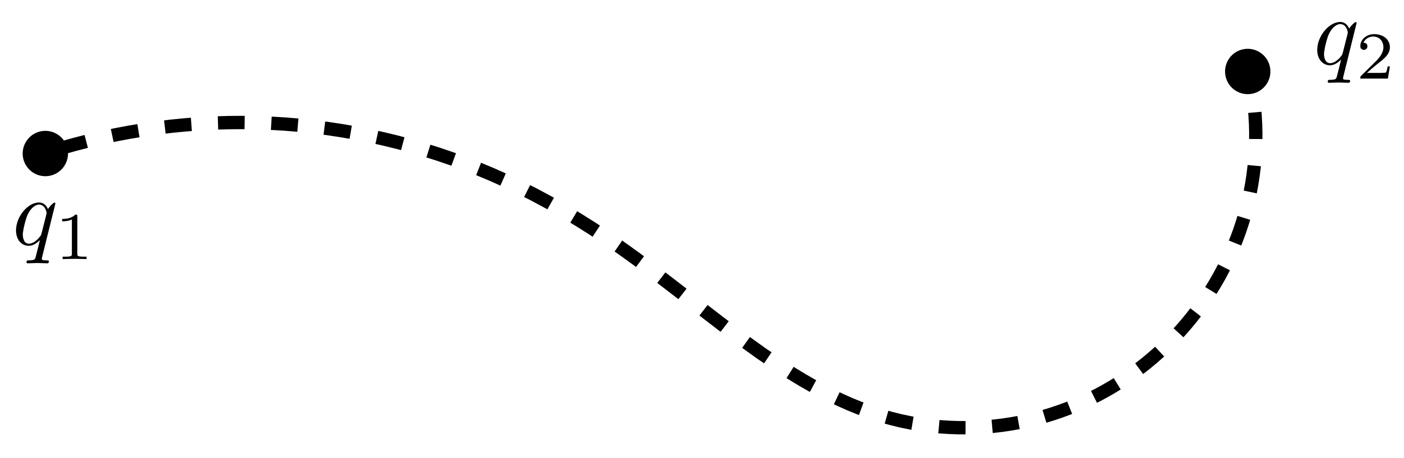


Example: Connect them with a straight line and check for feasibility

STEER function incorporate dynamics!

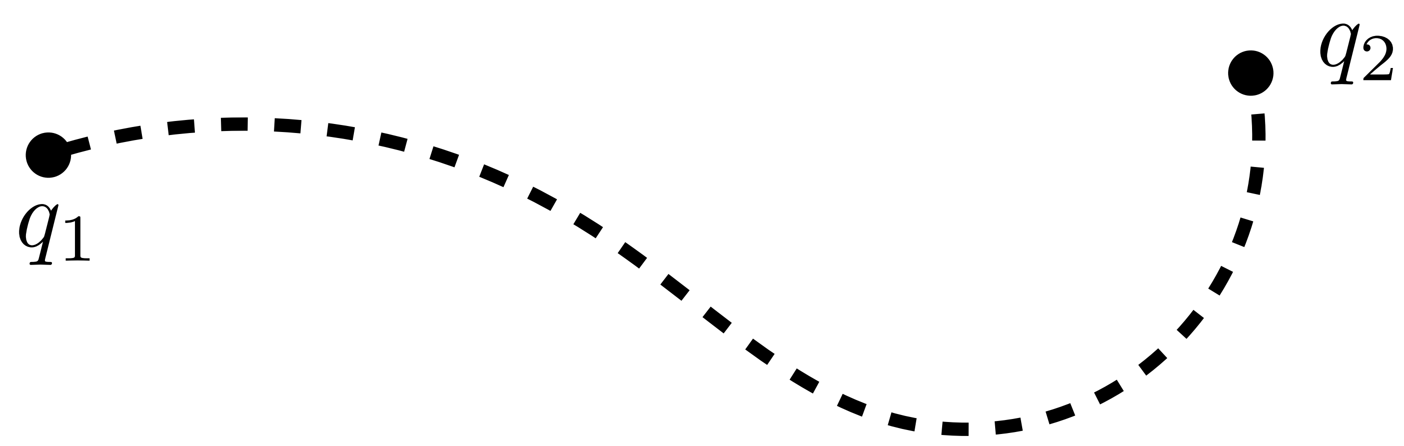
$$\texttt{steer}(q_1, q_2)$$

A steer function tries to join two configurations
with a dynamically feasible path



STEER function incorporate dynamics!

$$\texttt{steer}(q_1, q_2)$$



Formally called the boundary value problem (BVP)

Find a control trajectory $u(t) \in U$

Such that $q(0) = q_1$, $q(t_f) = q_2$

$$\dot{q}(t) = f(q(t), u(t))$$

How do we come up with STEER function?

There are three possible cases

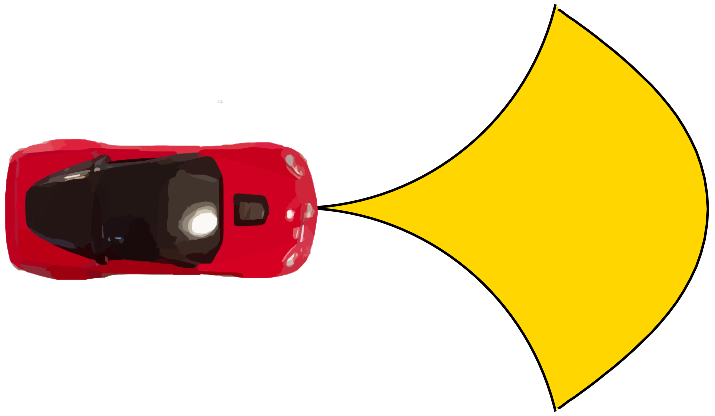
Case 1: We can analytically solve the BVP :)

Case 2: We need to numerically solve the BVP.

Case 3: We can't even solve the BVP!

Case 1: We can analytically solve the BVP

Consider the dynamics of your racecar



$$\begin{bmatrix} \dot{x} \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ C\tan\delta \end{bmatrix}$$

$$|\delta| \leq \delta_{max}$$

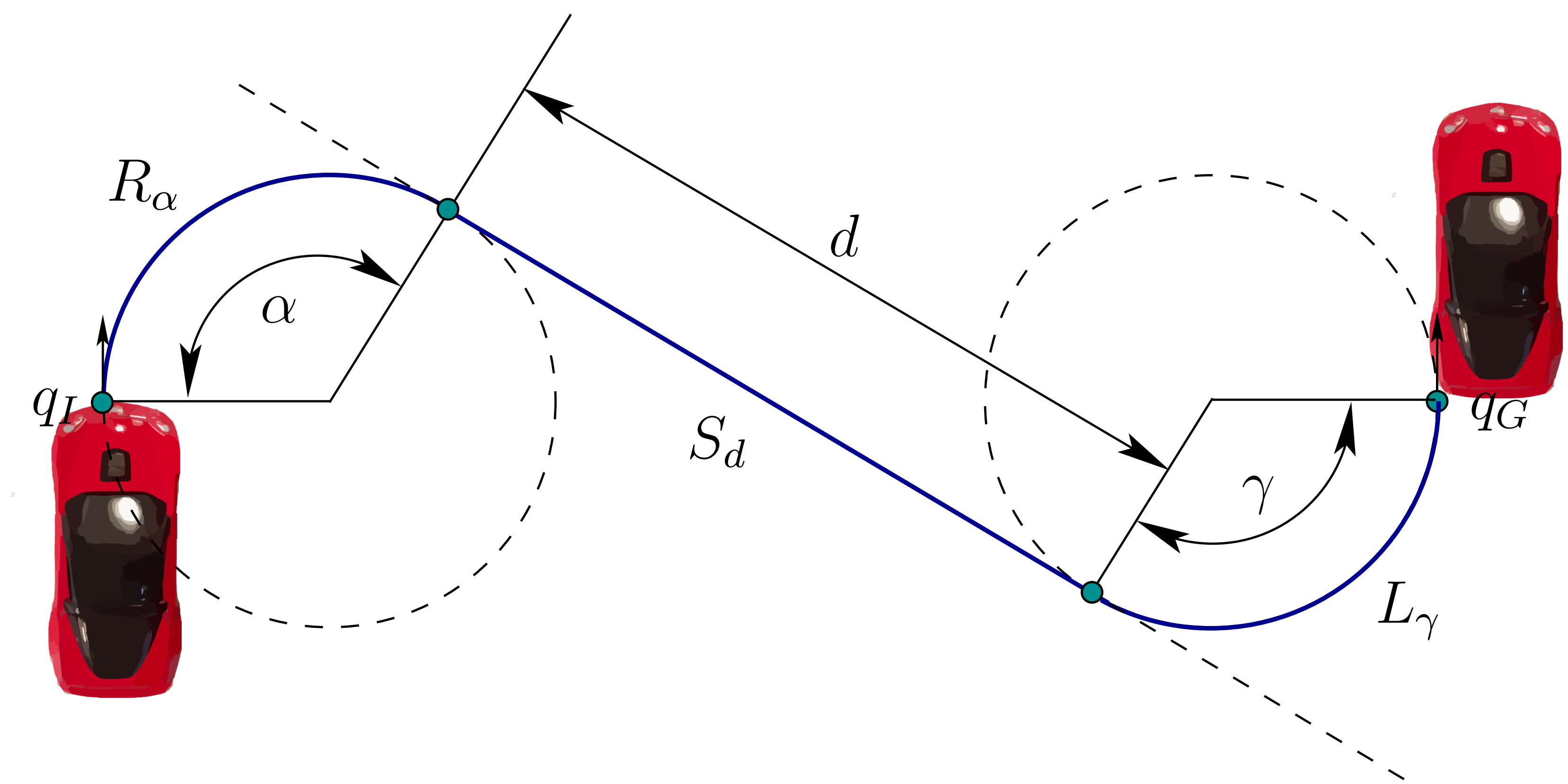
$$q_1 = (x_1, y_1, \theta_1)$$

$$q_2 = (x_2, y_2, \theta_2)$$

Can we solve this analytically?

Case 1: We can analytically solve the BVP

Yes! The solution is called the Dubins path 



$$R_\alpha S_d L_\gamma$$

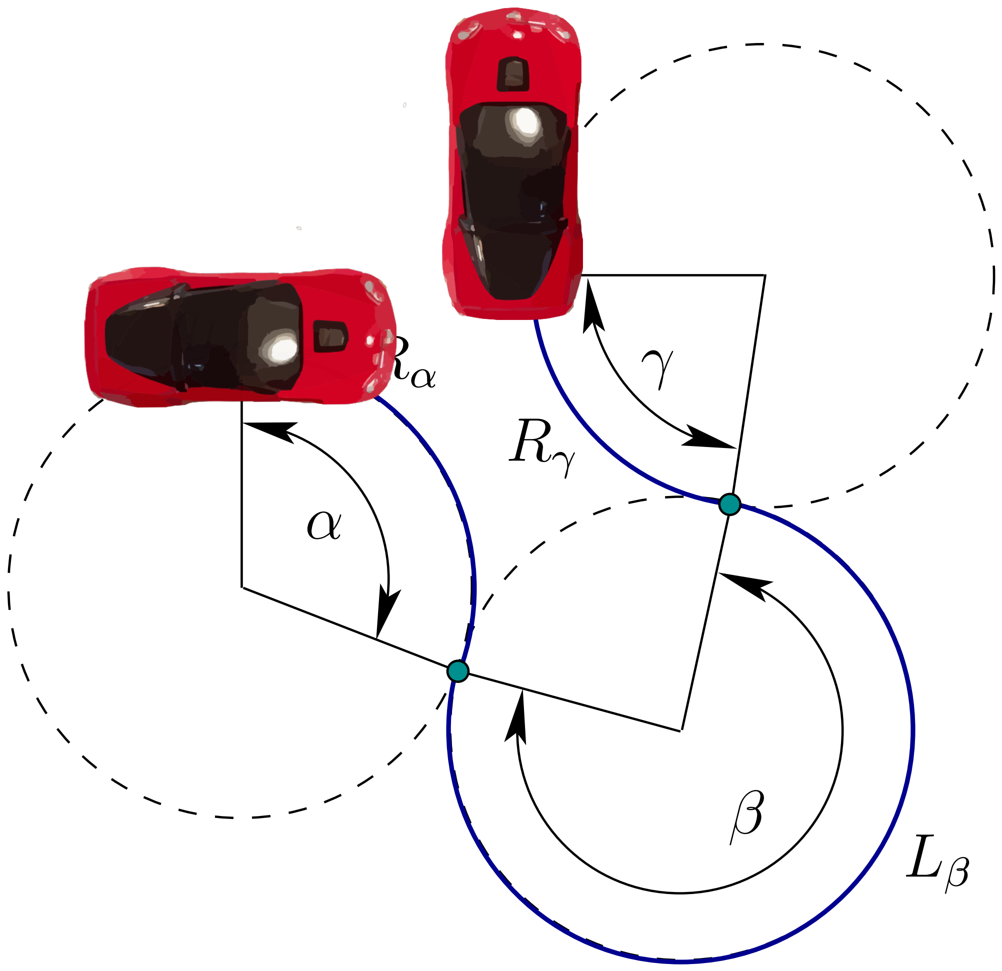
Right Straight Left

Case 1: We can analytically solve the BVP

Dubins showed that ALL solutions had to be one of 6 classes

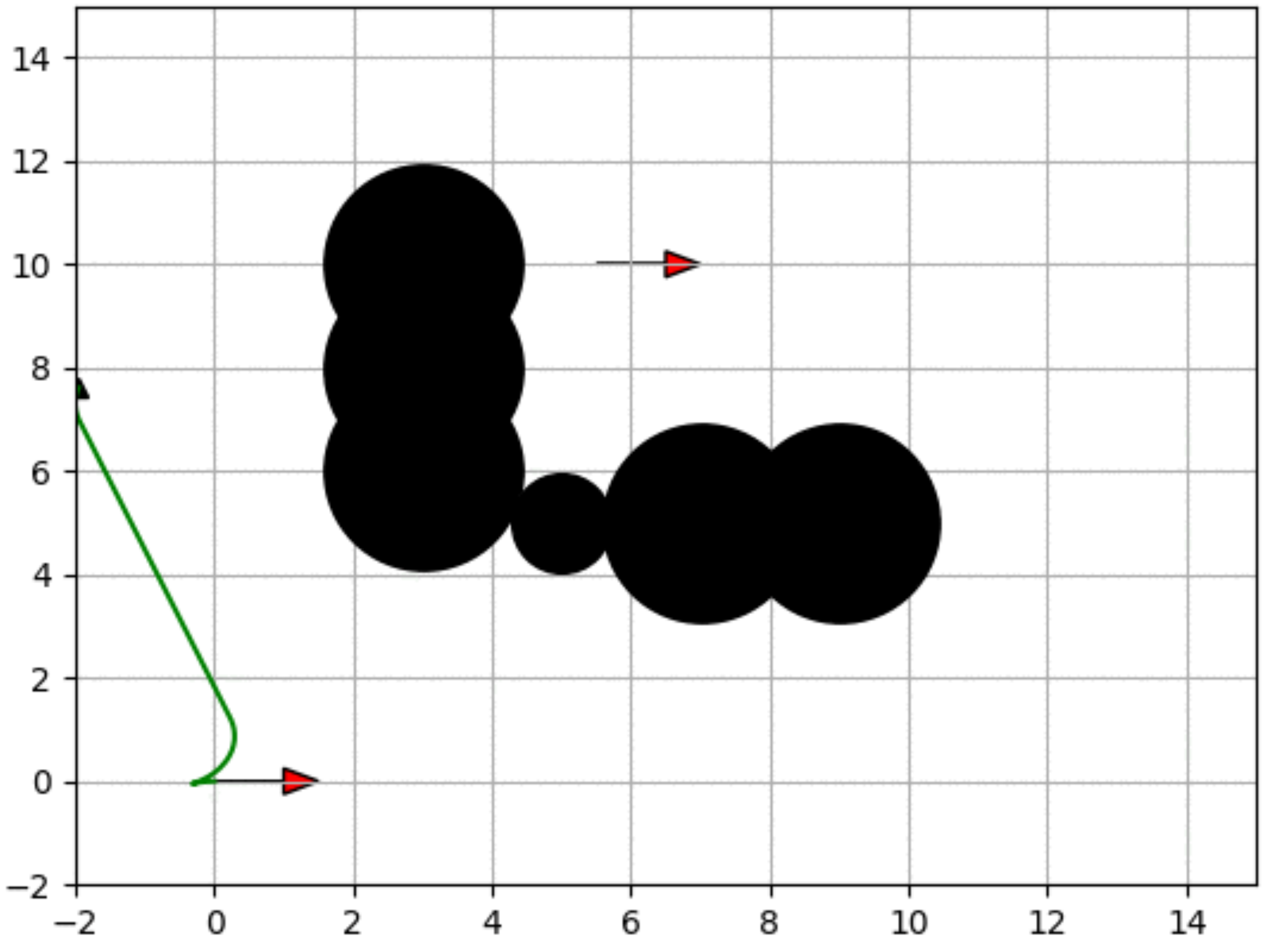
$$\{LRL,\ RLR,\ LSL,\ LSR,\ RSL,\ RSR\}.$$

Hence, given a query, evaluate ALL 6 options, and pick the shortest one!



$R_\alpha L_\beta R_\gamma$

Random sampling with Dubins steering



https://github.com/AtsushiSakai/PythonRobotics

Case 2: We need to numerically solve the BVP

What if you were changing the steering rate?

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \dot{\theta} \\ u \end{bmatrix}$$

$$q_1 = (x_1, y_1, \theta_1, \dot{\theta}_1)$$

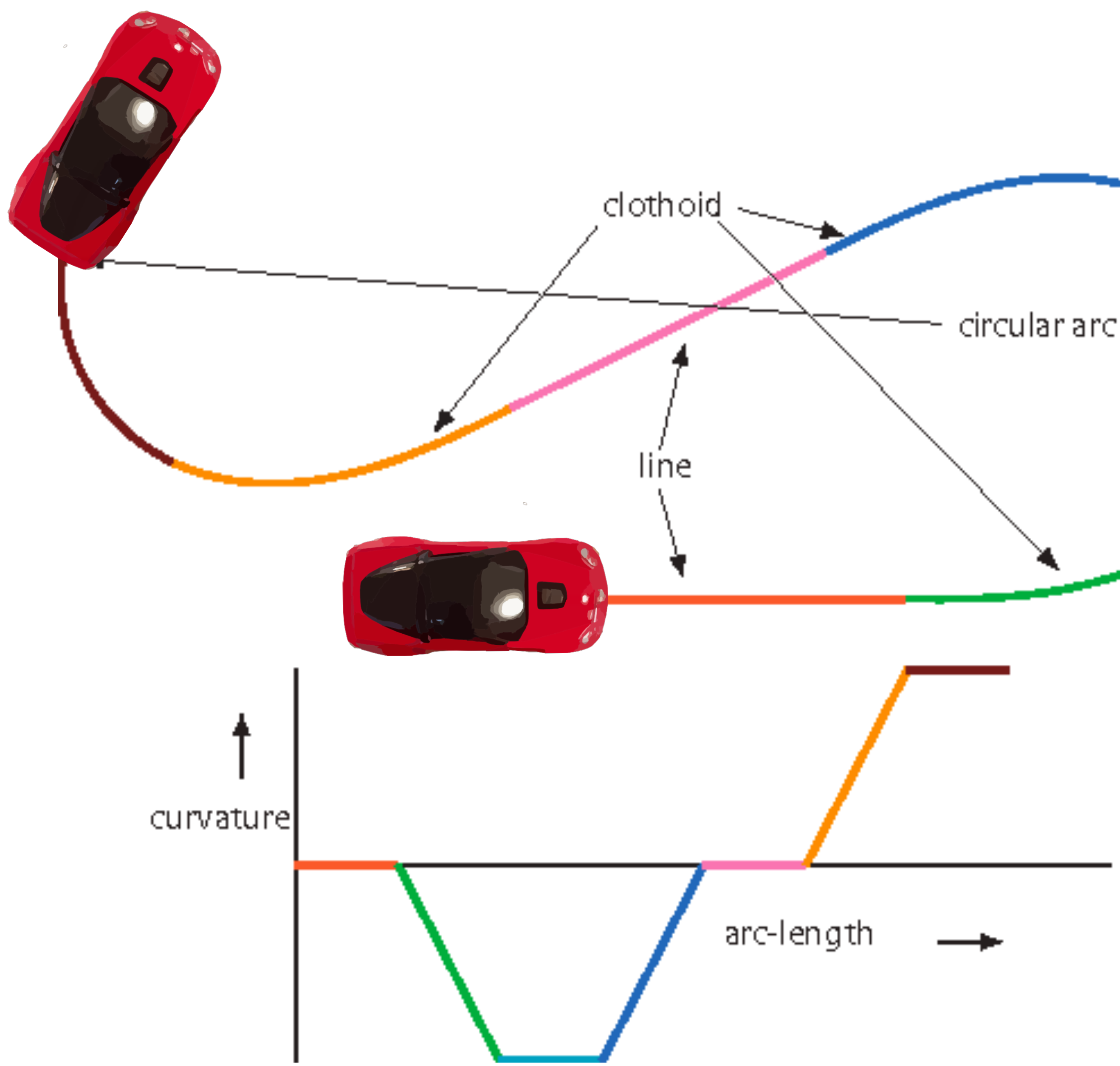
$$q_2 = (x_2, y_2, \theta_2, \dot{\theta}_2)$$

$$|\dot{\theta}| \leq C_1 \quad |u| \leq C_2$$

No longer called Dubins path, called Clothoid path

Can't solve analytically ... need numerical solutions 

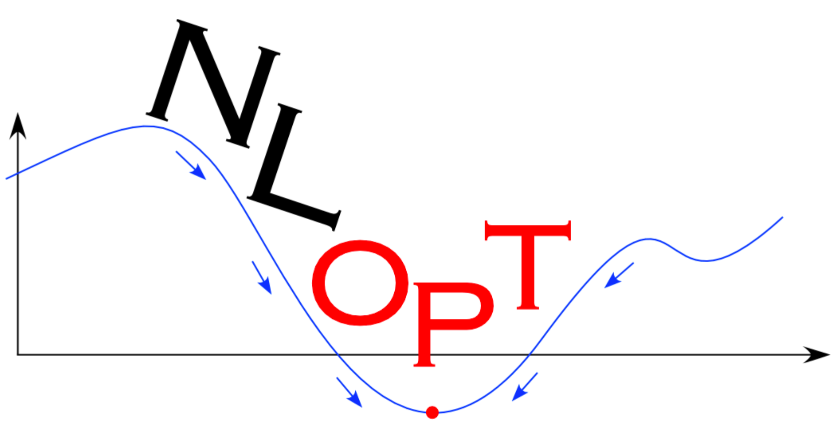
Case 2: We need to numerically solve the BVP



Create a non-linear optimization problem

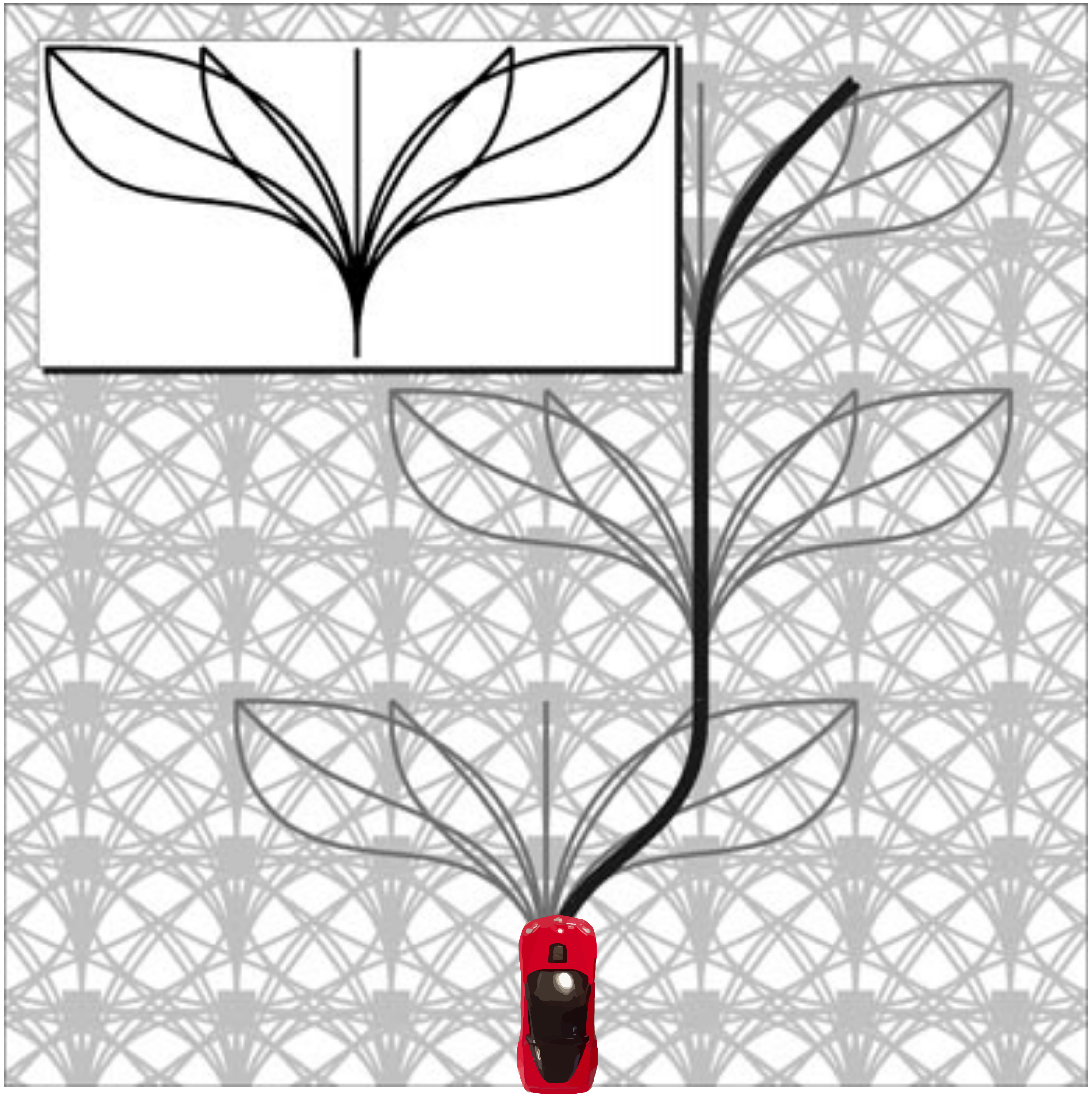
Represent the path as a curvature polynomial

Enforce end point constraints



https://nlopt.readthedocs.io/en/latest/

Example of a state lattice



Pivtoraiko et al. 2007

Case 3: We can't even solve the BVP

Typically rules out roadmap methods
because we cannot exactly connect 2 states

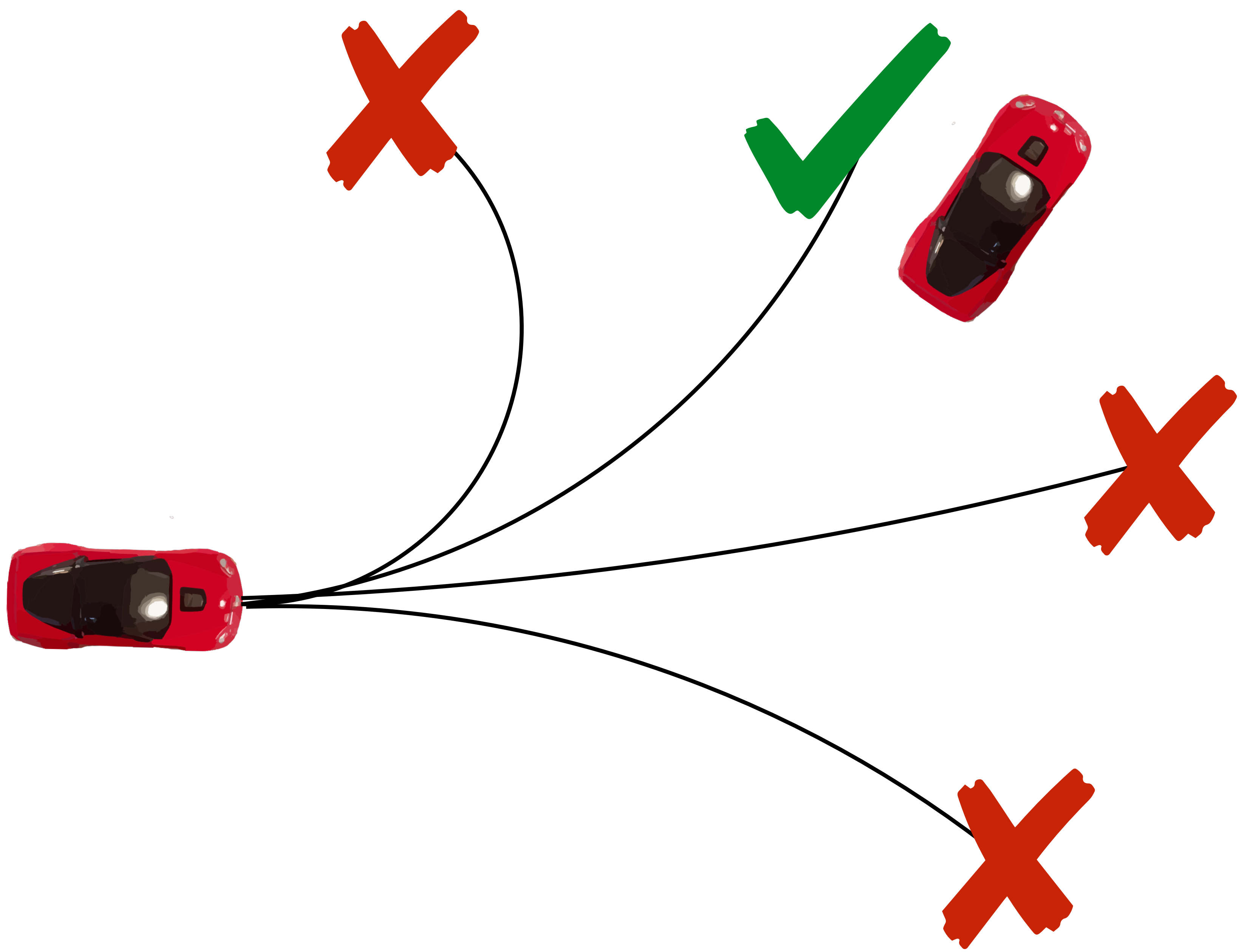
Why?

1. Dynamics too complicated - nonlinear optimization doesn't converge

2. Even if it did, too expensive to run online

Case 3: We can't even solve the BVP

If we are working with implicit graphs, we don't need to exactly connect two states.



Pretend this was the state you wanted all along!

Incremental Sampling-Based Planners

- Multiple-query methods such as PRM valuable in highly structured environments.
- Online planning does not require multiple queries
- Computing roadmap a priori computationally challenging/ infeasible.
- Incremental sampling-based planning algorithms single-query counterpart to PRMs.
- Key idea:

  Avoid necessity to set number of samples *a priori;*

  return solution as soon as set of trajectories is rich enough.

Karaman, S., & Frazzoli, E. (2011), “Sampling-based algorithms for optimal motion planning”, The International Journal of Robotics Research, 30(7), 846–894. https://doi.org/10.1177/0278364911406761

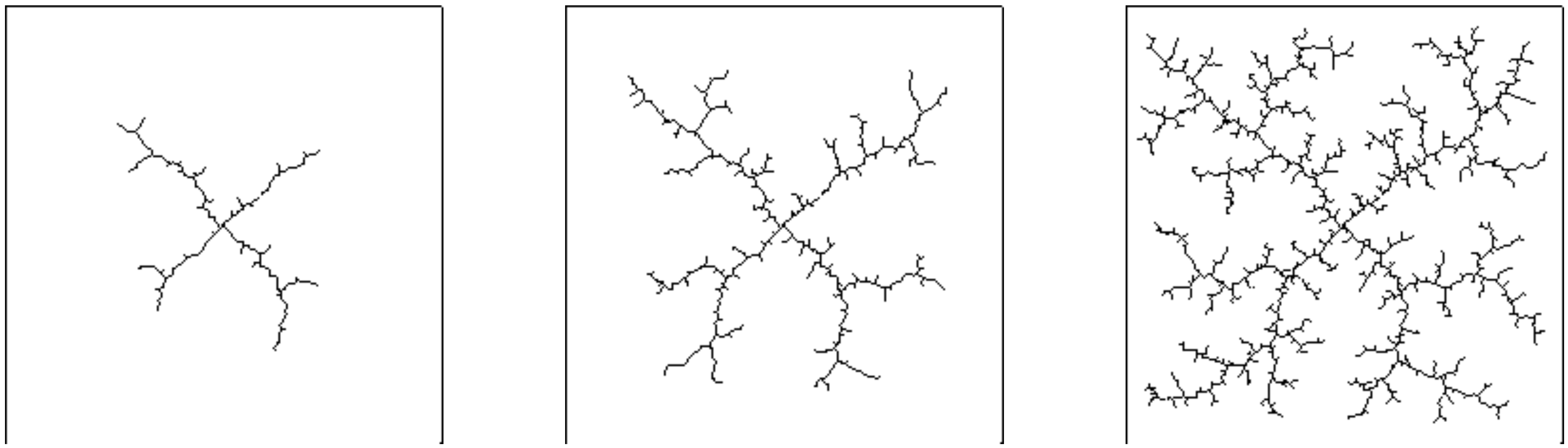
Rapidly Exploring Random Trees (RRTs)

http://msl.cs.uiuc.edu/rrt

Rapidly-exploring random trees: A new tool for path planning. S. M. LaValle. TR 98-11, Computer Science Dept., Iowa State University, October 1998

RRTs

- Build a tree by iteratively connecting *next* states via the execution of random controls
- Carefully sample controls to ensure good coverage



The RRT Algorithm

GENERATE_RRT($x_{init}, K, \Delta t$)

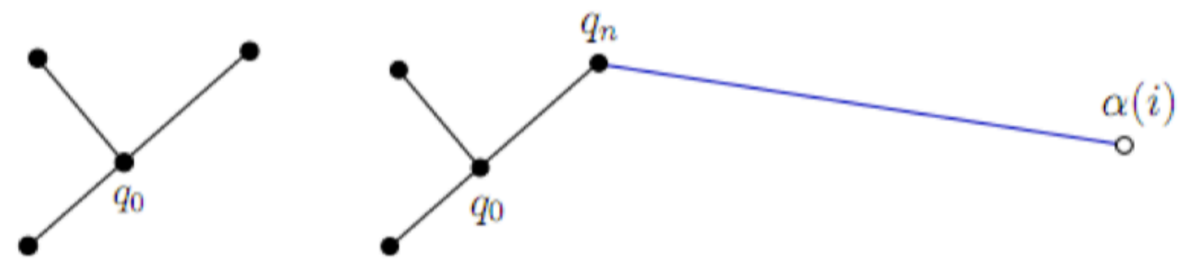
$\mathcal{T}$.init(x_{init});
for $k = 1$ **to** K **do**
    $x_{rand} \leftarrow$ RANDOM_STATE();
    $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
    $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
    $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
    $\mathcal{T}$.add_vertex(x_{new});
    $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
Return $\mathcal{T}$

Probabilistically Complete

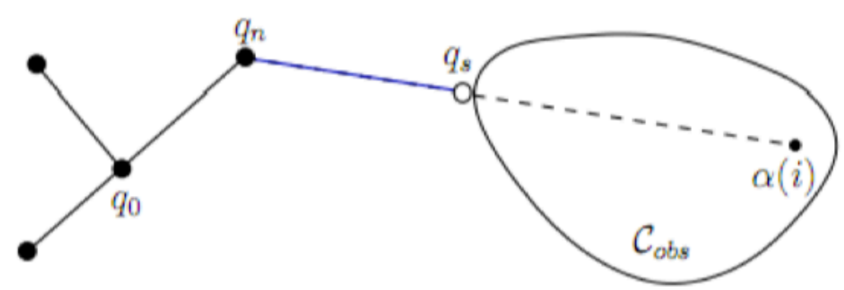
Exponential rate of decay for the probability of failure

Extend Function

- No obstacles, holonomic:



- With obstacles, holonomic:

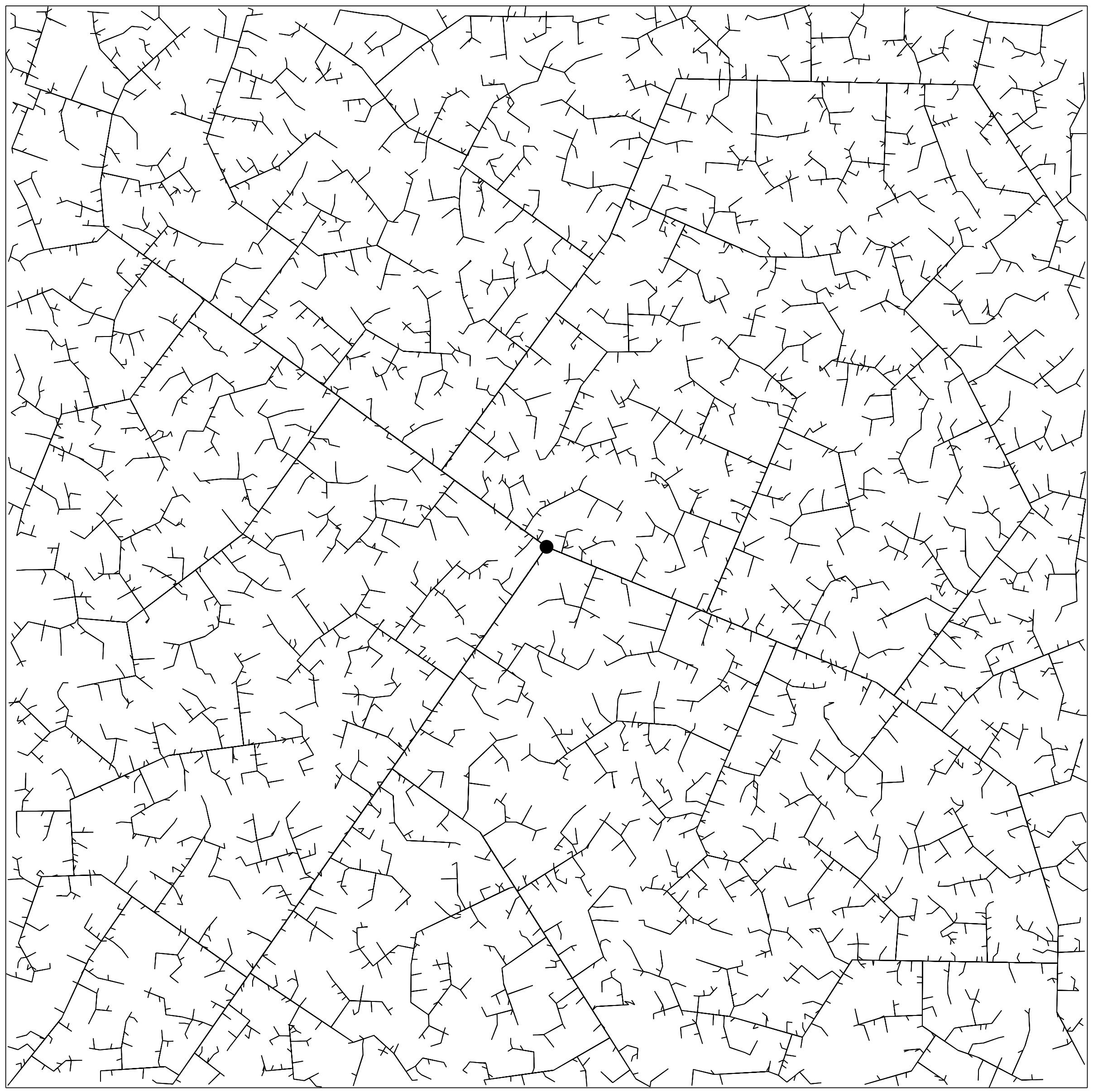


- Non-holonomic: approximately (sometimes as approximate as picking best of a few random control sequences) solve two-point boundary value problem

Slide credit: Pieter Abbeel

RRT Expansion

45 iterations

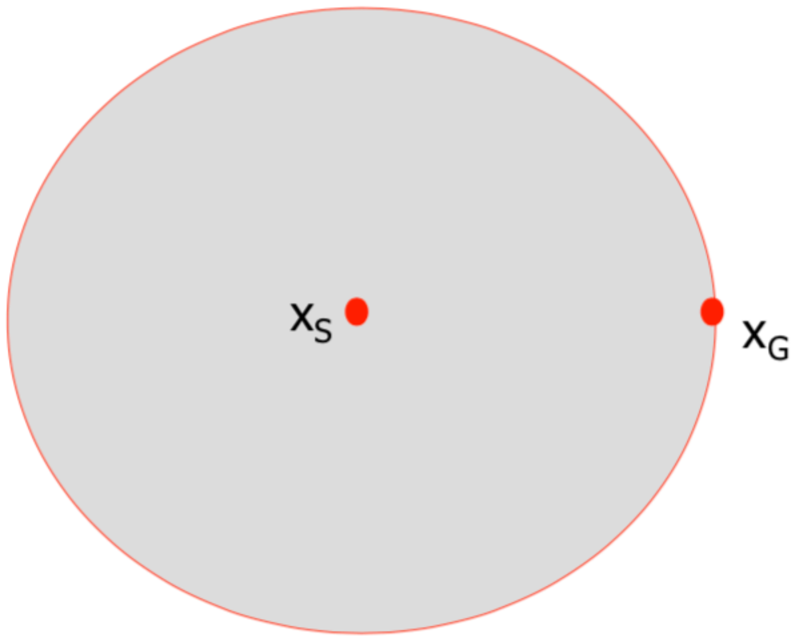


2345 iterations

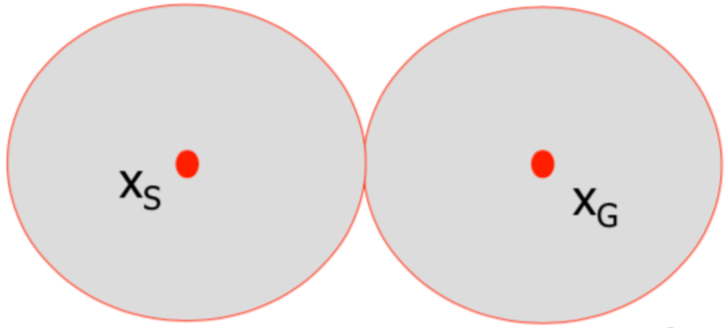
Notable variation: Bidirectional RRT

Key idea:
Grow 2 trees, 1 starting from start, 1 from goal, connect them, done.

- Volume swept out by unidirectional RRT:



- Volume swept out by bi-directional RRT:



- Difference becomes even more pronounced in higher dimensions

J. J. Kuffner and S. M. LaValle, "An efficient approach to path planning using balanced bidirectional RRT search", Technical Report CMU-RI-TR-05-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2005.

Slide credit: Pieter Abbeel

RRT*

- Asymptotically optimal
- Probabilistically complete
- Computationally efficient

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
        $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$
      then $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

Karaman, S., & Frazzoli, E. (2011), "Sampling-based algorithms for optimal motion planning", The International Journal of Robotics Research, 30(7), 846–894. https://doi.org/10.1177/0278364911406761

Slide credit: Pieter Abbeel

Smoothing

Paths extracted from sampling-based motion planners tend to be jerky.

Remedies:

1. Shortcutting

   Along the found path, pick two vertices x_{t1}, x_{t2} and try to connect them directly (skipping over all intermediate vertices)

2. Nonlinear optimization for optimal control

   Allows to specify an objective function that includes smoothness in state, control, small control inputs, etc.

Slide credit: Pieter Abbeel