

Planning with Sampling Roadmaps

Building a path to goal



EECS 398

Intro. to Autonomous Robotics

ME/EECS 567 ROB 510
Robot Modeling and Control

Fall 2018

autorob.org

Michigan EECS 398/567 ROB 510 - 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?

©2010 Google
©2015 Google

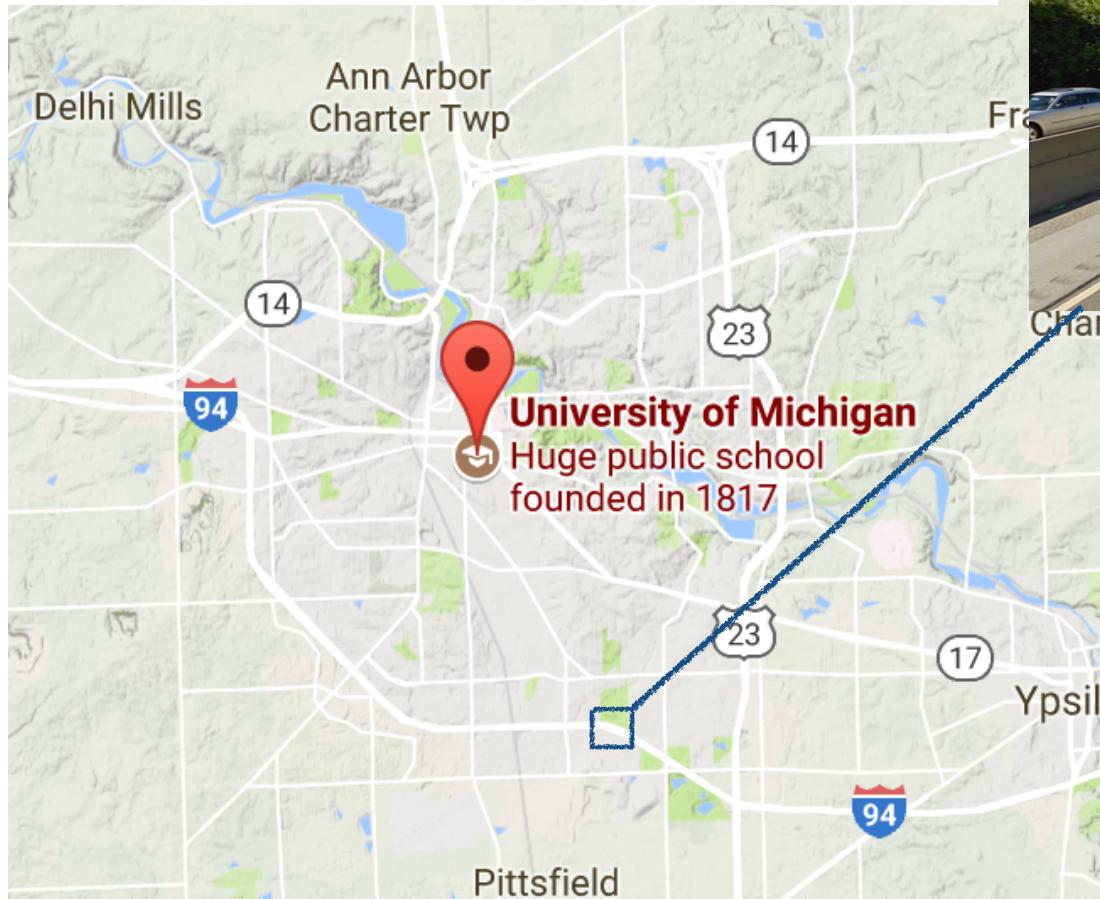
EXIT 177
State St
1 MILE



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

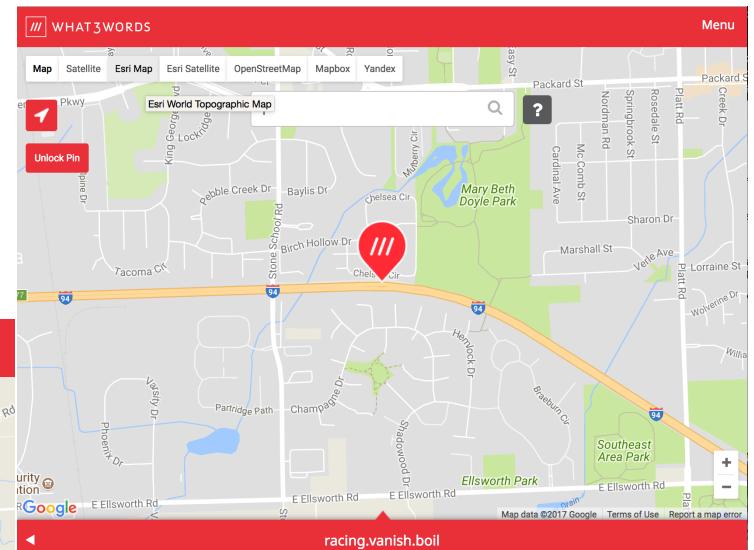
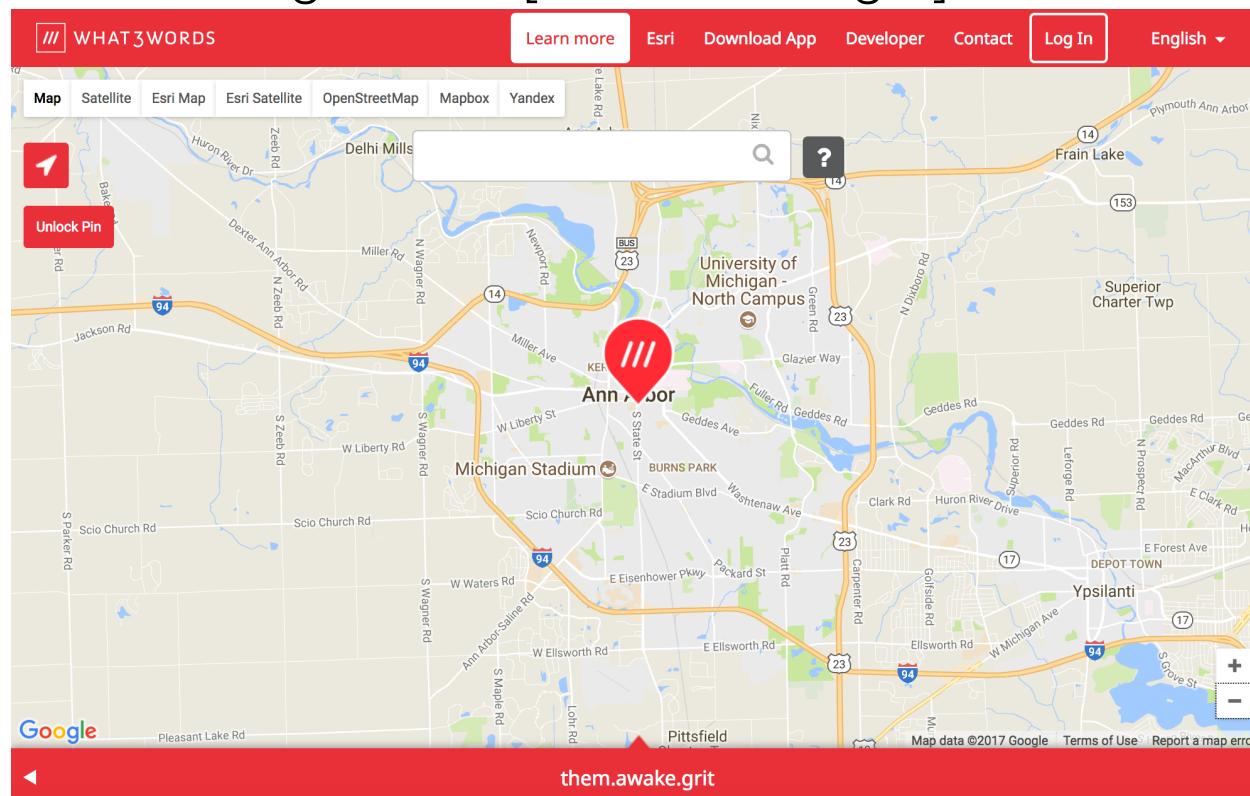
Michigan EECS 398/567 ROB 510 - autordb.org

Configuration: [42.2780436,-83.7404128]



Alternatively...

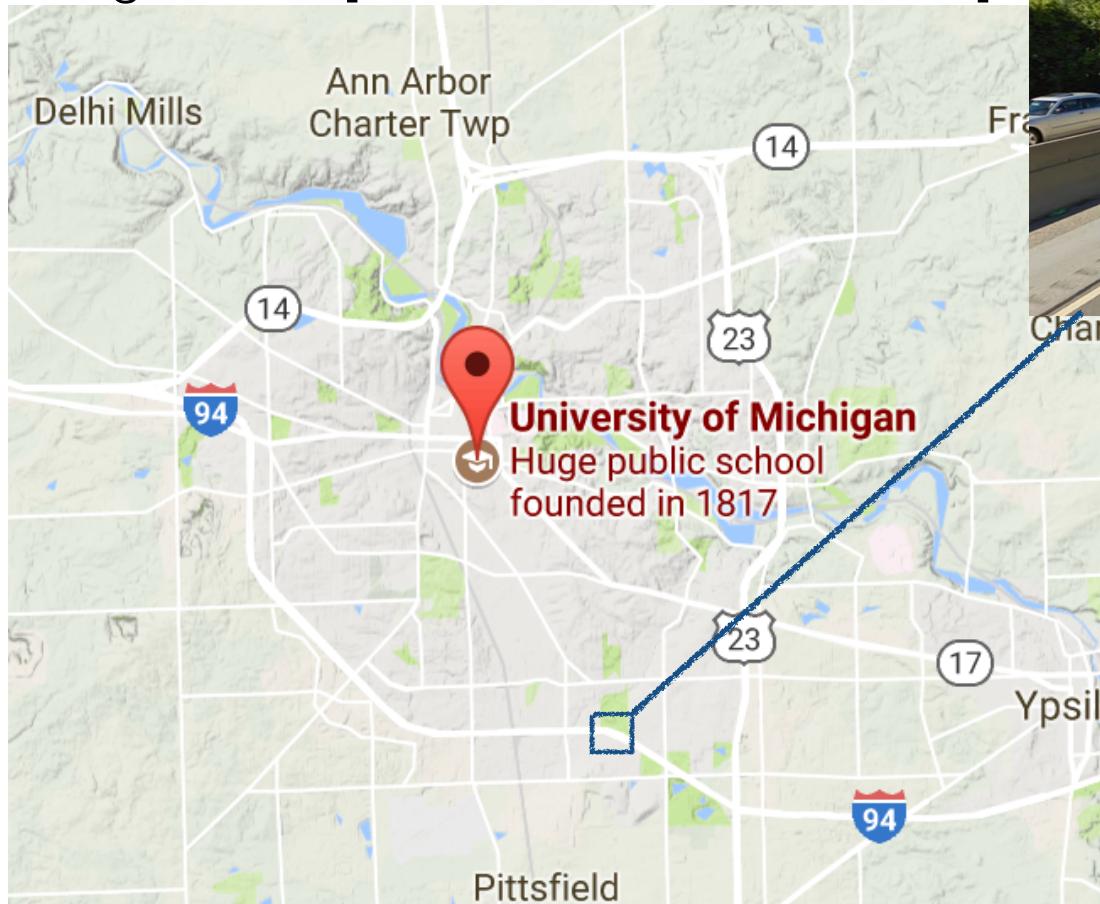
Configuration: [them.awake.grit]

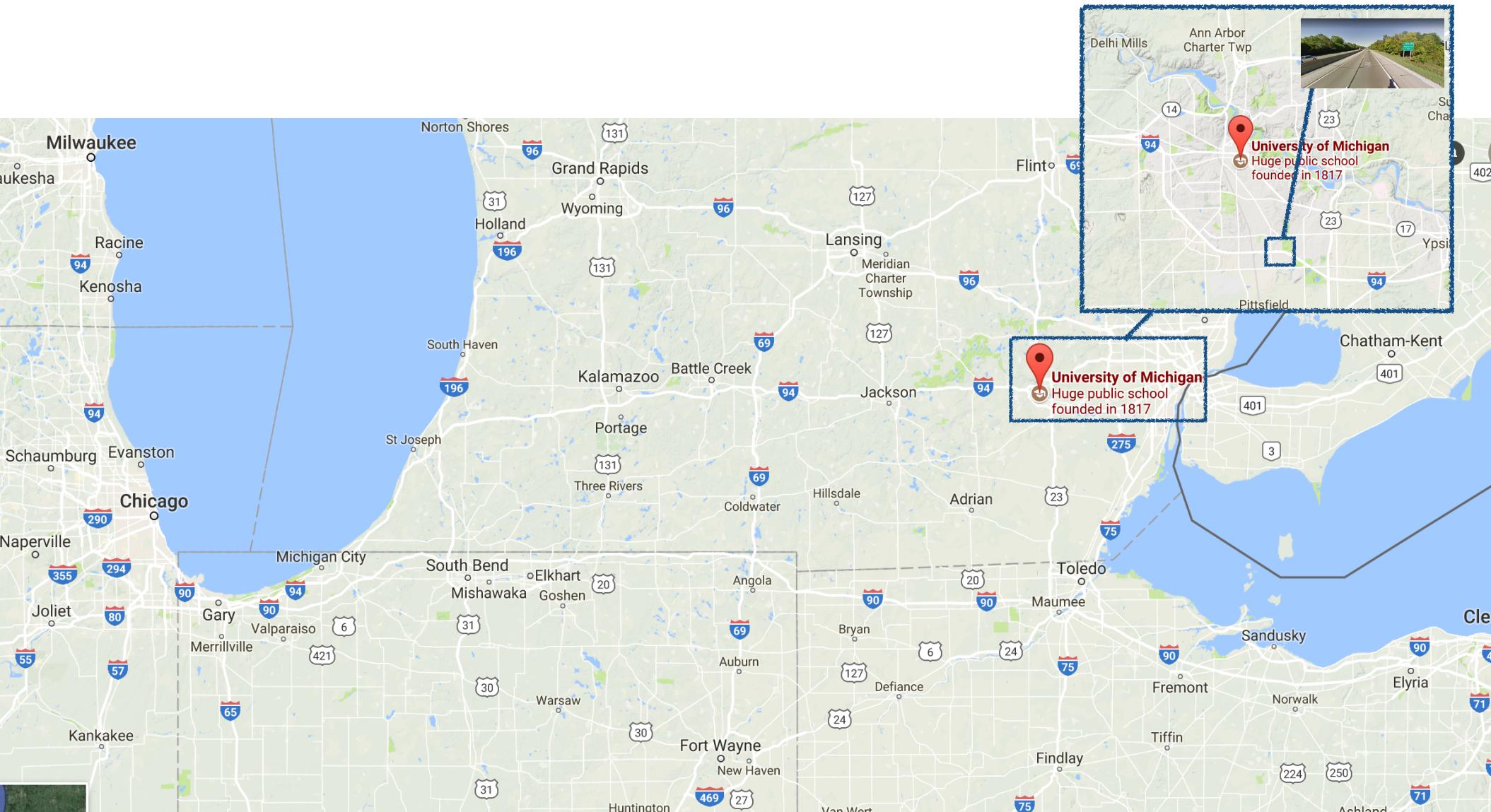


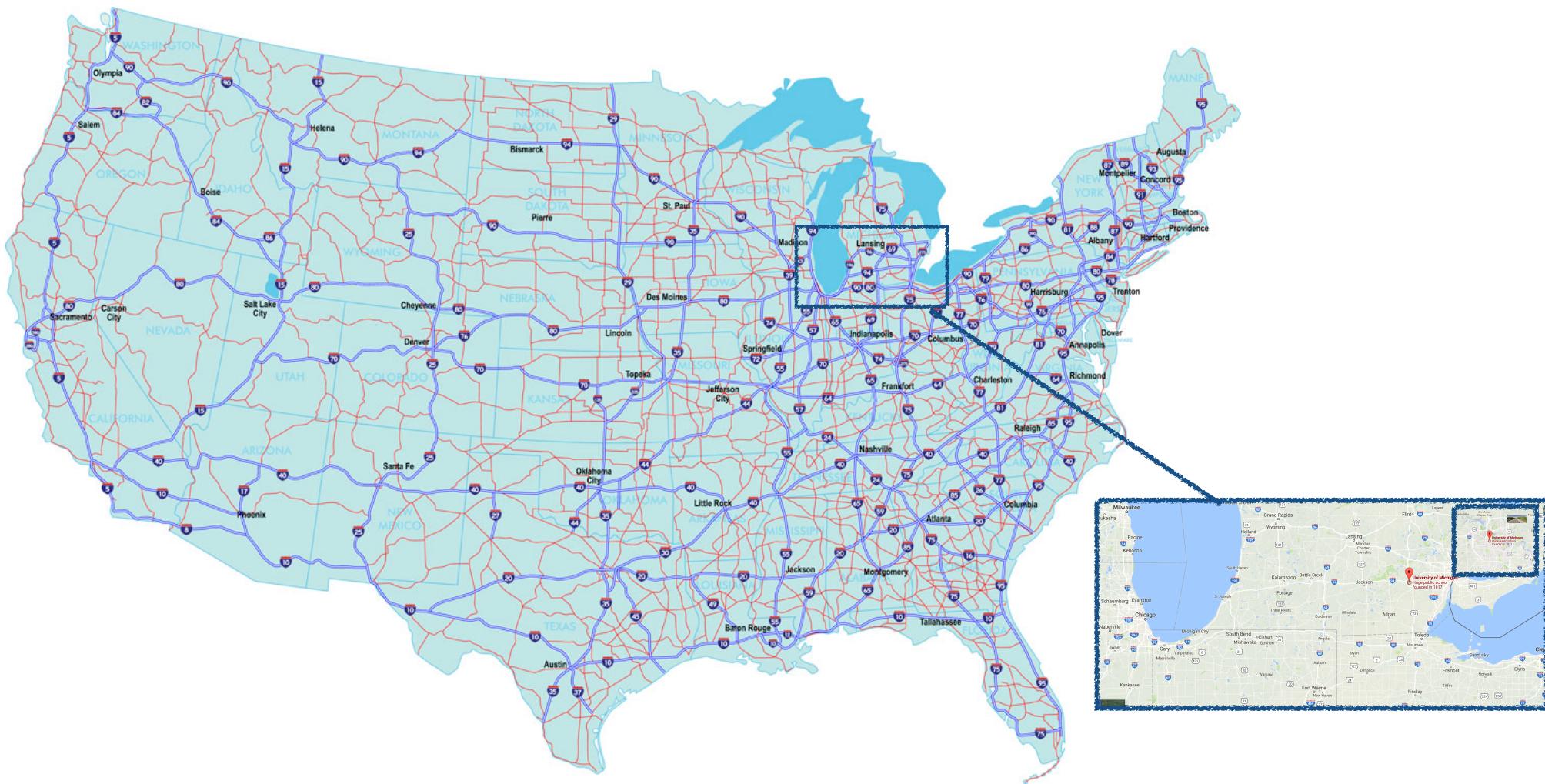
Configuration (lat,lon): [racing.vanish.boil]

Is this a usable
representation for
configuration?

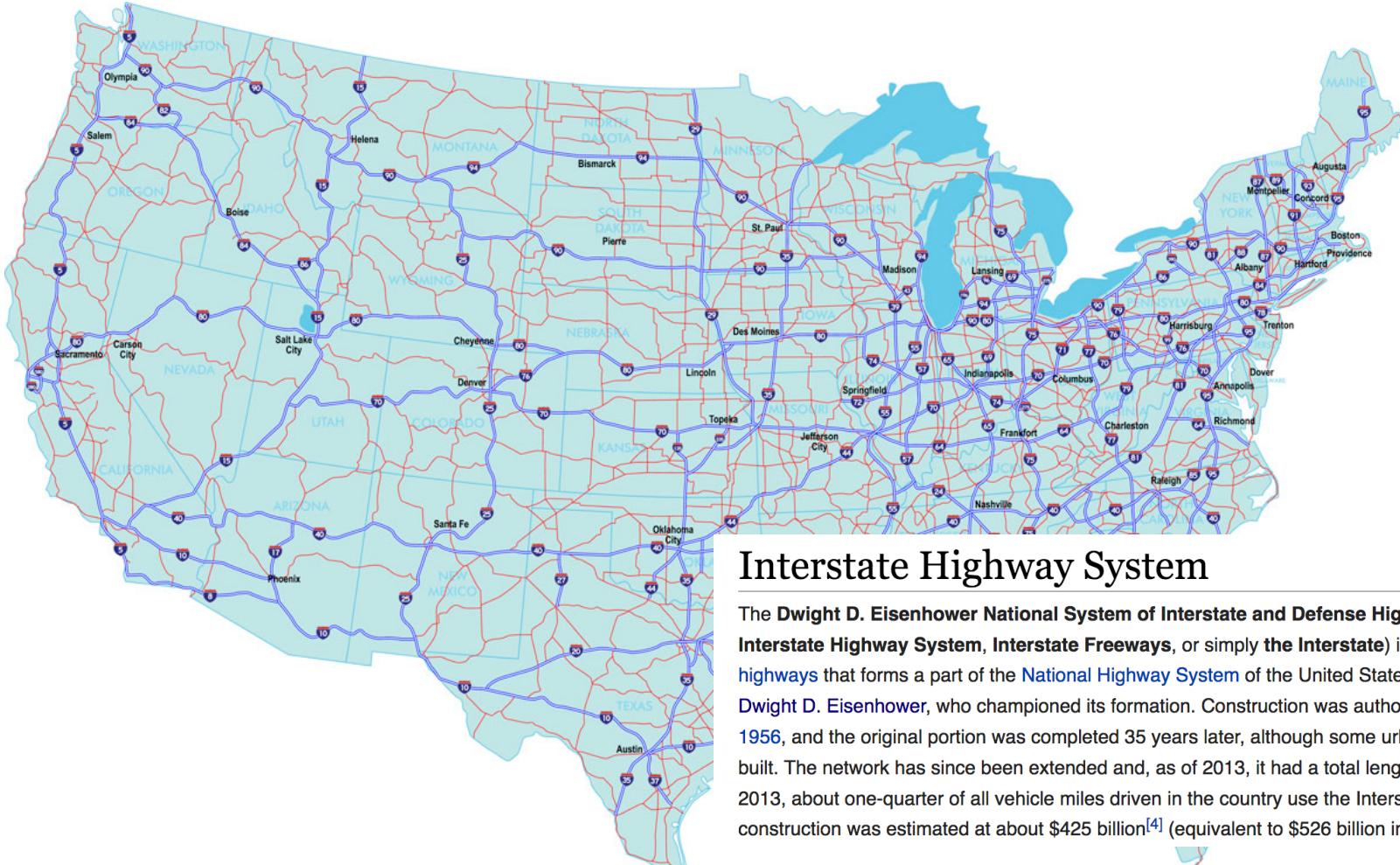
Configuration: [42.2780436, -83.7404128]







Michigan EECS 398/567 ROB 510 - autorob.org



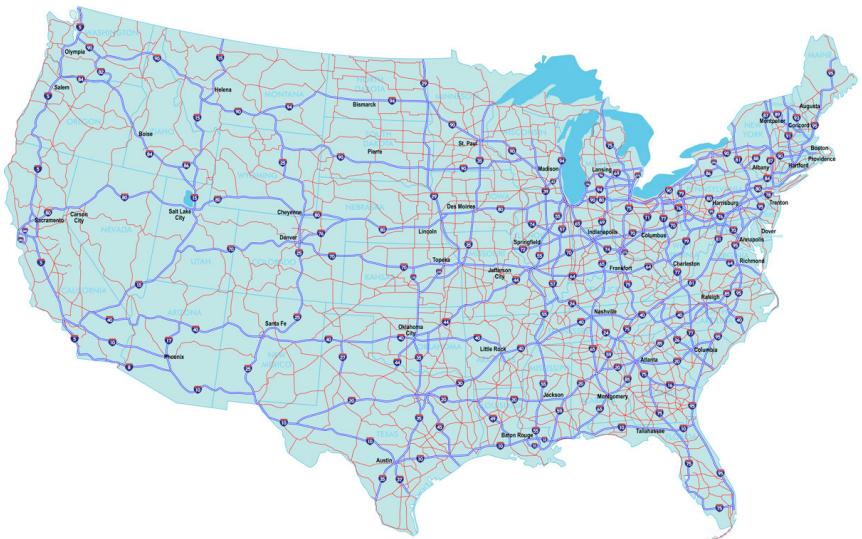
Interstate Highway System

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]).

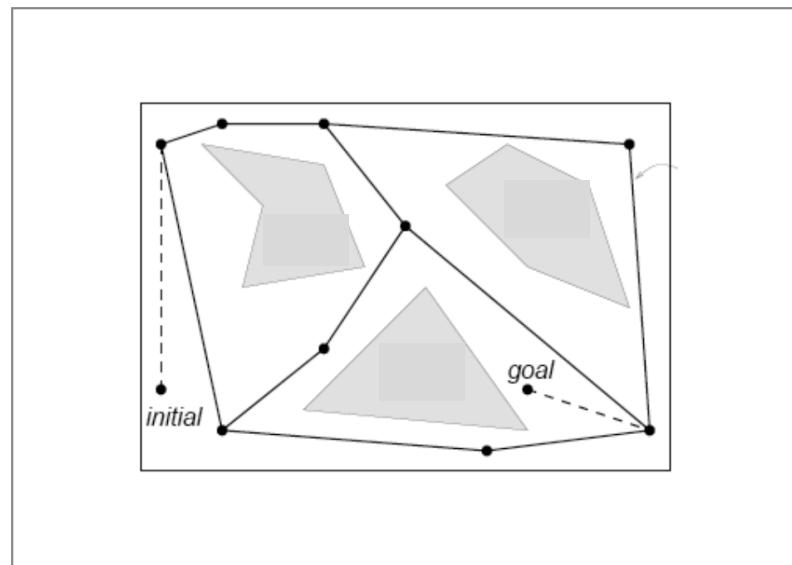
CHINA'S PROPOSED BELT AND ROAD INITIATIVE



Roadmaps



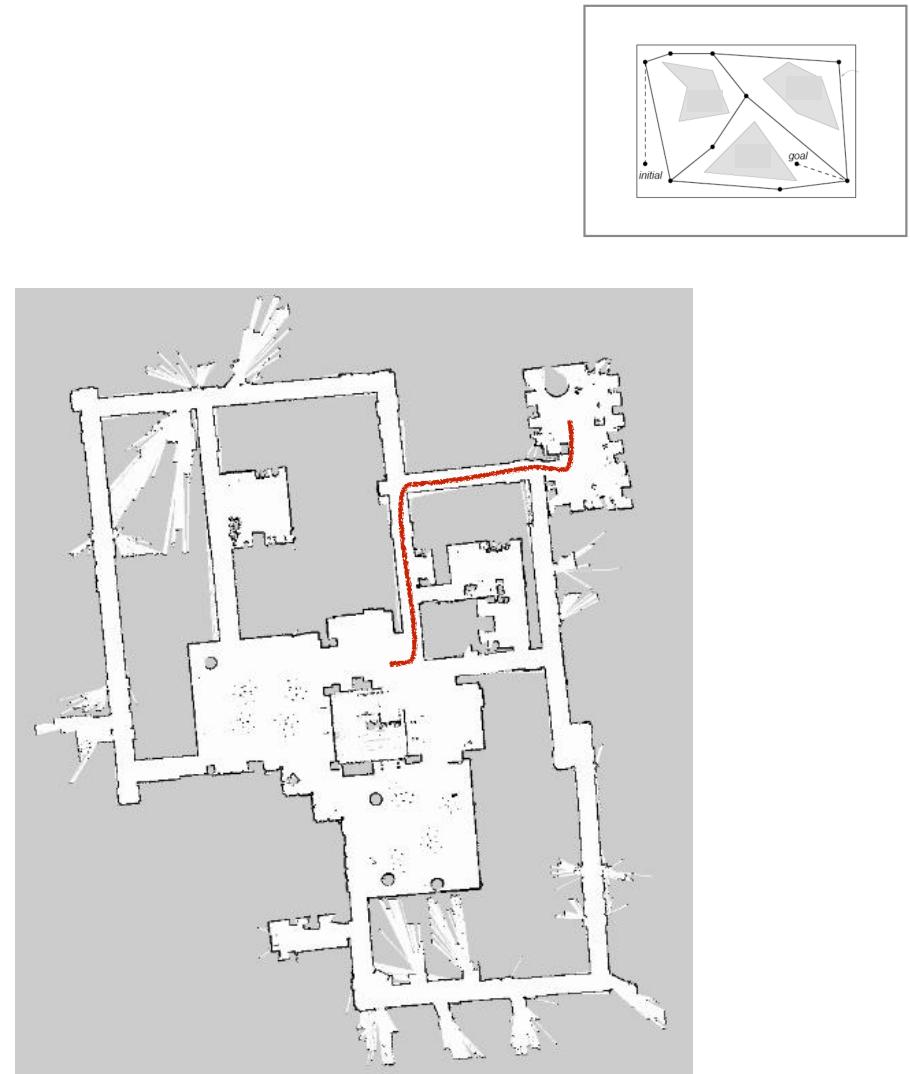
Roadmap over geolocations



Roadmap over robot configurations

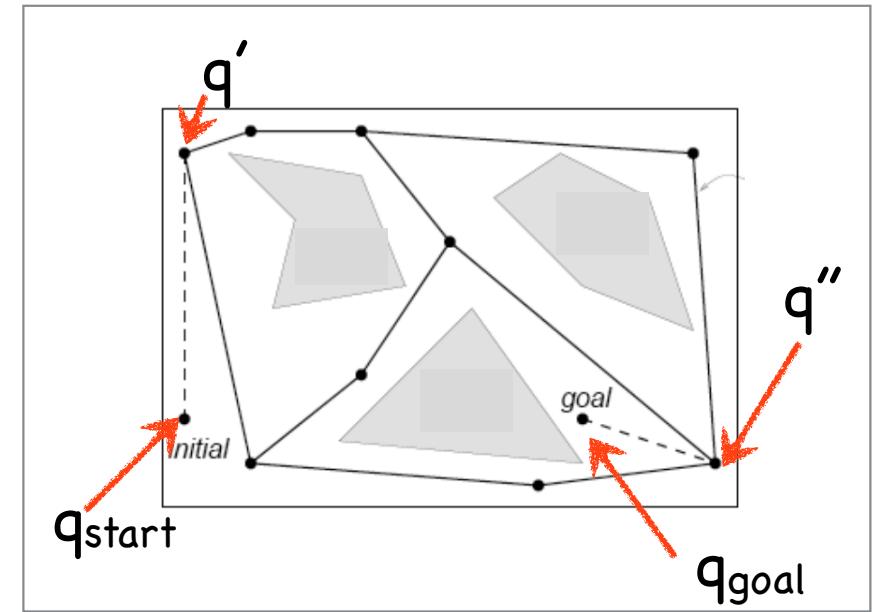
Roadmaps

- 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



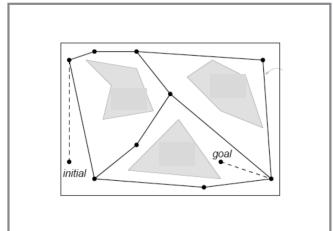
Roadmap Definition

- 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''

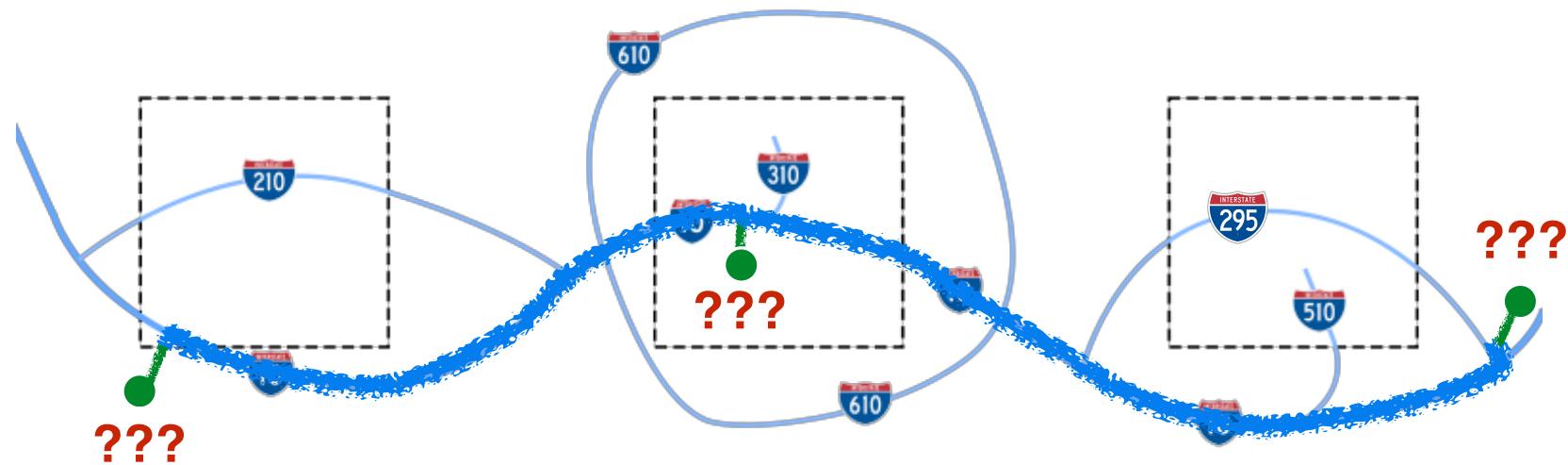
What cities? What universities?



CITY A ???

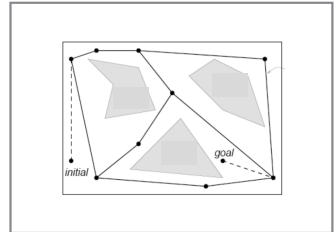
CITY B ???

CITY C ???



(to your best approximation)

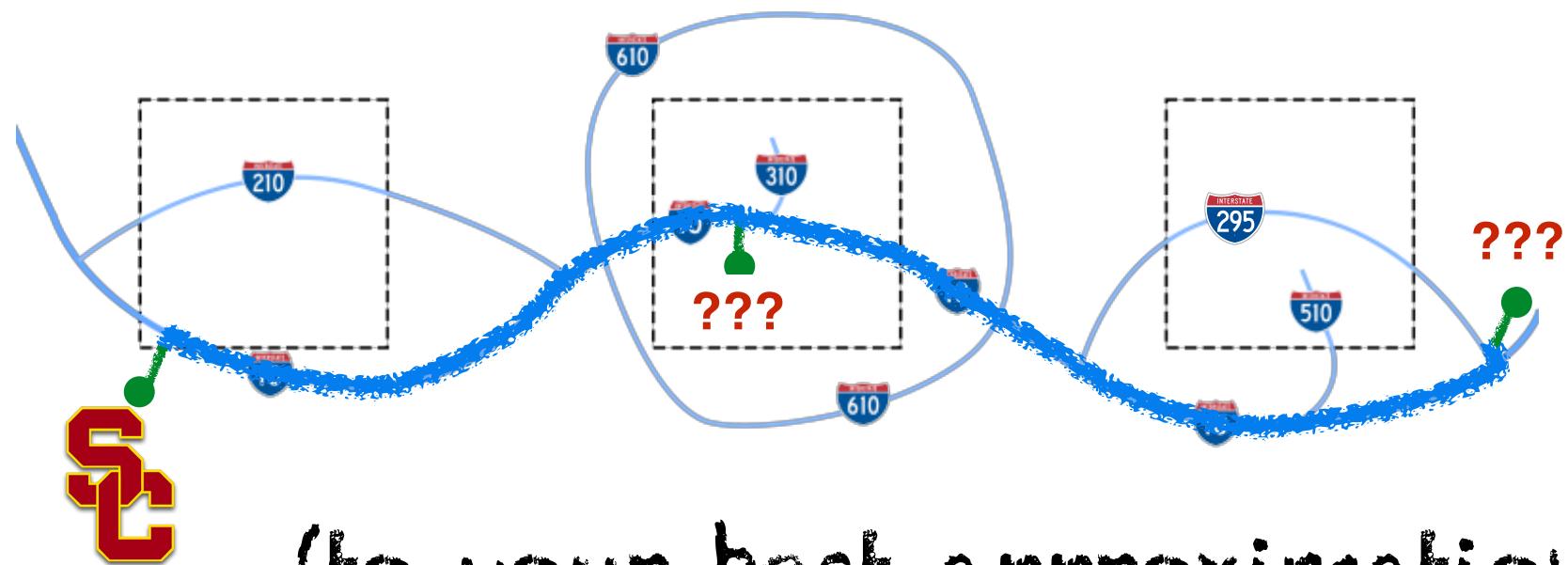
What cities? What universities?



“Los Angeles”

CITY B ???

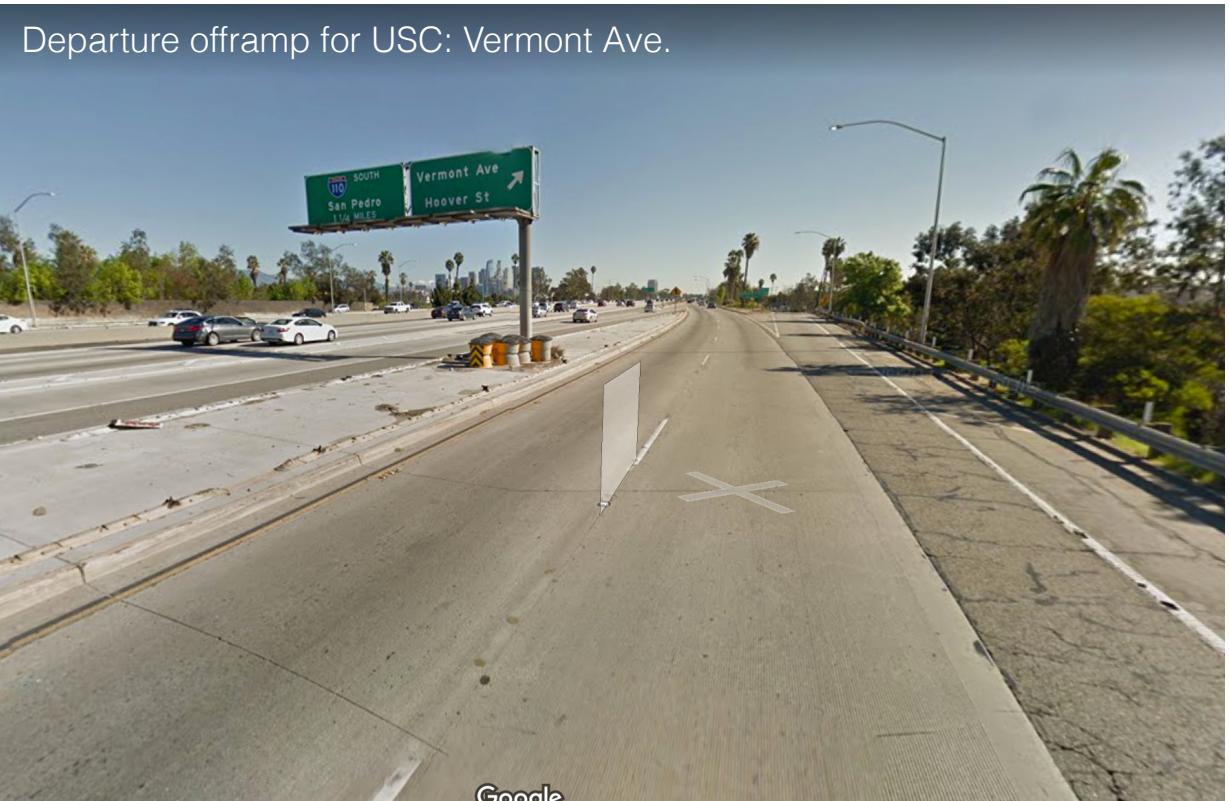
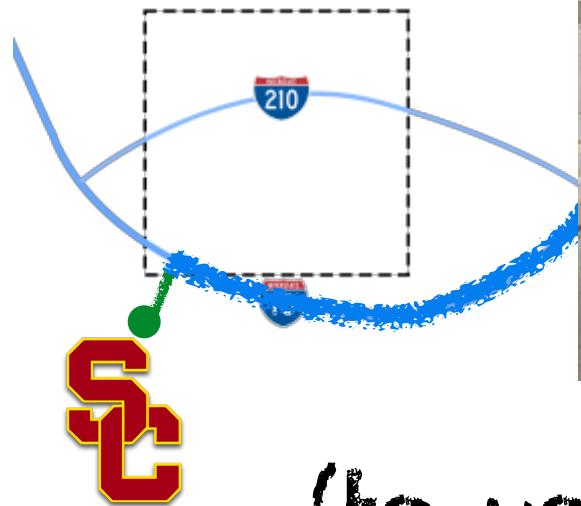
CITY C ???



(to your best approximation)

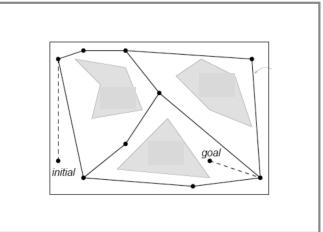
What cities? □

“Los Angeles”



(to your best approximation)

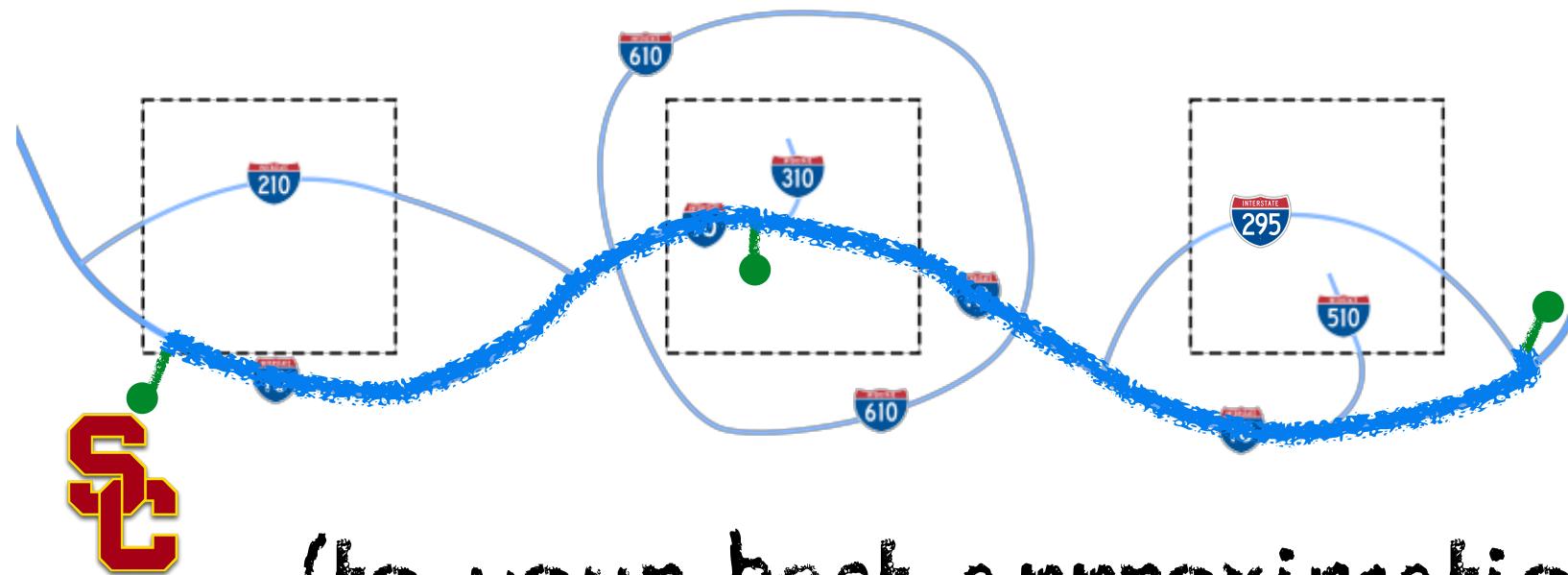
What cities? What universities?



“Los Angeles”

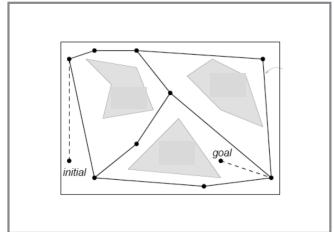
???????

?????????



(to your best approximation)

What cities? What universities?



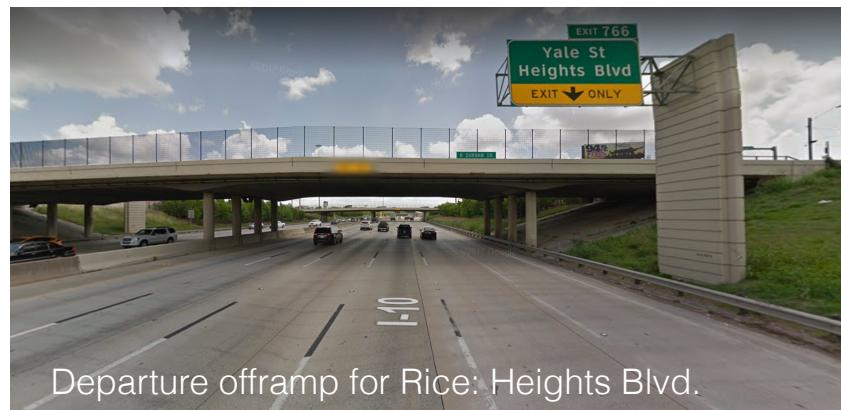
“Los Angeles”

“Houston”

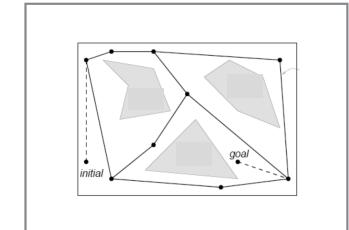
?????????



(to your best approximation)



that universities?



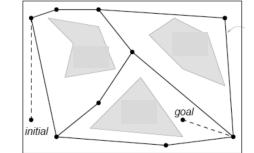
"Houston"

????????



(to your best approximation)

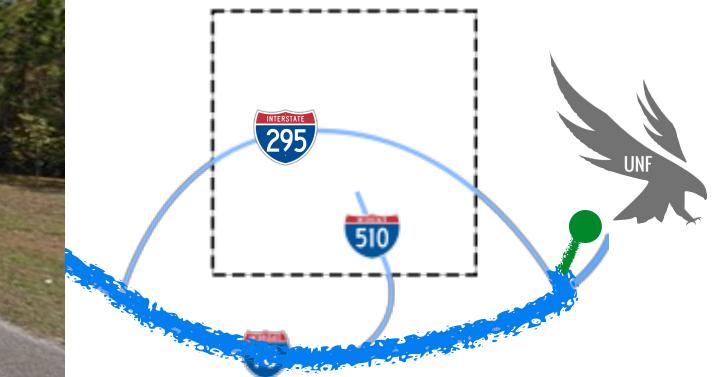
What cities? What universities?



Departure offramp for North Florida: Town Center Pkwy.



“Jacksonville”



(to your best approximation)

Departure offramp for Michigan: State St.

@2015 Google



EXIT 177
State St
1 MILE

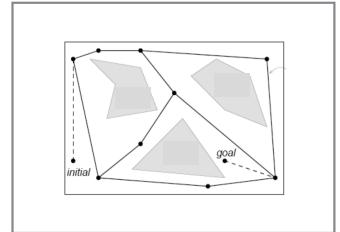
Better departure offramp for Michigan: Plymouth Rd.

@2016 Google



US 23

Basic Roadmap Planner



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

- ④ 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
- ④ 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

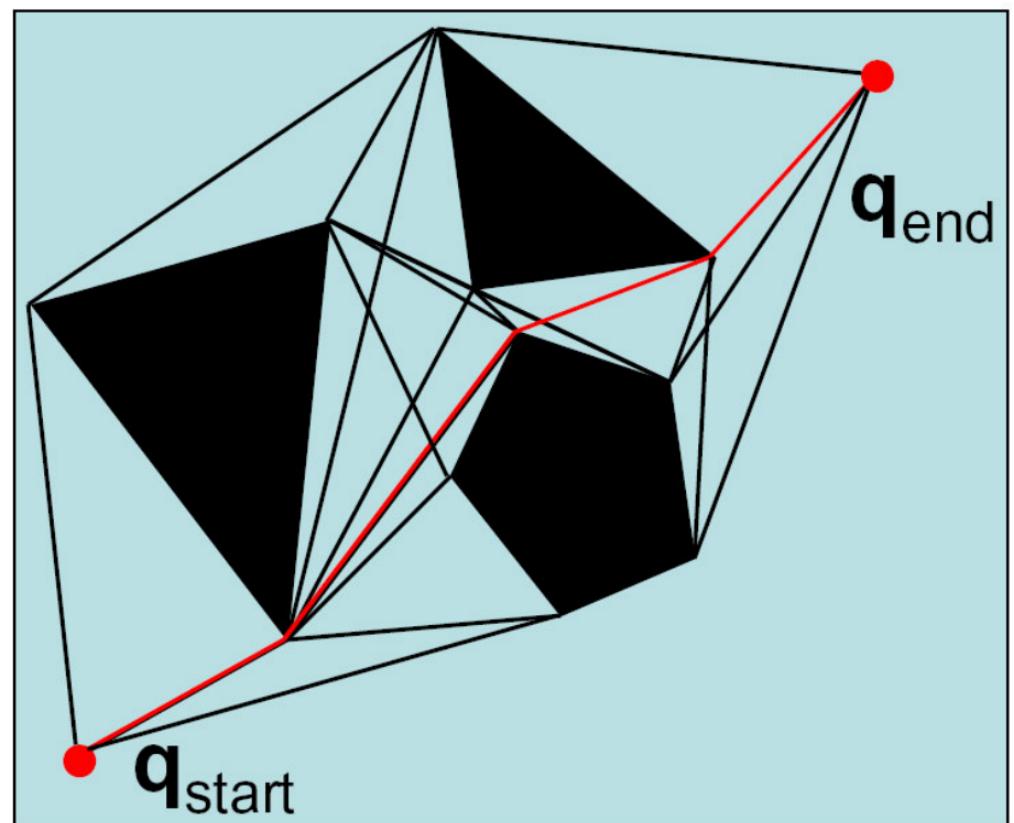
- Probabilistic Roadmap (PRM)
 - sample and connect vertices in graph for multiple planning queries
- Rapidly-exploring Random Tree (RRT)
 - sample and connect vertices in trees rooted at start and goal configuration

2 Approaches to Roadmaps

Deterministic:

complete algorithms

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

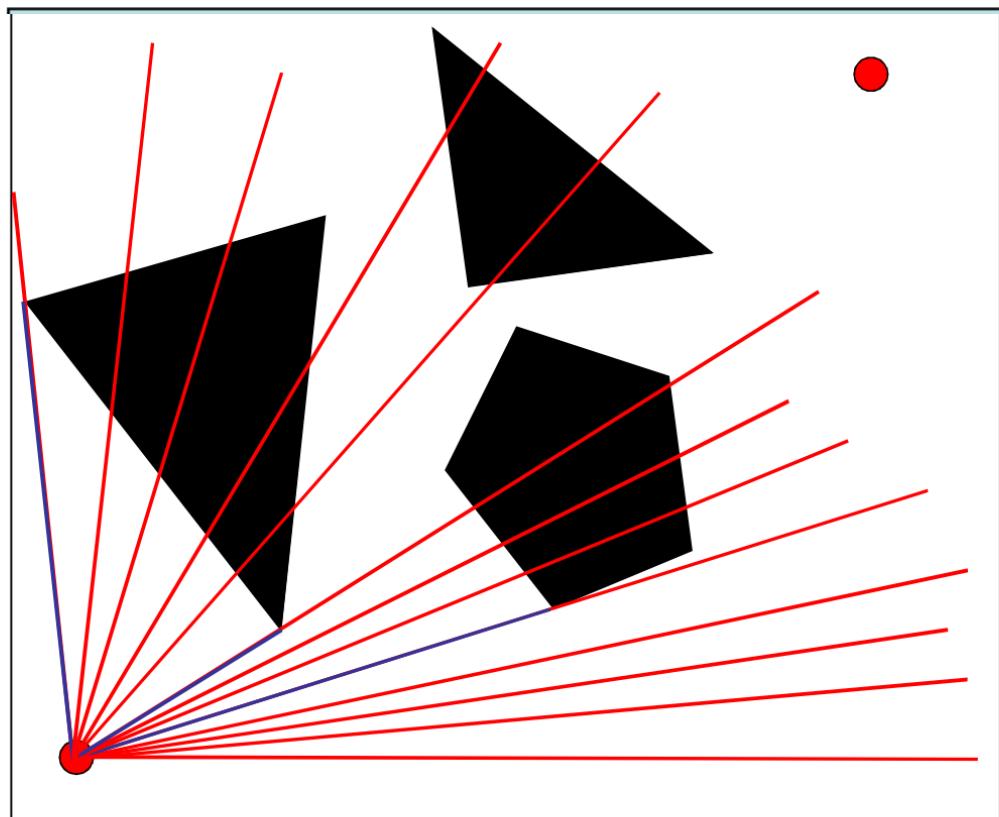


2 Approaches to Roadmaps

Deterministic:

complete algorithms

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

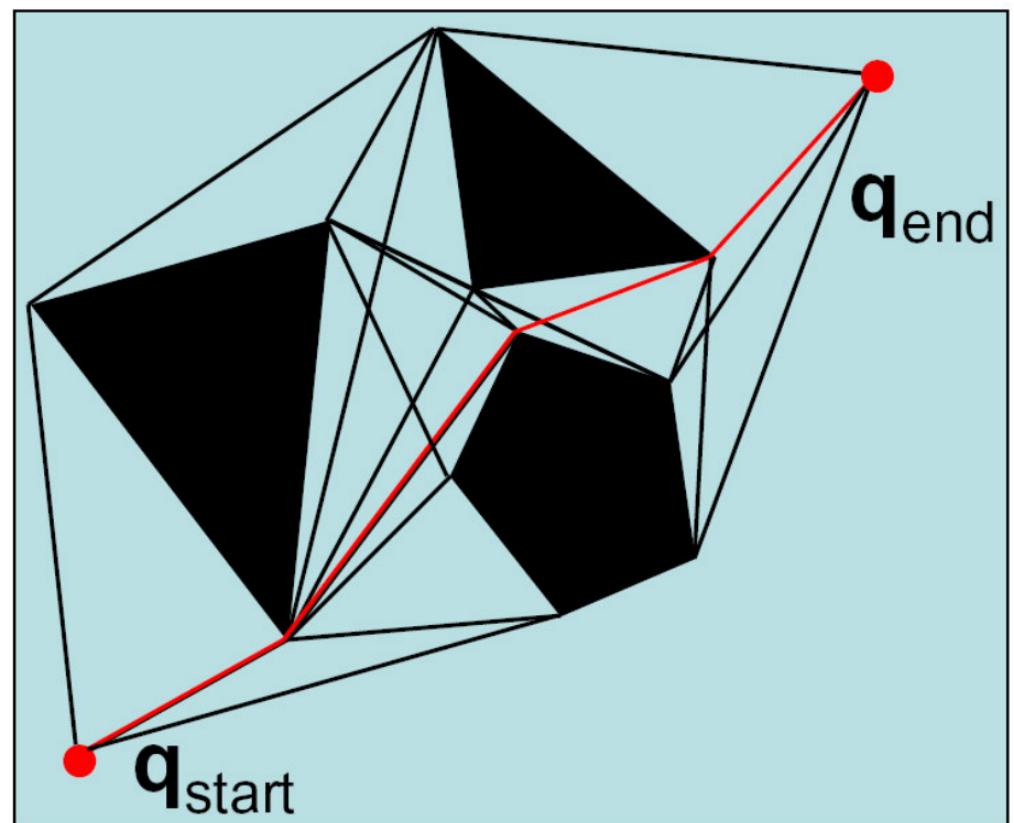


2 Approaches to Roadmaps

Deterministic:

complete algorithms

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

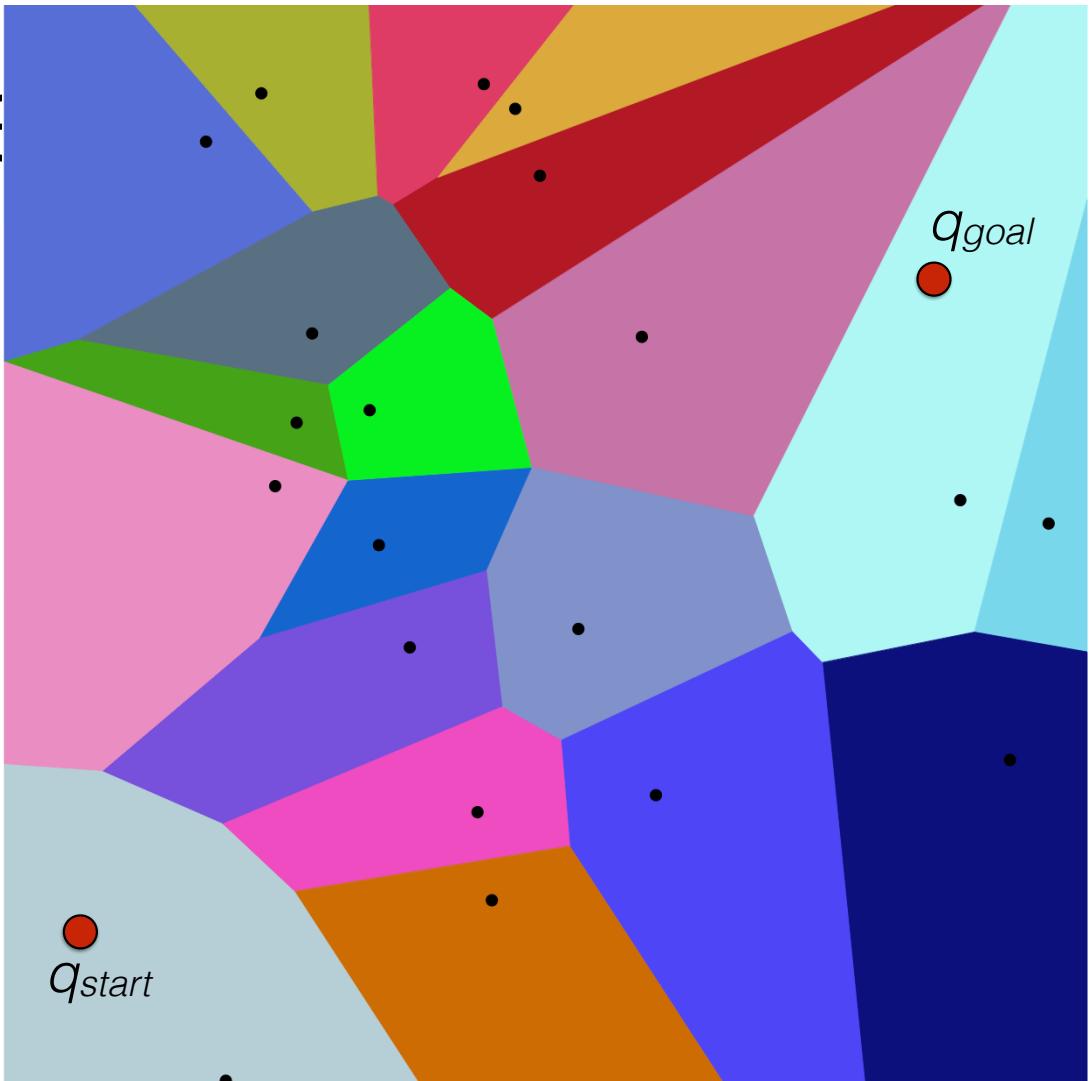


2 Approaches

Deterministic:

complete algorithms

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

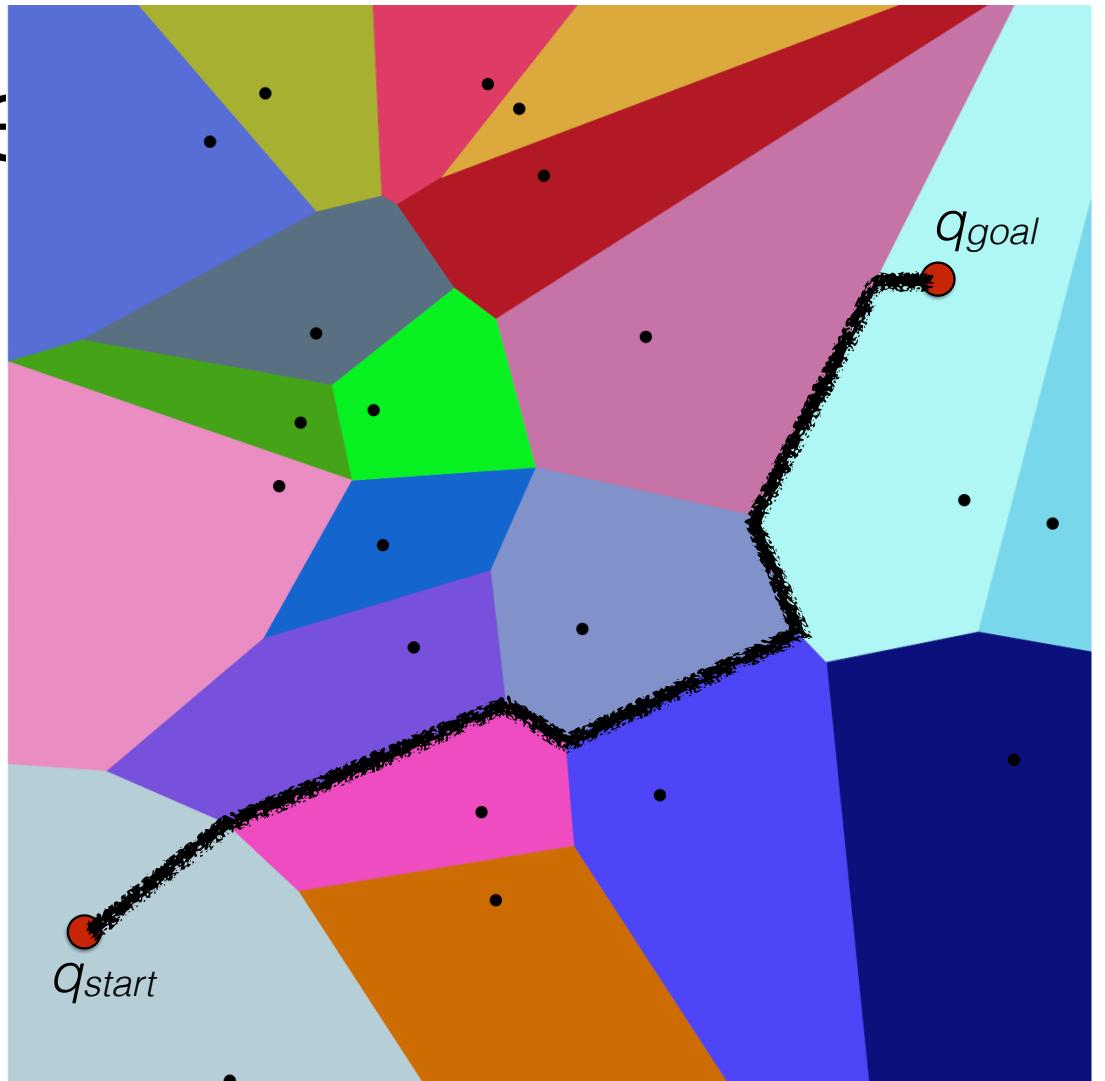


2 Approaches

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

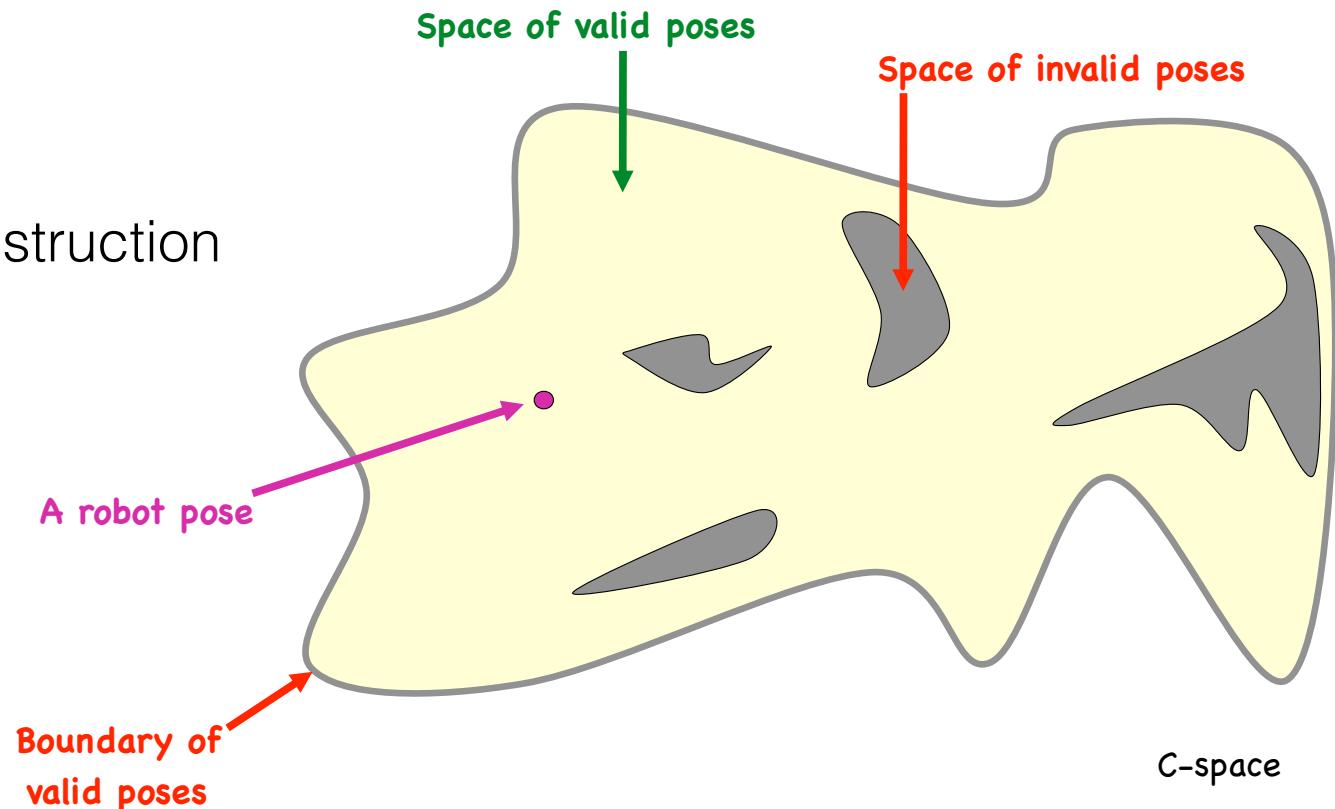
Probabilistic:

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

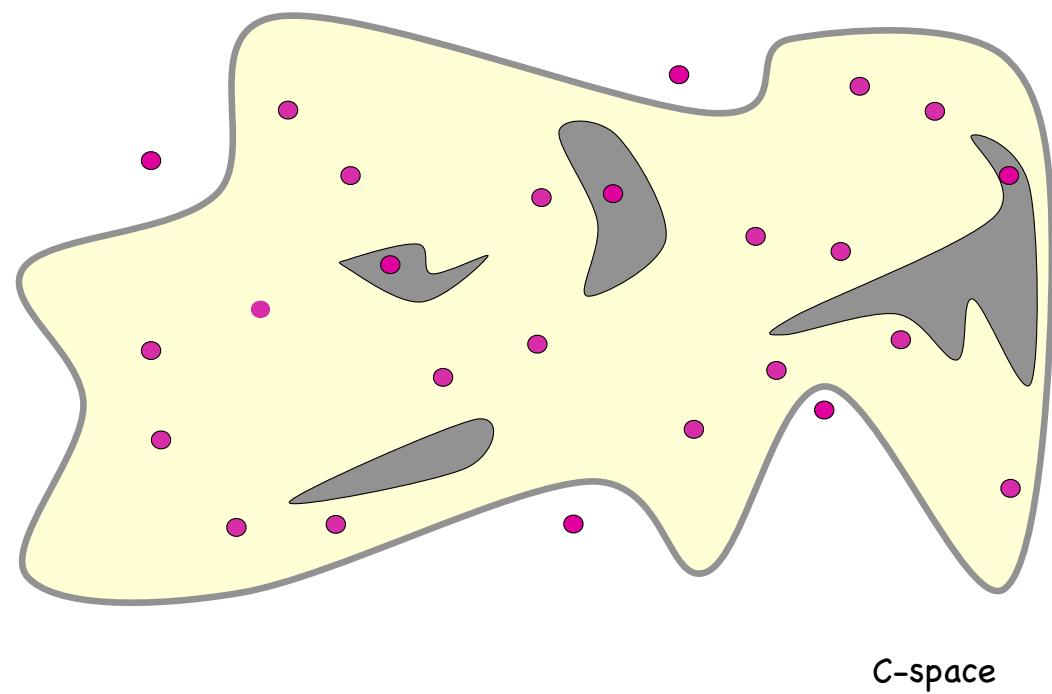
Probabilistic road maps

- Two phases
 - Roadmap construction
 - Path Query



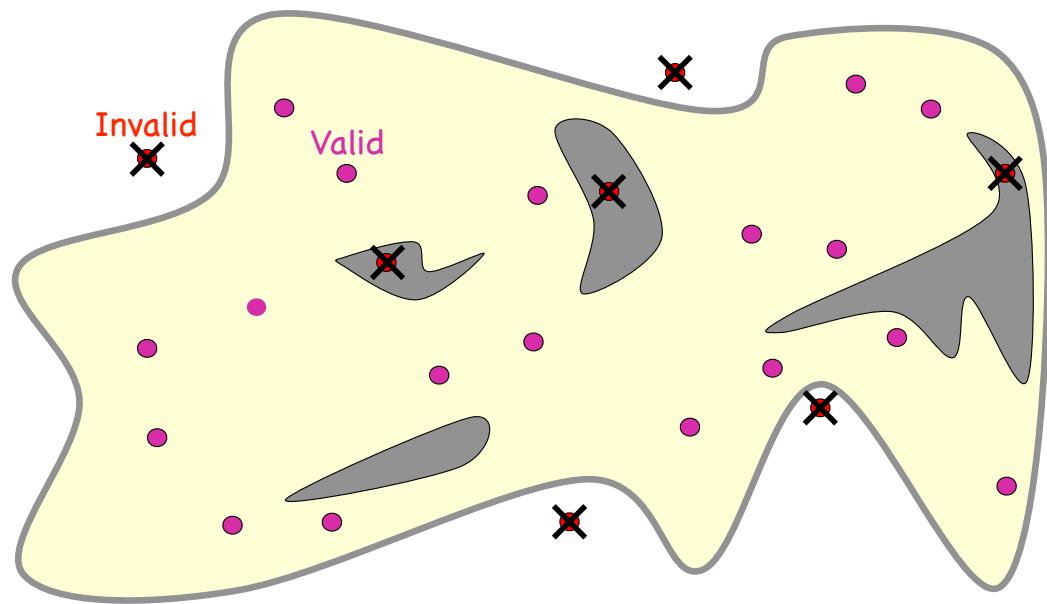
PRM: construction phase

- 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

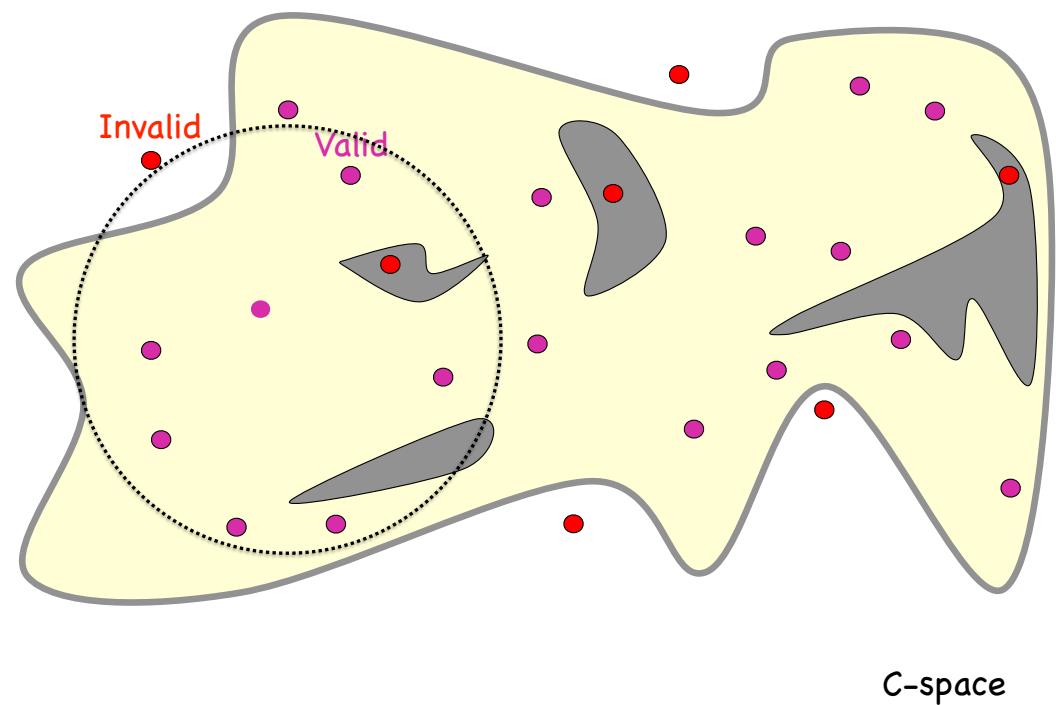


Collision detection
will be covered later

C-space

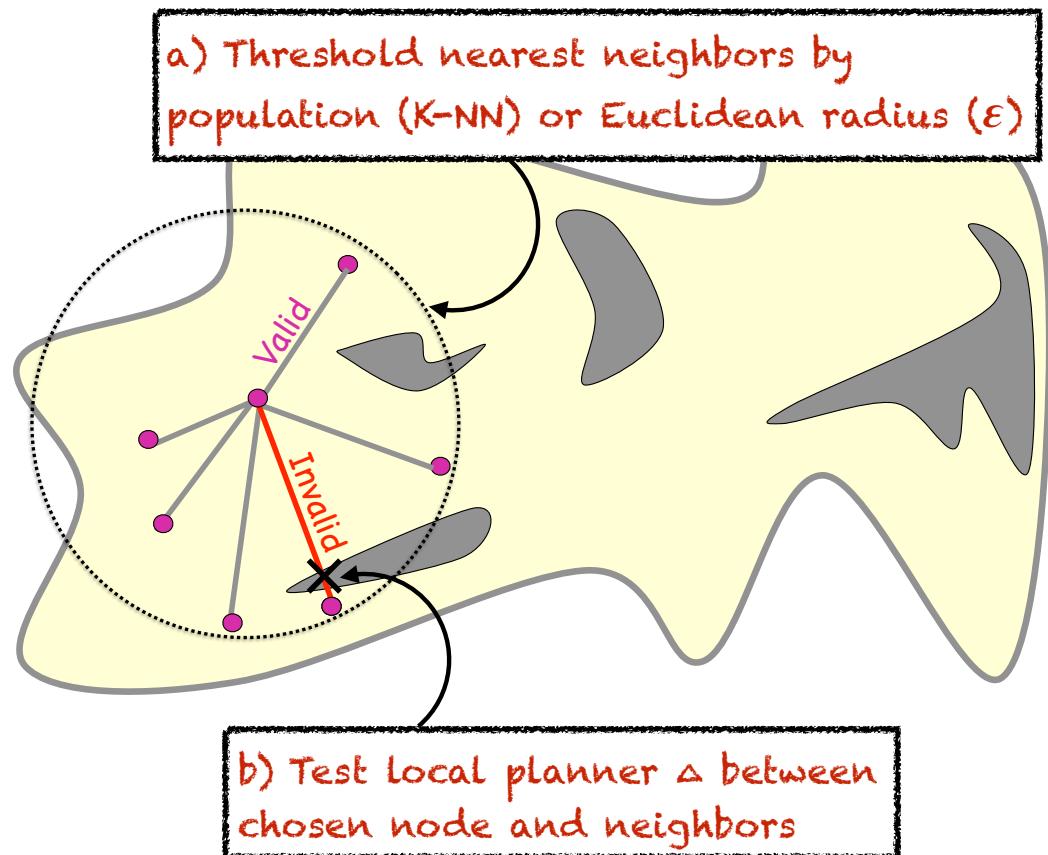
PRM: construction phase

- 1) Select N sample poses at random
- 2) Eliminate invalid poses
- 3) **Connect neighboring poses**



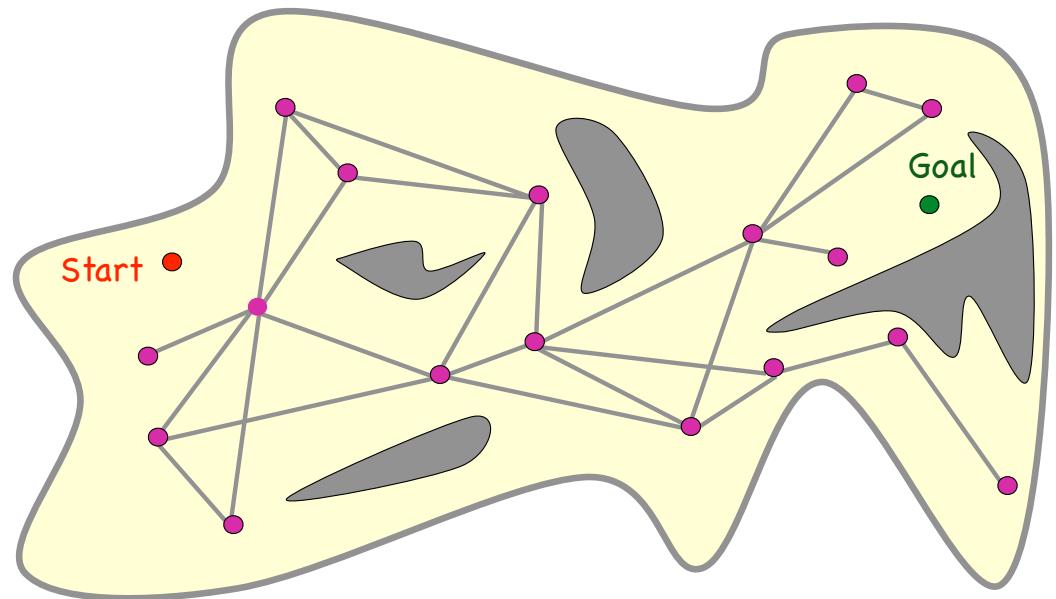
PRM: construction phase

- 1) Select N sample poses at random
- 2) Eliminate invalid poses
- 3) Connect neighboring poses



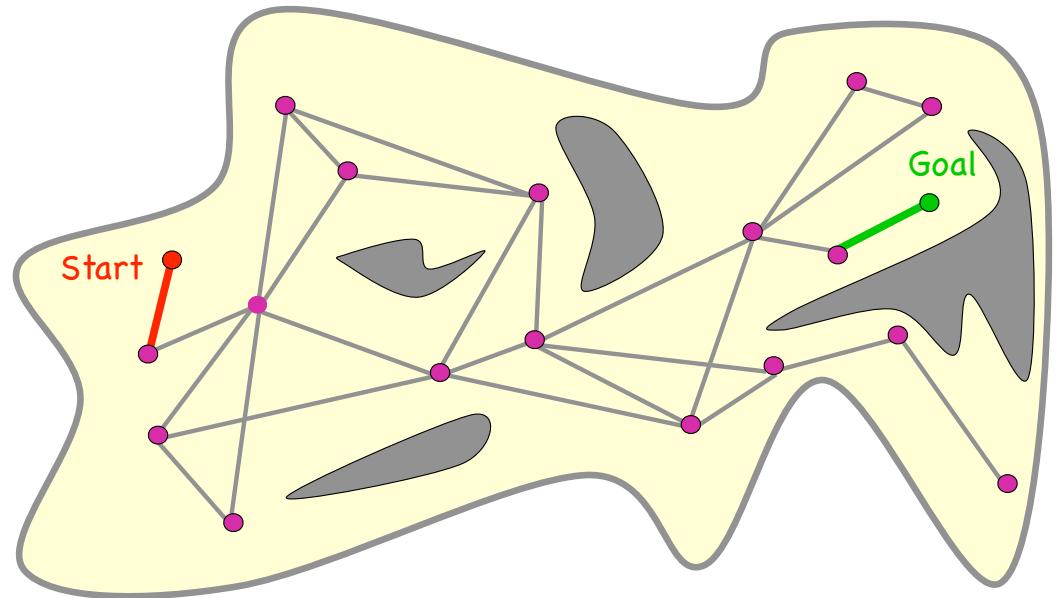
PRM: query phase

- 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



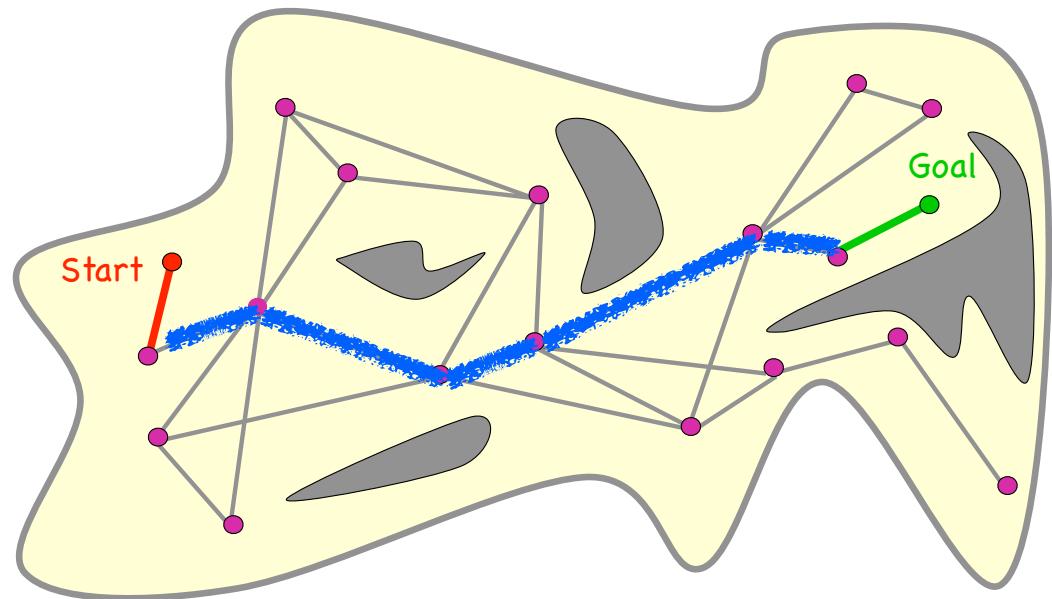
PRM: query phase

- 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

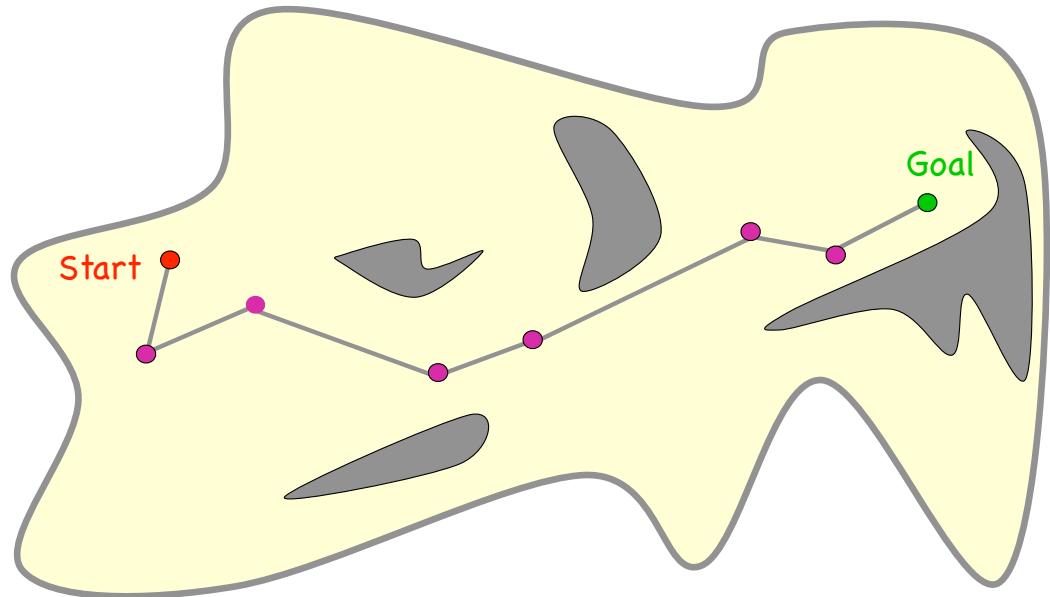
- 1) Given constructed roadmap, start pose, and goal pose
- 2) Attach goal and start to nearest roadmap entry nodes
- 3) **Search for path between roadmap entry nodes**
- 4) Return path with entry and departure edges



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)**

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        $q_{rand} \leftarrow \text{RANDOM.CONFIG}()$ ;
4       EXTEND( $T, q_{rand}$ );
5   Return  $T$ 
```

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        $q_{rand} \leftarrow \text{RANDOM.CONFIG}()$ ;
4       EXTEND( $T, q_{rand}$ );
5   Return  $T$ 
```

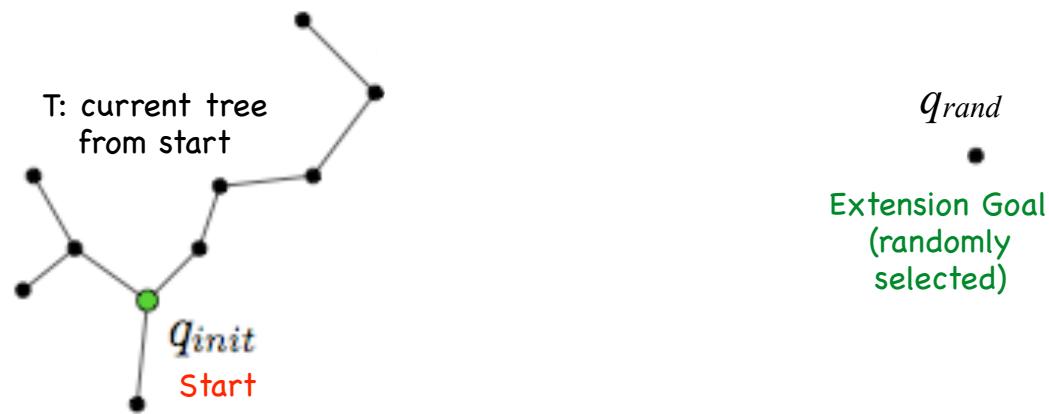


Figure 3: The EXTEND operation.

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        $q_{rand} \leftarrow RANDOM.CONFIG()$ ;
4       EXTEND( $T, q_{rand}$ );
5   Return  $T$ 
```

```
EXTEND( $T, q$ )
1    $q_{near} \leftarrow NEAREST.NEIGHBOR(q, T)$ ;
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

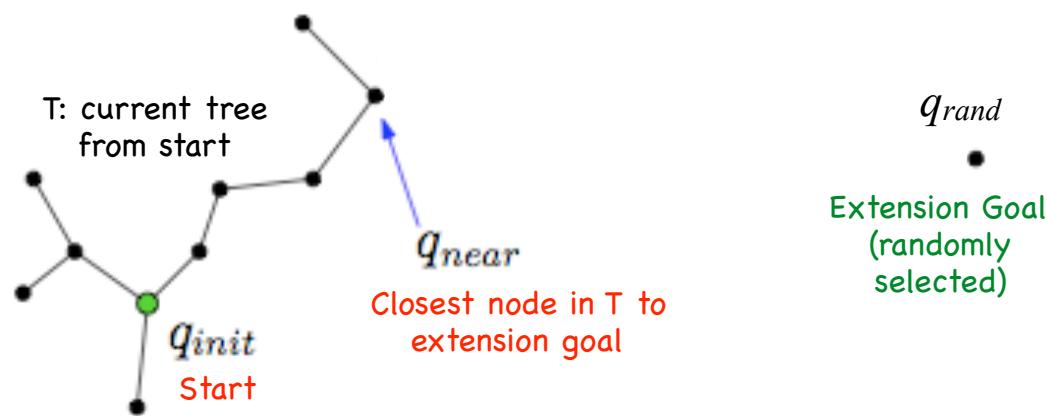


Figure 3: The EXTEND operation.

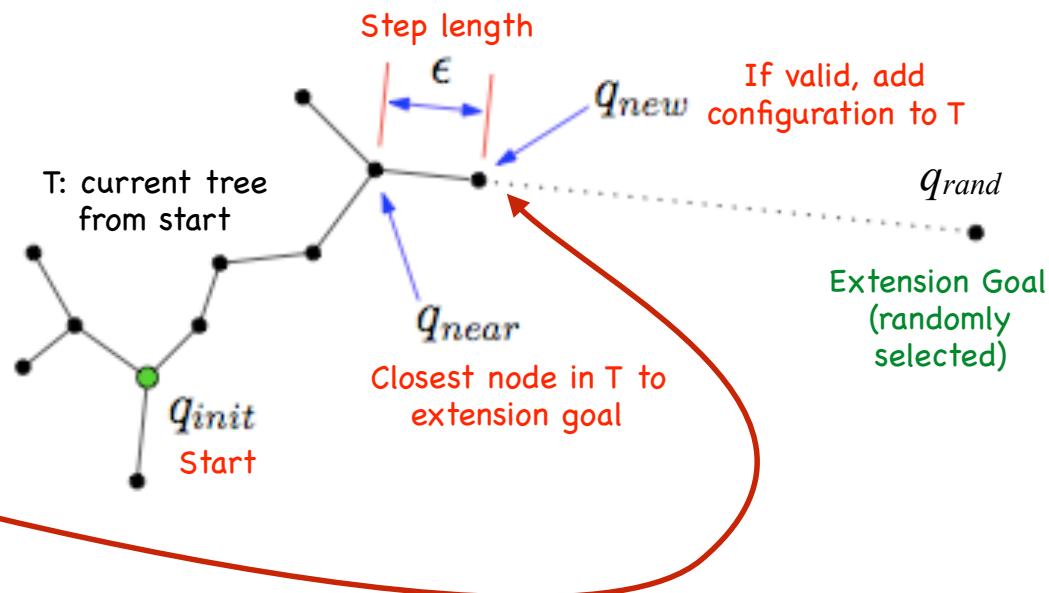
RRT Algorithm

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        $q_{rand} \leftarrow RANDOM.CONFIG()$ ;
4       EXTEND( $T, q_{rand}$ );
5   Return  $T$ 
```

```
EXTEND( $T, q$ )
1    $q_{near} \leftarrow NEAREST.NEIGHBOR(q, T)$ ;
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;
```

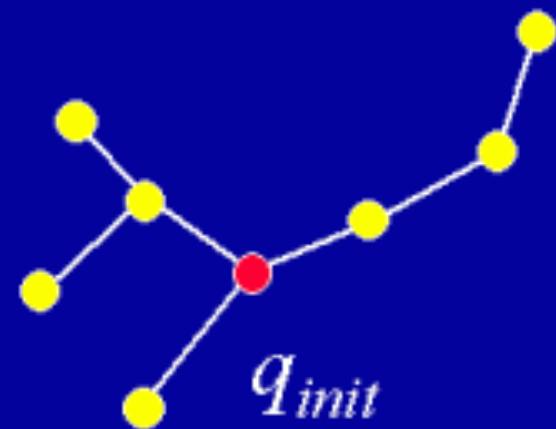
Extend graph towards a random configuration



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

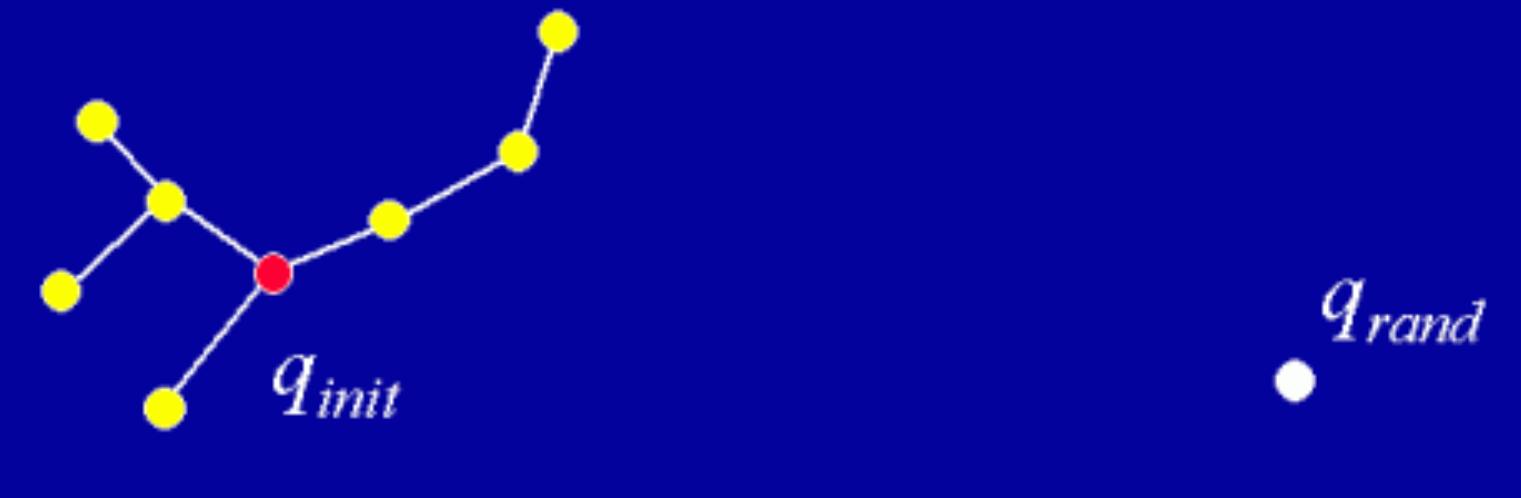
RRT Extend animation

Existing RRT is “grown” as follows...



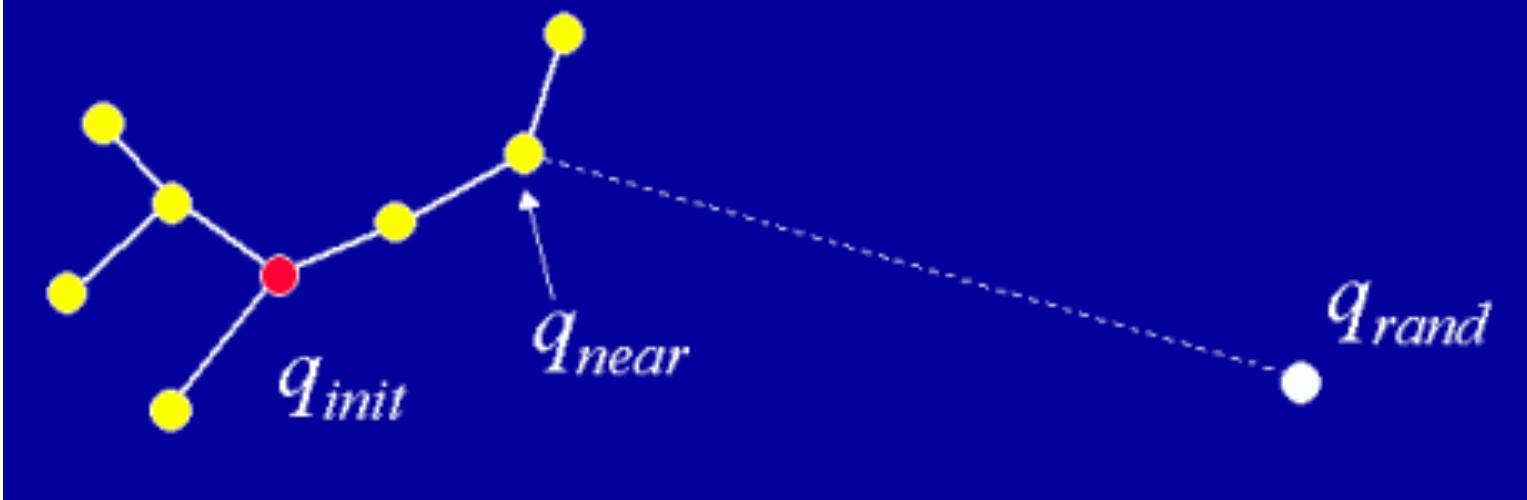
RRT Extend animation

- 1) Select a random “target” node



RRT Extend animation

2) Calculate “nearest” node in tree

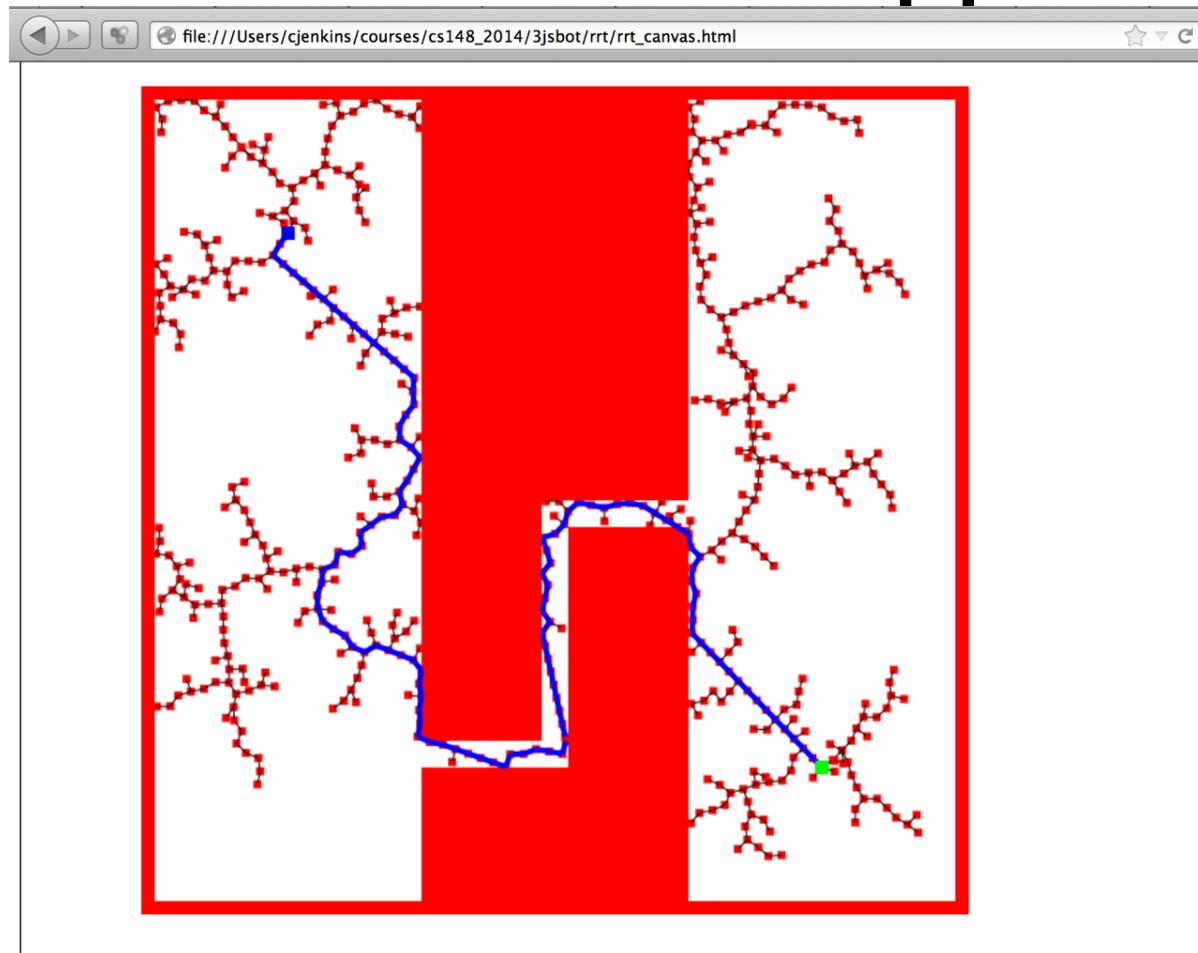


RRT Extend animation

3) Try to add new collision-free branch



Let's see what happens

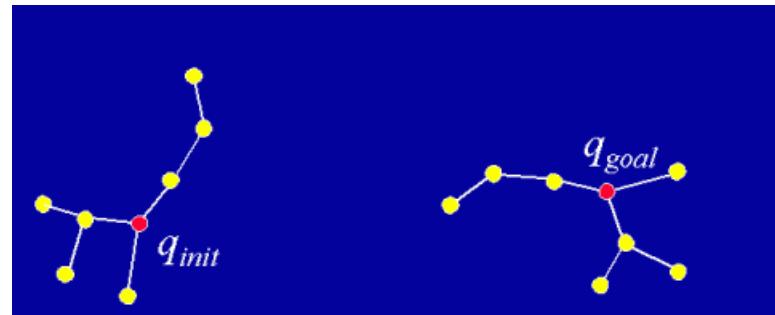


[Kuffner, LaValle 2000]

RRT Connect

- 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        $q_{rand} \leftarrow \text{RANDOM\_CONFIG}();$ 
4       if not ( $\text{EXTEND}(T_a, q_{rand}) = \text{Trapped}$ ) then
5           if ( $\text{CONNECT}(T_b, q_{new}) = \text{Reached}$ ) then
6               Return PATH( $T_a, T_b$ );
7           SWAP( $T_a, T_b$ );
8   Return Failure
```



RRT Connect

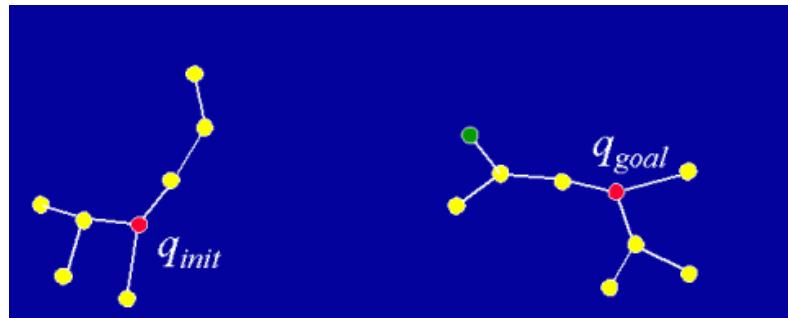
- 0) Use 2 trees (A and B) rooted at start and goal configurations

```
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    $q_{rand} \leftarrow \text{RANDOM\_CONFIG}();$ 
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
```

EXTEND(\mathcal{T}, q)

```
1  $q_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(q, \mathcal{T});$ 
2 if NEW_CONFIG( $q, q_{near}, q_{new}$ ) then
3    $\mathcal{T}.add\_vertex(q_{new});$ 
4    $\mathcal{T}.add\_edge(q_{near}, q_{new});$ 
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

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    $q_{rand} \leftarrow RANDOM\_CONFIG();$ 
4   if not (EXTEND( $T_a, q_{rand}$ ) = Trapped) then
5     if (CONNECT( $T_b, q_{near}$ ) = Reached) then
6       Return PATH( $T_a, T_b$ );
7     SWAP( $T_a, T_b$ );
8   Return Failure

```

EXTEND(T, q)

```

1  $q_{near} \leftarrow NEAREST\_NEIGHBOR(q, T);$ 
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    $S \leftarrow EXTEND(T, q);$ 
3 until not ( $S = Advanced$ )
4 Return S;

```

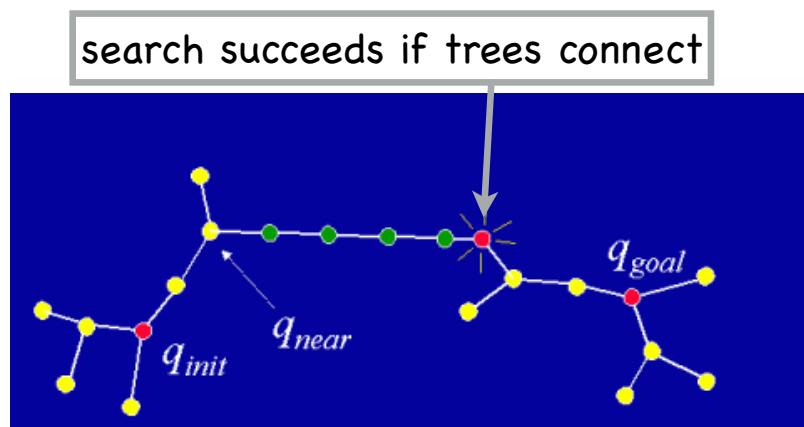
2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor



RRT Connect

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    $q_{rand} \leftarrow RANDOM\_CONFIG();$ 
4   if not (EXTEND( $T_a, q_{rand}$ ) = Trapped) then
5     if (CONNECT( $T_b, q_{near}$ ) = Reached) then
6       Return PATH( $T_a, T_b$ );
7     SWAP( $T_a, T_b$ );
8   Return Failure
```



EXTEND(T, q)

```
1  $q_{near} \leftarrow NEAREST\_NEIGHBOR(q, T);$ 
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    $S \leftarrow EXTEND(T, q);$ 
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

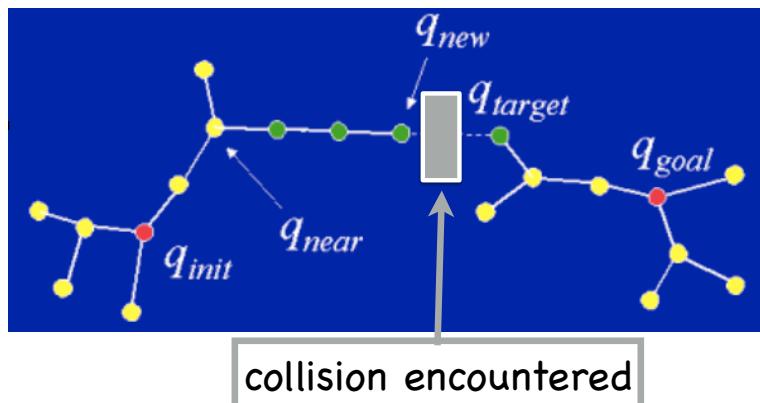
- 0) Use 2 trees (A and B) rooted at start and goal configurations

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    $q_{rand} \leftarrow RANDOM\_CONFIG();$ 
4   if not (EXTEND( $T_a, q_{rand}$ ) = Trapped) then
5     if (CONNECT( $T_b, q_{near}$ ) = Reached) then
6       Return PATH( $T_a, T_b$ );
7     SWAP( $T_a, T_b$ );
8   Return Failure
```

- 3) reverse roles for trees A and B and repeat



EXTEND(\mathcal{T}, q)

```

1  $q_{near} \leftarrow NEAREST\_NEIGHBOR(q, \mathcal{T});$ 
2 if NEW_CONFIG( $q, q_{near}, q_{new}$ ) then
3    $\mathcal{T}.add\_vertex(q_{new});$ 
4    $\mathcal{T}.add\_edge(q_{near}, q_{new});$ 
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

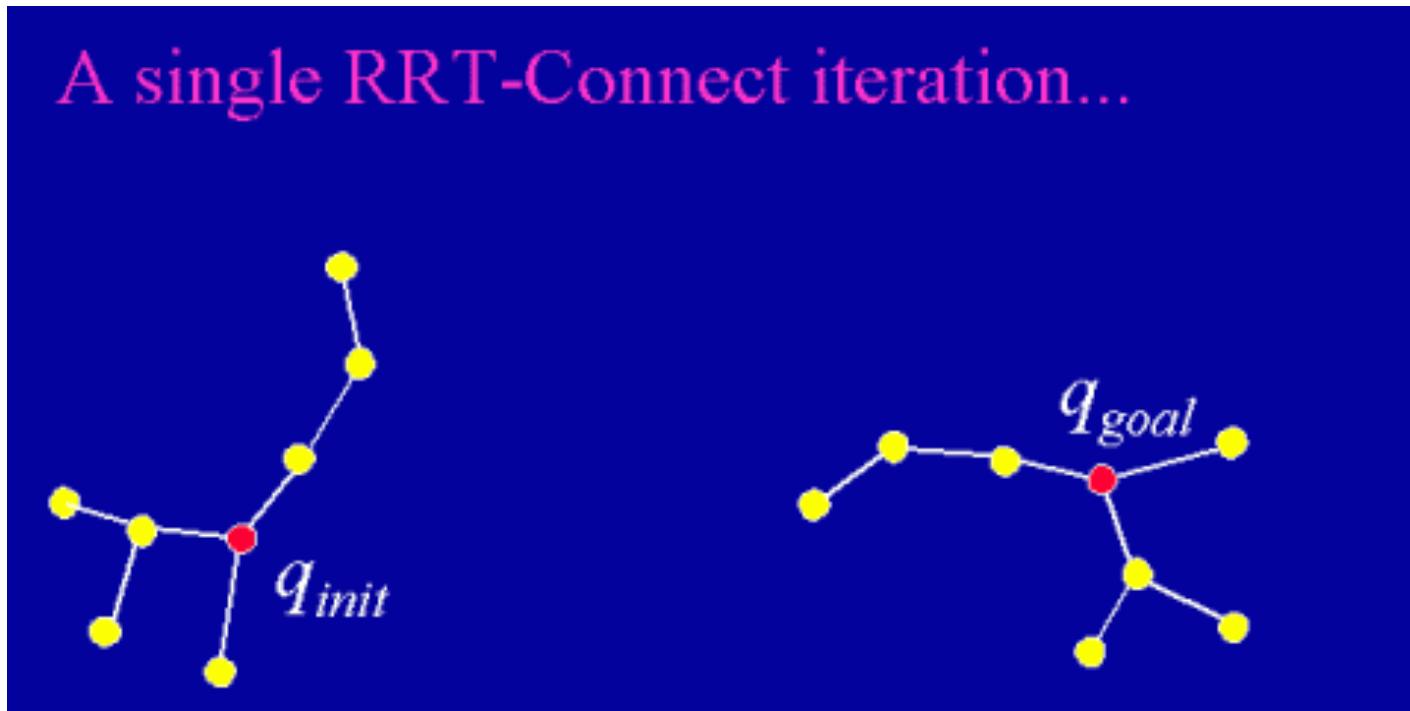
CONNECT(\mathcal{T}, q)

```

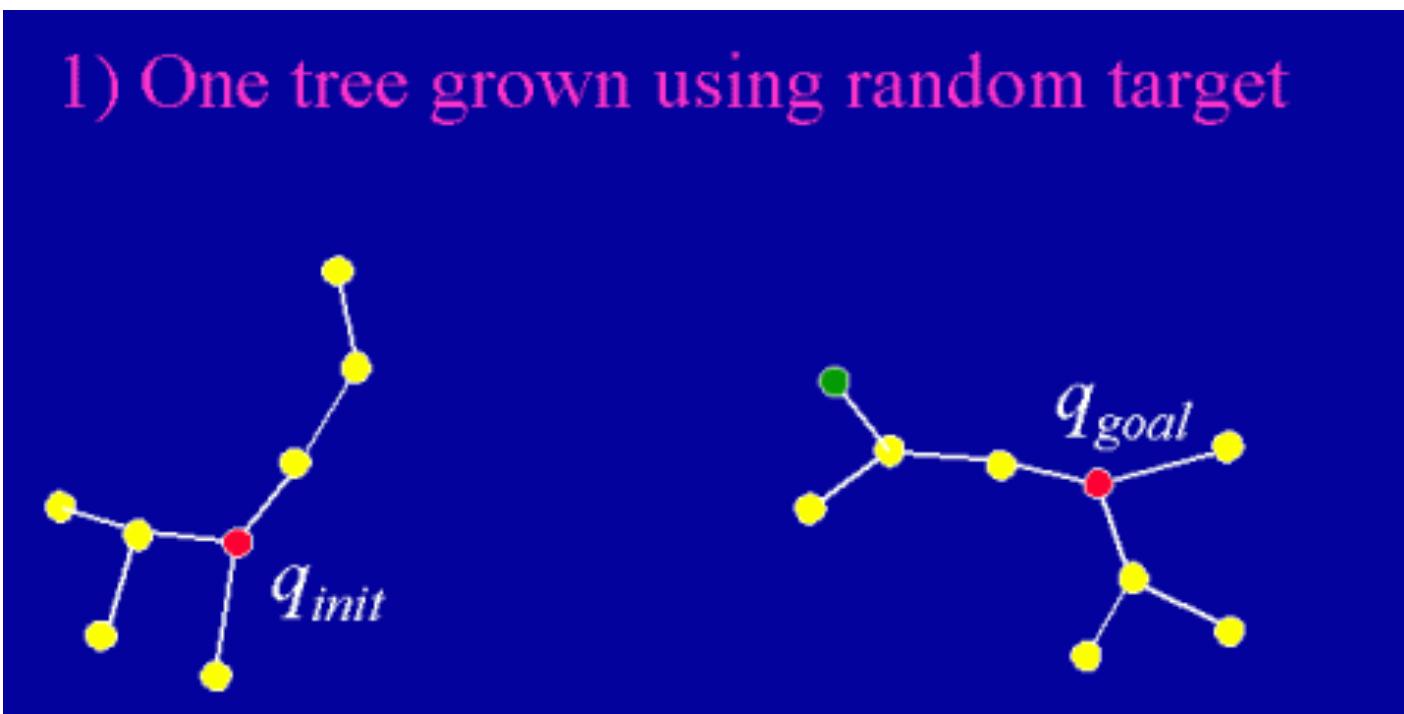
1 repeat
2    $S \leftarrow EXTEND(\mathcal{T}, q);$ 
3 until not ( $S = Advanced$ )
4 Return S;
```

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

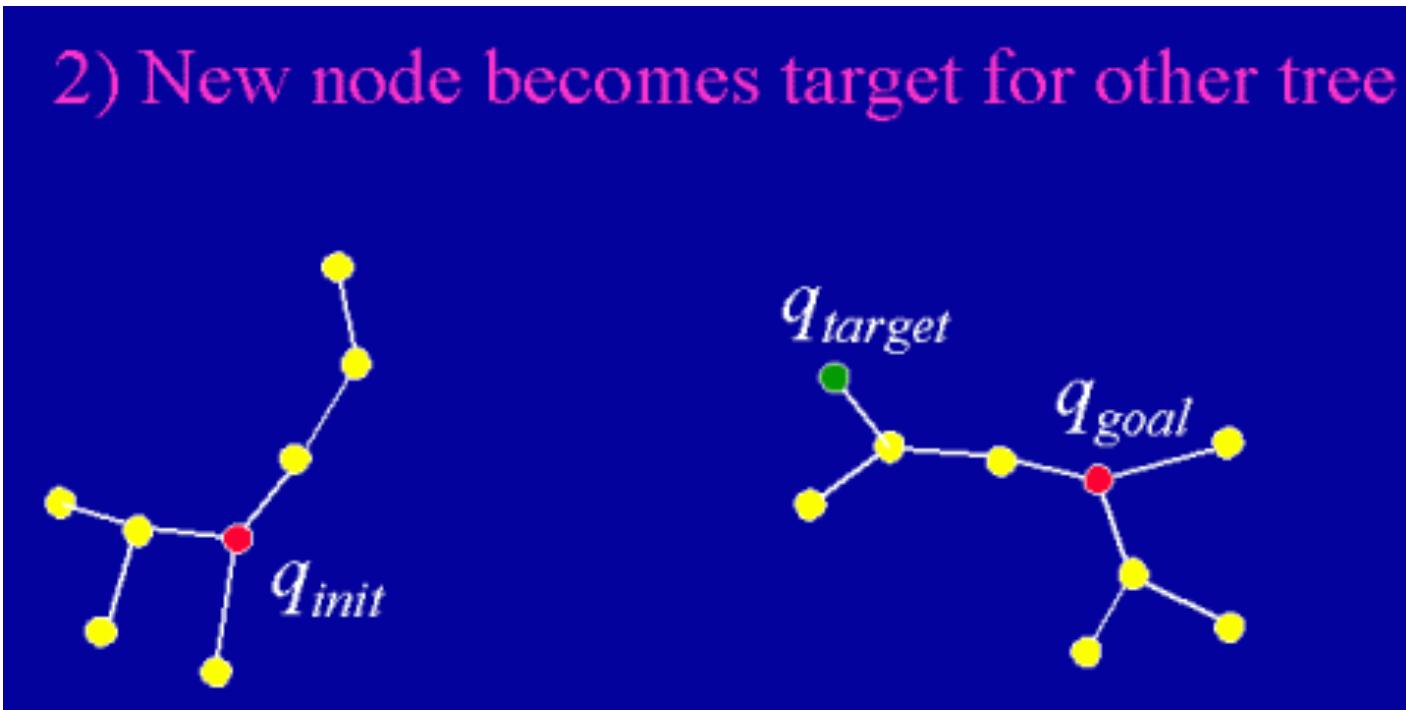
RRT-Connect animation



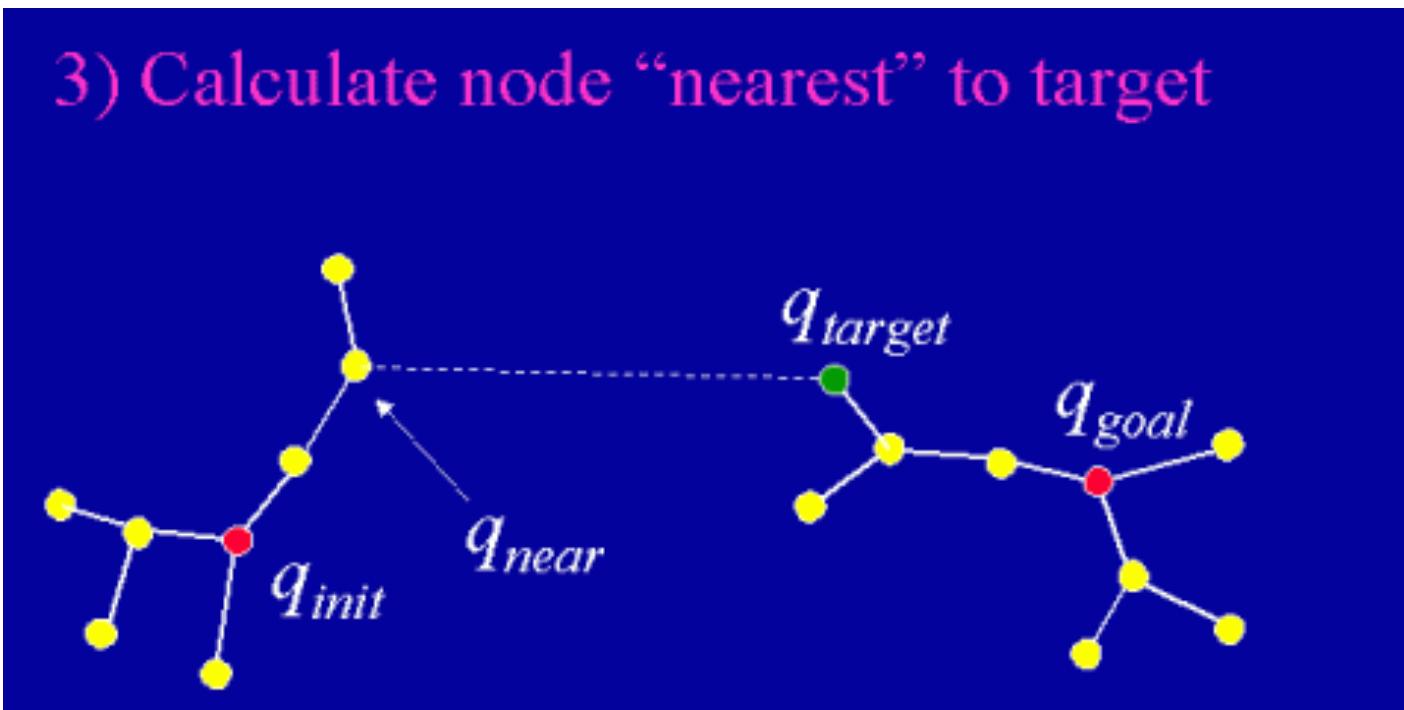
RRT-Connect animation



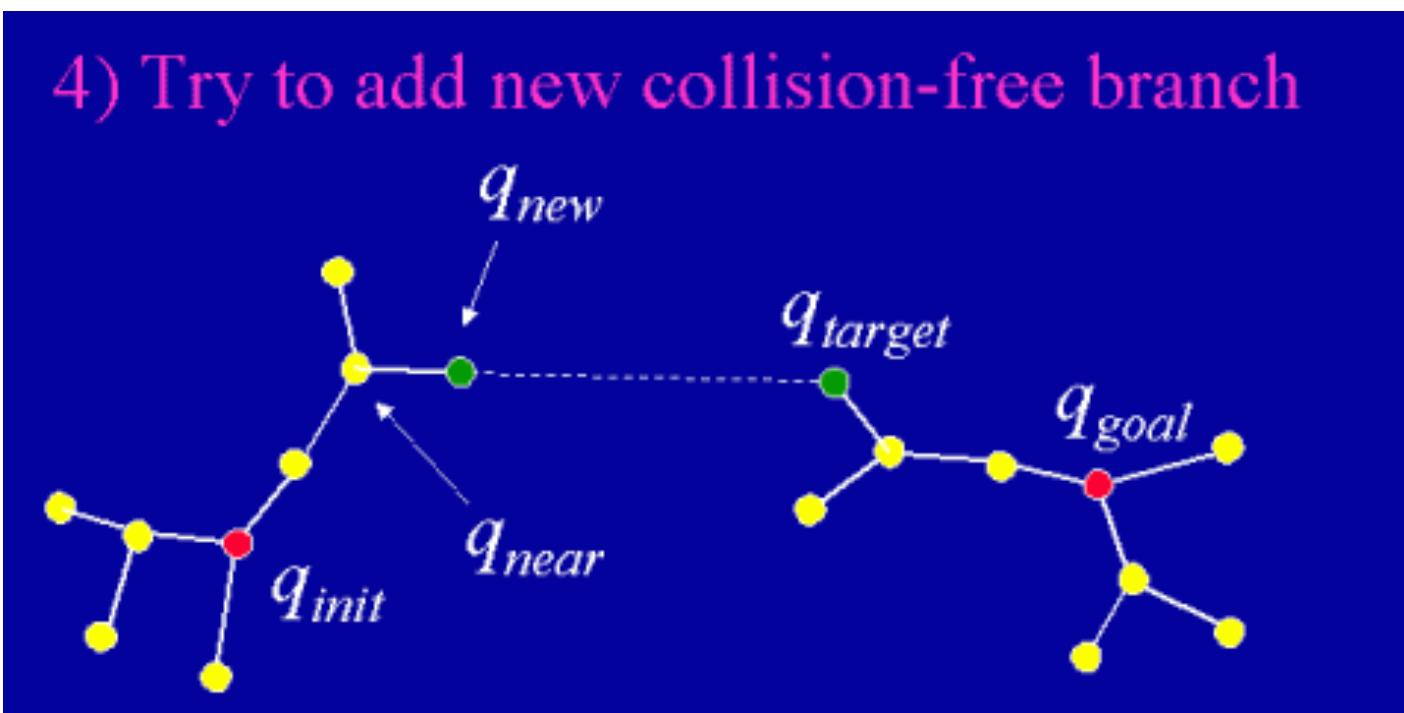
RRT-Connect animation



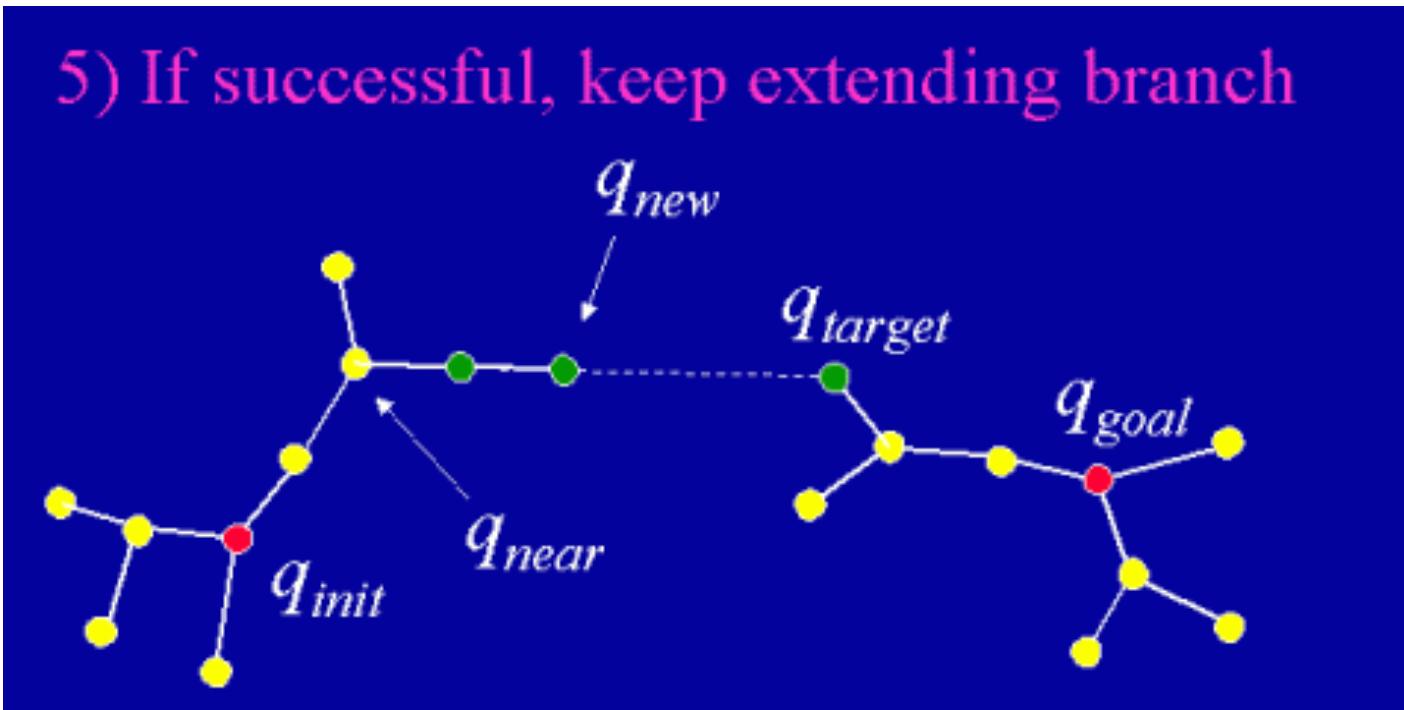
RRT-Connect animation



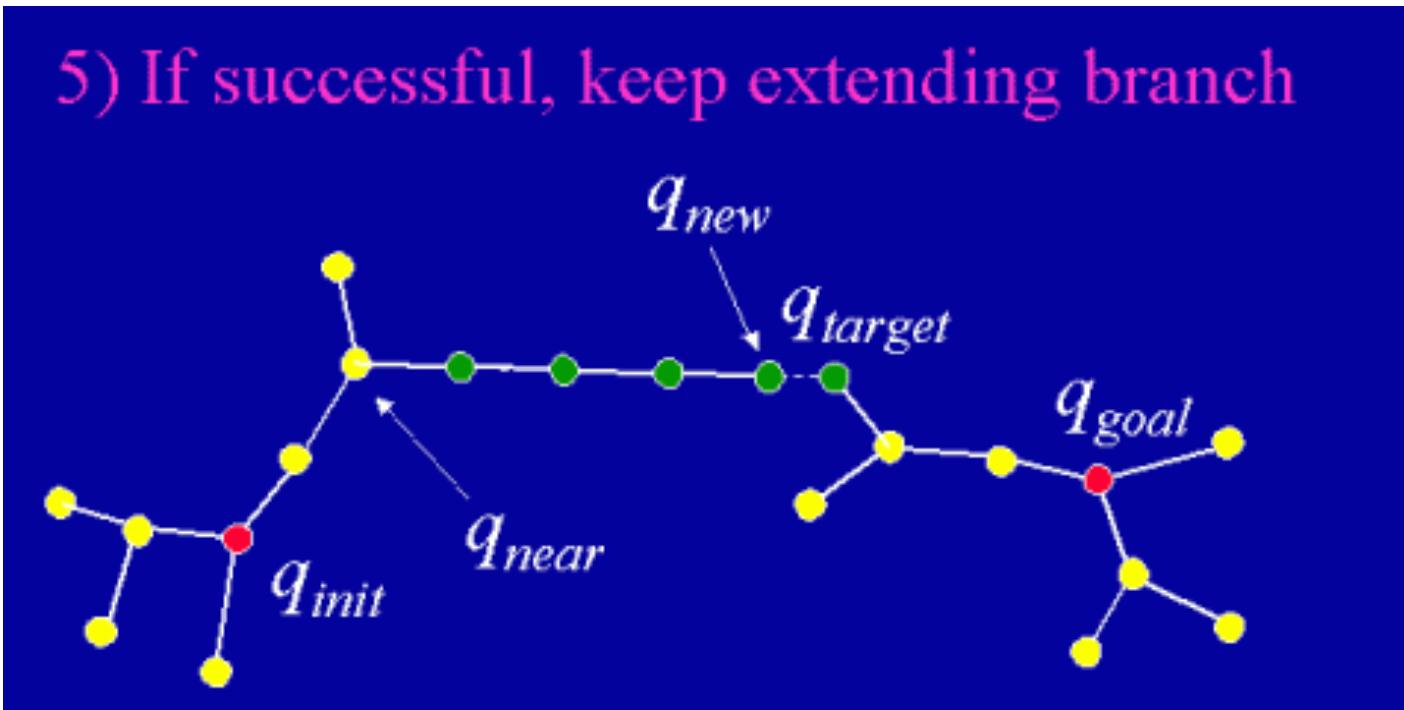
RRT-Connect animation



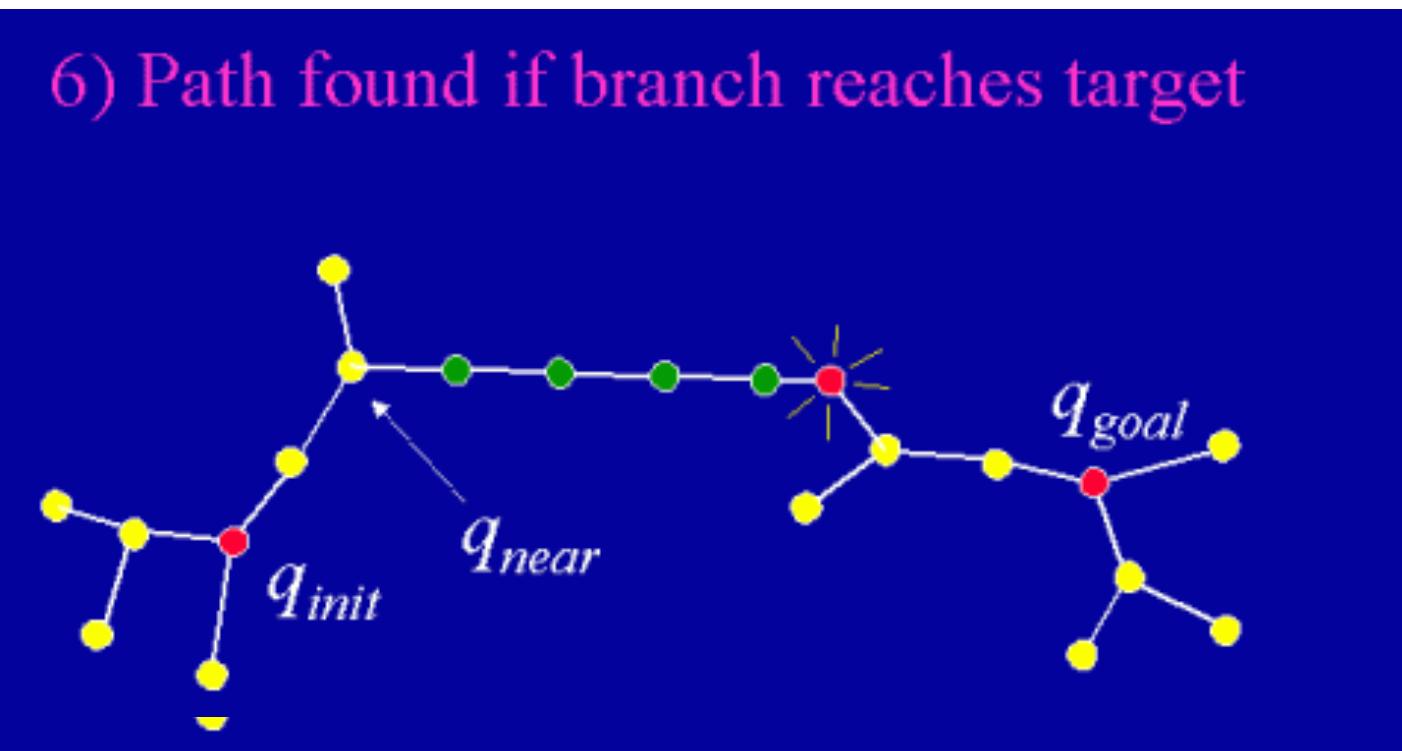
RRT-Connect animation



RRT-Connect animation

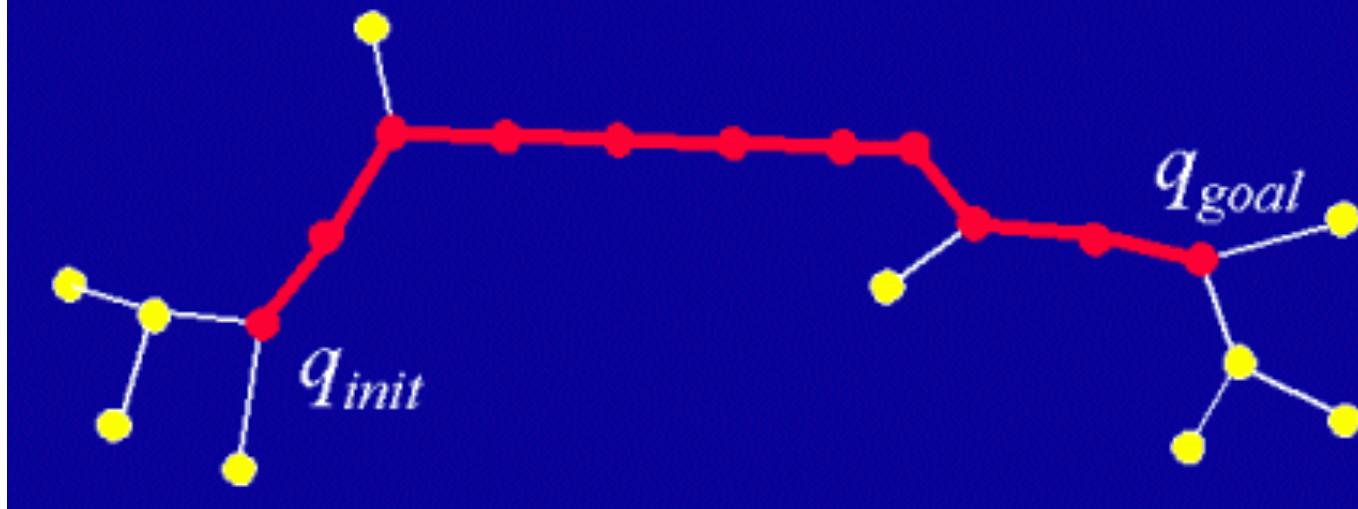


RRT-Connect animation

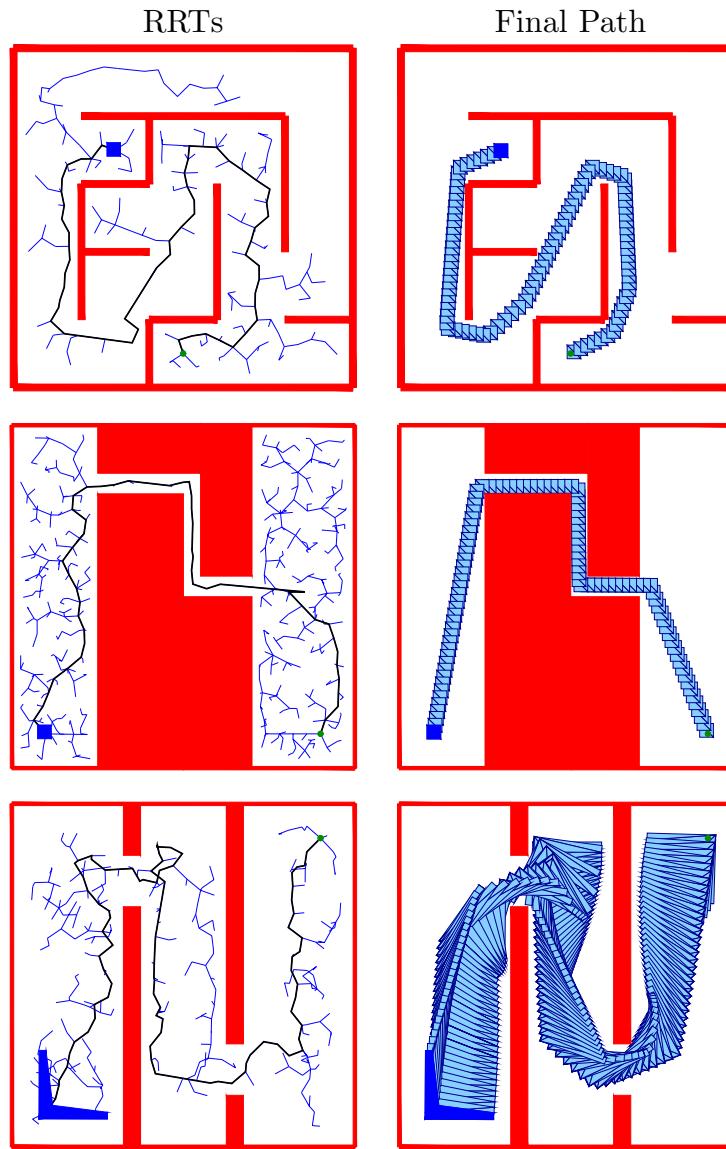


RRT-Connect animation

7) Return path connecting start and goal



Examples of RRT-Connect

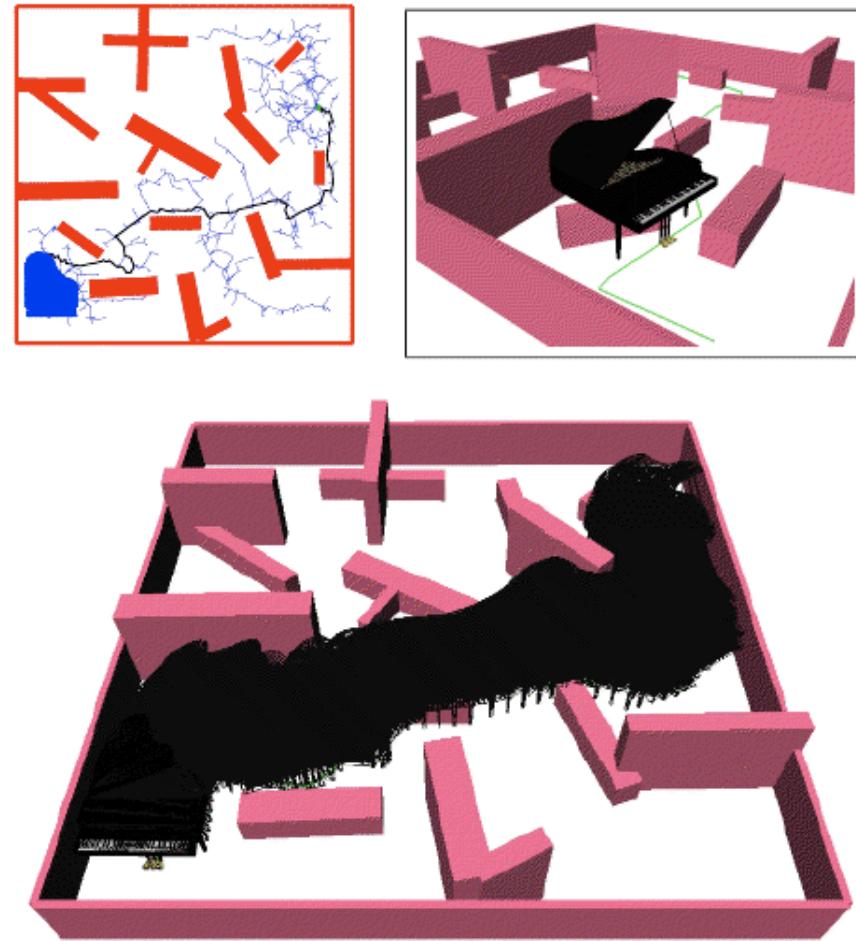


2 DOF maze

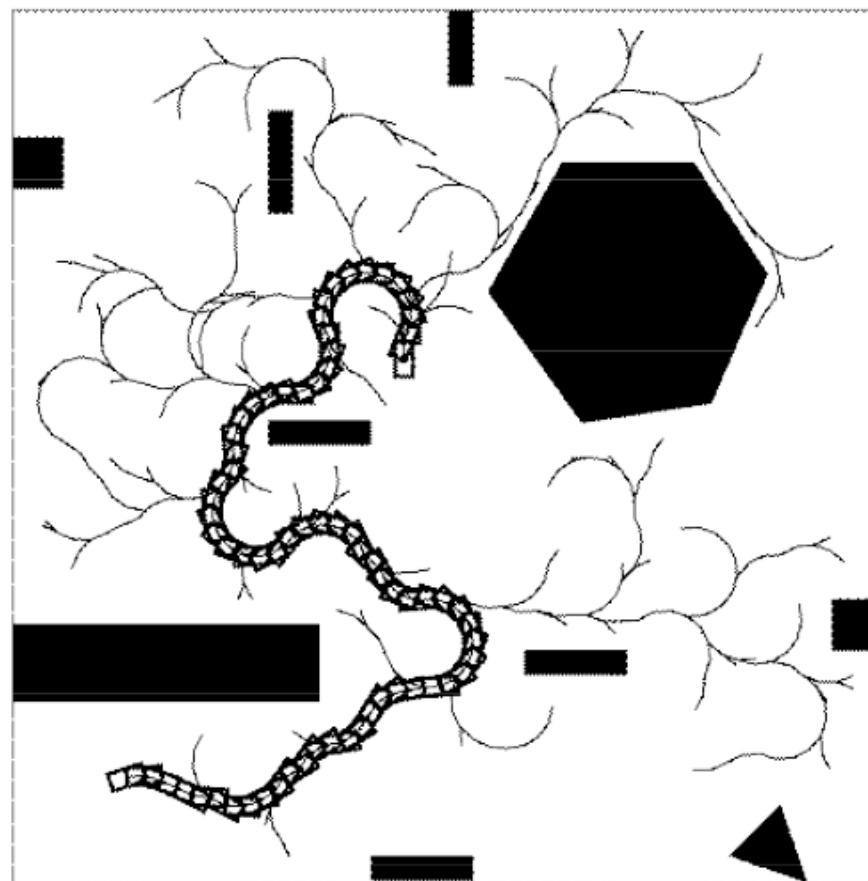
2 DOF single passway

3 DOF single passway
(with non-point geometry)

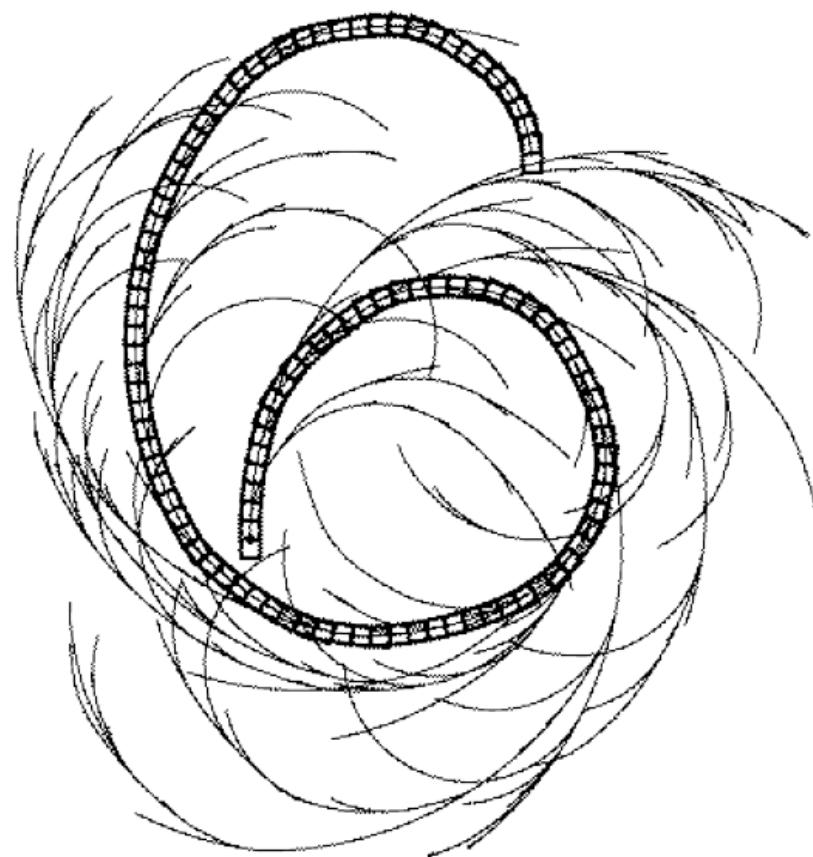
Piano Mover's Problem



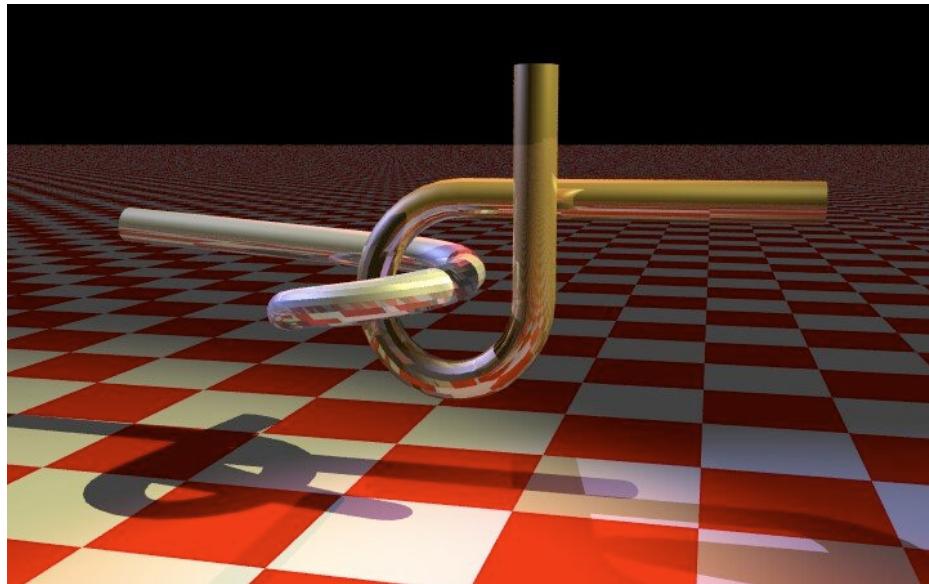
A Car-Like Robot



A Right-Turn Only Car

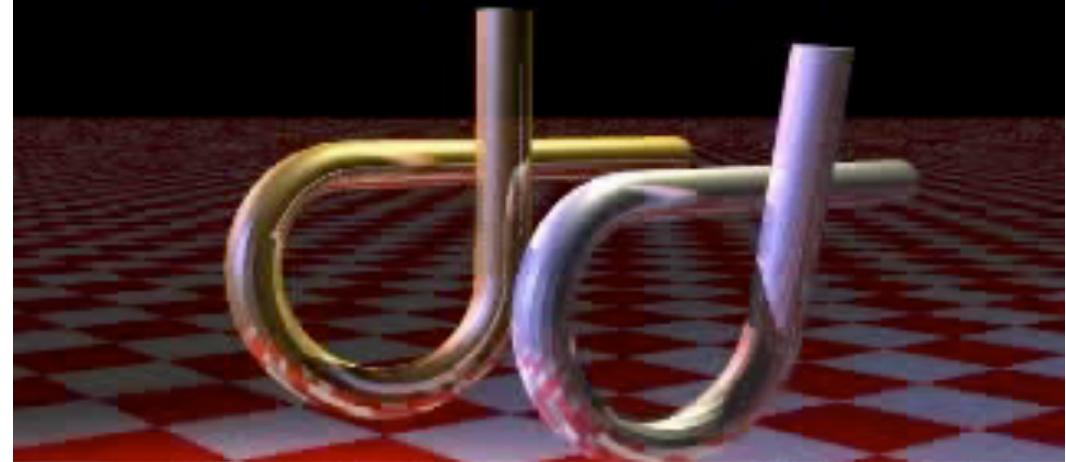


Alpha Puzzle



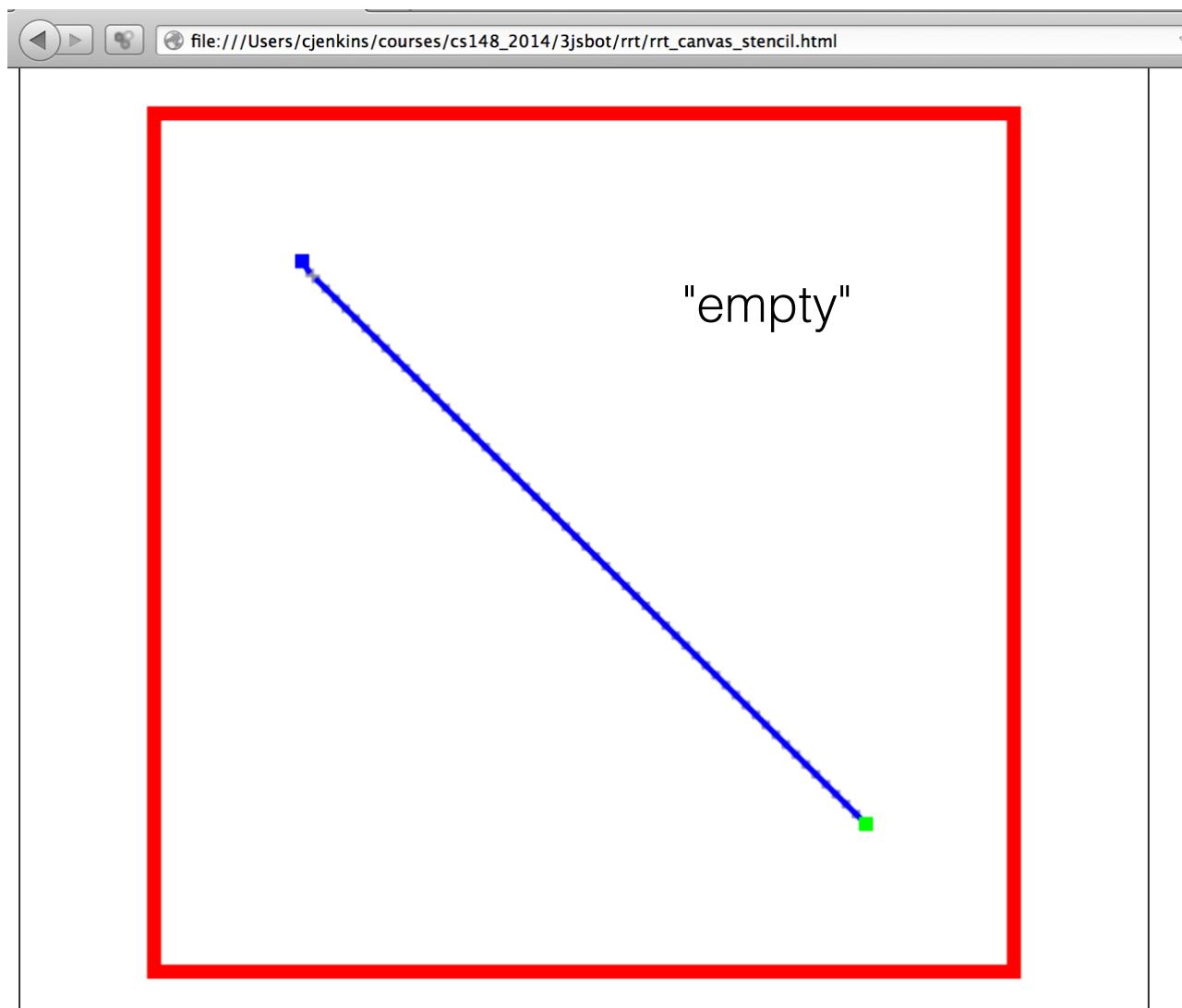
Alpha Puzzle 1.0 Solution

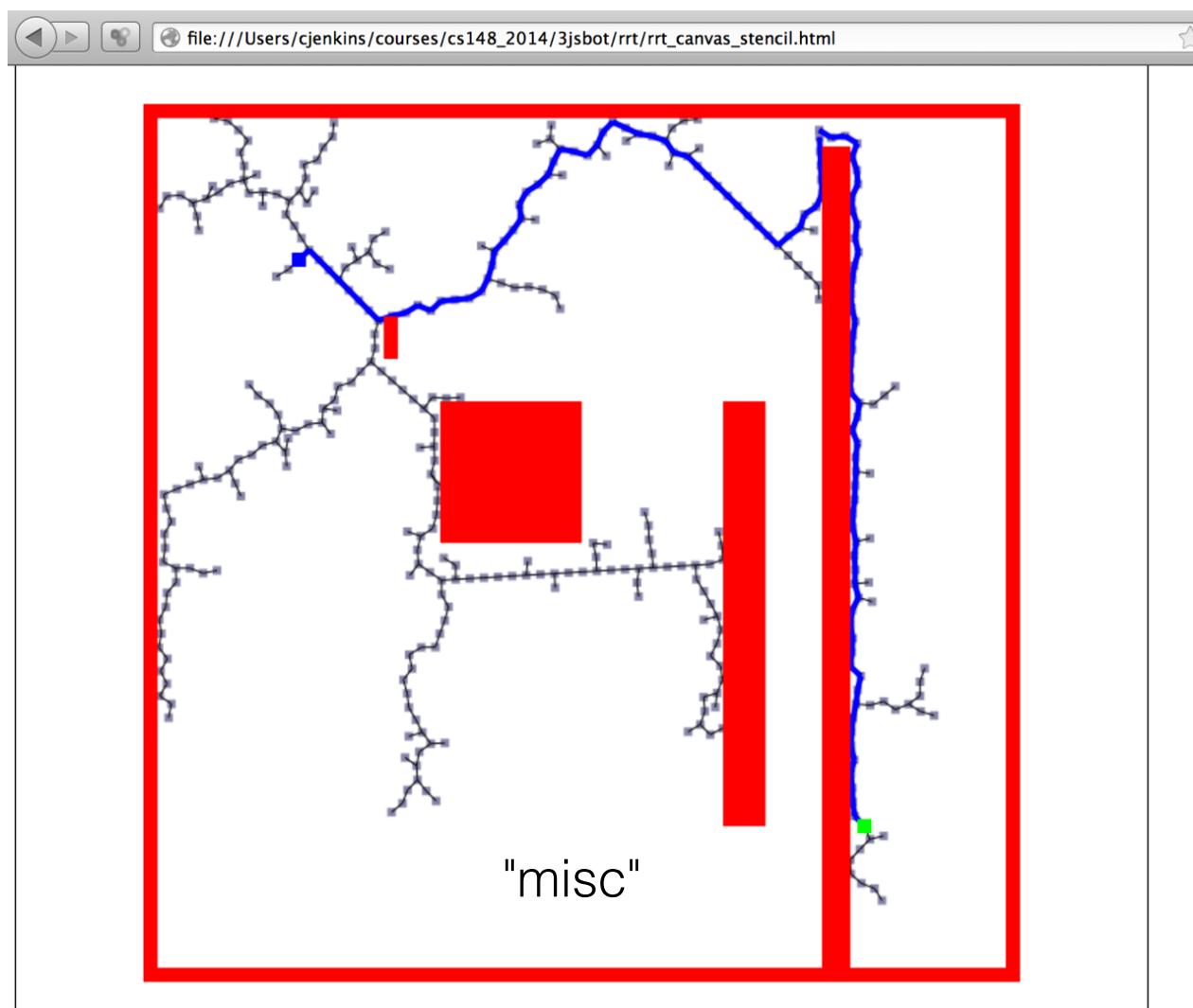
James Kuffner, Feb. 2001

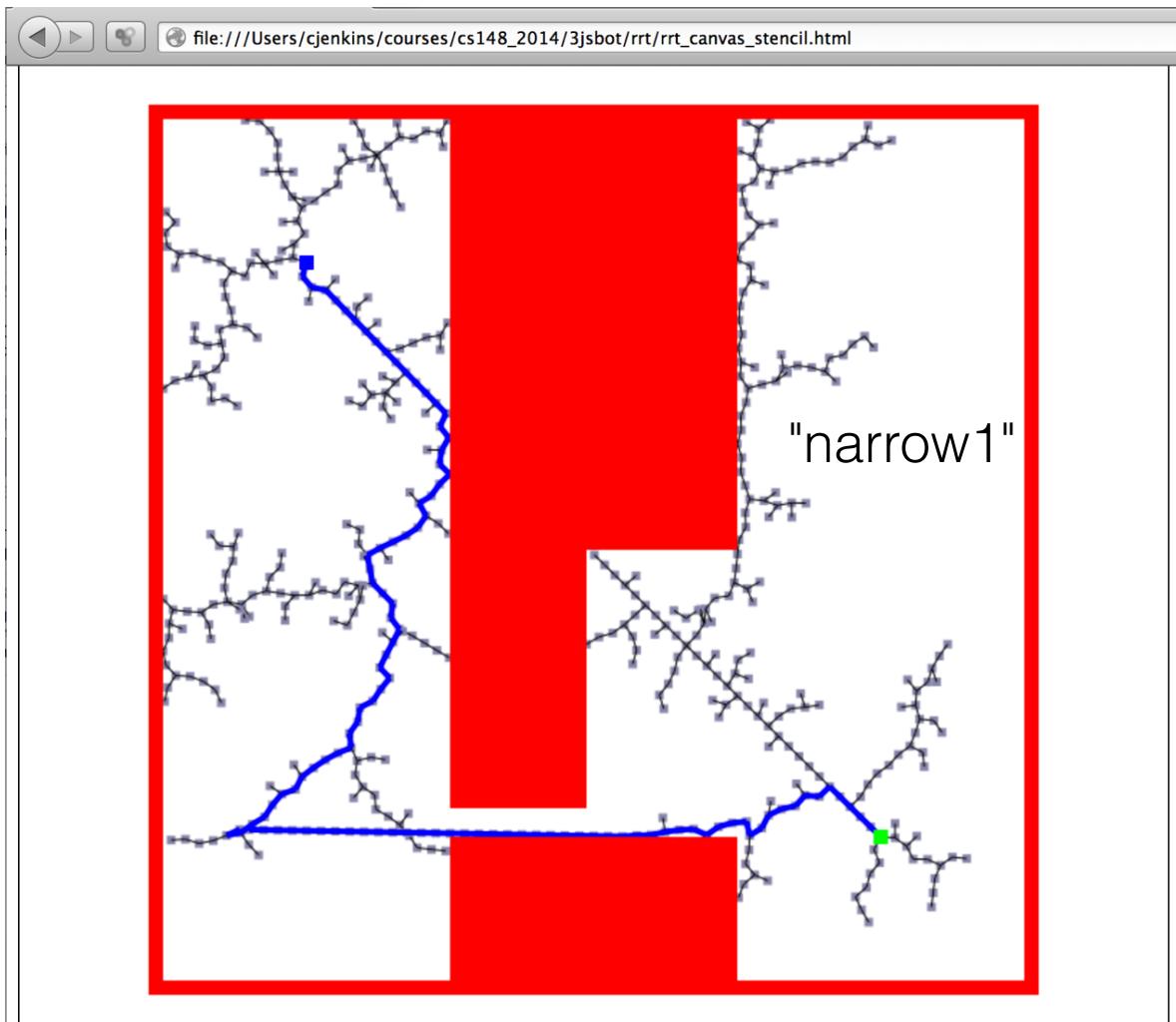


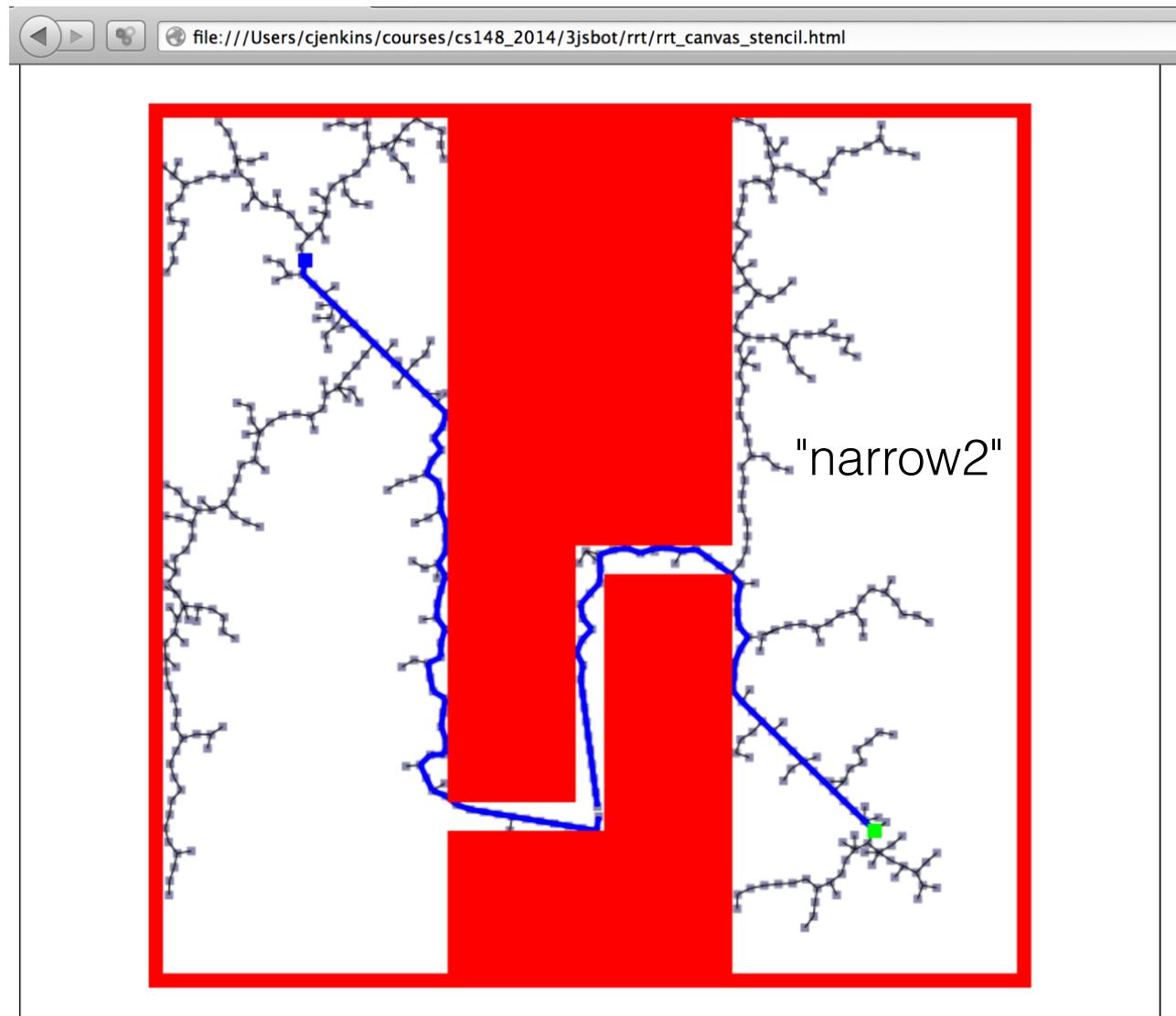
model by DSMFT group, Texas A&M Univ.
original model by Boris Yamrom, GE

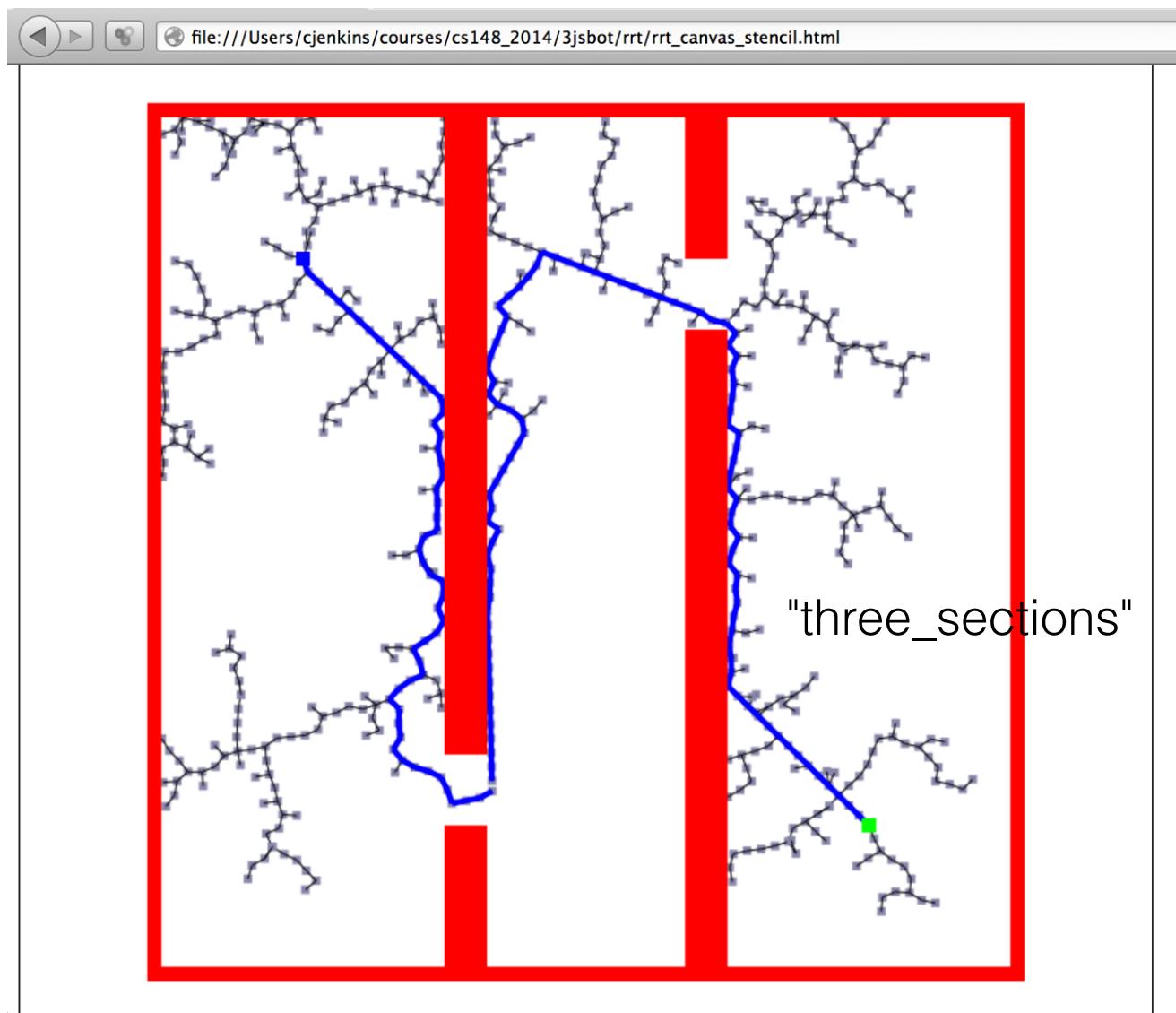
Canvas Stencil Examples



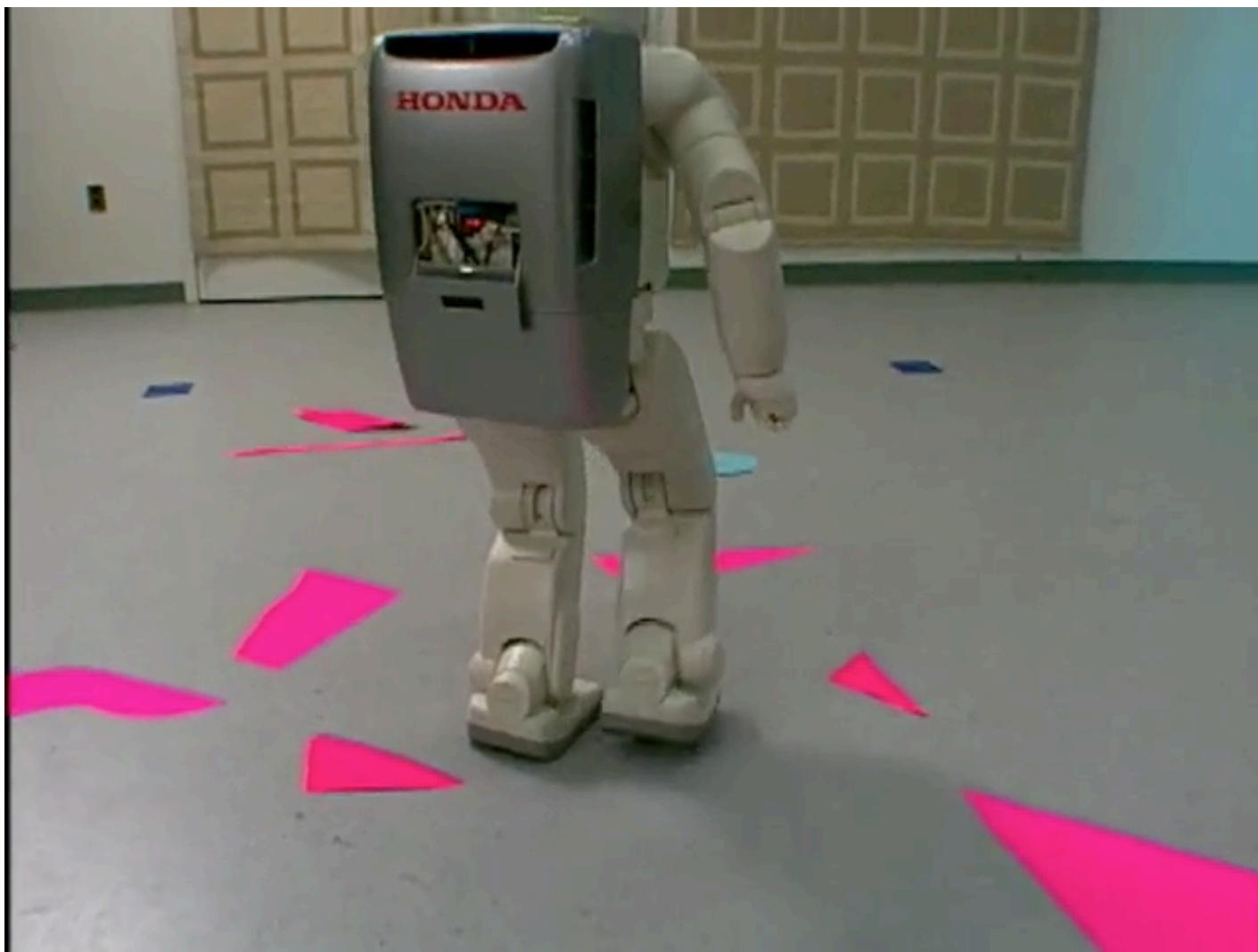








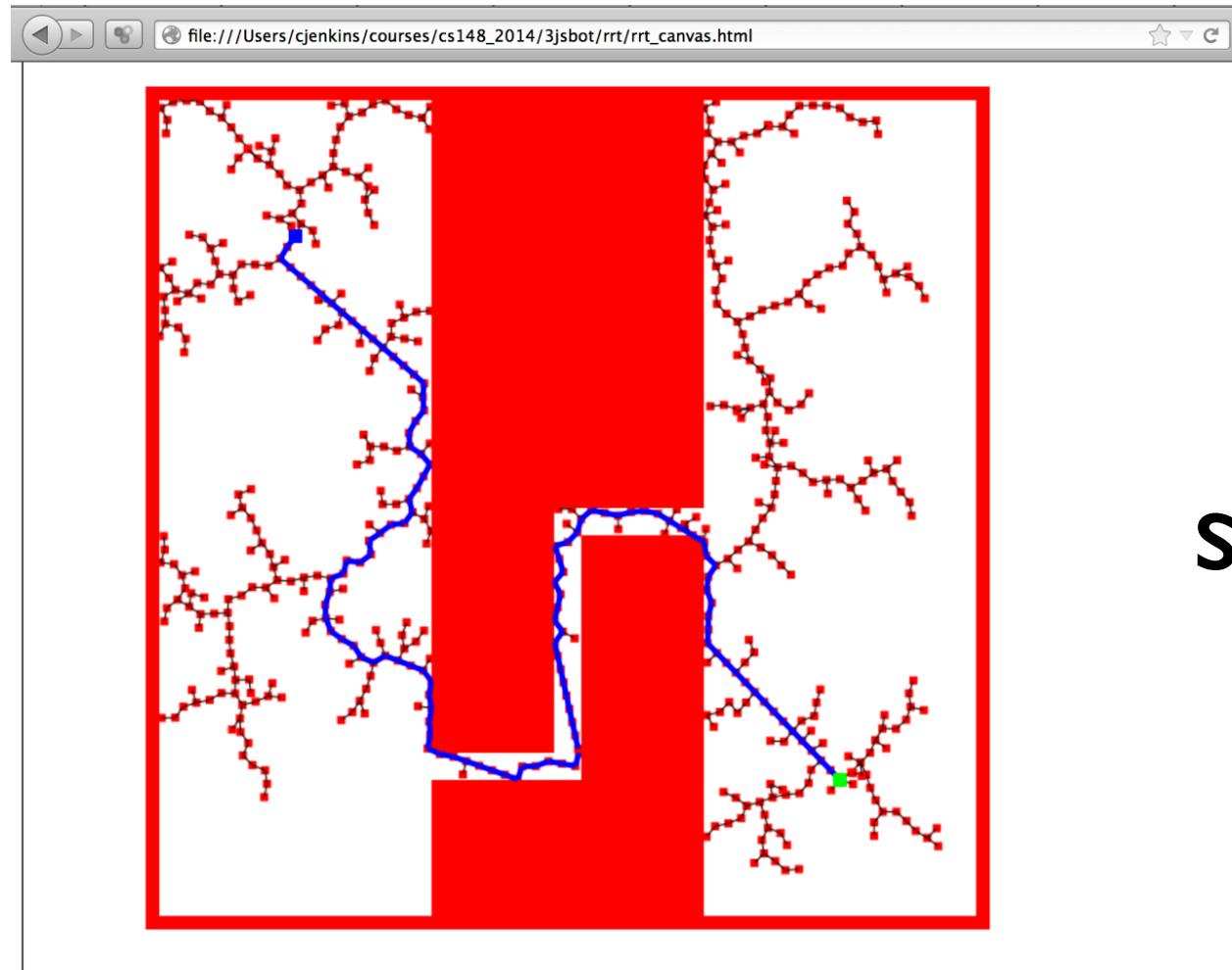
“We've made robot history”



Kuffner/Asimo Discovery Channel feature - <https://www.youtube.com/watch?v=wtVmhiTfm0Q>

Michigan EECS 398/567 ROB 510 - autorob.org

RRTs can take a lot of time...



Is there a
simpler way?



Potential fields

follow your
potential