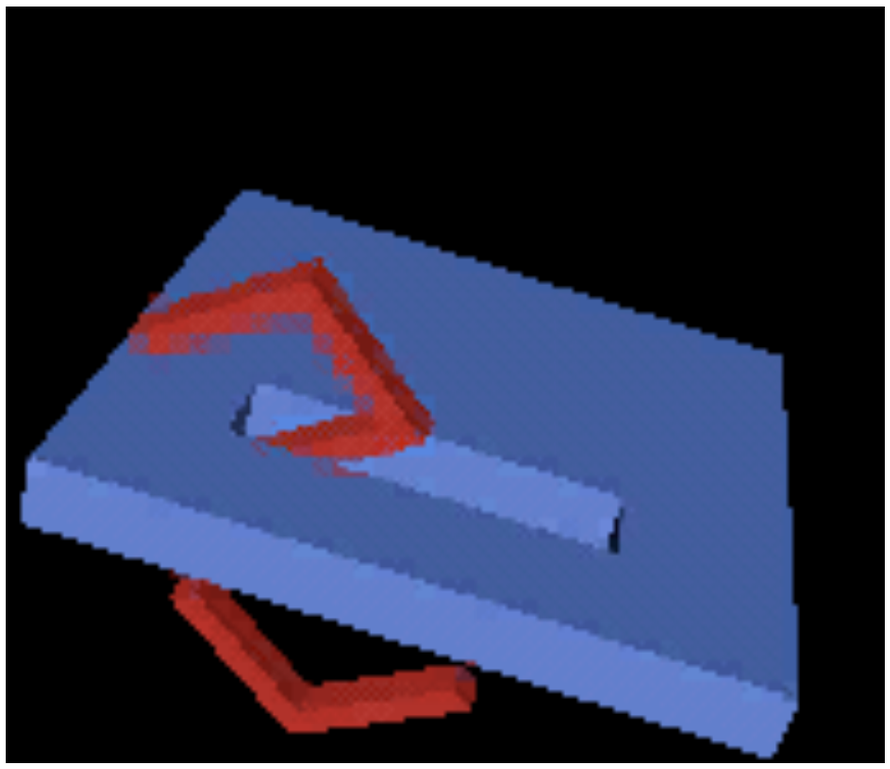
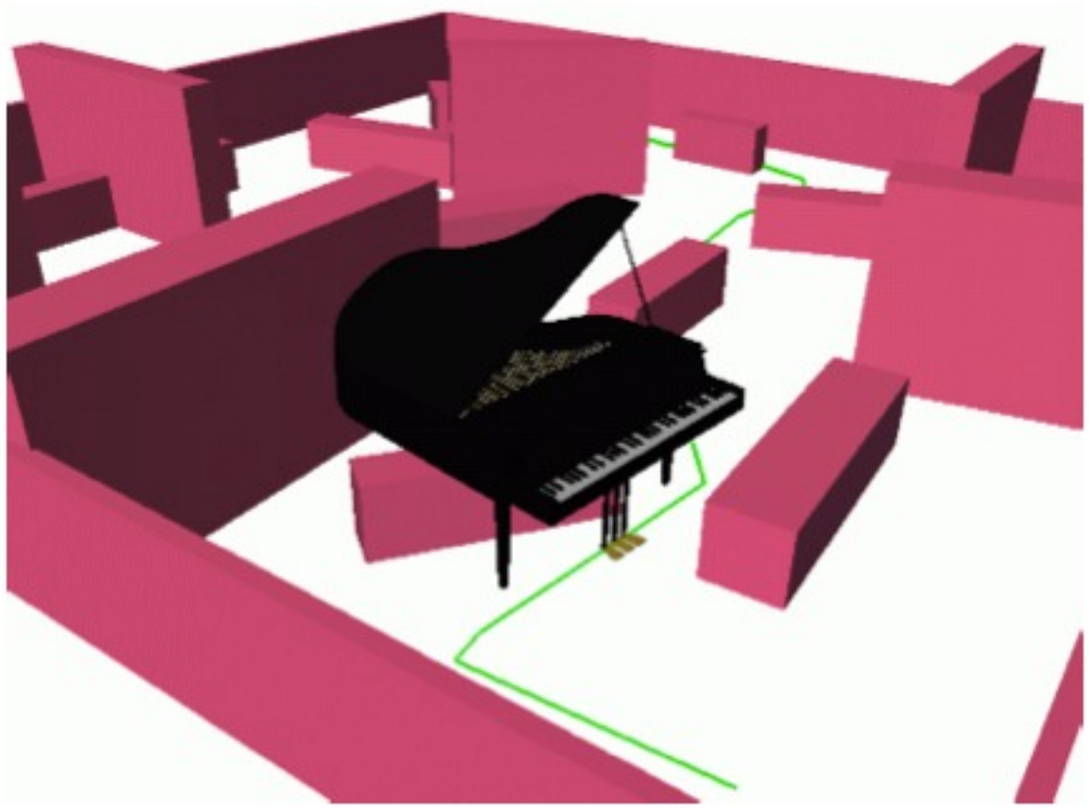
PRM, RRT, and RRT*

Frank Dellaert
Based on materials by Steve Lavalle, Lydia Kavraki,
Emilio Frazzoli and their students

Holonomic Planning



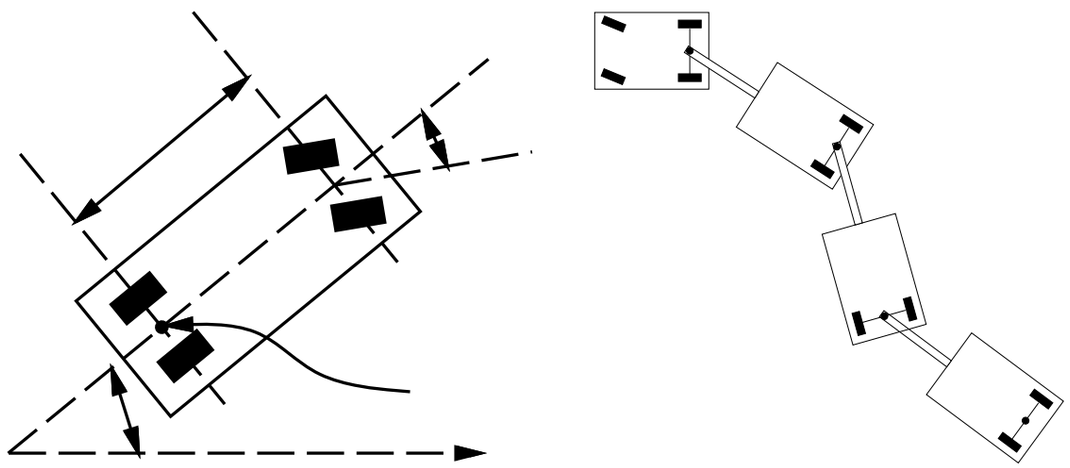


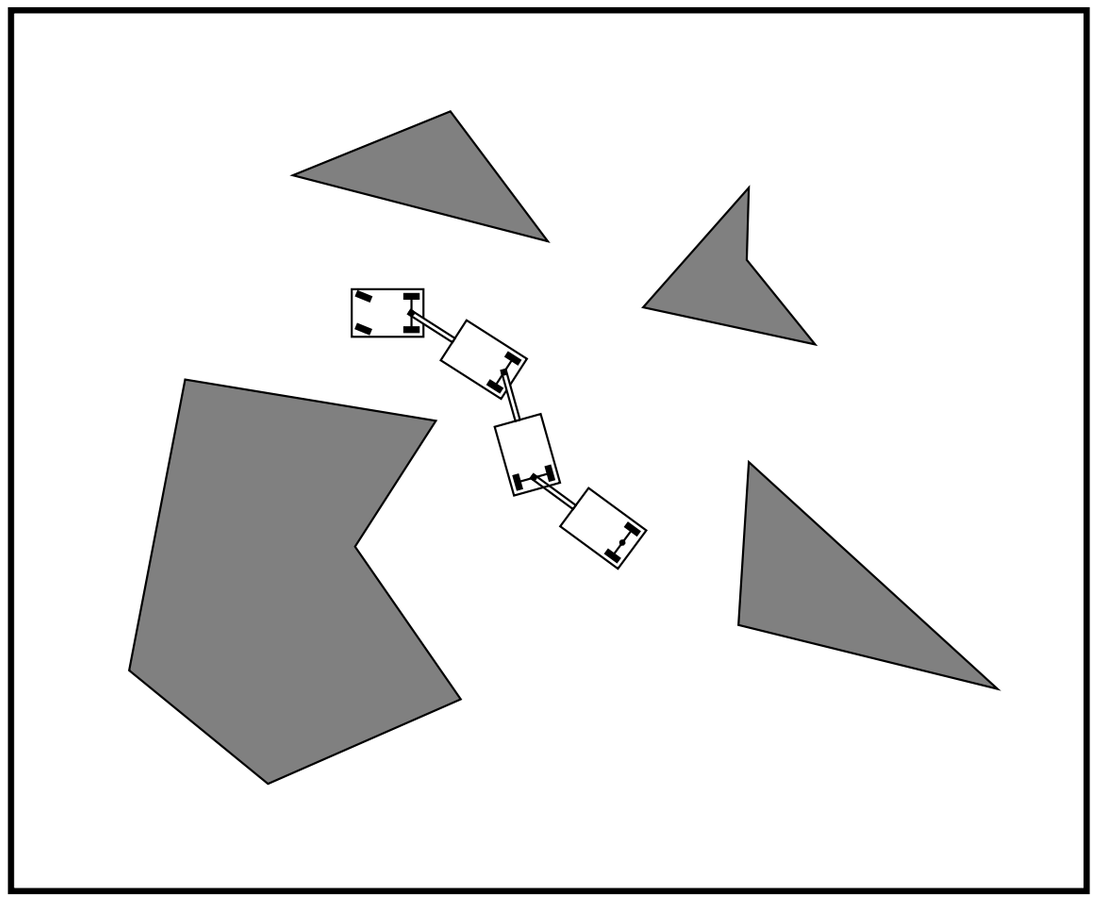
http://www.youtube.com/watch?v=cXm3WW-geD8

Non-Holonomic

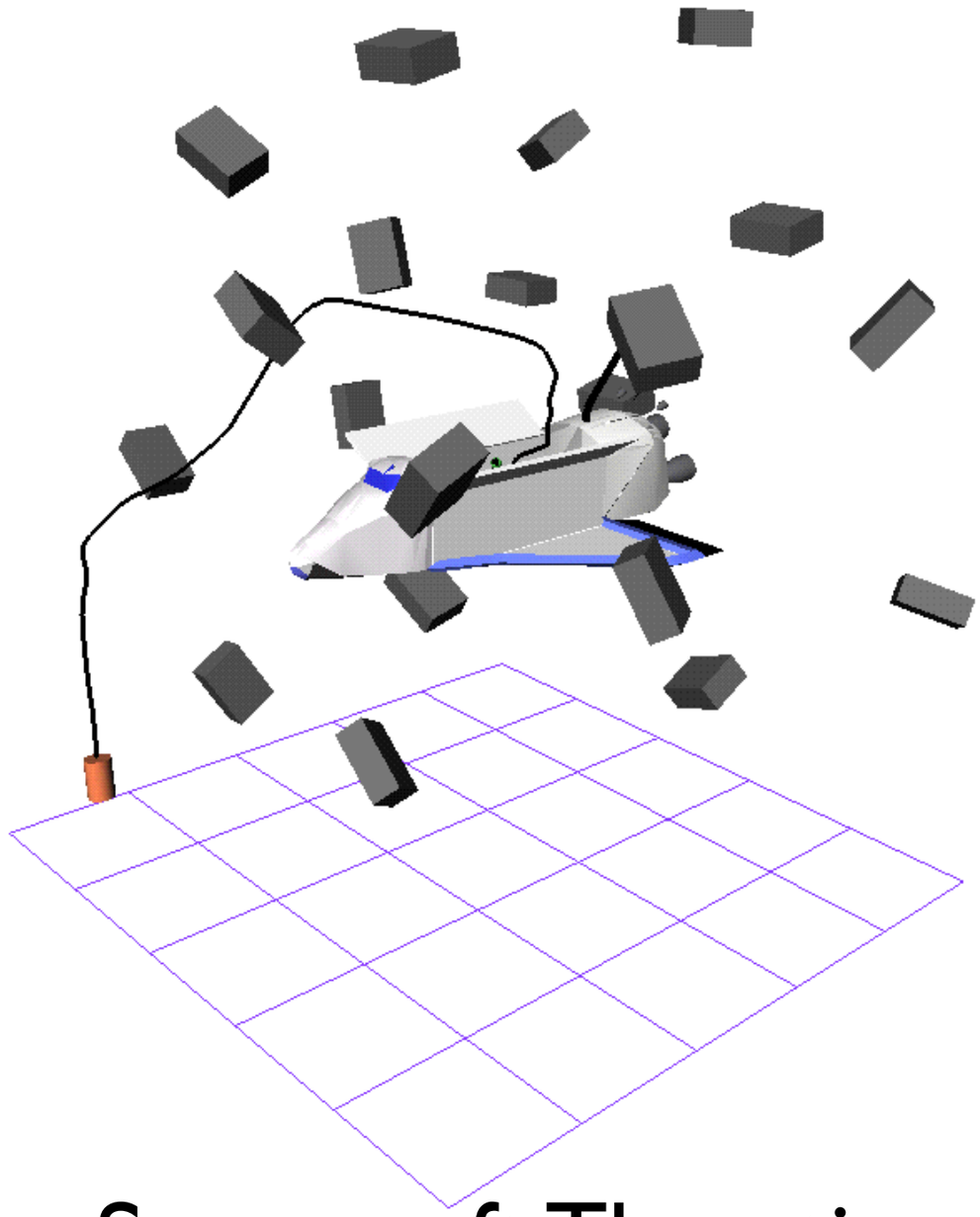
Nonholonomic Path Planning

"Driving Cars"





Non-Holonomic



Spacecraft Thrusting

PRM

RRT

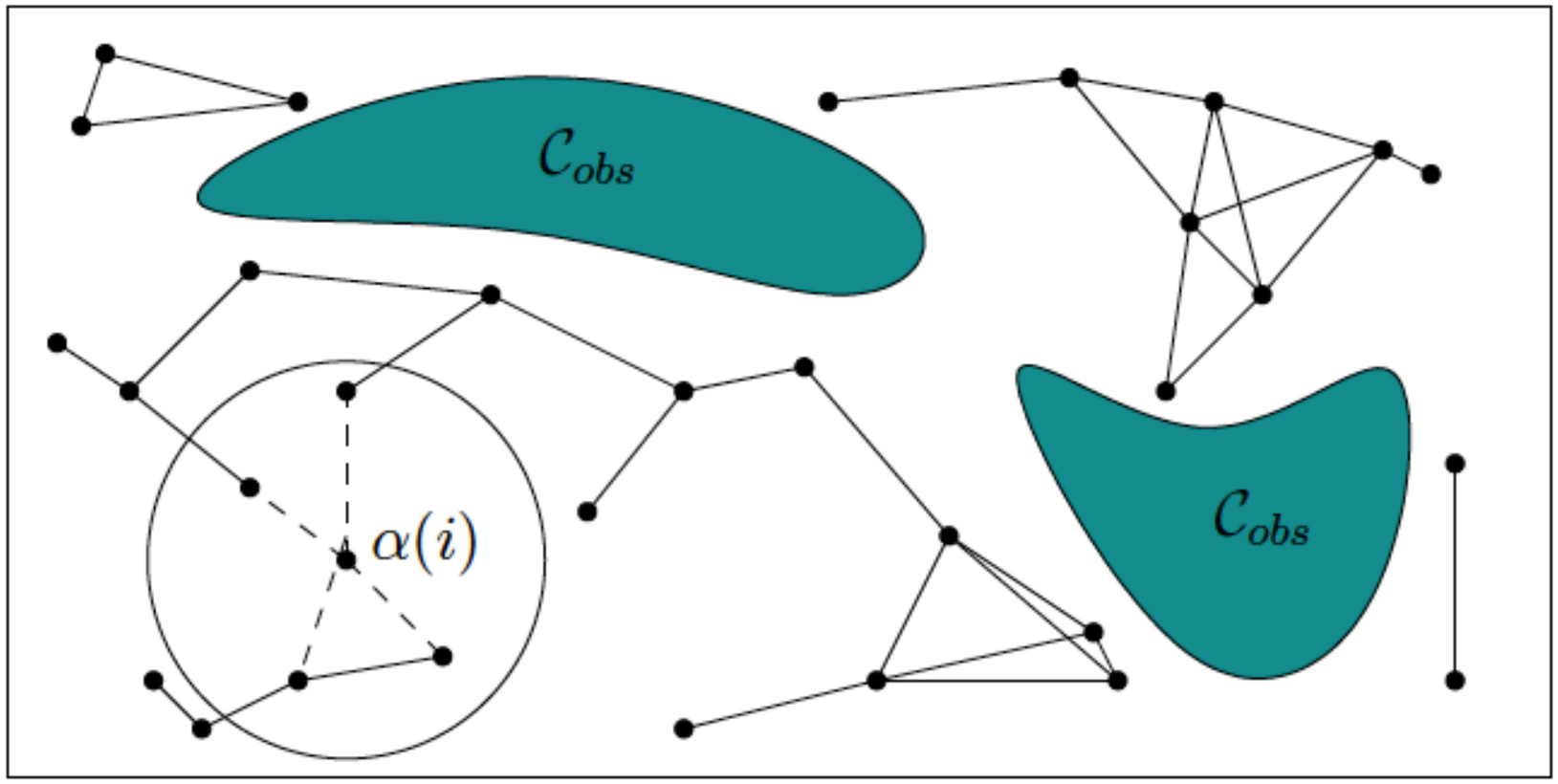
RRT*

PRM

RRT

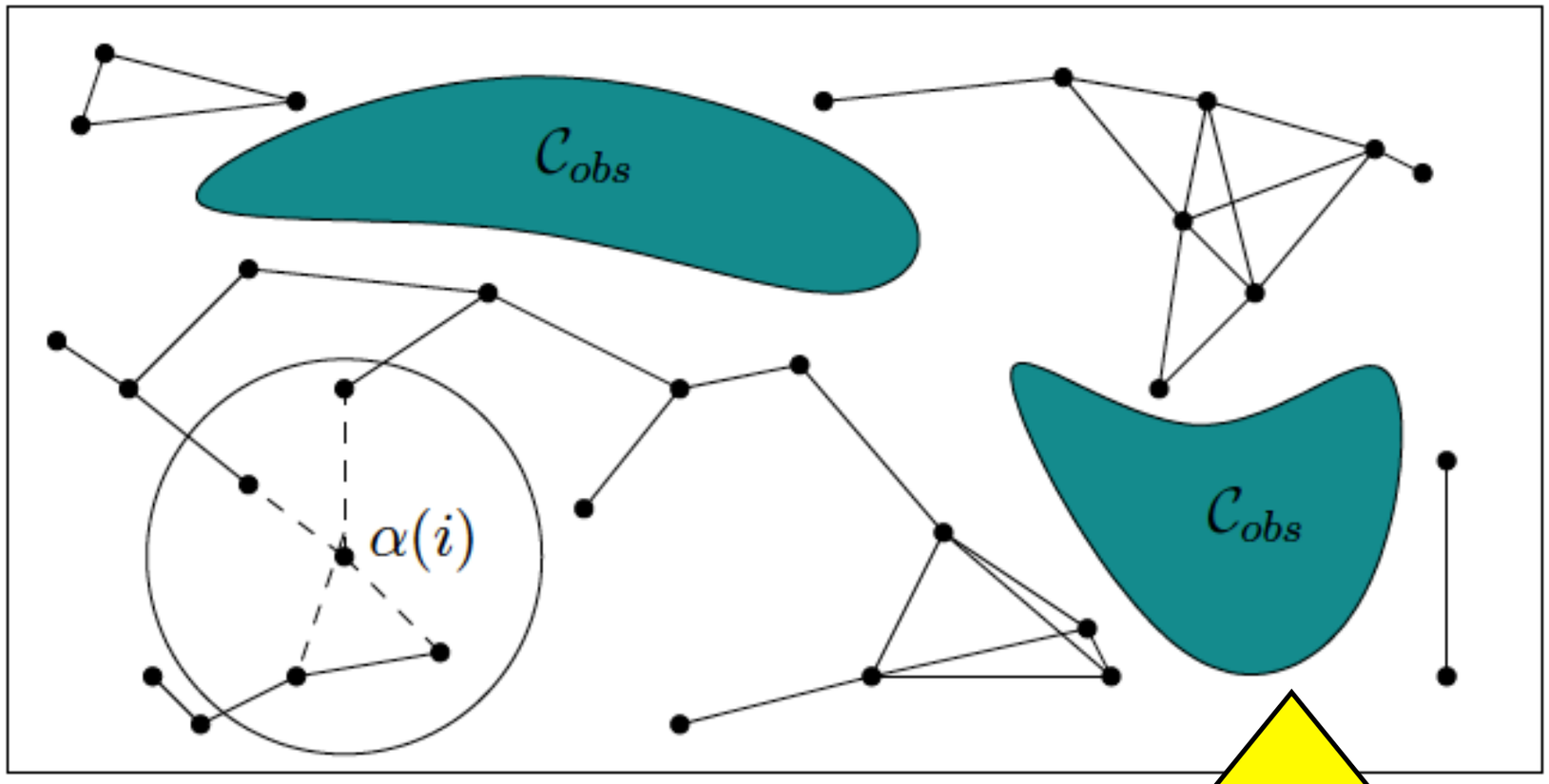
RRT*

Probabilistic Roadmap




```
BUILD_ROADMAP
 1  G.init(); i ← 0;
 2  while i < N
 3      if α(i) ∈ C_free then
 4          G.add_vertex(α(i)); i ← i + 1;
 5          for each q ∈ NEIGHBORHOOD(α(i),G)
 6              if ((not G.same_component(α(i), q)) and CONNECT(α(i), q)) then
 7                  G.add_edge(α(i), q);
```

Probabilistic Roadmap



Demo!

```
BUILD_ROADMAP
 1  G.init(); i ← 0;
 2  while i < N
 3      if α(i) ∈ C_free then
 4          G.add_vertex(α(i)); i ← i + 1;
 5          for each q ∈ NEIGHBORHOOD(α(i),G)
 6              if ((not G.same_component(α(i), q)) and CONNECT(α(i), q)) then
 7                  G.add_edge(α(i), q);
```

PRM Properties

- Good
  - Multiple Queries
  - Probabilistically Complete
- Bad:
  - Determining connectivity can be hard
  - Especially in non-holonomic planning

PRM

RRT

RRT*

Problem Formulation

State Space

$$X = T(\mathcal{C}), \text{ or } X = \mathcal{C}, \text{ etc.} \quad (x = [q \;\; \dot{q}], \text{ or } x = q, \text{ etc.})$$

Metric

$$\rho : X \times X \to [0, \infty)$$

State Transition Equation

$$\dot{x} = f(x, u) \quad (\text{solve } h_i \text{ for } \dot{x};\ u \text{ is an input})$$

Initial and Goal Conditions

$$x_{init} \in X \text{ and } X_{goal} \subset X \text{ or } x_{goal} \in X$$

Collision Detection

Given $x \in X$, determine whether $x \in X_{obs}$

Incremental Simulator

$$x(t + \Delta t) \approx x(t) + \Delta t \; f(x(t), u(t)) \quad (\text{or Runge-Kutta})$$

RRT

Rapidly-exploring Random Trees (RRTs):

PLUS: Little or no heuristics

Steering ability not required

Suited for high dimensions

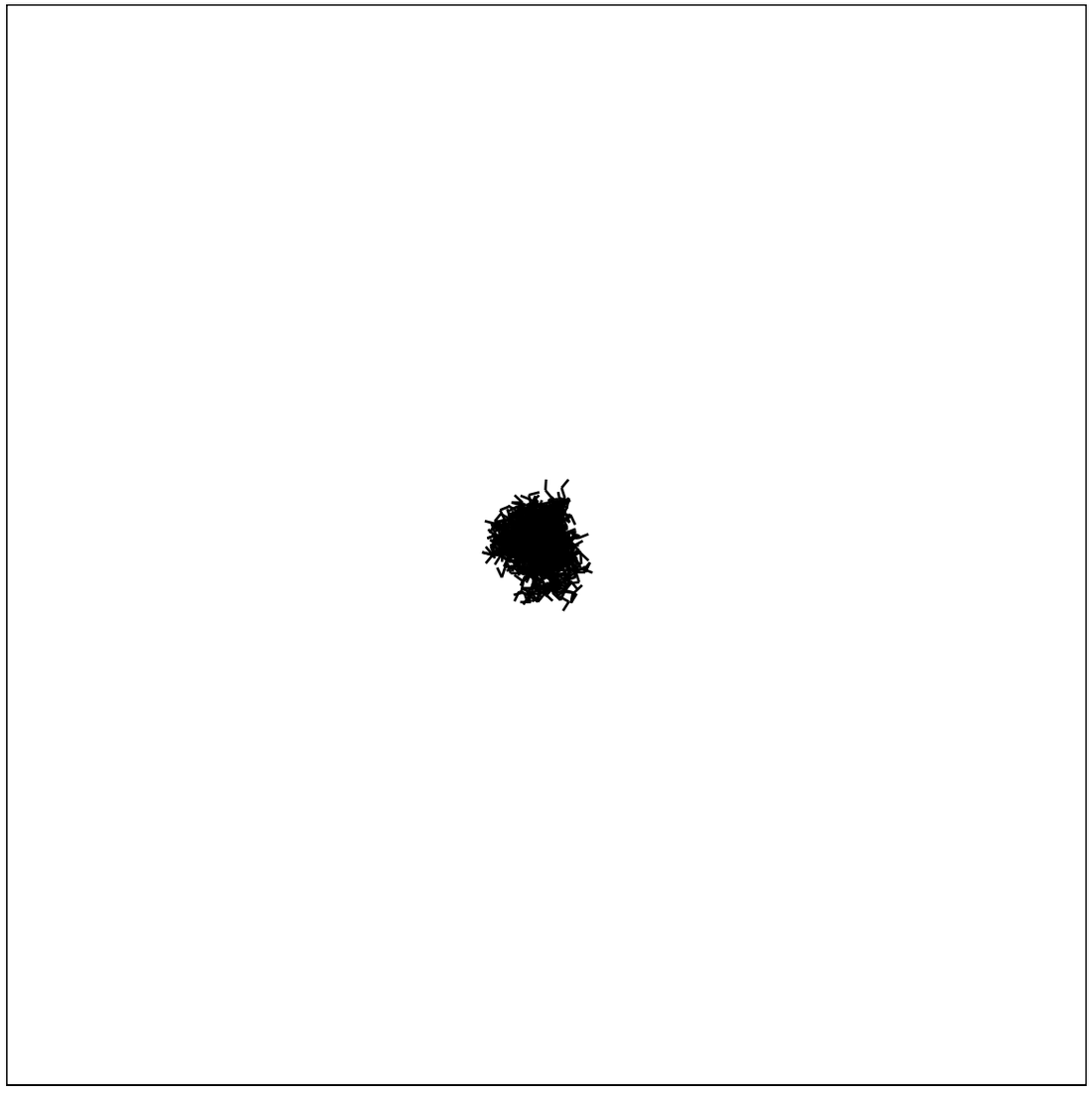
MINUS: Completeness sacrificed

Metric issues

An RRT is a search tree:

- grown from an initial state
- expanded by performing incremental motions

Naive Random Tree



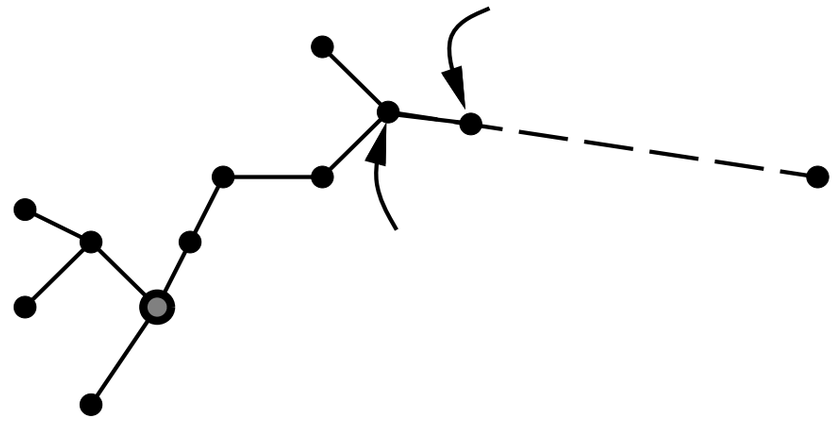
1. Pick a vertex at random
2. Move in a random direction to generate a new vertex
3. Repeat...

Algorithm

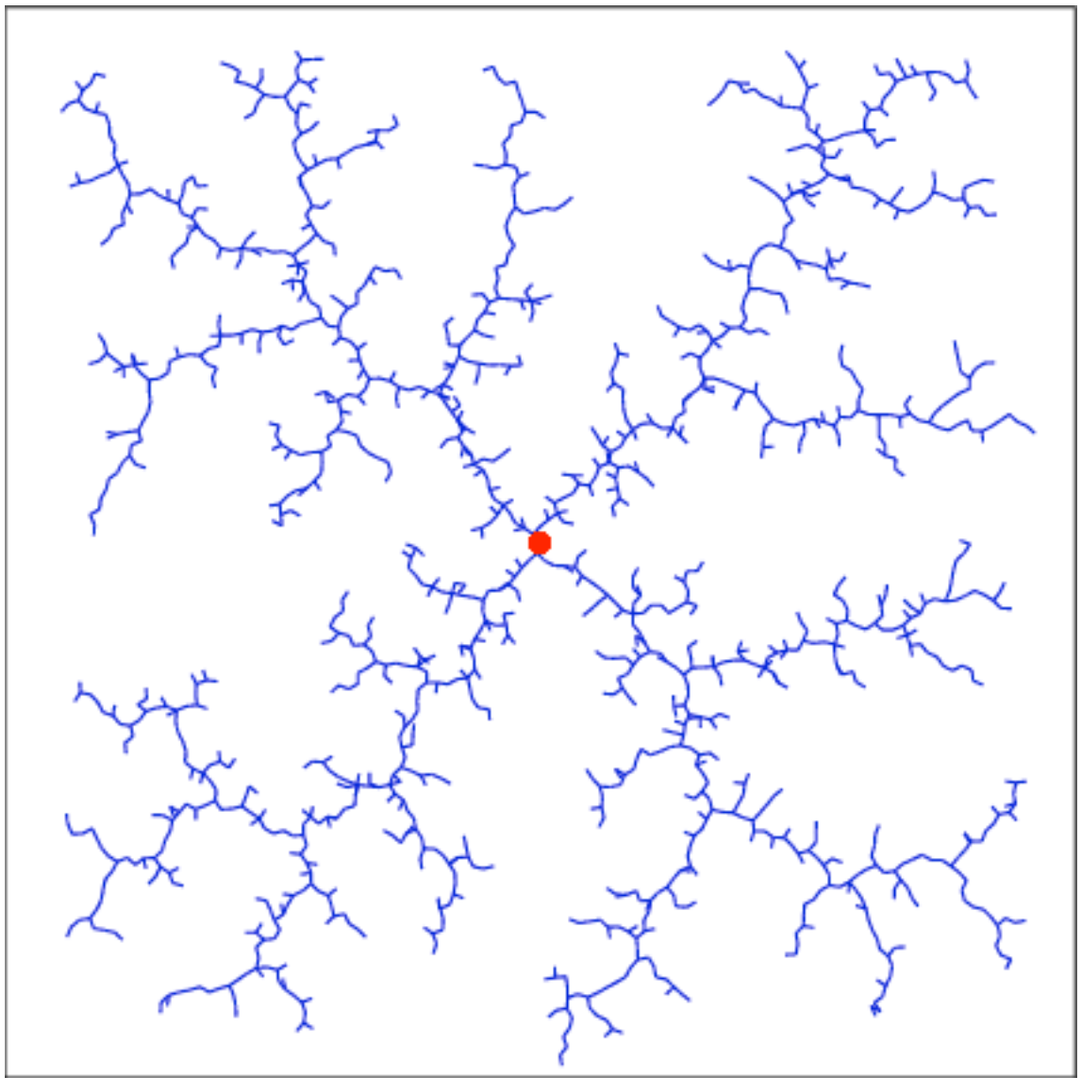
GENERATE_RRT($x_{init}, K, \Delta t$)

1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 $\mathcal{T}$.add_vertex(x_{new});
8 $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

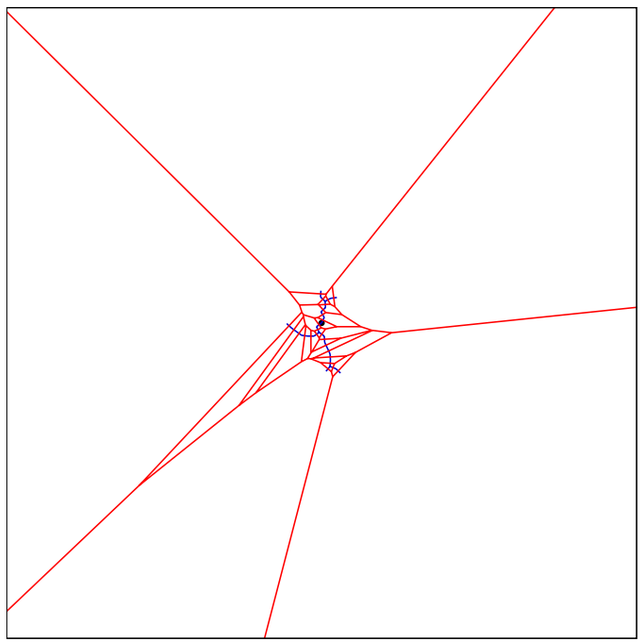
The result is a tree, $\mathcal{T}$, rooted at x_{init}.

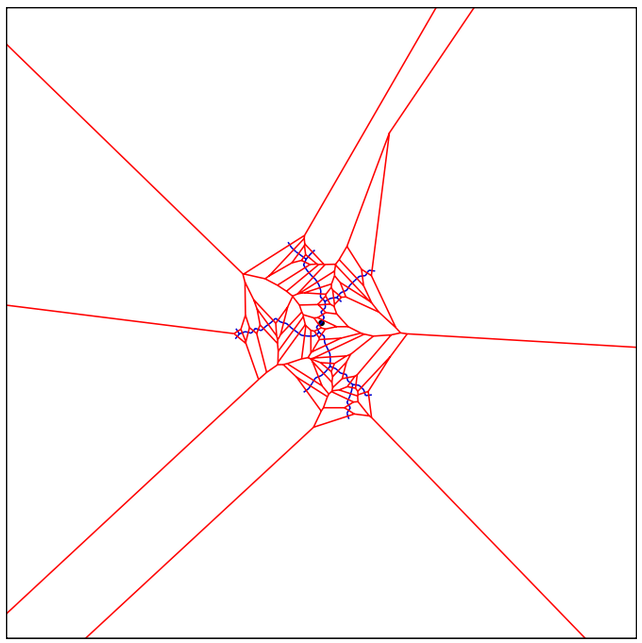


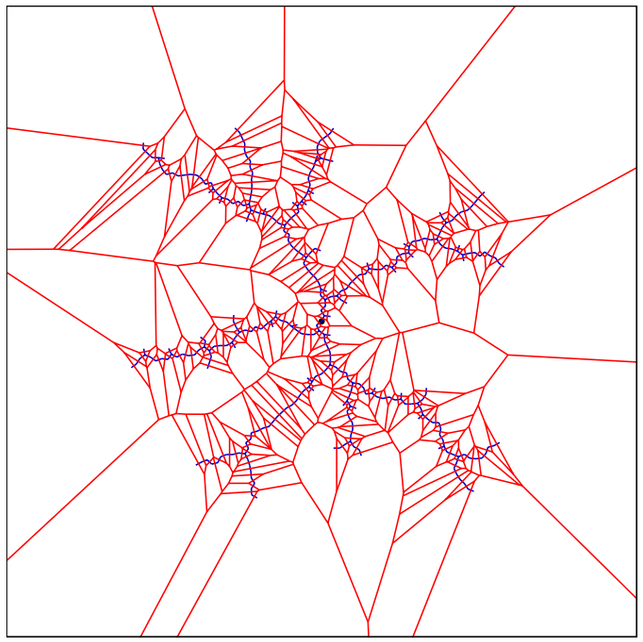
Convex Region

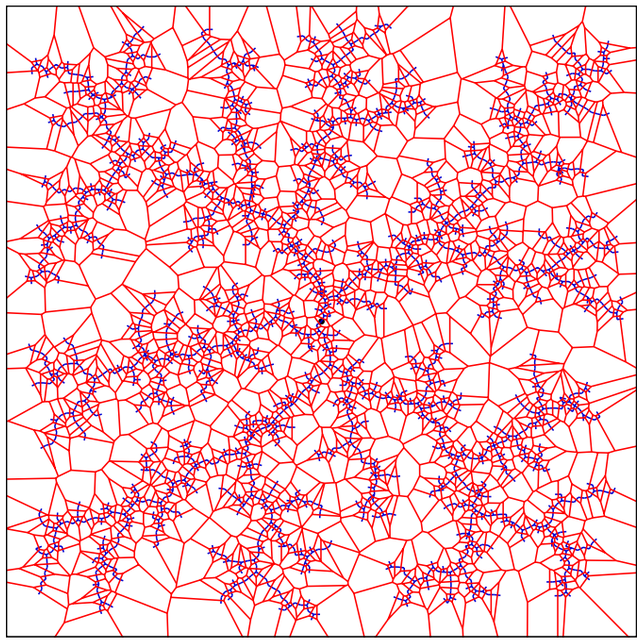


Voronoi Interpretation









Vertices on frontier selected more often

RRT Advantages

Desirable Properties:

- Suited for high degrees of freedom
- Steering ability is not required
- Exploration biased toward unexplored free space
- Probabilistic completeness and uniform coverage
- Simplicity (few parameters or biases)
- Connected structure using minimal edges

Exploits Existing Algorithms:

- Incremental collision detection (Mirtich, 1997)
- Incremental dynamical simulation (Kuffner, Mirtich, 1998)
- Efficient nearest neighbors (Arya, Mount, 1997)

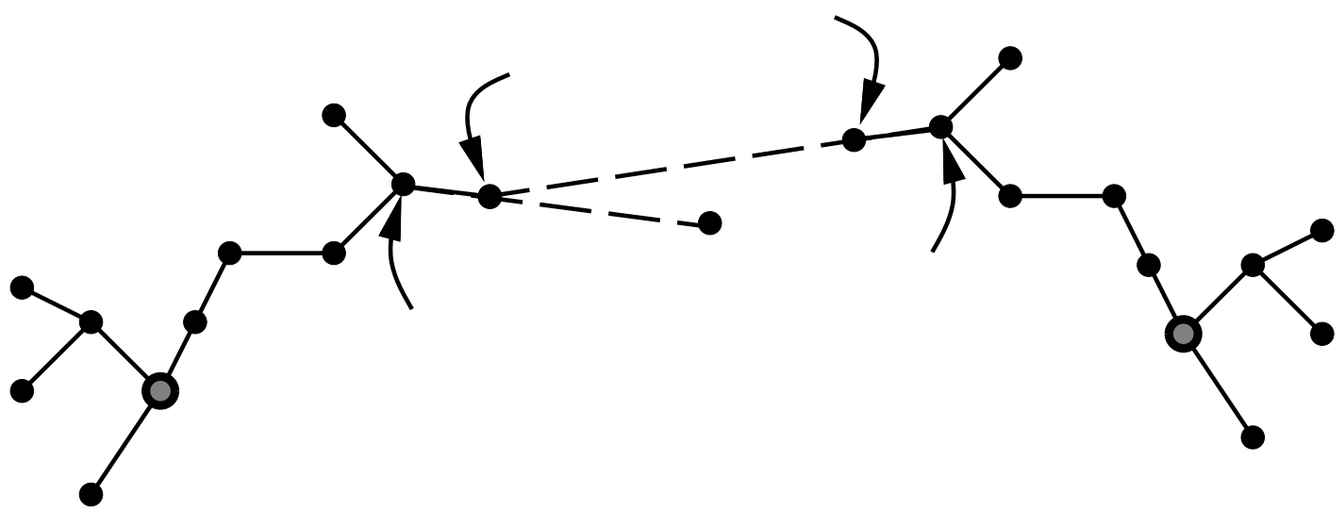
RRT in Planning

Single RRT Planners:

- Goal-biased sampling
- The Connect heuristic

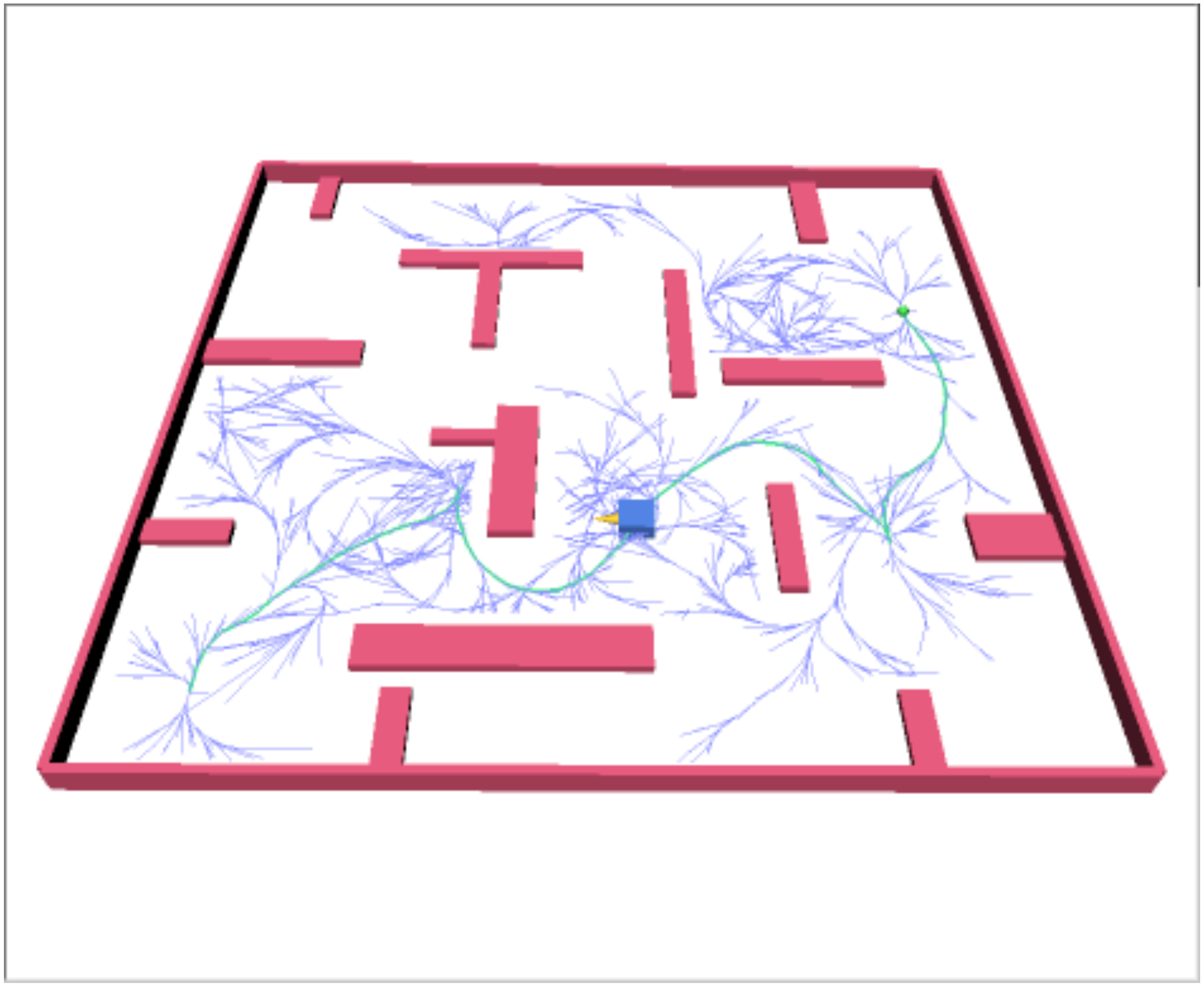
Dual RRT Planners:

- Extend both trees toward samples (ICRA '99)
- Extend trees toward each other and samples



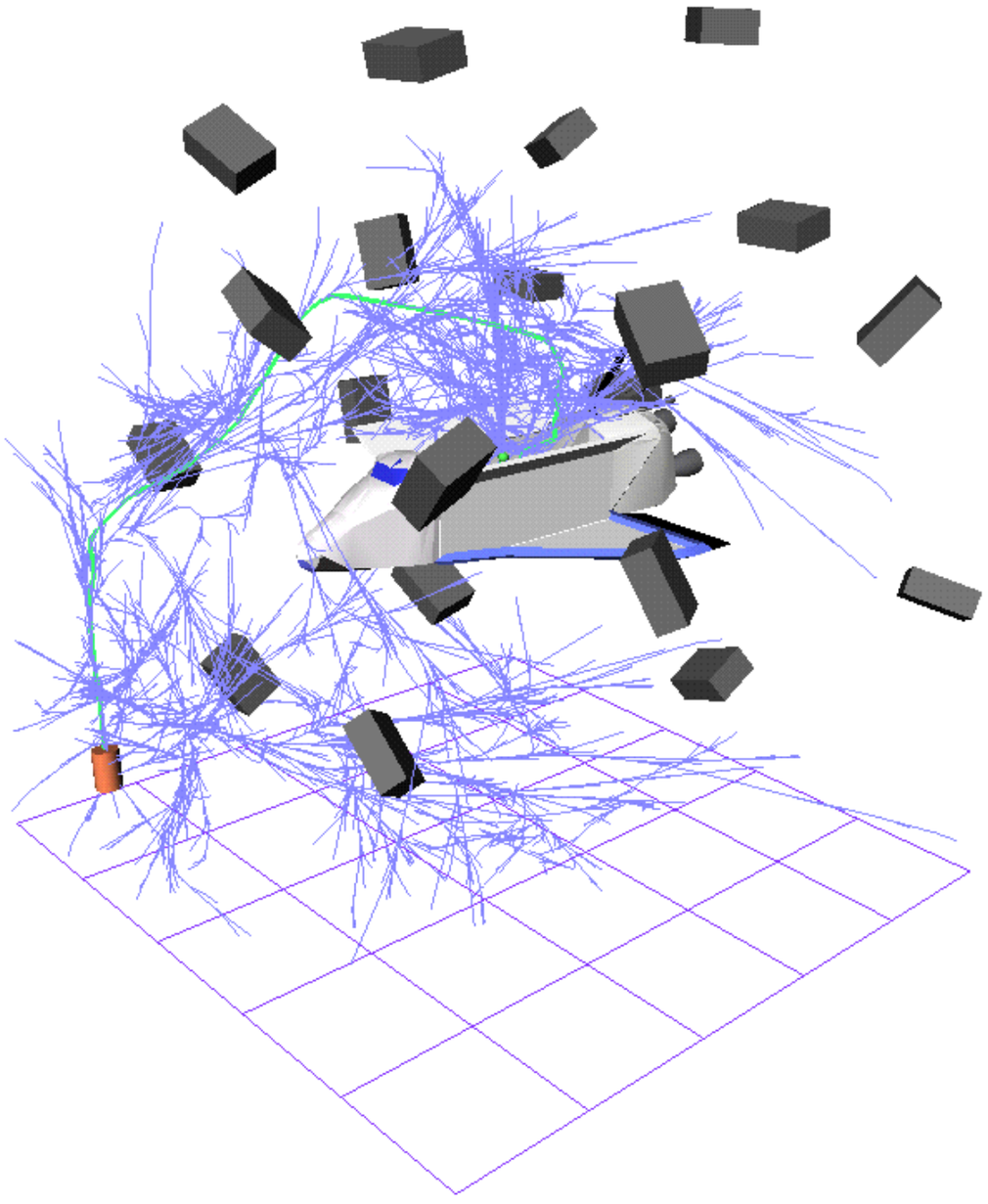
Hovercraft

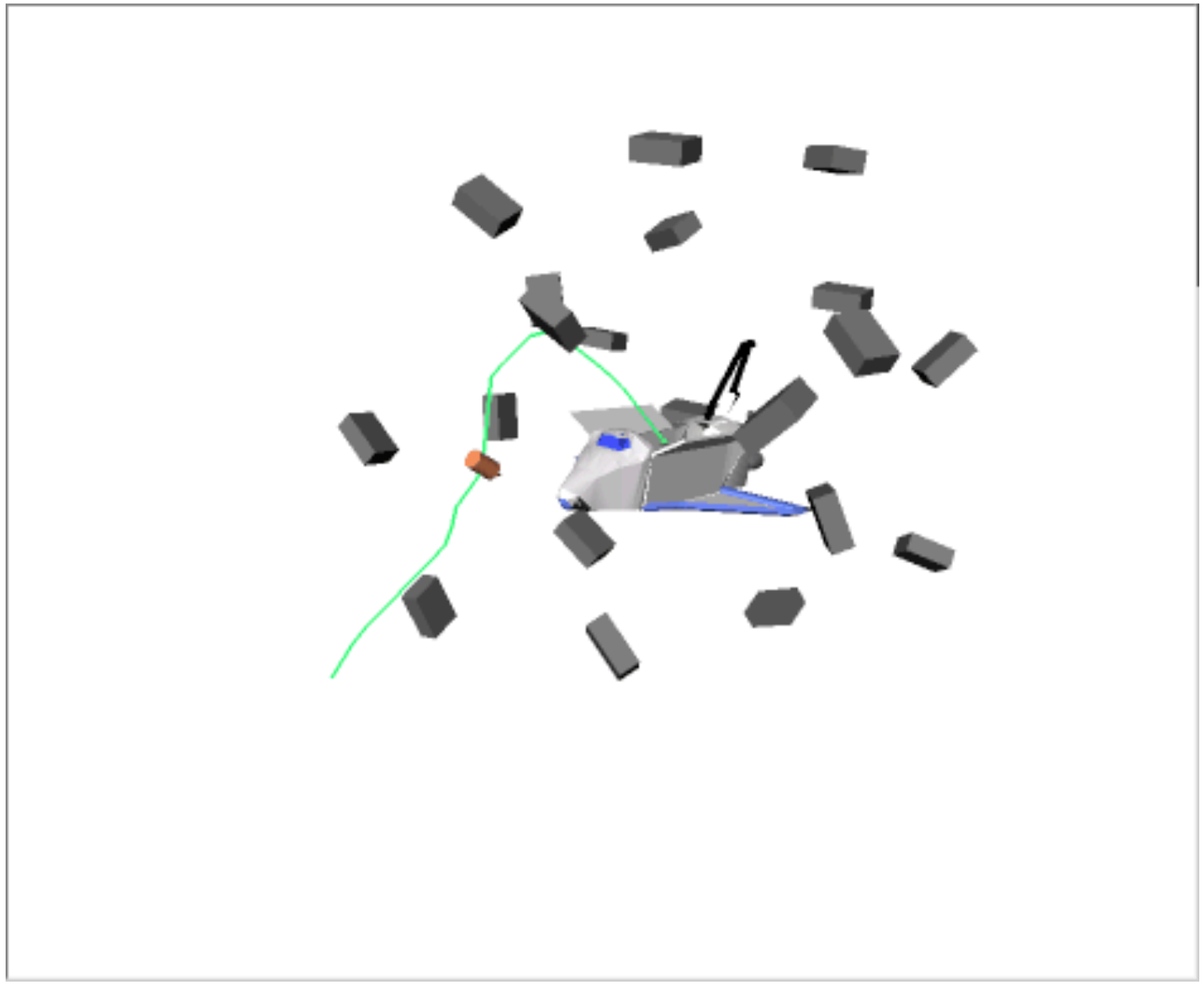
- 4D state space, 2D configuration space
- Two pairs of opposing thrusters
- Computation Time: 10 seconds



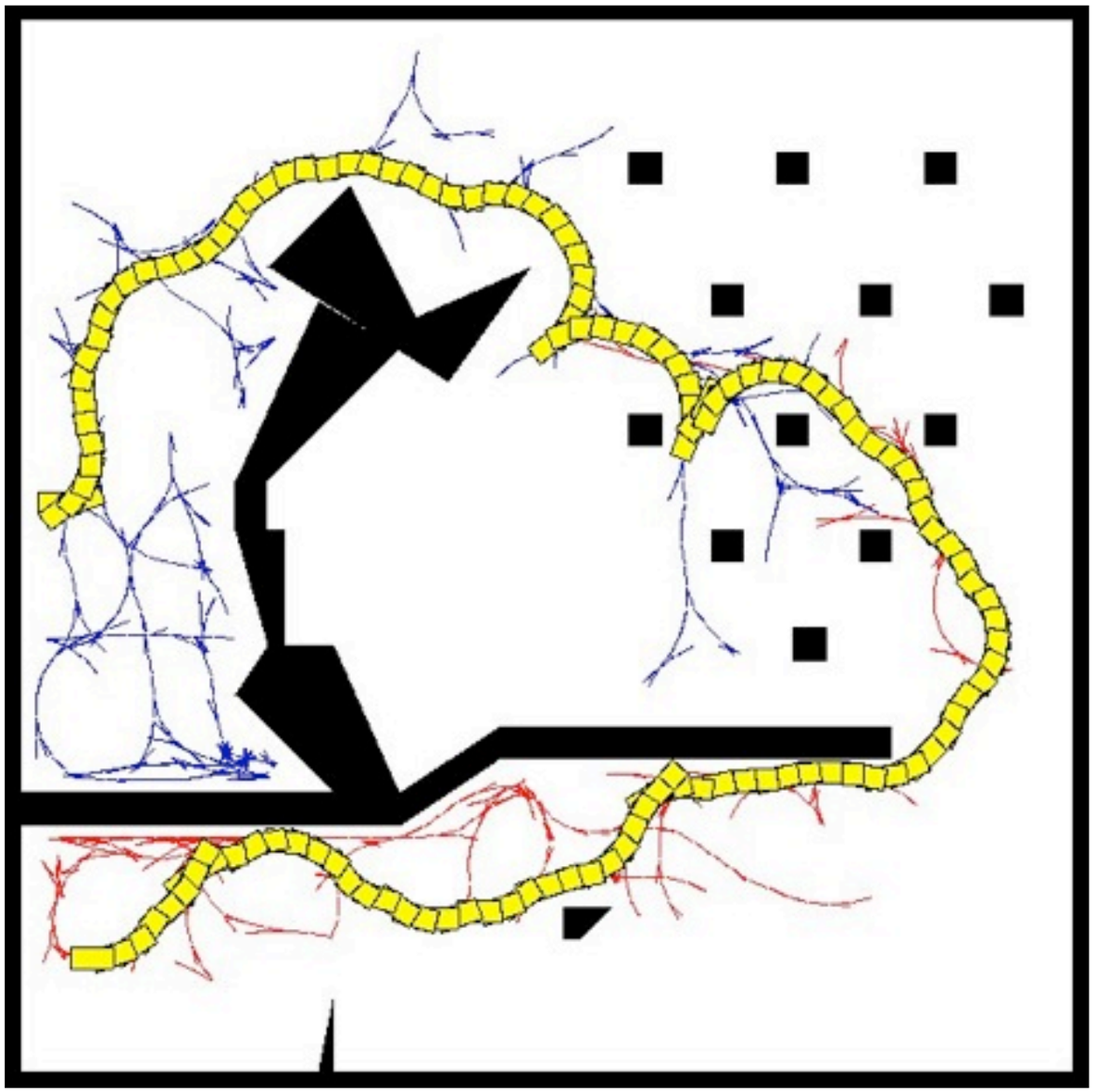
Free Satellite in 3D

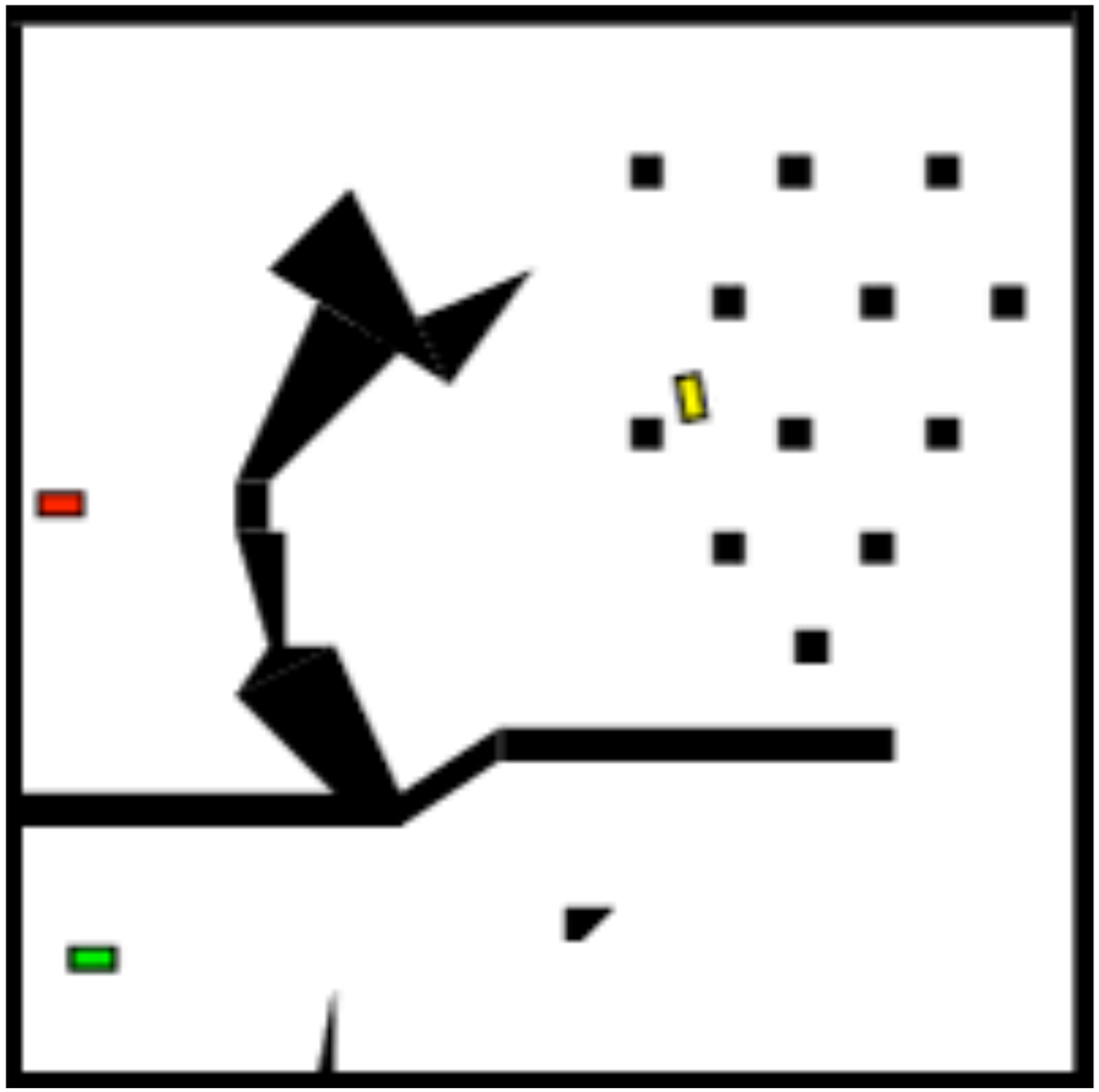
- 12D state space, 6D configuration space
- Momentum wheels and an opposing pair of thrusters
- Computation Time: 8.4 minutes





Planning for a Car-Like Robot





PRM

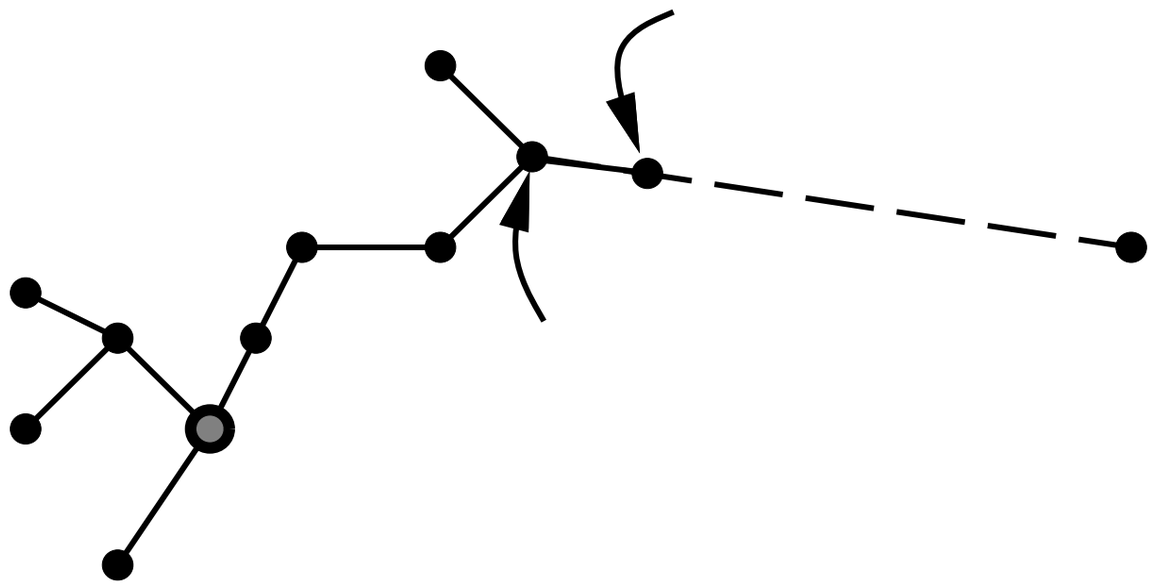
RRT

RRT*

Problems with RRT

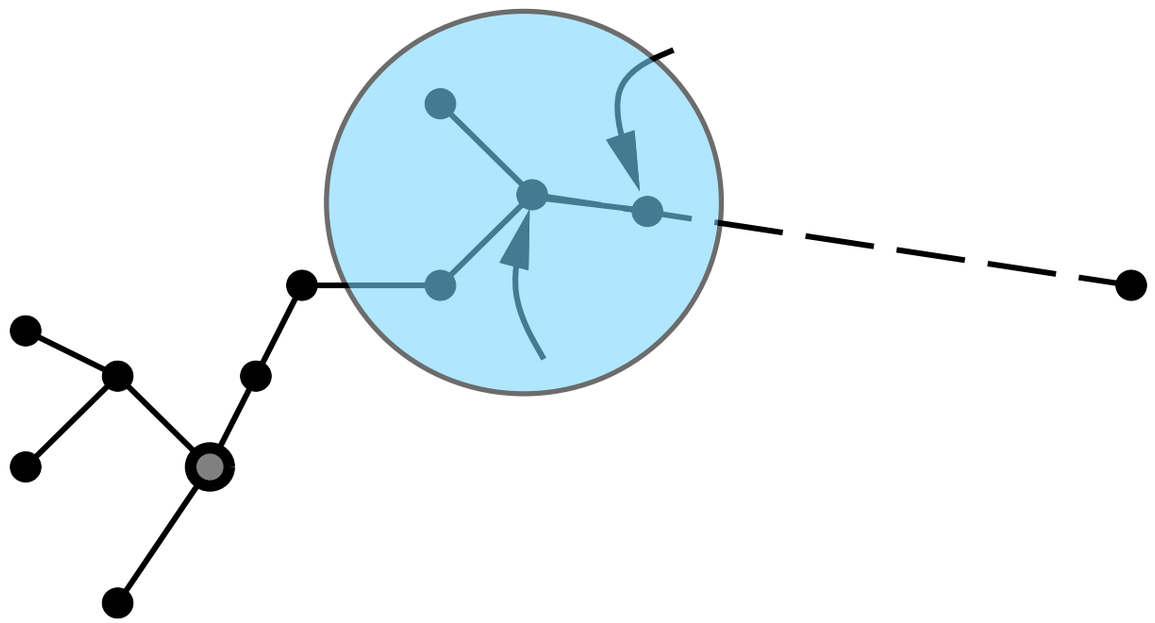
- Solution far from optimal
- Karaman & Frazzoli 2010: Probability of RRT converging to an optimal solution is **0**
- Rapidly-exploring Random Graph (RRG): **I**
- **Downside:** back to finding connections!

RRG



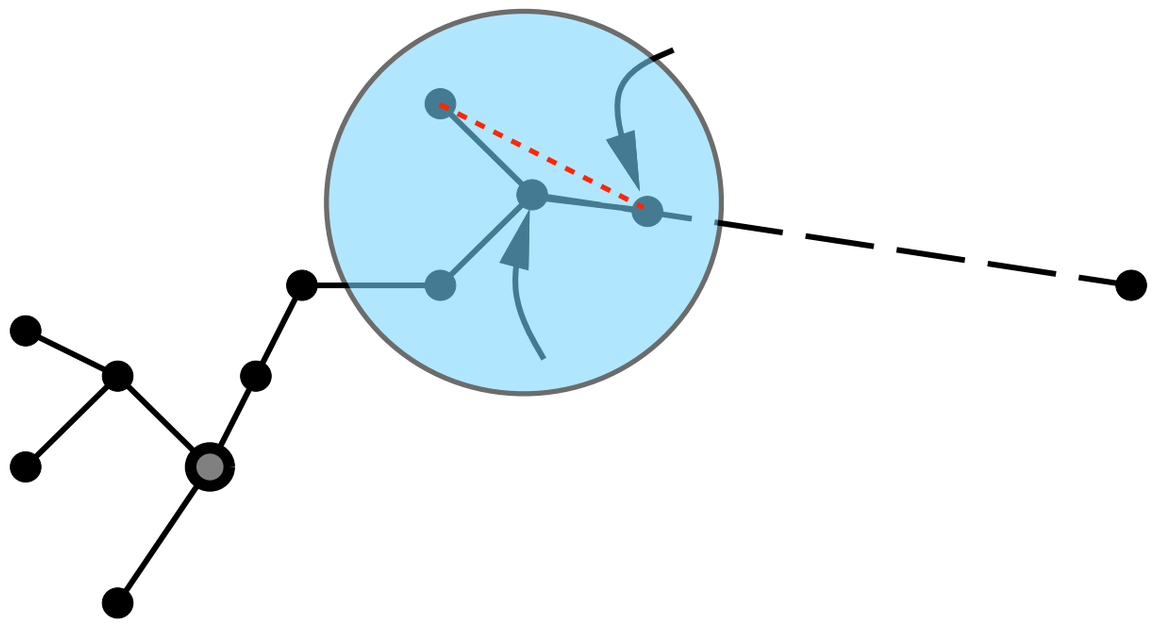
- RRT algorithm extends the nearest vertex towards the sample.
- RRG also extends all vertices returned by the Near procedure (if first was success).

RRG



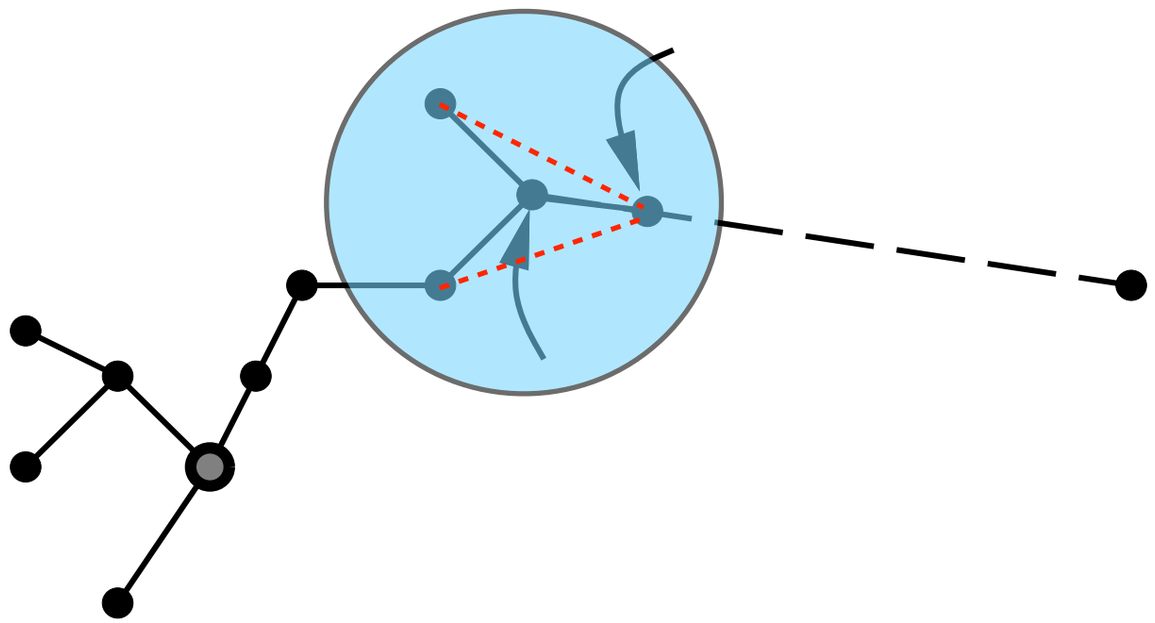
- RRT algorithm extends the nearest vertex towards the sample.
- RRG also extends all vertices returned by the Near procedure (if first was success).

RRG



- RRT algorithm extends the nearest vertex towards the sample.
- RRG also extends all vertices returned by the Near procedure (if first was success).

RRG

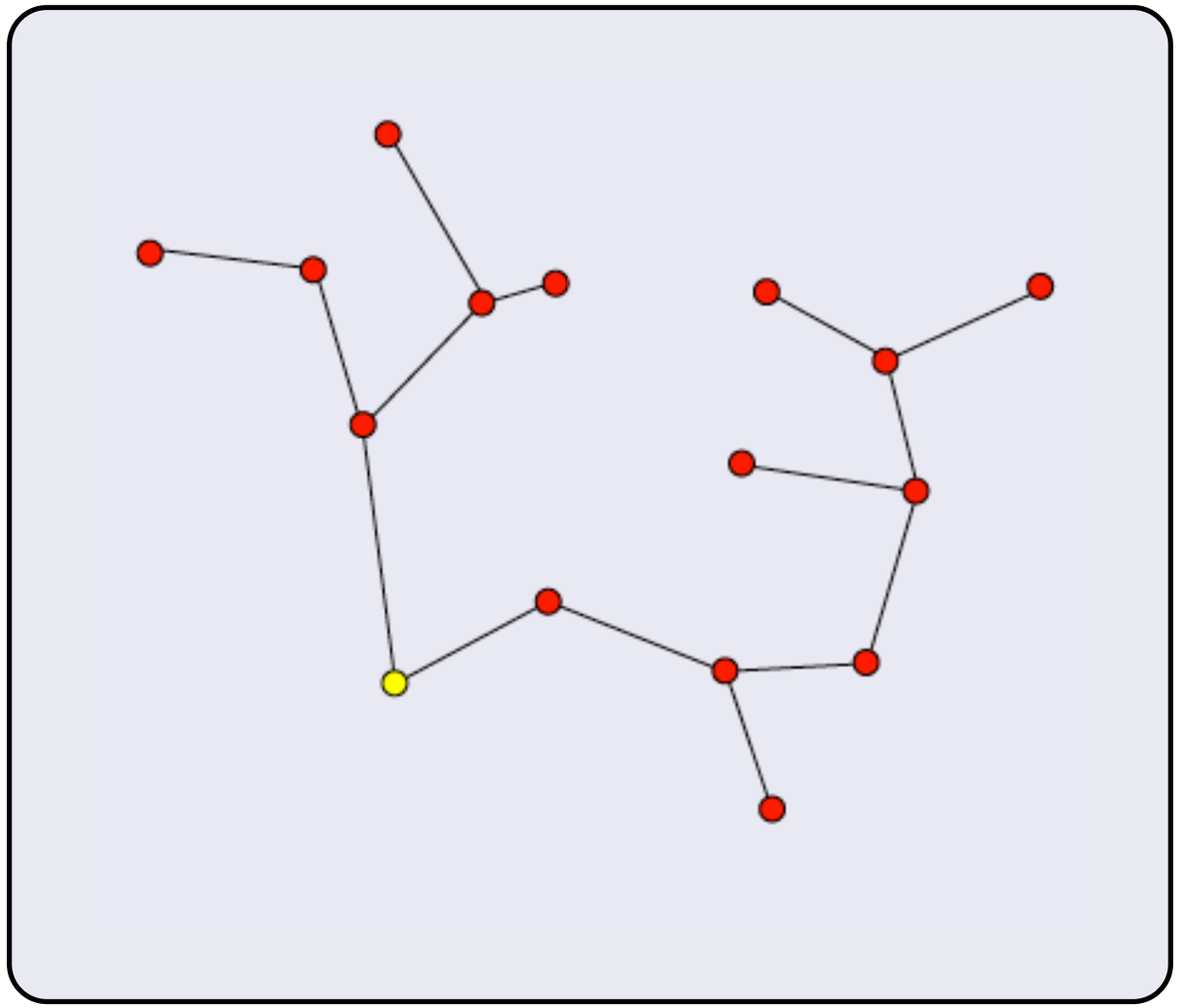


- RRT algorithm extends the nearest vertex towards the sample.
- RRG also extends all vertices returned by the Near procedure (if first was success).

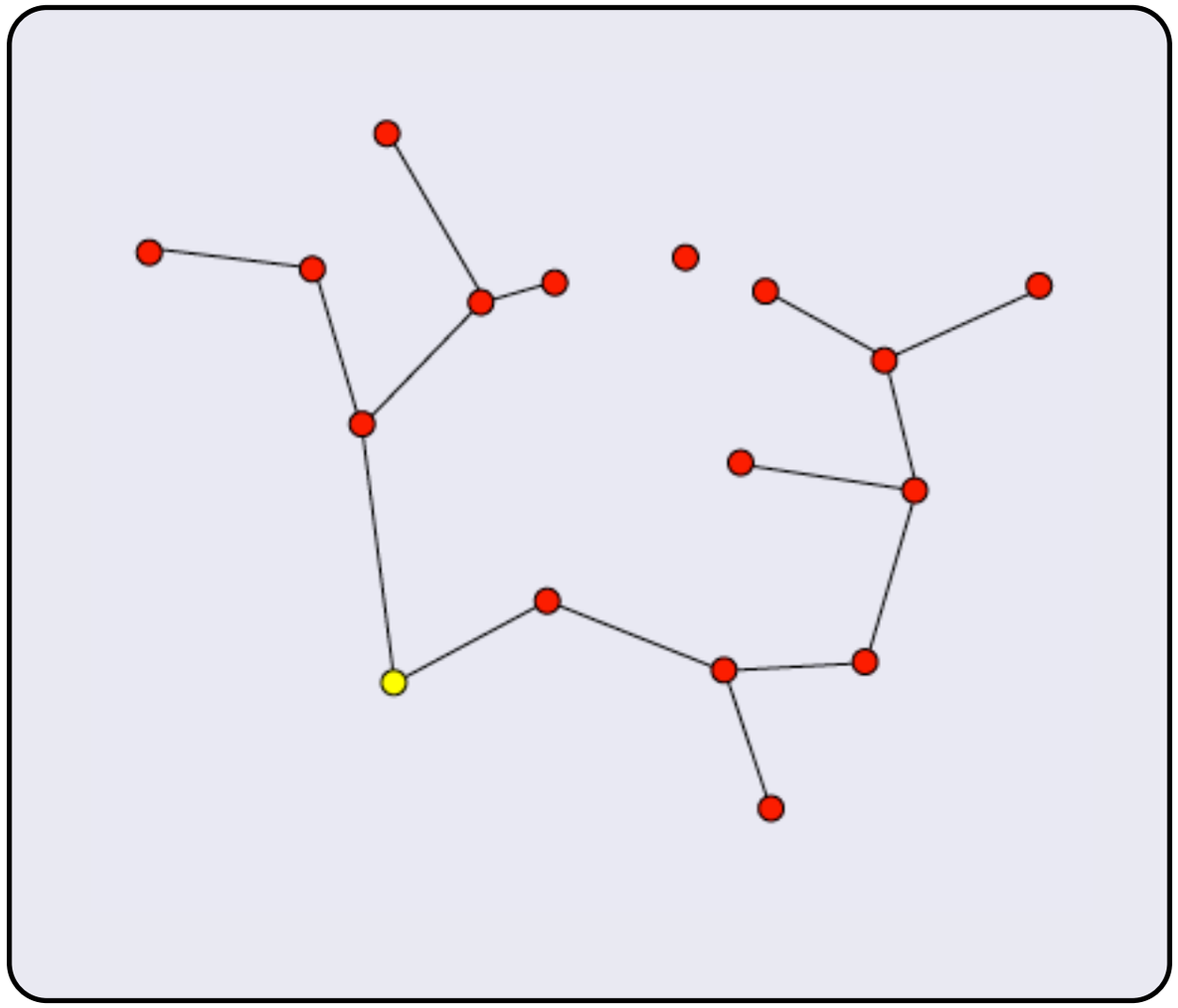
RRT*

- Frazzoli:
  - RRT is a variant of RRG that essentially “rewires" the tree as better paths are discovered.
  - After rewiring the cost has to be propagated along the leaves.

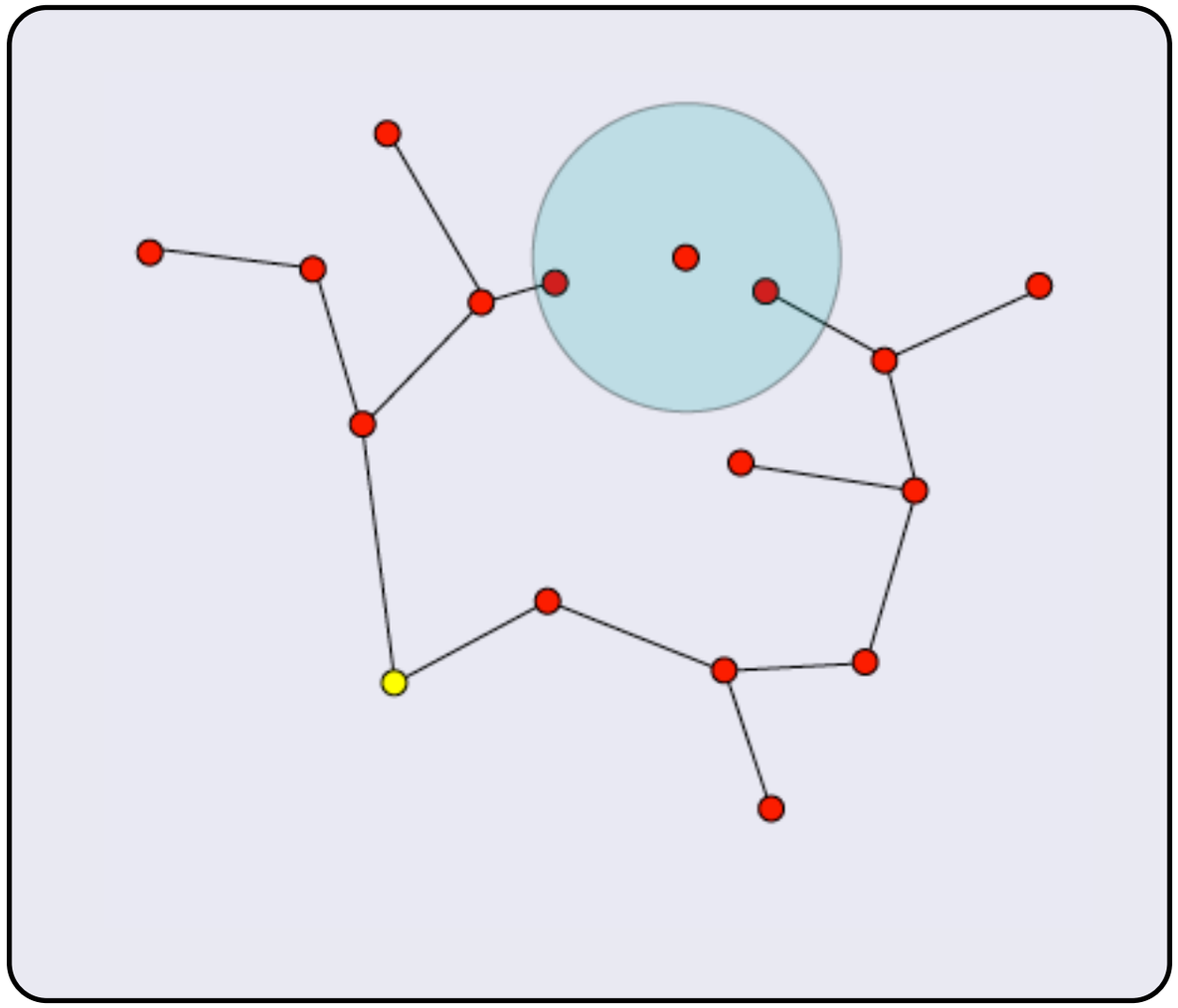
RRT*



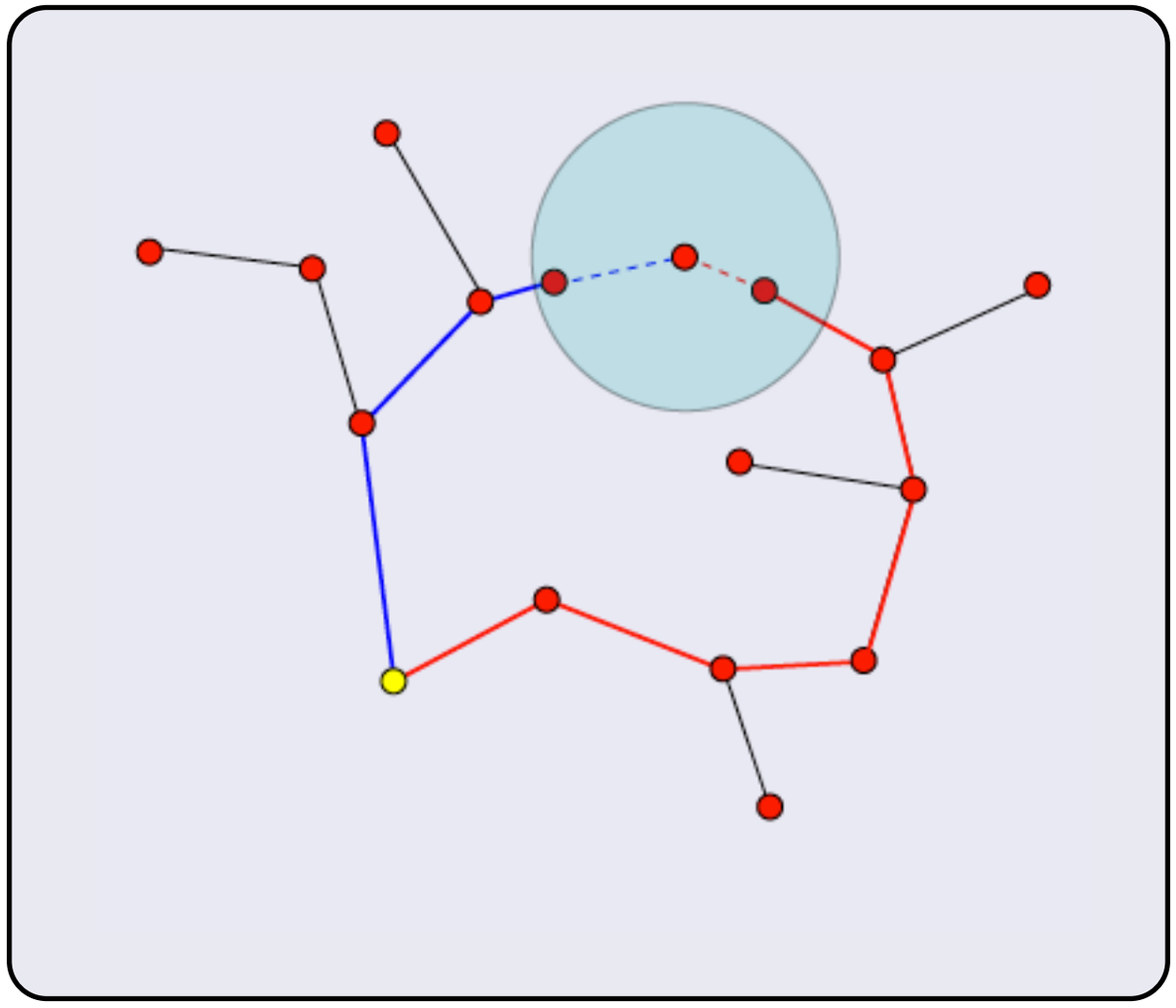
RRT*



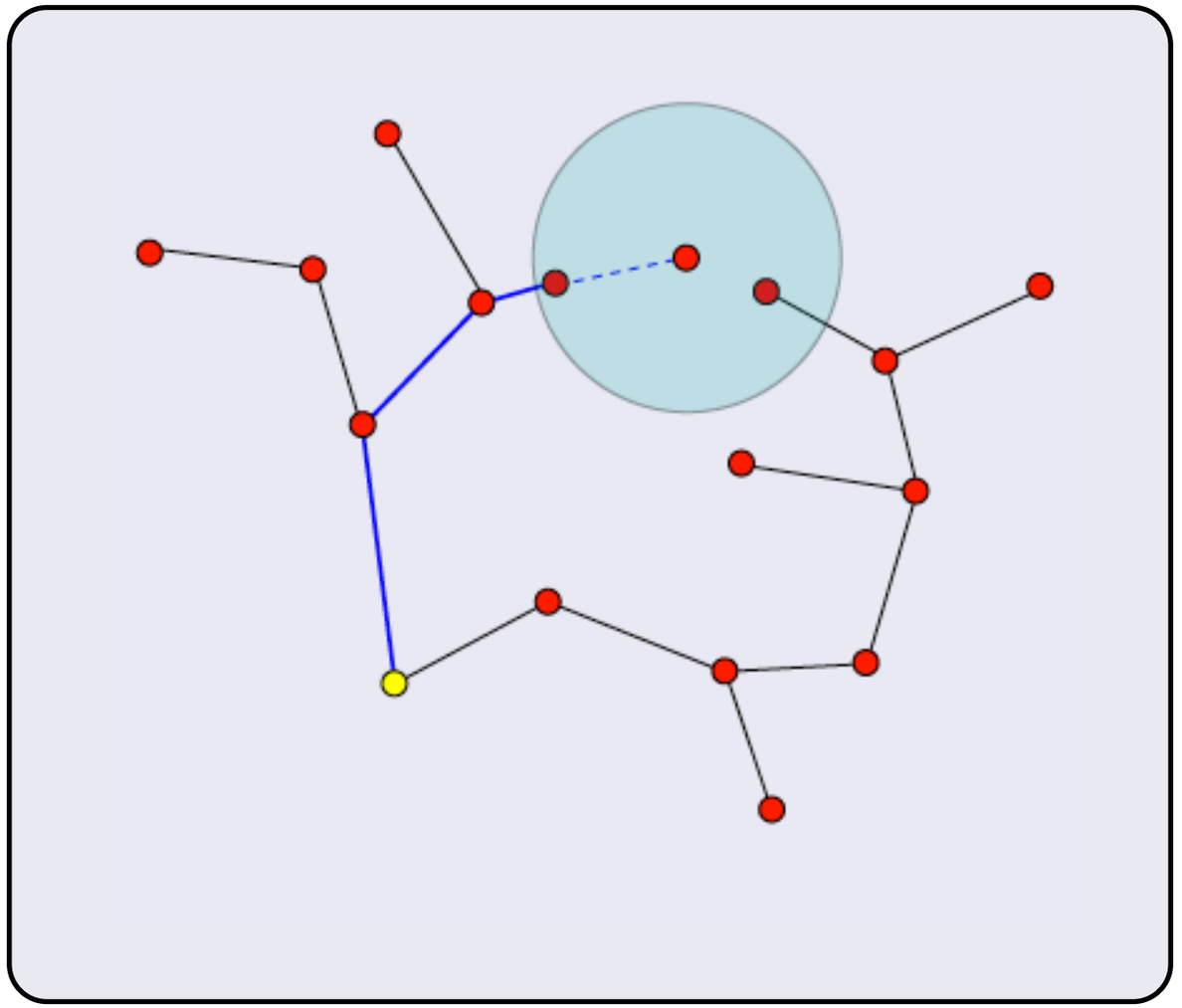
RRT*



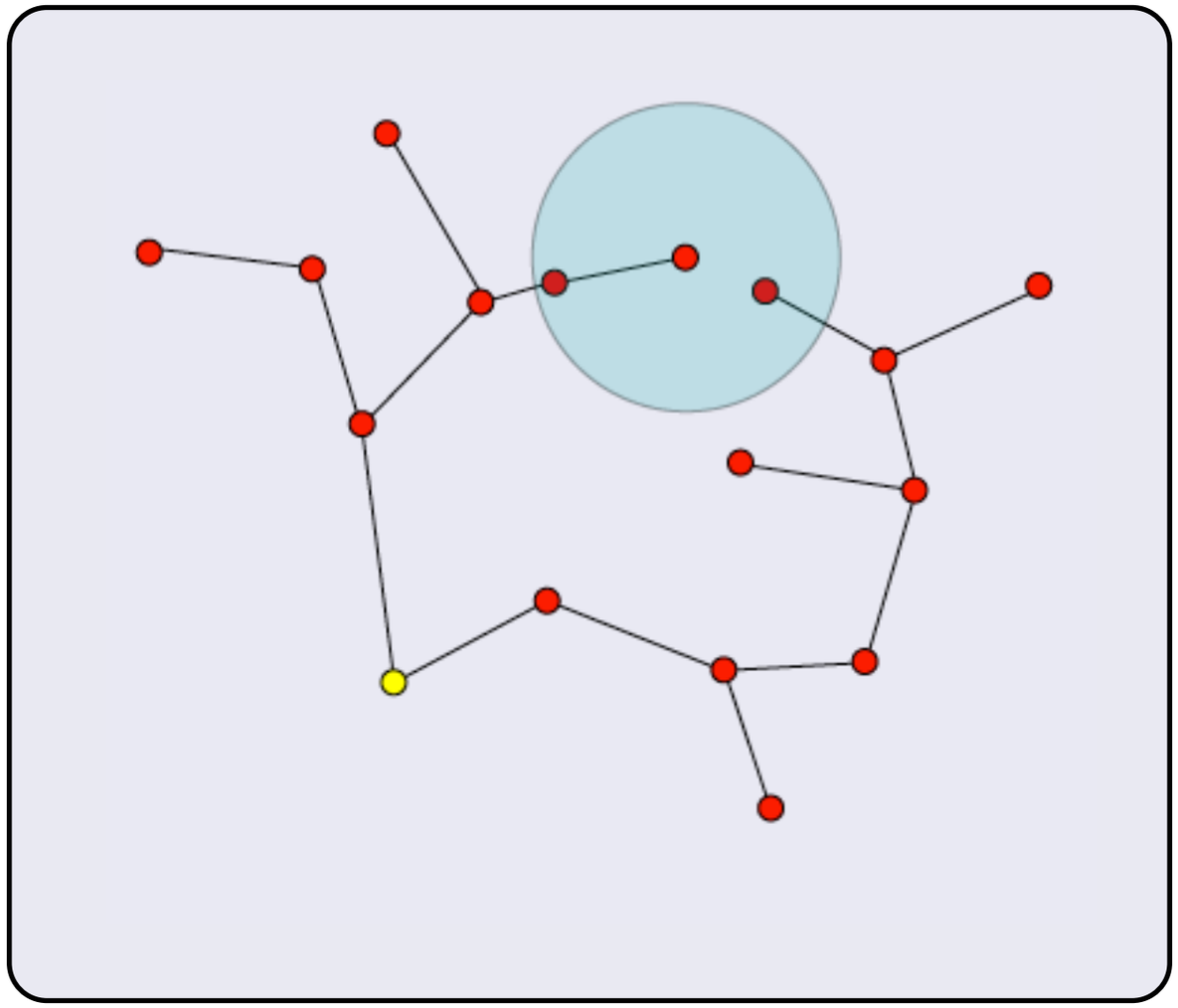
RRT*



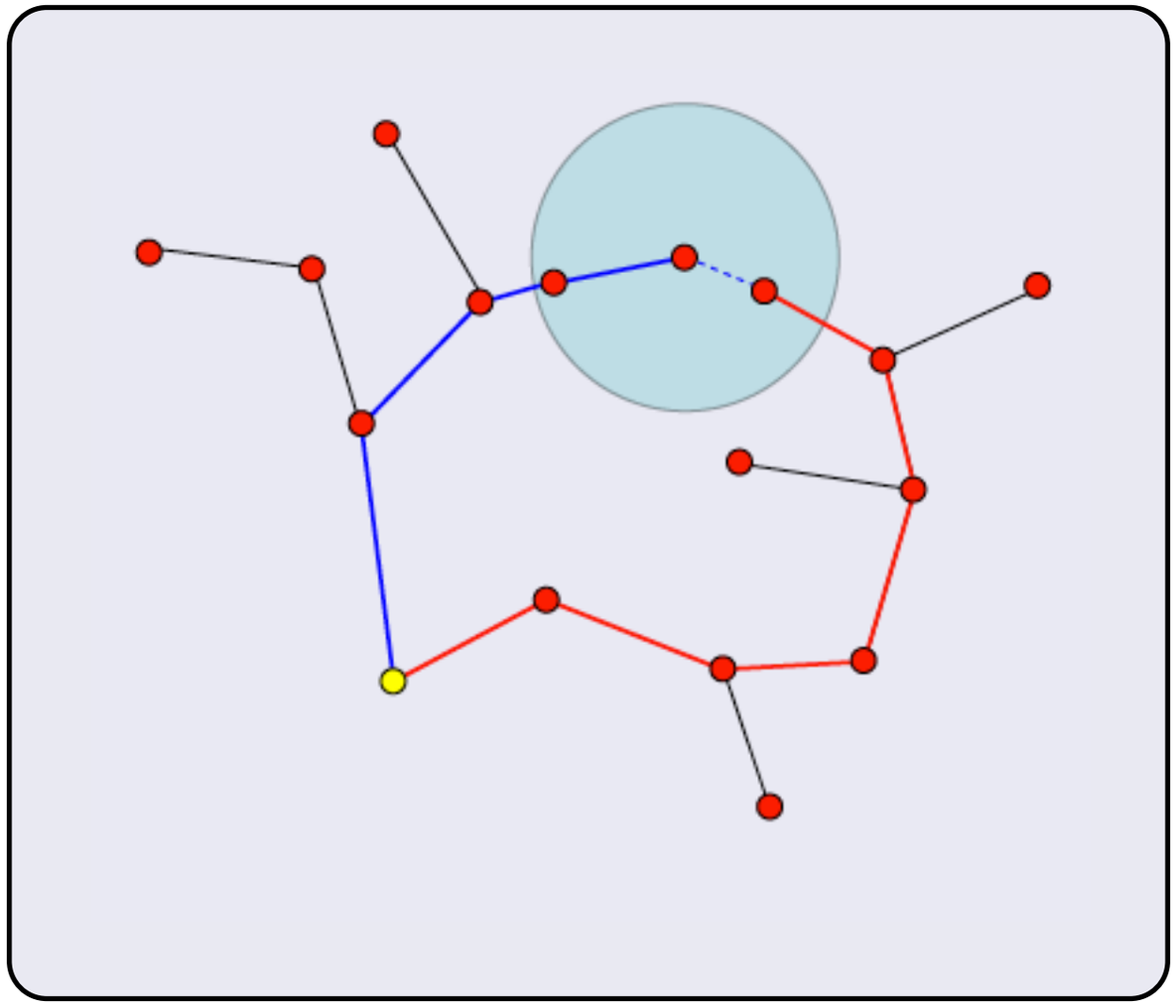
RRT*



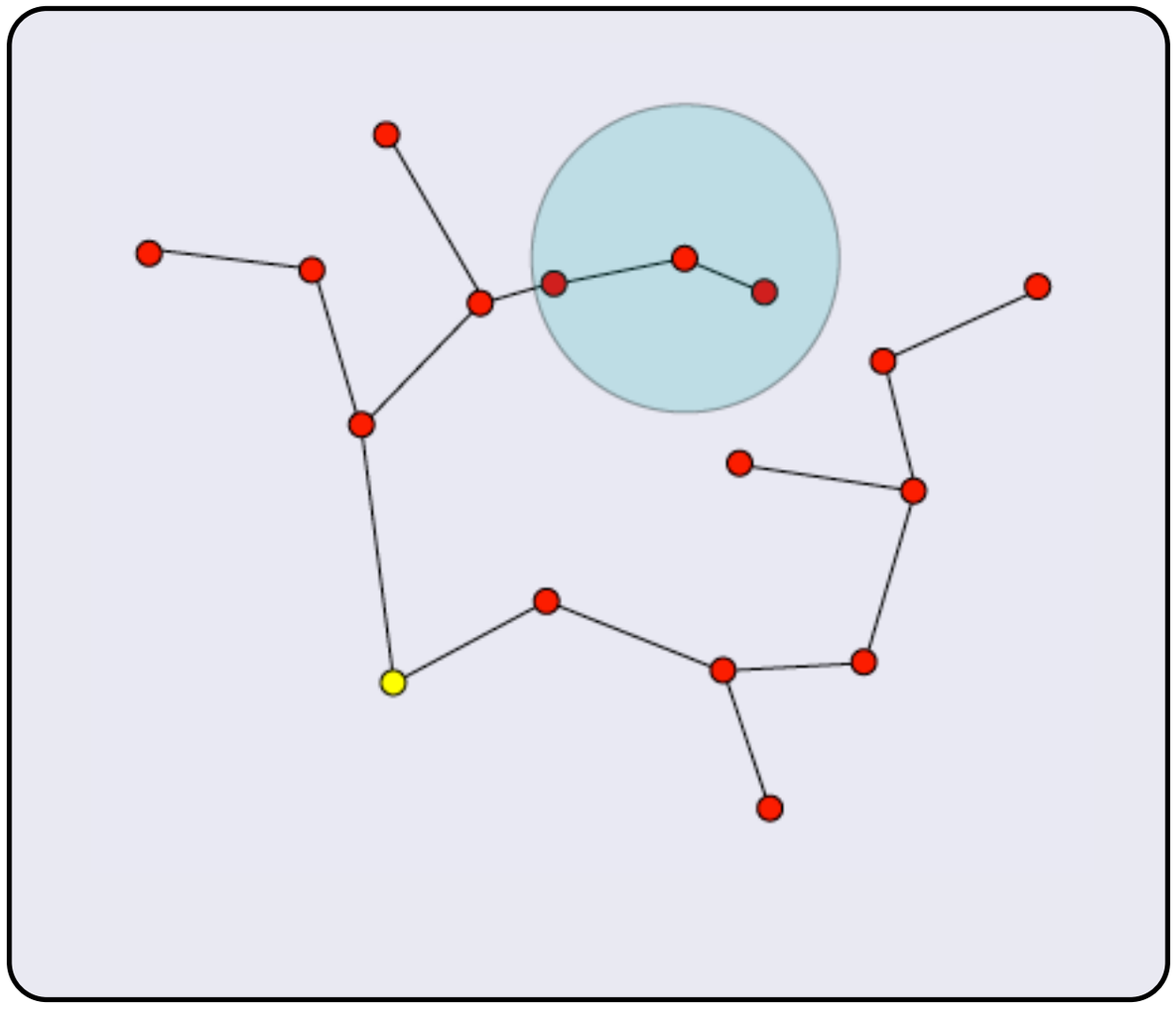
RRT*



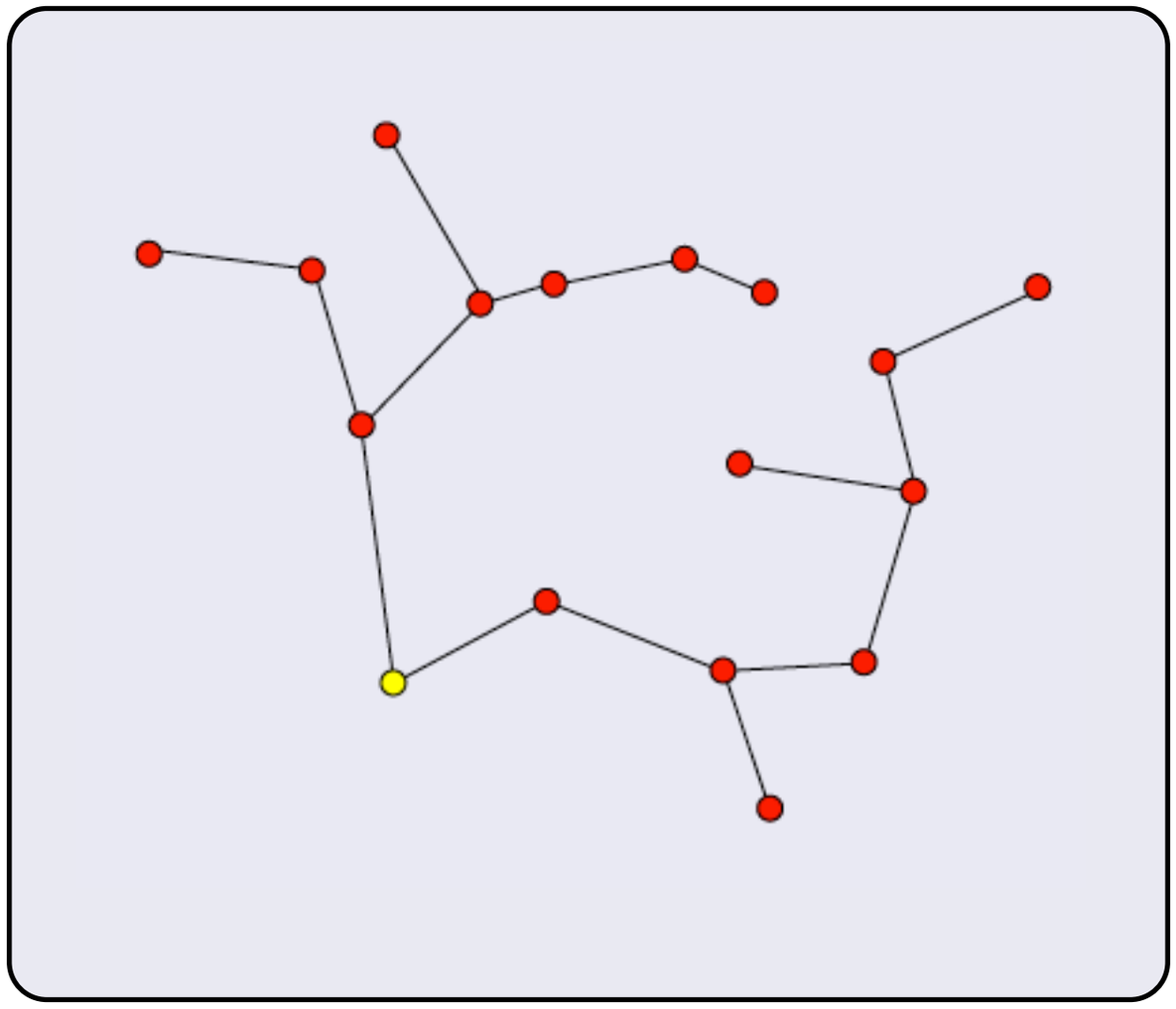
RRT*



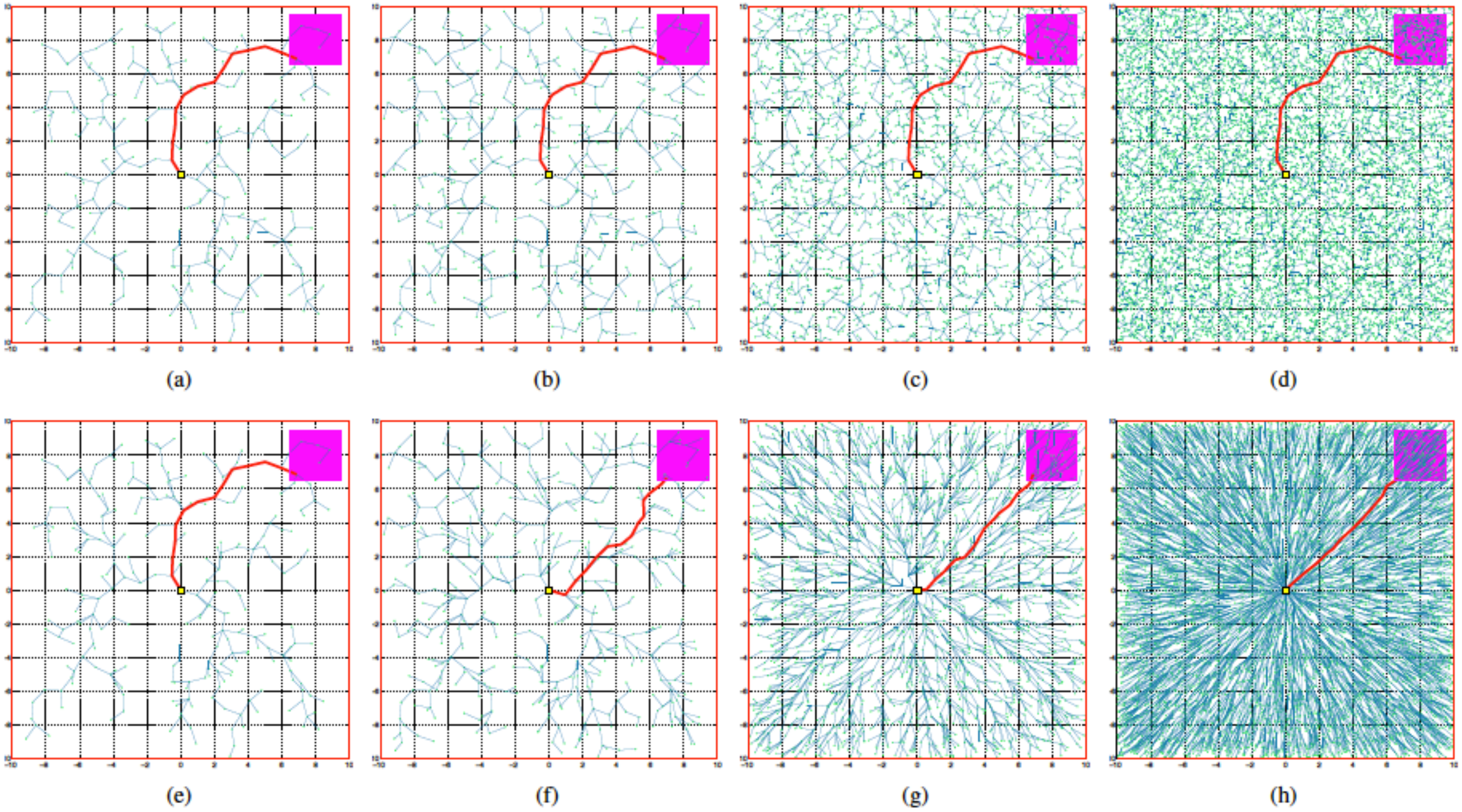
RRT*



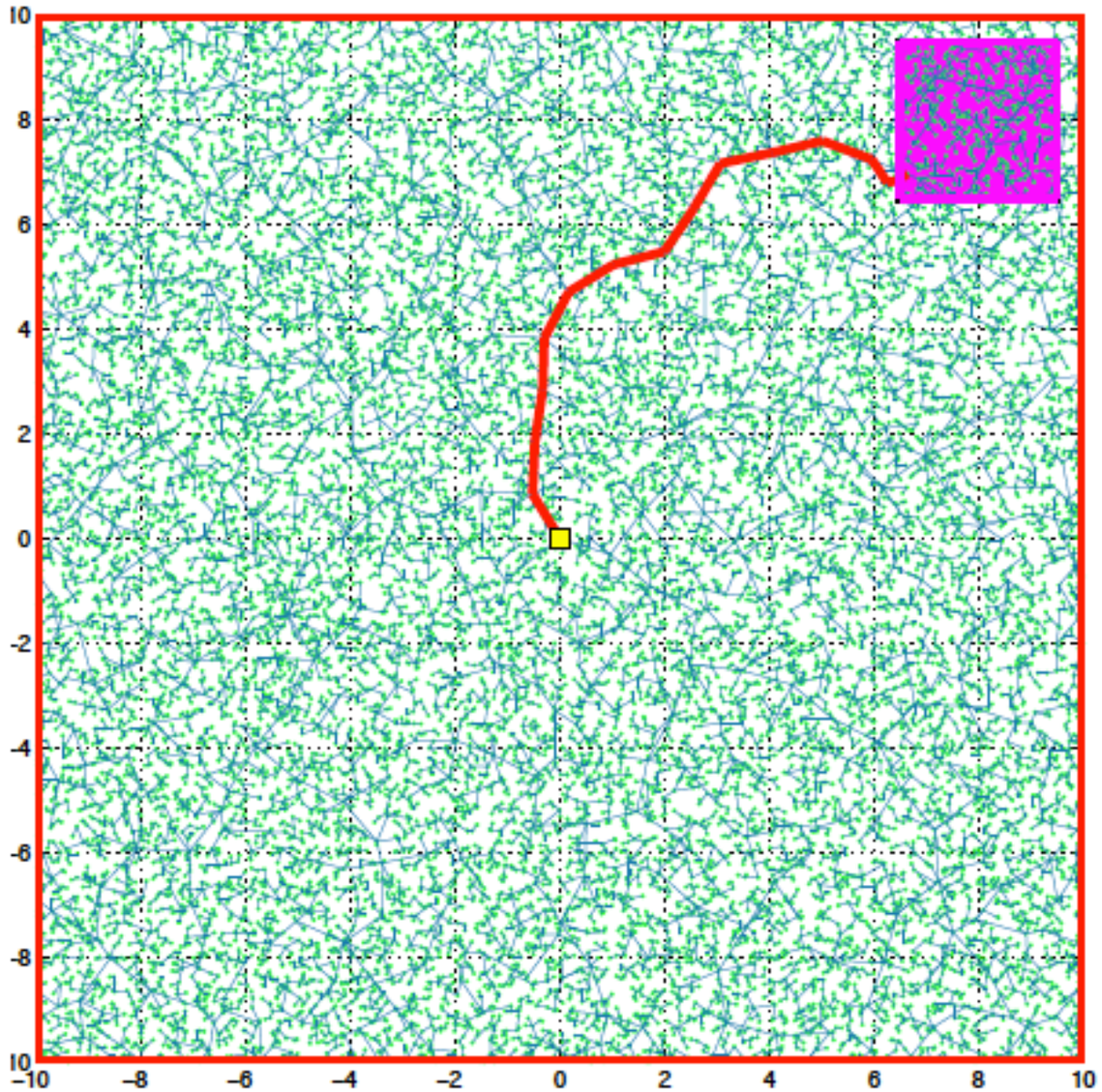
RRT*

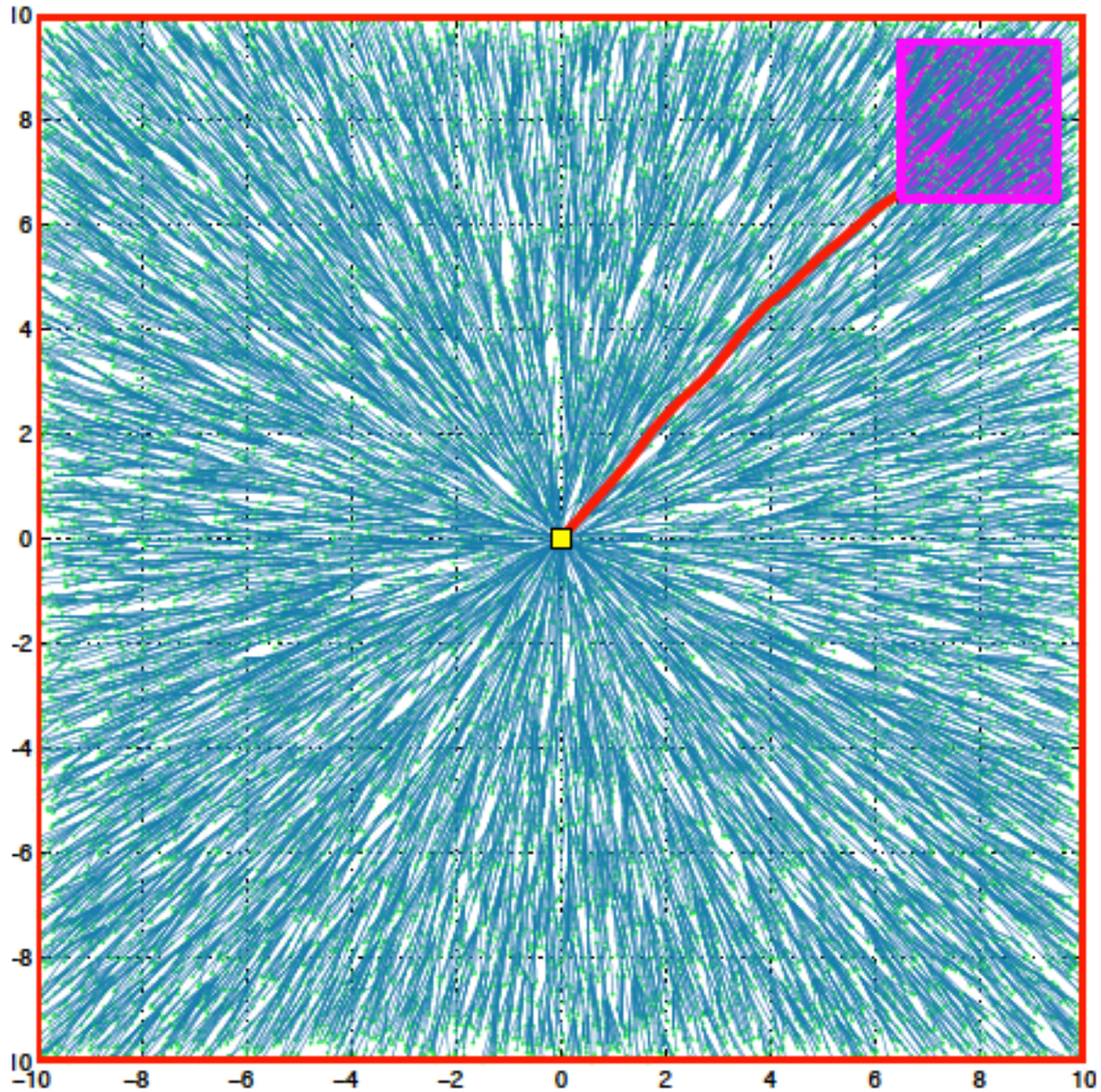


RRT vs RRT*

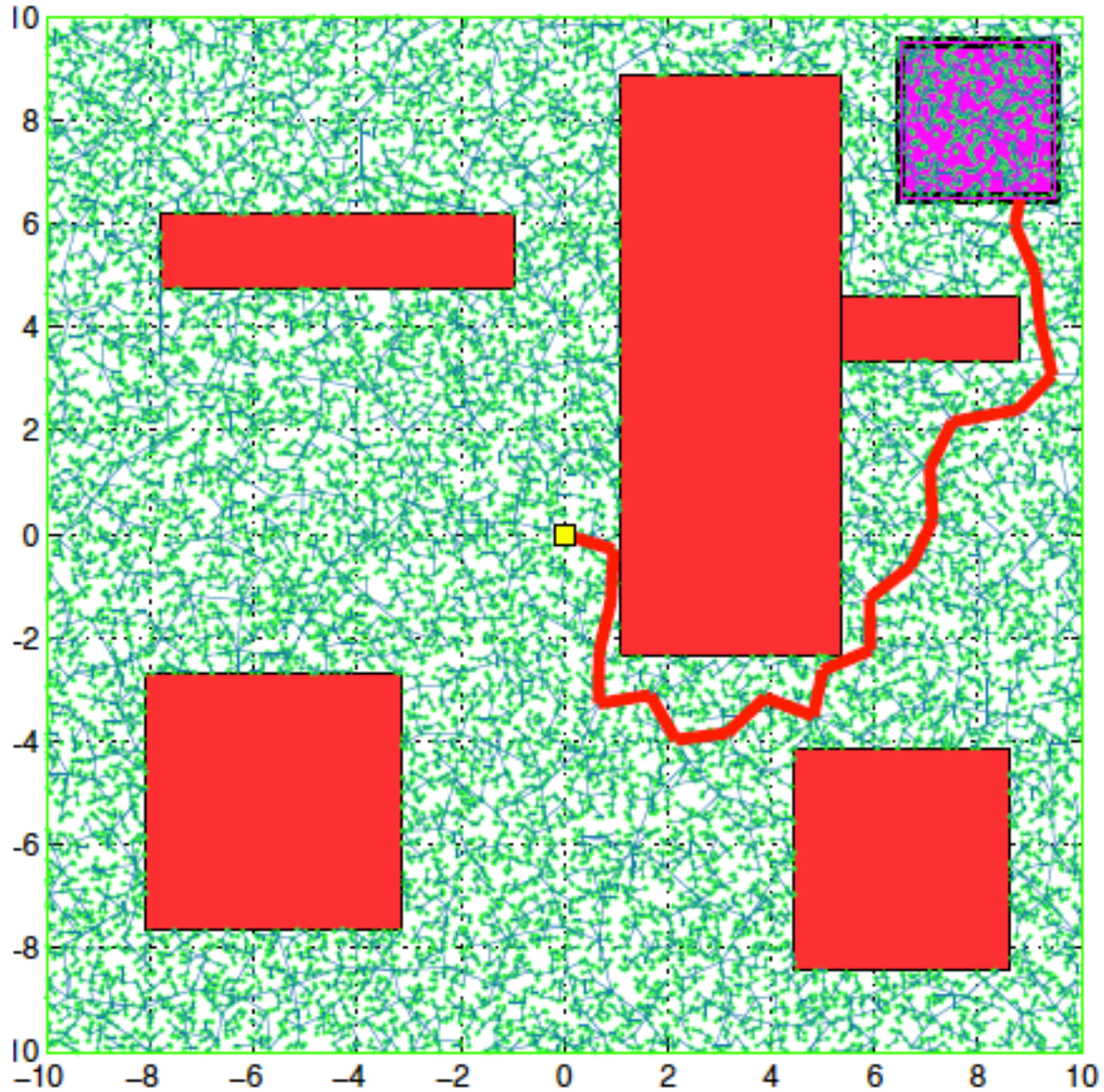


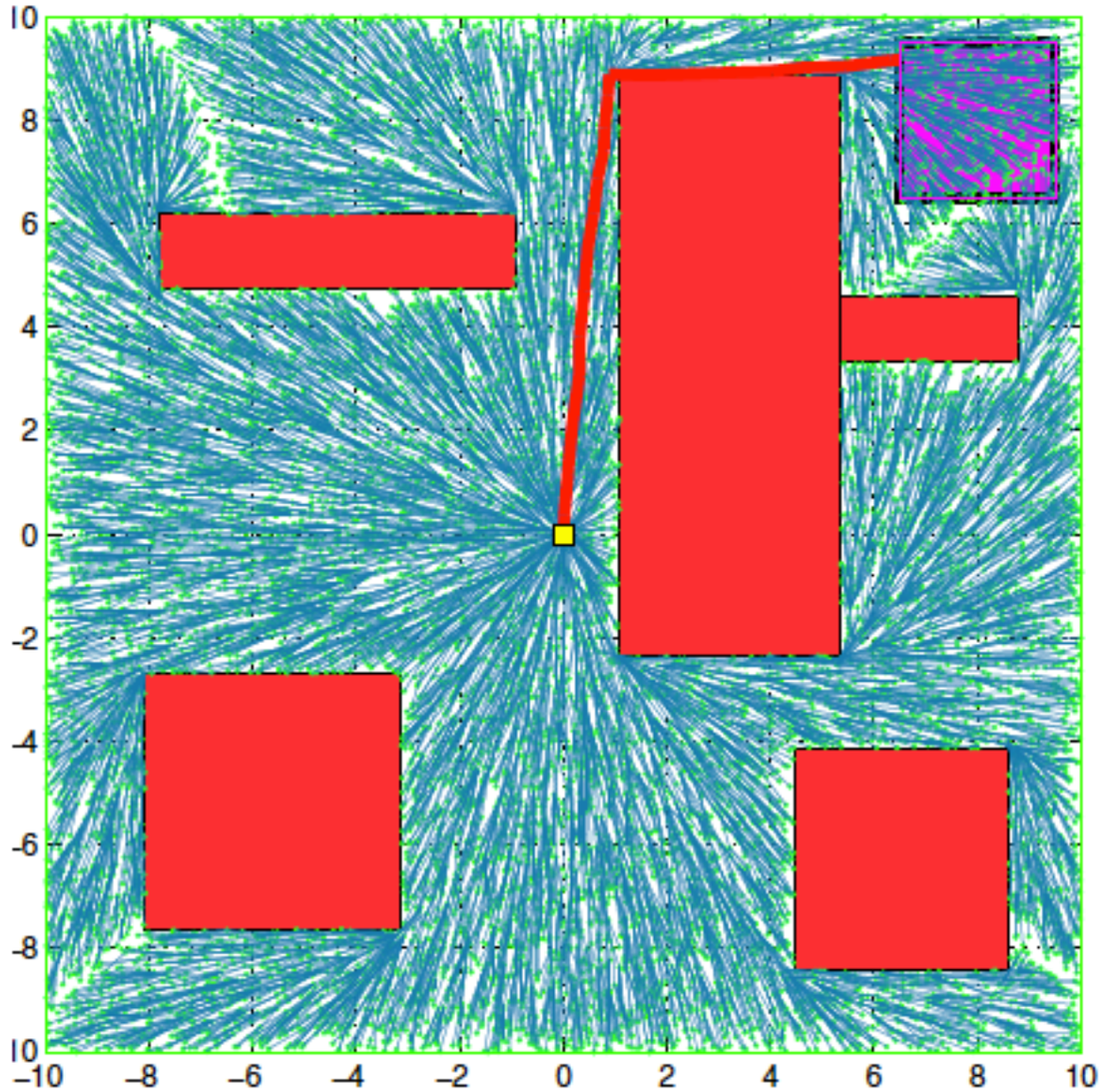
RRT vs RRT*

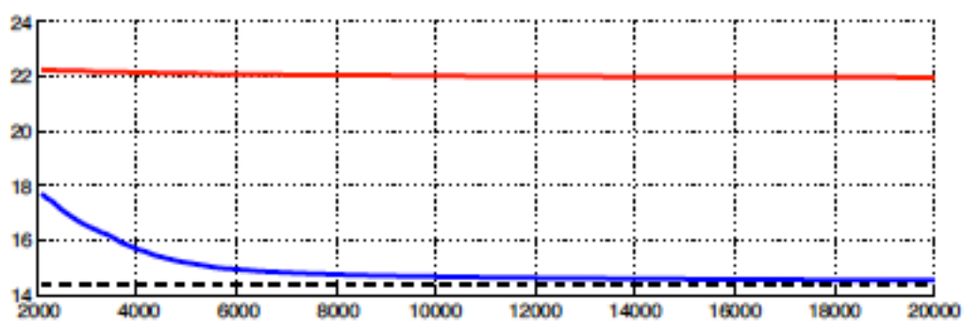

10
8
6
4
2
0
-2
-4
-6
-8
-10
−10 −8 −6 −4 −2 0 2 4 6 8 10


10
8
6
4
2
0
-2
-4
-6
-8
-10
−10 −8 −6 −4 −2 0 2 4 6 8 10

RRT* with Obstacles







Anytime RRT*

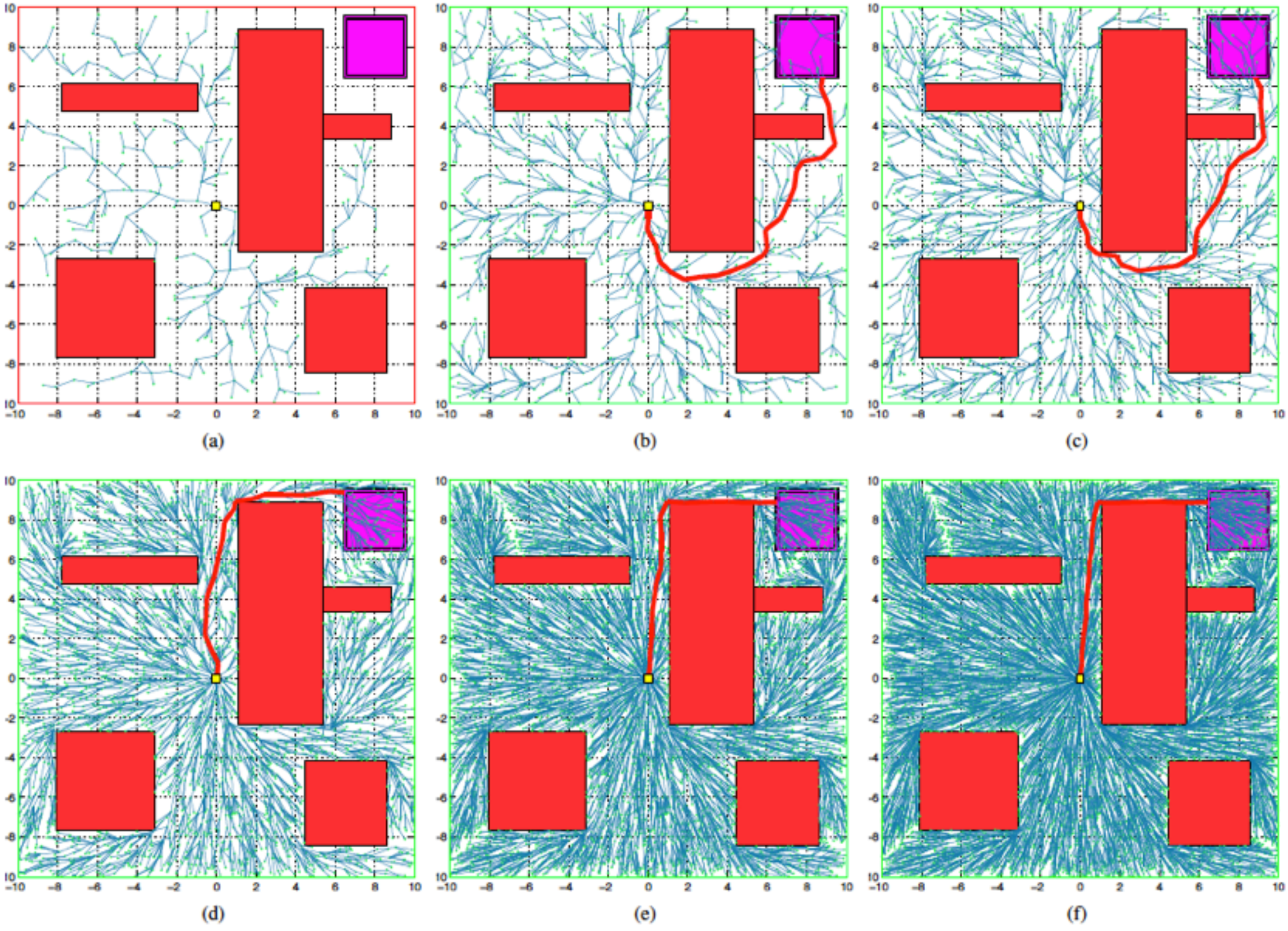
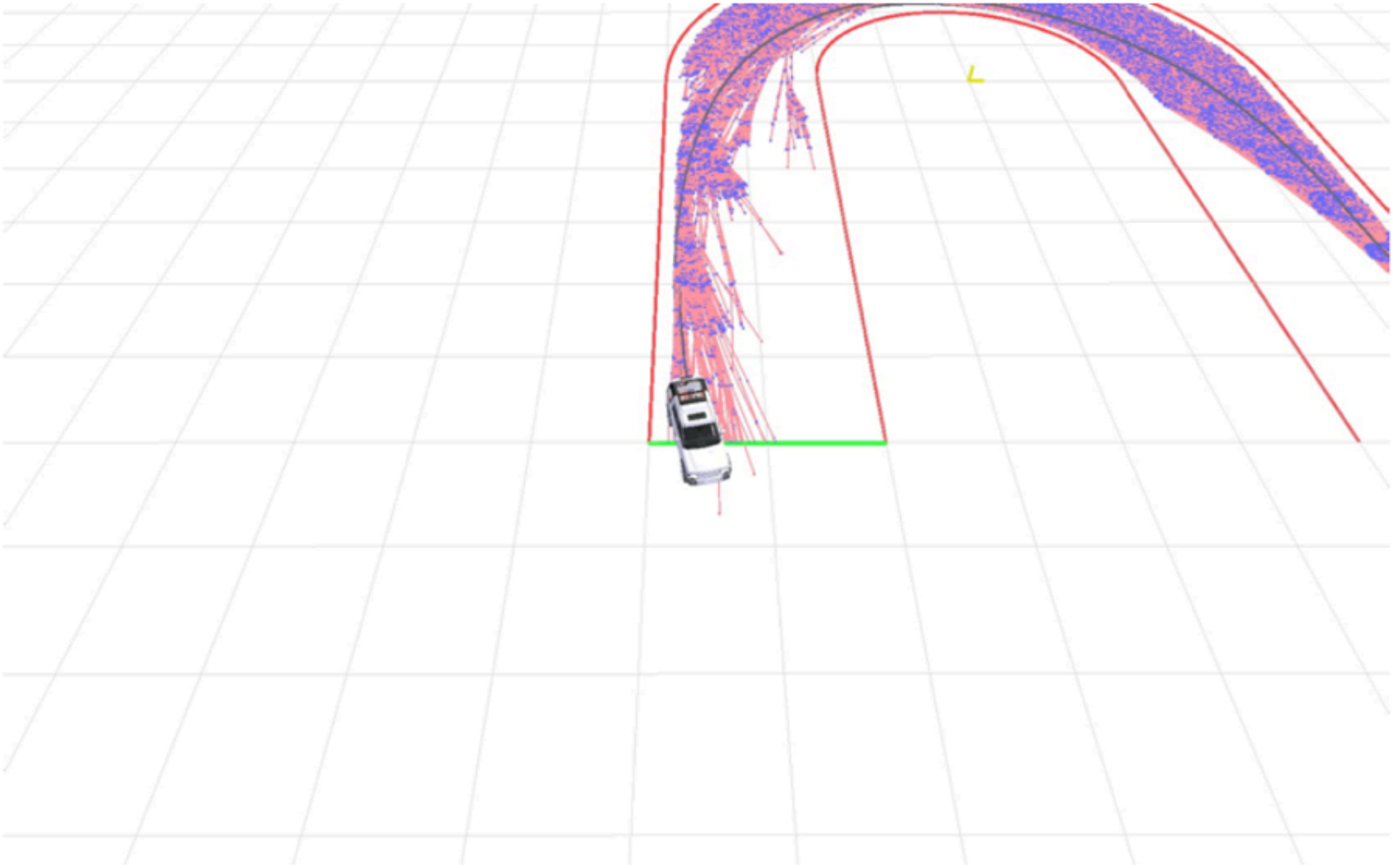


Fig. 4. RRT* algorithm shown after 500 (a), 1,500 (b), 2,500 (c), 5,000 (d), 10,000 (e), 15,000 (f) iterations.

Aggressive Driving

http://www.youtube.com/watch?v=Tdmm3i52WBc



Conclusion

- RPM: multiple-query, holonomic
- RRT: single-query, non-holonomic, suboptimal
- RRG, RRT*: single-query, holonomic, optimal