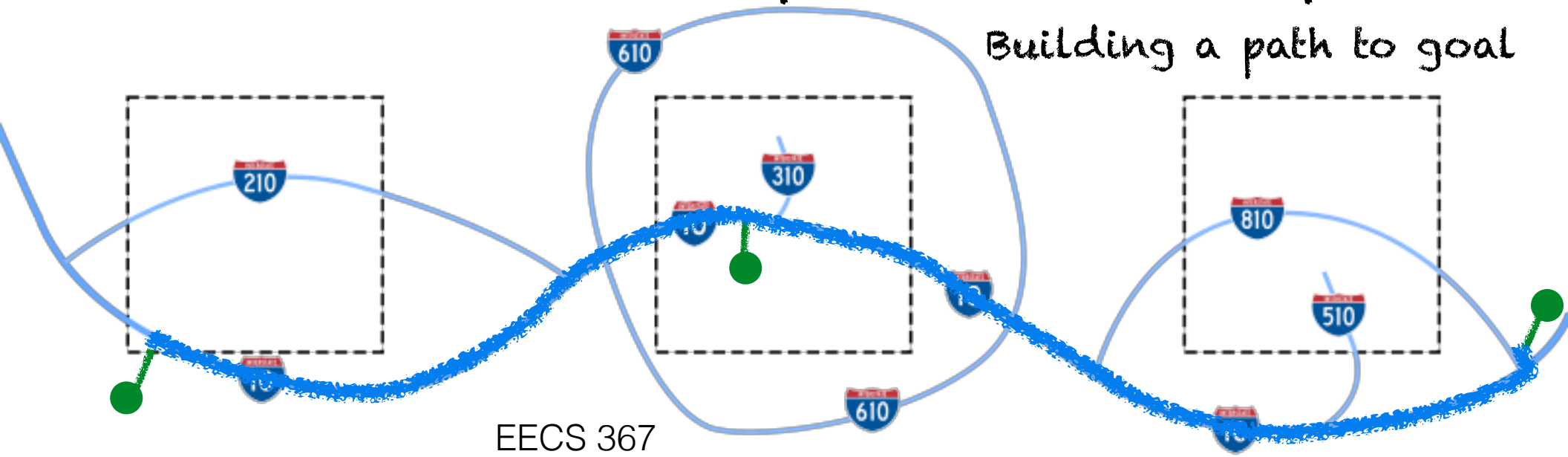
Planning with Sampling Roadmaps



EECS 367
Intro. to Autonomous Robotics

ME/EECS 567 ROB 510
Robot Modeling and Control

Fall 2019

autorob.org

Approaches to motion planning

- Bug algorithms: Bug[0-2], Tangent Bug
- Graph Search (fixed graph)
  - Depth-first, Breadth-first, Dijkstra, A-star, Greedy best-first
- **Sampling-based Search (build graph):**
  - **Probabilistic Road Maps, Rapidly-exploring Random Trees**
- Optimization and local search:
  - Gradient descent, Potential fields, Simulated annealing, Wavefront

Where is this?

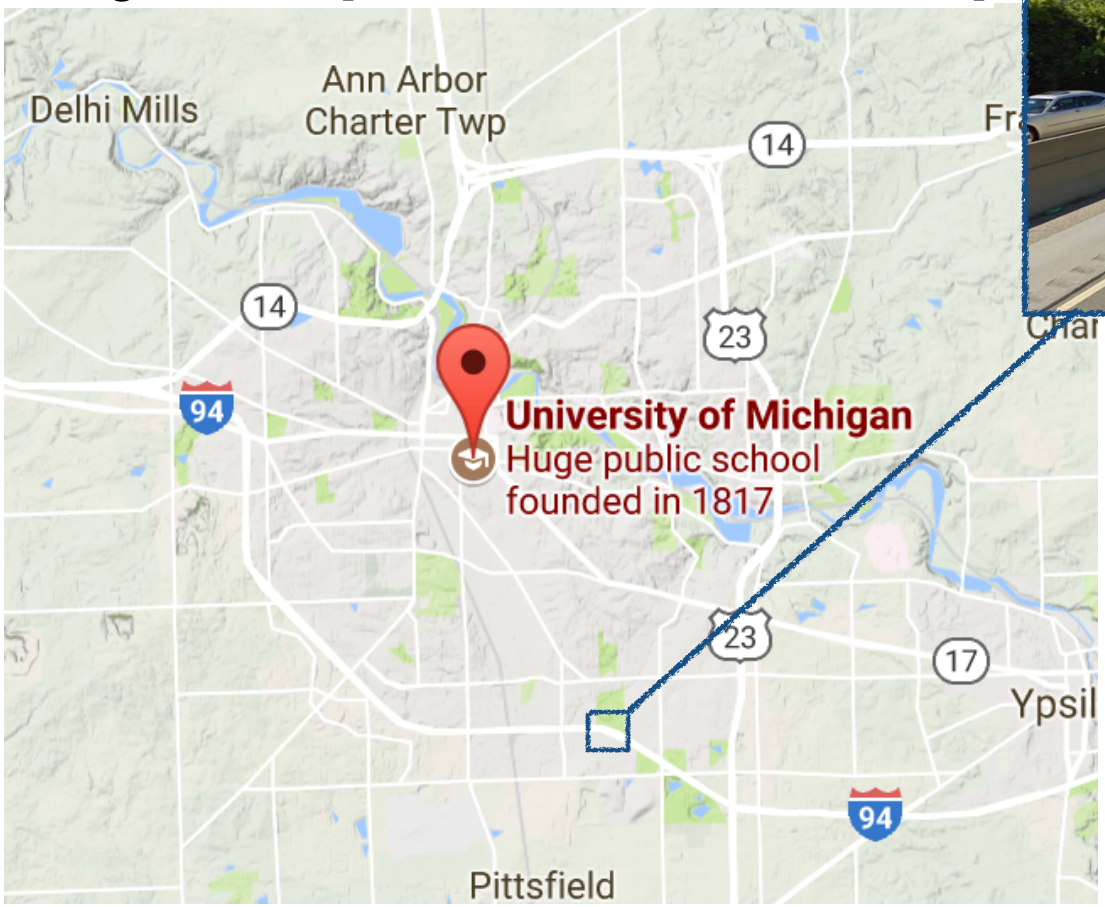

EXIT 177
State St
1 MILE
I-94

What is the configuration of this location?

What is the configuration of this location?

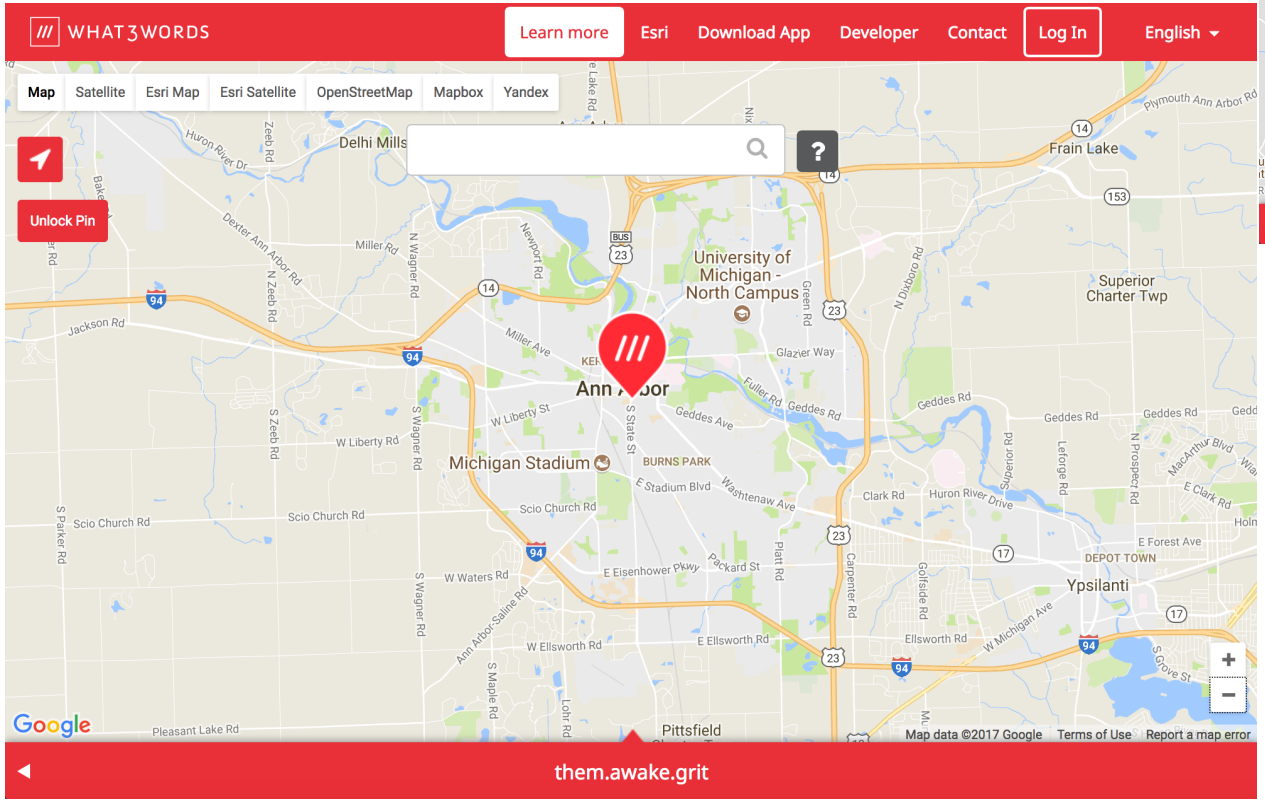
Configuration (lat,lon): [42.237631, -83.7147289]

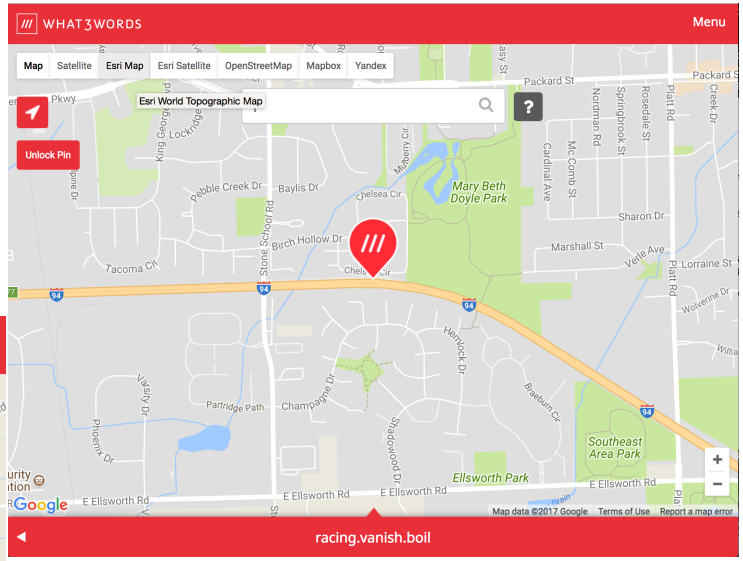
Configuration: [42.2780436,-83.7404128]



Alternatively...

Configuration: [them.awake.grit]

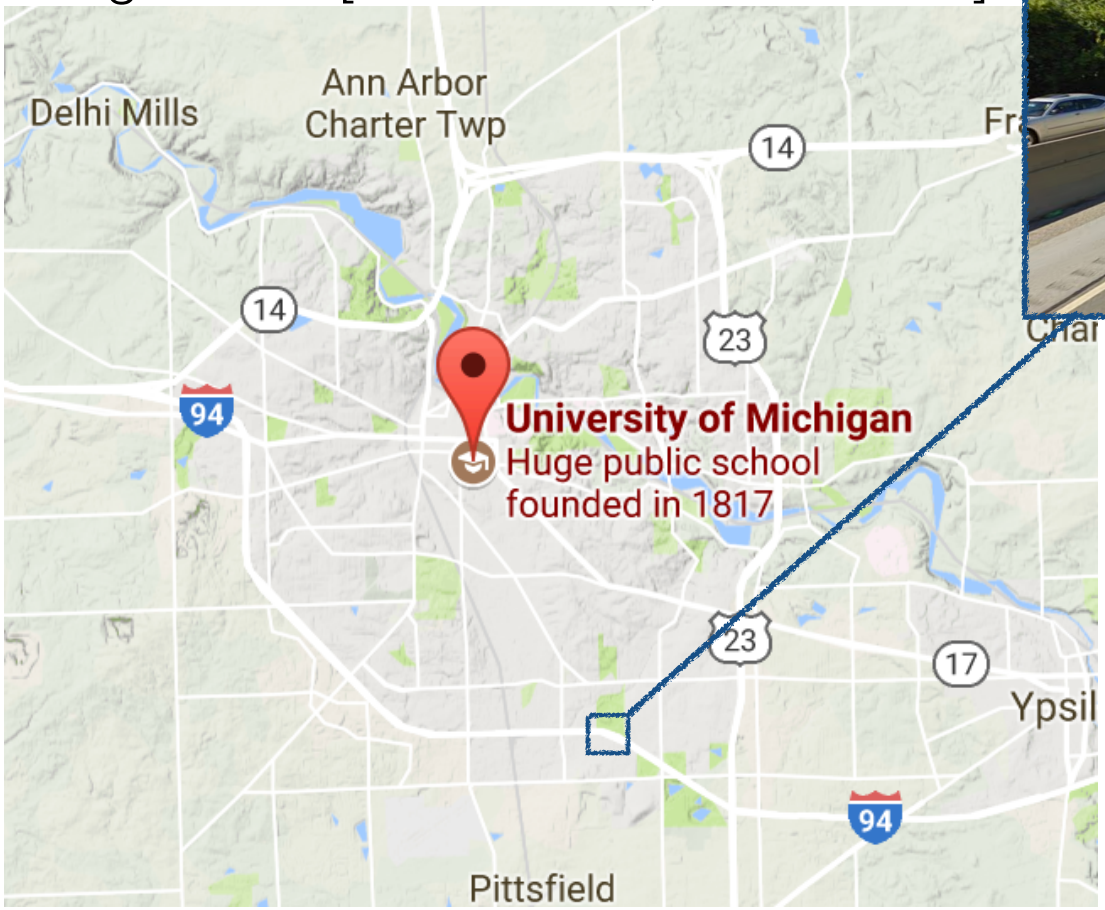


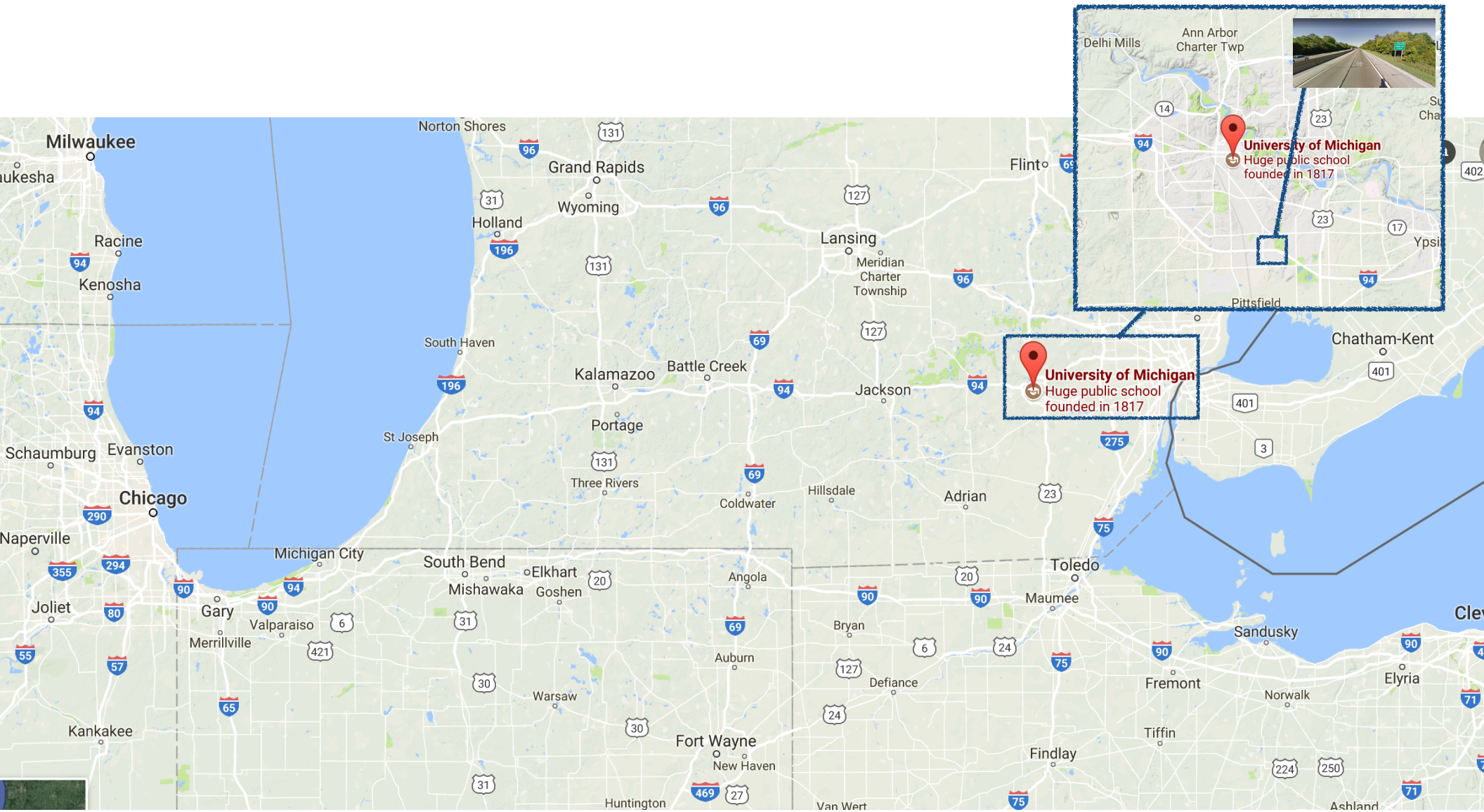


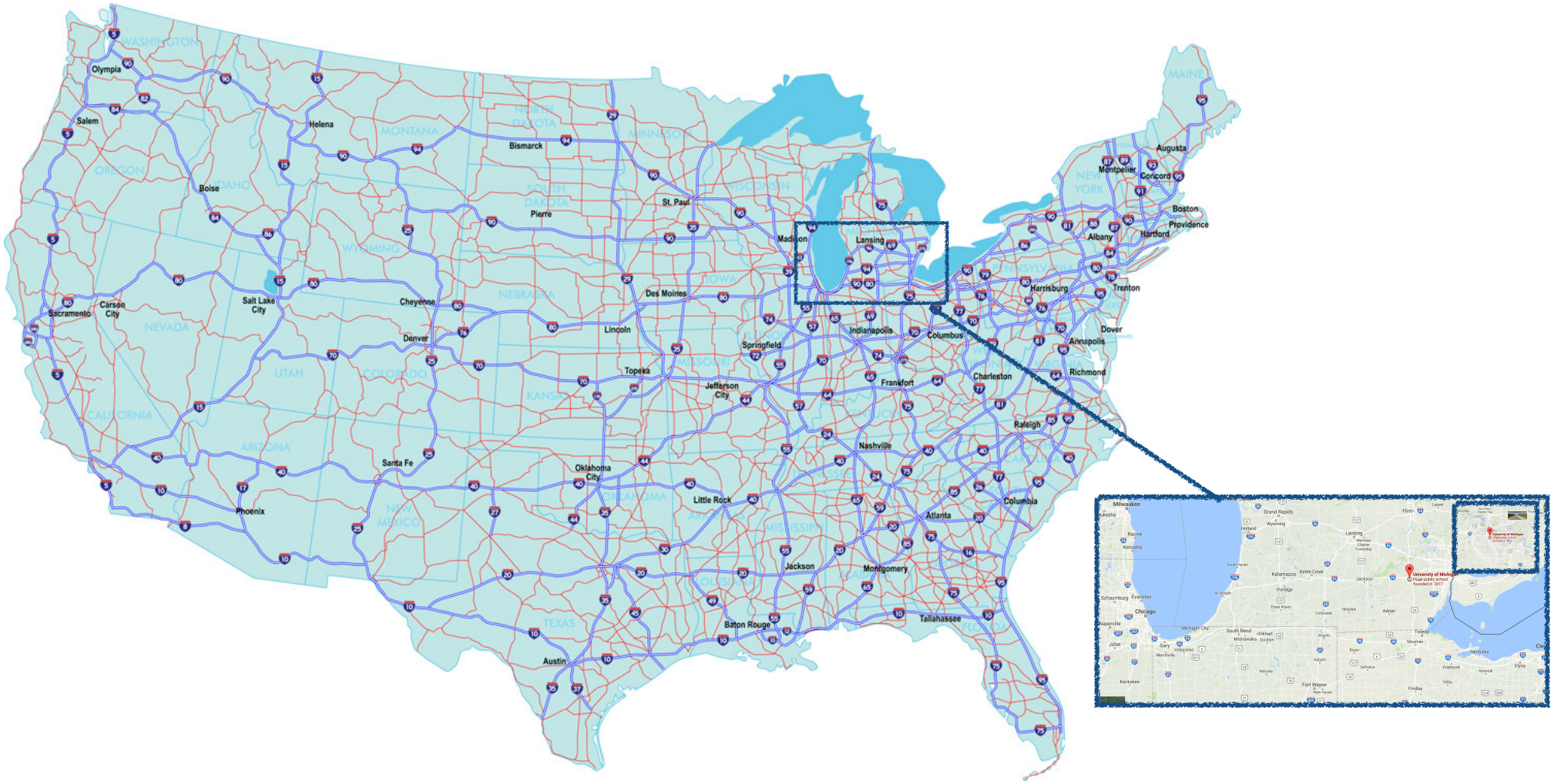
Configuration: [racing.vanish.boil]

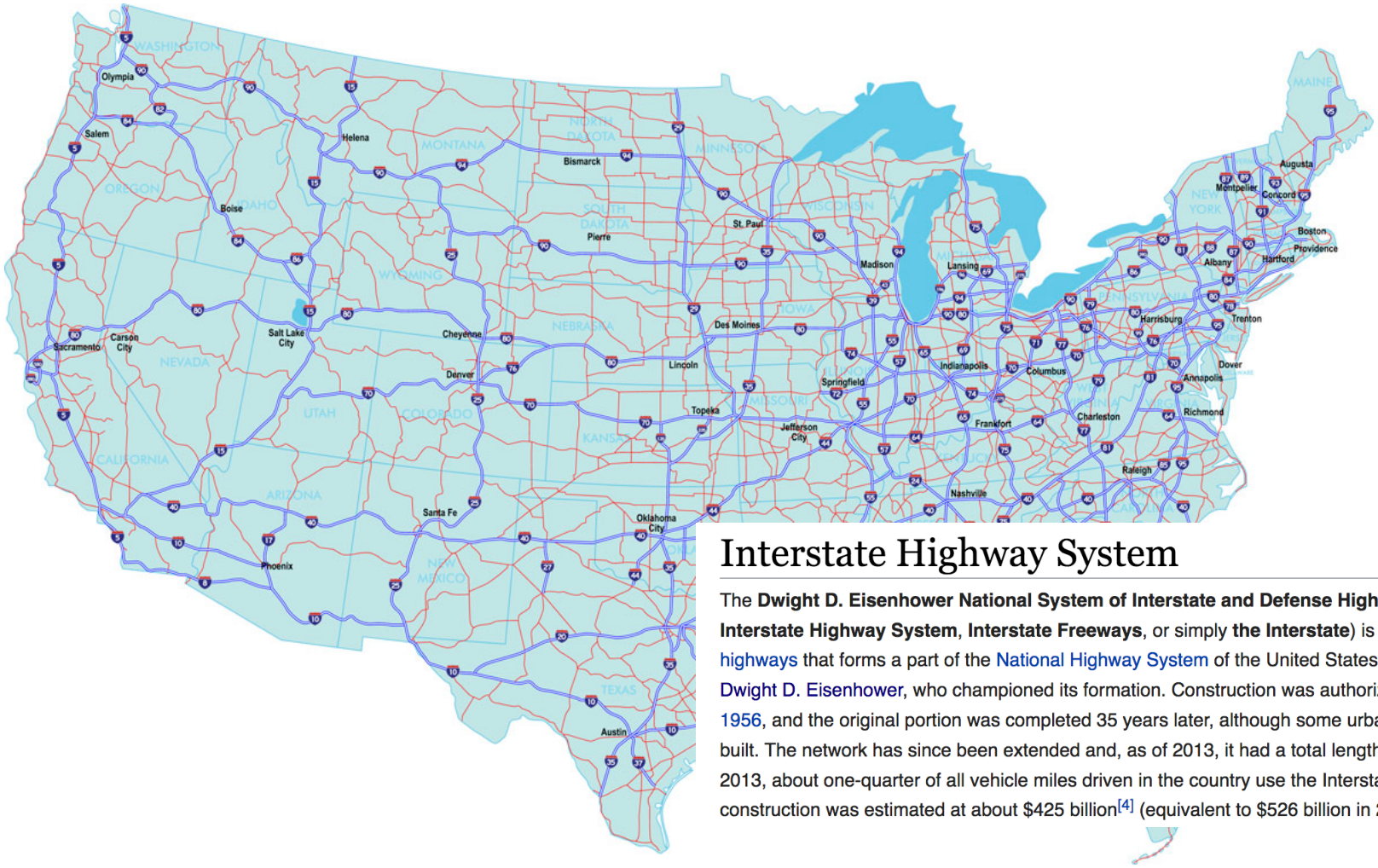
Is this a usable representation for configuration?

Configuration: [42.2780436,-83.7404128]




Milwaukee
Racine
Kenosha
Norton Shores
Grand Rapids
Wyoming
Holland
South Haven
Kalamazoo
Battle Creek
Portage
St Joseph
Three Rivers
Coldwater
Hillsdale
Jackson
Lansing
Meridian Charter Township
Flint
Pittsfield
Adrian
Toledo
Maumee
Bryan
Defiance
Fremont
Sandusky
Norwalk
Elyria
Tiffin
Findlay
Chatham-Kent
Ann Arbor Charter Twp
Delhi Mills
University of Michigan
Huge public school founded in 1817
University of Michigan
Huge public school founded in 1817
Schaumburg
Evanston
Chicago
Naperville
Joliet
Gary
Valparaiso
Merrillville
Michigan City
South Bend
Mishawaka
Elkhart
Goshen
Angola
Auburn
Warsaw
Fort Wayne
New Haven
Huntington
Kankakee

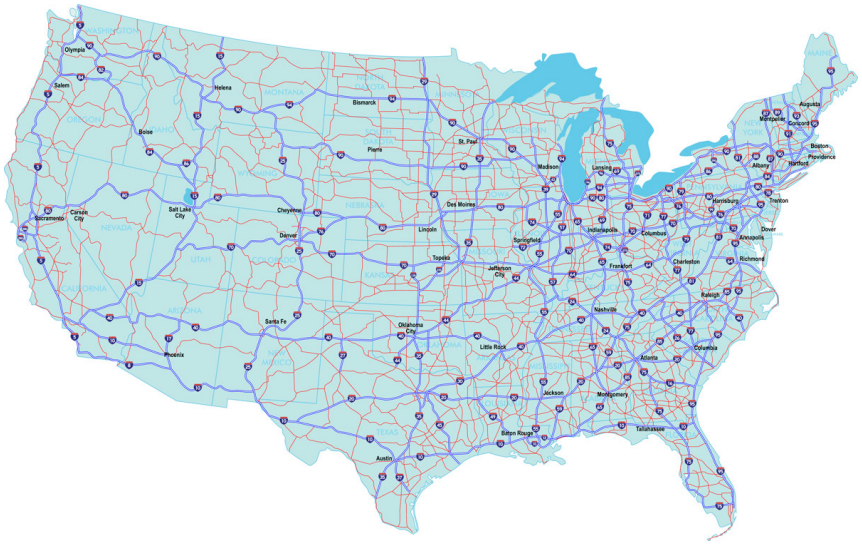




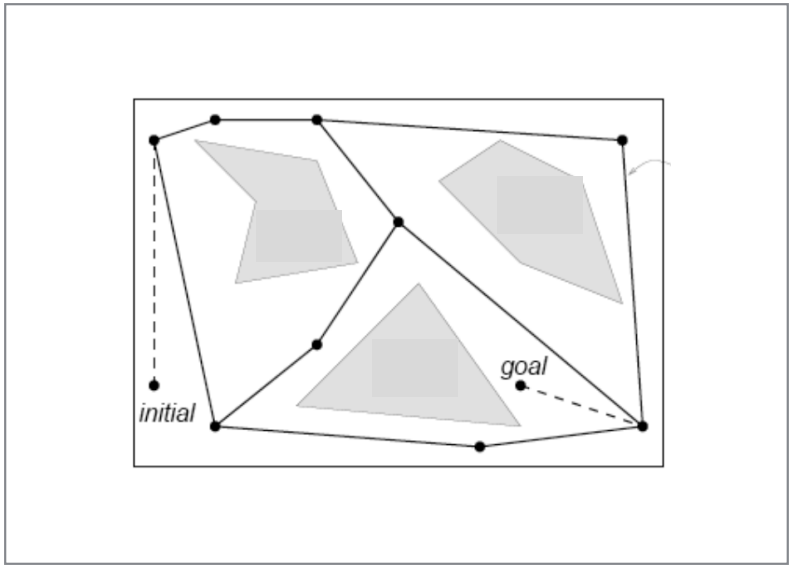
Interstate Highway System

The **Dwight D. Eisenhower National System of Interstate and Defense Highways** (commonly known as the **Interstate Highway System**, **Interstate Freeways**, or simply **the Interstate**) is a network of controlled-access highways that forms a part of the National Highway System of the United States. The system is named for President Dwight D. Eisenhower, who championed its formation. Construction was authorized by the Federal Aid Highway Act of 1956, and the original portion was completed 35 years later, although some urban routes were cancelled and never built. The network has since been extended and, as of 2013, it had a total length of 47,856 miles (77,017 km).[2] As of 2013, about one-quarter of all vehicle miles driven in the country use the Interstate system.[3] In 2006, the cost of construction was estimated at about $425 billion[4] (equivalent to $526 billion in 2016[5]).

Roadmaps

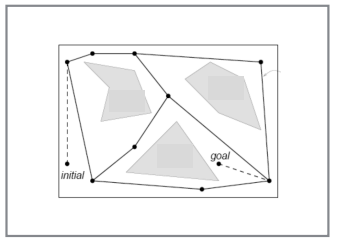


Roadmap over geolocations

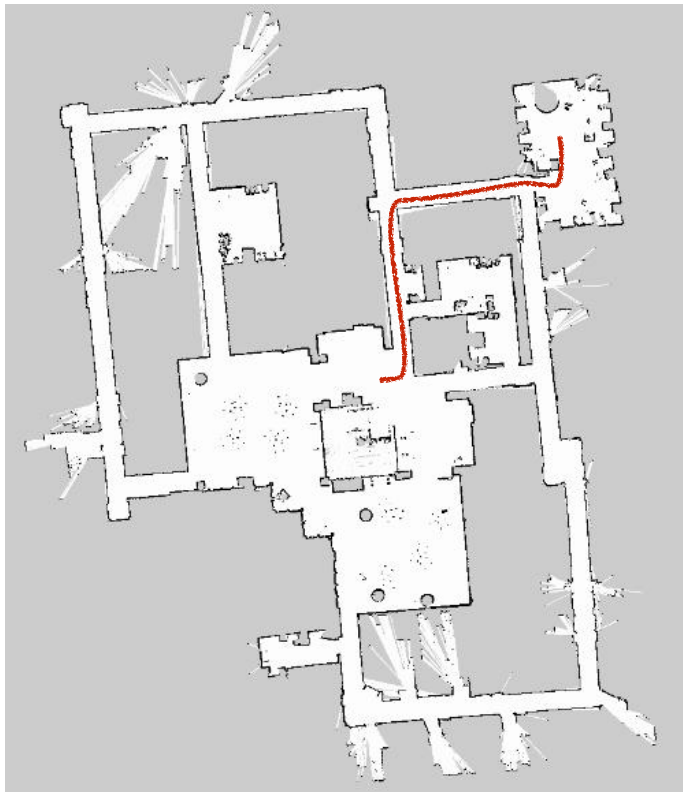


Roadmap over robot configurations

Roadmaps

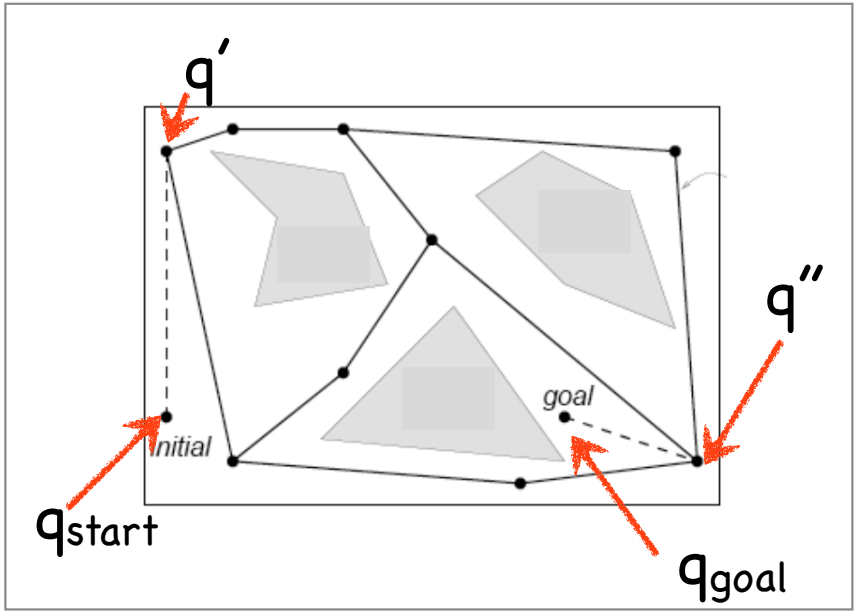


- Graph search assumed C-space as a fixed uniform grid
  - finite set of discretized cells
- How does this scale beyond planar navigation?
  - curse of dimensionality
- Roadmaps are a more general notion of graphs in C-space



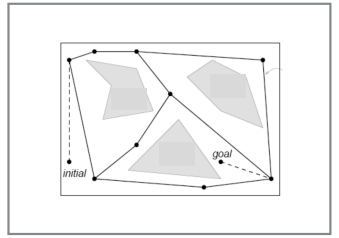
Roadmap Definition

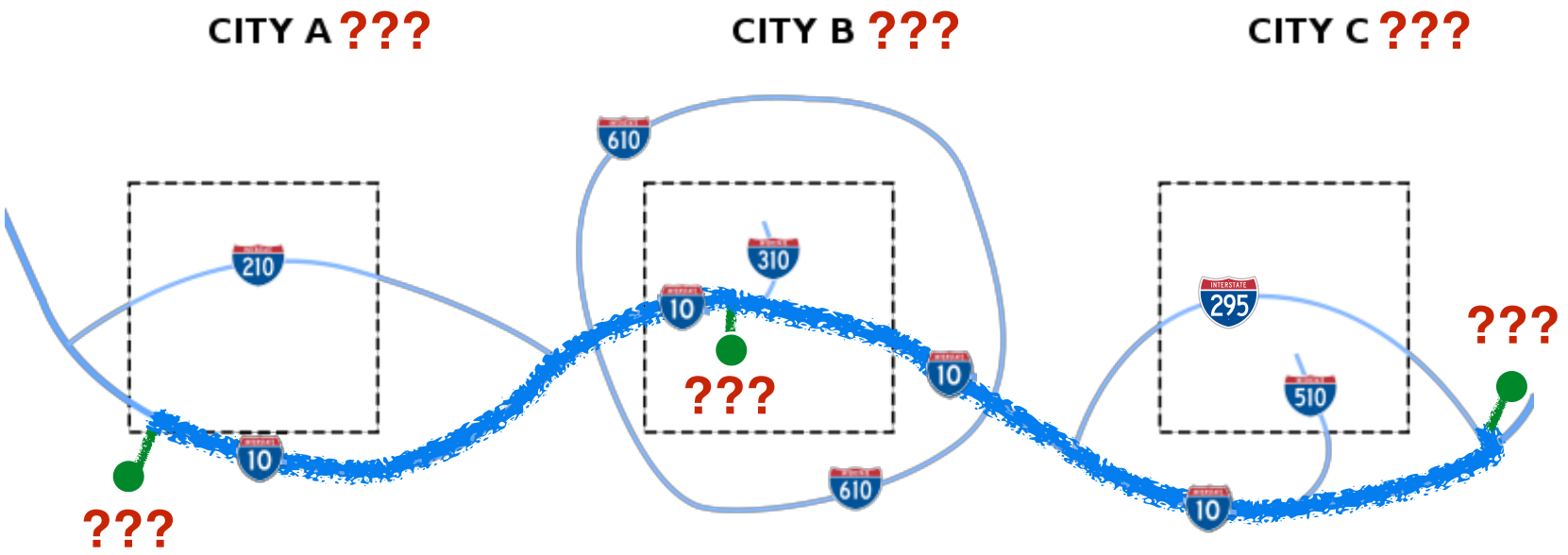
- A roadmap *RM* is a union of curves s.t. all start and goal points in C-space (Q_{free}) can be connected by a path



- Roadmap properties:
  - **Accessibility**: There is a path from $q_{start} \in Q_{free}$ to some $q' \in RM$
  - **Departability**: There is a path from $q'' \in RM$ to $q_{goal} \in Q_{free}$
  - **Connectivity**: there exists a path in *RM* between *q'* and *q''*

What cities? What universities?

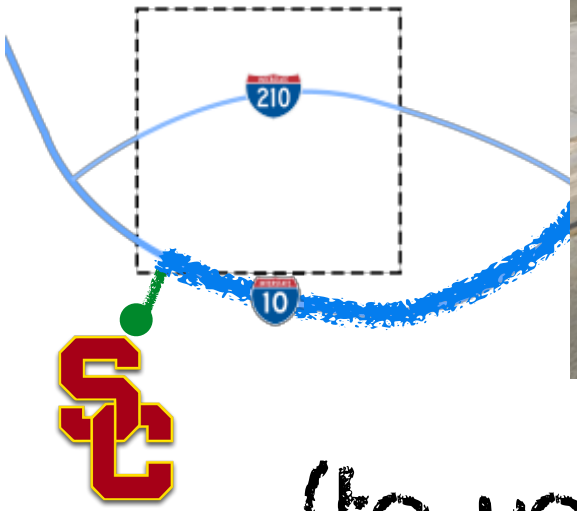




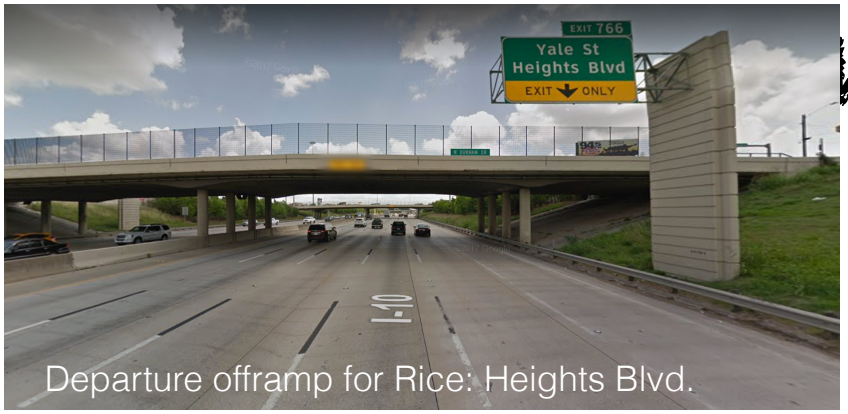
(to your best approximation)

What cities? W

"Los Angeles"

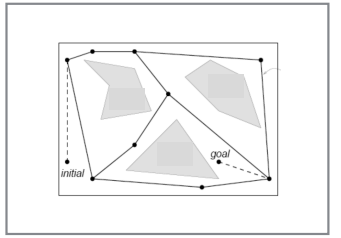


(to your best approximation)

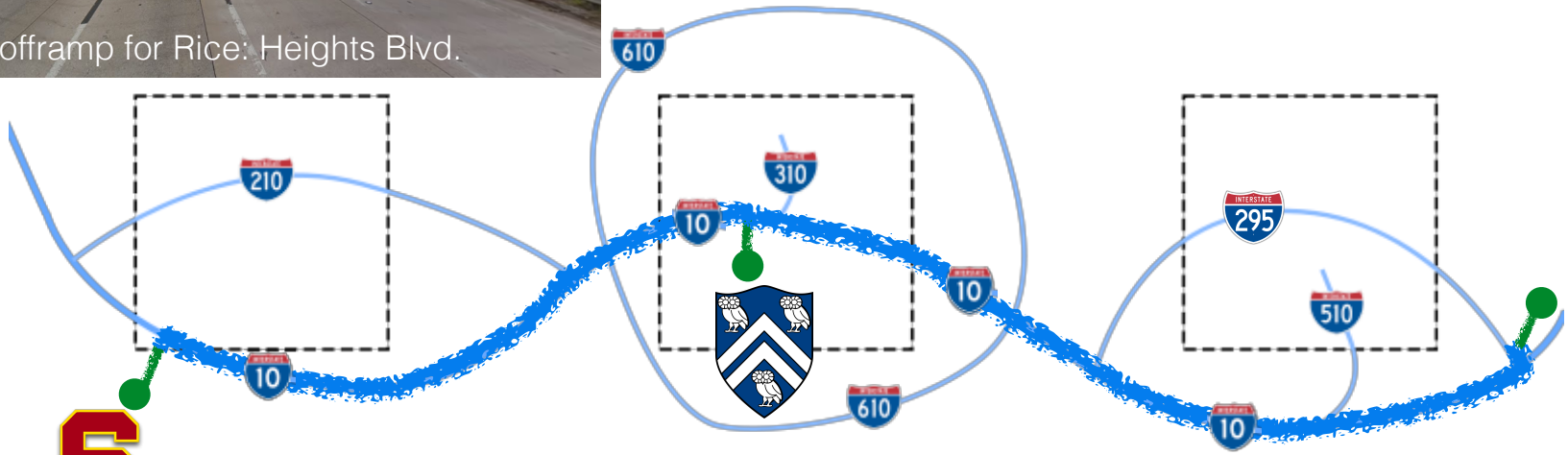


Departure offramp for Rice: Heights Blvd.

hat universities?

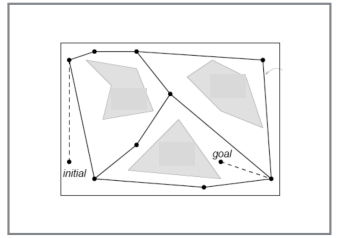


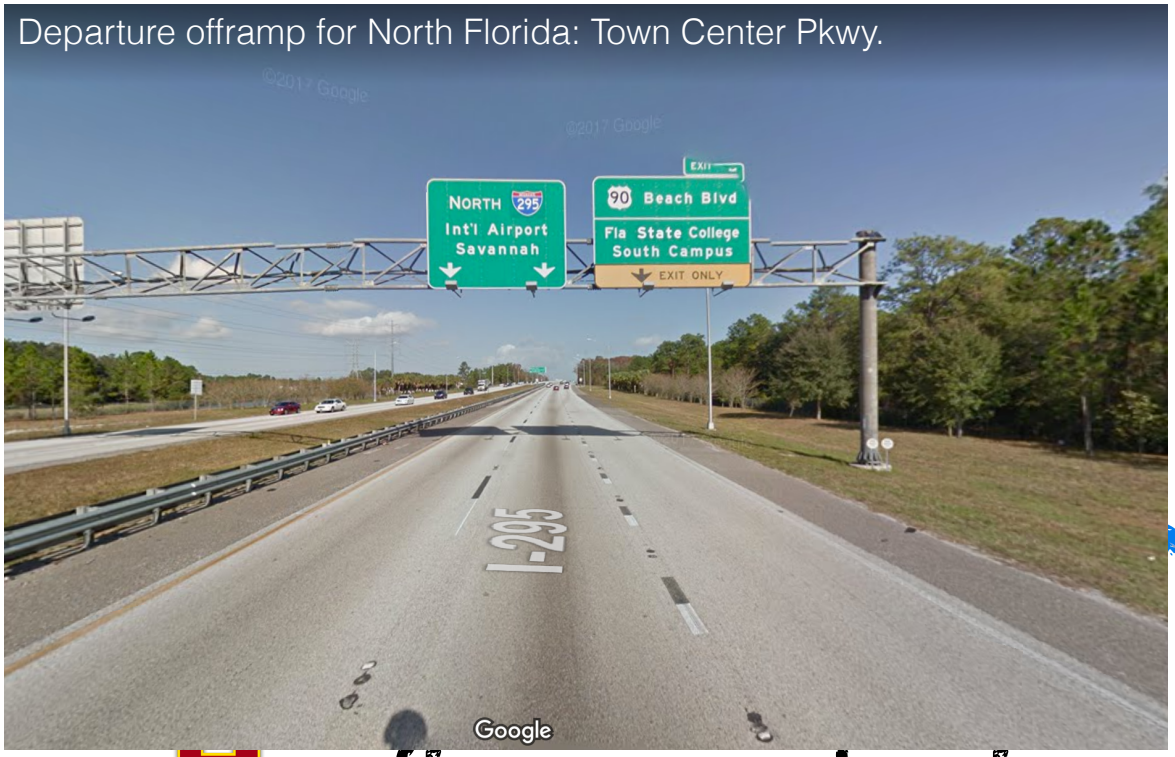
"Houston" ????????



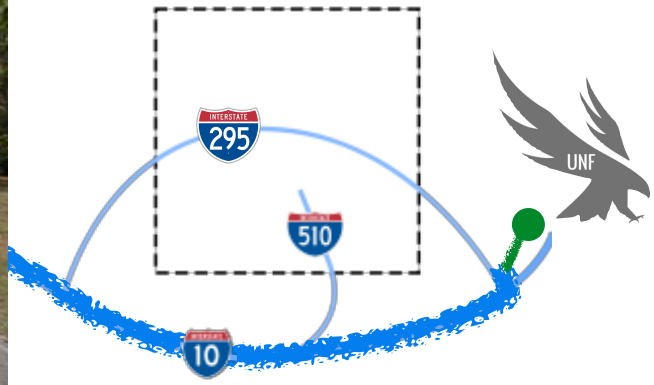
(to your best approximation)

What cities? What universities?



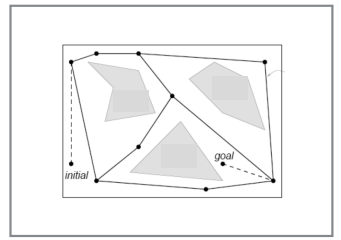


"Jacksonville"



(to your best approximation)

Basic Roadmap Planner



1) Build the roadmap *RM* as graph *G(V,E)*

   - *V*: nodes are "valid" in C-space in Q_{free}
     - a configuration *q* is valid if it is not in collision and within joint limits
   - *E*: an edge $e(q_1,q_2)$ connects two nodes if a free path connects q_1 and q_2
     - all configurations along edge assumed to be valid

2) Connect start and goal configurations to *RM* at *q'* and *q''*, respectively

3) Find path in *RM* between *q'* and *q''*

2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

Probabilistic:

C-space sampling

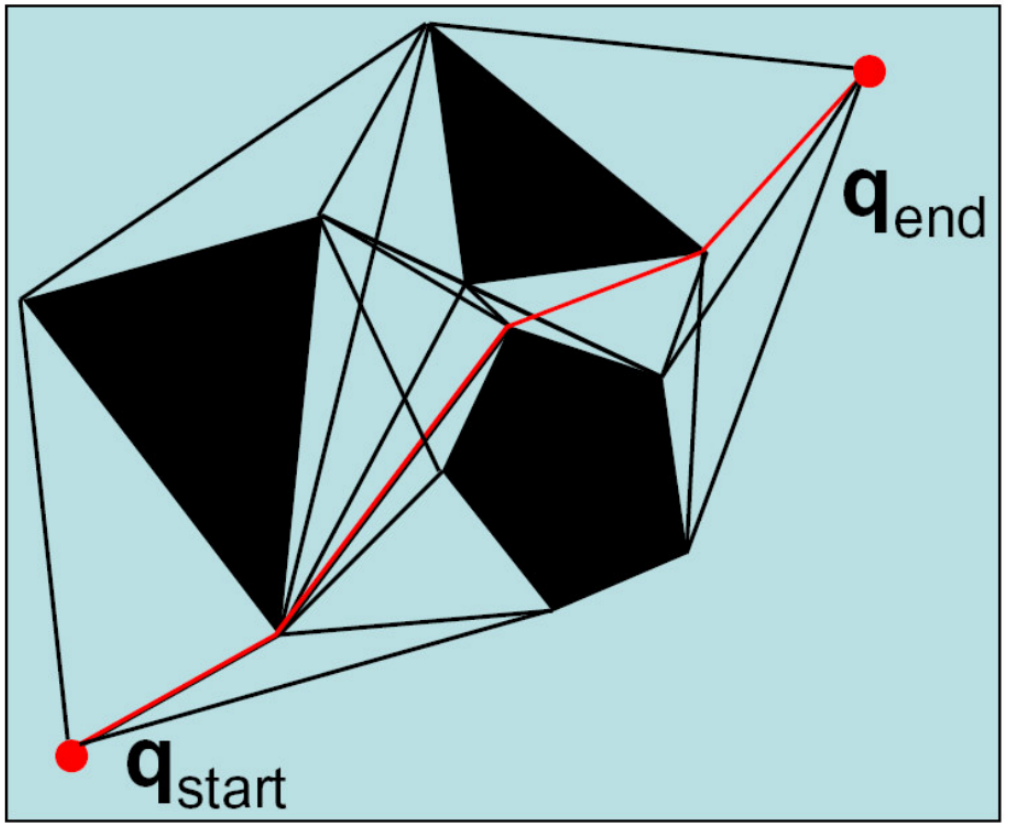
- Probabilistic Roadmap (PRM)
  - sample and connect vertices in graph for multiple planning queries
- Rapidly-exploring Random Tree (RRT)
  - sample and connect vertices in trees rooted at start and goal configuration

2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

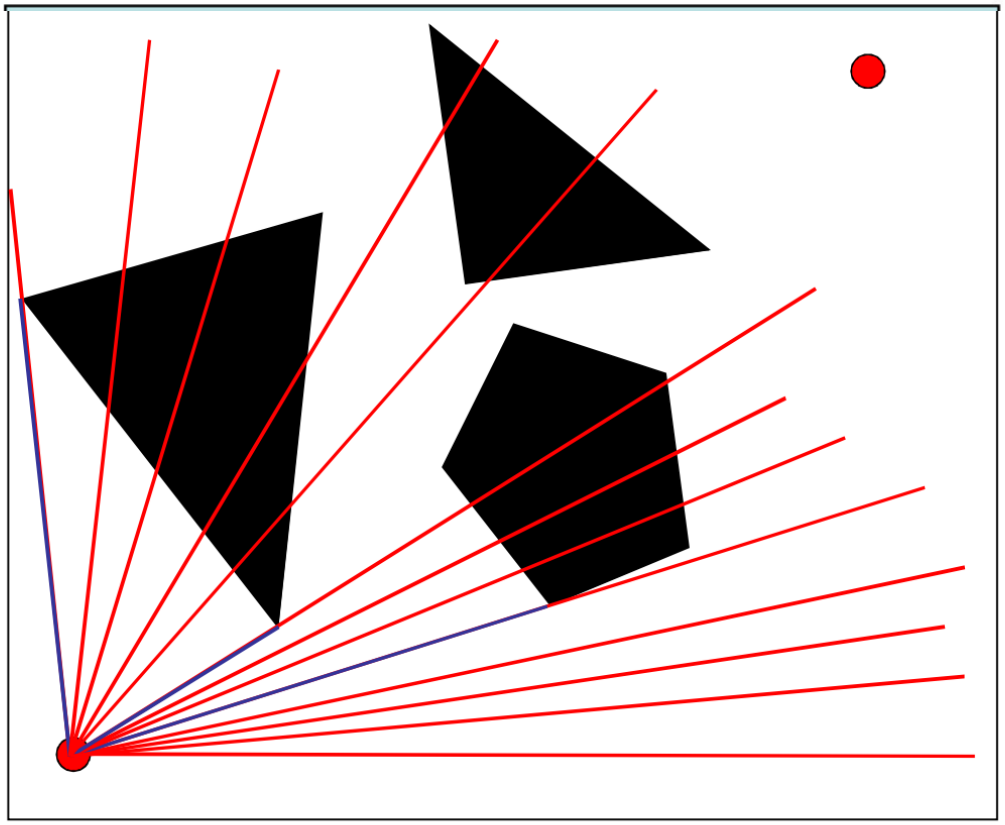


2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

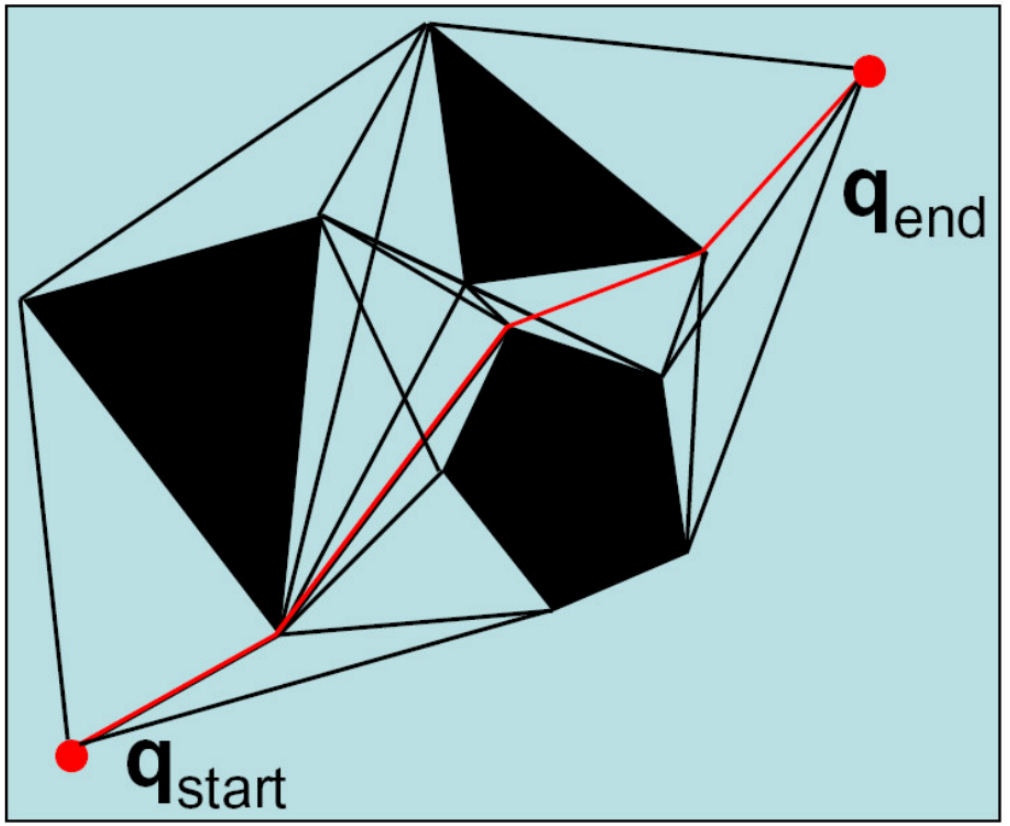


2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

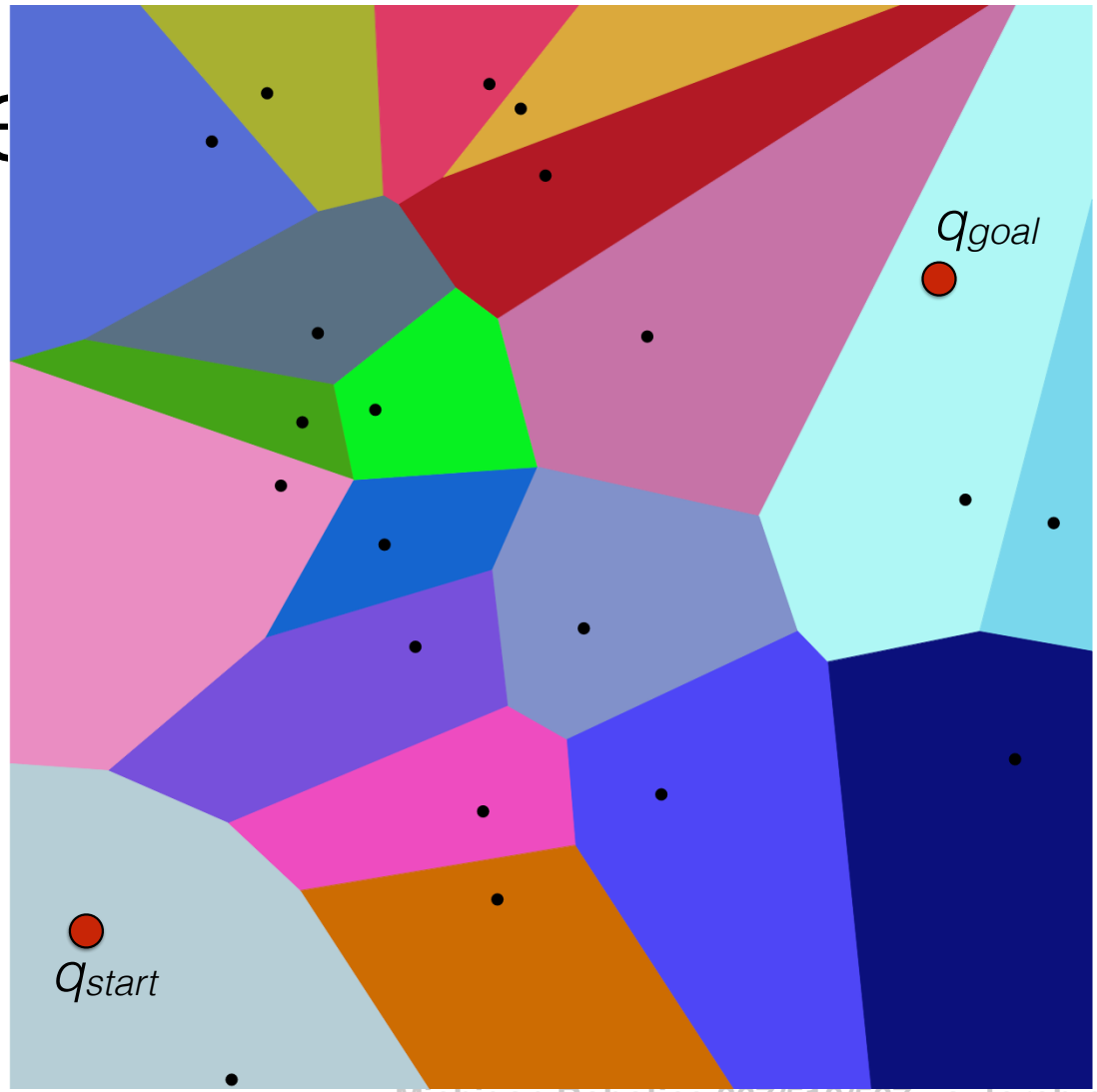


2 Approache

Deterministic:

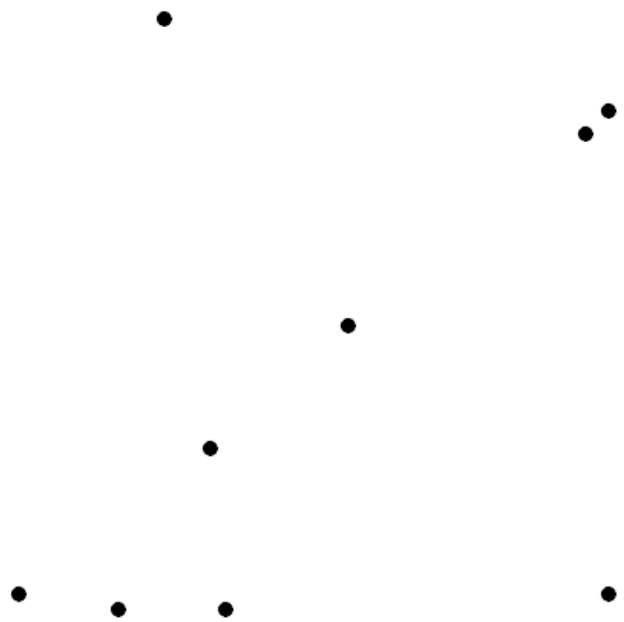
complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles



Voronoi Diagram

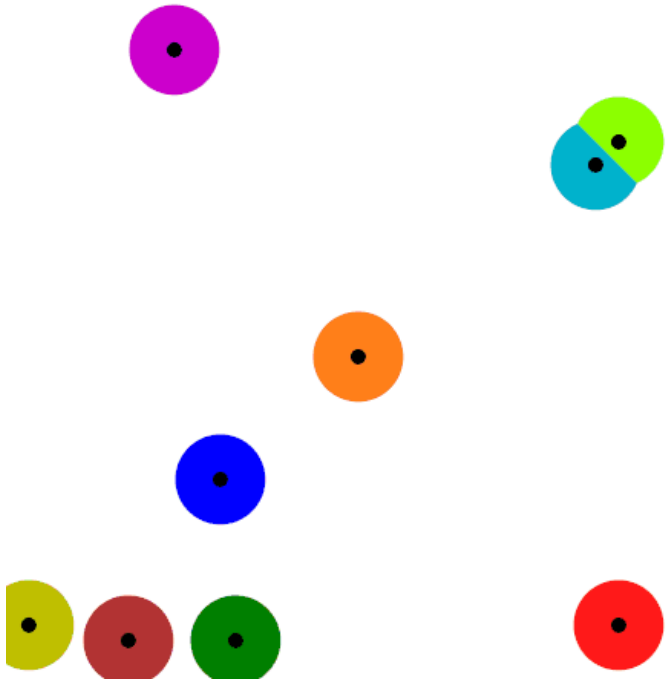
- Given *N* input points in a *d* dimensional space
- Find region boundaries such that each point on a boundary are equidistant to two or more input points
- Delaunay triangulation is a dual to the Voronoi diagram



https://en.wikipedia.org/wiki/Voronoi_diagram#/media/File:Voronoi_growth_euclidean.gif

Voronoi Diagram

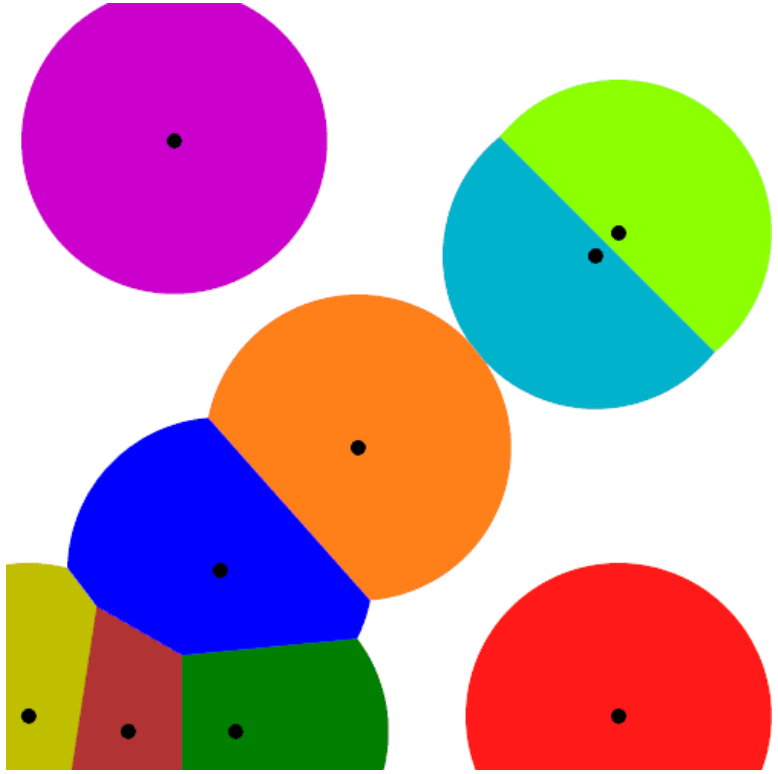
- Given *N* input points in a *d* dimensional space
- Find region boundaries such that each point on a boundary are equidistant to two or more input points
- Delaunay triangulation is a dual to the Voronoi diagram



https://en.wikipedia.org/wiki/Voronoi_diagram#/media/File:Voronoi_growth_euclidean.gif

Voronoi Diagram

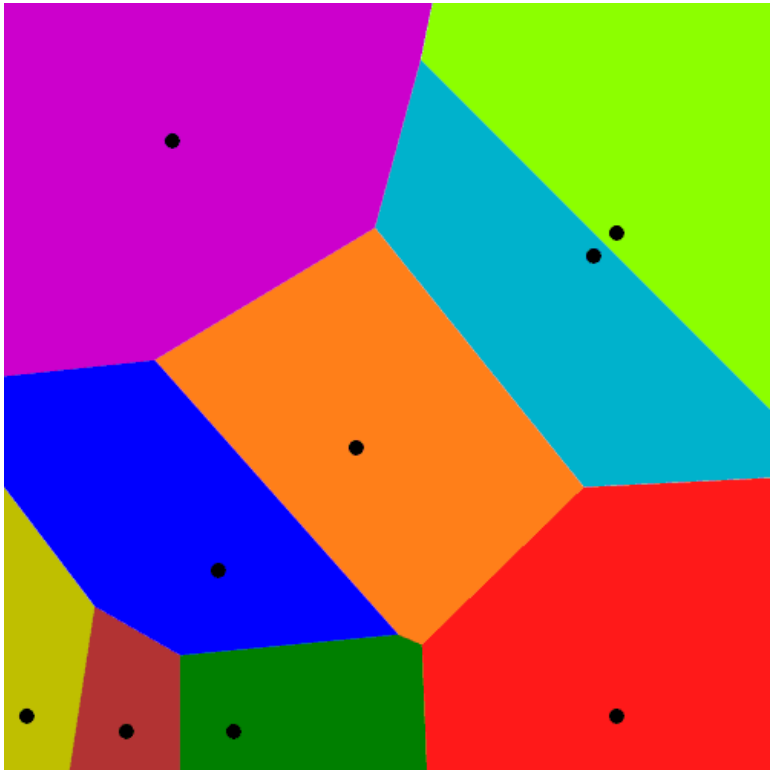
- Given *N* input points in a *d* dimensional space
- Find region boundaries such that each point on a boundary are equidistant to two or more input points
- Delaunay triangulation is a dual to the Voronoi diagram



https://en.wikipedia.org/wiki/Voronoi_diagram#/media/File:Voronoi_growth_euclidean.gif

Voronoi Diagram

- Given *N* input points in a *d* dimensional space
- Find region boundaries such that each point on a boundary are equidistant to two or more input points
- Delaunay triangulation is a dual to the Voronoi diagram



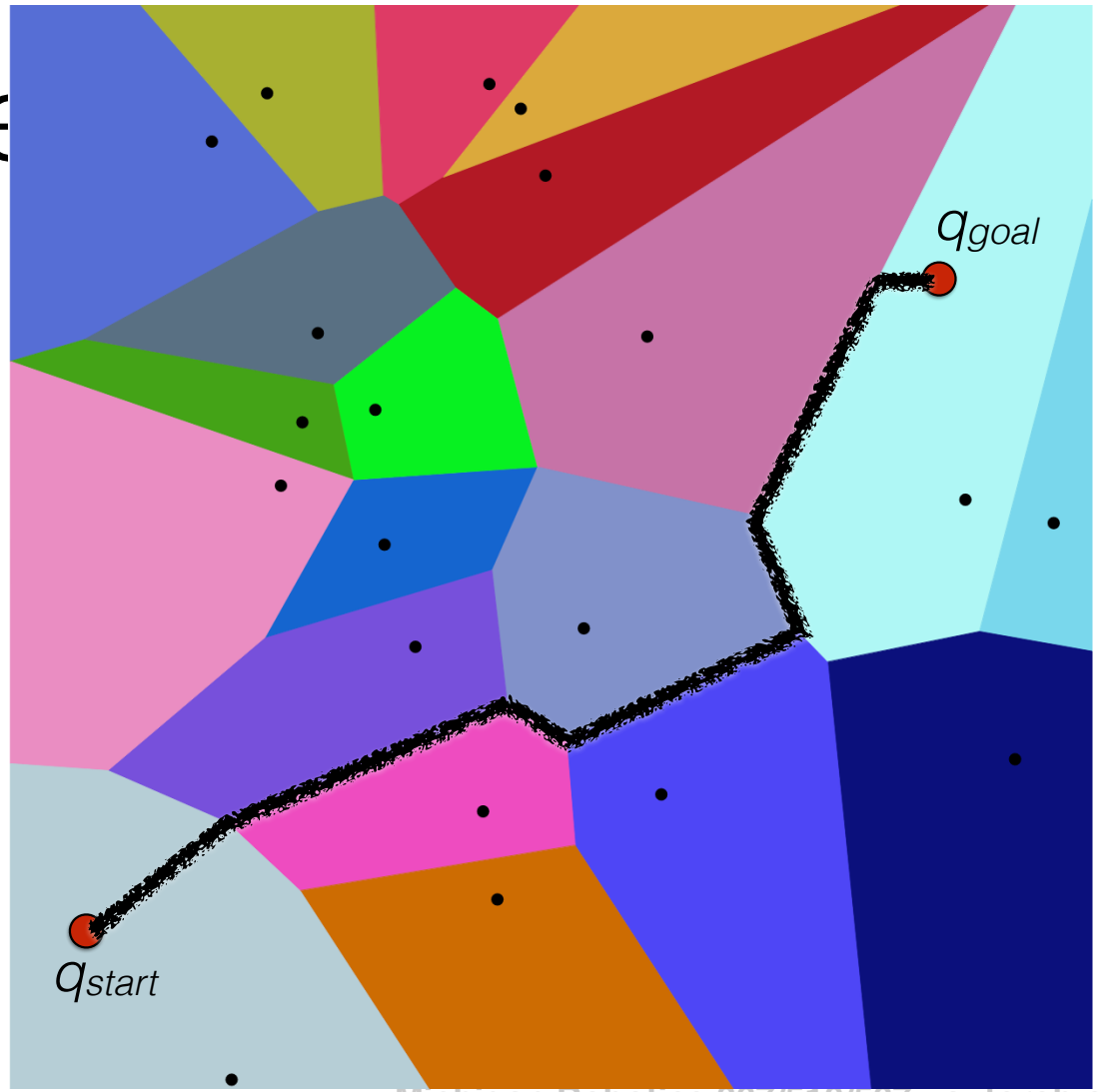
https://en.wikipedia.org/wiki/Voronoi_diagram#/media/File:Voronoi_growth_euclidean.gif

2 Approache

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles



2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

Probabilistic:

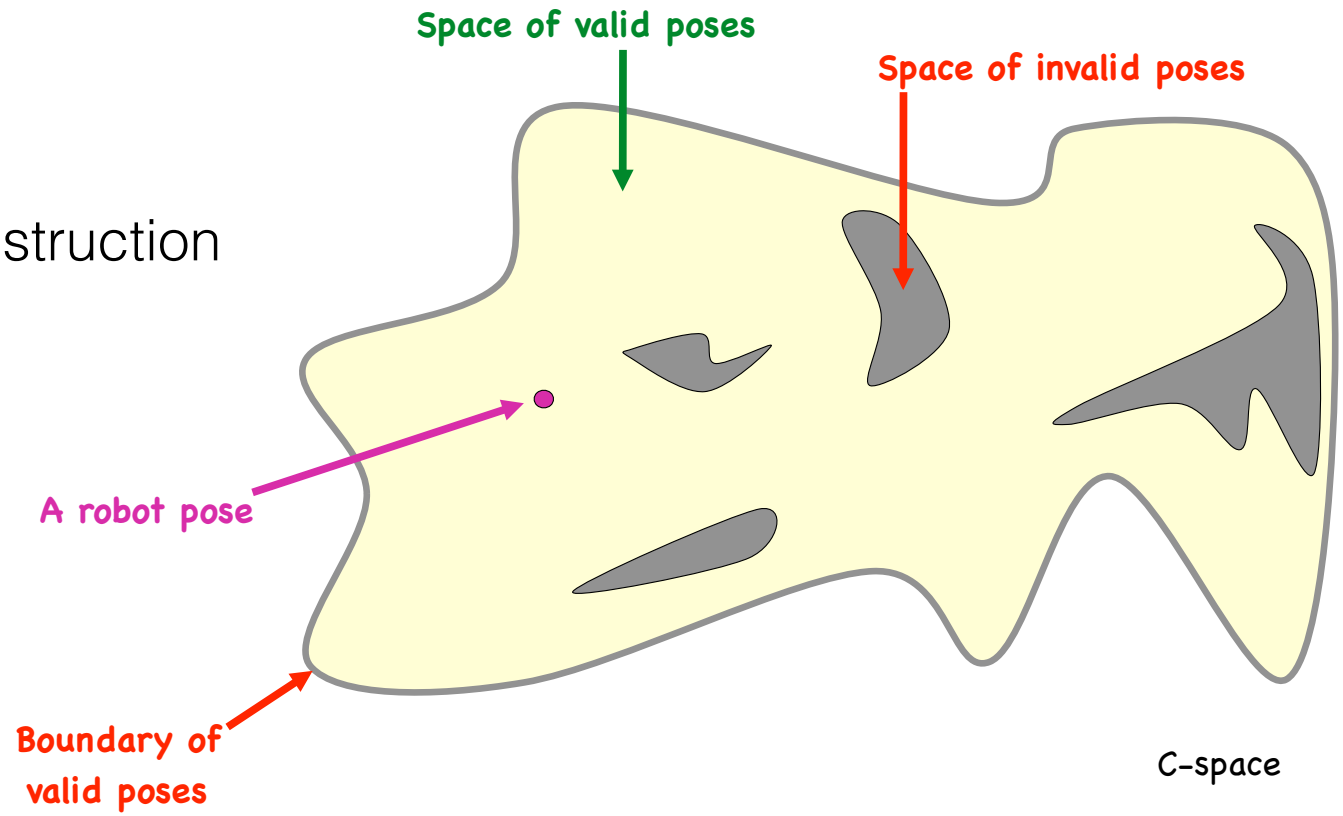
C-space sampling

- Probabilistic Roadmap (PRM)
  - sample and connect vertices in graph for multiple planning queries
- Rapidly-exploring Random Tree (RRT)
  - sample and connect vertices in trees rooted at start and goal configuration

[Kavraki, Svetska, Latombe, Overmars, 95]

Probabilistic road maps

- Two phases
  - Roadmap construction
  - Path Query

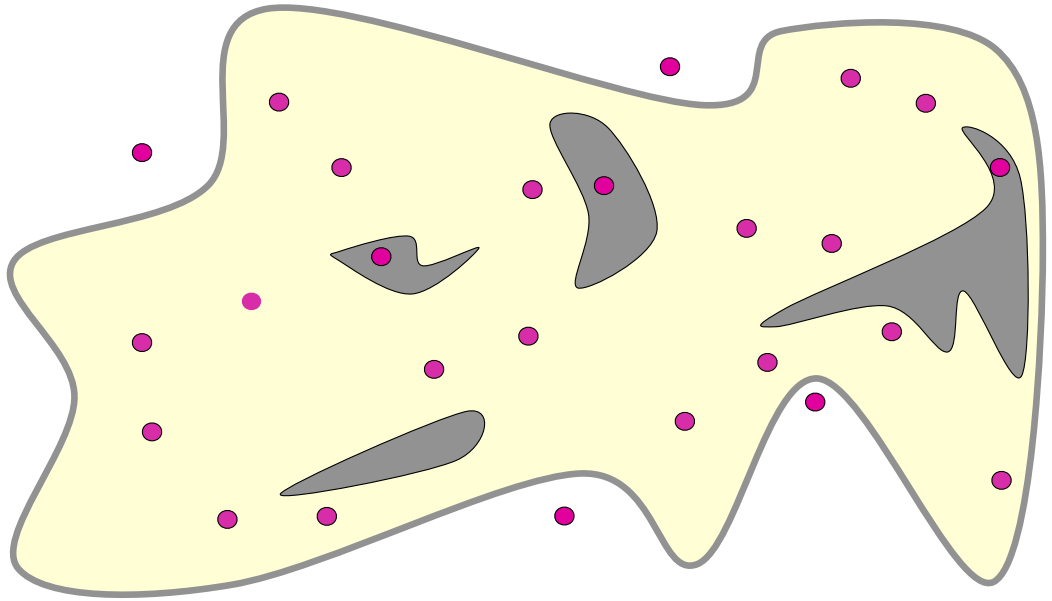


PRM: construction phase

1) **Select N sample poses at random**

2) Eliminate invalid poses

3) Connect neighboring poses

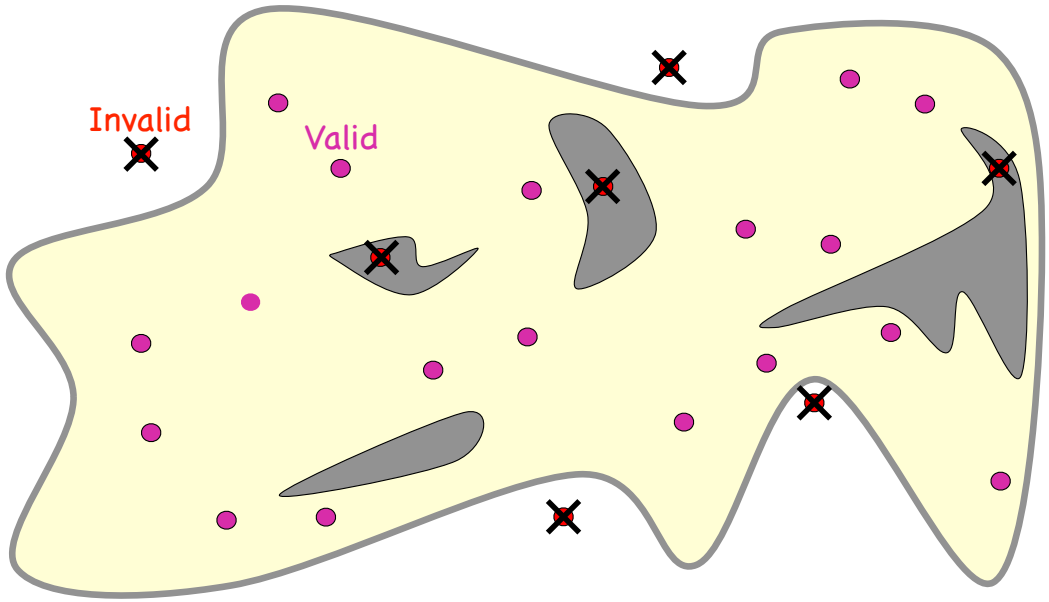


PRM: construction phase

1) Select N sample poses at random

2) **Eliminate invalid poses**

3) Connect neighboring poses

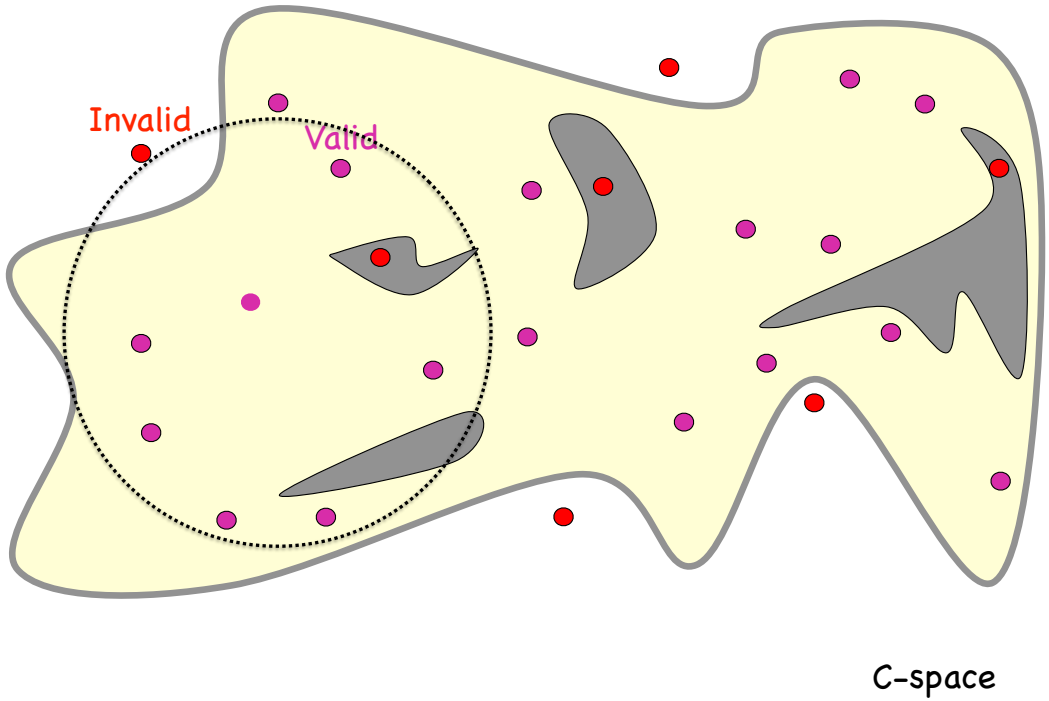


PRM: construction phase

1) Select N sample poses at random

2) Eliminate invalid poses

3) **Connect neighboring poses**

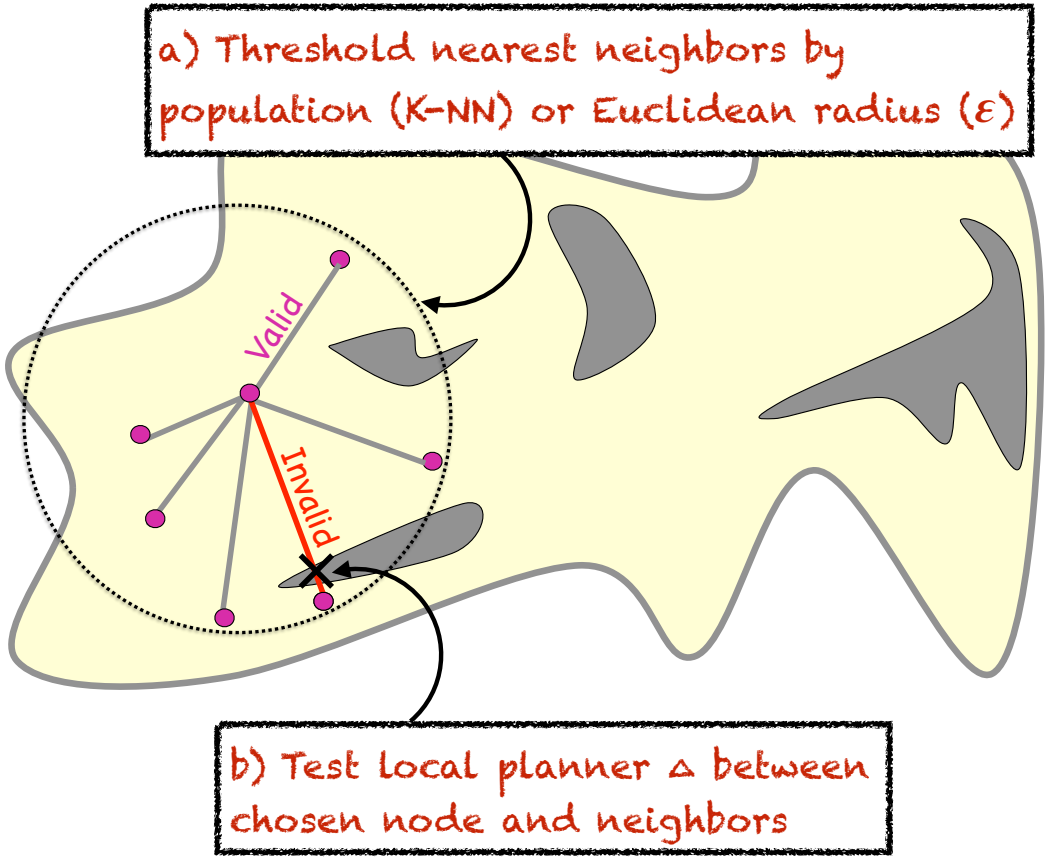


PRM: construction phase

1) Select N sample poses at random

2) Eliminate invalid poses

3) **Connect neighboring poses**



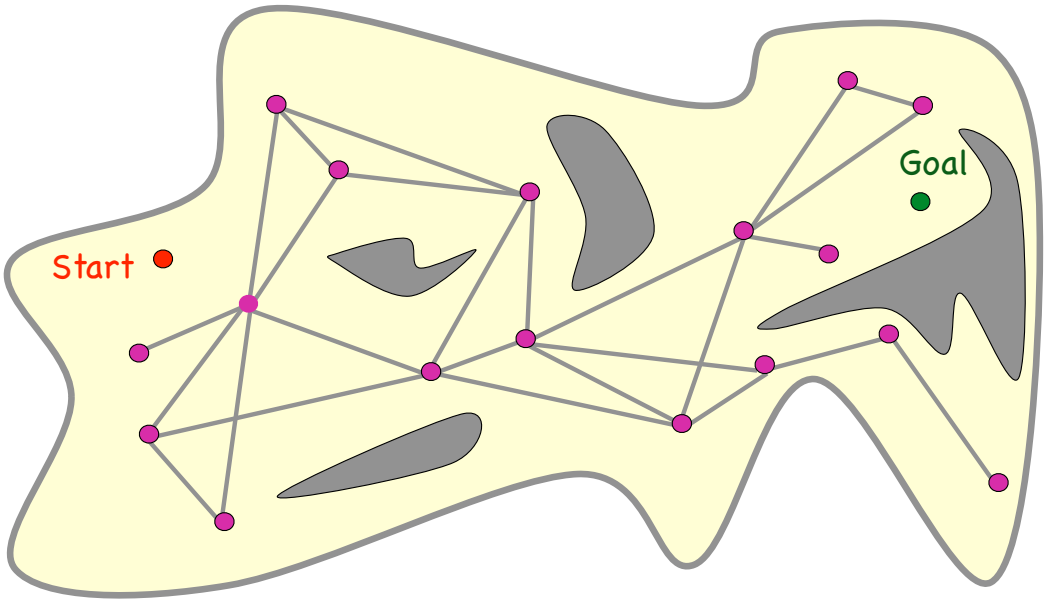
PRM: query phase

1) **Given constructed roadmap, start pose, and goal pose**

2) Attach goal and start to nearest roadmap entry nodes

3) Search for path between roadmap entry nodes

4) Return path with entry and departure edges



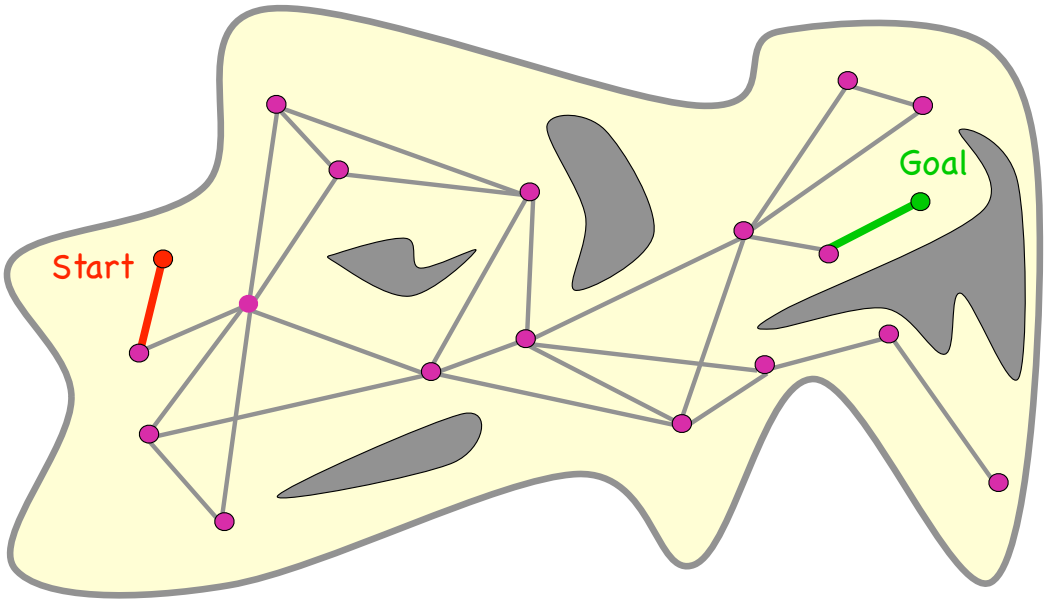
PRM: query phase

1) Given constructed roadmap, start pose, and goal pose

2) **Attach goal and start to nearest roadmap entry nodes**

3) Search for path between roadmap entry nodes

4) Return path with entry and departure edges



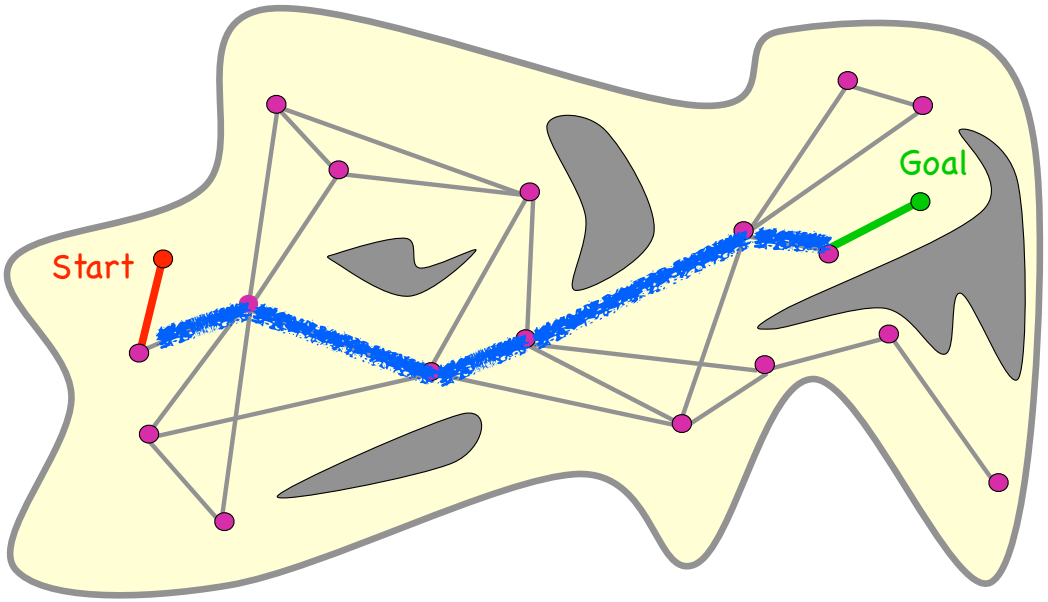
PRM: query phase

1) Given constructed roadmap, start pose, and goal pose

2) Attach goal and start to nearest roadmap entry nodes

3) **Search for path between roadmap entry nodes**

4) Return path with entry and departure edges



Remember: graph search algorithms
A*, Dijkstra, BFS, DFS

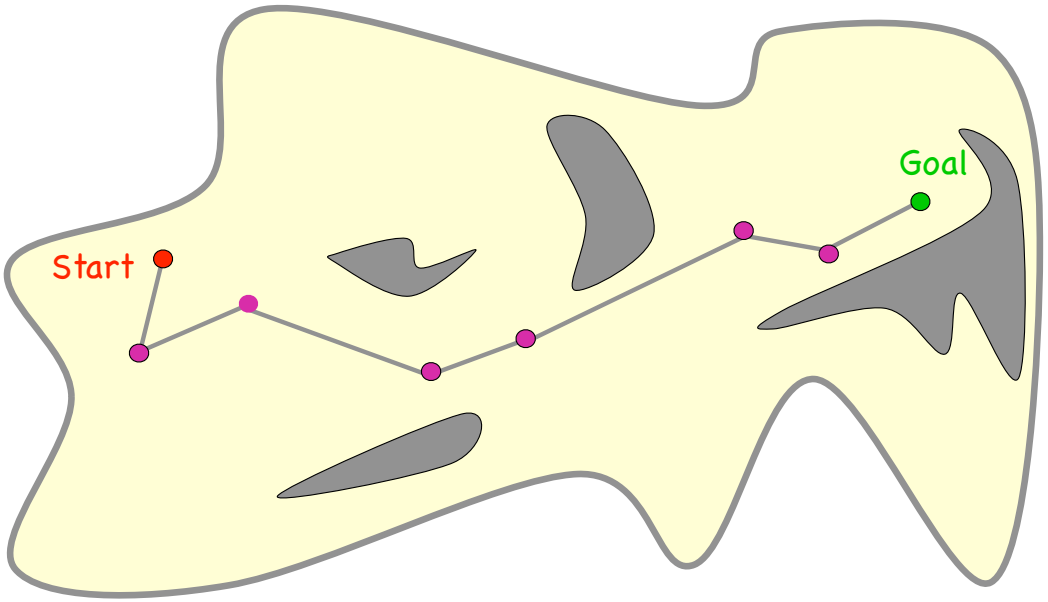
PRM: query phase

1) Given constructed roadmap, start pose, and goal pose

2) Attach goal and start to nearest roadmap entry nodes

3) Search for path between roadmap entry nodes

4) **Return path with entry and departure edges**



Multi-query planning: Considerations

- Number of samples wrt. C-space dimensionality
- Balanced sampling over C-space
- Choice of distance (e.g., Euclidean)
- Choice of local planner (e.g., line subdivision)
- Selecting neighbors: (e.g., K-NN, kd-tree, cell hashing)

2 Approaches to Roadmaps

Deterministic:

complete algorithms

- Visibility Graph
  - trace lines connecting obstacle polygon vertices
- Voronoi Planning
  - trace edges equidistant from obstacles

Probabilistic:

C-space sampling

- Probabilistic Roadmap (PRM)
  - sample and connect vertices in graph for multiple planning queries
- Rapidly-exploring Random Tree (RRT)
  - sample and connect vertices in trees rooted at start and goal configuration

Single Query Planning

- Given specific start and goal configurations
- Grow trees from start and goal towards each other
- Path is found once trees connect
- Focus sampling in unexplored areas of C-space and moving towards start/goal
- Common algorithms:
  - ESTs (expansive space trees)
  - **RRTs (rapidly exploring random trees)**

[Kuffner, LaValle 2000]

RRT Algorithm

Extend graph towards a random configuration and repeat

BUILD_RRT(q_{init})
1 $\mathcal{T}$.init(q_{init});
2 **for** $k = 1$ **to** K **do**
3 $q_{rand} \leftarrow$ RANDOM_CONFIG();
4 EXTEND($\mathcal{T}, q_{rand}$);
5 Return $\mathcal{T}$

[Kuffner, LaValle 2000]

RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD_RRT(q_init)
1  T.init(q_init);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      EXTEND(T, q_rand);
5  Return T
```

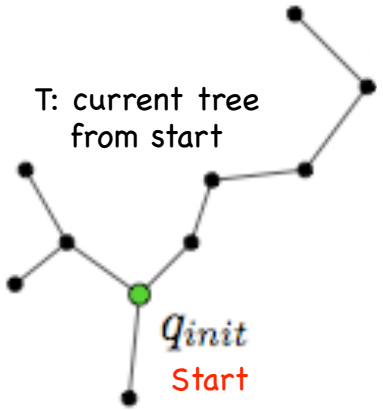


Figure 3: The EXTEND operation.

[Kuffner, LaValle 2000]

RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD_RRT(q_init)
1  T.init(q_init);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      EXTEND(T, q_rand);
5  Return T
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

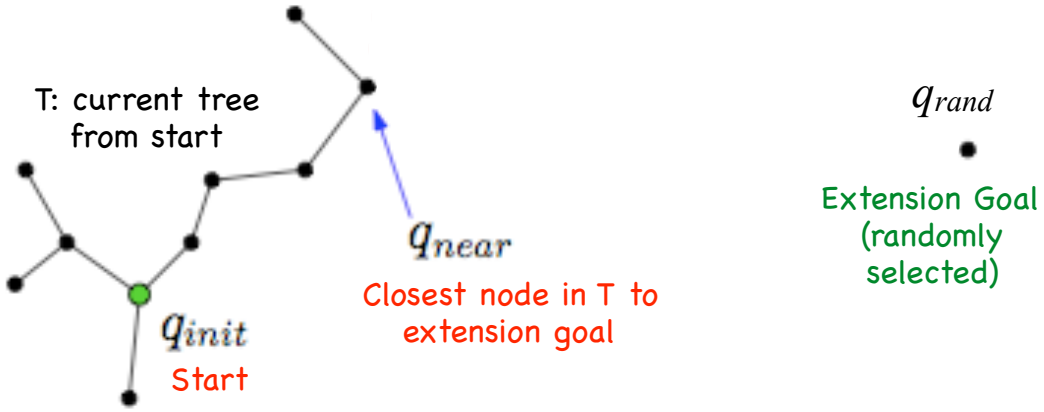


Figure 3: The EXTEND operation.

Extend graph towards a random configuration

[Kuffner, LaValle 2000]

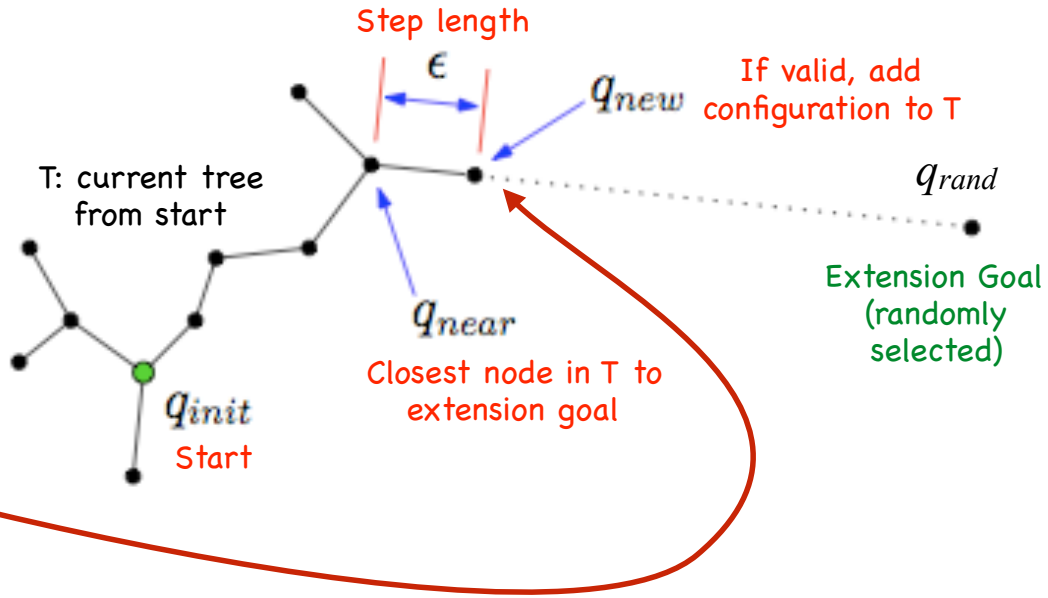
RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD_RRT(q_init)
1  T.init(q_init);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      EXTEND(T, q_rand);
5  Return T
```

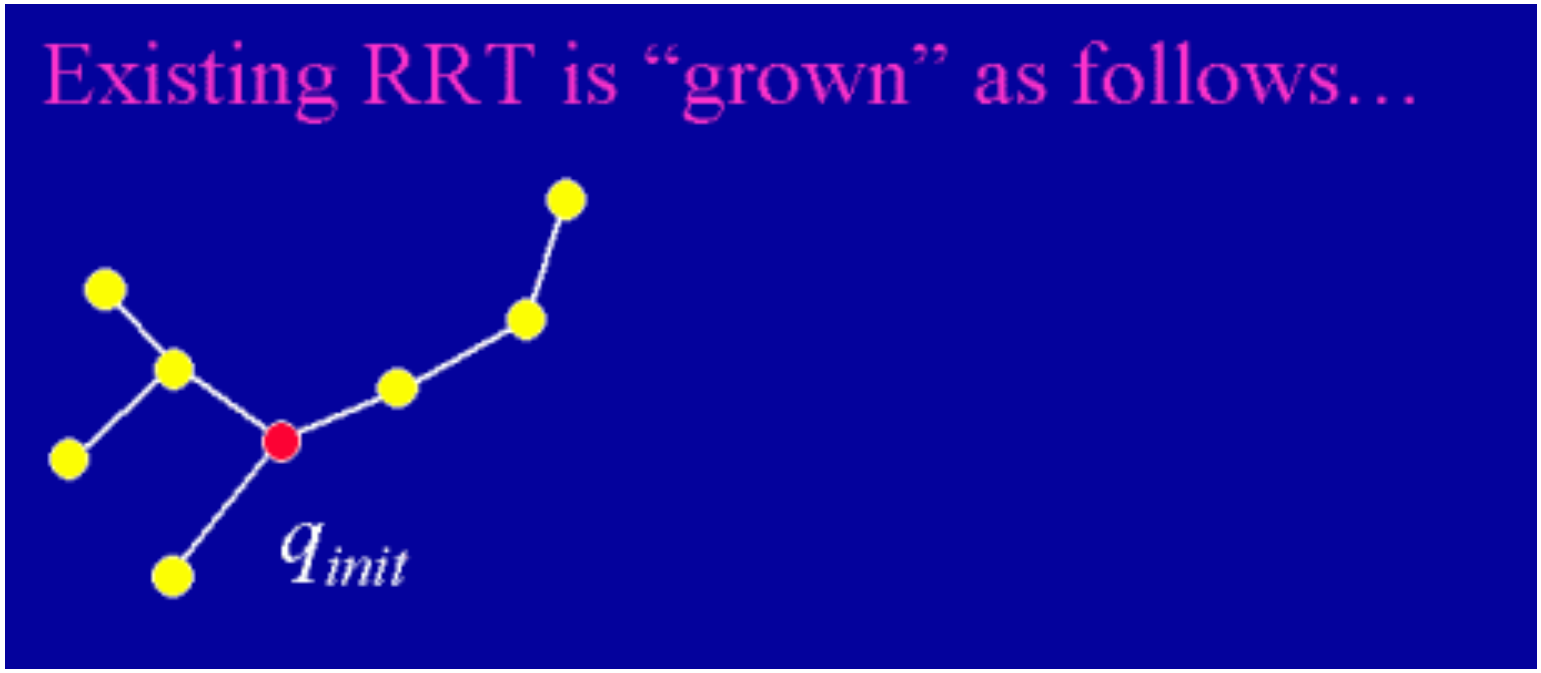
```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

Extend graph towards a random configuration



Generate and test new configuration along vector in C-space from q_{near} to q_{rand}

RRT Extend animation

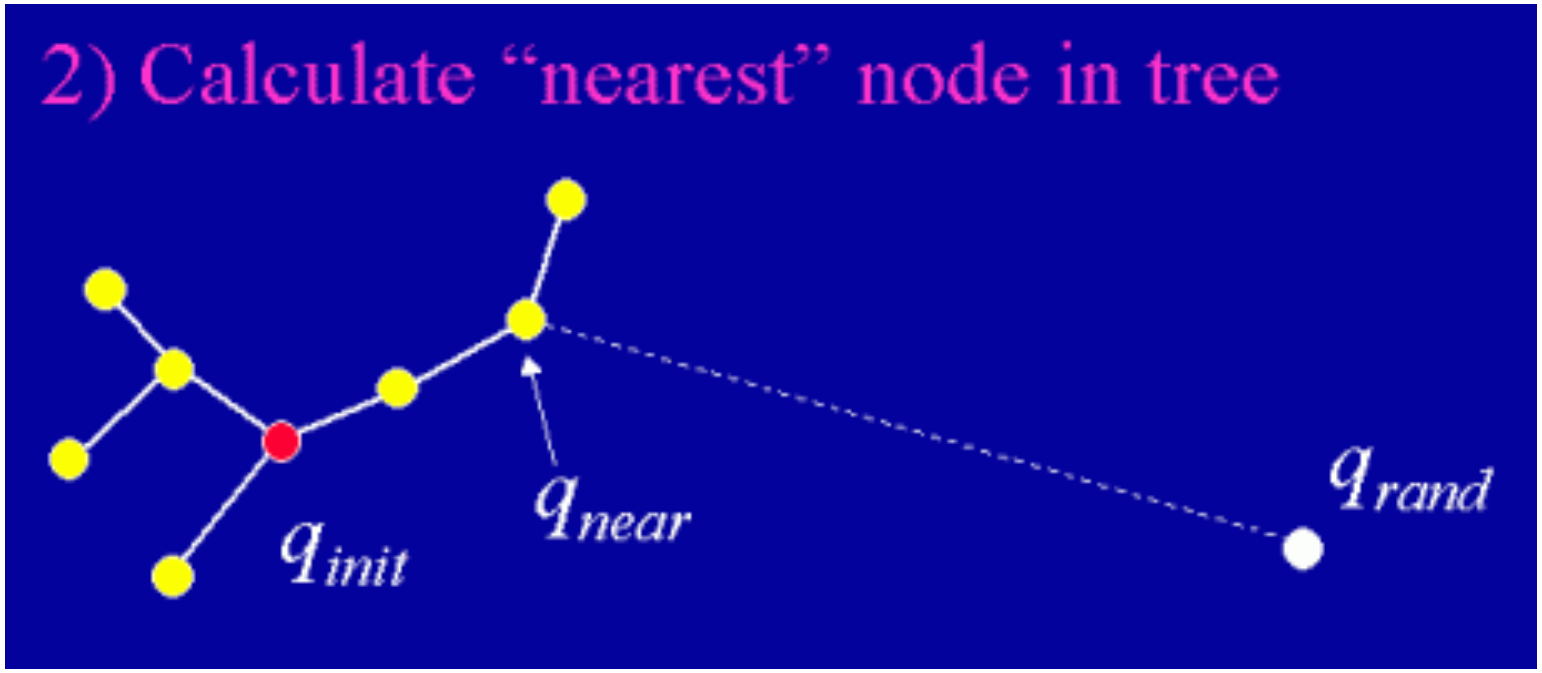


http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation

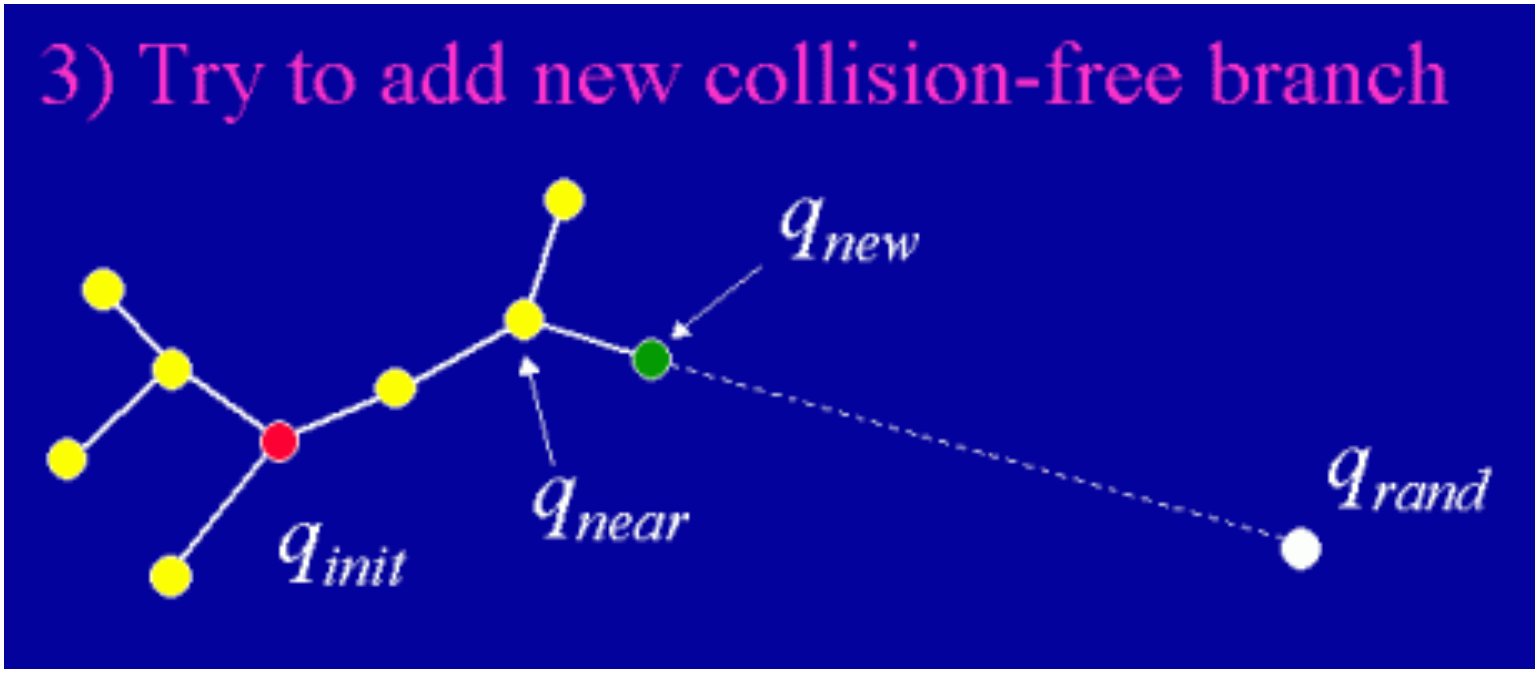
http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation



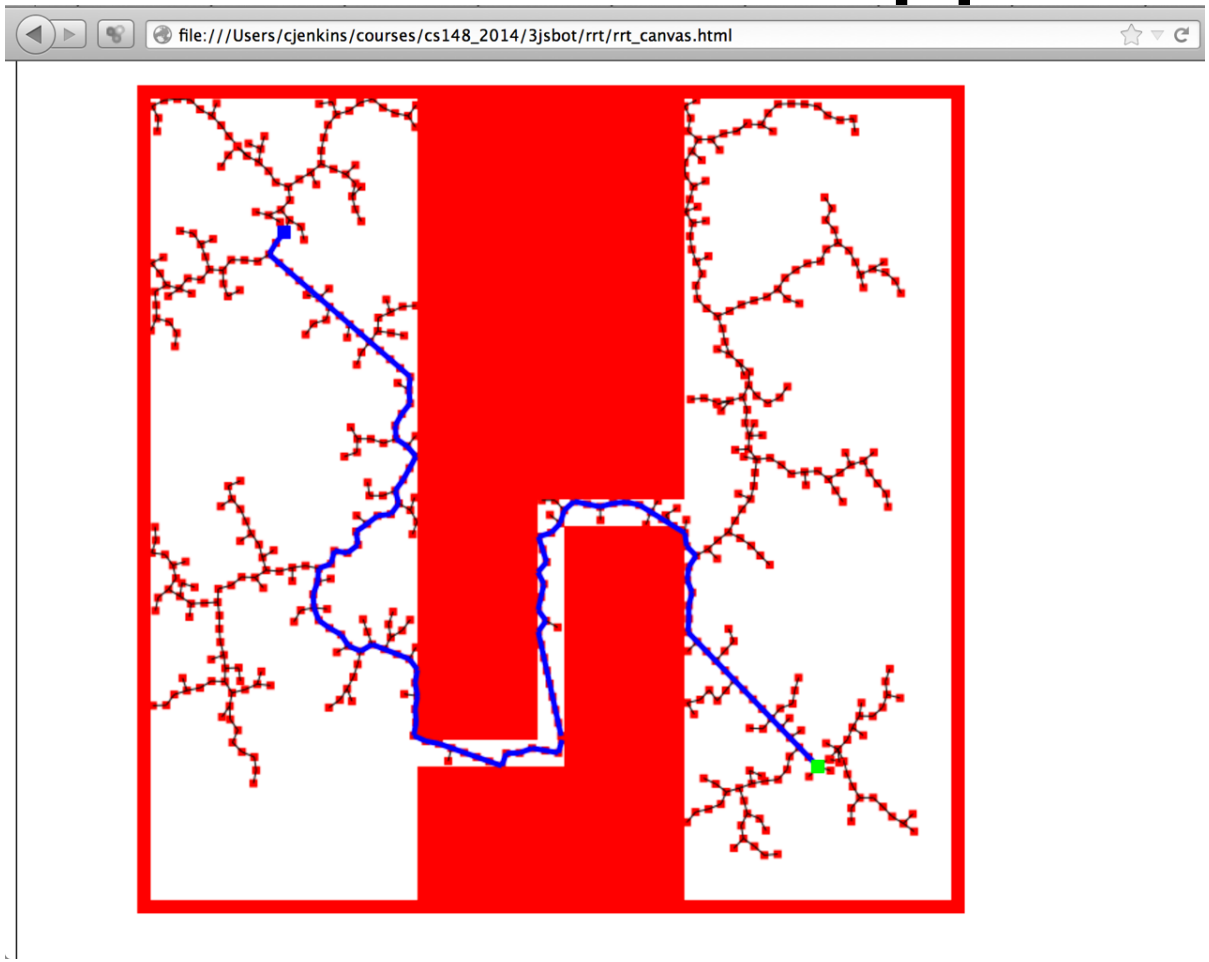
http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation



http://www.kuffner.org/james/plan/algorithm.php

Let's see what happens

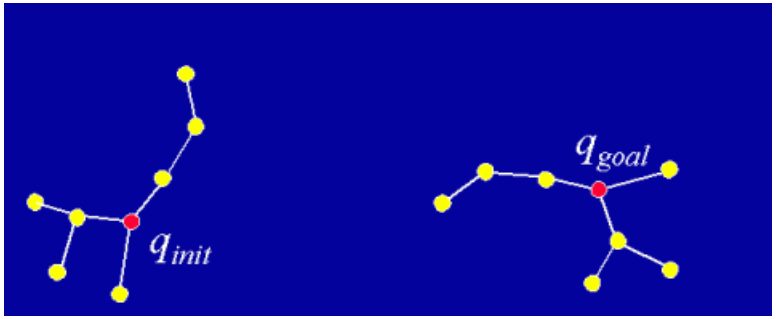


RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

RRT_CONNECT_PLANNER(q_{init}, q_{goal})
1 $\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
2 **for** $k = 1$ **to** K **do**
3 $q_{rand} \leftarrow$ RANDOM_CONFIG();
4 **if not** (EXTEND($\mathcal{T}_a, q_{rand}$) = *Trapped*) **then**
5 **if** (CONNECT($\mathcal{T}_b, q_{new}$) = *Reached*) **then**
6 Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7 SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8 Return *Failure*



RRT Connect

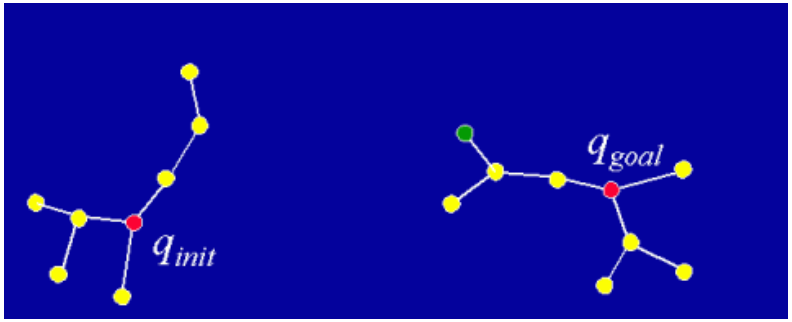
[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration



RRT Connect

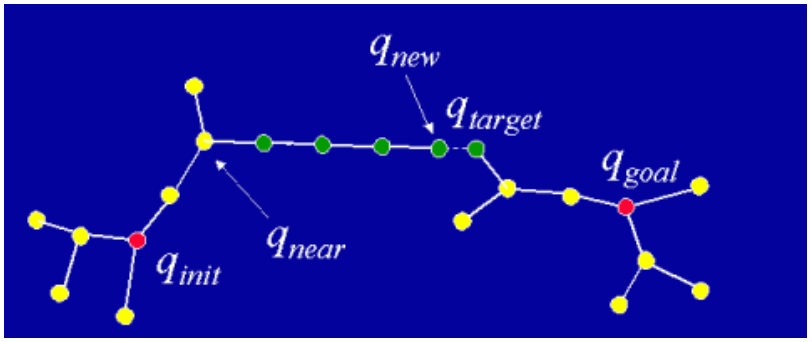
[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration




```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

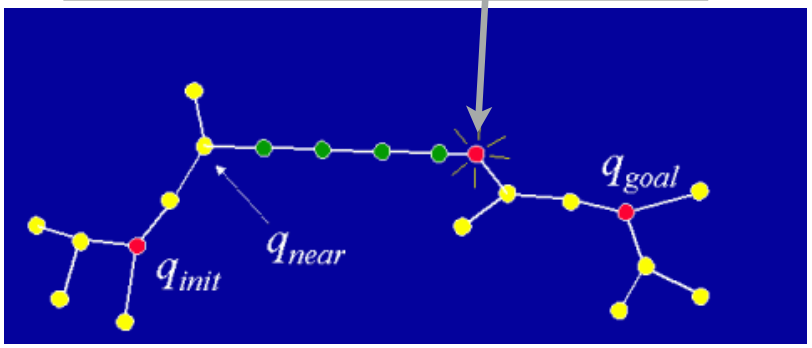
RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

search succeeds if trees connect




```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration

```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

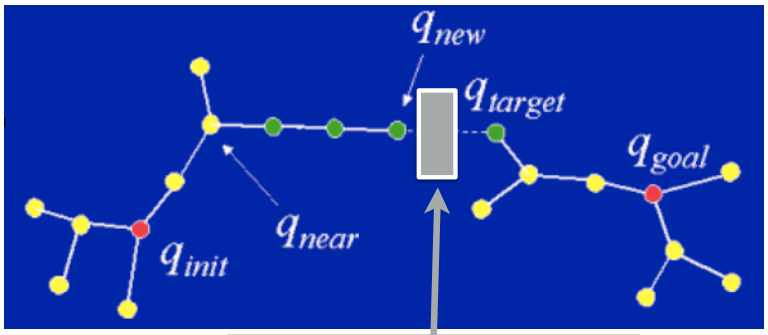
```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration

```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

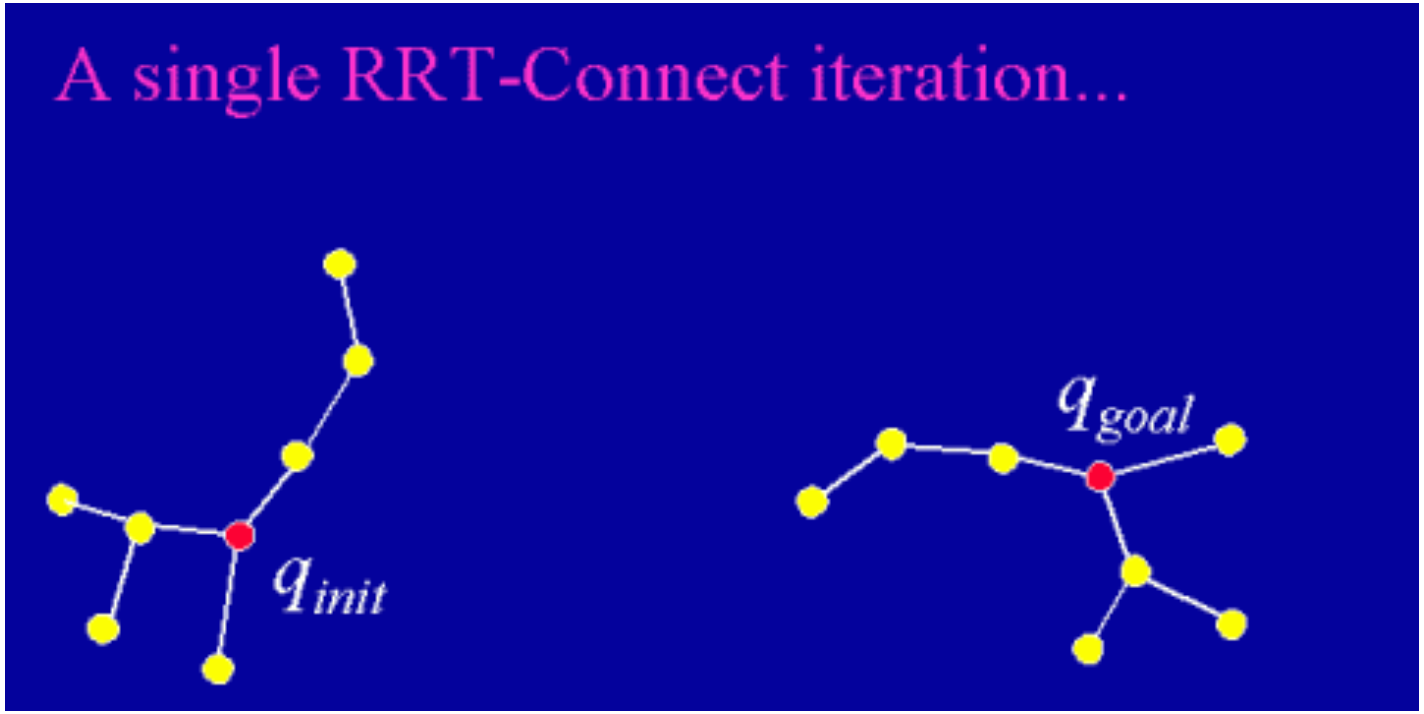
2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

3) reverse roles for trees A and B and repeat



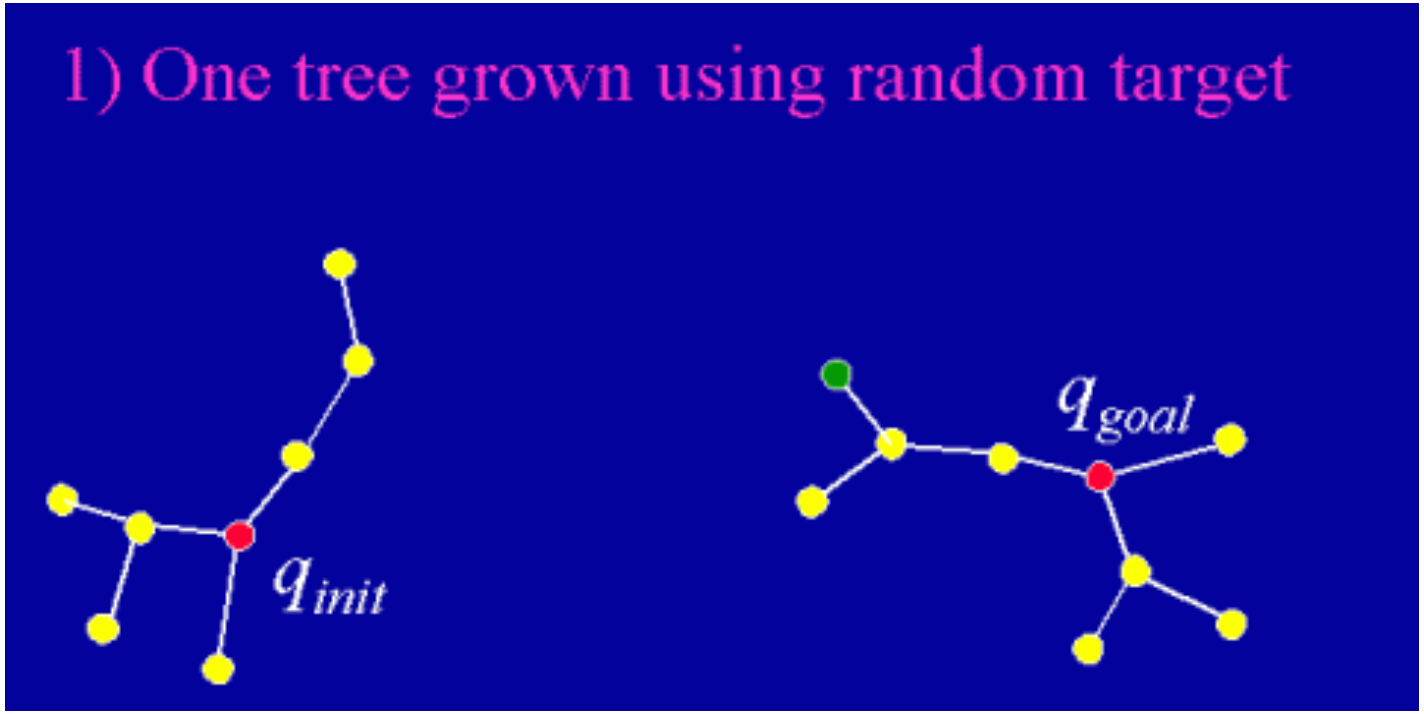
collision encountered

RRT-Connect animation



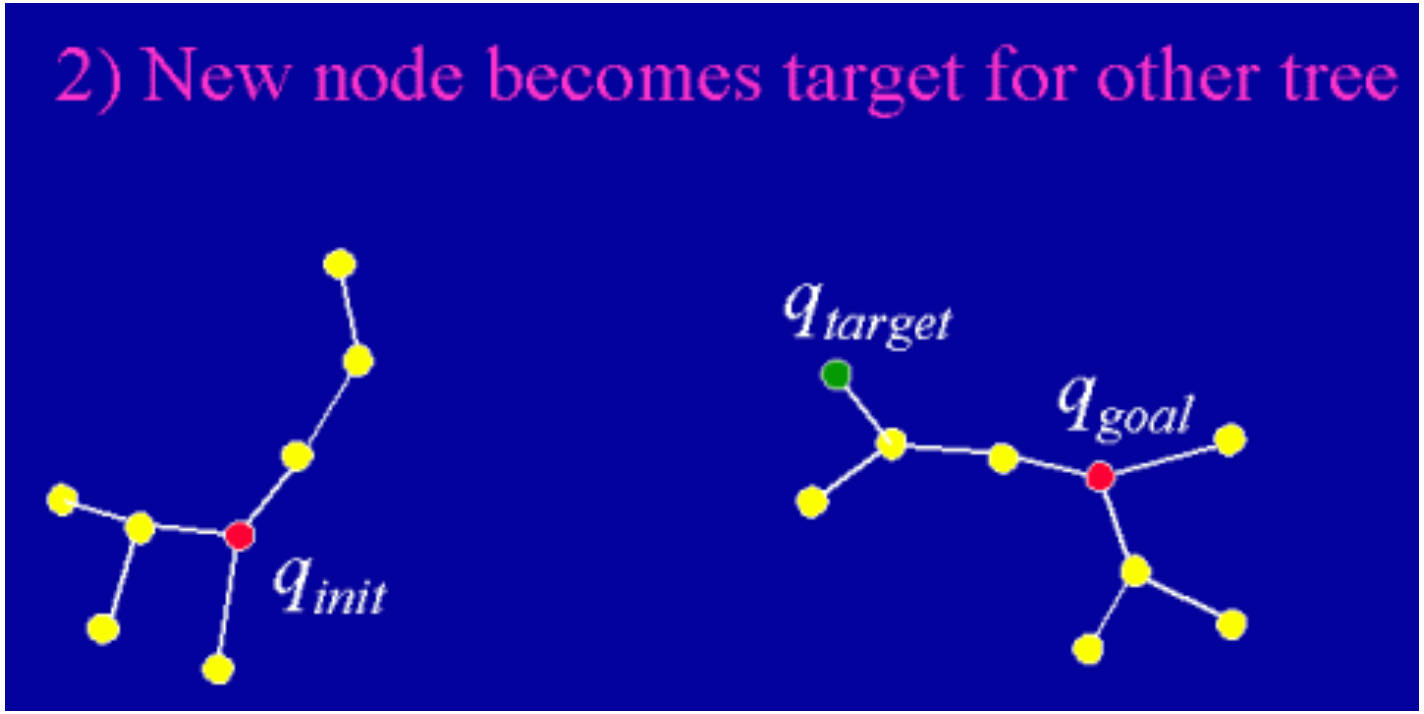
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



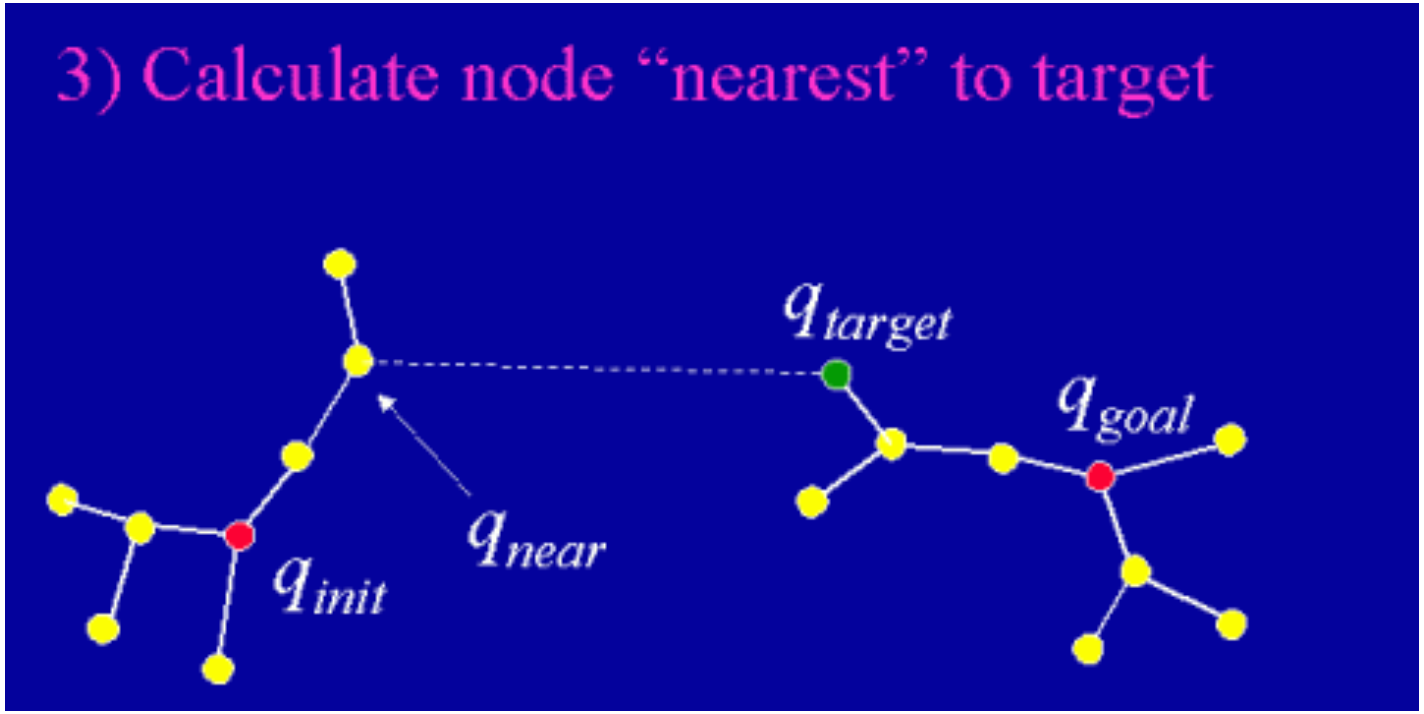
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



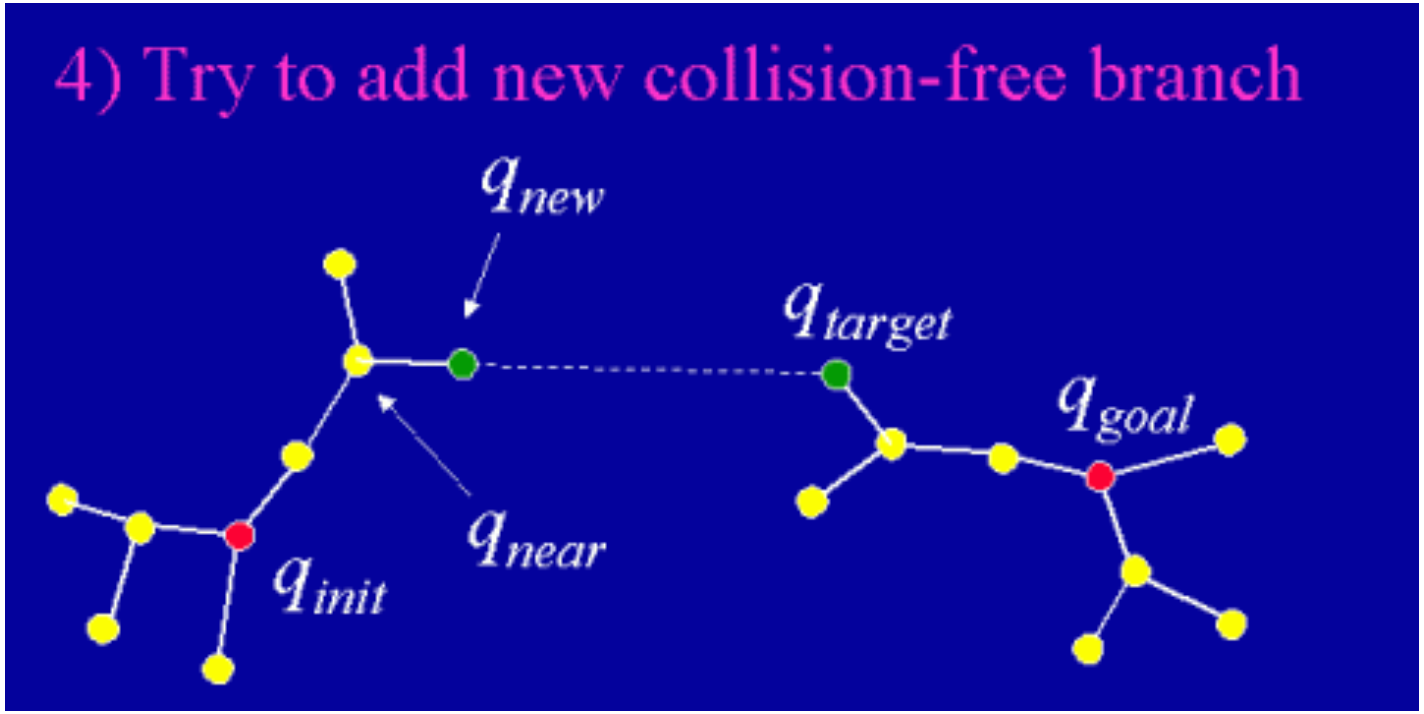
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation

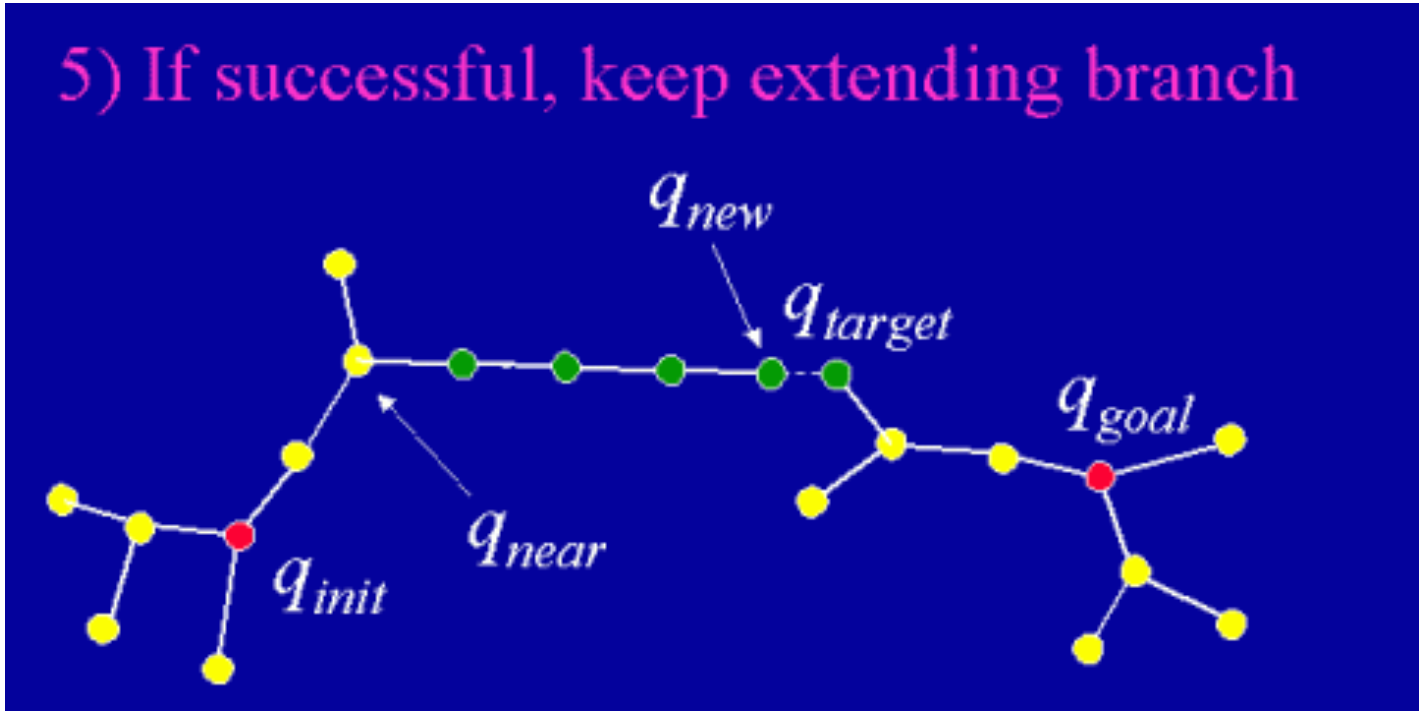


http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation

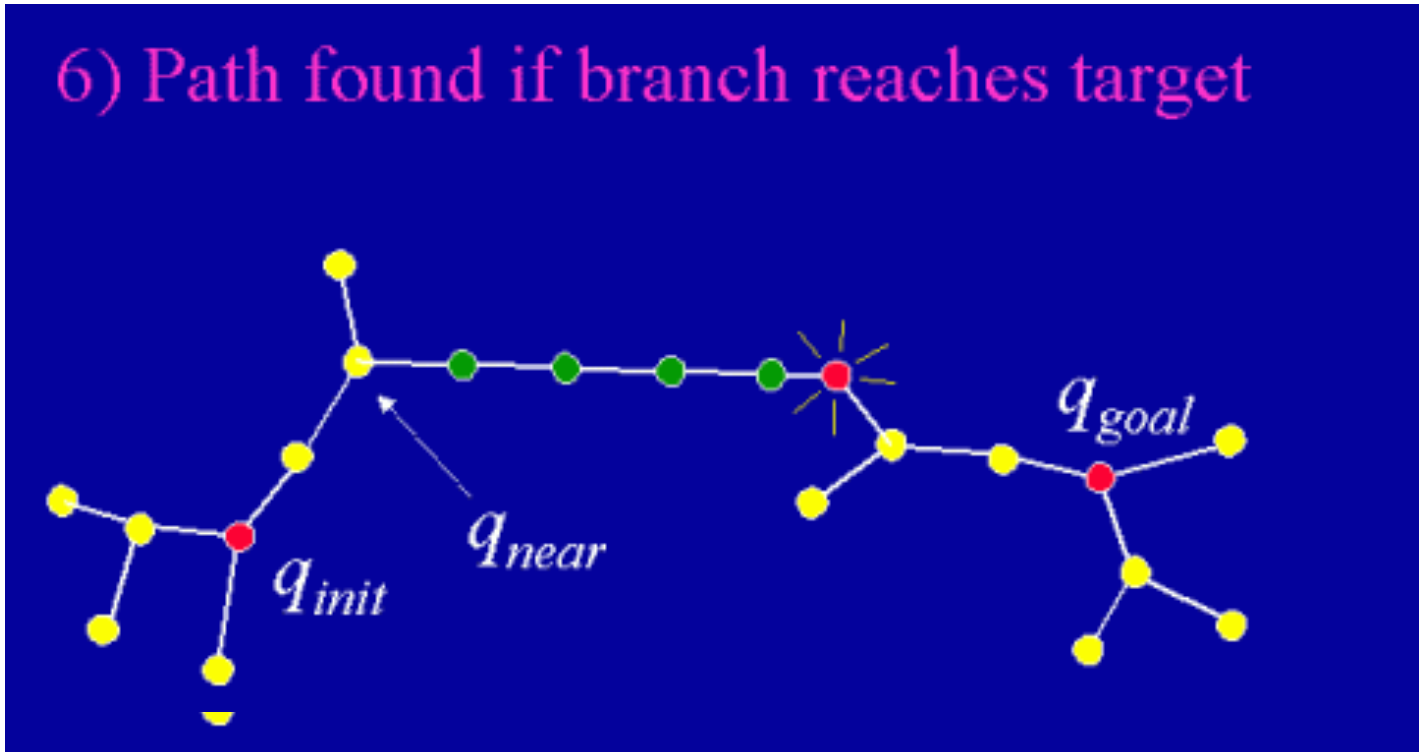
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



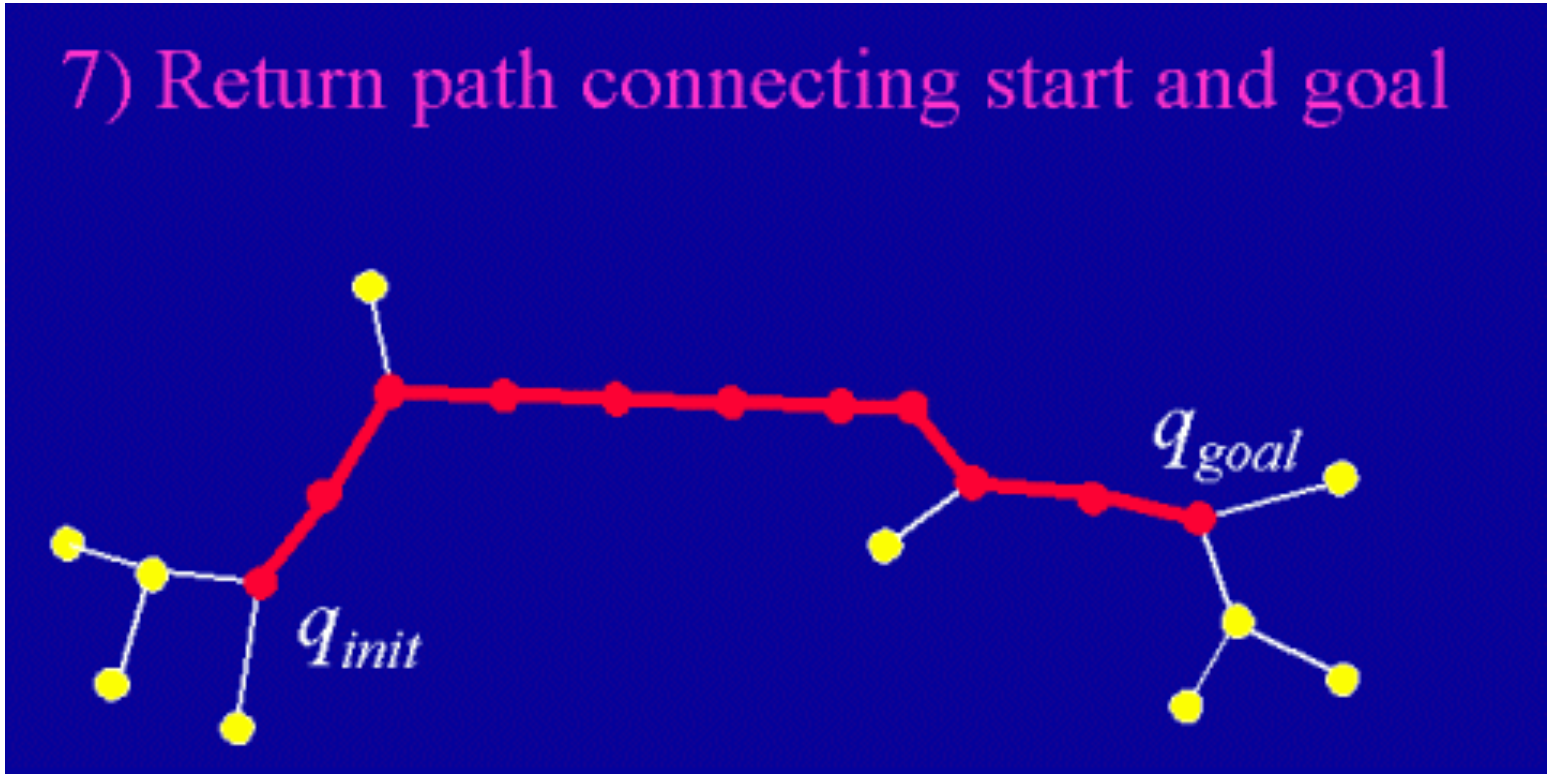
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



http://www.kuffner.org/james/plan/algorithm.php

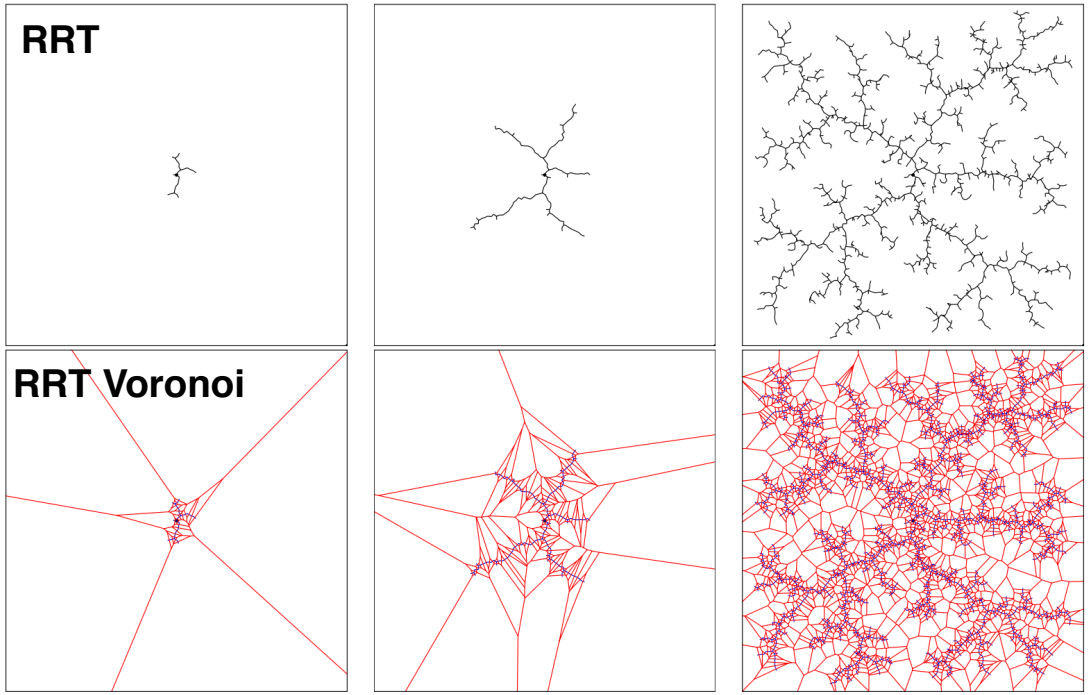
RRT-Connect animation



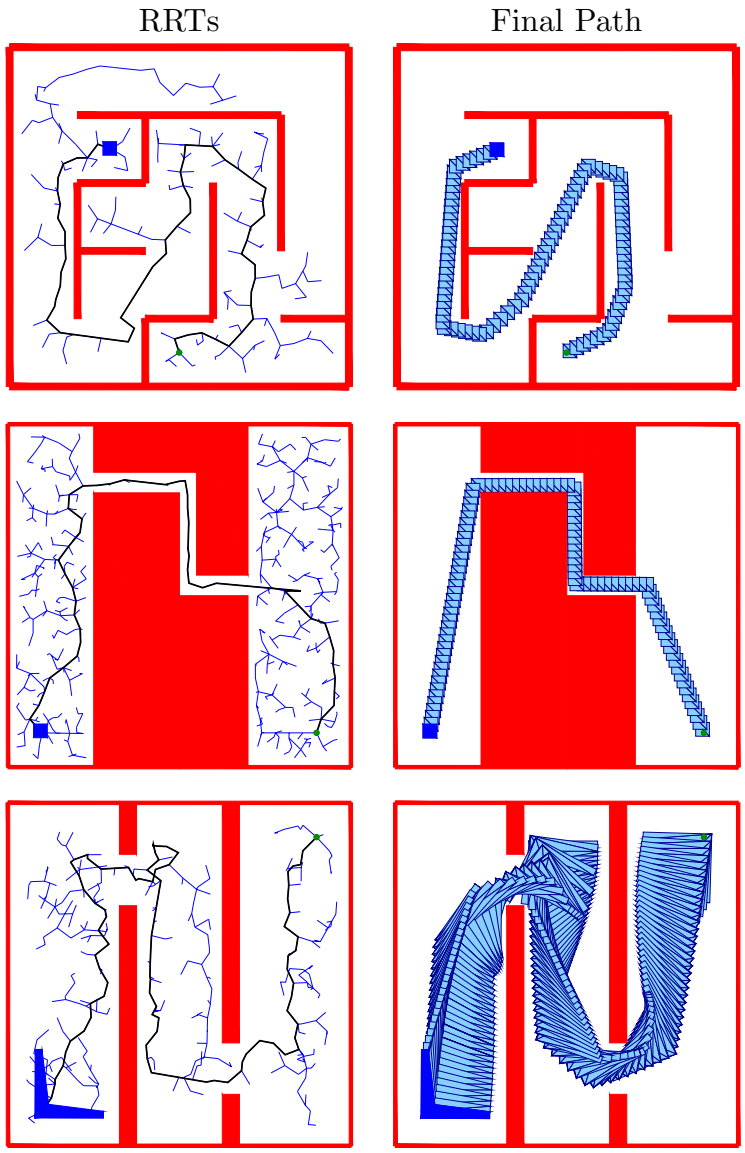
http://www.kuffner.org/james/plan/algorithm.php

RRT Probabilistic Completeness

- Probability a vertex is selected for extension is proportional to its area in Voronoi diagram
- RRTs converge to a uniform coverage of C-space as the number of samples increases



Examples of RRT-Connect

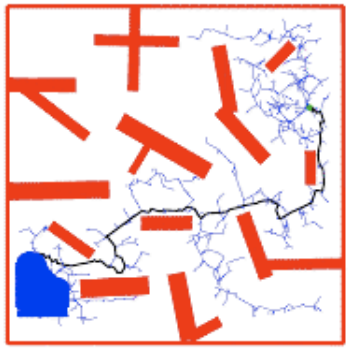


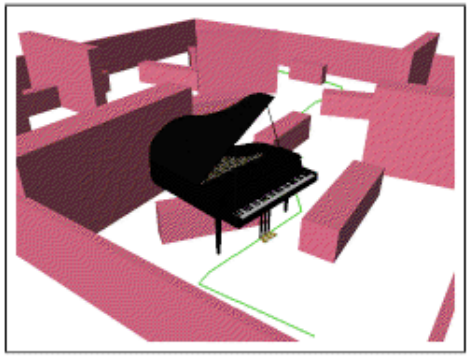
2 DOF maze

2 DOF single passway

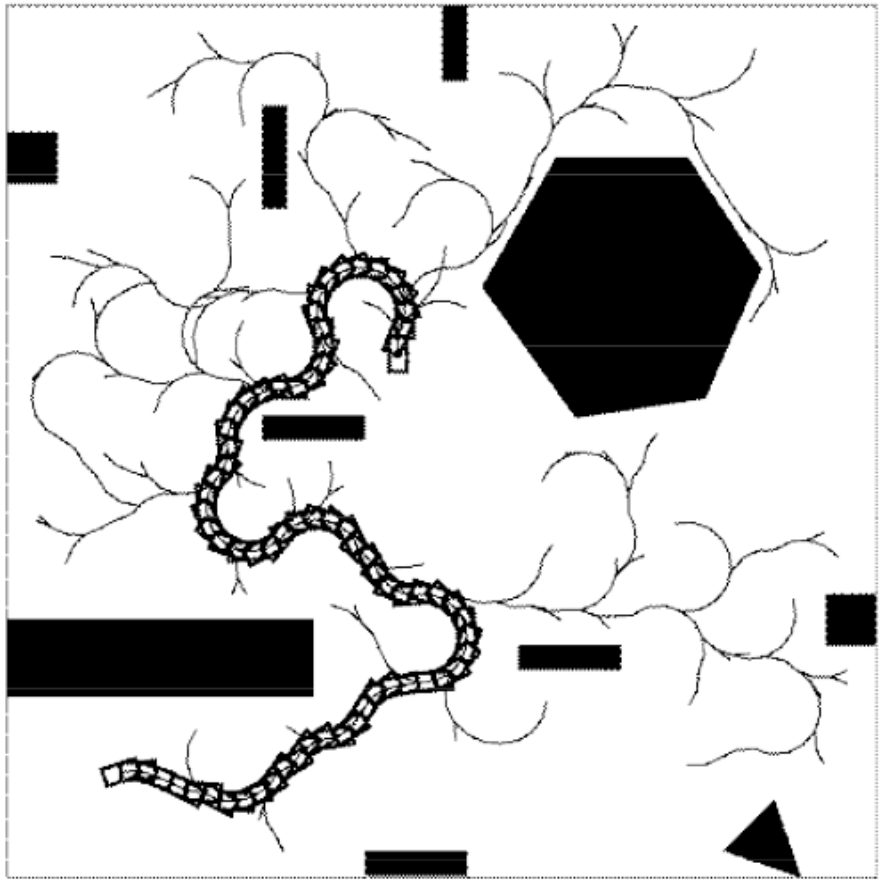
3 DOF single passway
(with non-point geometry)

Piano Mover's Problem

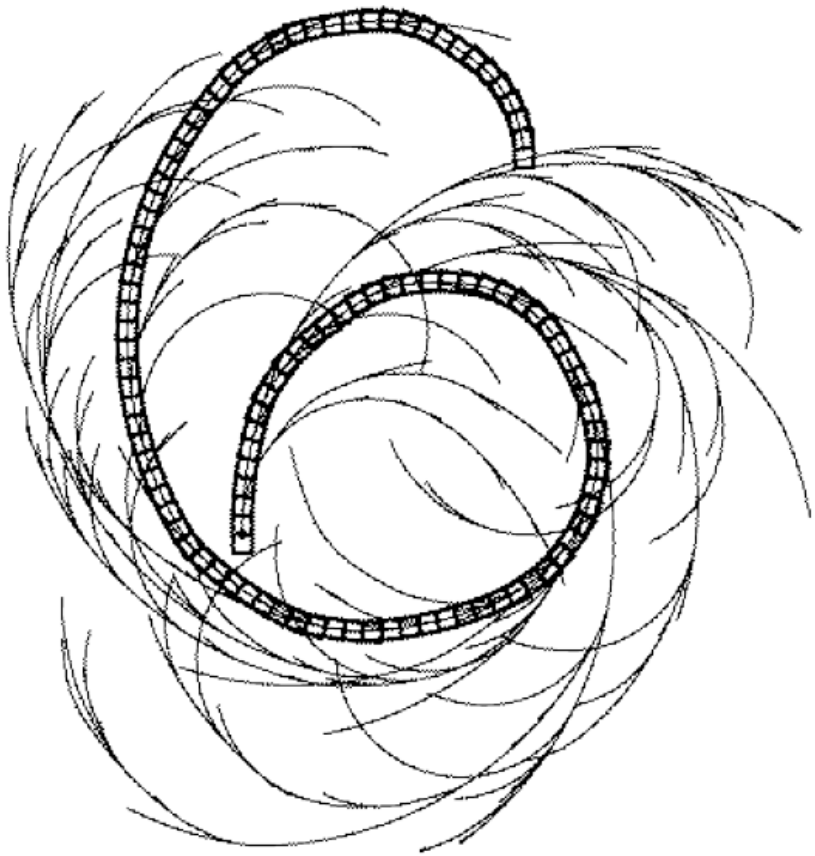




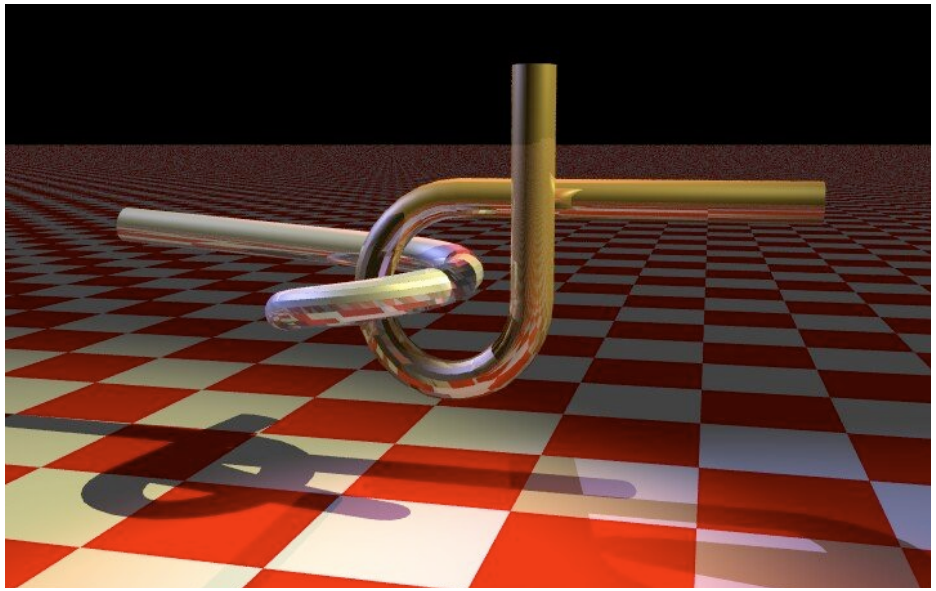
A Car-Like Robot

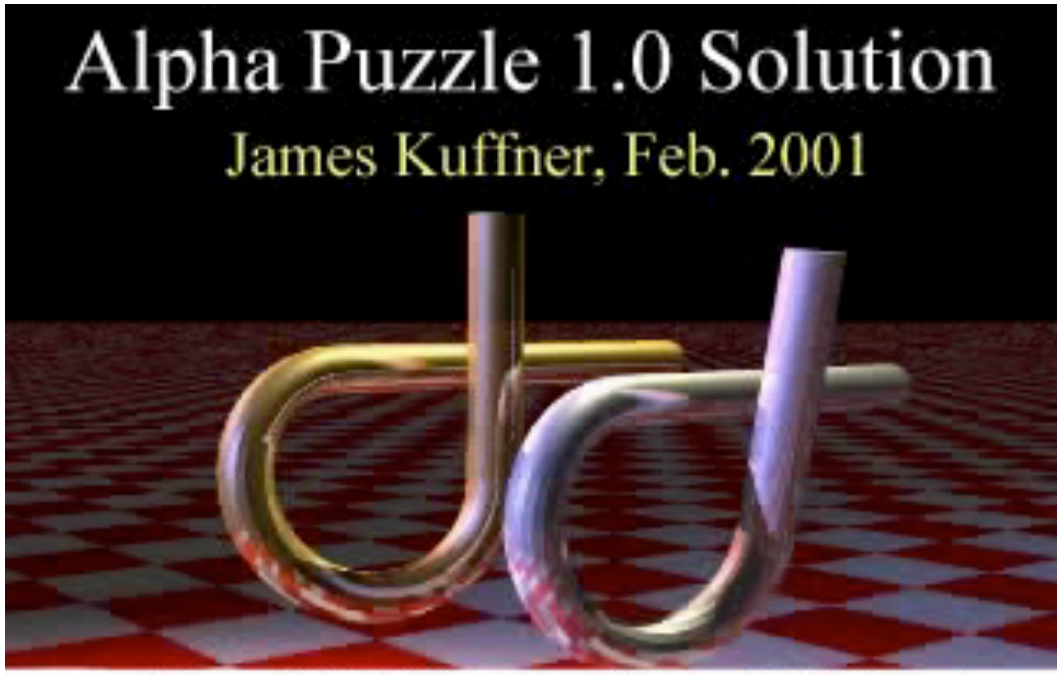


A Right-Turn Only Car



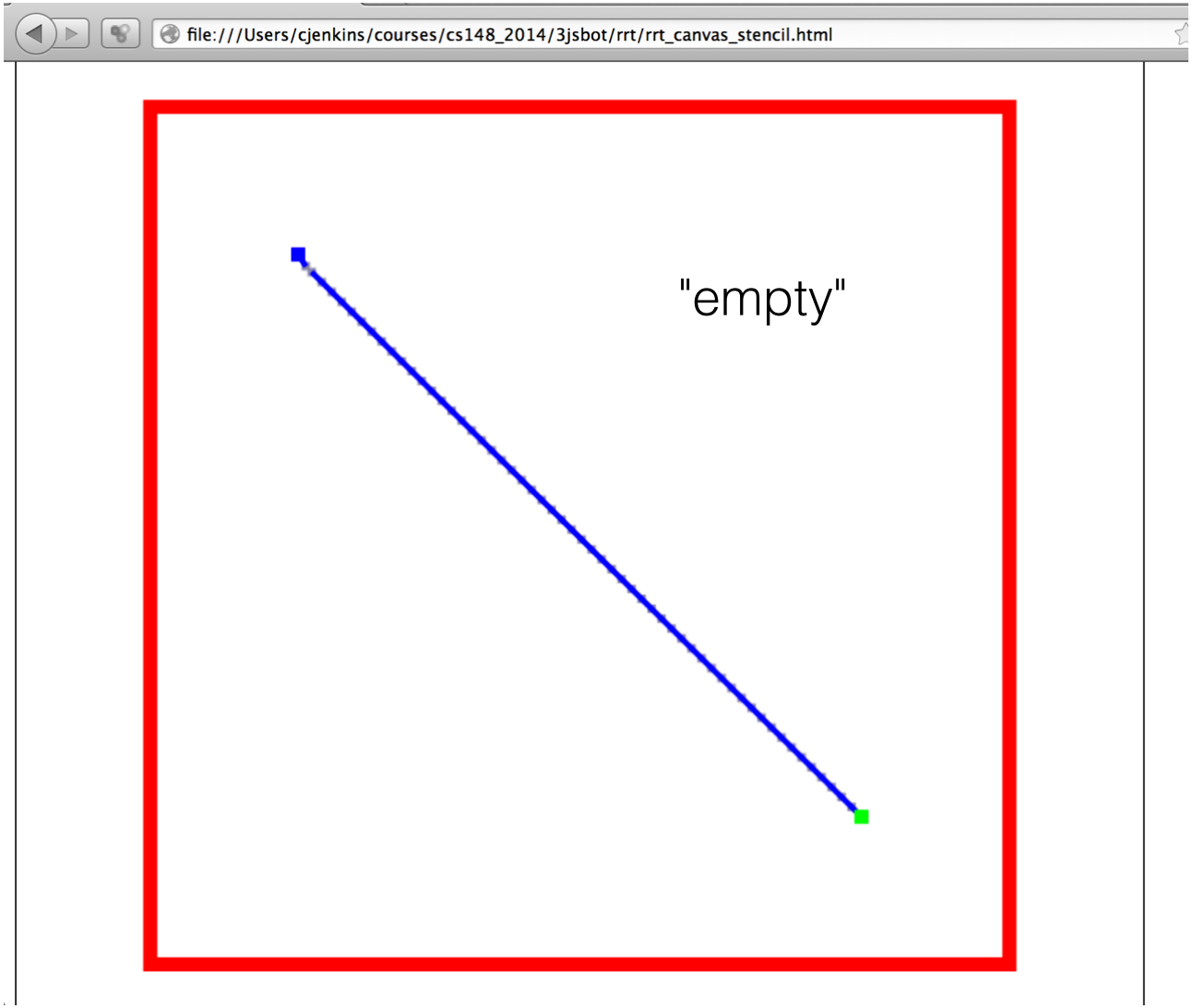
Alpha Puzzle

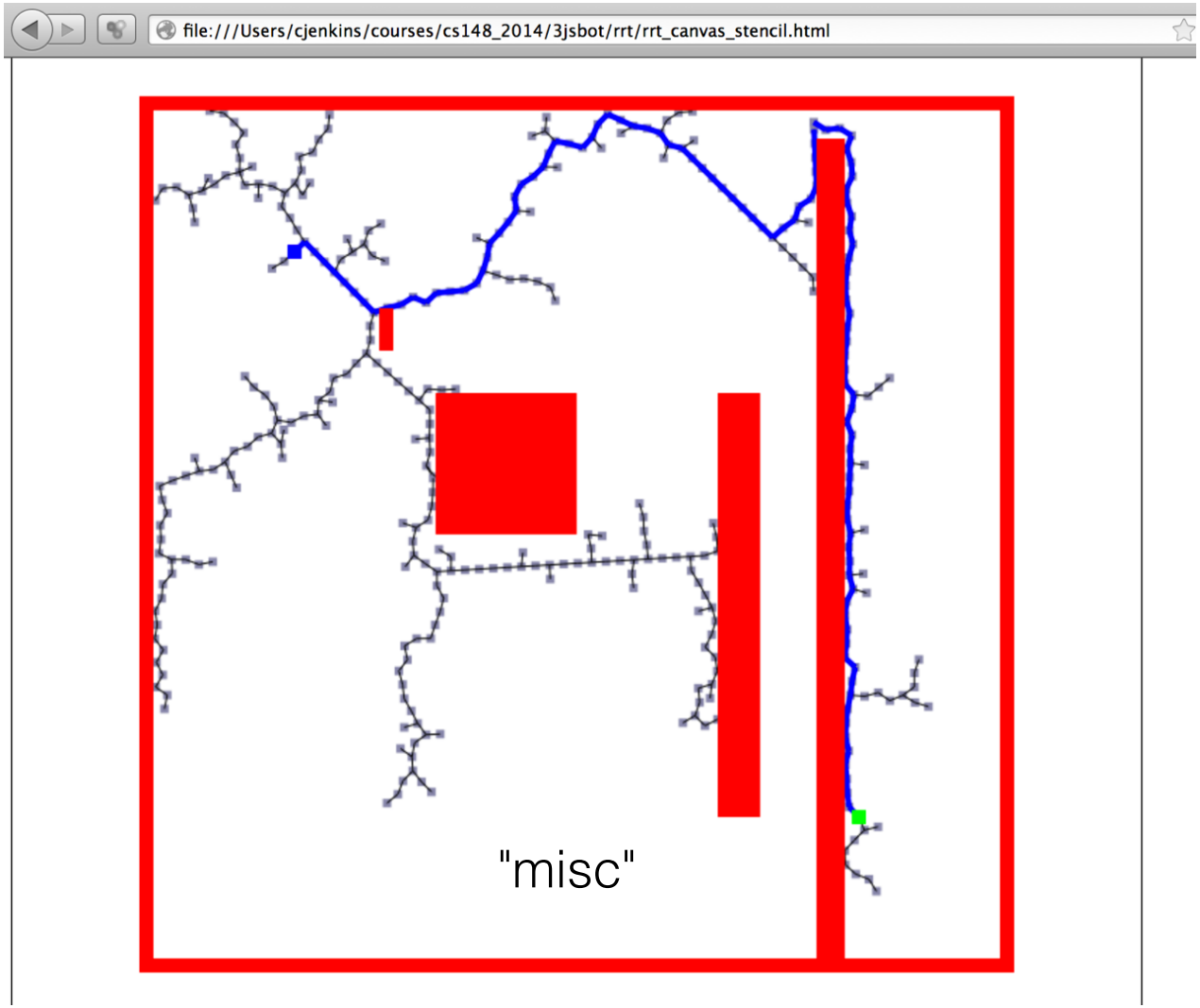


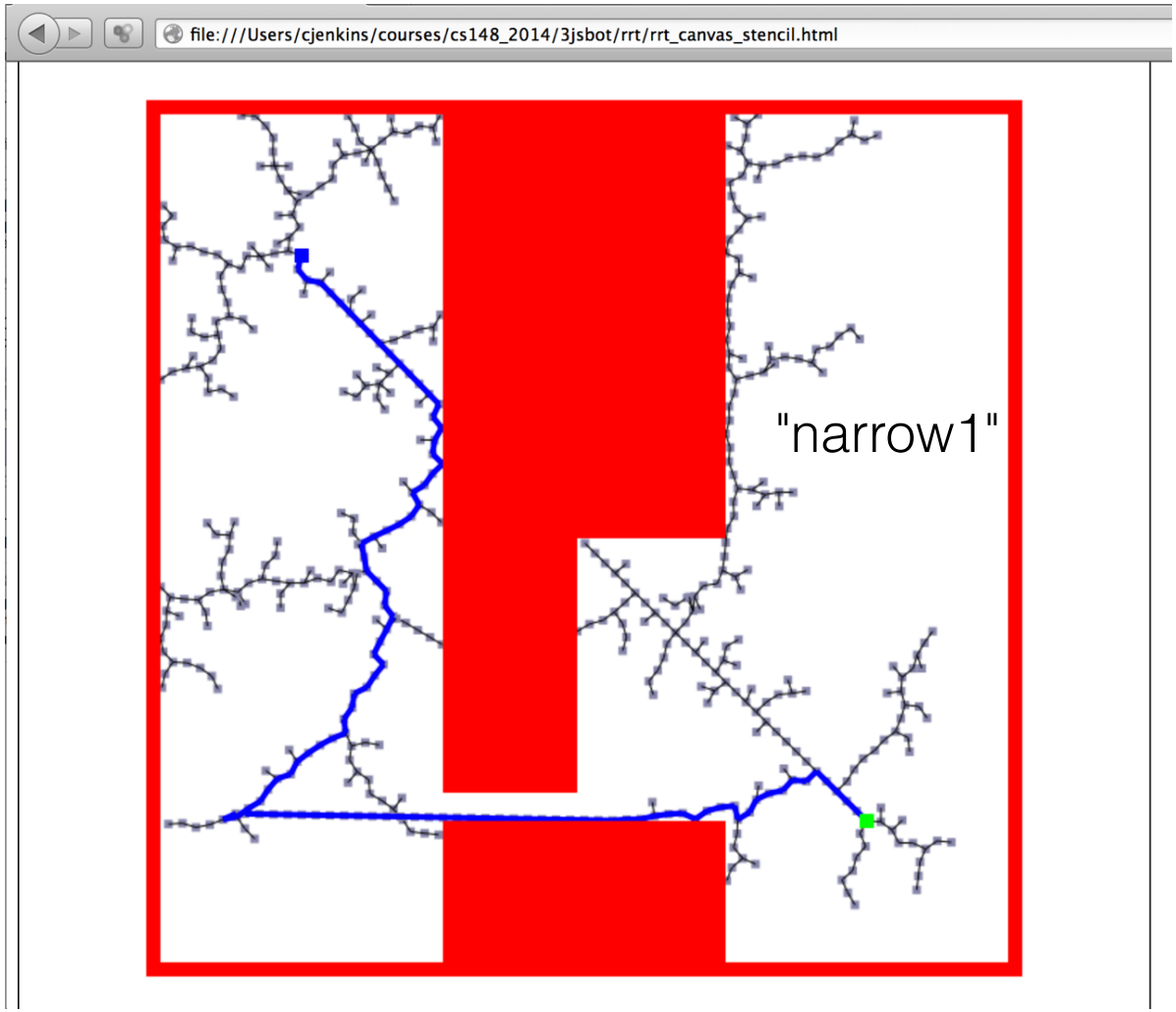


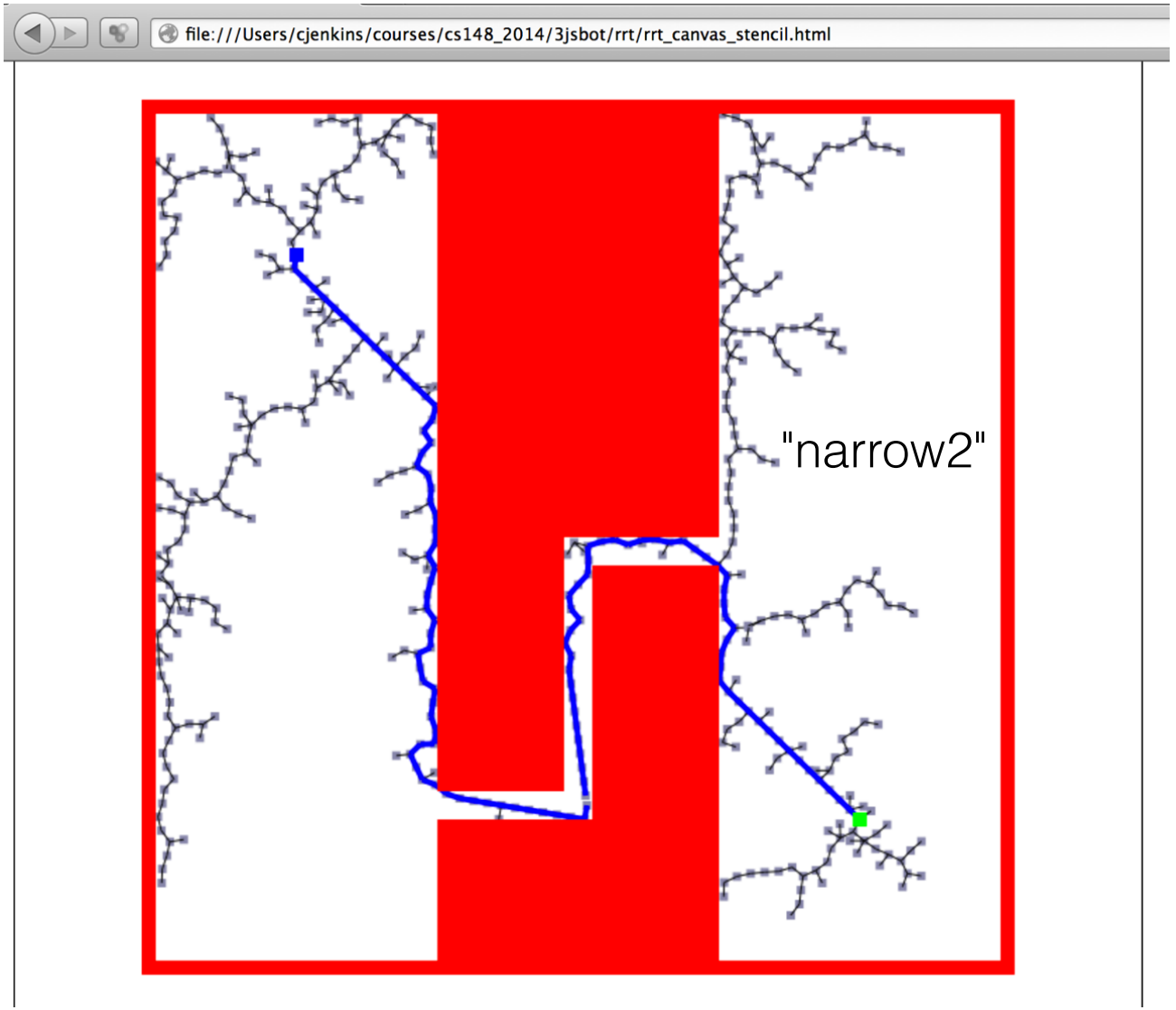
model by DSMFT group, Texas A&M Univ.
original model by Boris Yamrom, GE

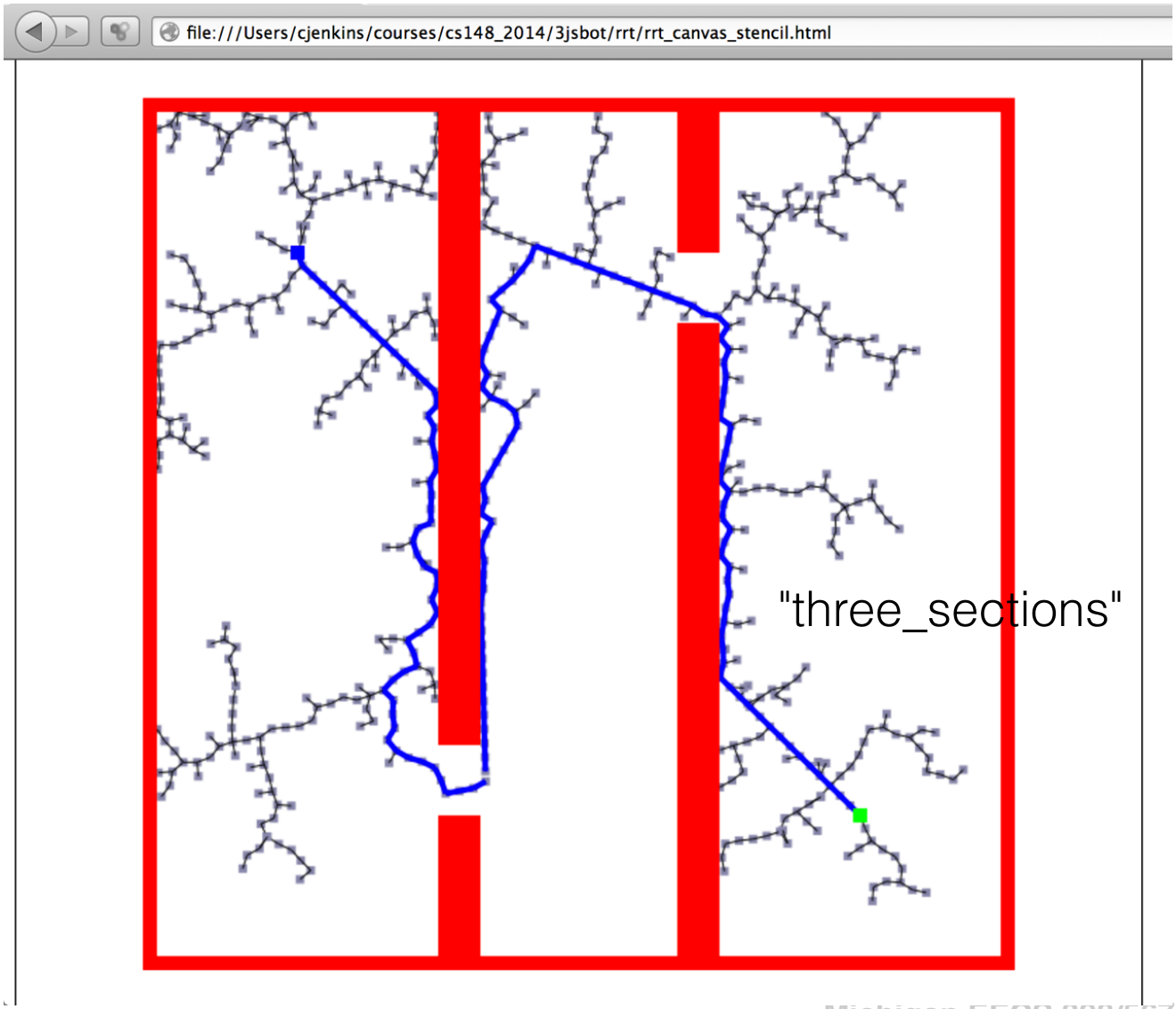
Canvas Stencil Examples


file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"empty"

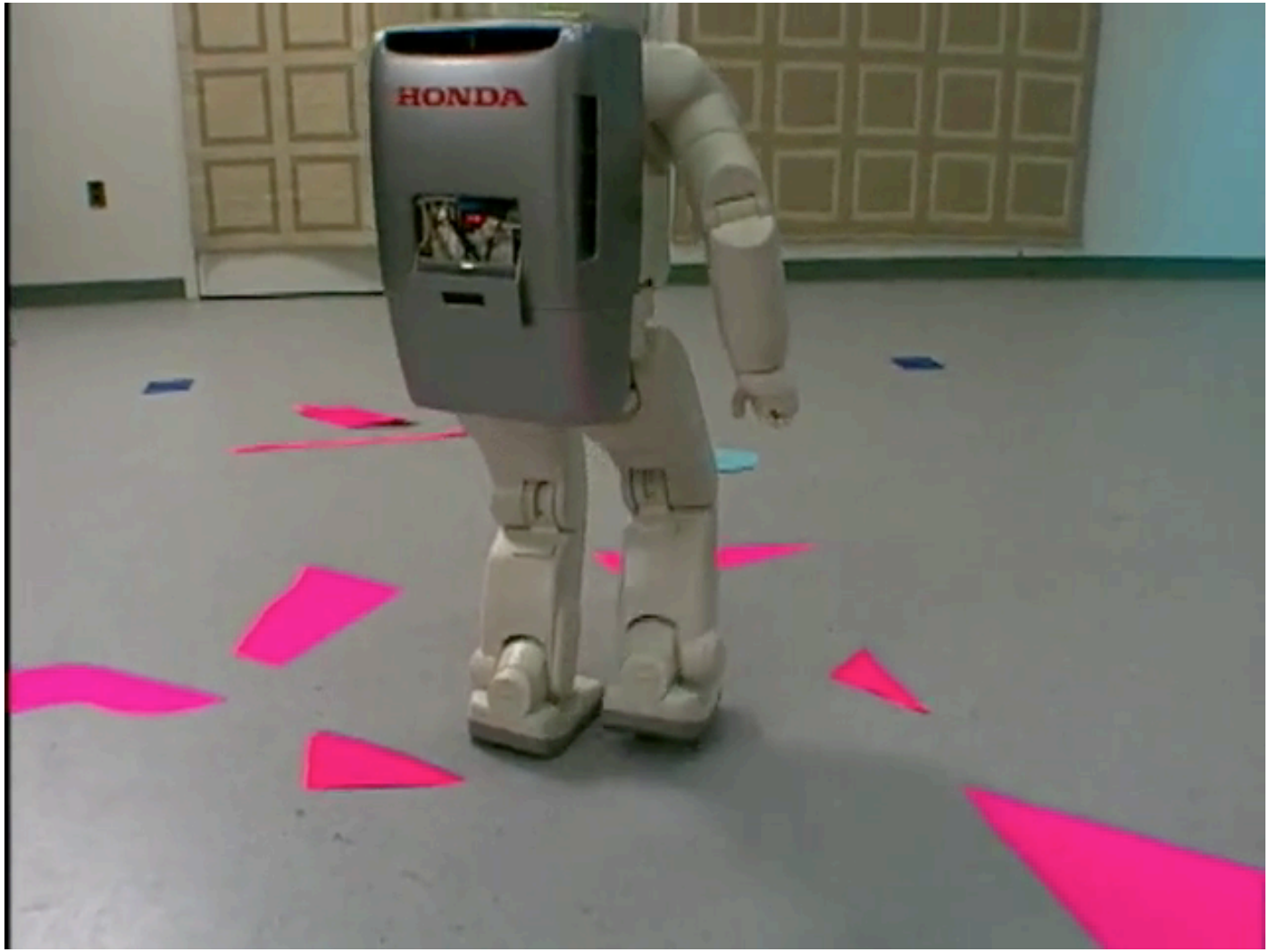

file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"misc"


file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"narrow1"


file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"narrow2"

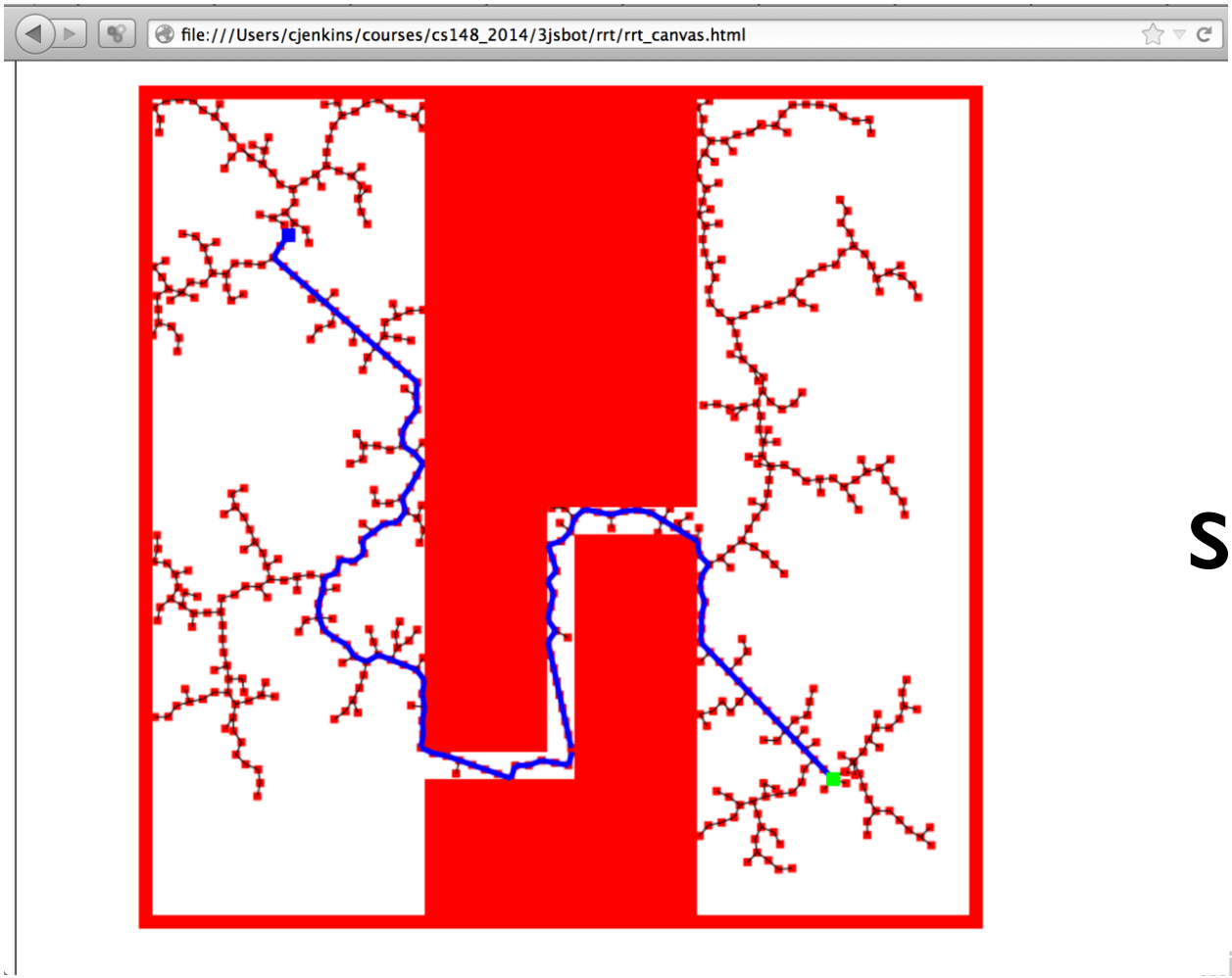

file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"three_sections"

“We’ve made robot history”



Kuffner/Asimo Discovery Channel feature - https://www.youtube.com/watch?v=wtVmbiTfm0Q

RRTs can take a lot of time...



Is there a simpler way?

Potential fields

follow your potential