Content adapted from LaValle

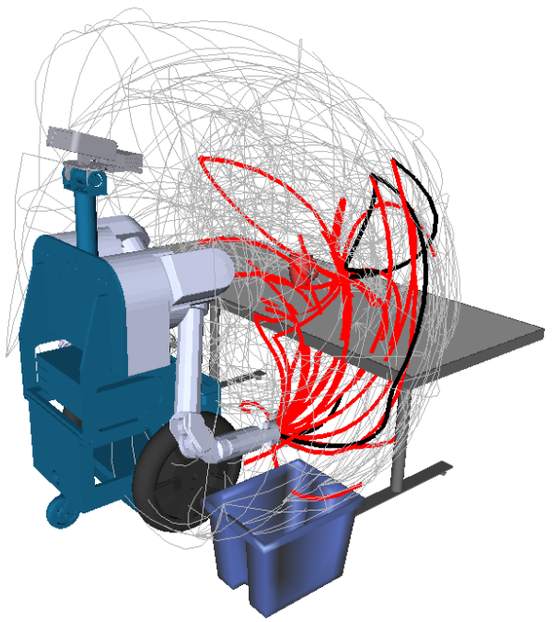
Planning on Roadmaps (contd)

Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

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

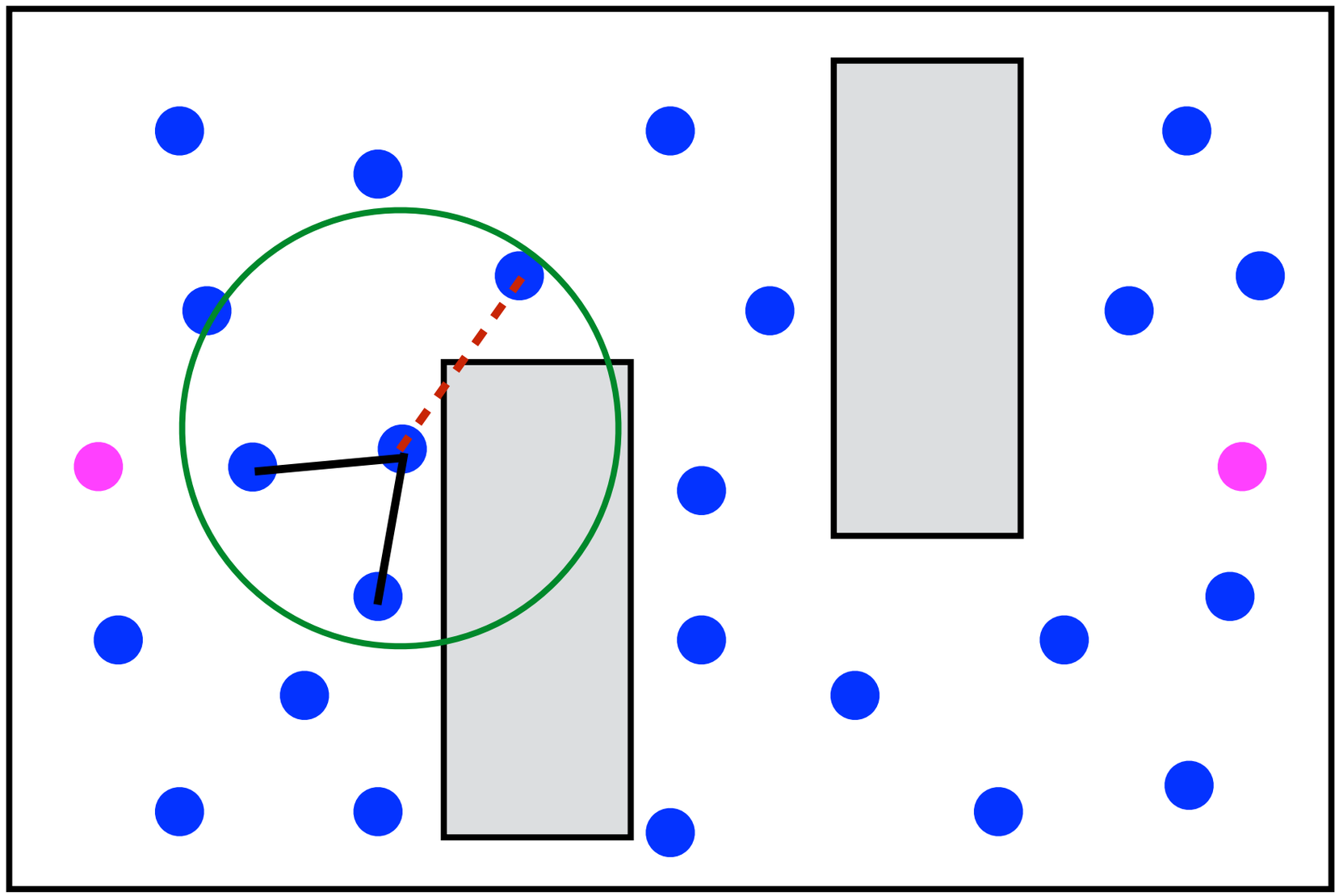
3. Search graph to find a solution
   - can reuse graph across multiple queries

Theoretical Guarantees: It depends ...

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?

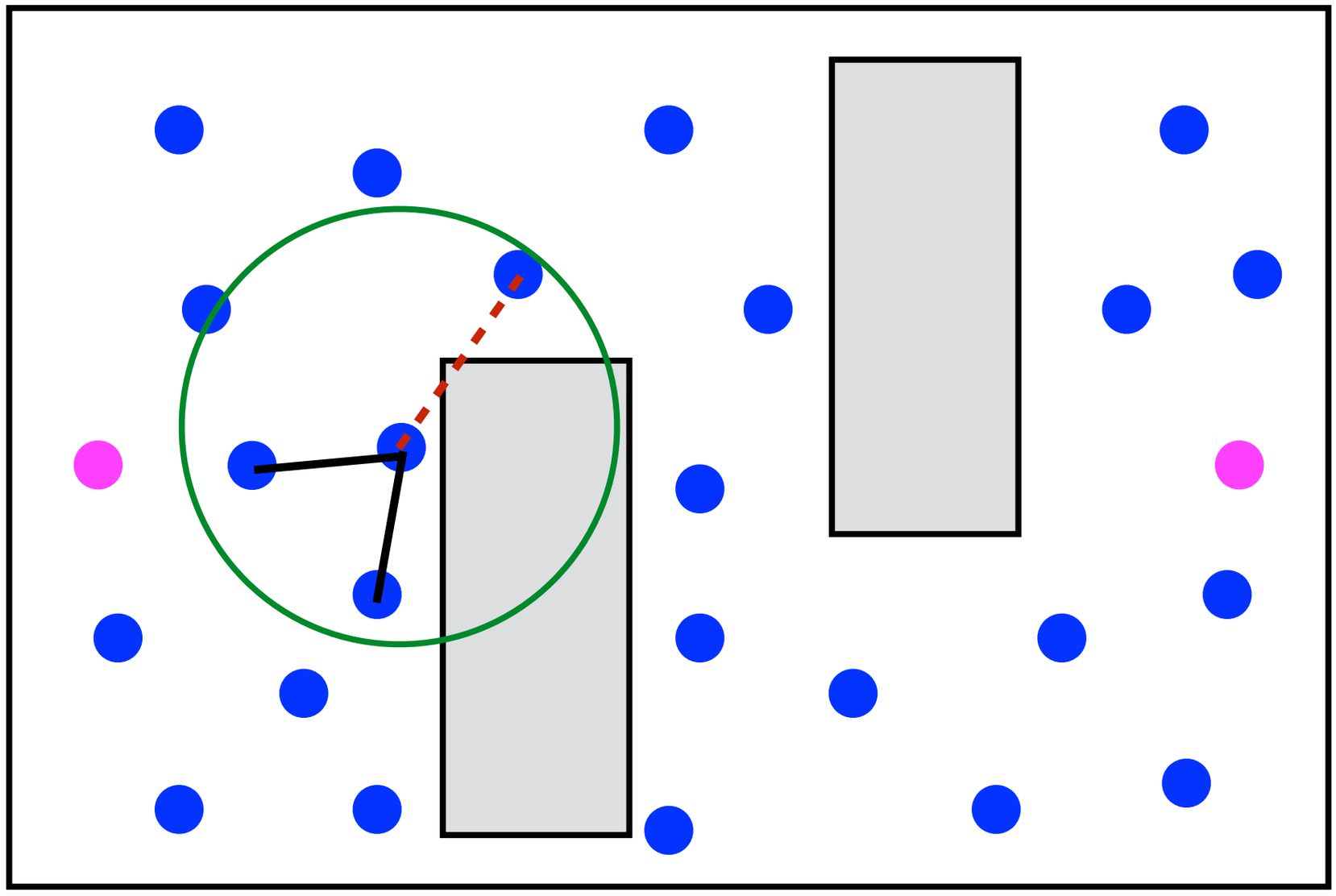


What is the optimal radius?

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

where magic constant!

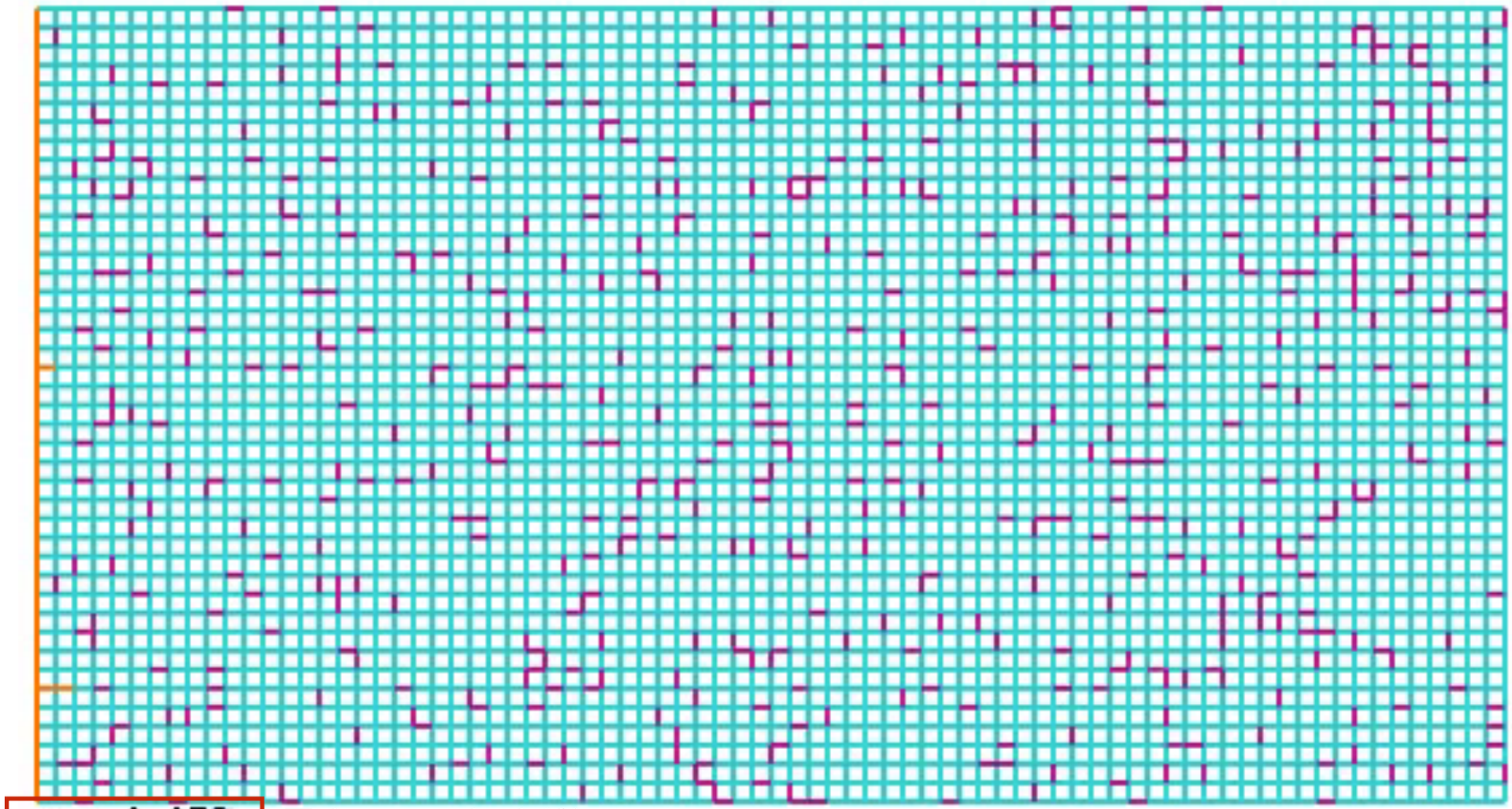
$$\gamma \geq 2(1+1/d)^{1/d} \frac{\mu(\mathcal{C}_{free})}{\zeta_d}$$



Also known as a Random Geometric Graph (RGG)

Aside: Percolation theory

http://www.univ-orleans.fr/mapmo/membres/berglund/ressim.html

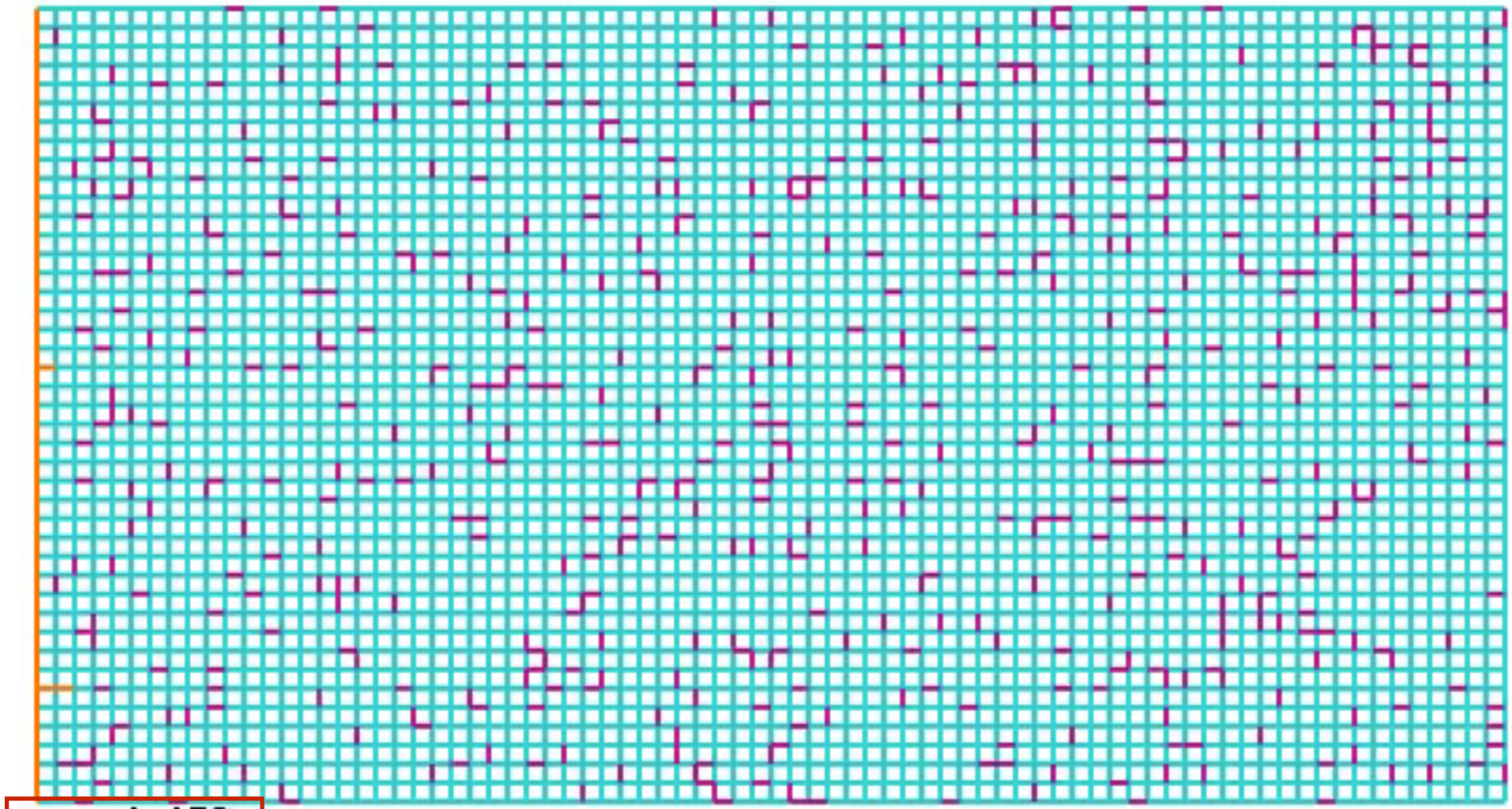


Other uses of percolation theory in planning

Theoretical Limits of Speed and Resolution for Kinodynamic Planning in a Poisson Forest
Sanjiban Choudhury, Sebastian Scherer and J. Andrew (Drew) Bagnell

Aside: Percolation theory

http://www.univ-orleans.fr/mapmo/membres/berglund/ressim.html



Other uses of percolation theory in planning

Theoretical Limits of Speed and Resolution for Kinodynamic Planning in a Poisson Forest
Sanjiban Choudhury, Sebastian Scherer and J. Andrew (Drew) Bagnell

This is the PRM* Algorithm!

1. Sample vertices randomly

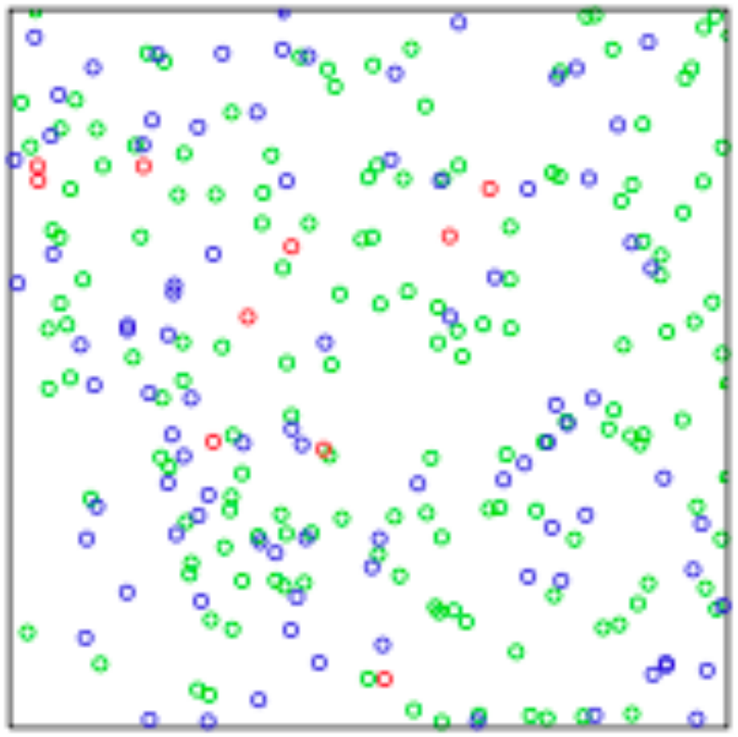
2. Use optimal radius formula to connect vertices

3. Search graph to find a solution

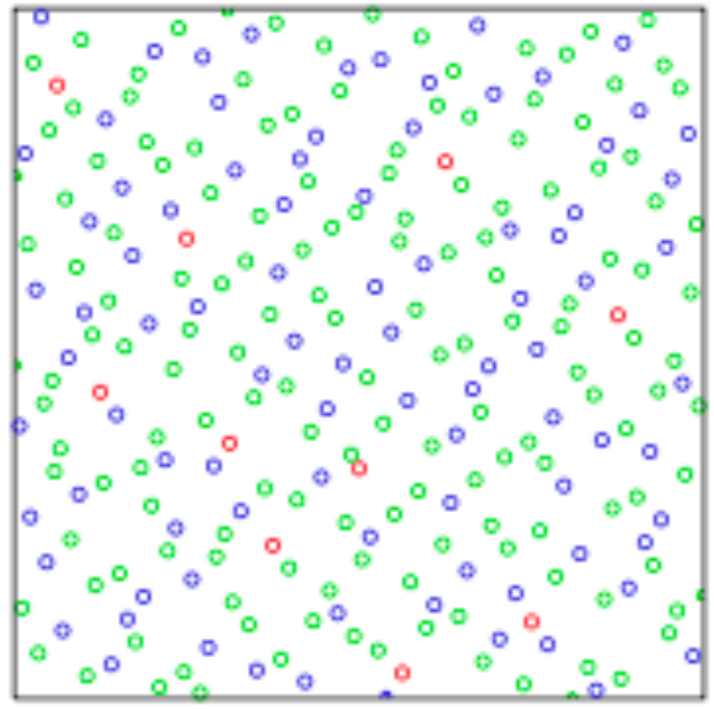
Theorem: Probabilistically complete AND Asymptotically optimal

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

Can we do better than random?



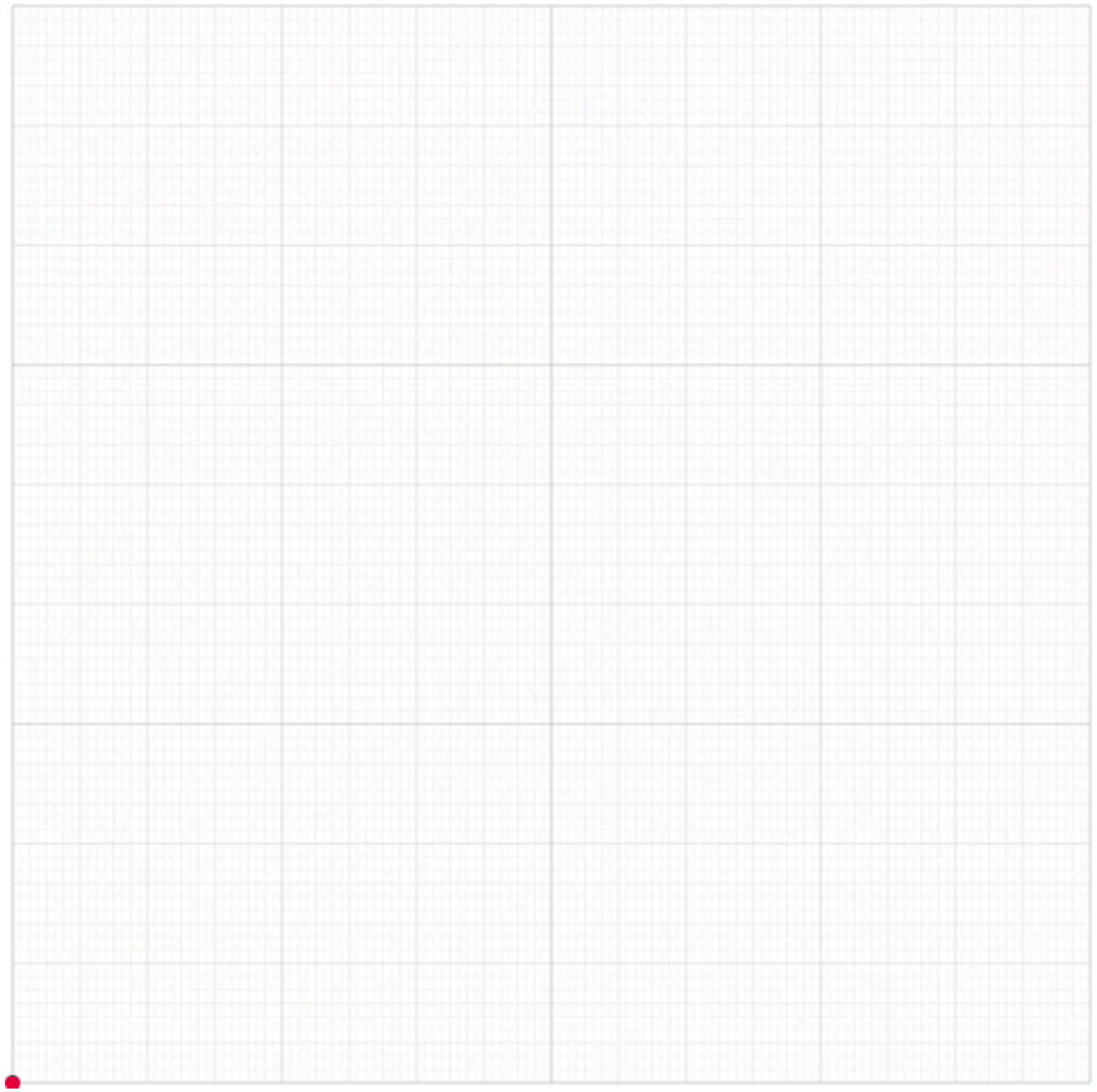
Uniform random sampling tends to clump



Ideally we would want points to be spread out evenly

Question: How do we do this without discretization?

Halton Sequence

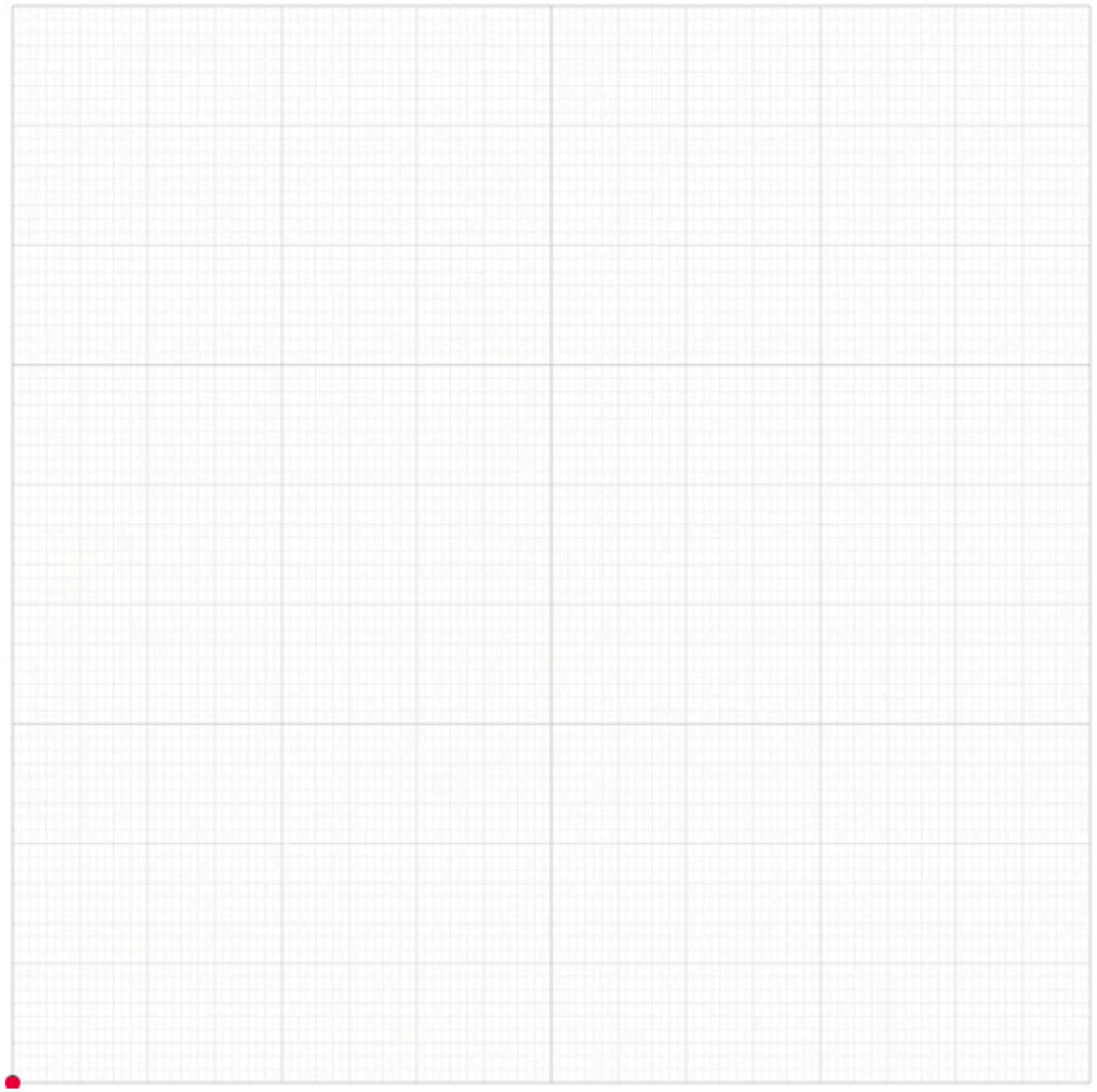


Generalization of
Van de Coruput Sequence

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

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

Halton Sequence

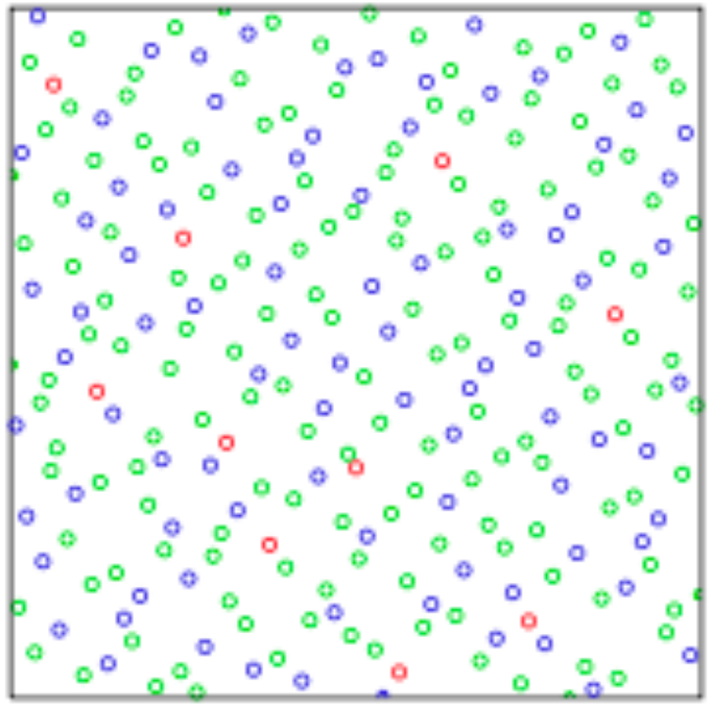


Generalization of
Van de Coruput Sequence

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?

What makes a good graph?

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

What makes a good graph?

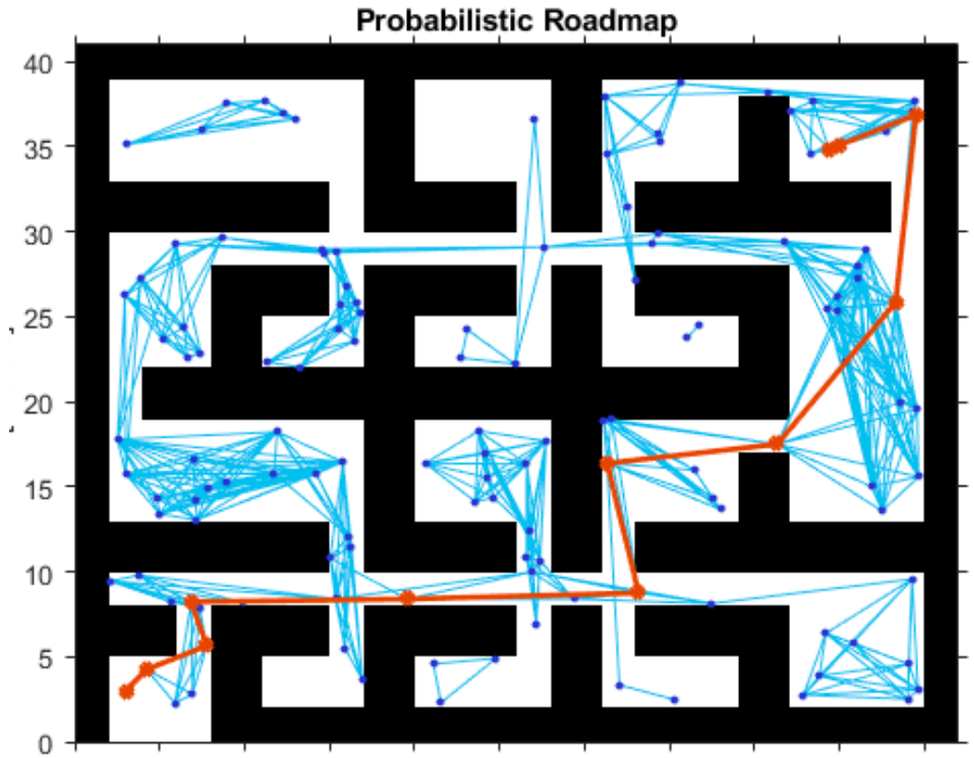
1. A good graph must be sparse (both in vertices and edges)

2. A good graph must have good coverage

What makes a good graph?

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

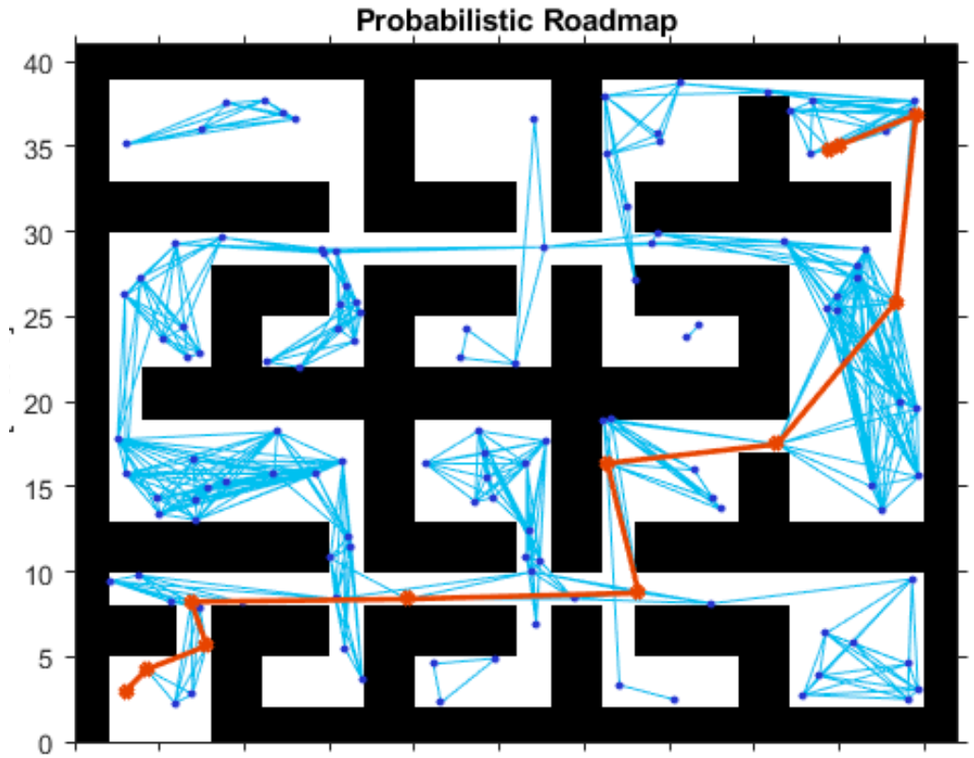
2. A good graph must have good coverage



What makes a good graph?

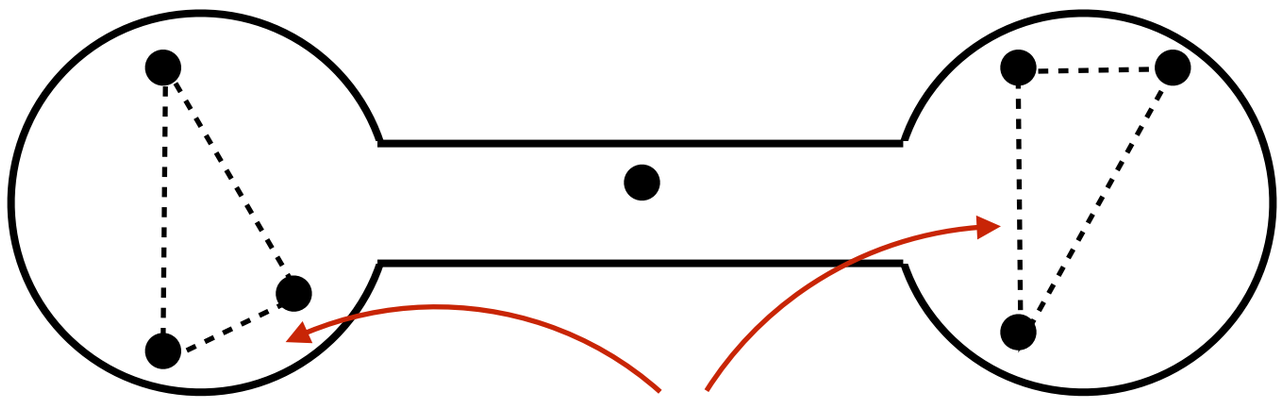
1. A good graph must be sparse (both in vertices and edges)

2. A good graph must have good coverage



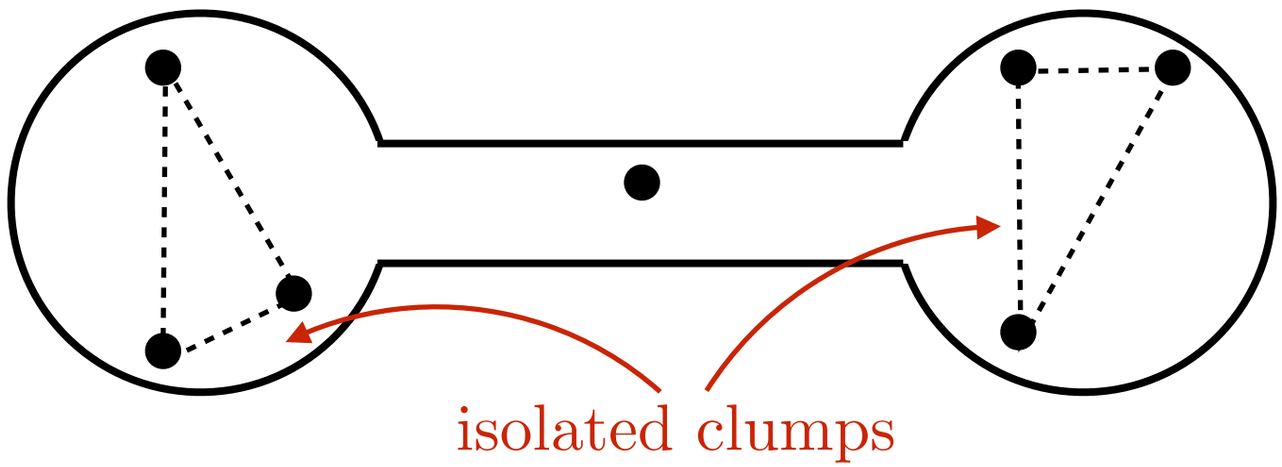
3. A good graph must have the same connectivity of free space

The Narrow Passage: Planning's boogie man!



Why is narrow passage mathematically hard to plan in?

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:

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

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

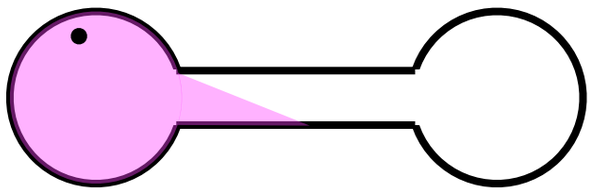
How many samples do we need?

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:

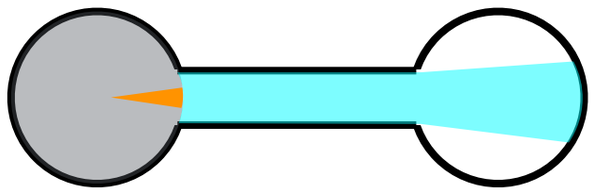
$$n \geq \left\lceil 8 \frac{\log(\frac{8}{\epsilon\alpha\gamma})}{\epsilon\alpha} + \frac{3}{\beta} + 2 \right\rceil$$

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)

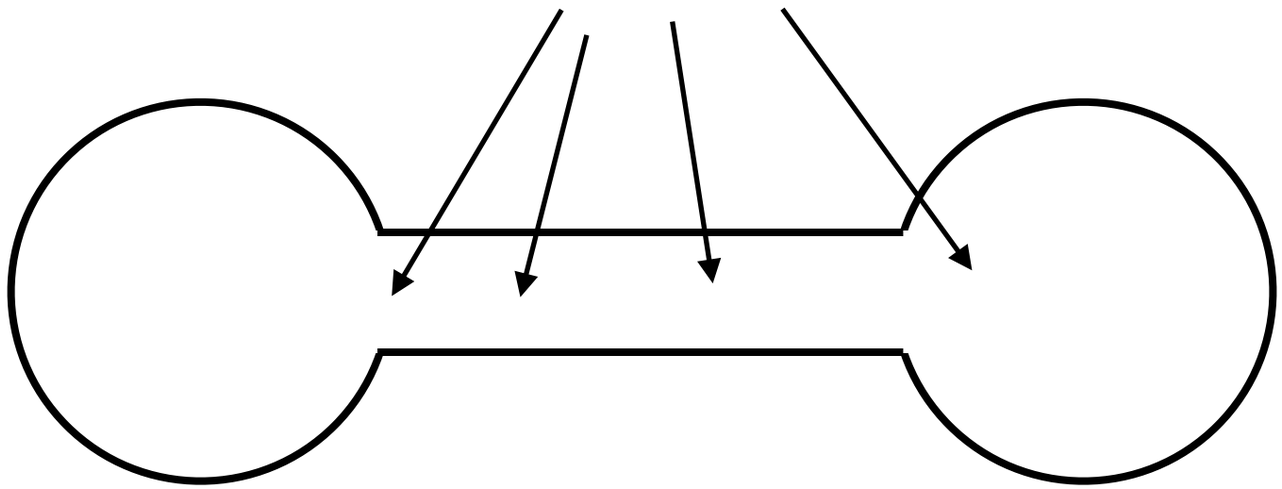


Narrow passage has small values of α, β, ϵ

Hence, needs more samples to find a path

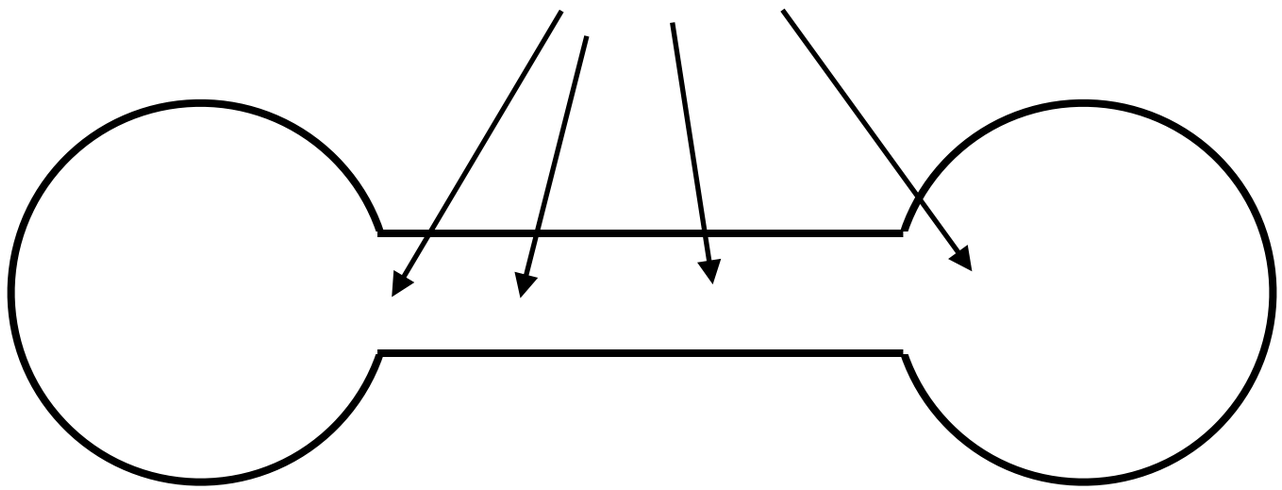
How do we bias sampling?

We somehow need more samples here



How do we bias sampling?

We somehow need more samples here

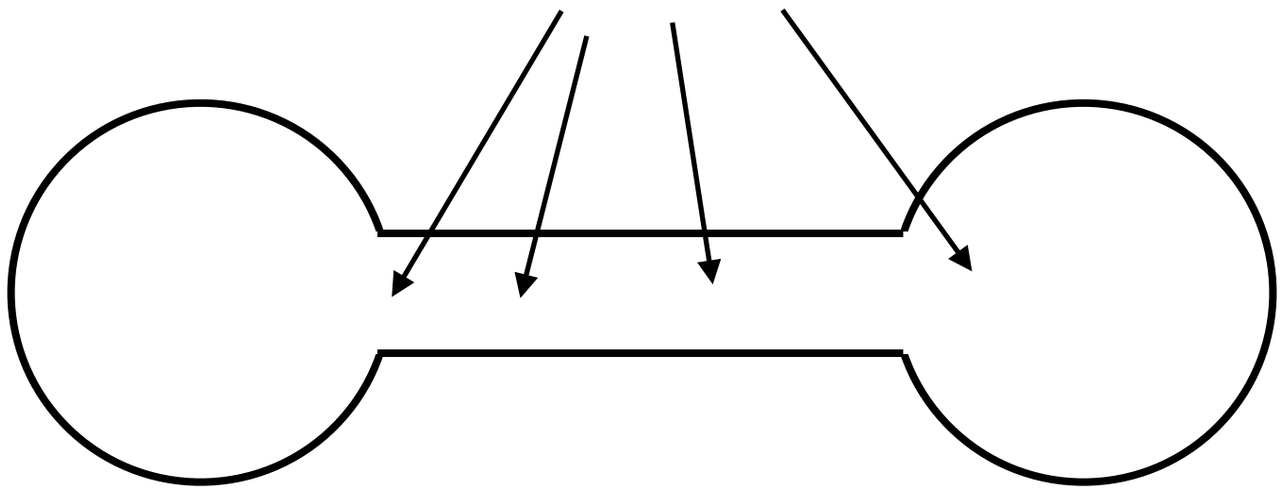


1. Sample near obstacle surface?

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

How do we bias sampling?

We somehow need more samples here



1. Sample near obstacle surface?

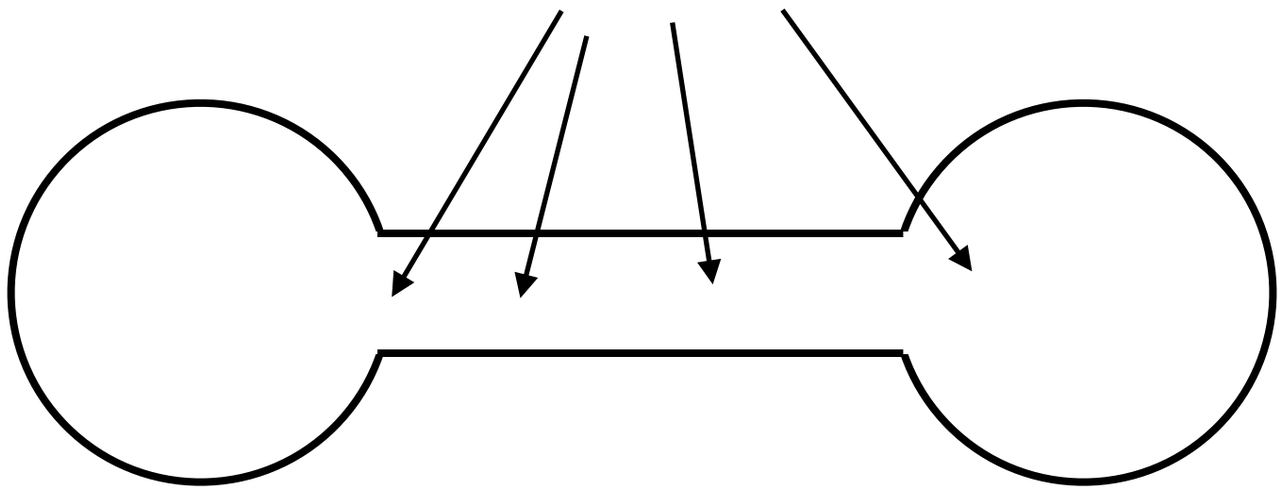
V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. 1999

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.

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

So far we have looked at an

Explicit Evaluated Graph

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

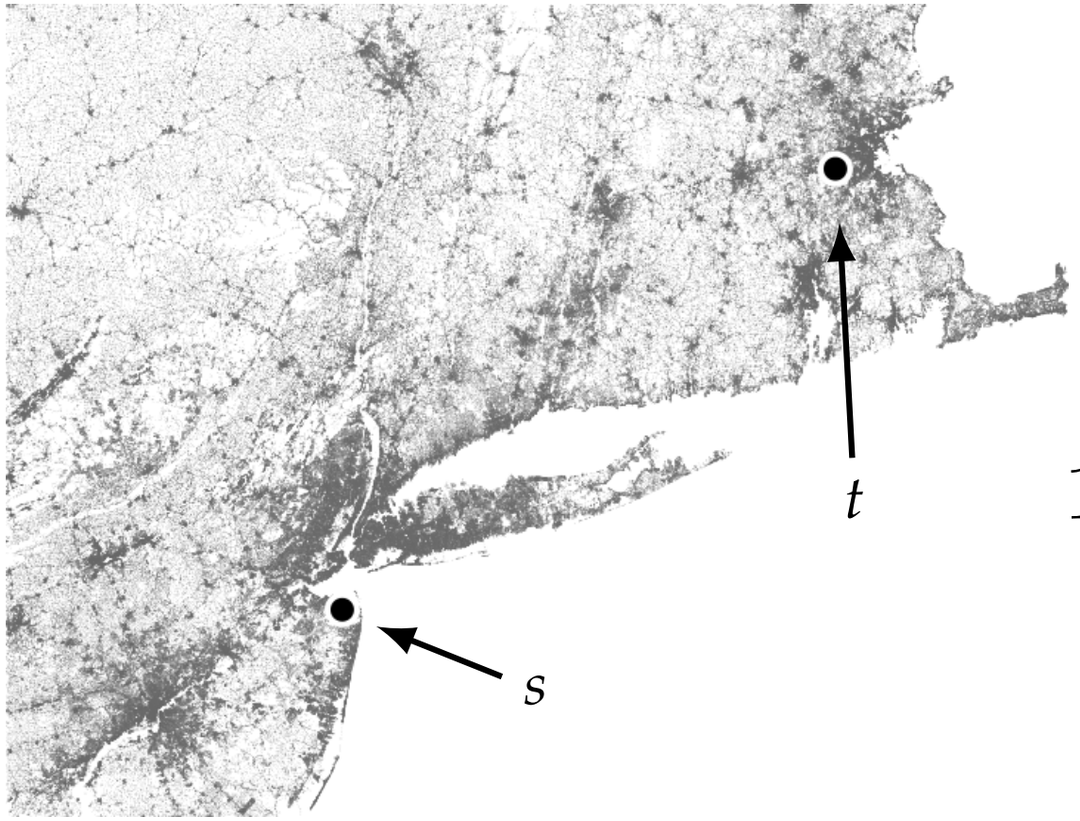
So far we have looked at an

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

Implicit graph

1. Initialize with at least one seed vertex

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

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

Searching with implicit graph

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

Searching with implicit graph

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

2. Whenever search chooses to "expand a vertex"

Searching with implicit graph

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

2. Whenever search chooses to "expand a vertex"

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

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

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

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

Planning with differential constraints

Differential constraints

So far we assumed only kinematic constraints

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

Differential constraints

So far we assumed only kinematic constraints

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

When is this assumption true?

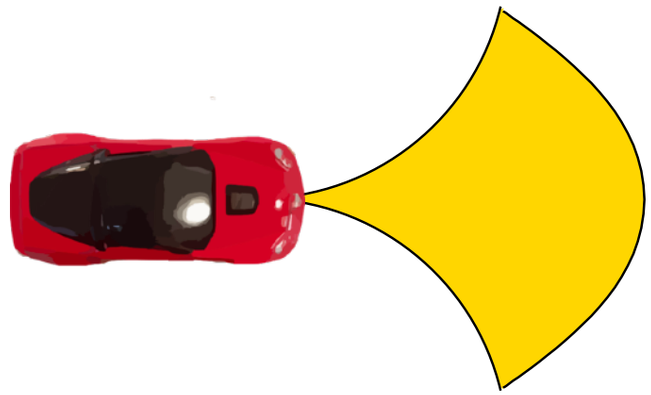
- when controller can track any path

- 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

Differential constraints make things even harder

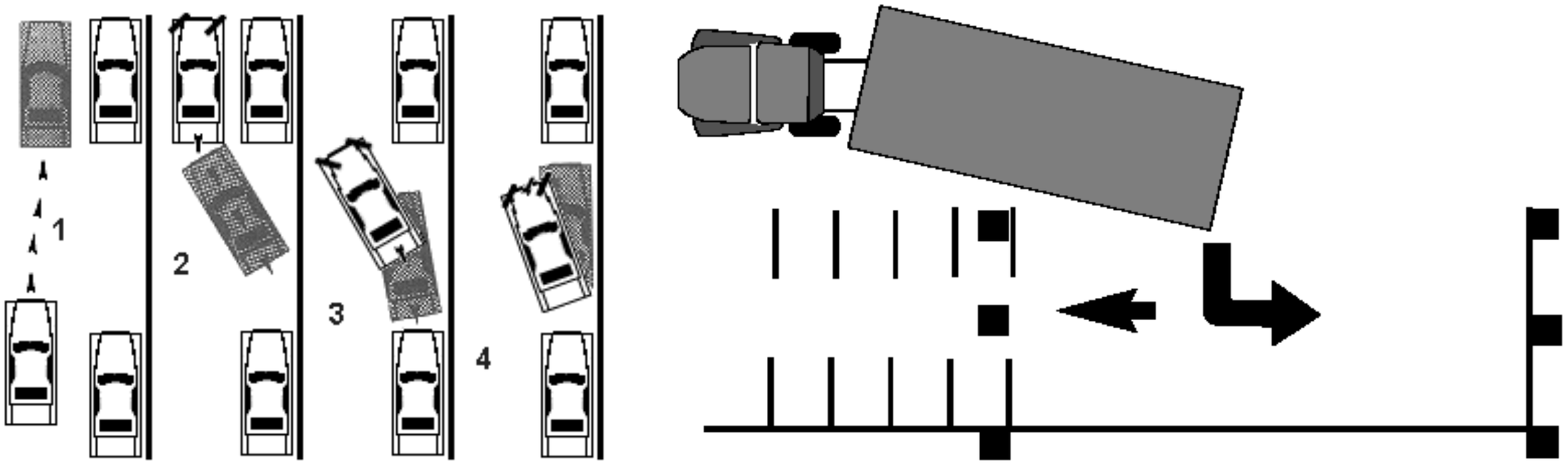
Holonomic constraints: Constraints that can be integrated, i.e.

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

Differential constraints make things even harder

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

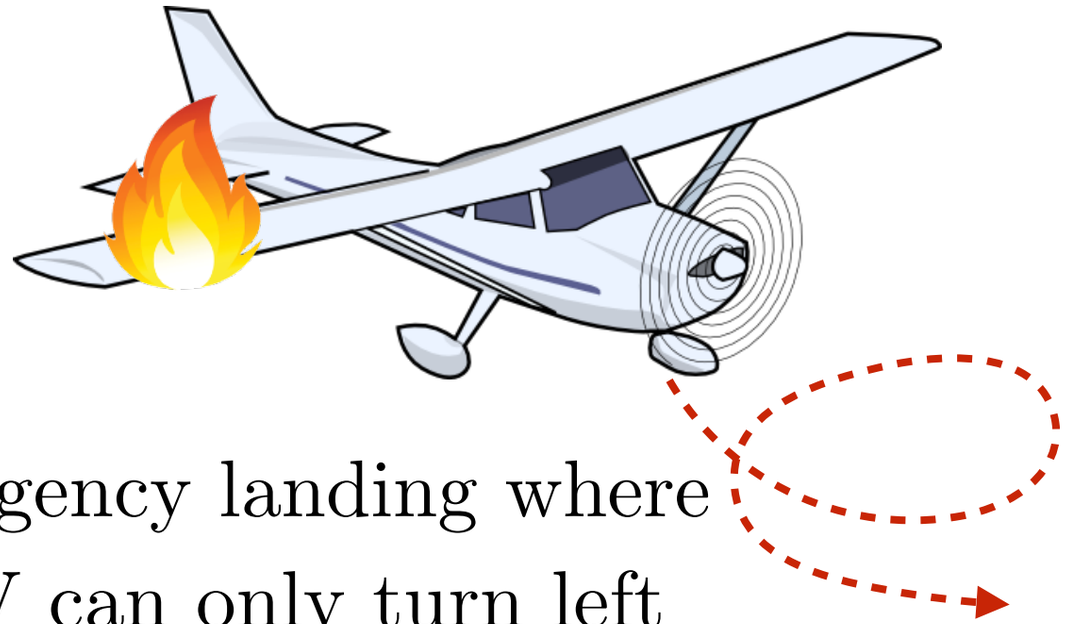
$$\dot{q} = f(q, u) \longrightarrow g(q, u) = 0$$



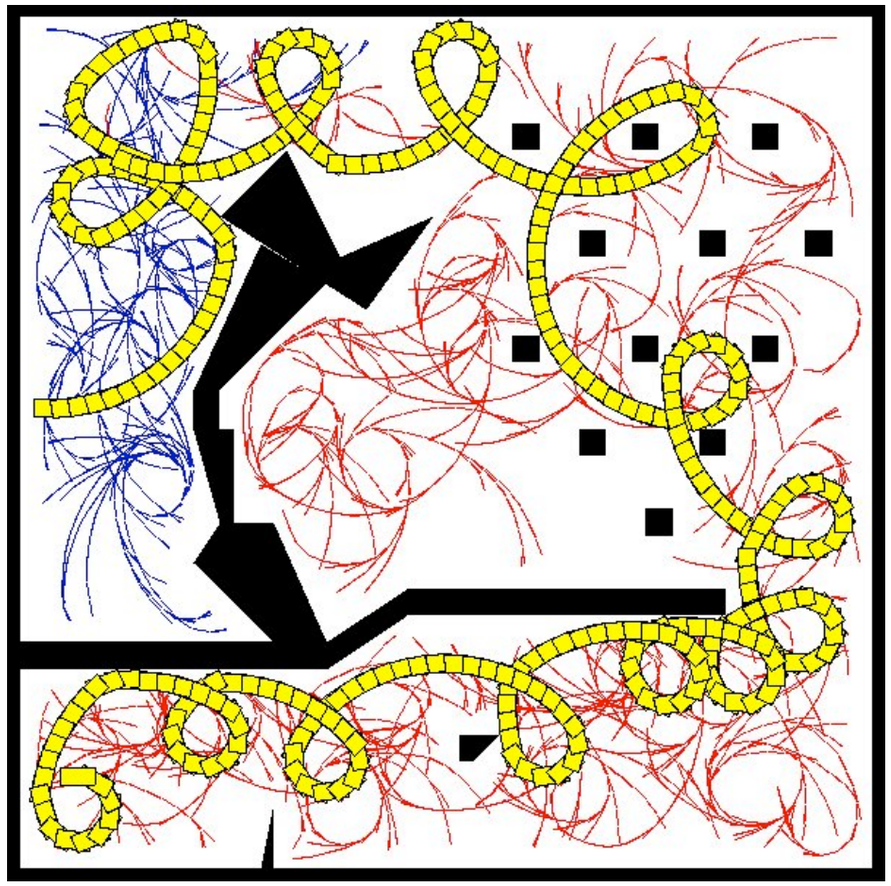
Non-holonomic constraints: Constraints that can't be integrated, i.e.

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

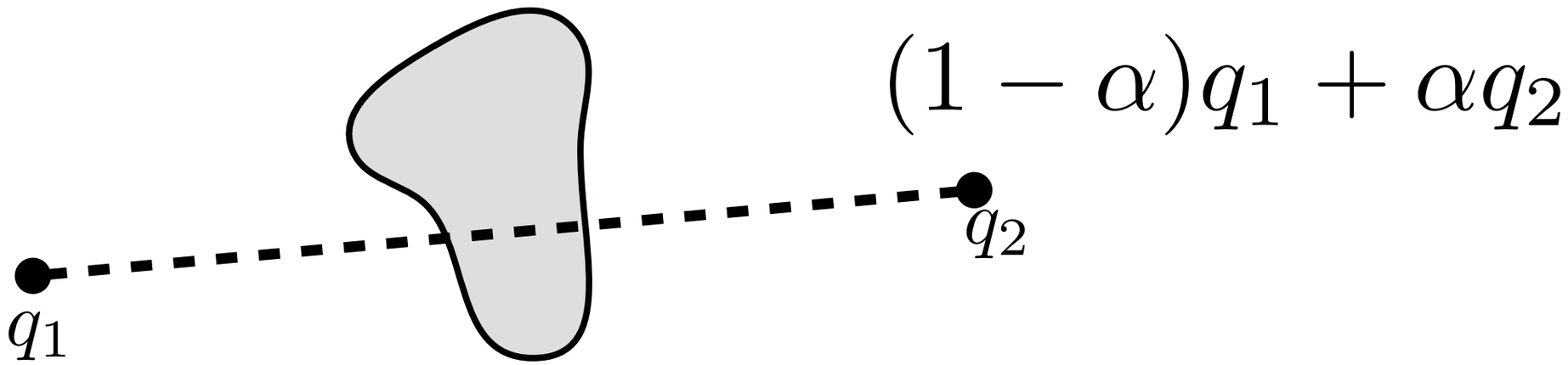
How do we incorporate differential constraints in our framework?

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

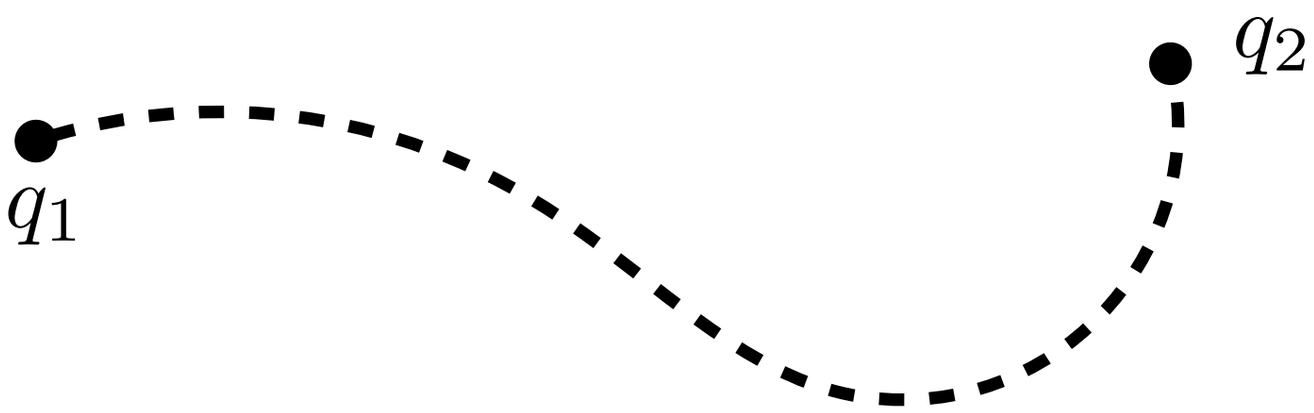


Example: Connect them with a straight line and check for feasibility

STEER function incorporate dynamics!

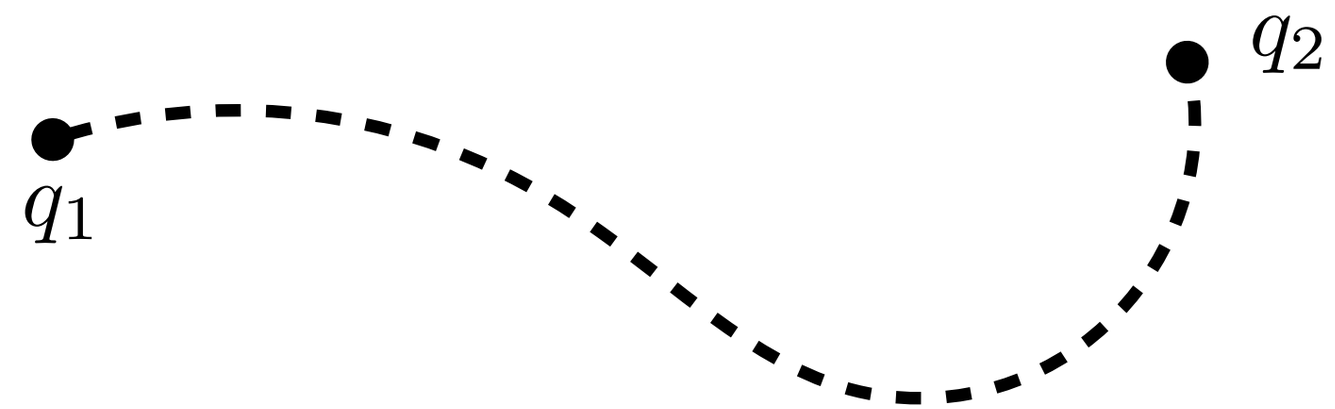
$$\texttt{steer}(q_1, q_2)$$

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

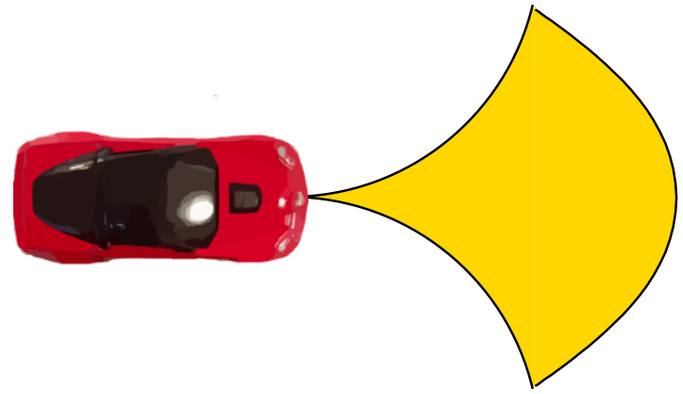
Case 1: We can analytically solve the BVP :)

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



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

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

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

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

Can we solve this analytically?

Case 1: We can analytically solve the BVP

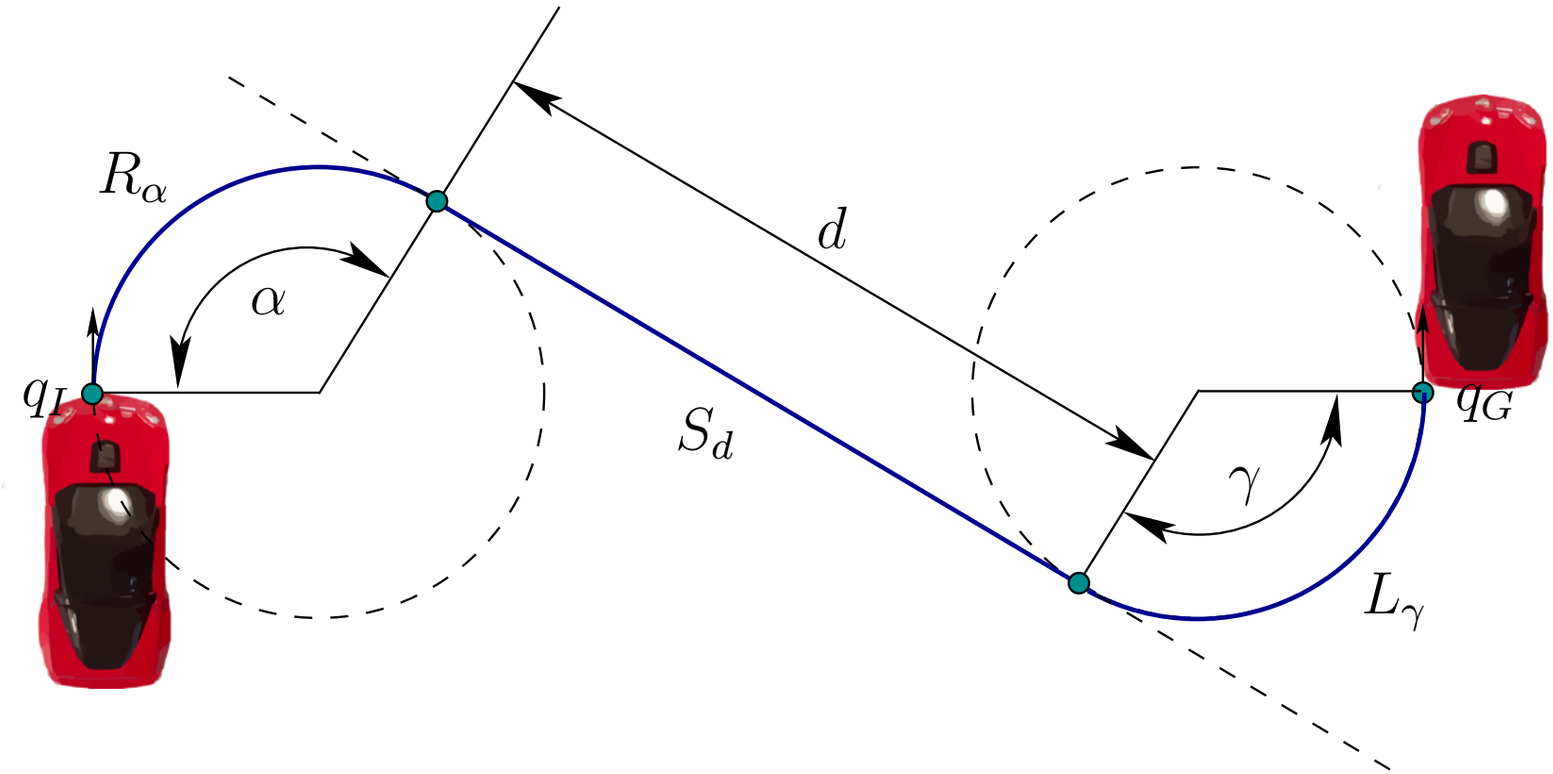
Yes! The solution is called the Dubins path 

Case 1: We can analytically solve the BVP

Yes! The solution is called the Dubins path

Case 1: We can analytically solve the BVP

Yes! The solution is called the Dubins path 

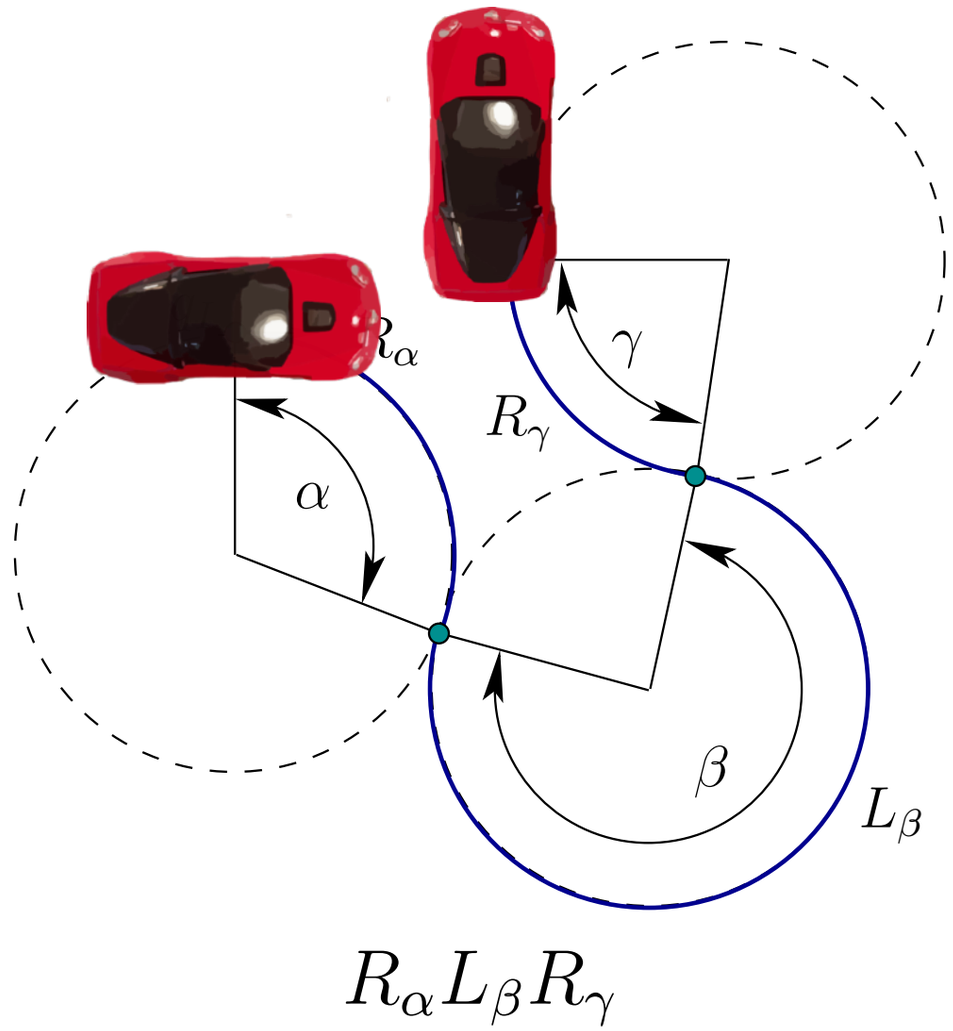


$$R_\alpha S_d L_\gamma$$

Right Straight Left

Case 1: We can analytically solve the BVP

Case 1: We can analytically solve the BVP

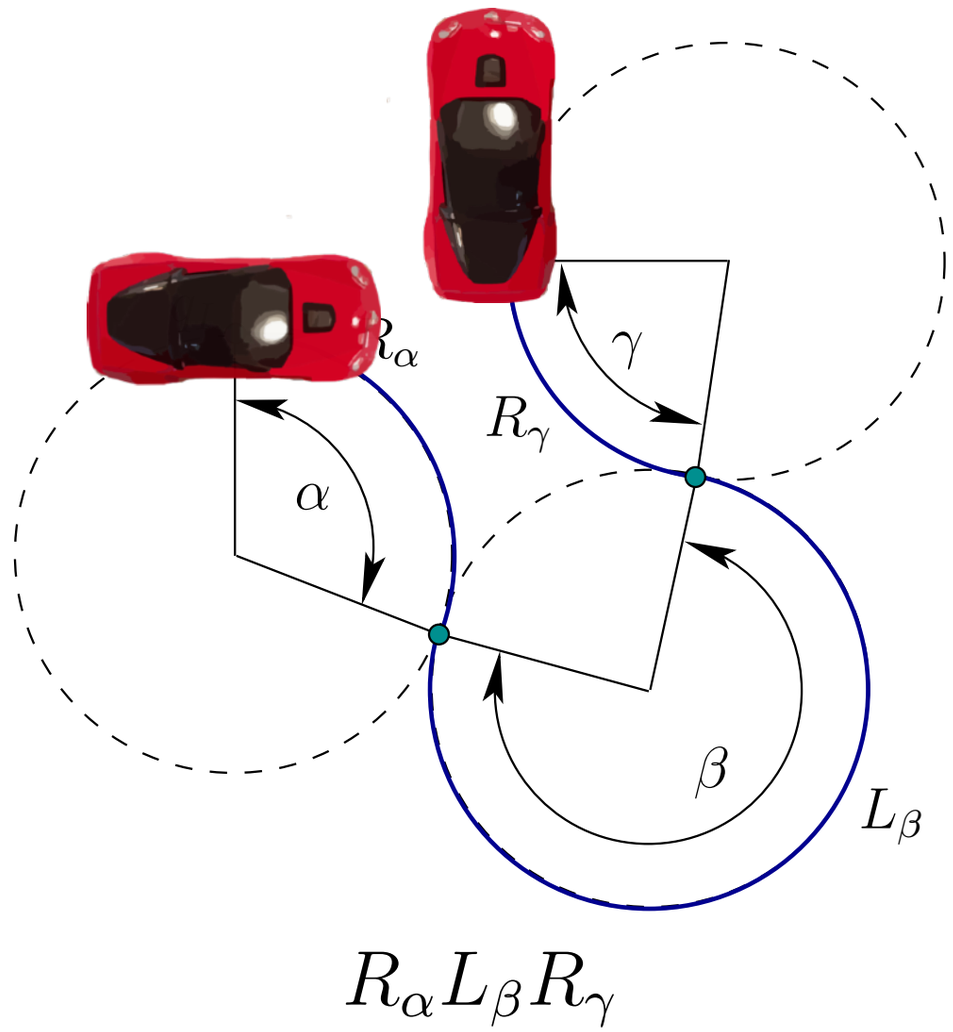


$R_\alpha L_\beta R_\gamma$

Case 1: We can analytically solve the BVP

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

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



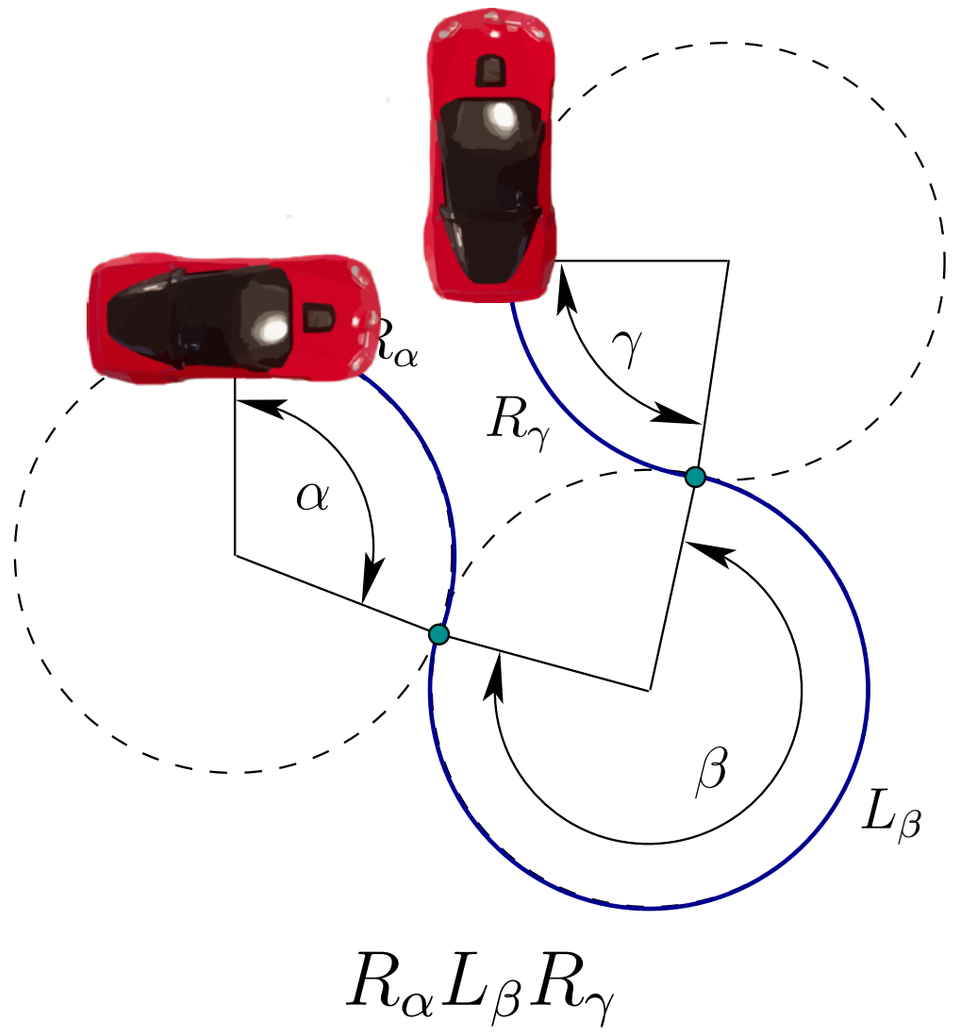
$R_\alpha L_\beta R_\gamma$

Case 1: We can analytically solve the BVP

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

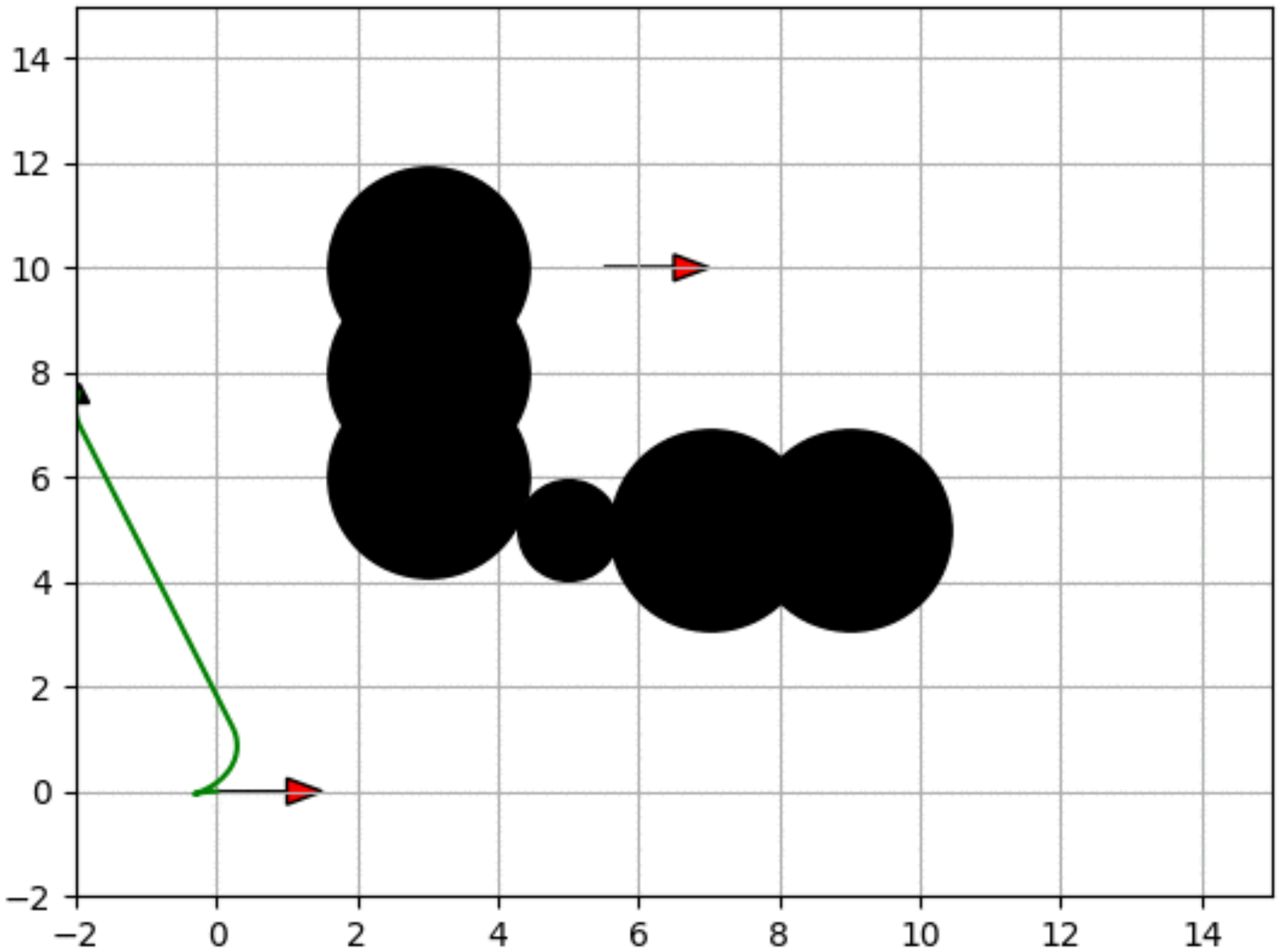
$$\{LRL,\ RLR,\ LSL,\ LSR,\ RSL,\ RSR\}.$$

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



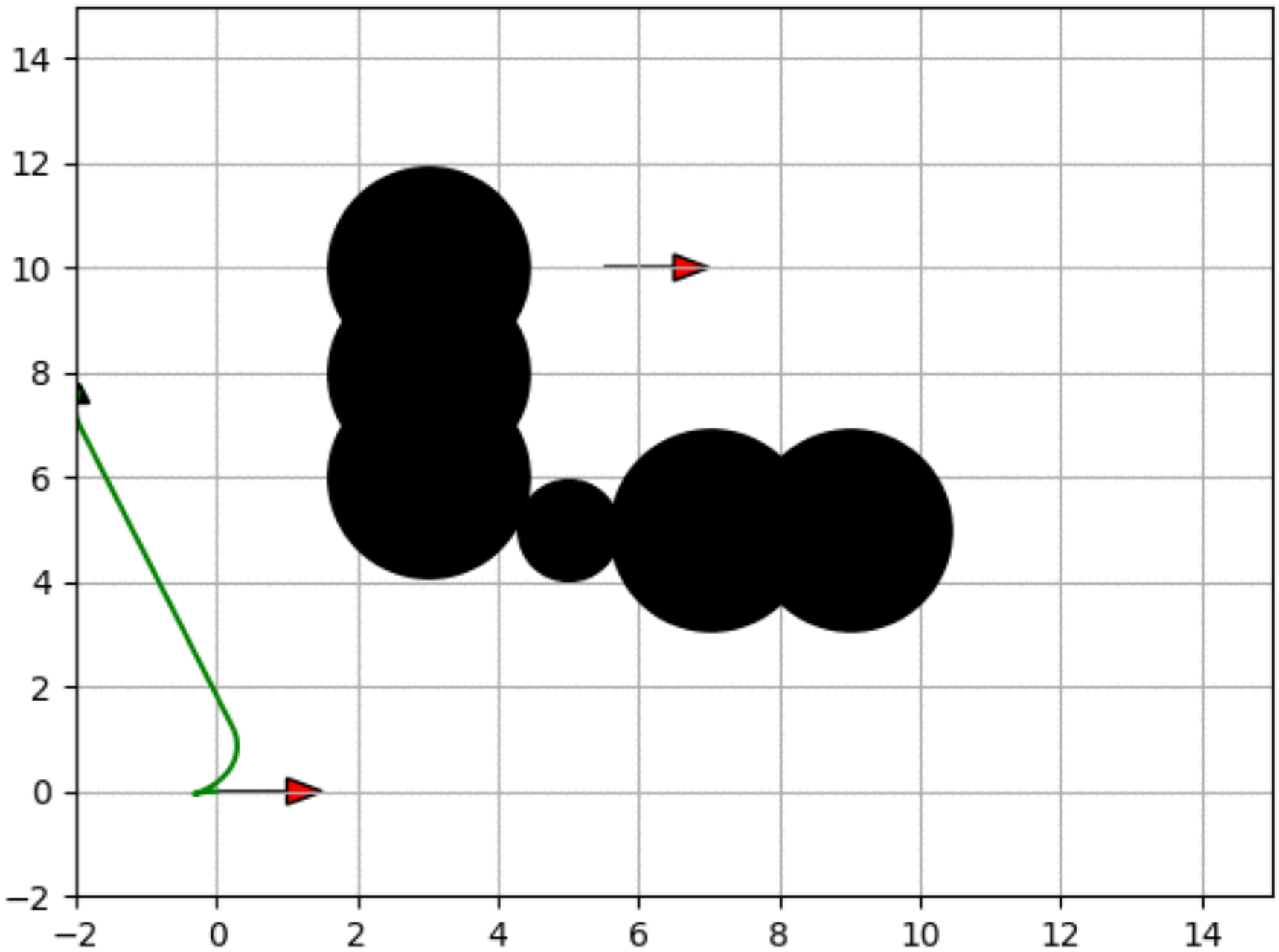
$R_\alpha L_\beta R_\gamma$

Random sampling with Dubins steering



https://github.com/AtsushiSakai/PythonRobotics

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?

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

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

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

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

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

Case 2: We need to numerically solve the BVP

What if you were changing the steering rate?

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

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

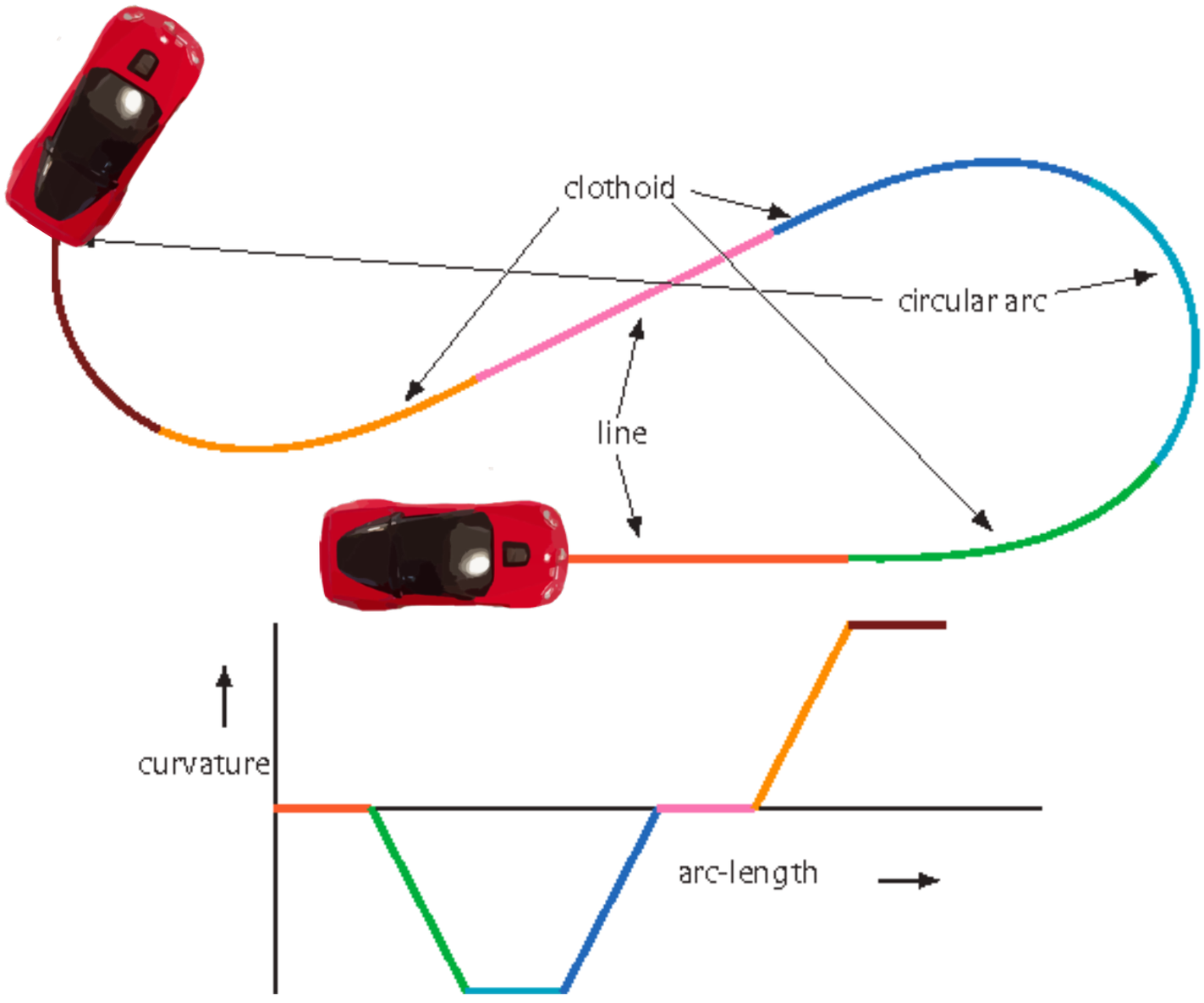
$$q_2 = (x_2, y_2, \theta_2, \dot{\theta}_2)$$

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

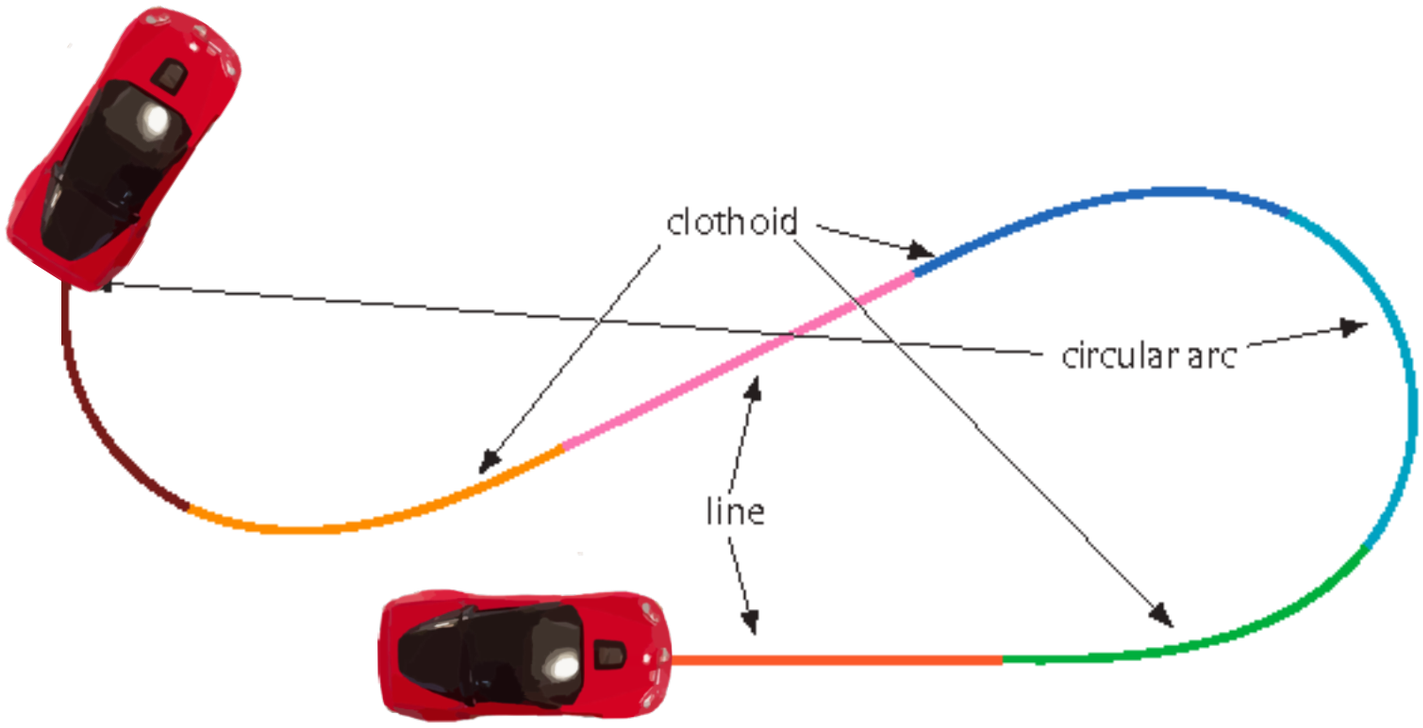
No longer called Dubins path, called Clothoid path

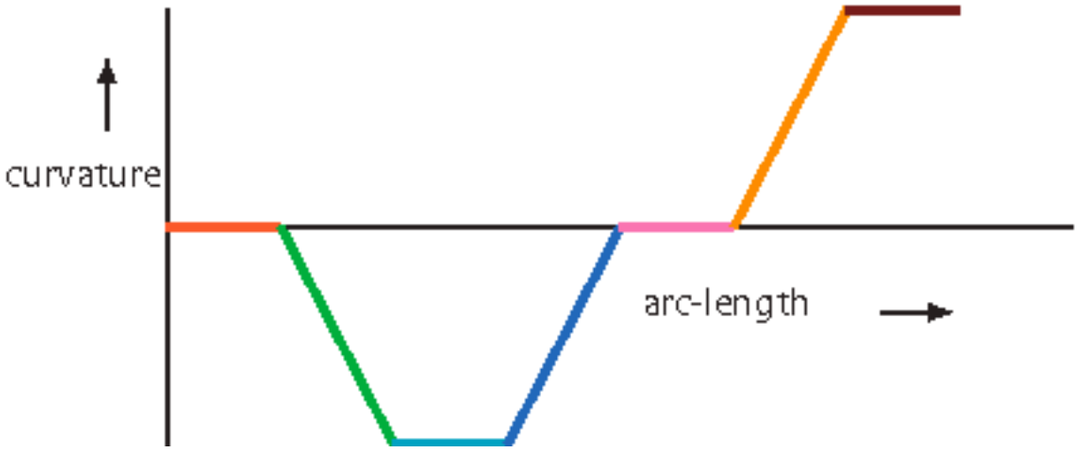
Can't solve analytically ... need numerical solutions

Case 2: We need to numerically solve the BVP



Case 2: We need to numerically solve the BVP

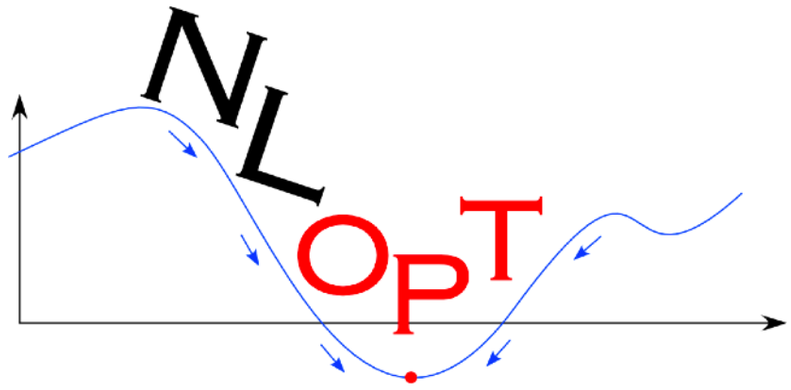




Create a non-linear optimization problem

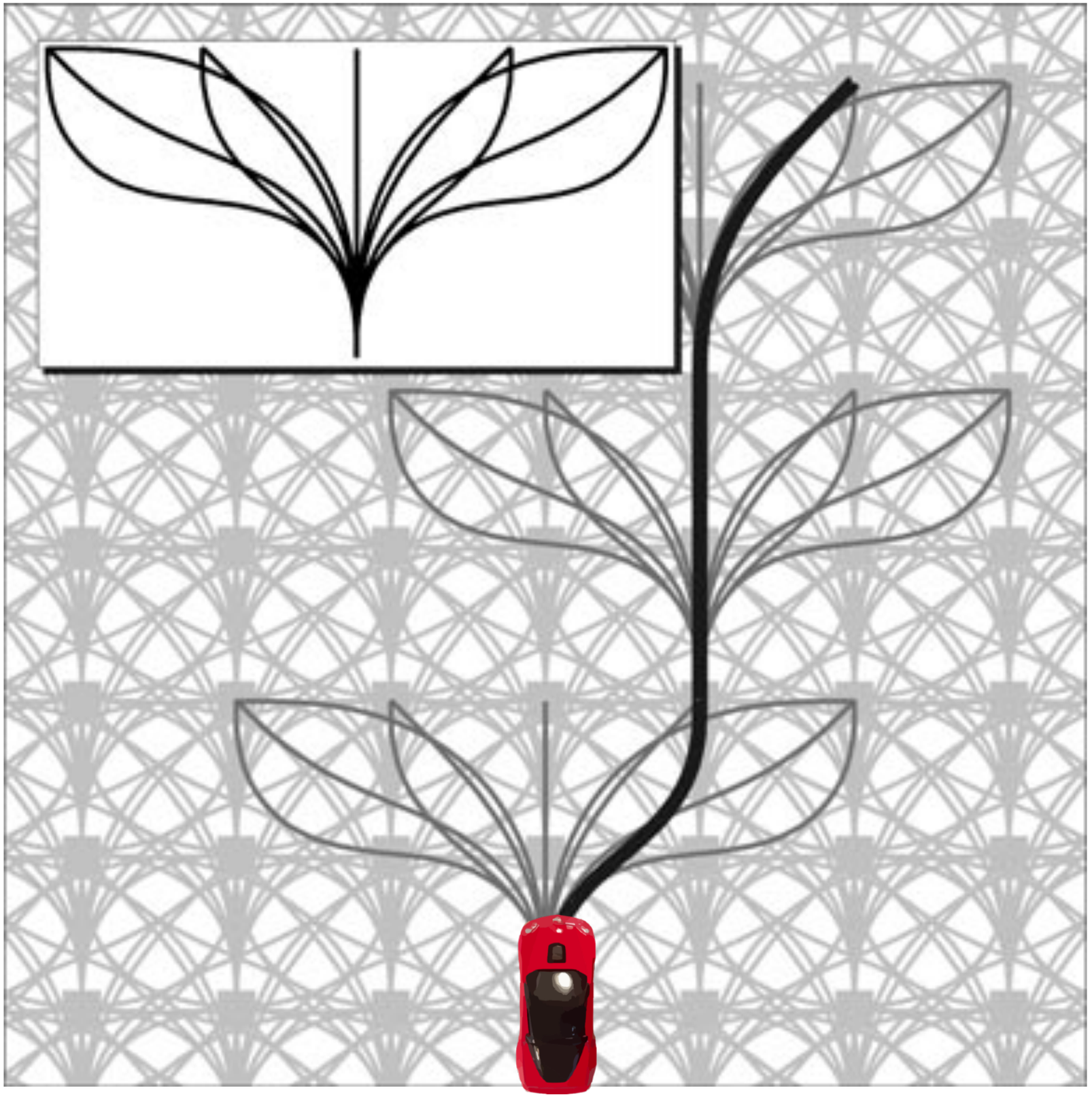
Represent the path as a curvature polynomial

Enforce end point constraints



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

Example of a state lattice



Pivtariko 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?

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

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

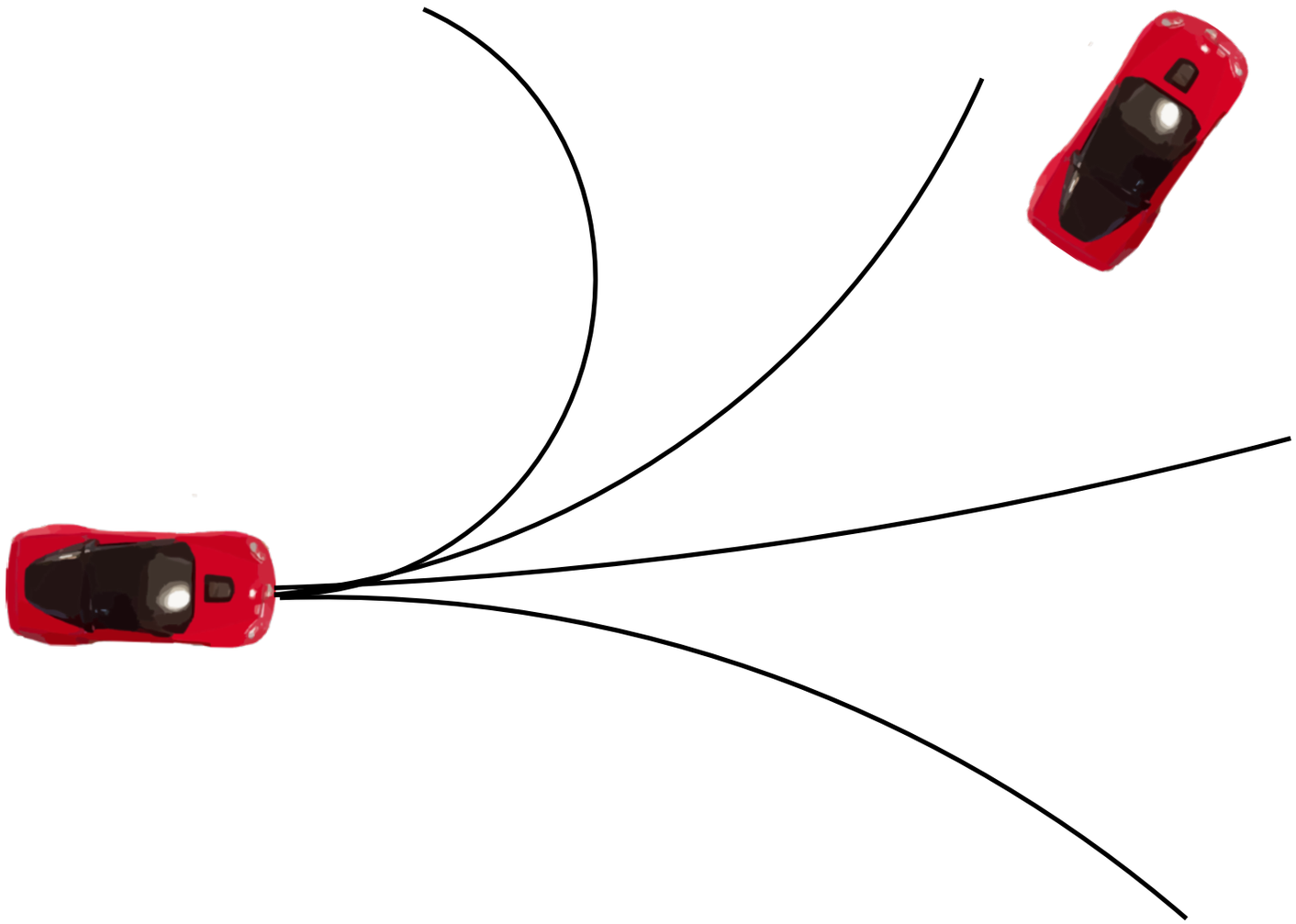
2. Even if it did, too expensive to run online

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.

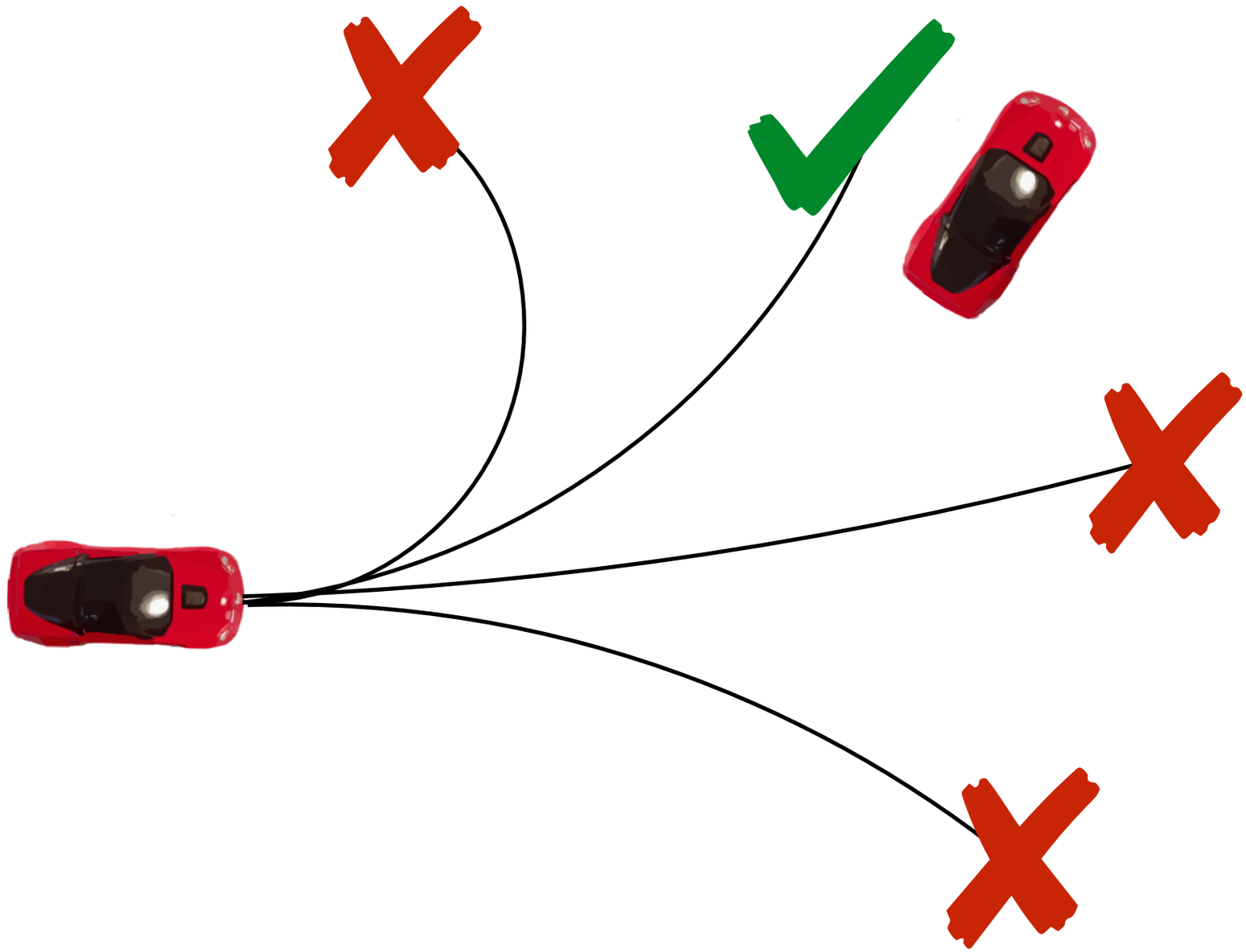
Case 3: We can't even solve the BVP

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



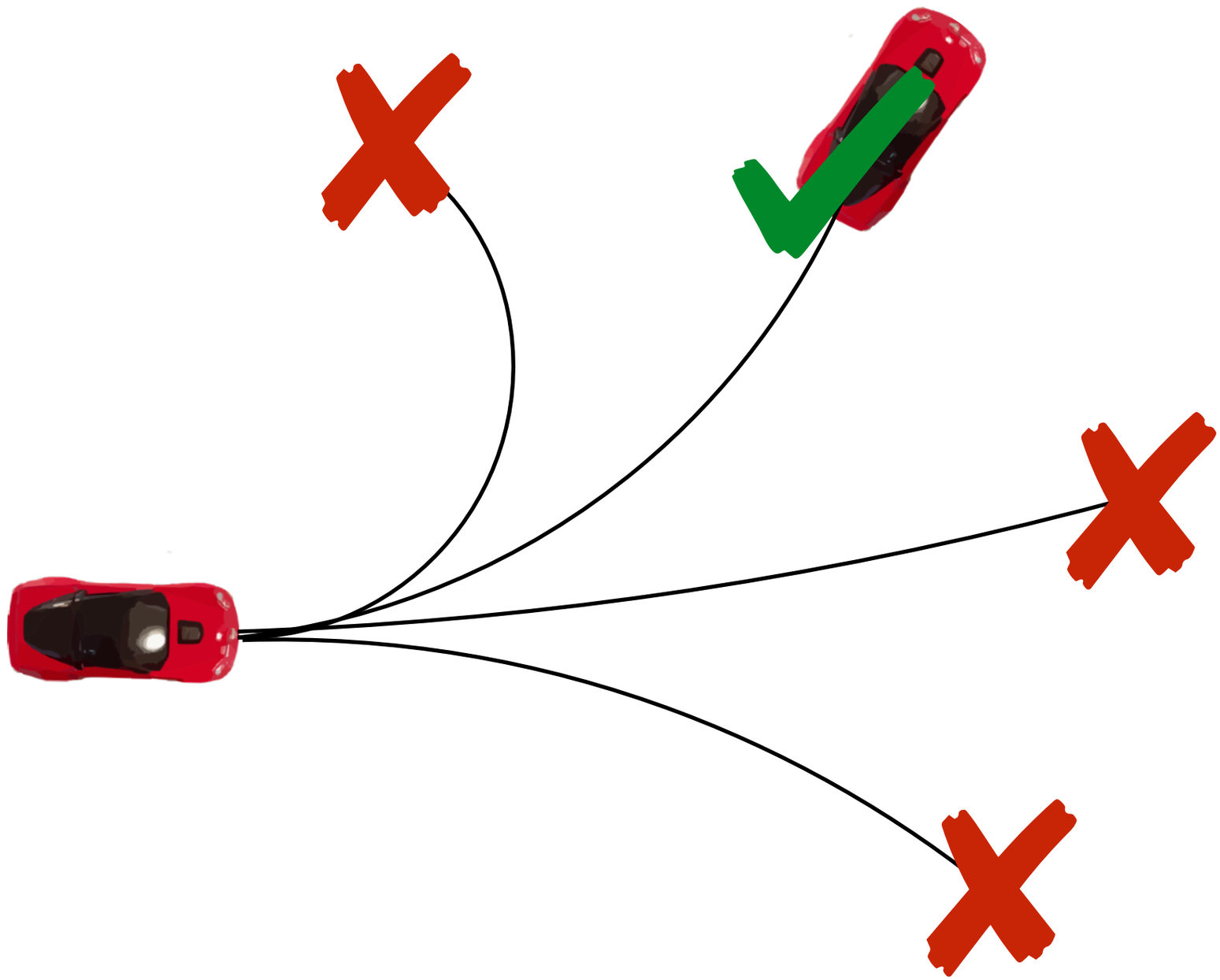
Case 3: We can't even solve the BVP

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



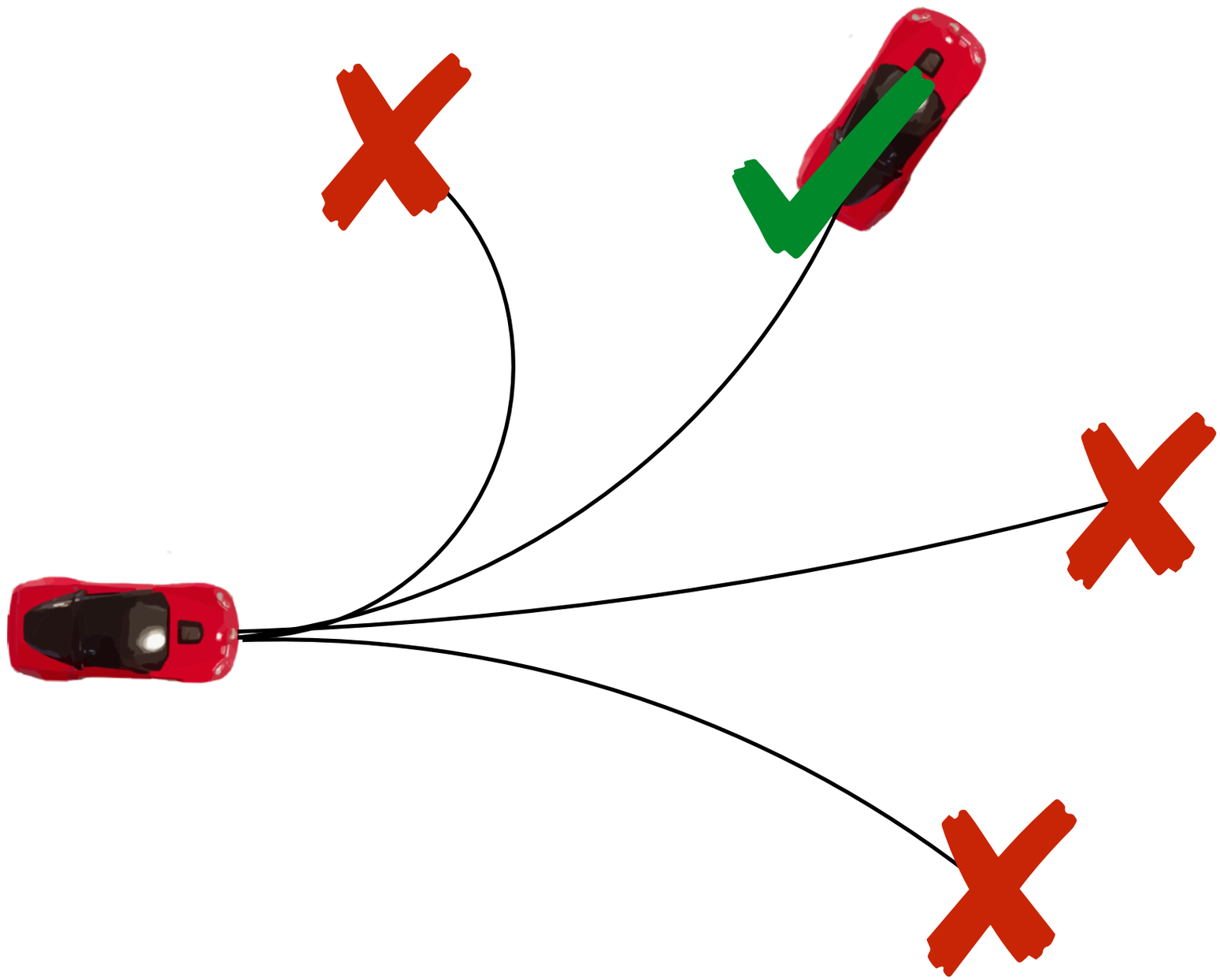
Case 3: We can't even solve the BVP

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



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!

Key Takeaways

Planning for racecar is just like Piano Movers

Replace straight line STEER function with Dubins path

Search technique remains the same