Dealing with complex cost functions: Trajectory library

Sanjiban Choudhury

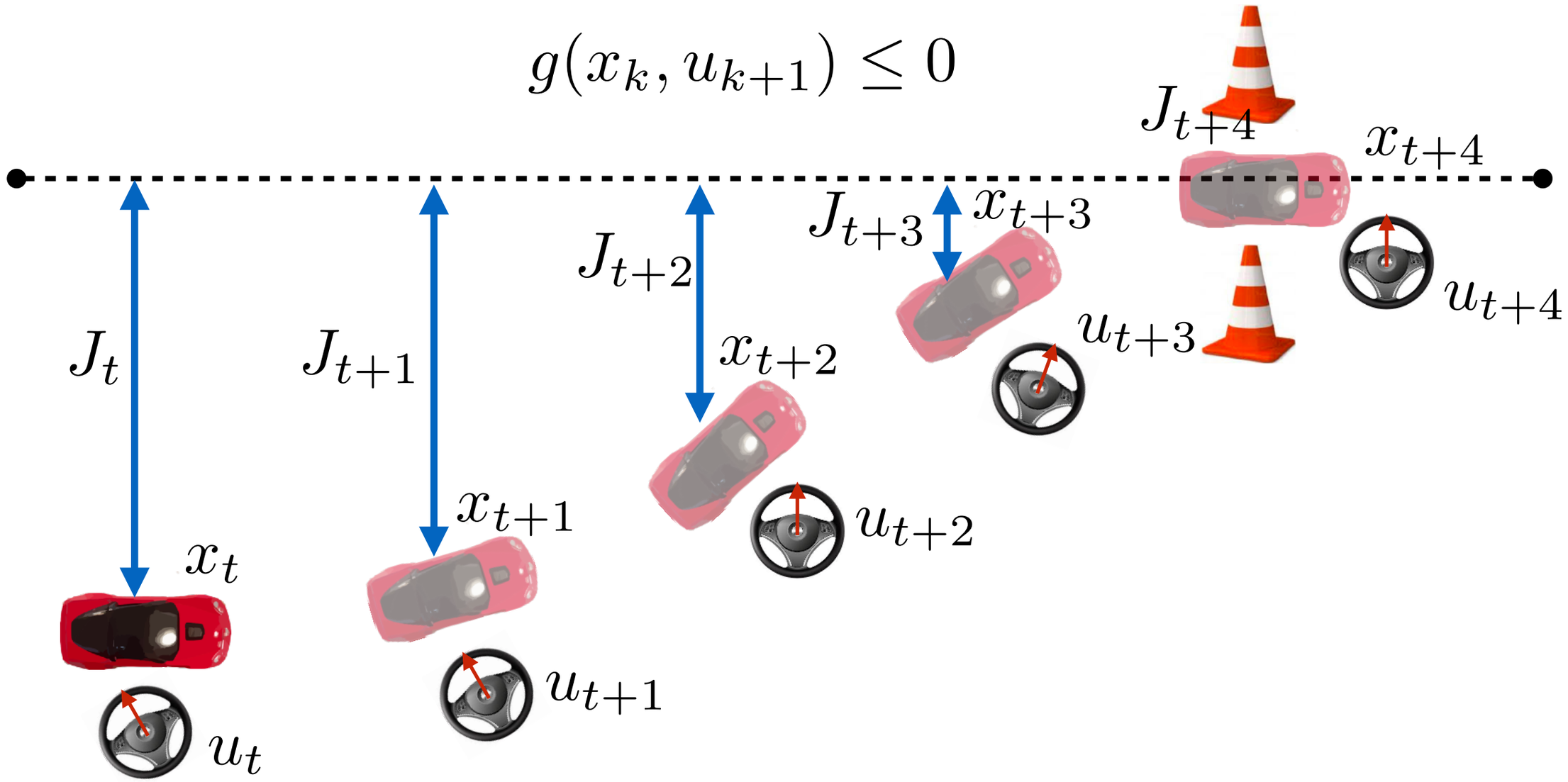
TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

Recap: Model predictive control (MPC)

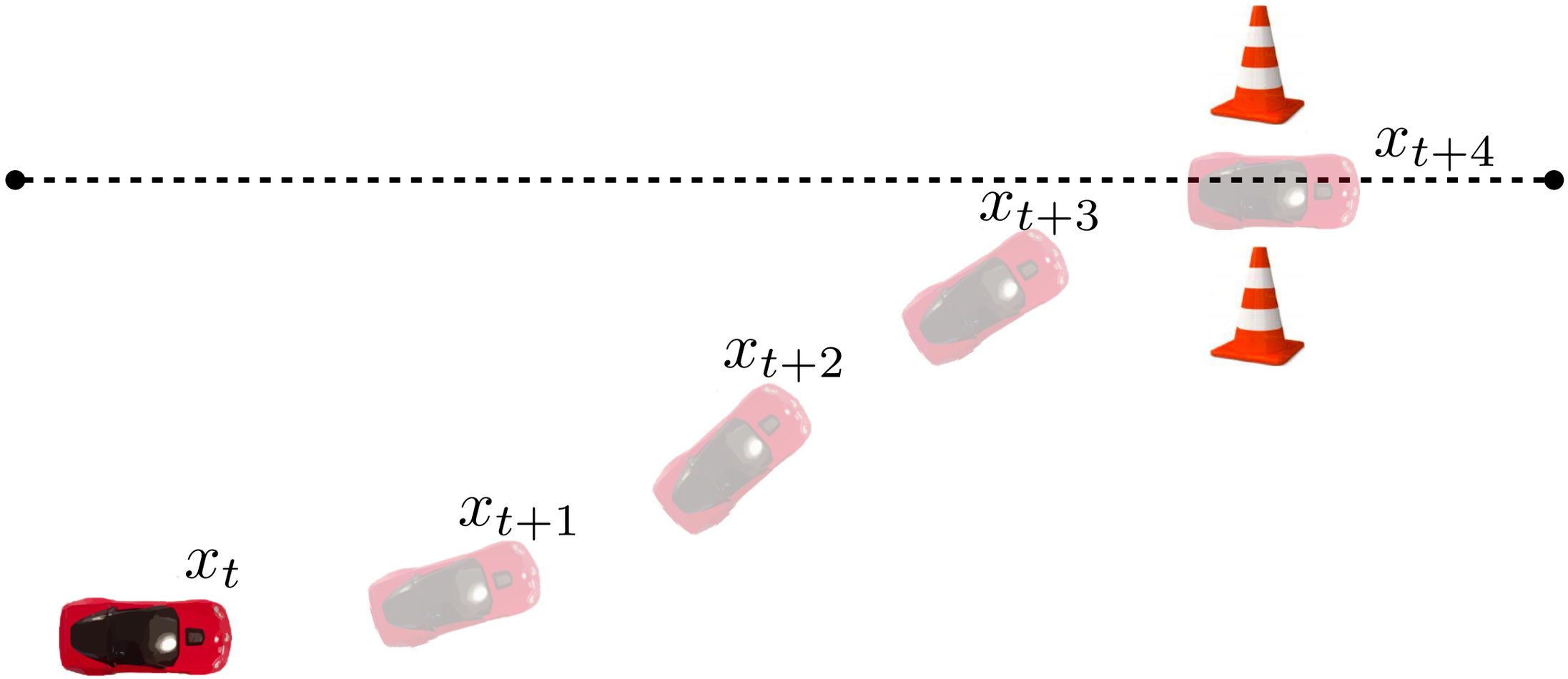
$$\min_{u_{t+1},\dots u_{t+H}} \sum_{k=t}^{t+H-1} J(x_k, u_{k+1})$$

$$x_{k+1} = f(x_k, u_{k+1})$$

$$g(x_k, u_{k+1}) \leq 0$$

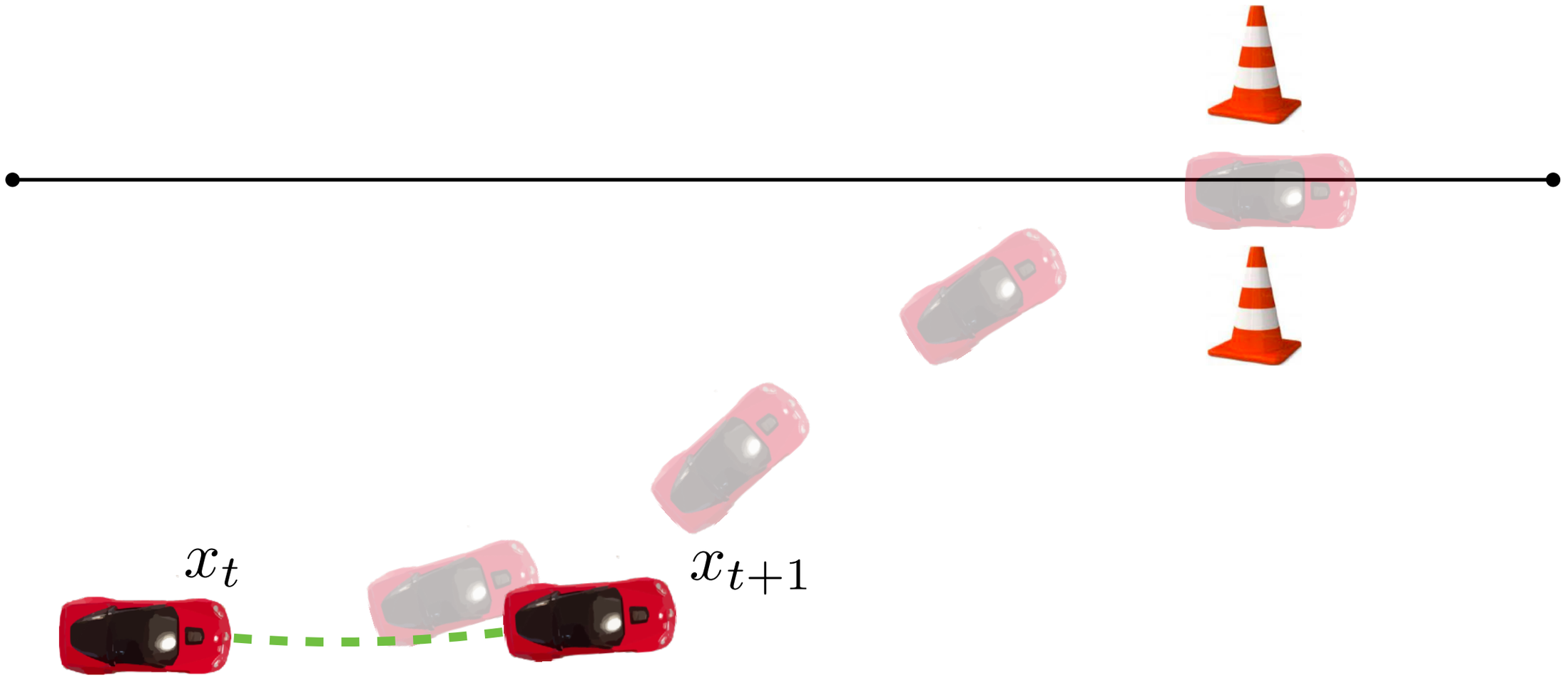


How are the controls executed?



Step 1: Solve optimization problem to a horizon

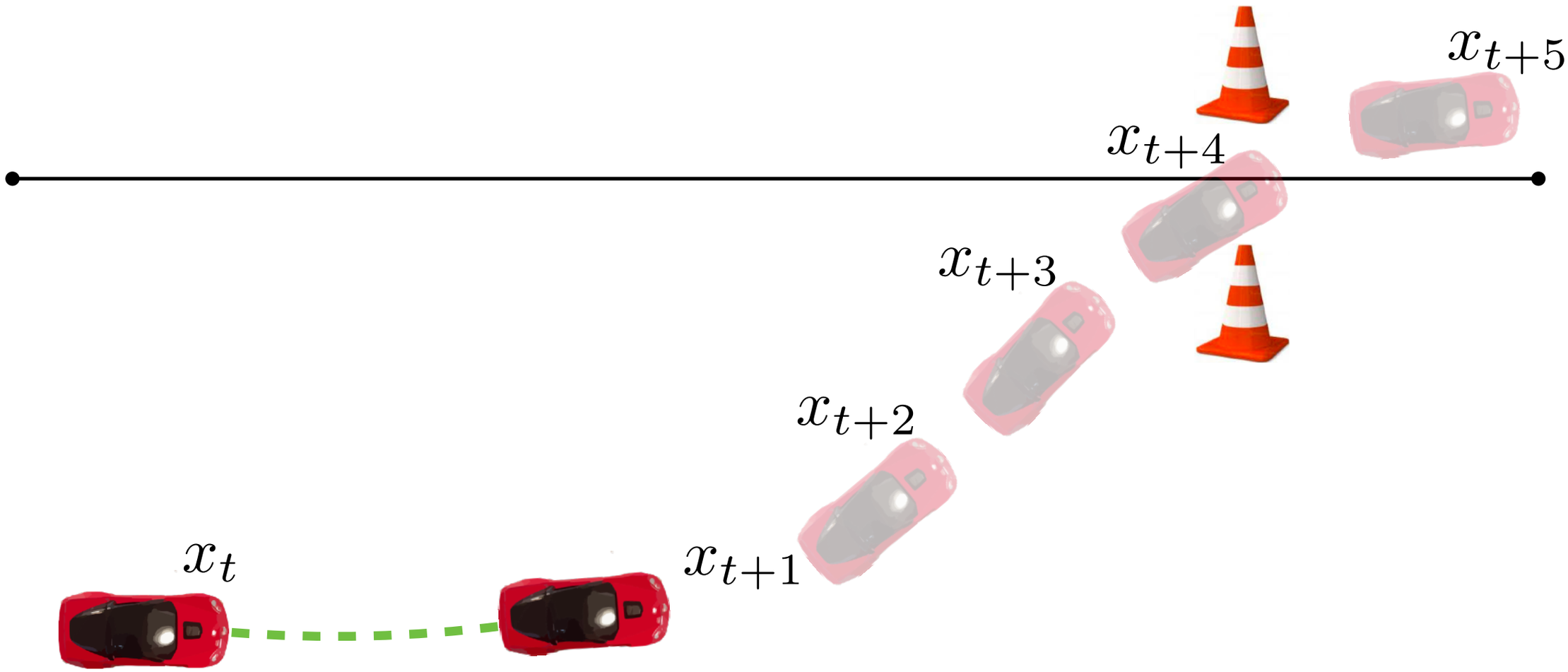
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

Step 3: Repeat!

What is the cost? What constraints?

$$\min_{u_{t+1},\dots u_{t+H}} \sum_{k=t}^{t+H-1} \text{Cost}$$

(plan till horizon H)

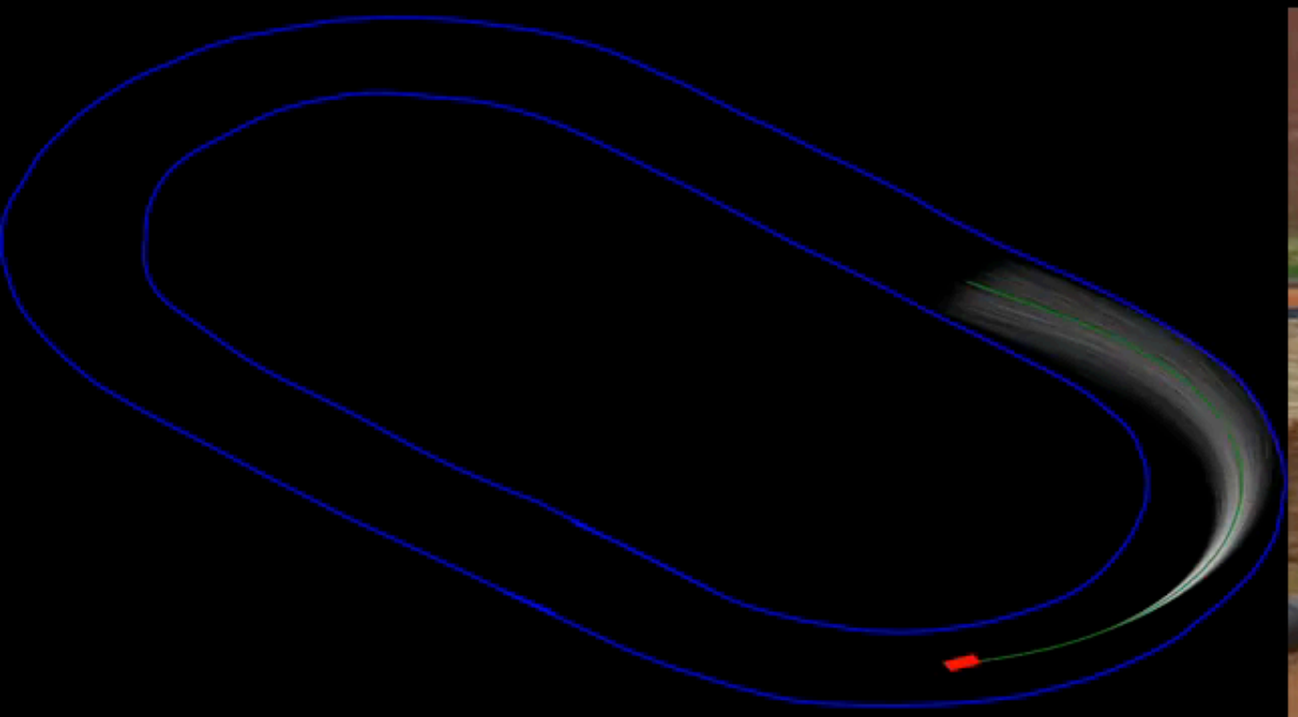
$$x_{k+1} = f(x_k, u_{k+1})$$

$$g(x_k, u_{k+1}) \leq 0$$

Constraints

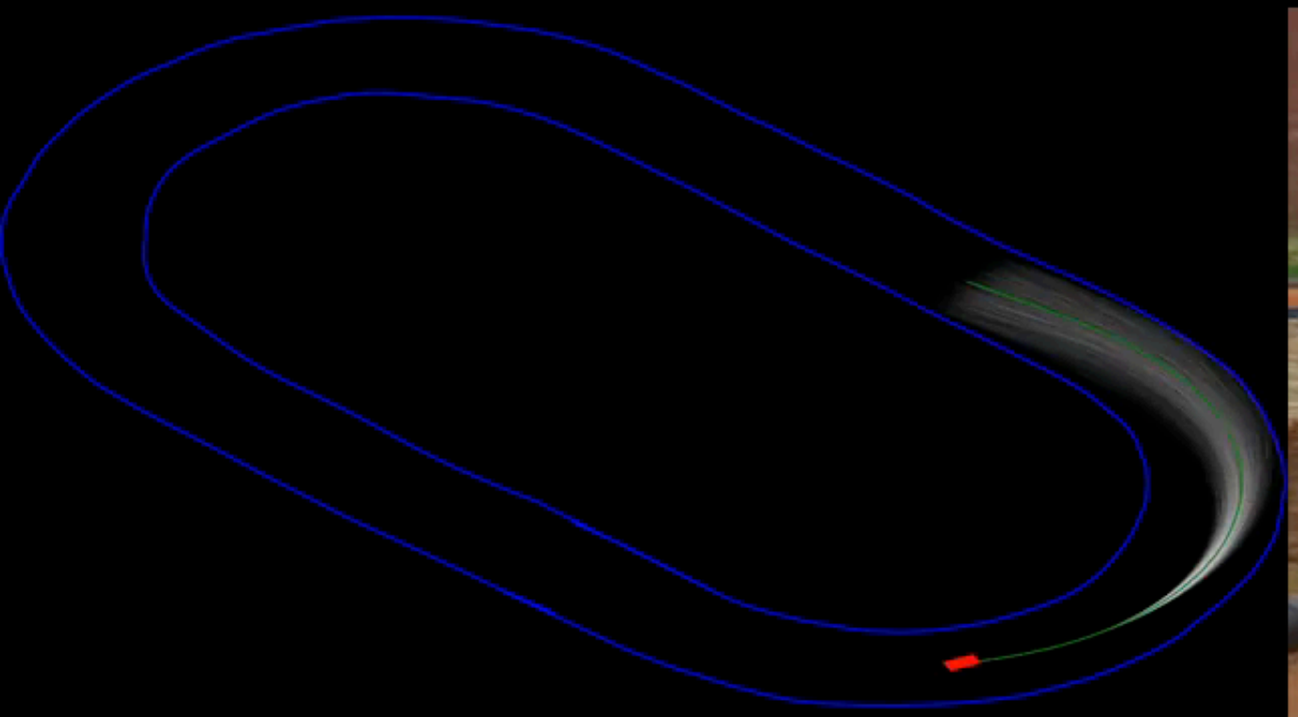
What is the cost? What constraints?

2560, 2.5 second trajectories sampled
with cost-weighted average @ 60 Hz



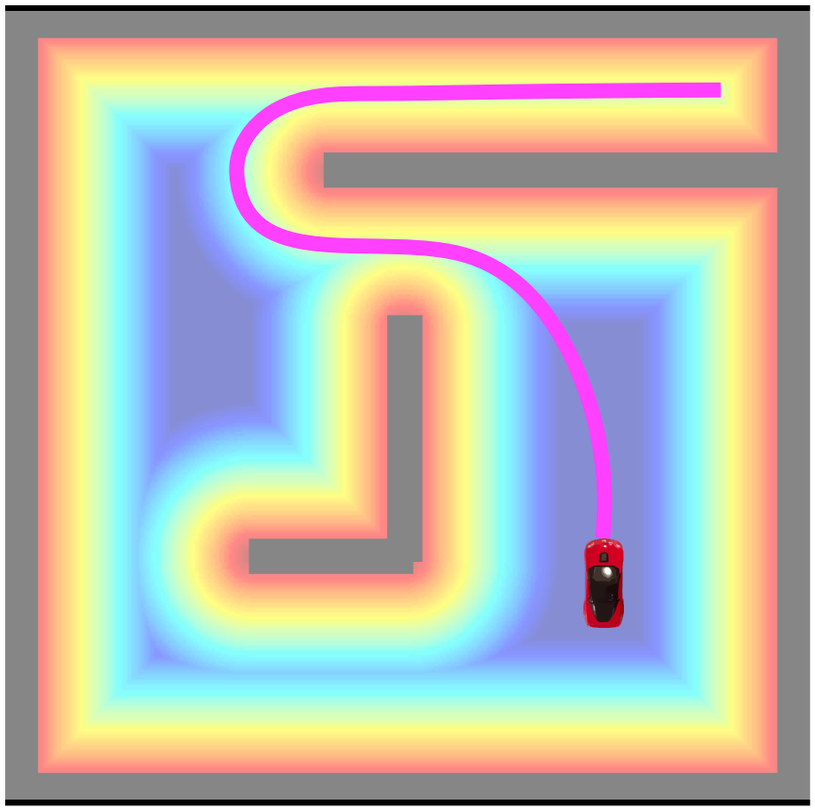
What is the cost? What constraints?

2560, 2.5 second trajectories sampled
with cost-weighted average @ 60 Hz



Examples of complex cost functions

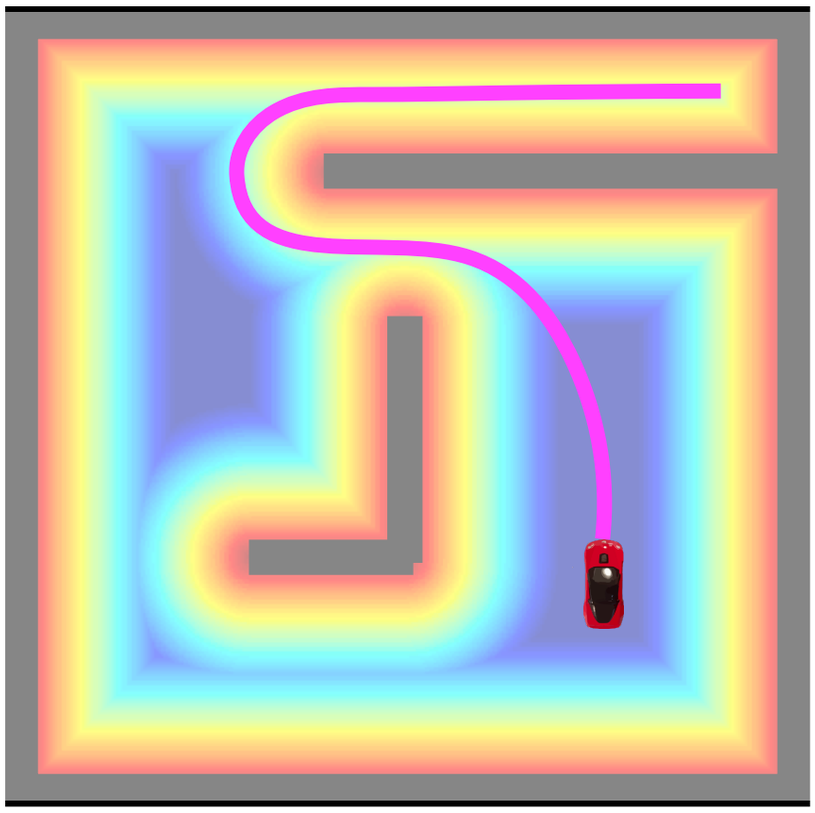
Examples of complex cost functions



Proximity to
obstacles

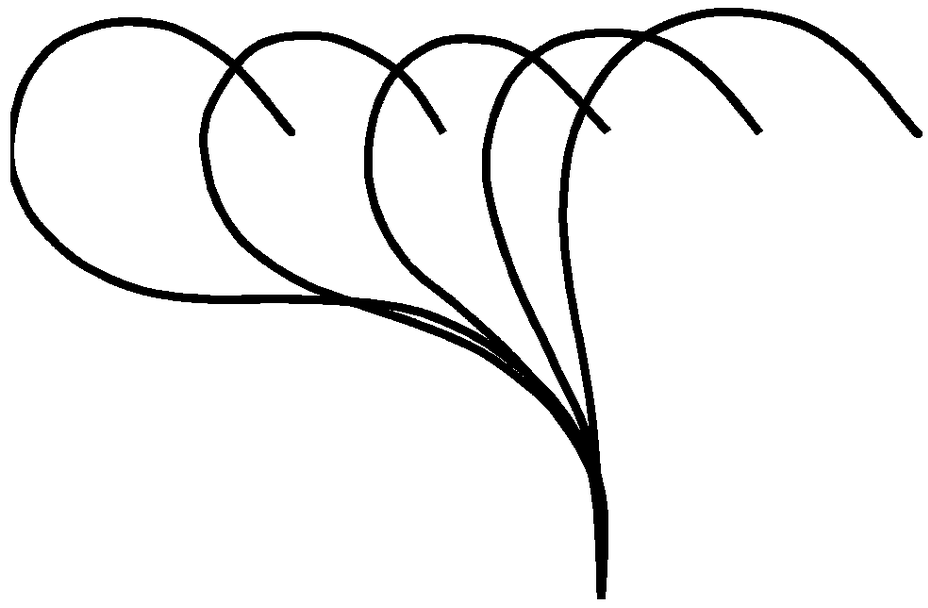
(Ratliff et al. 2009)

Examples of complex cost functions



Proximity to obstacles

(Ratliff et al. 2009)



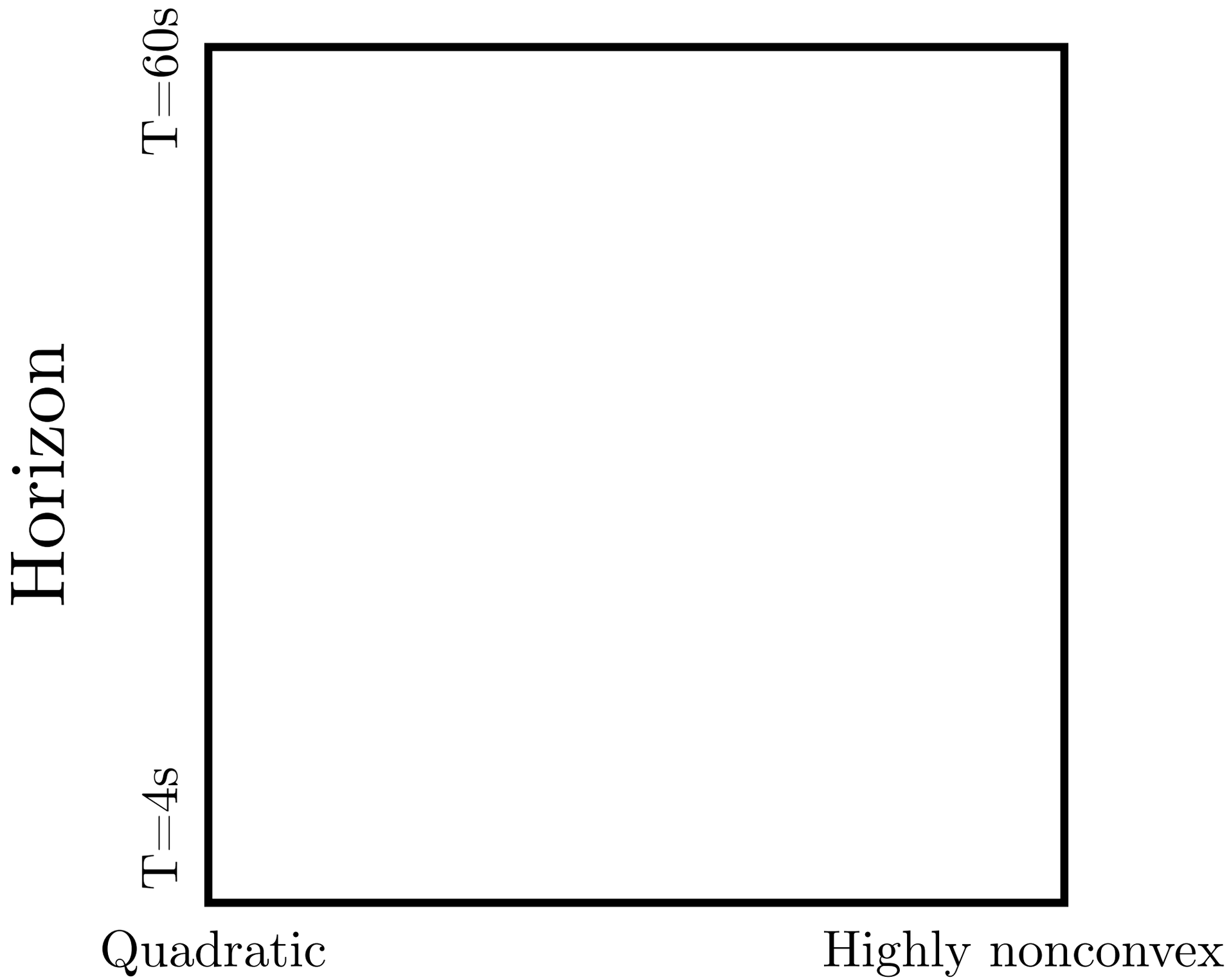
Curvature

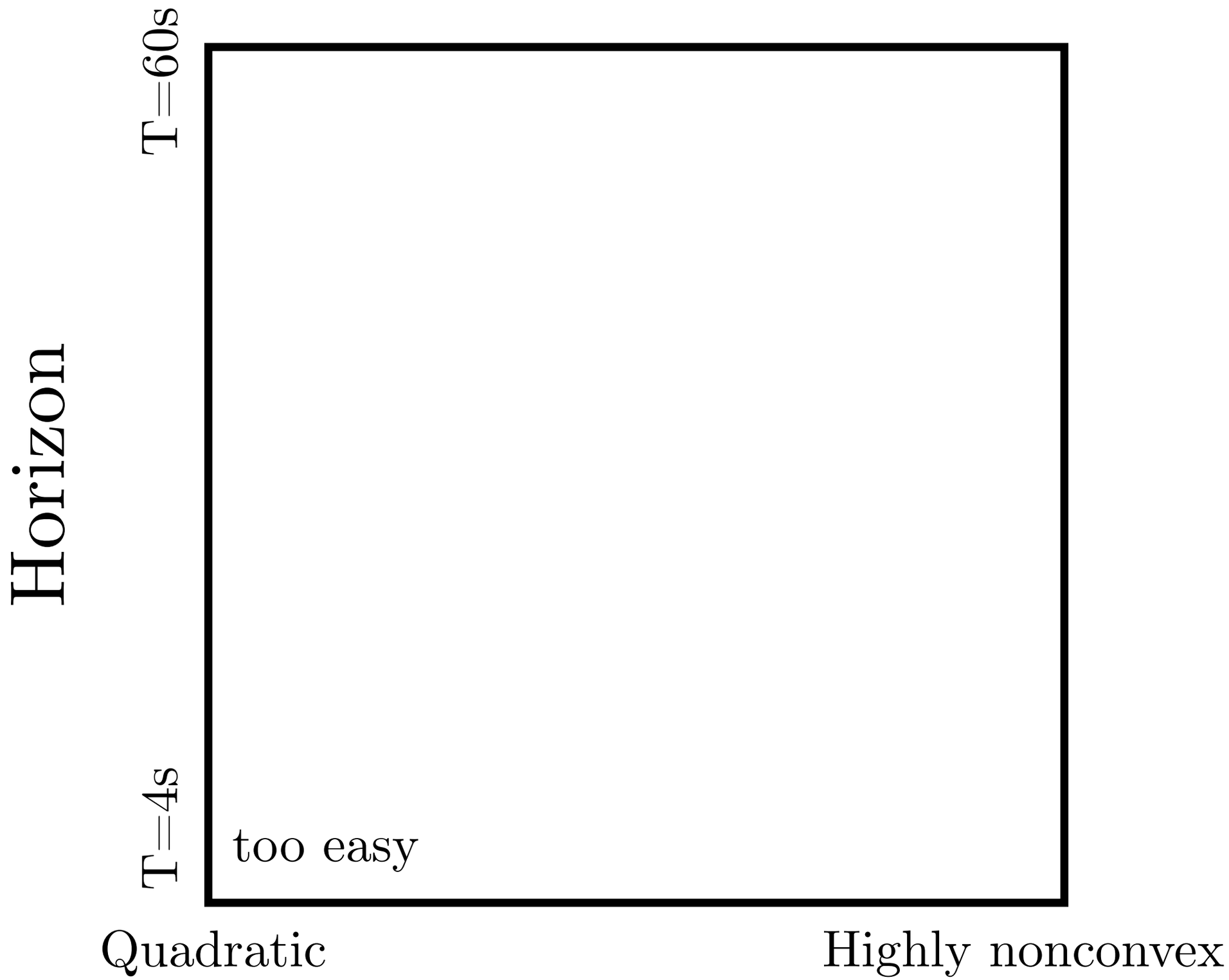
(Kelly et al. 2003)

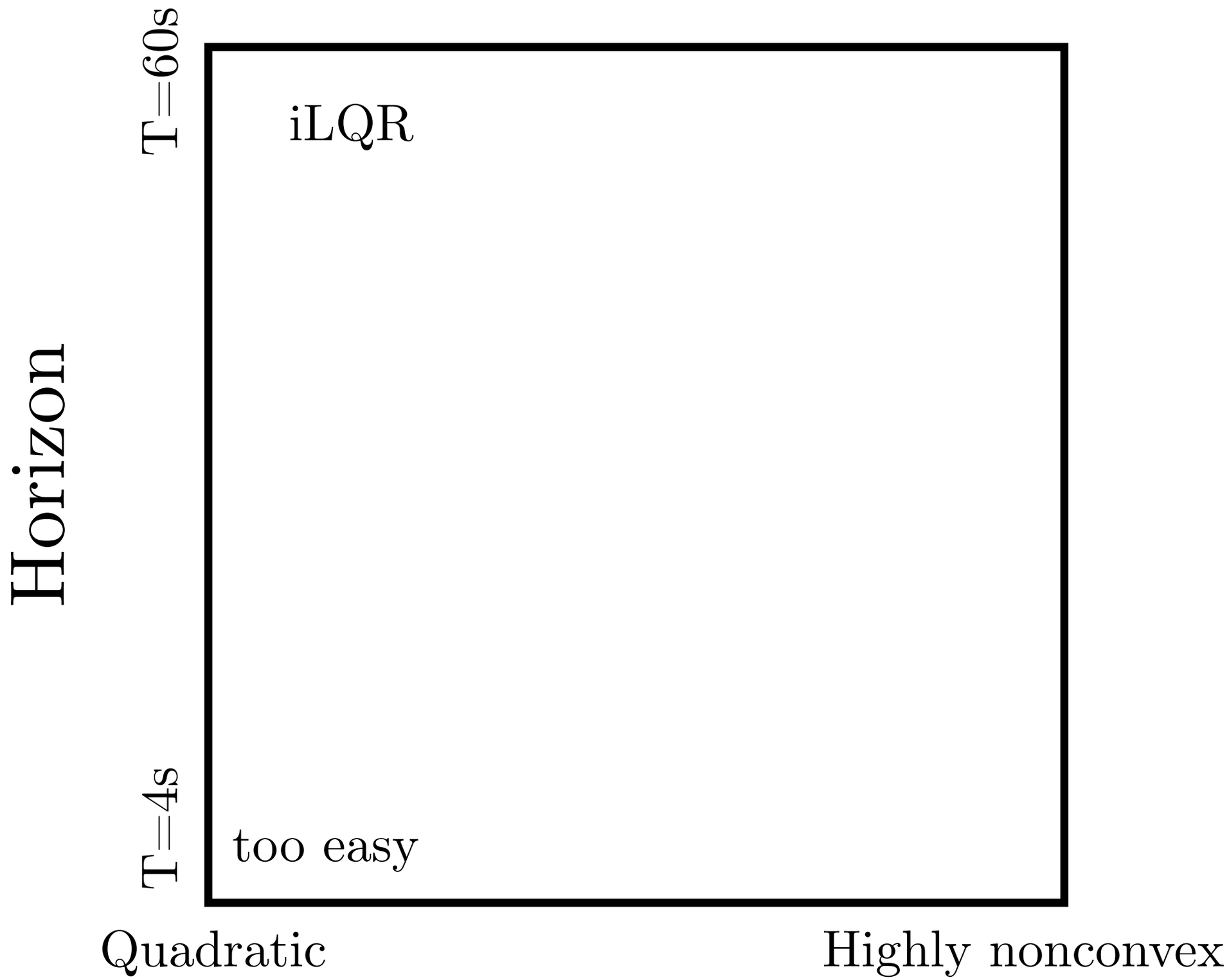
Question: Why cant we use iLQR for everything?

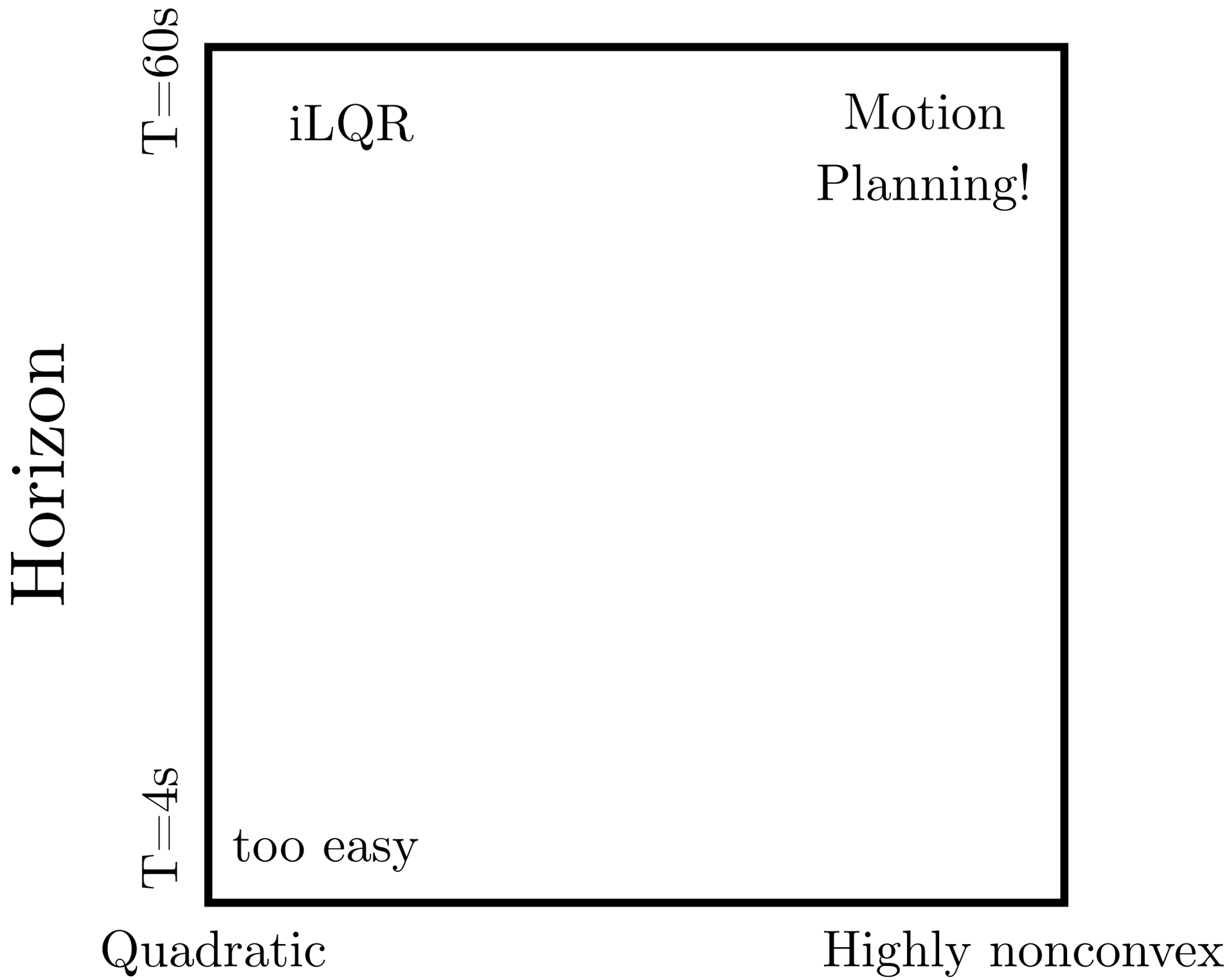
Question: Why cant we use iLQR for everything?

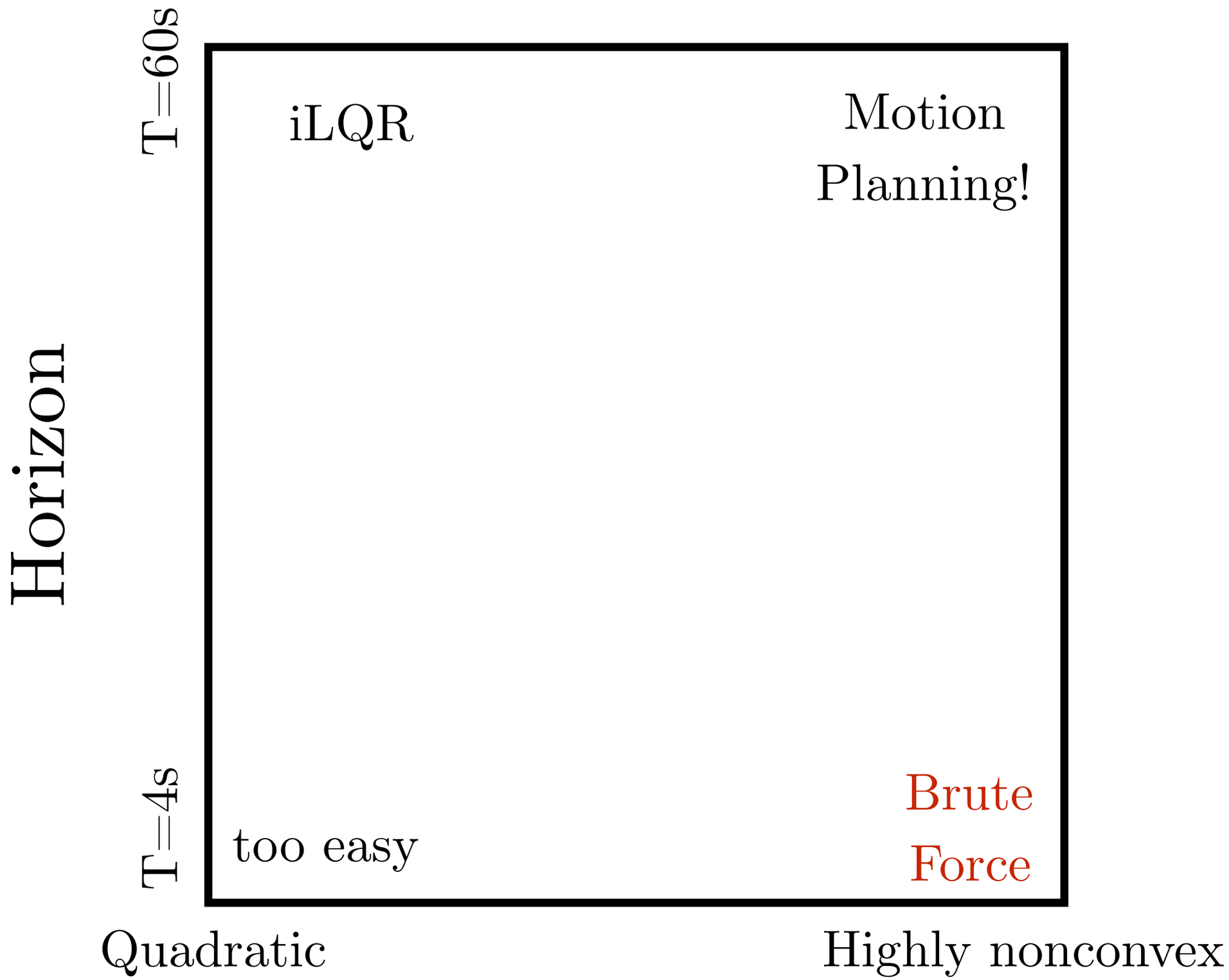
Highly non-convex - can converge to terrible local minima


Horizon
T=60s
T=4s
Quadratic
Highly nonconvex
Cost function complexity


Horizon
T=60s
T=4s
too easy
Quadratic
Highly nonconvex
Cost function complexity


Horizon
T=60s
T=4s
iLQR
too easy
Quadratic
Highly nonconvex
Cost function complexity


Horizon
T=60s
T=4s
iLQR
Motion
Planning!
too easy
Quadratic
Highly nonconvex
Cost function complexity


Horizon
T=60s
T=4s
iLQR
Motion
Planning!
too easy
Brute
Force
Quadratic
Highly nonconvex
Cost function complexity

Brute Force??? Really??

Brute Force??? Really??

But wait ... it's more nuanced than that

Brute Force??? Really??

But wait ... it's more nuanced than that

Brute Force??? Really??

But wait ... it's more nuanced than that

Dealing with exponential branching factor

Brute Force??? Really??

But wait ... it's more nuanced than that

Dealing with exponential branching factor

Still need to evaluate this online (>10 Hz)

Brute Force??? Really??

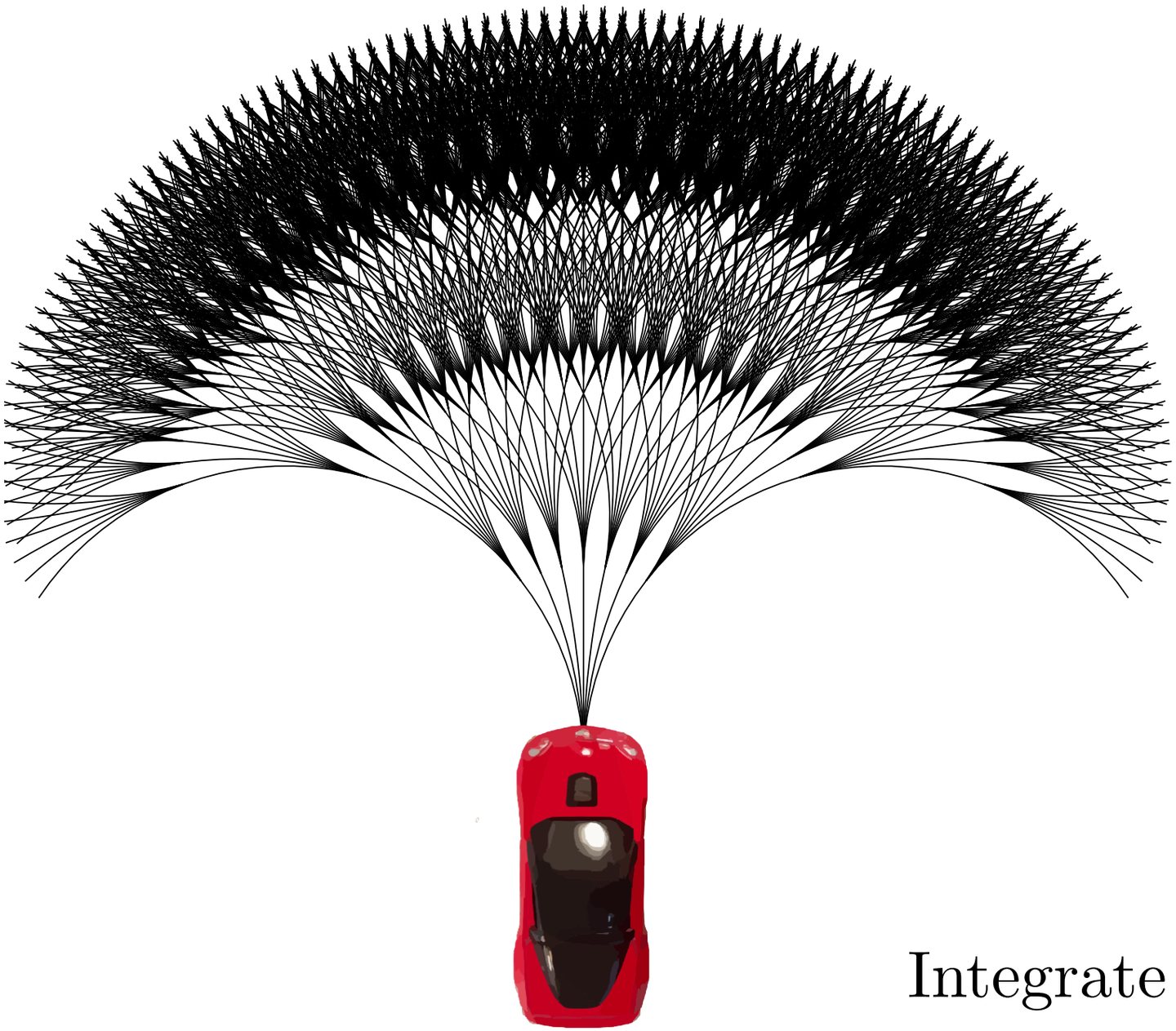
But wait ... it's more nuanced than that

Dealing with exponential branching factor

Still need to evaluate this online (>10 Hz)

Can always post-process with iLQR

Create a gigantic library of paths



Discretize steering angle
into d bins (set d=7)

Branch every t sec
for N*t horizon
(N=5)

Get 7^5 = 16,807
control trajectories

$u(t)$

Integrate dynamics to get $x(t)$

How will we use it for MPC?

1. Iterate (2-3) for every path in library.

2. Compute constraint. If violated, chuck out path

3. Compute cost.

4. Pick path with least cost.

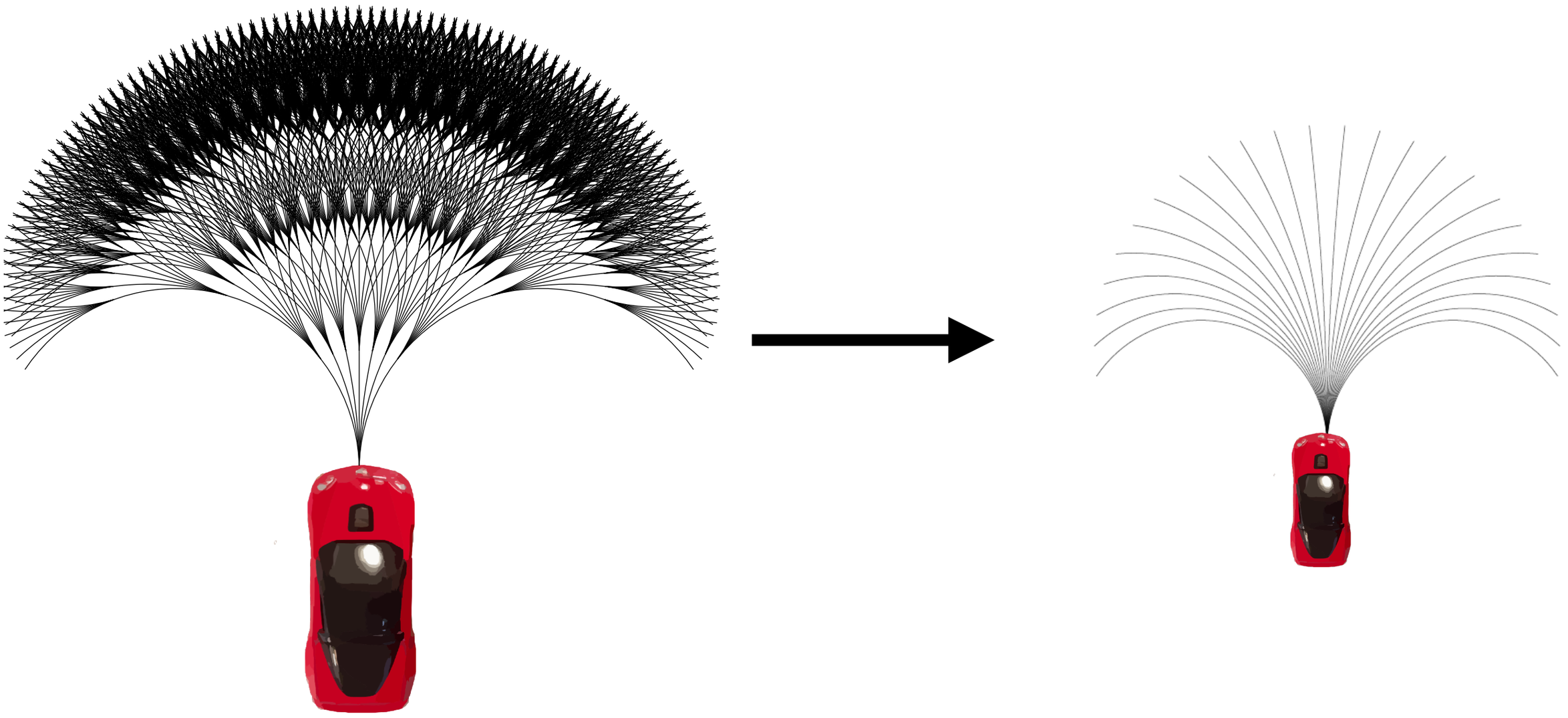
5. Execute first control action. Robot moves. Replan.

Problem: Library too big!!

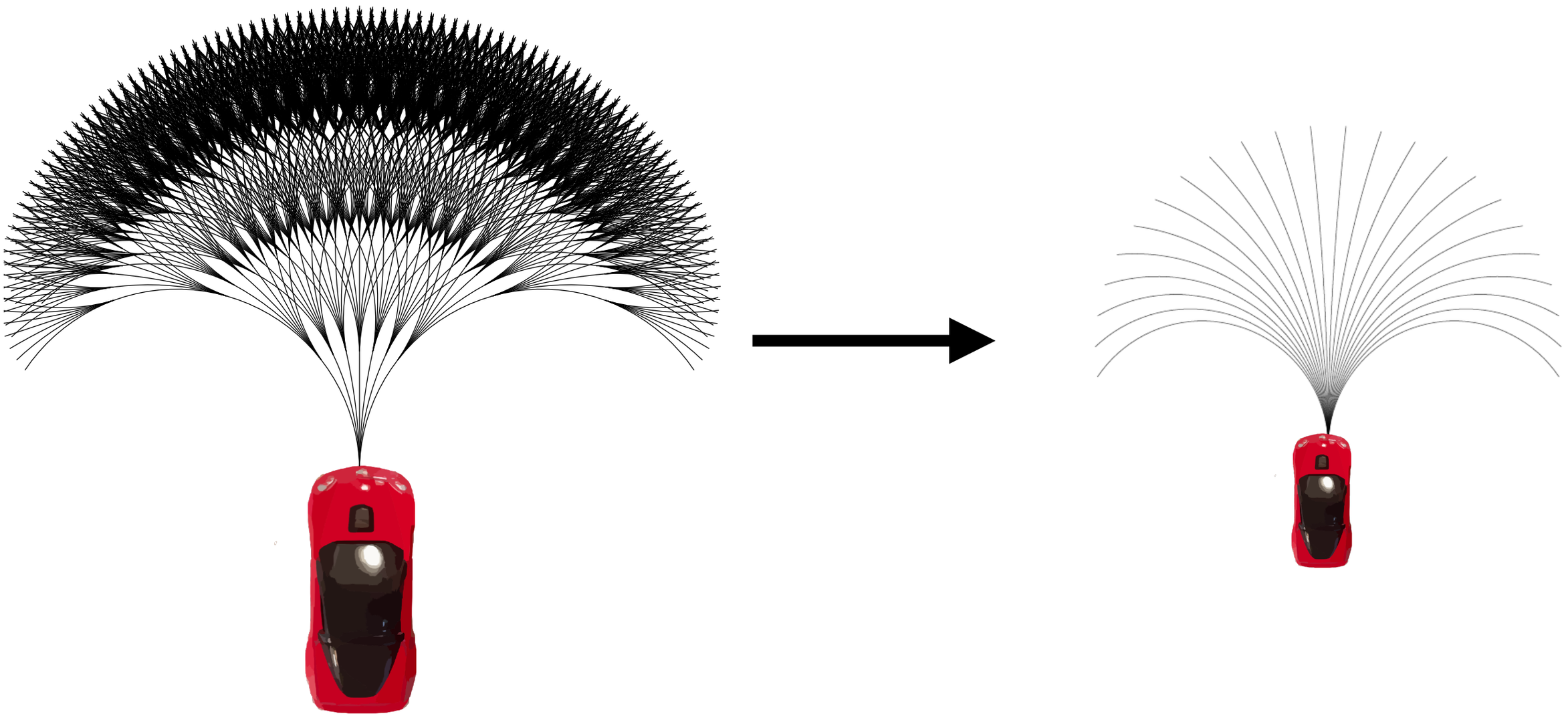
Problem: Library too big!!

Solution: Subsample library

Strategy 1: Uniformly subsample

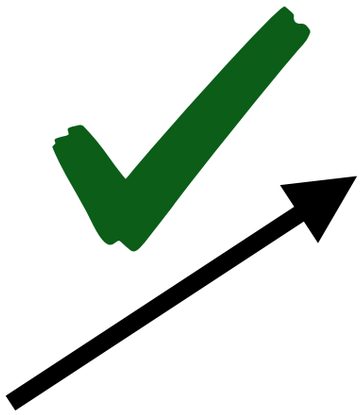


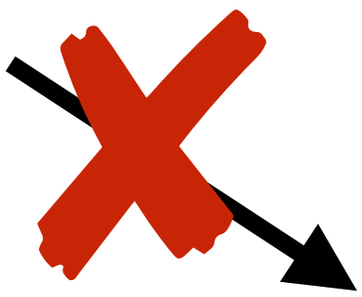
Strategy 1: Uniformly subsample

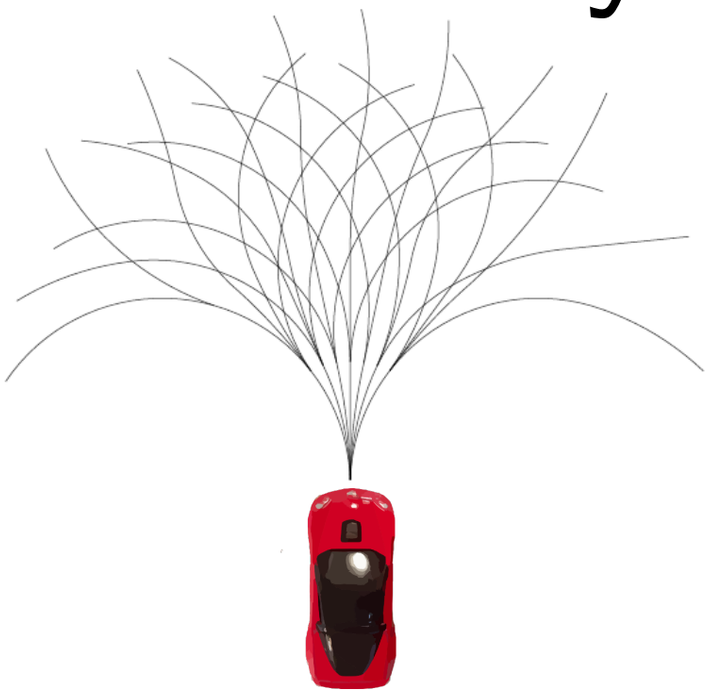


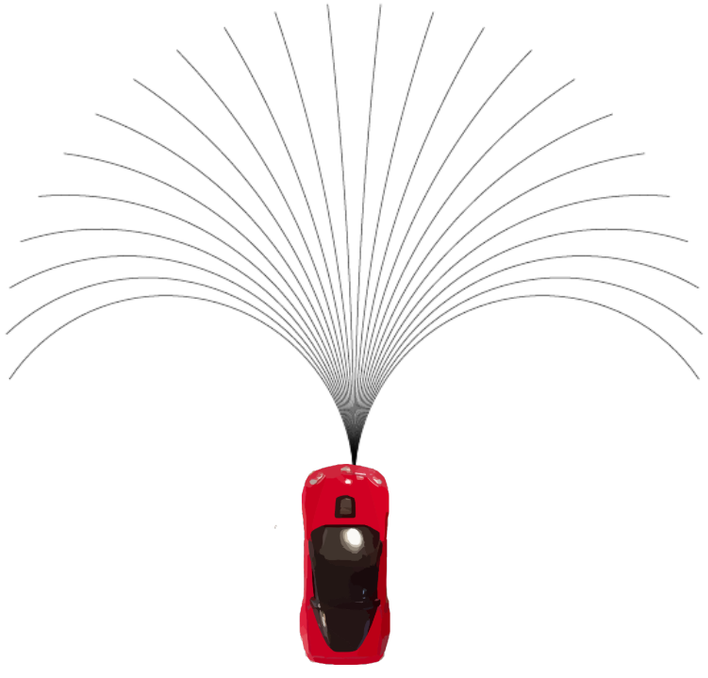
Why is this not a good library?

Strategy 2: Get more diversity





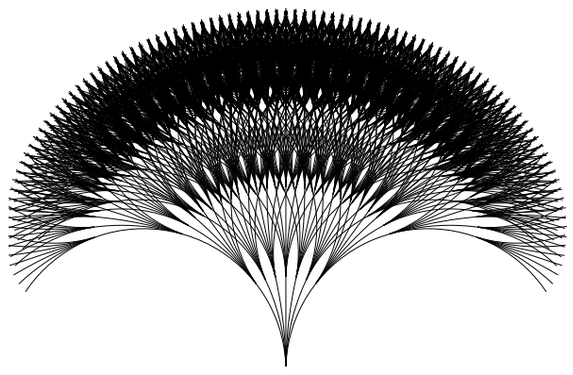




Proposed problem formulation

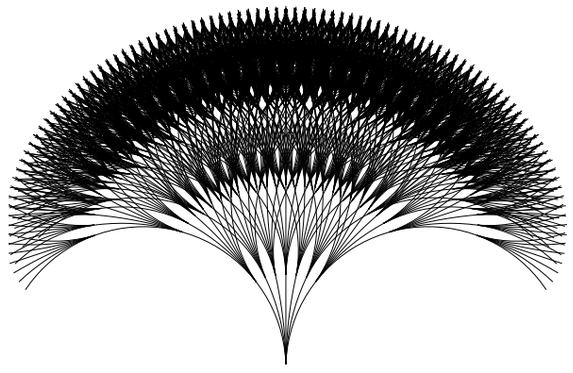
Proposed problem formulation

Given a dense path set D

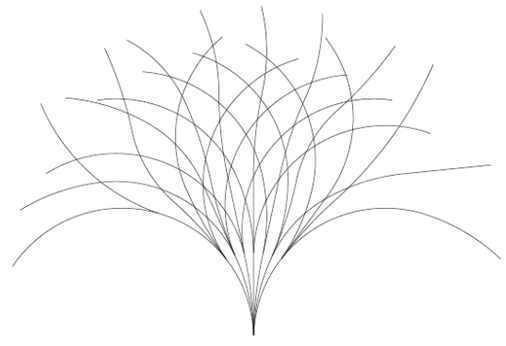


Proposed problem formulation

Given a dense path set D

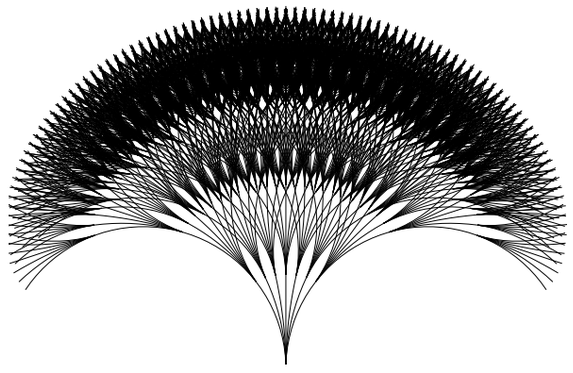


Find a sparse subset $S \subset D$

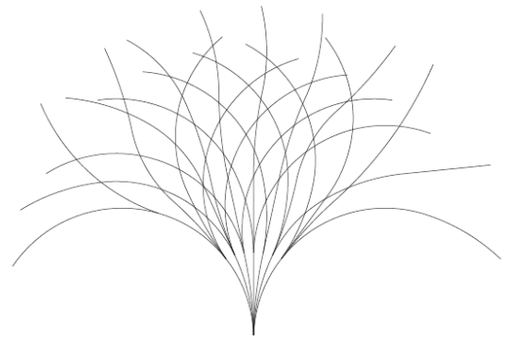


Proposed problem formulation

Given a dense path set D



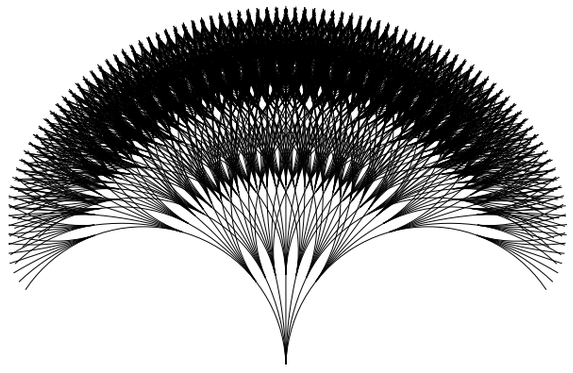
Find a sparse subset $S \subset D$



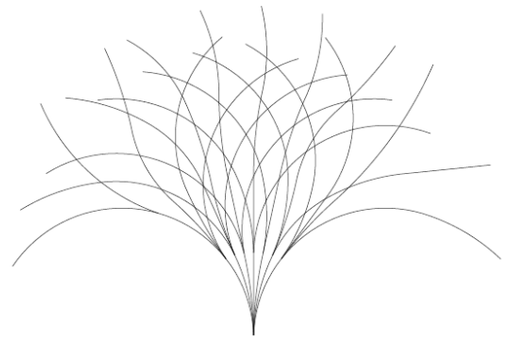
$$\arg\max_{S \subset D} \textsc{Coverage}(S, D)$$

Proposed problem formulation

Given a dense path set D



Find a sparse subset $S \subset D$



$$\arg\max_{S \subset D} \textsc{Coverage}(S, D)$$

What is coverage? Why do we want coverage?

Why do we want coverage?

Why do we want coverage?

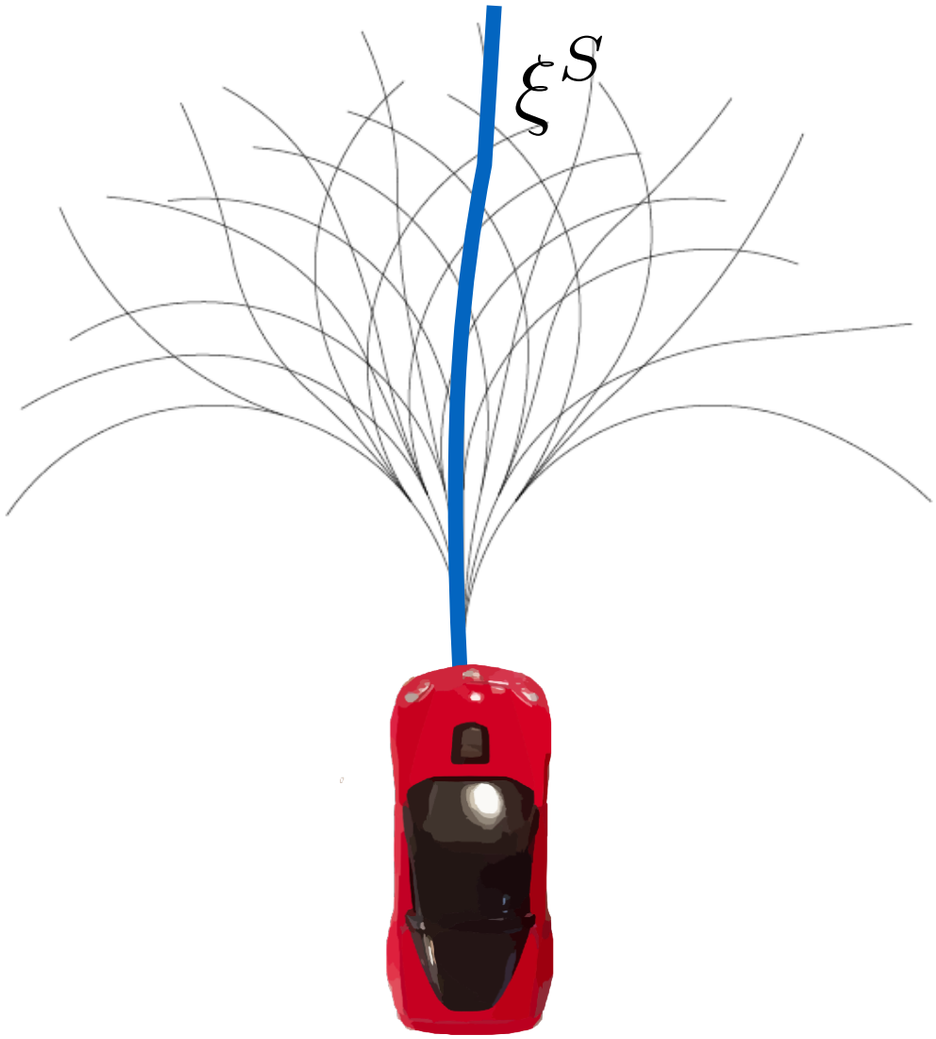
Let's say the dense set has an optimal path $\xi^D = \arg\min_{\xi \in D} J(\xi)$

Why do we want coverage?

Let's say the dense set has an optimal path $\xi^D = \arg\min_{\xi \in D} J(\xi)$

We want to make sure the sparse set has a path ξ^S "close to" ξ^D

such that $J(\xi^S) \leq J(\xi^D) + \epsilon$



Lipschitz continuity

Lipschitz continuity

A Lipschitz continuous function satisfies the following inequality

$$|f(x) - f(y)| \leq L||x - y||$$

(Close points have the same function value)

Lipschitz continuity

A Lipschitz continuous function satisfies the following inequality

$$|f(x) - f(y)| \leq L||x - y||$$

(Close points have the same function value)

The cost functions we consider are Lipschitz continuous

$$|J(\xi_1) - J(\xi_2)| \leq Ld(\xi_1, \xi_2)$$

(Close trajectories have the same cost value)

Defining a "closeness" between paths

$$d(\xi_1, \xi_2)$$

Path 1 Path2

Defining a "closeness" between paths

$$d(\xi_1, \xi_2)$$

Path 1 Path2

We will choose a metric in the space of paths

Defining a "closeness" between paths

$$d(\xi_1, \xi_2)$$

Path 1 Path2

We will choose a metric in the space of paths

1. Non-negative

Defining a "closeness" between paths

$$d(\xi_1, \xi_2)$$

Path 1 Path2

We will choose a metric in the space of paths

1. Non-negative

2. Triangle inequality

Defining a "closeness" between paths

$$d(\xi_1, \xi_2)$$

Path 1 Path2

We will choose a metric in the space of paths

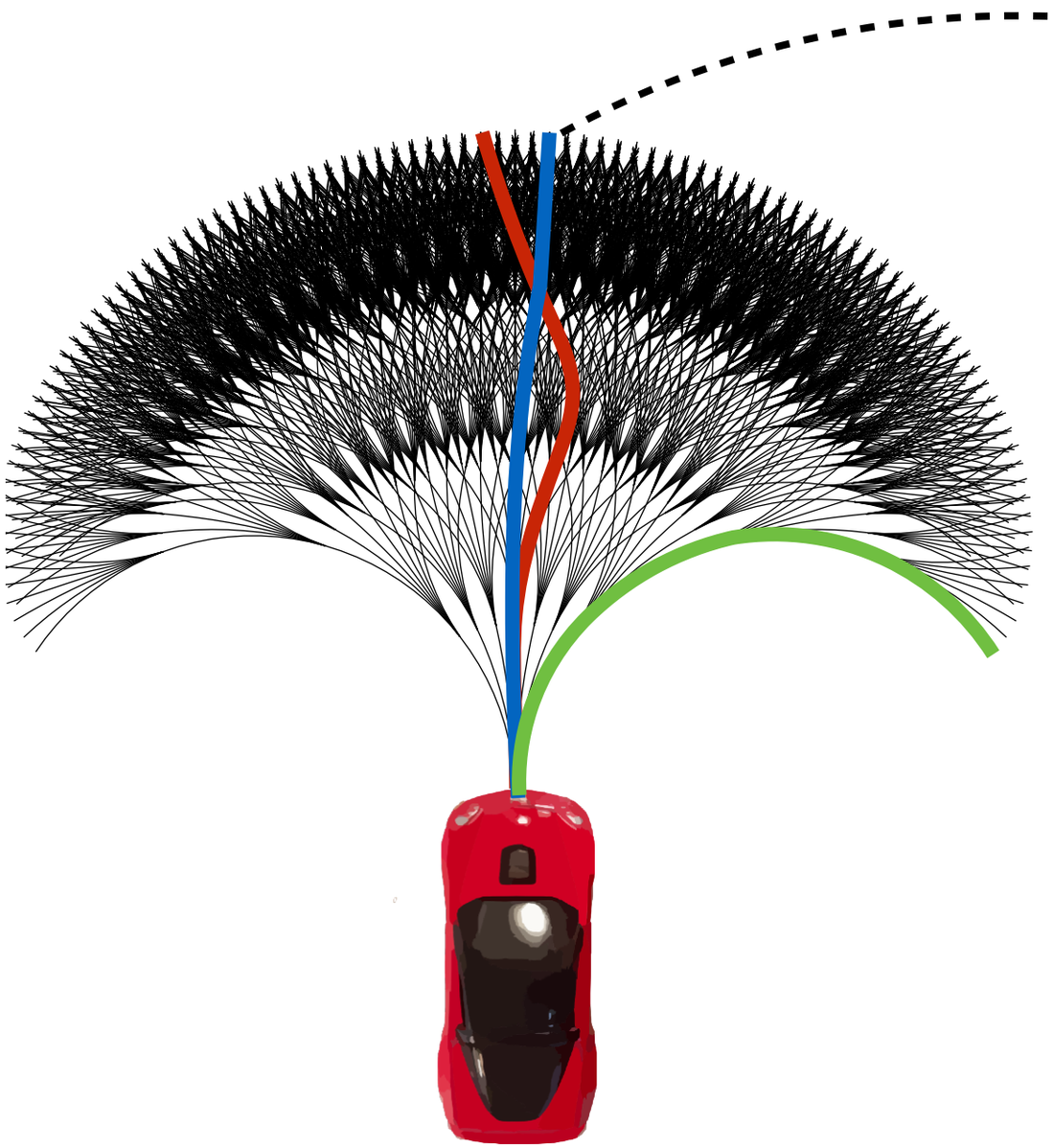
1. Non-negative

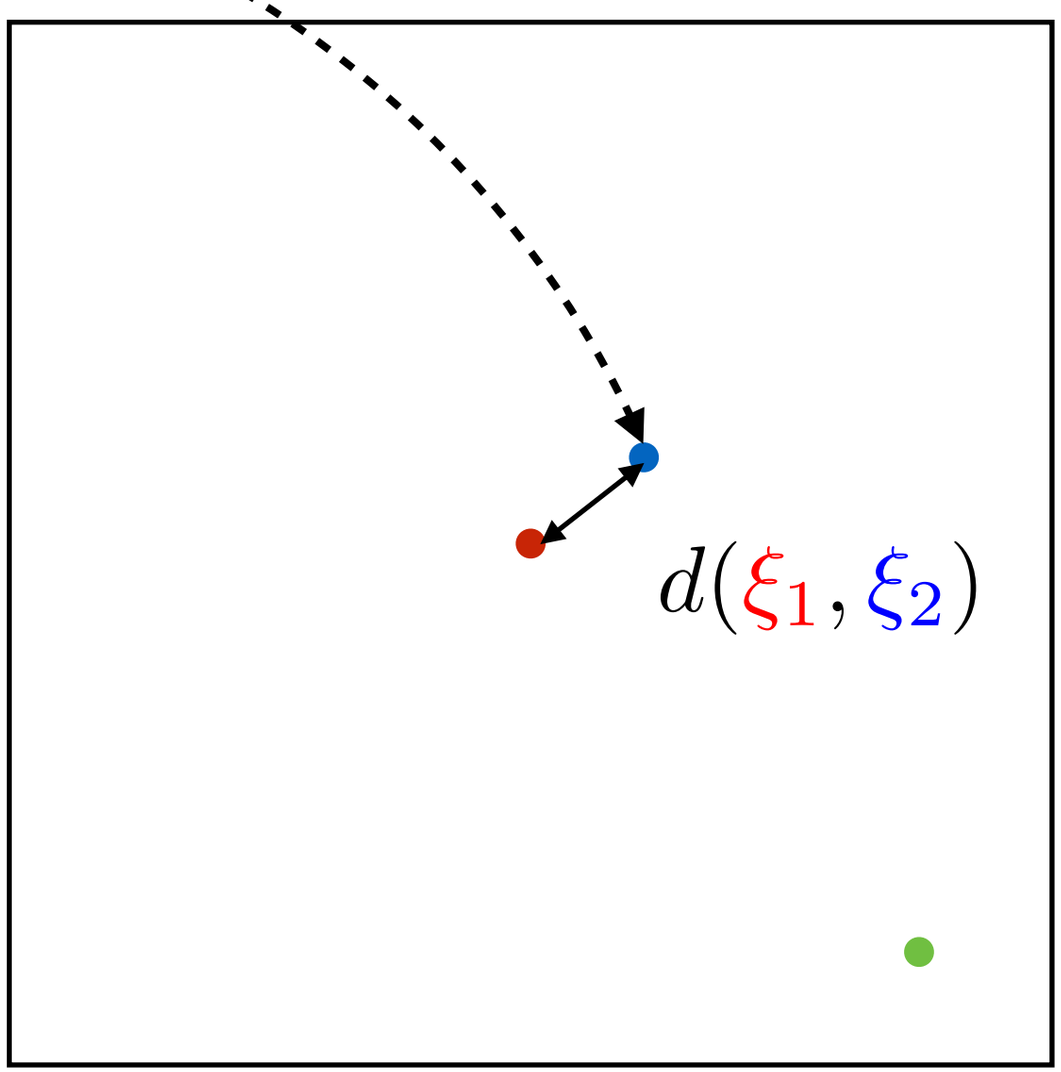
2. Triangle inequality

There are plenty of metrics to choose from!

Defining a "closeness" between paths

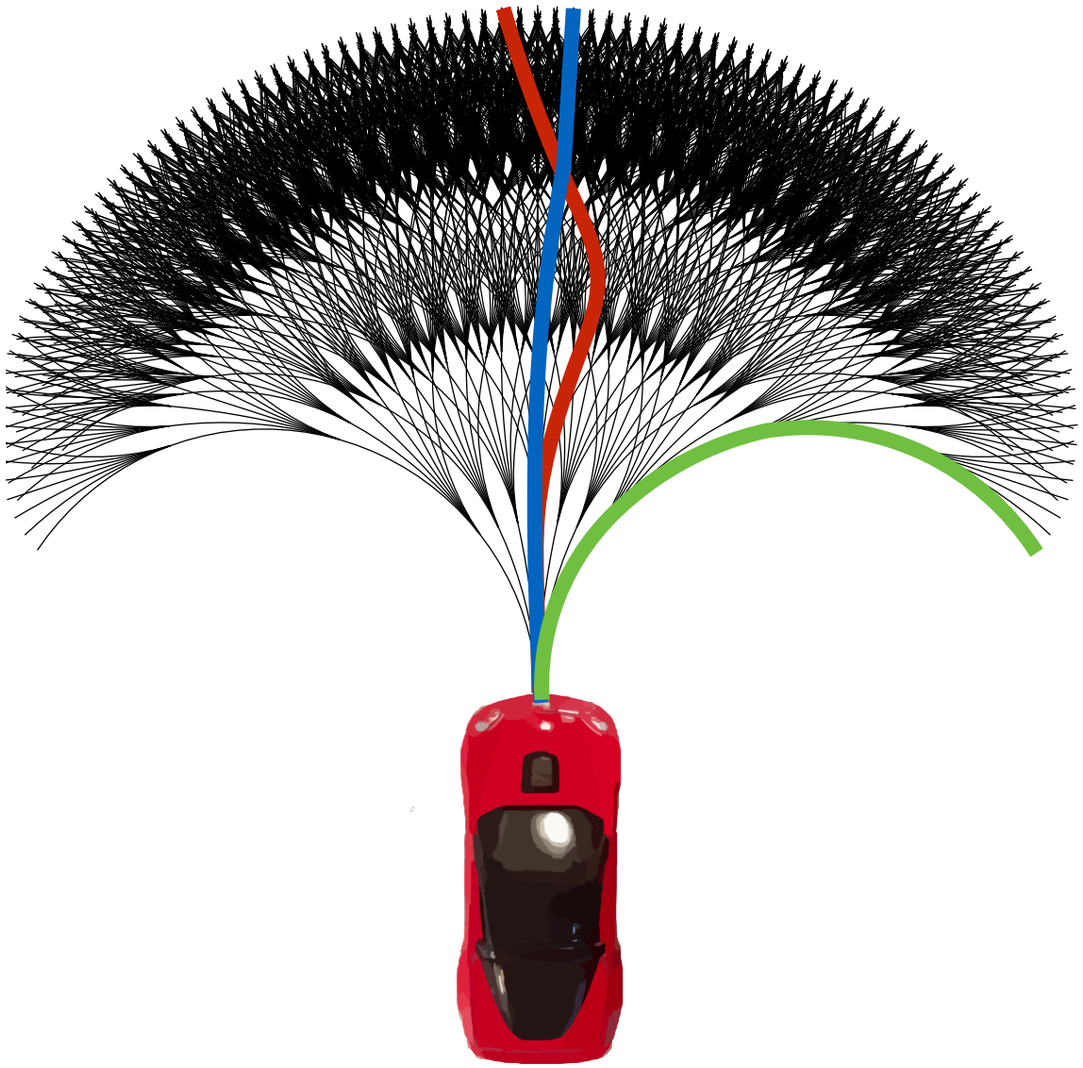
Project paths on to a metric space where each dot is a path

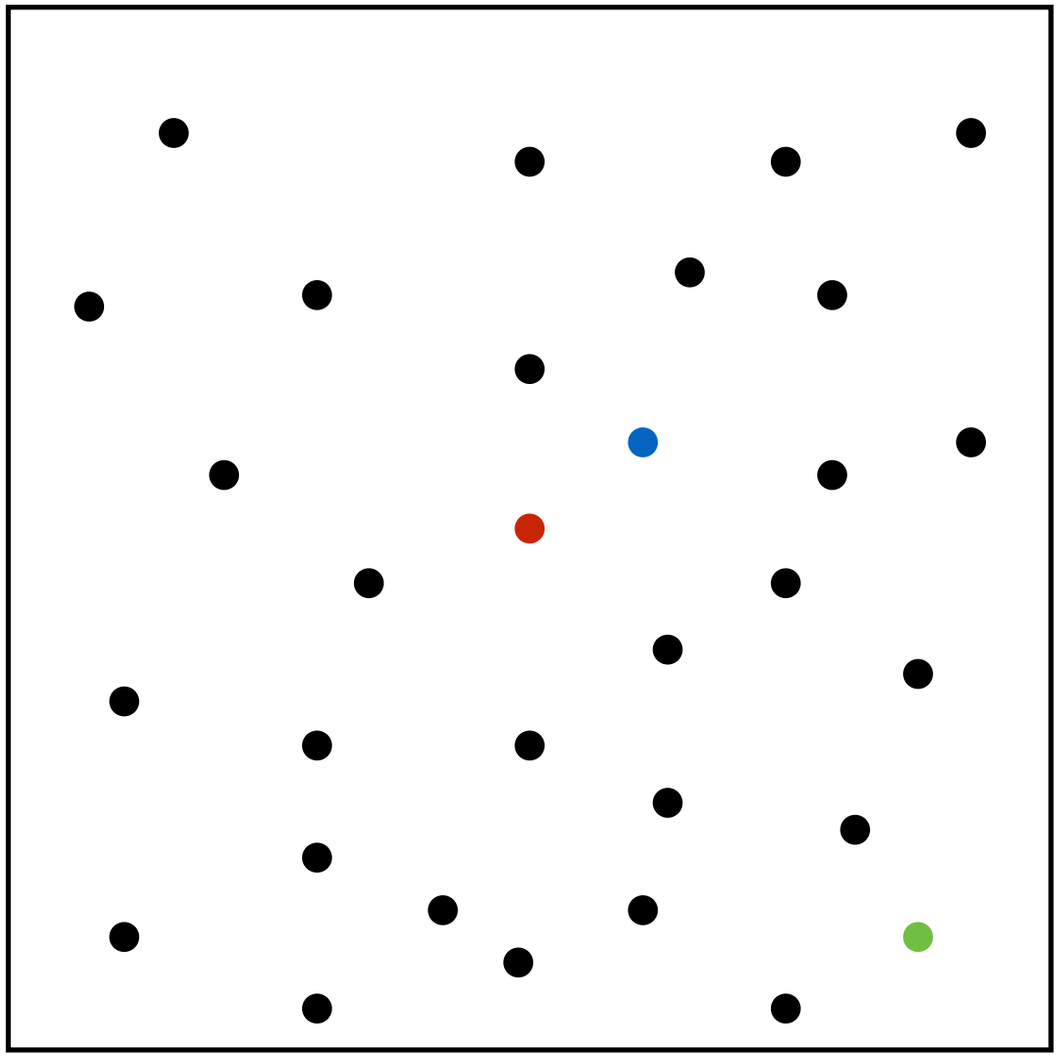




Defining a “closeness” between paths

Every dot is a path - neighboring dots are close



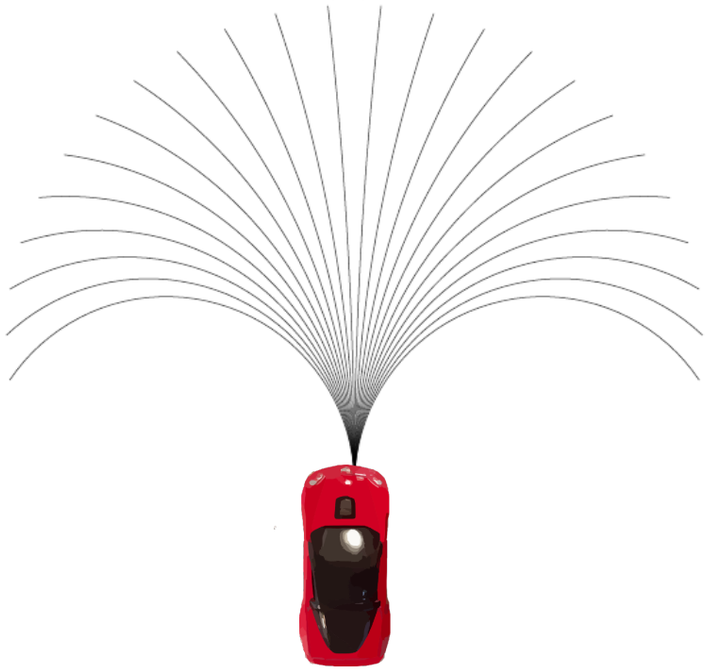


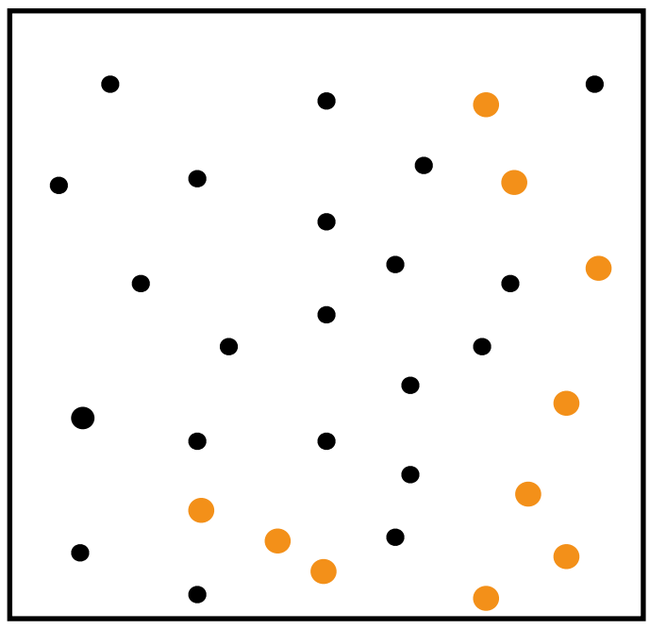
Objective: "Cover" as many black dots

Which library does a better job?

Objective: "Cover" as many black dots

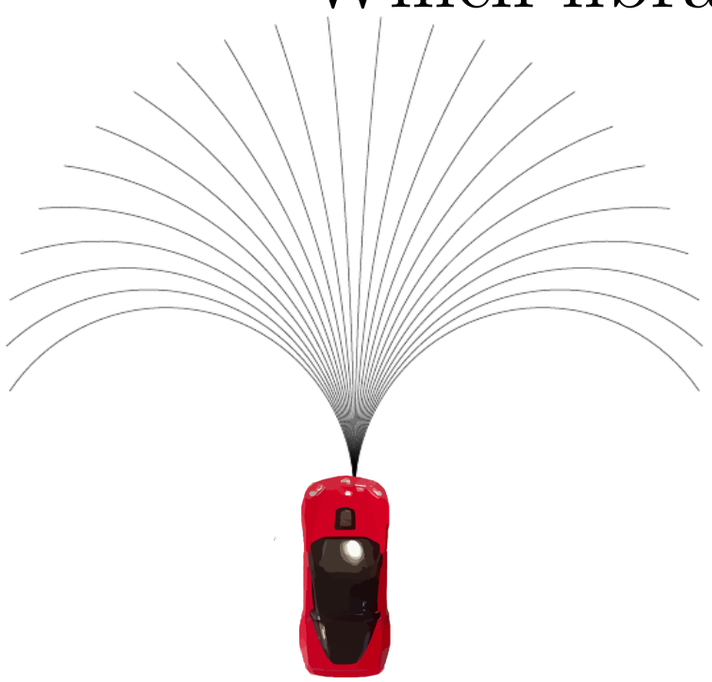
Which library does a better job?

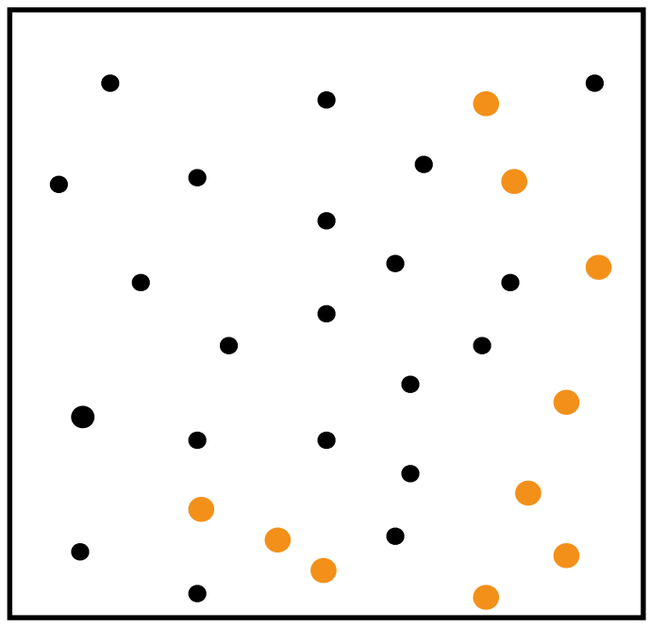


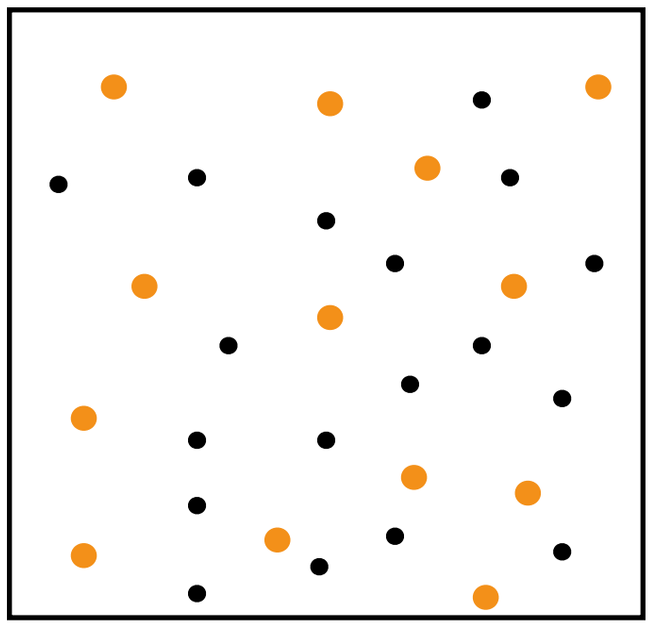


Objective: "Cover" as many black dots

Which library does a better job?







Formalizing coverage as dispersion

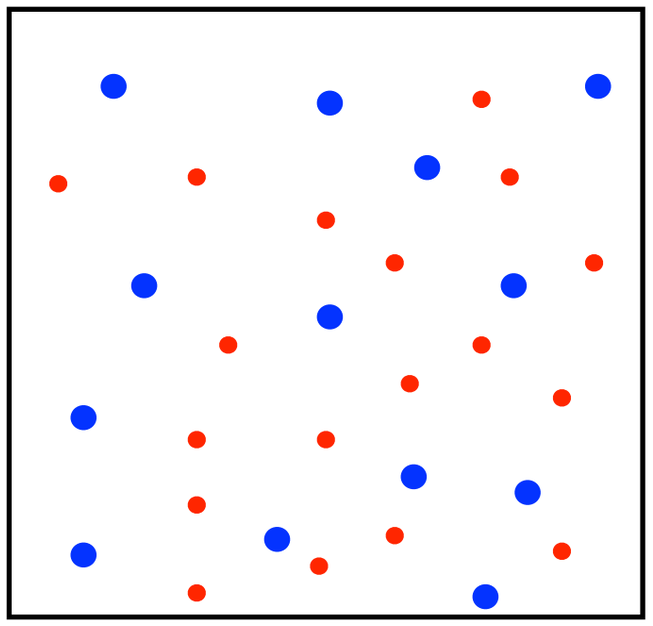
Formalizing coverage as dispersion

Now that we have projected everything to a metric space,
we can bring in statistical tools such as dispersion

Formalizing coverage as dispersion

Now that we have projected everything to a metric space,
we can bring in statistical tools such as dispersion

$$\textsc{Coverage}(S, D) \equiv -\max_{\xi_1 \in D} \min_{\xi_2 \in S} d(\xi_1, \xi_2)$$

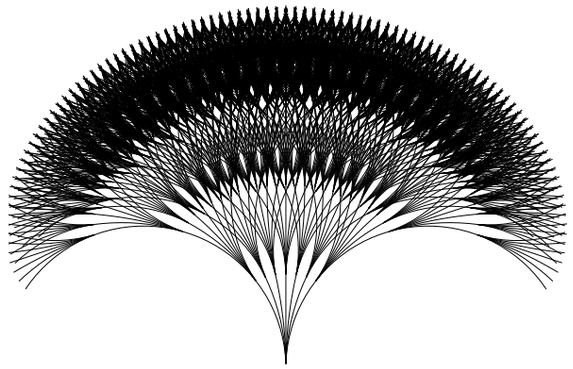


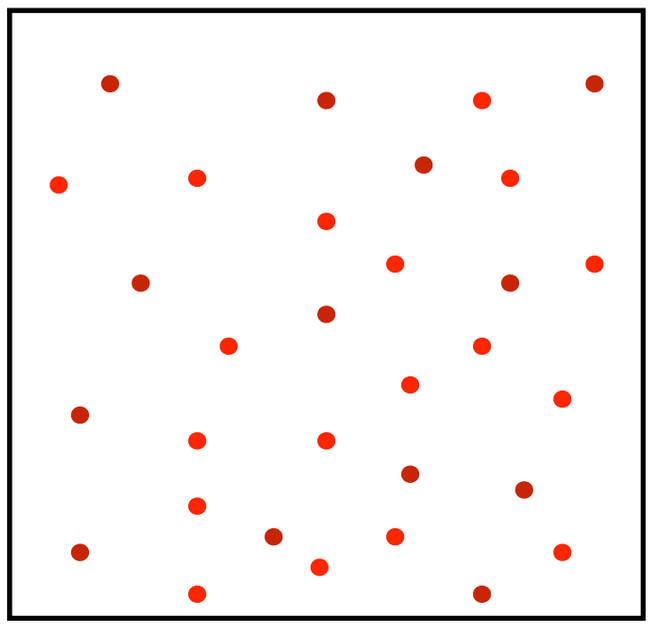
Dispersion is the radius of the largest ball
around a point in D that does not have a point in S

Low dispersion = Good coverage

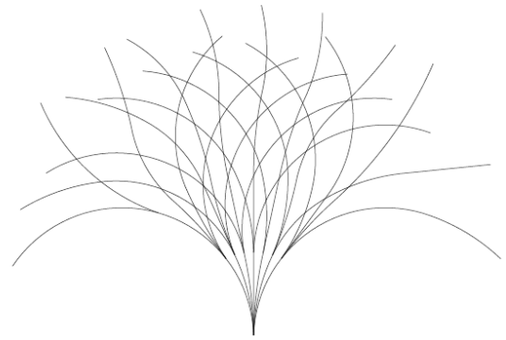
Problem: Minimize dispersion

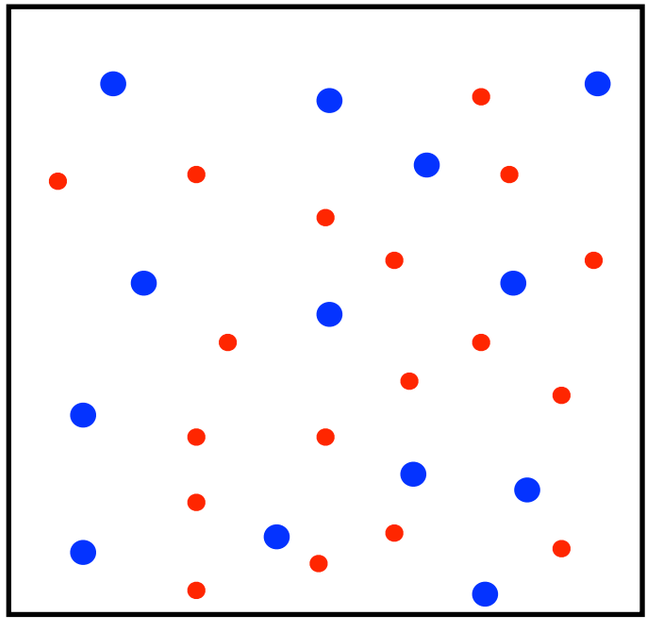
Given a dense path set D





Find a sparse subset $S \subset D$





$$\underset{S \subset D}{\arg\min} \left[\max_{\xi_1 \in D} \min_{\xi_2 \in S} d(\xi_1, \xi_2) \right]$$

Questions

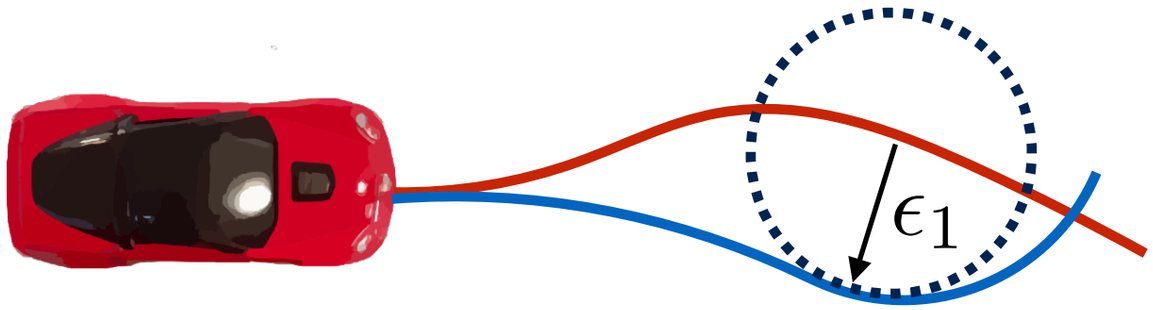
1. What distance metric should we use?

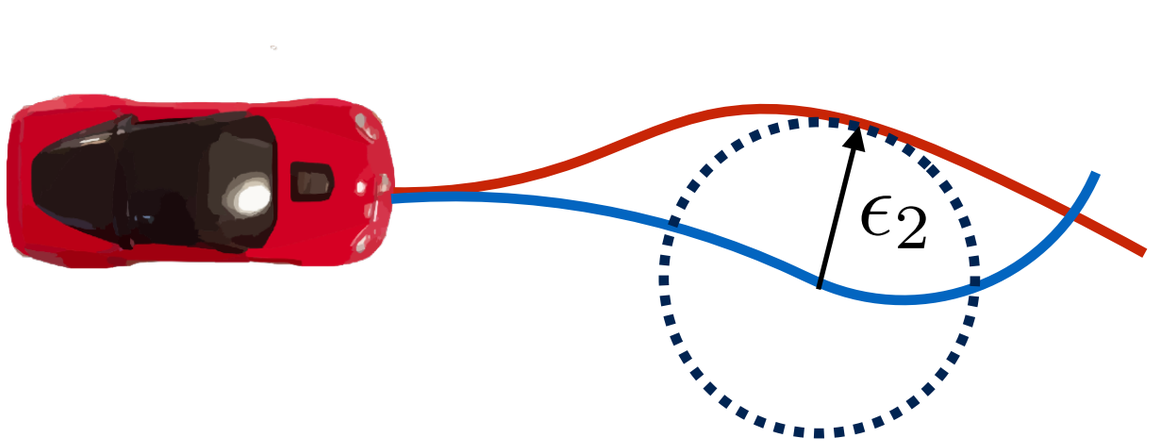
2. How do we solve the combinatorially hard problem of dispersion minimization?

Choosing a metric: Hausdorff distance

$$d(\xi_1, \xi_2) = \max(\max_{p \in \xi_1} \min_{q \in \xi_2} ||p - q||,$$

$$\max_{p \in \xi_2} \min_{q \in \xi_1} ||p - q||)$$





$$\max(\epsilon_1, \epsilon_2)$$

Minimizing dispersion: Be greedy!

Greedily add a path to S that minimizes dispersion the most

Minimizing dispersion: Be greedy!

Greedily add a path to S that minimizes dispersion the most

Why?

Has a similar flavor to set-cover problem (where greedy is near-optimal)

Minimizing dispersion: Be greedy!

Greedily add a path to S that minimizes dispersion the most

Why?

Has a similar flavor to set-cover problem (where greedy is near-optimal)

1. Dispersion decreases monotonically
2. Dispersion tends to have diminishing returns*

*not always though

Algorithm

1. Generate a dense set of paths D

2. Initialize S with a default path (straight line)

3. While $|S| <$ budget

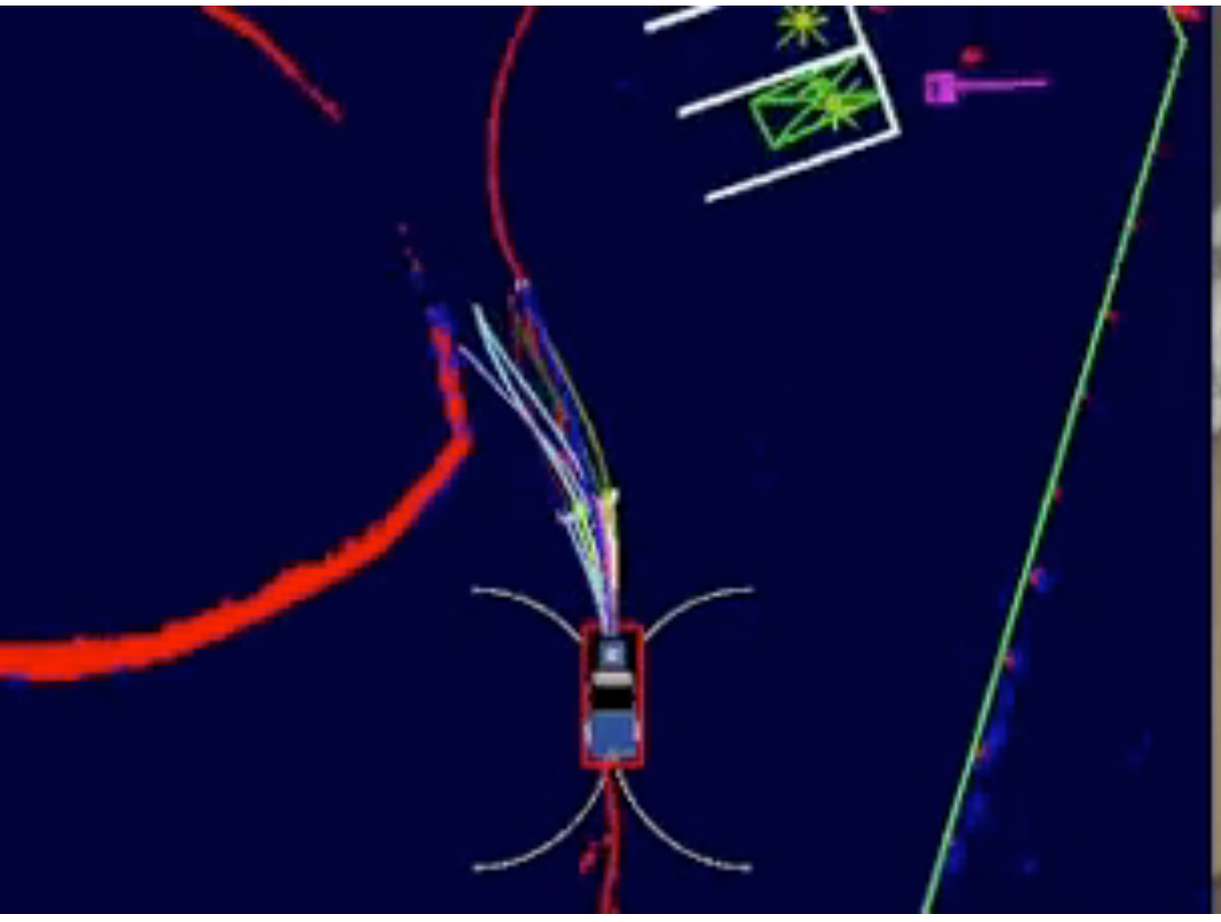
4. Find path in D with maximum dispersion

$$\xi = \arg\max_{\xi_1 \in D} \min_{\xi_2 \in S} d(\xi_1, \xi_2)$$

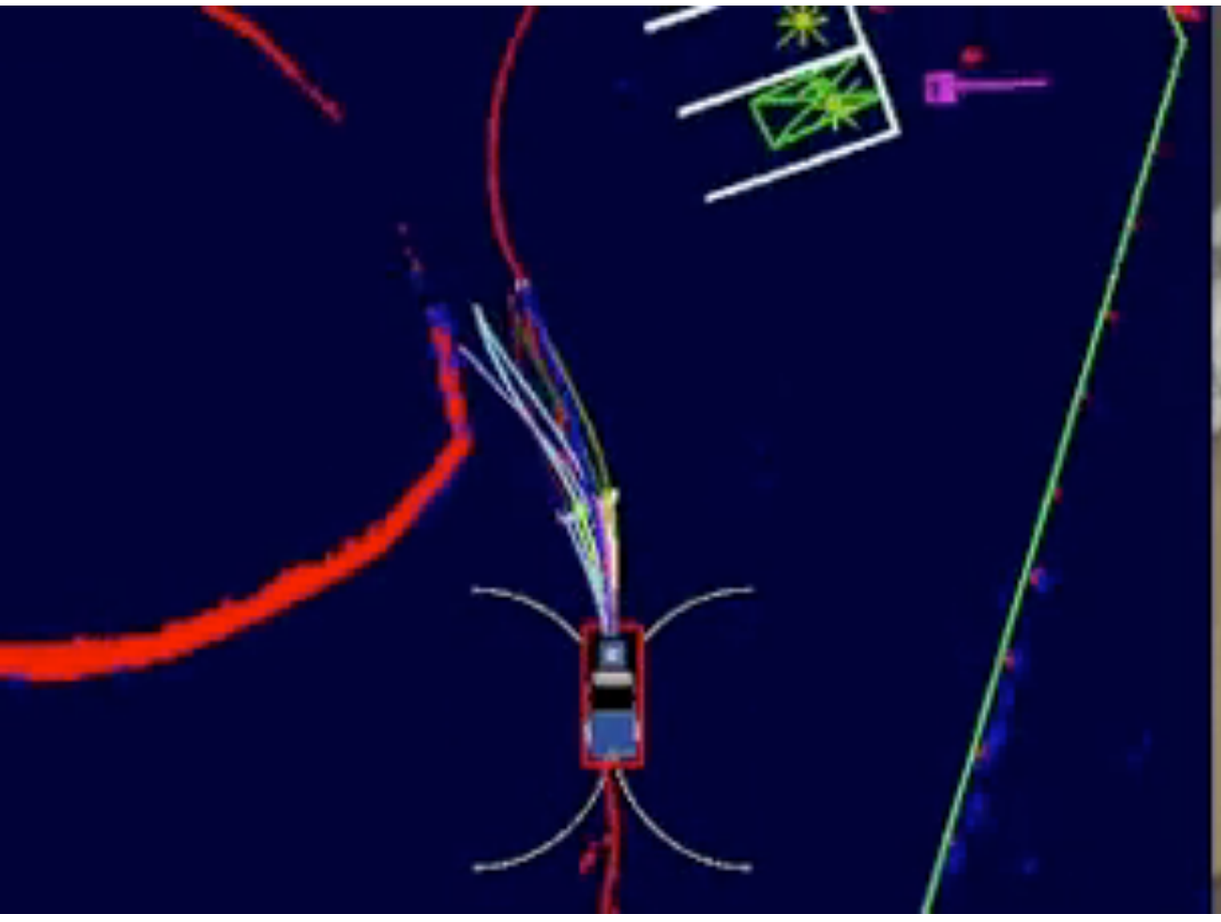
5. Append path to set $S \leftarrow S \cup \{\xi\}$

6. Return path set S

Trajectory library in action

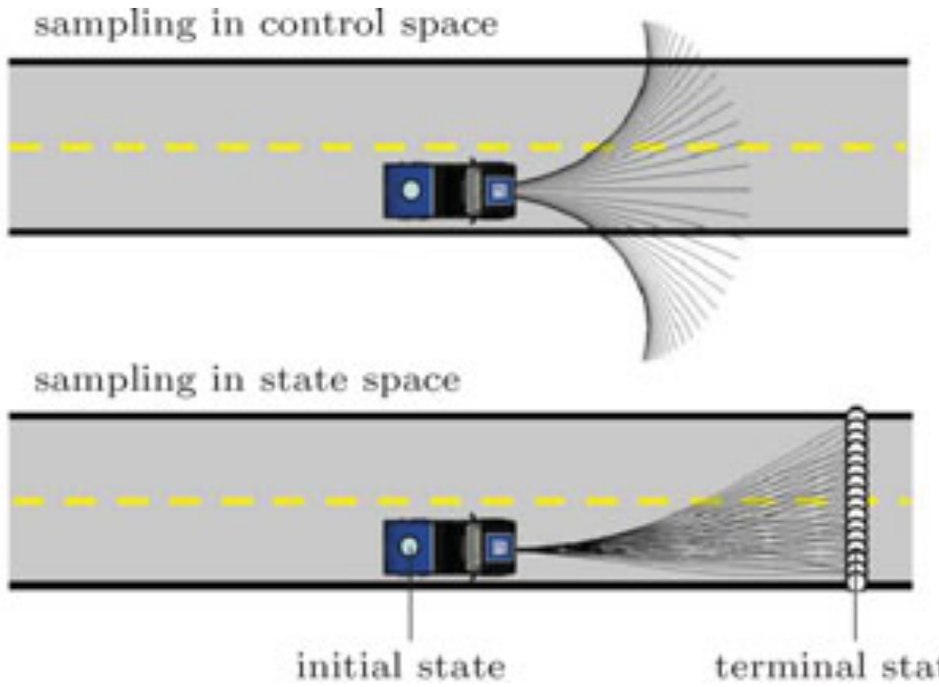


Trajectory library in action

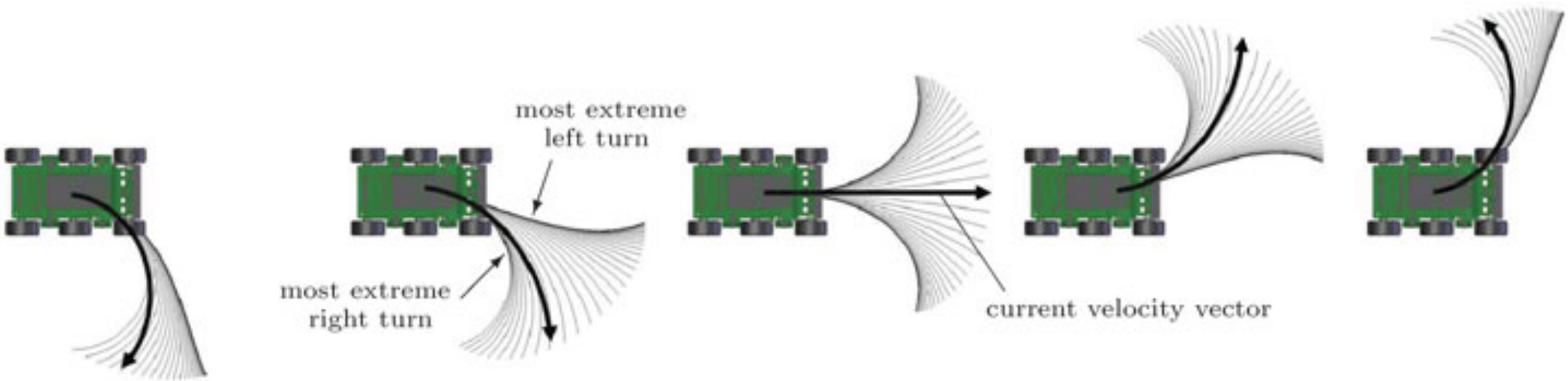


Extra Credits

What is wrong with control space?



Uniform sampling in control space is non-uniform in state space.



T.Howard "State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments" JFR, 2008

Why should we care about state space?

$$u(t) \longrightarrow \int_0^t f(x(t), u(t))dt \longrightarrow x(t)$$

Control trajectory

Dynamics

State trajectory

Space in which we sample

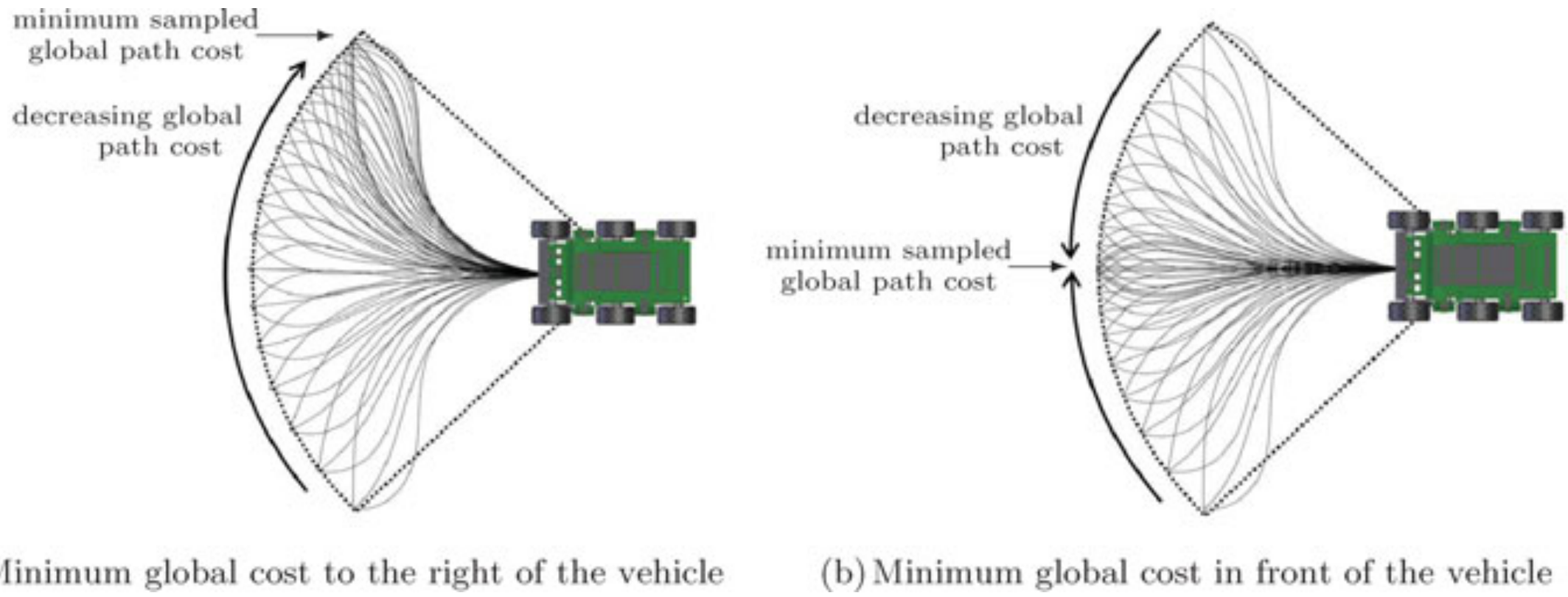
Space in which cost functions defined

State space sampling

Do optimization to different terminal points

Store these trajectories

Load path set dependent on the global cost



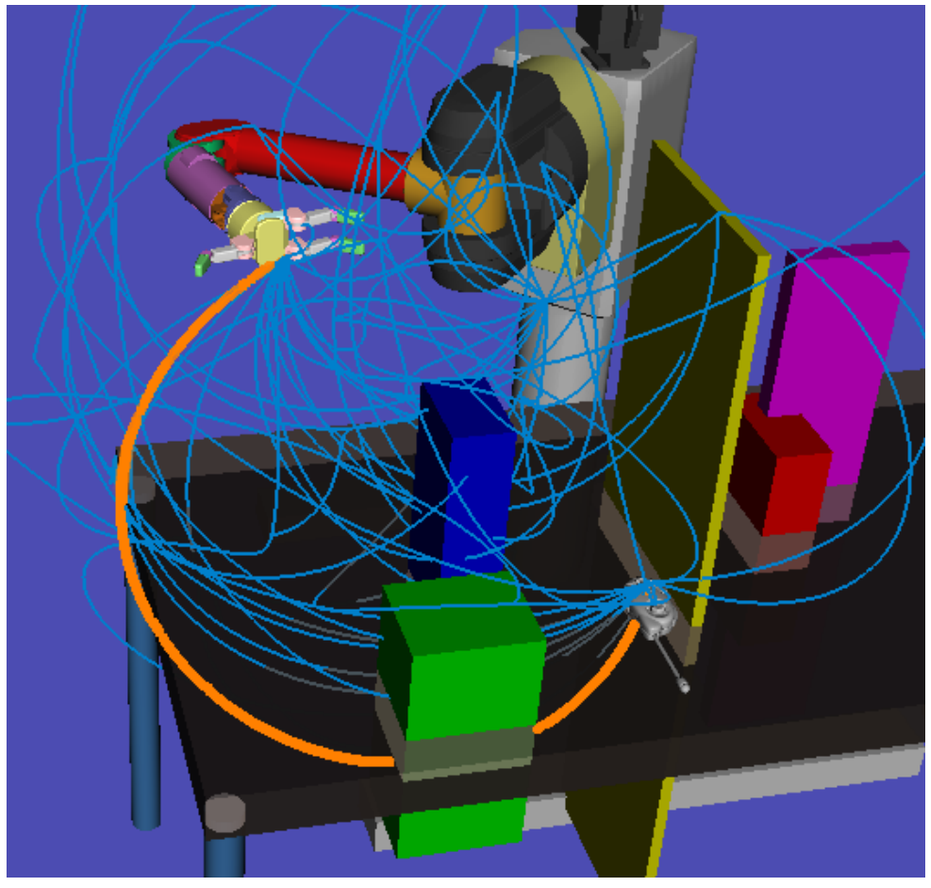
(a) Minimum global cost to the right of the vehicle

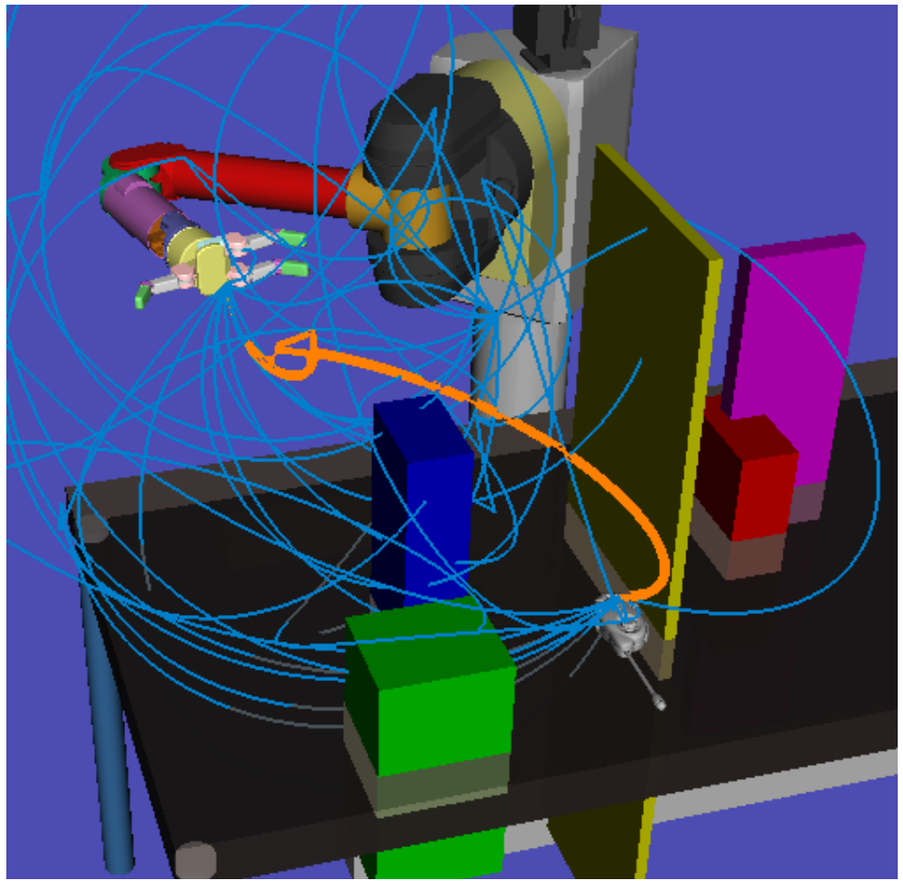
(b) Minimum global cost in front of the vehicle

T.Howard "State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments" JFR, 2008

Using learning to select libraries

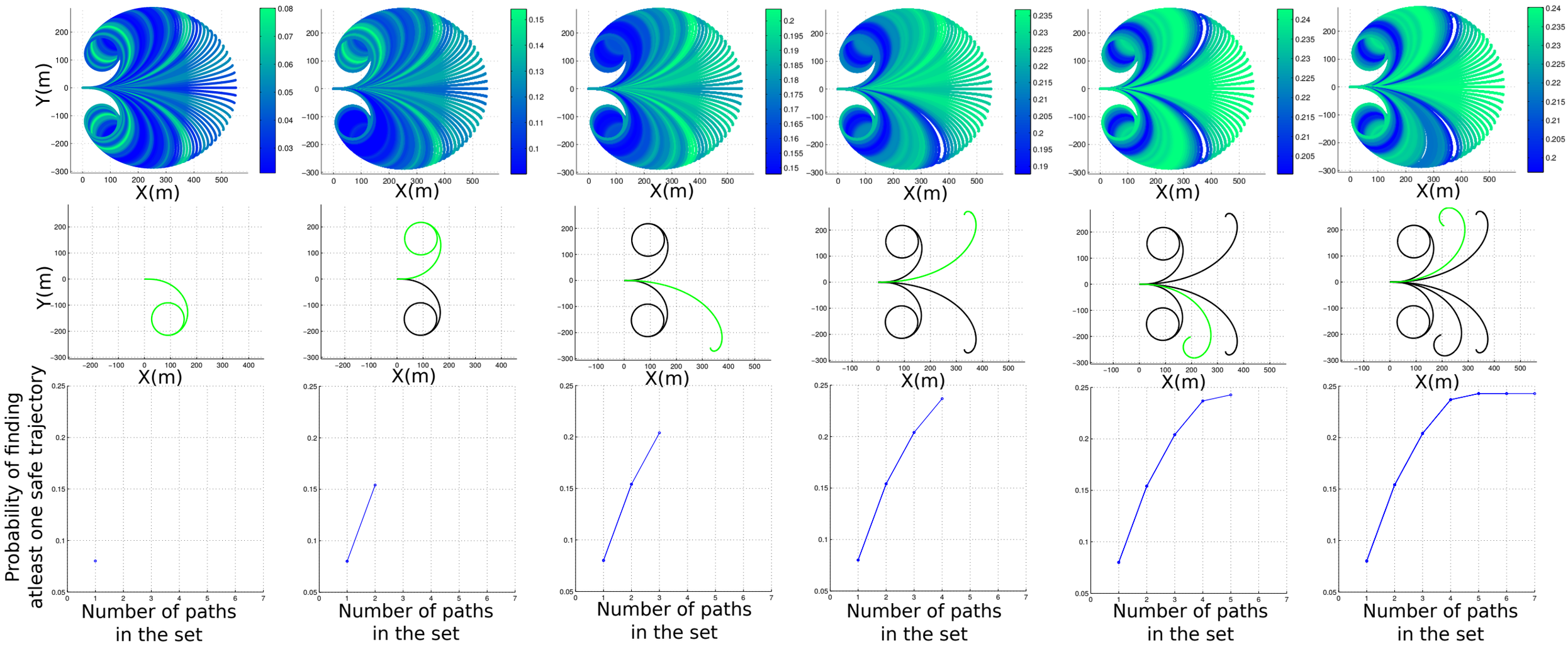
Instead of selecting a static library, we can select a contextual one





D.Dey et al “Contextual Sequence Prediction with Application to Control Library Optimization”, 2013

Guaranteeing safety with libraries



S.Arora “Emergency Maneuver Library – Ensuring Safe Navigation in Partially Known Environments”, 2015