Comparison of two randomized motion planning algorithms

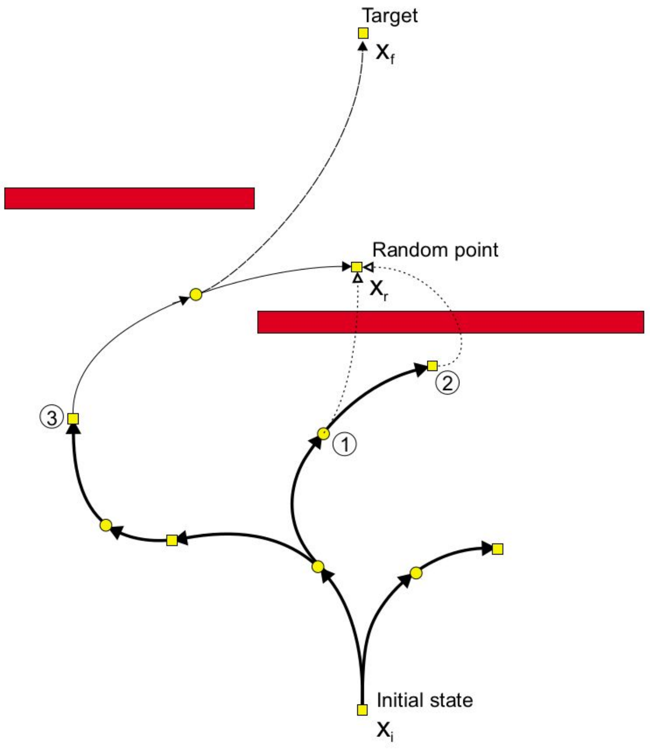
RRT*

Real-time motion planning based on optimal policy

Randomized planning with pre-computed optimal policies

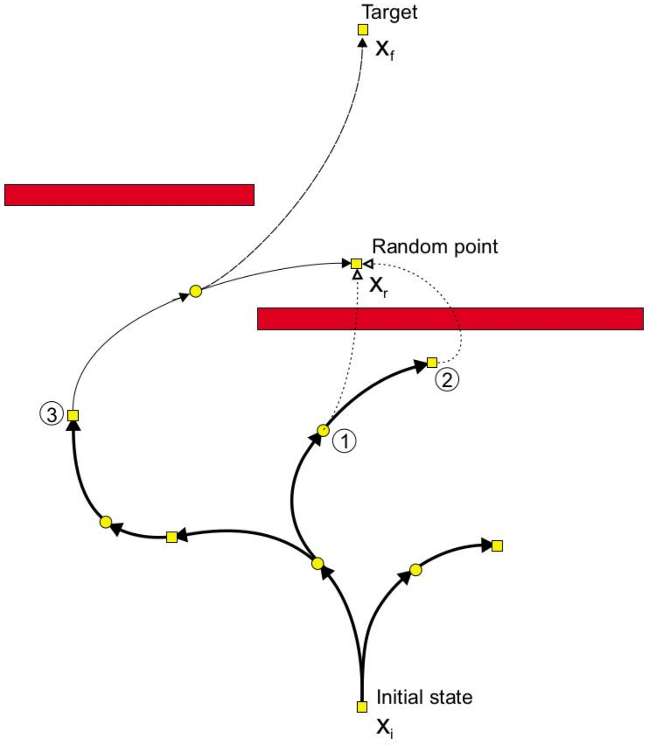
Randomized planning with pre-computed optimal policies

Pre compute optimal cost to go from all states in obstacle free environment with value iteration



Randomized planning with pre-computed optimal policies

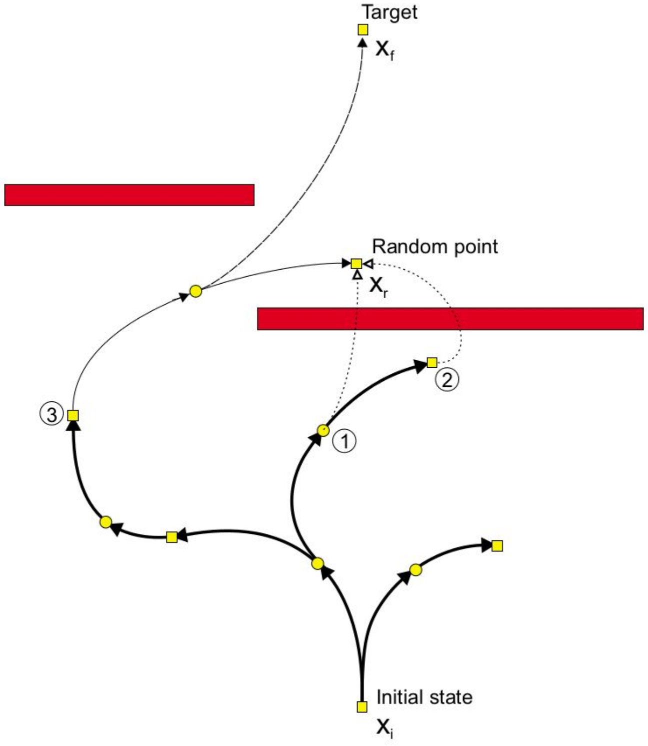
1) Check if optimal path to goal is collision free



Randomized planning with pre-computed optimal policies

1) Check if optimal path to goal is collision free
2) If not:

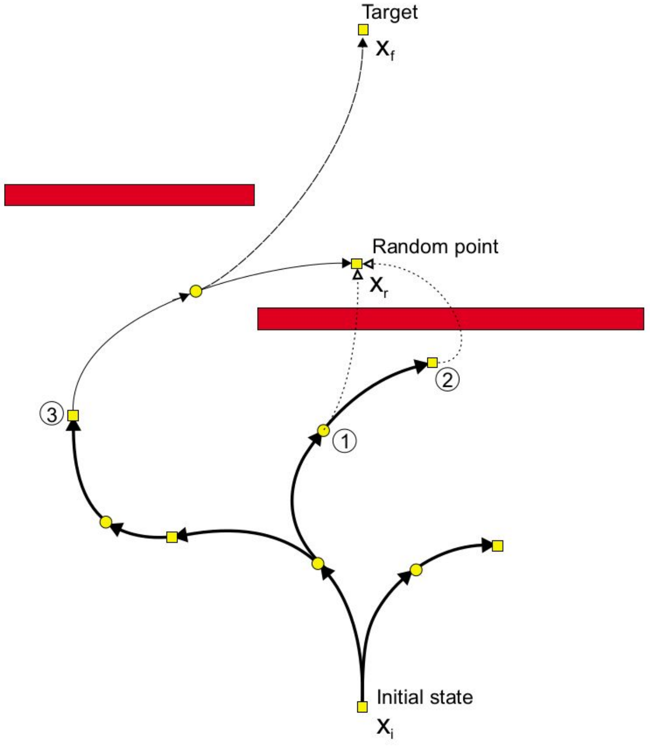
   Loop for n iterations:



Randomized planning with pre-computed optimal policies

1) Check if optimal path to goal is collision free
2) If not:

   Loop for n iterations:
3) Expand tree with random point.

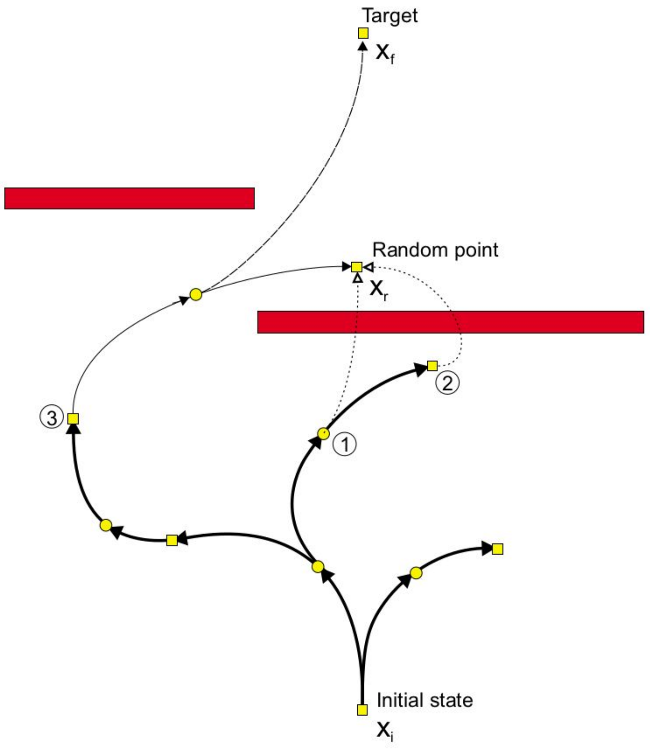


Randomized planning with pre-computed optimal policies

1) Check if optimal path to goal is collision free
2) If not:

   Loop for n iterations:

3) Expand tree with random point.
4) Try to connect it to nodes in tree with precomputed optimal path

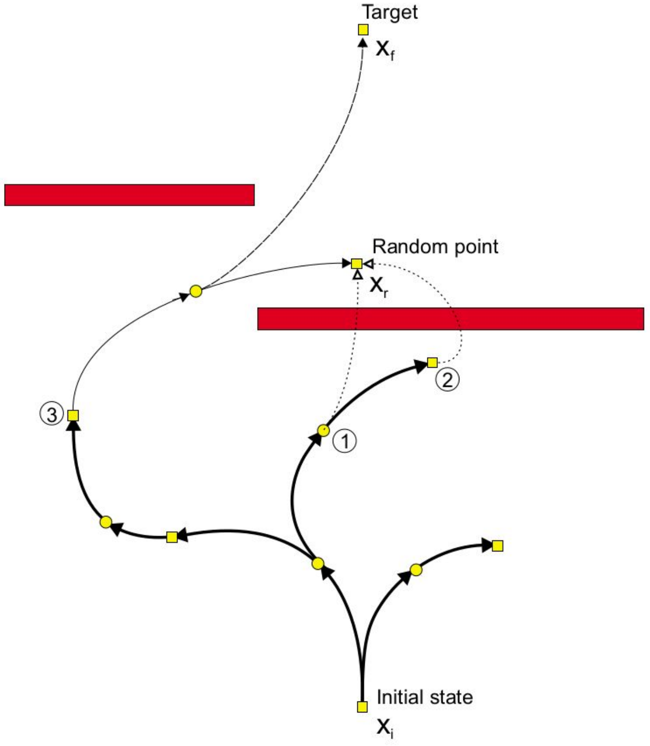


Randomized planning with pre-computed optimal policies

1) Check if optimal path to goal is collision free
2) If not:

   Loop for n iterations:

3) Expand tree with random point.
4) Try to connect it to nodes in tree with precomputed optimal path
5) If the trajectory is collision free:
   - update the cost estimates of the tree

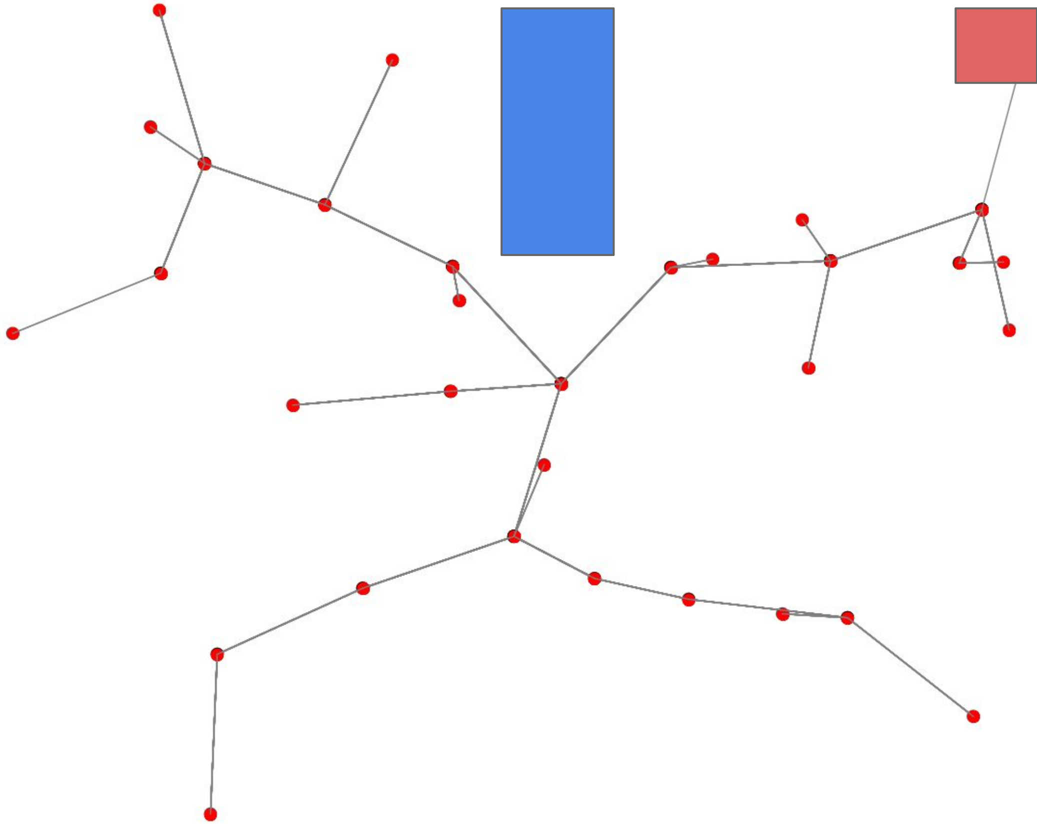


Rapidly-exploring random tree

Rapidly-exploring random tree

RRT

1) Start tree from root
2) For n iterations:
    - Sample random node
3) Steer towards and check for obstacles - add to tree
4) Always add sampled node to nearest neighbor
5) End when:
    - Goal is reached
    - Max iterations



Rapidly-exploring random tree

RRT*

Improve path by

a) Connecting according to cheapest path, not nearest node
b) Rewire path

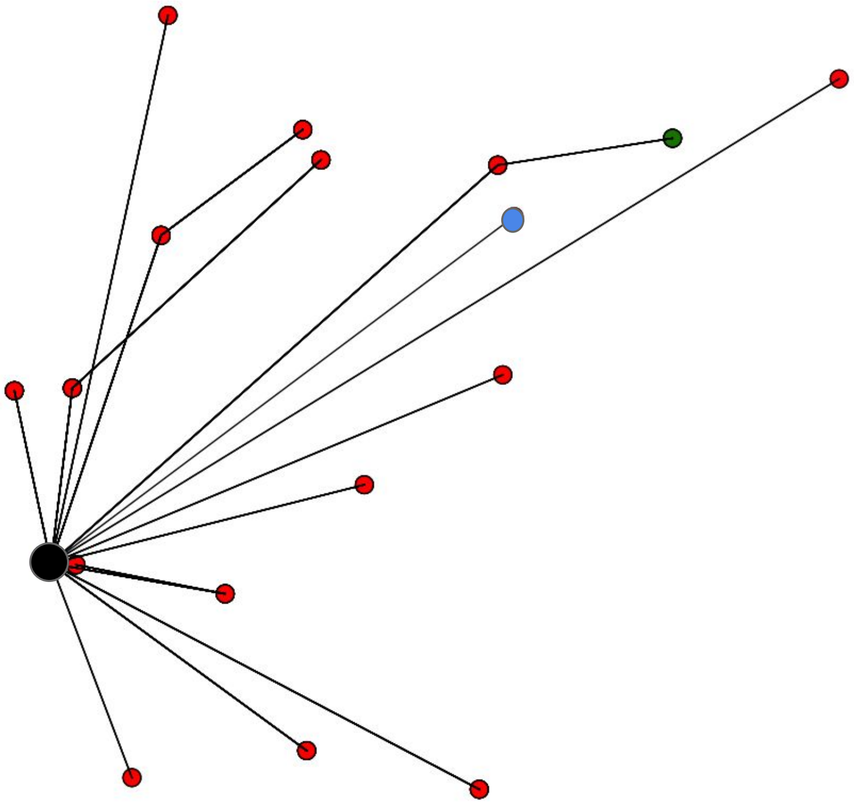
Rapidly-exploring random tree

RRT*

a) Cheapest path

b) Rewire neighbors

- Path to an old node is cheaper through the new neighbor

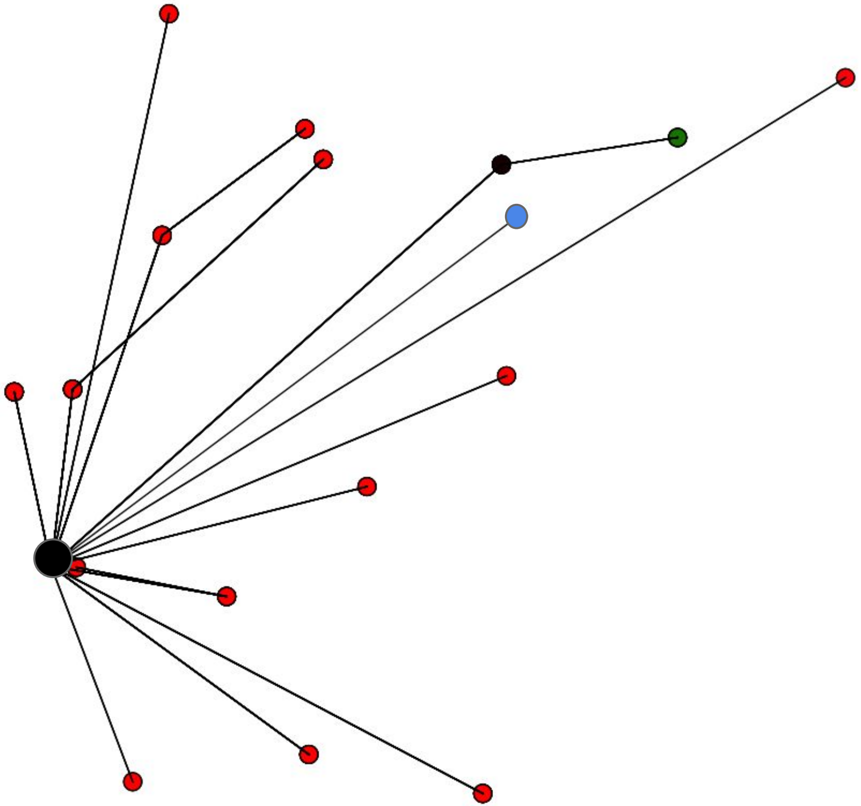


Rapidly-exploring random tree

RRT*

b) Rewire neighbors

- Remove old parent

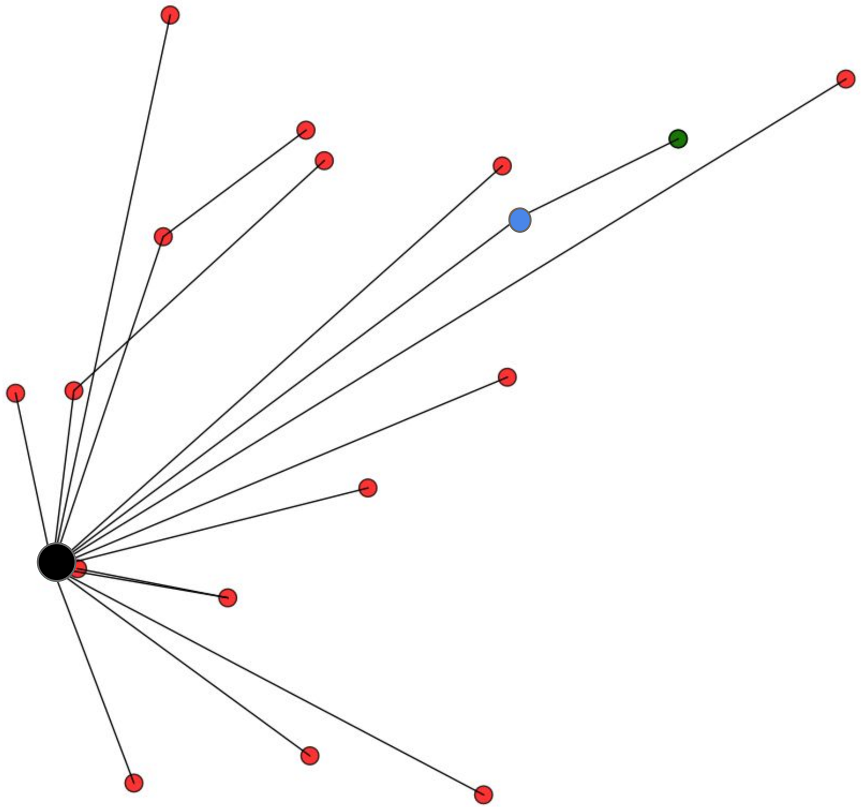


Rapidly-exploring random tree

RRT*

b) Rewire neighbors

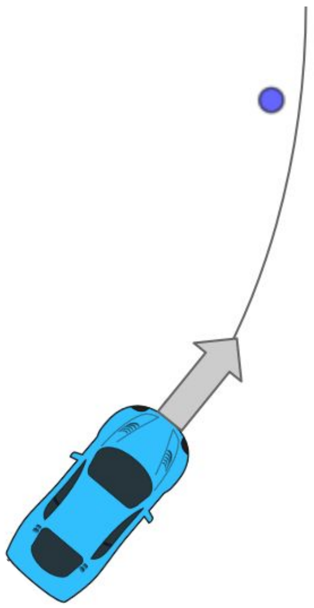
- Add new parent and update cost



RRT*

Non-holonomic constraints

- Always check reachability
- Makes rewiring difficult and **slow**



Rapidly-exploring random tree

RRT*

Improve:

- Reduce nearest neighbors while maintaining optimality
- Practically upper bound

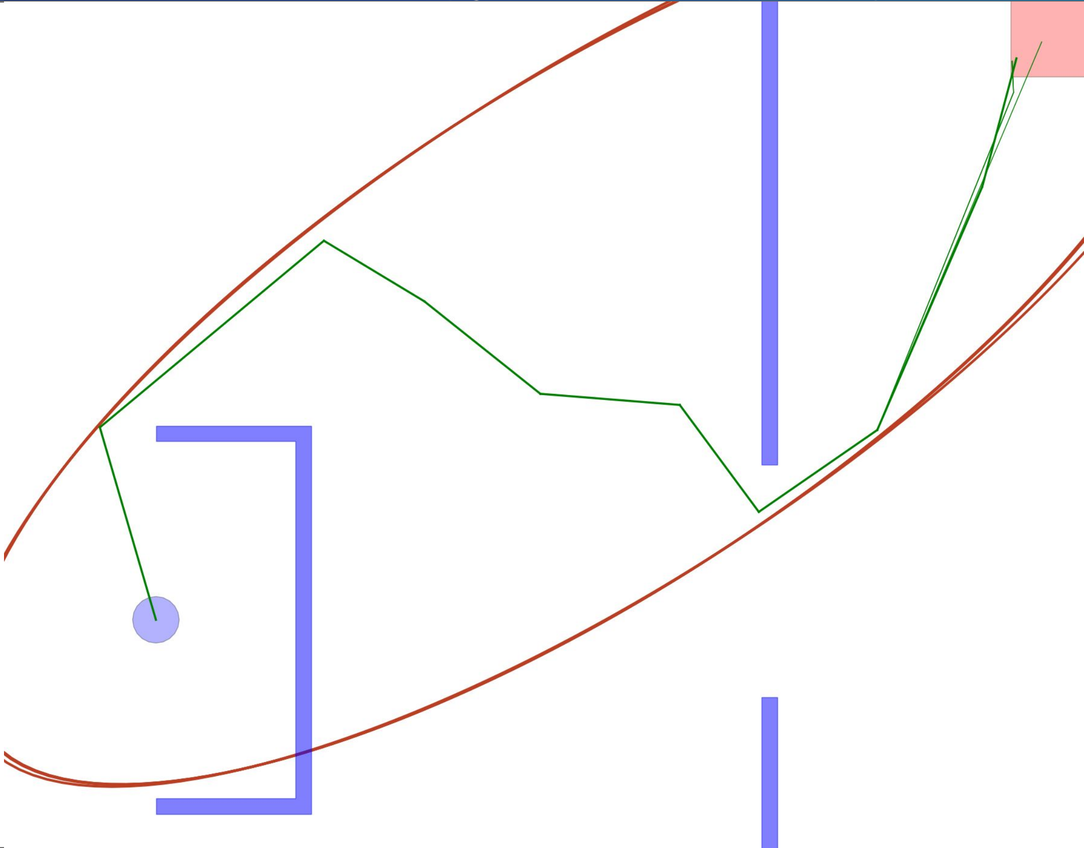
$$||x_{new} - x||_\infty \geq \left(\frac{log(n)}{n}\right)^{\frac{1}{D}}$$

Rapidly-exploring random tree

RRT*

Improve:

- Informed RRT*



Comparison of two randomized motion planning algorithms

RRT*

+ Implementation wise intuitive
+ Effective when modified

- Gets slow, fast

Value iteration

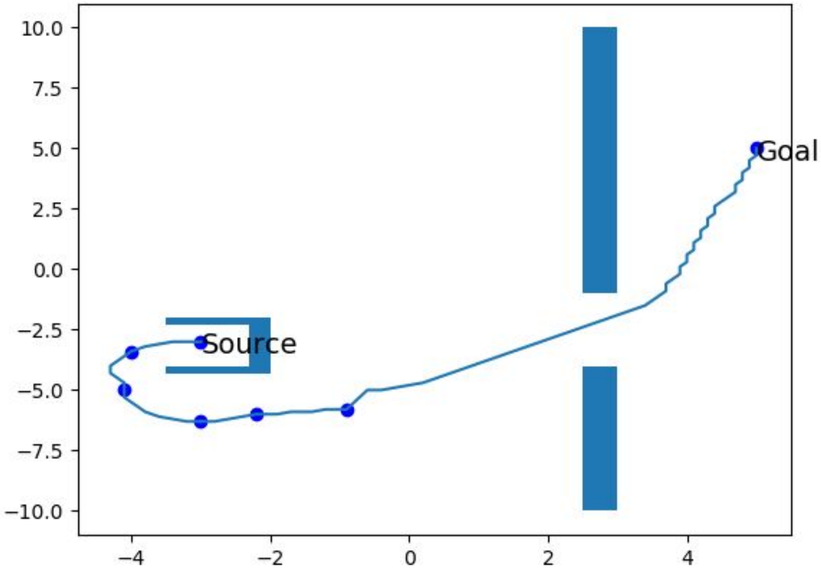
+ Optimal cost
+ Real-time

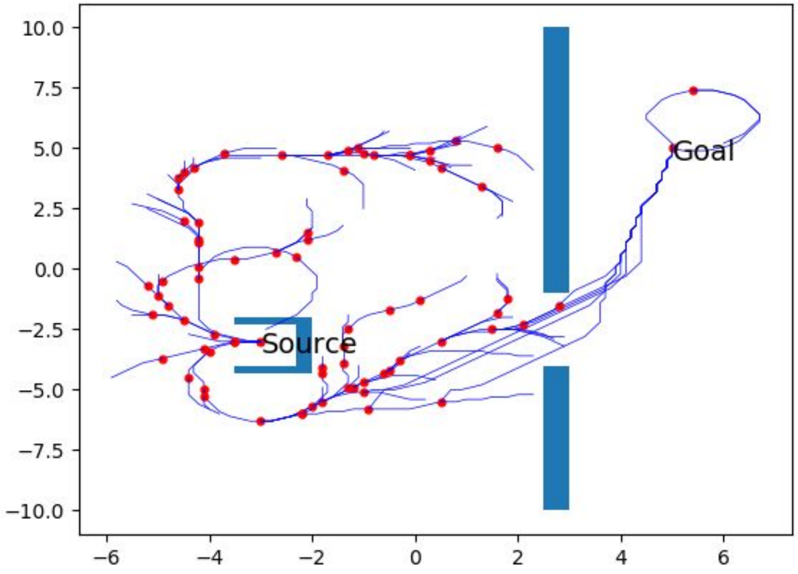
- Discretizing
- Long initialization

Tests

Steer according to simple car model at low, constant, velocity

- 100 iterations
- 1 second

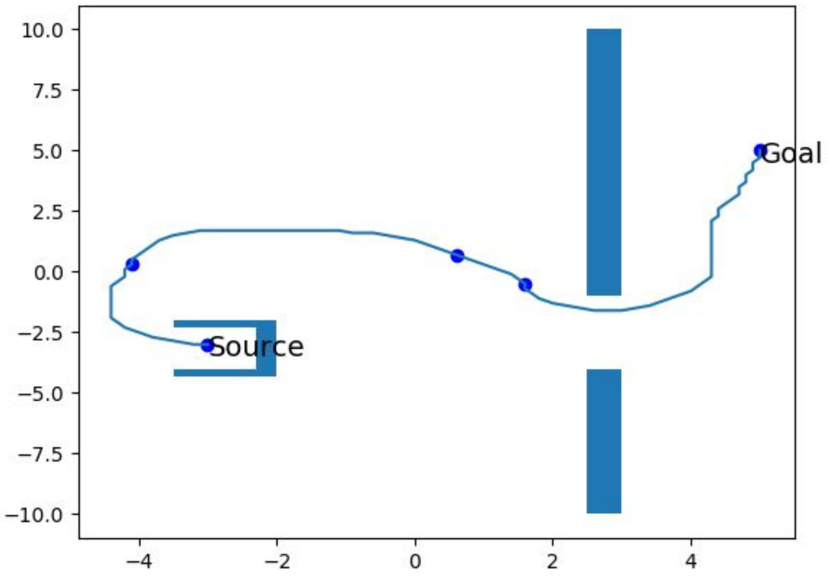


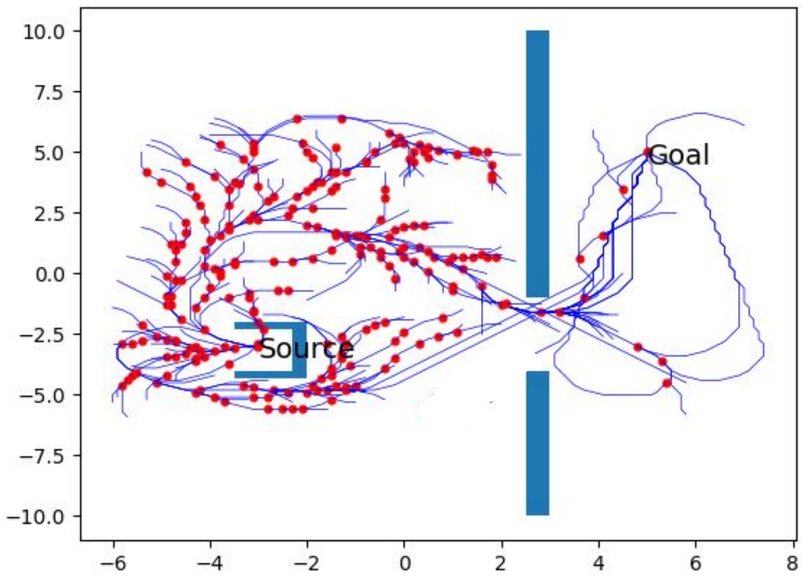


Tests

Steer according to simple car model at low, constant, velocity

- 300 iterations
- 5 seconds

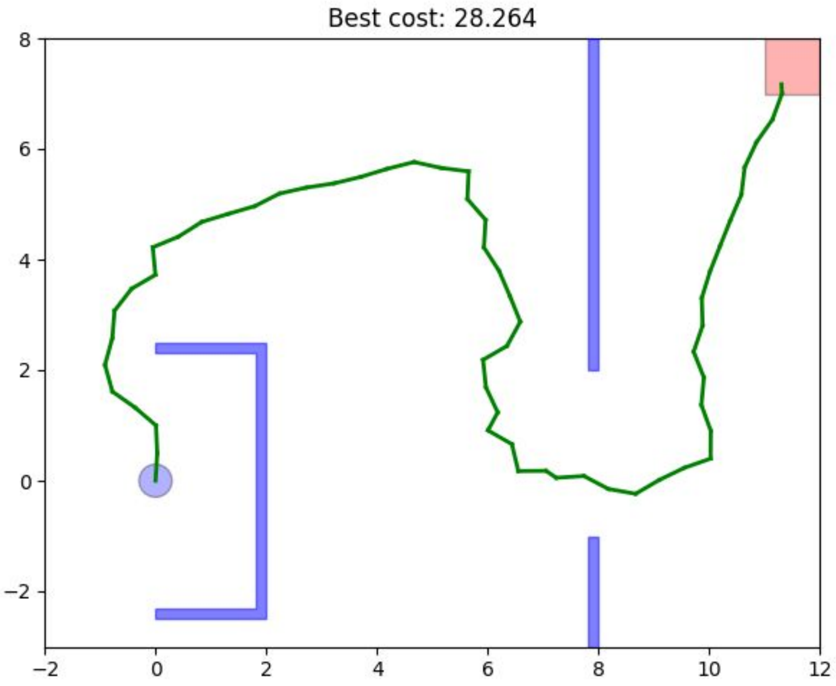




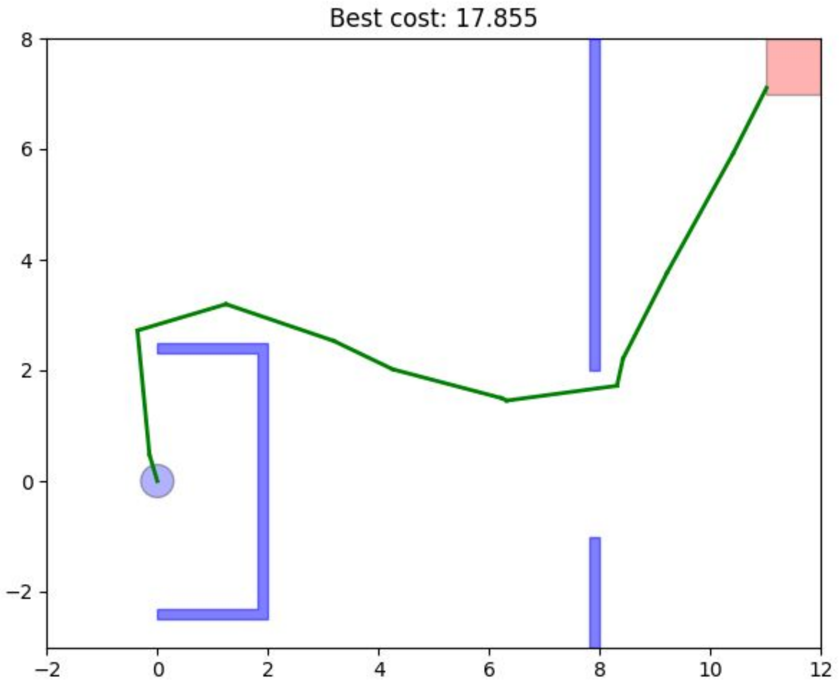
Tests

Holonomic RRT

- Goal bias: 3%
- 9 seconds
- Eucl cost: 28.26



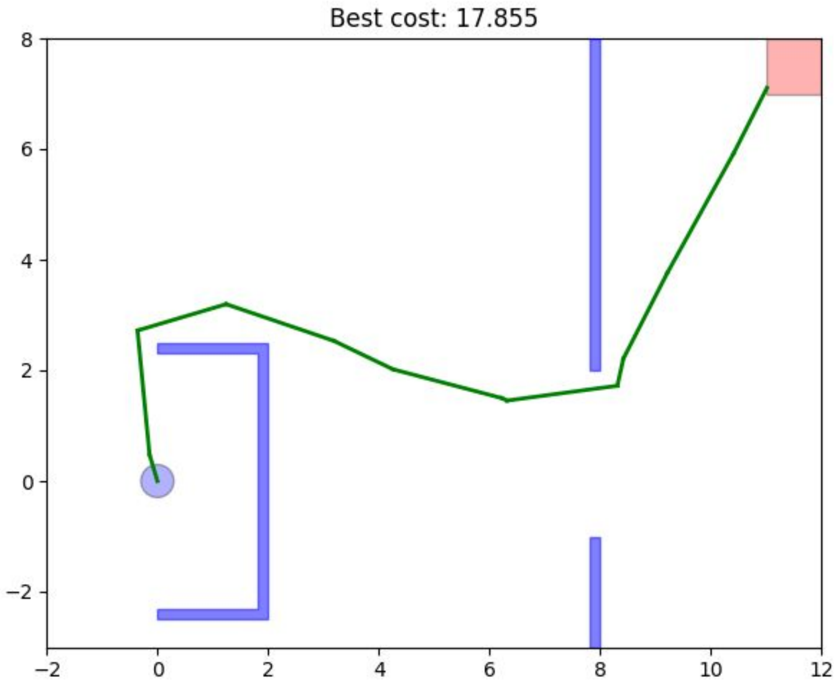
Tests

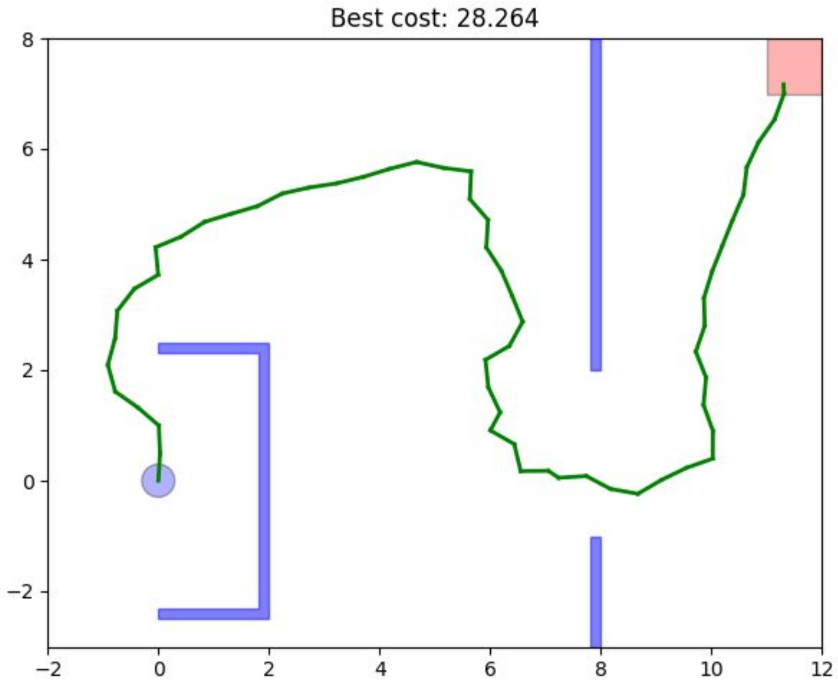


Holonomic RRT*

- Goal bias: 3%
- 247 seconds
- Eucl cost: 17.86

Tests

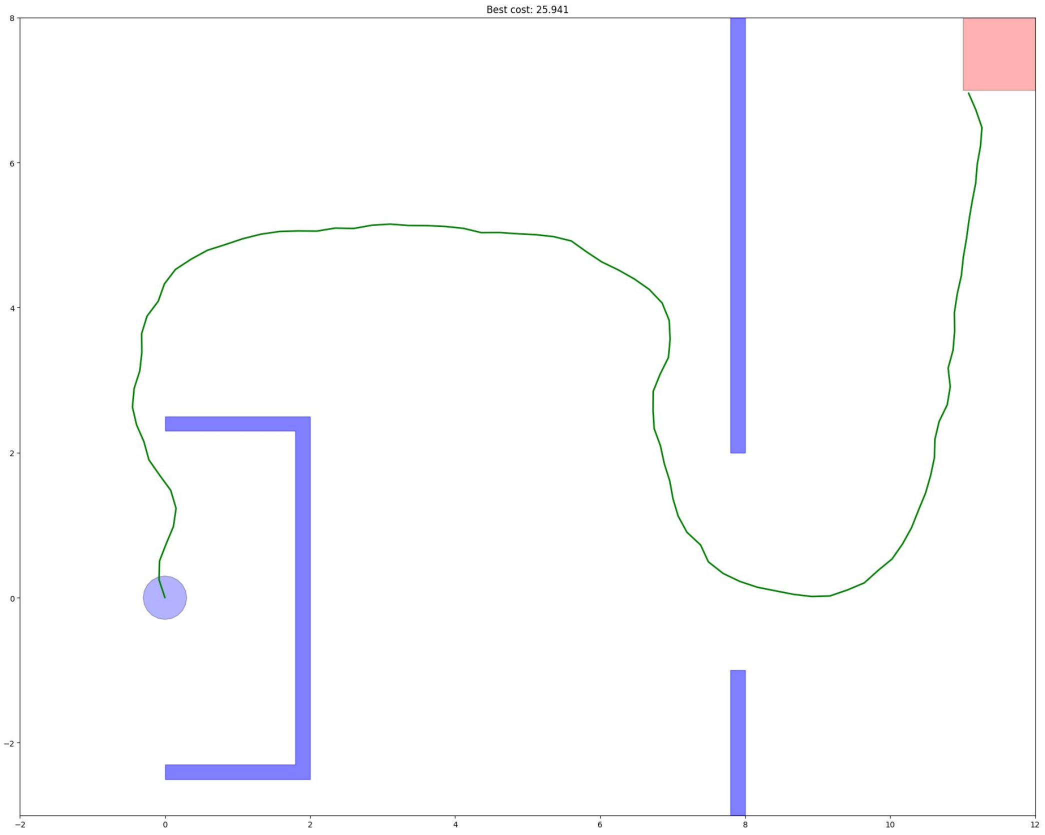

Best cost: 17.855
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12


Best cost: 28.264
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12

Tests

Non-holonomic RRT

- Steer according to simple car model at low, constant, velocity
- Goal bias: 0.03%
- 43 seconds (several runs)

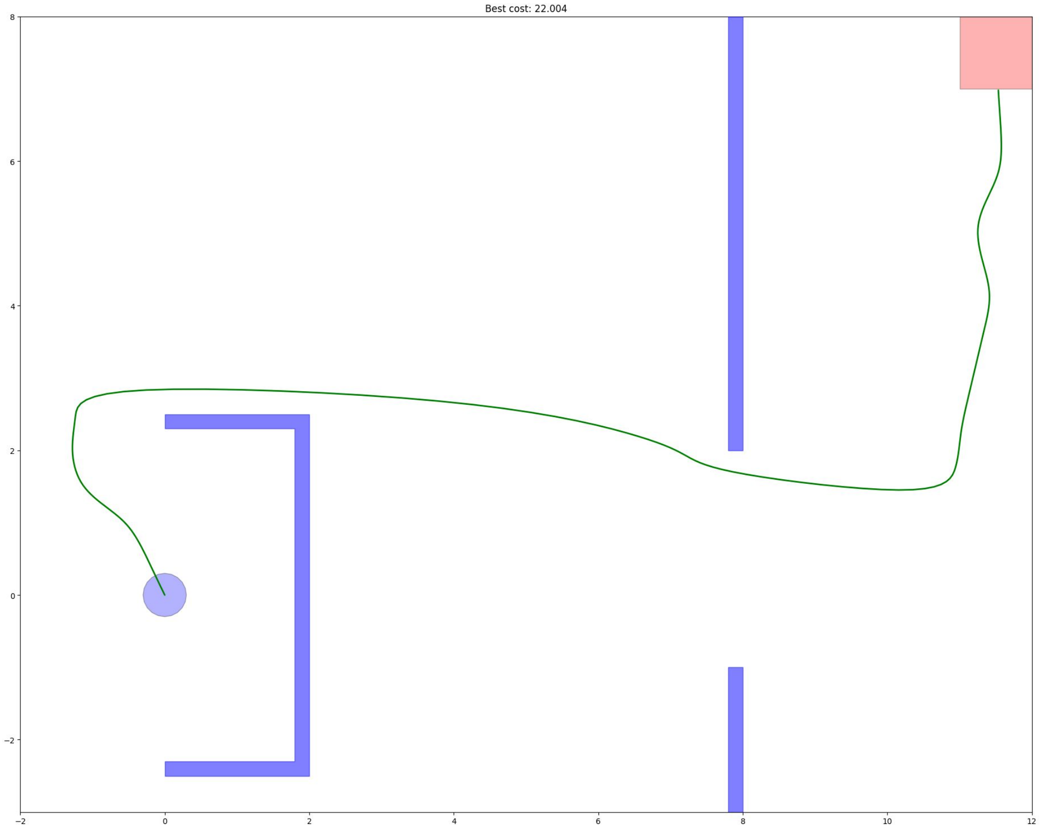


Tests

Non-holonomic

Informed RRT*

- Steer according to simple car model at low, constant, velocity
- Goal bias: 0.03%
- 20 seconds

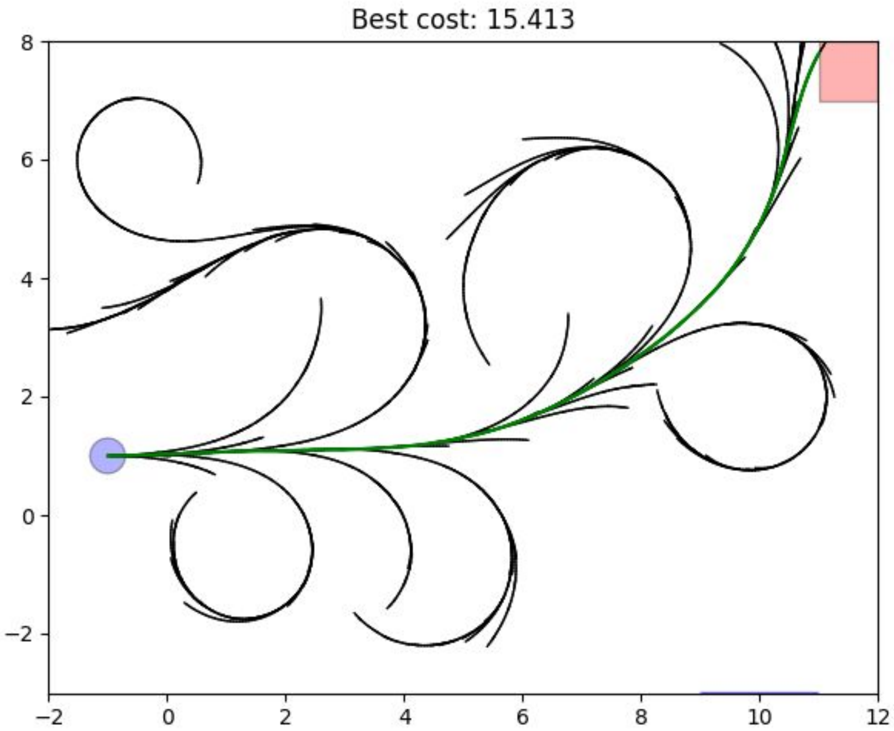


Tests

Non-holonomic

Informed RRT*

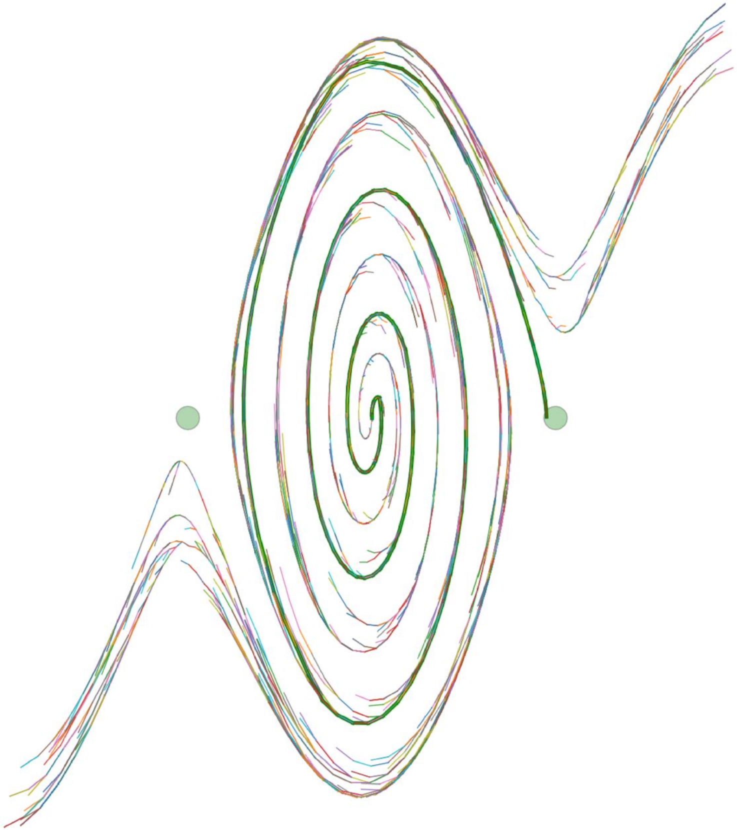
- Steer according to car model with mo complex dynamics: slip and friction



Experiments

Simple pendulum with RRT

- Swing up with torque set to max or min
- Torque decided based on sampled (θ, dθ/dt)
- RK4 + dynamics OK



Thank you!