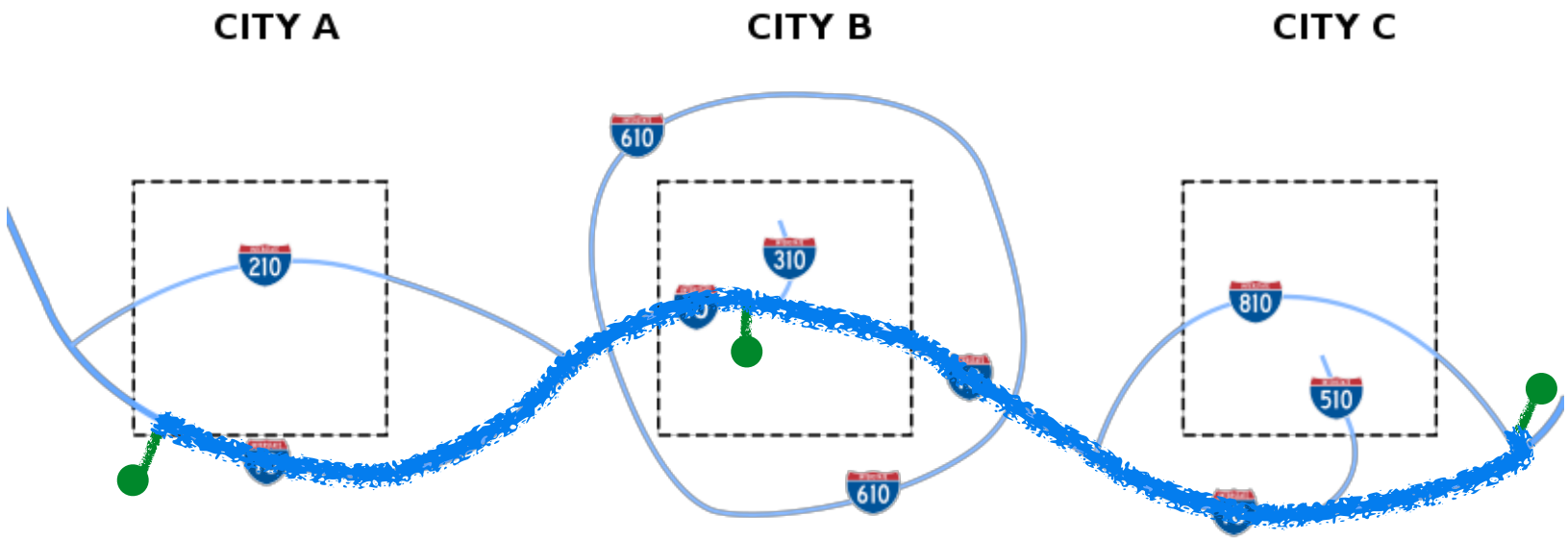
Planning with Sampling Roadmaps



Building a path to goal

Approaches to motion planning

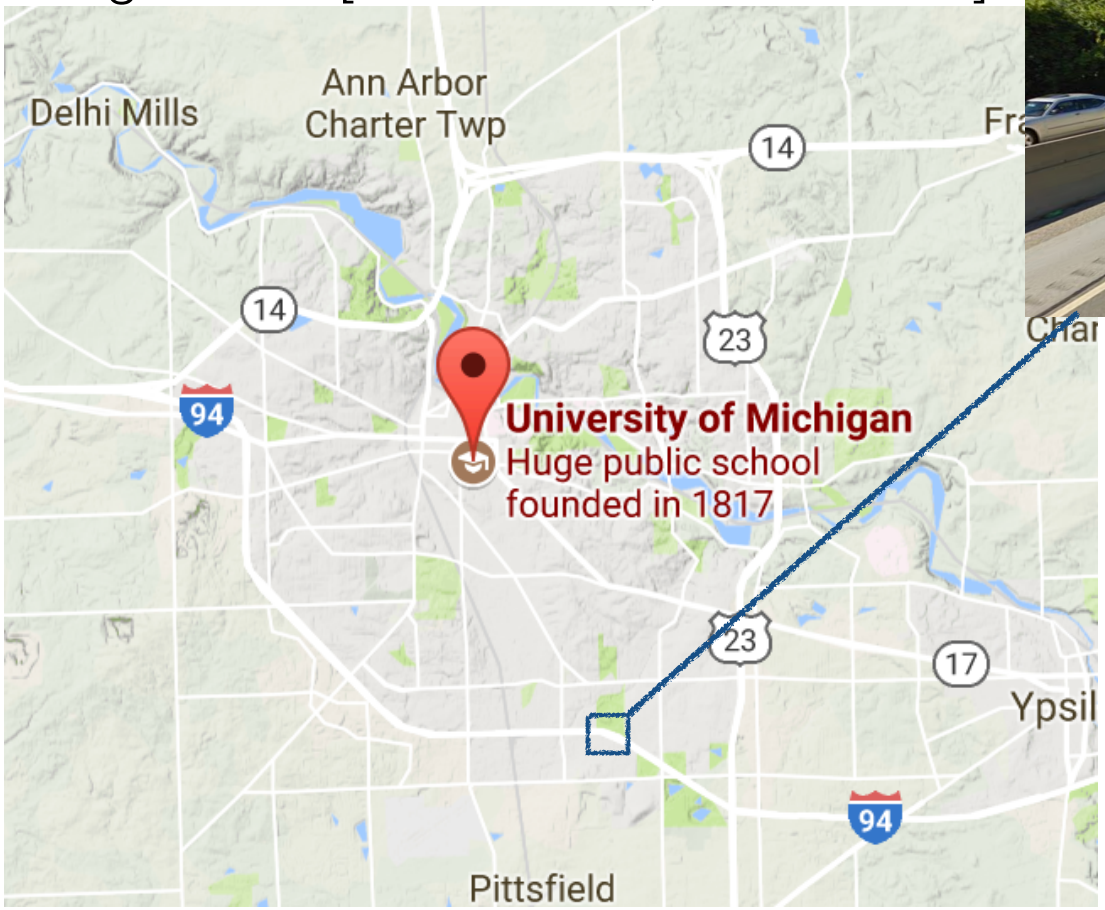
- Bug algorithms: Bug[0-2], Tangent Bug
- Graph Search (fixed graph)
  - Depth-first, Breadth-first, Dijkstra, A-star, Greedy best-first
- **Sampling-based Search (build graph):**
  - **Probabilistic Road Maps, Rapidly-exploring Random Trees**
- Optimization and local search:
  - Gradient descent, Potential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

Where is this?


EXIT 177
State St
1 MILE
I-94

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

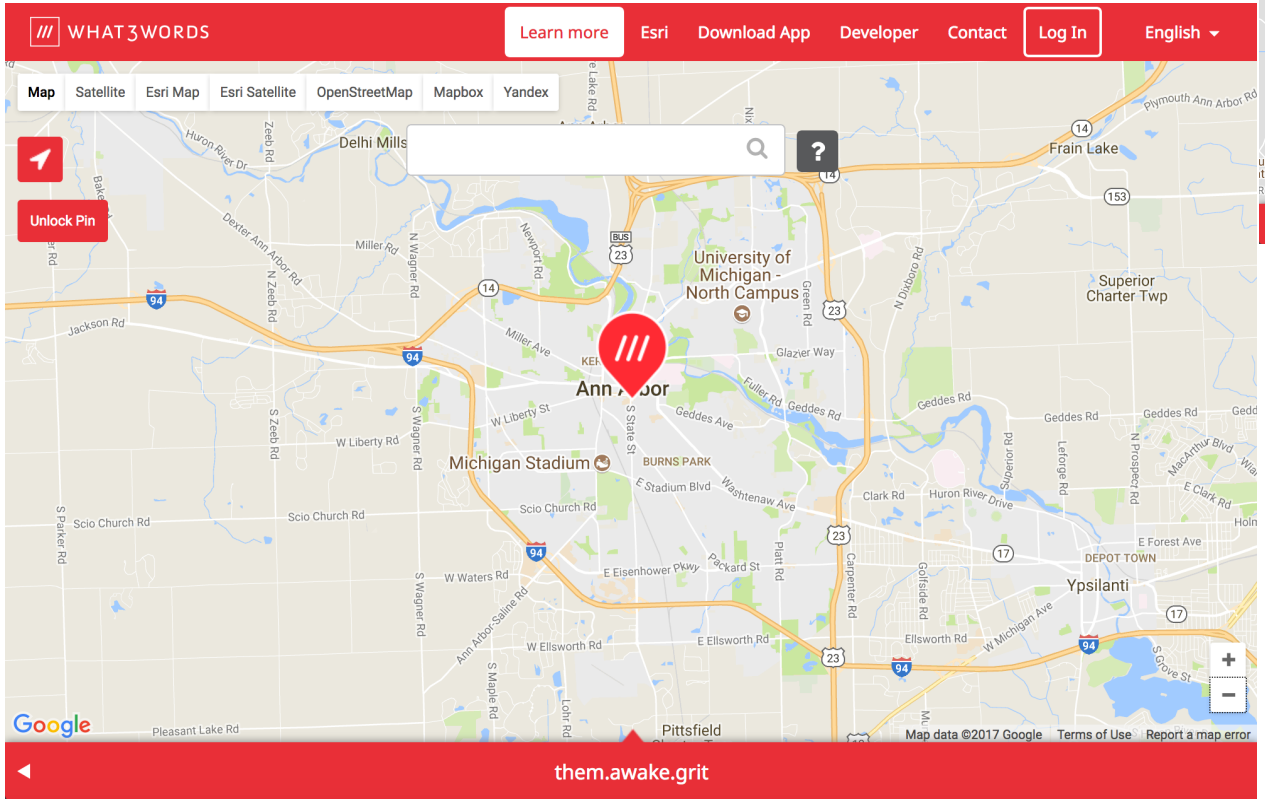
Configuration: [42.2780436,-83.7404128]

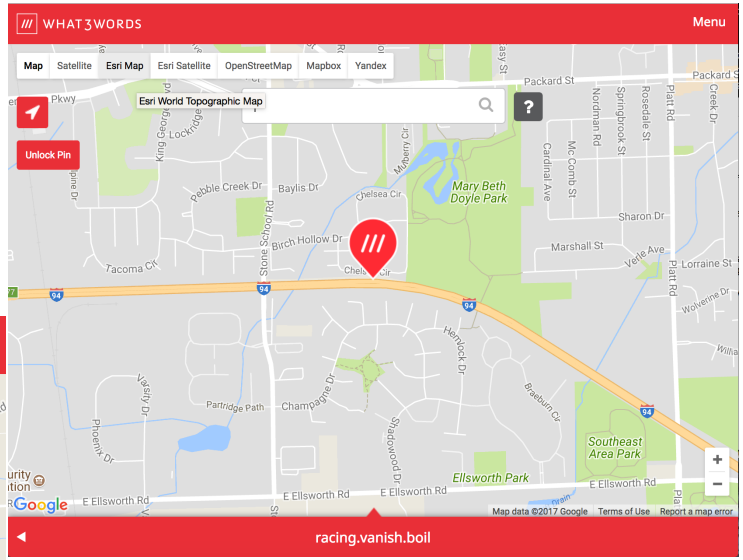


Configuration (lat,lon): [42.237631, -83.7147289]

Alternatively...

Configuration: [them.awake.grit]

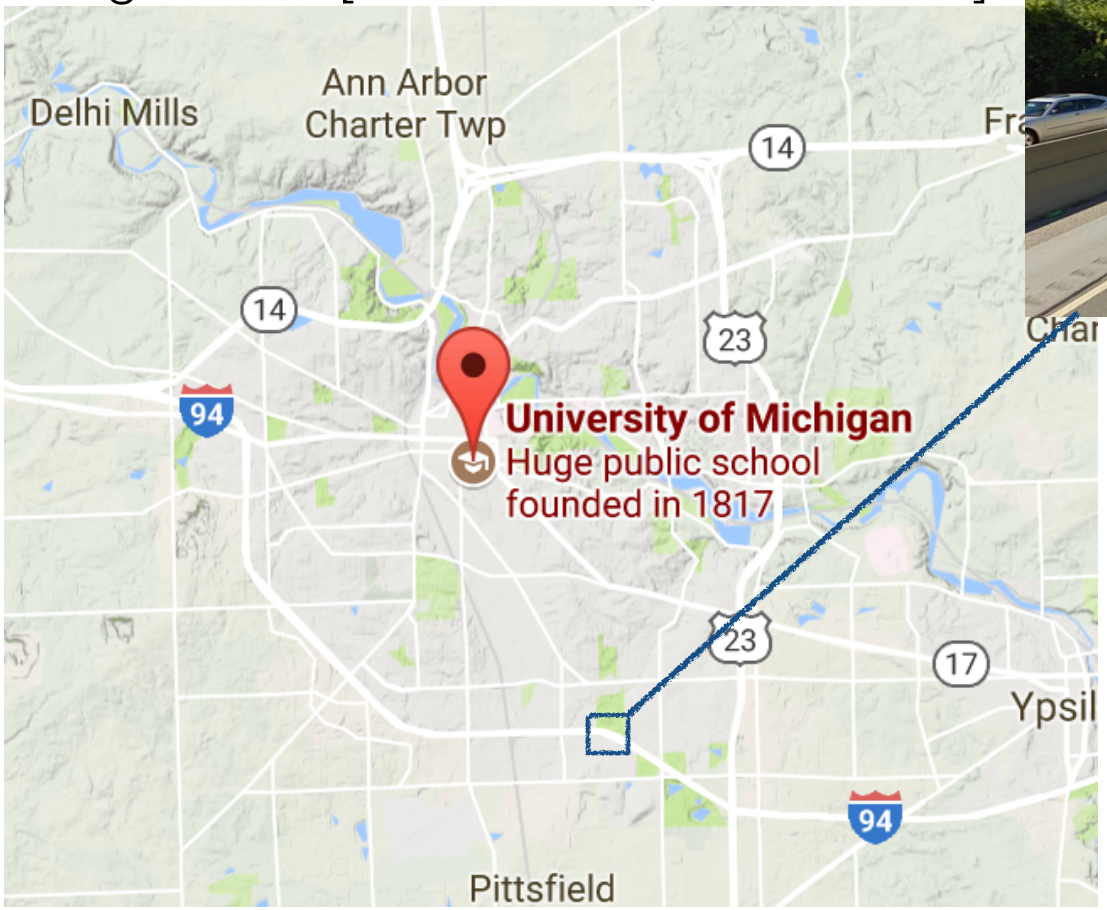




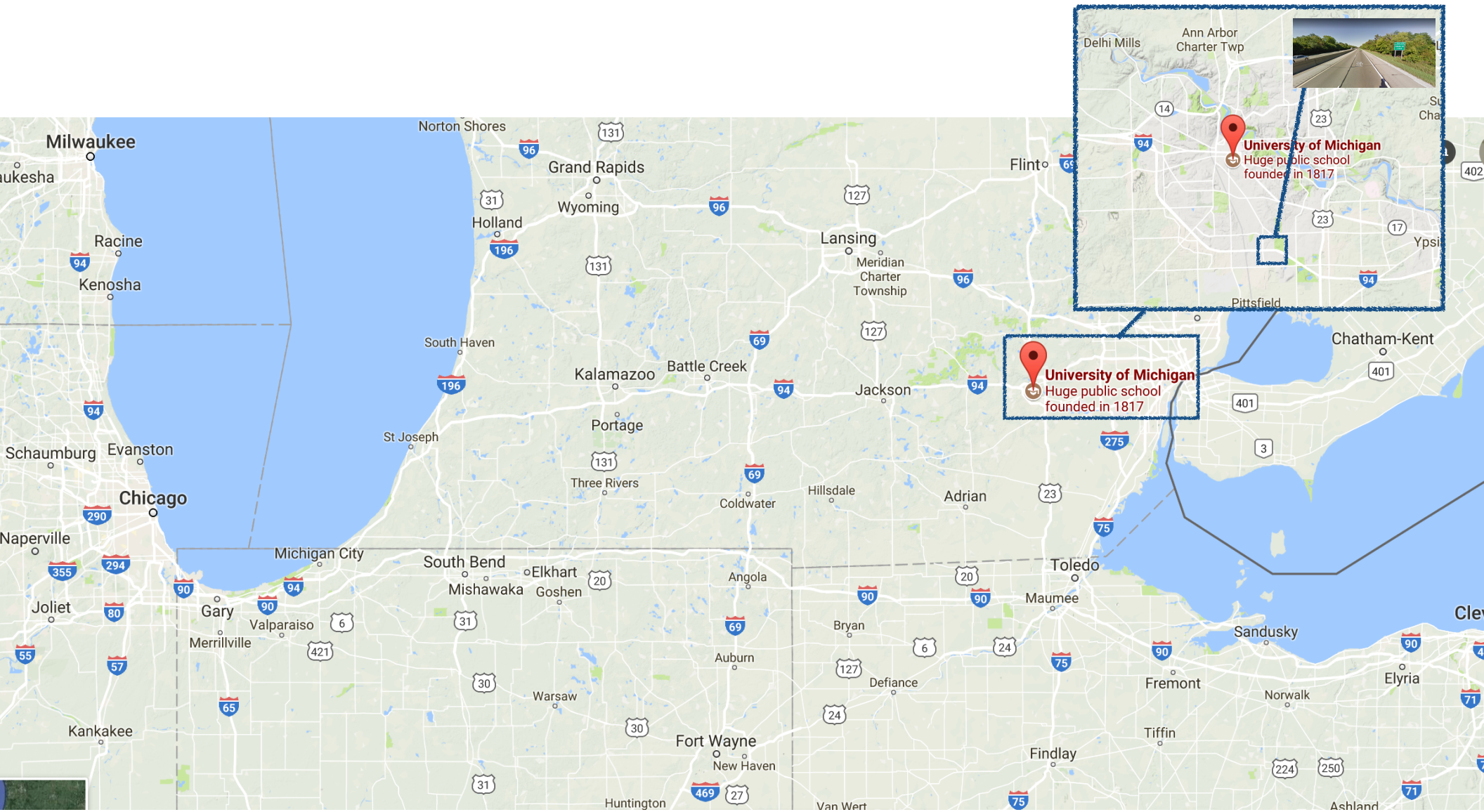
Configuration (lat,lon): [racing.vanish.boil]

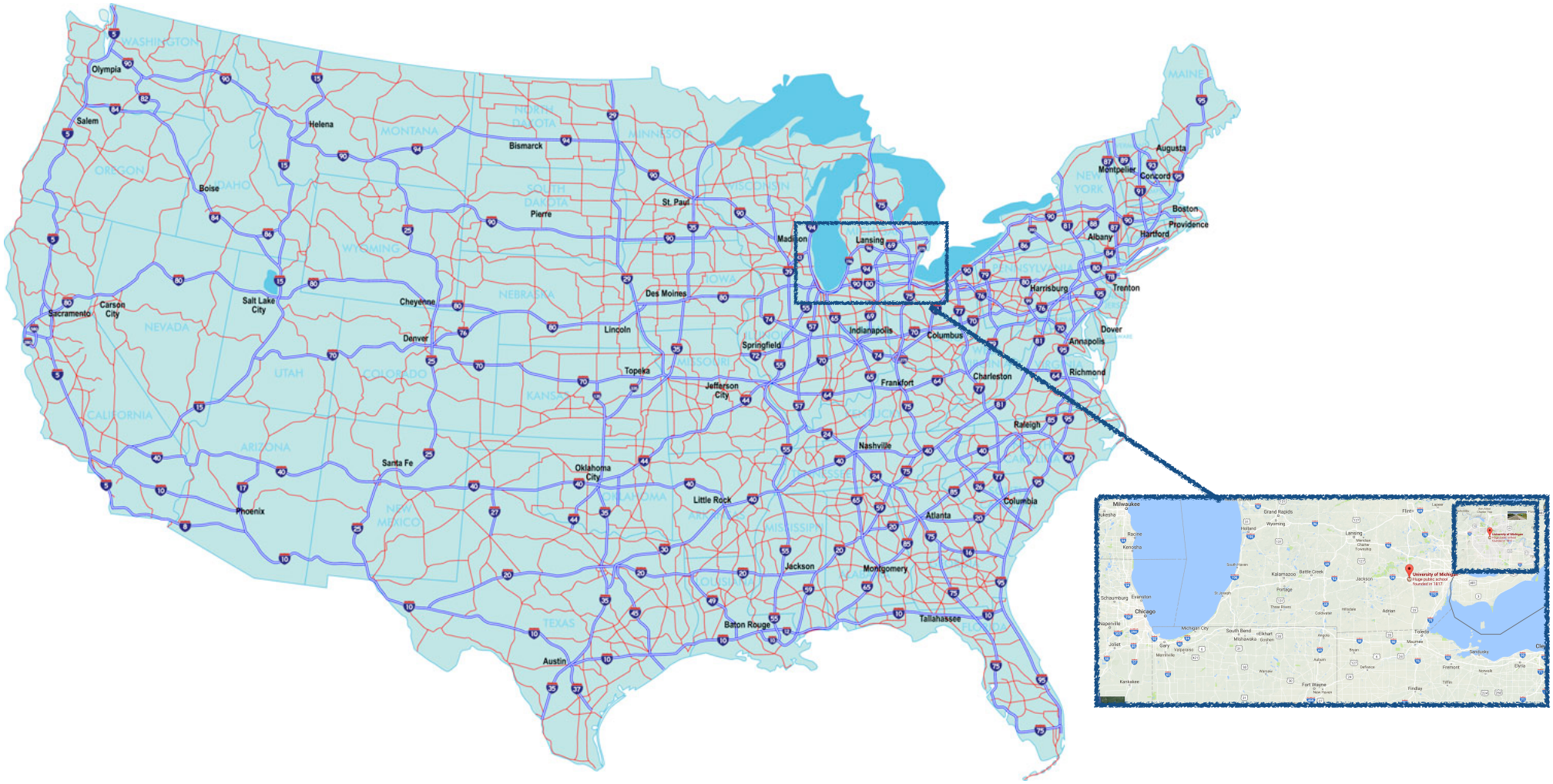
Is this a usable representation for configuration?

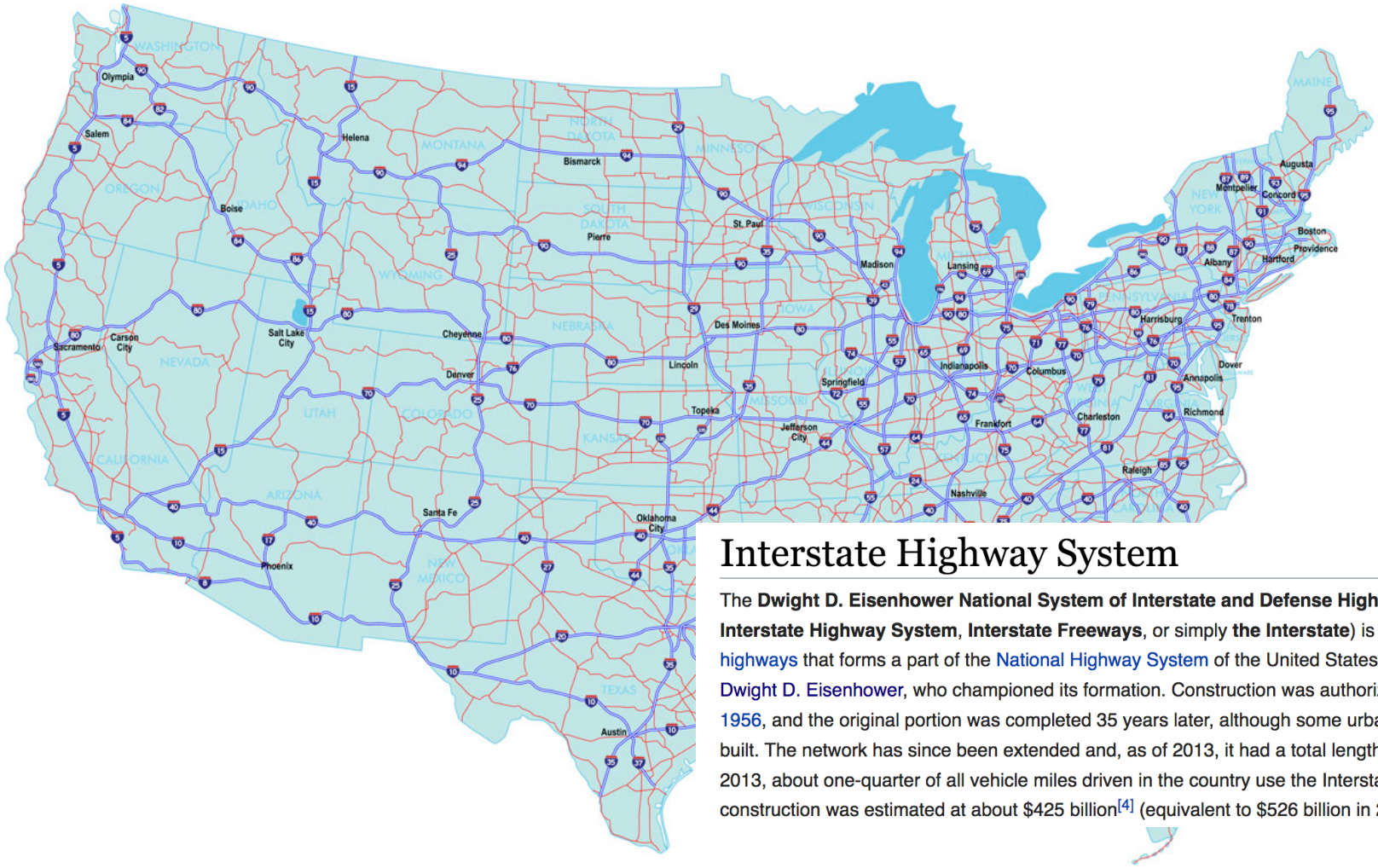
Configuration: [42.2780436,-83.7404128]



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


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
Schaumburg
Evanston
Chicago
Naperville
Joliet
Gary
Valparaiso
Merrillville
Michigan City
South Bend
Mishawaka
Elkhart
Goshen
Angola
Auburn
Warsaw
Fort Wayne
New Haven
Huntington
Kankakee
University of Michigan
Huge public school
founded in 1817
Delhi Mills
Ann Arbor Charter Twp
University of Michigan
Huge public school
founded in 1817

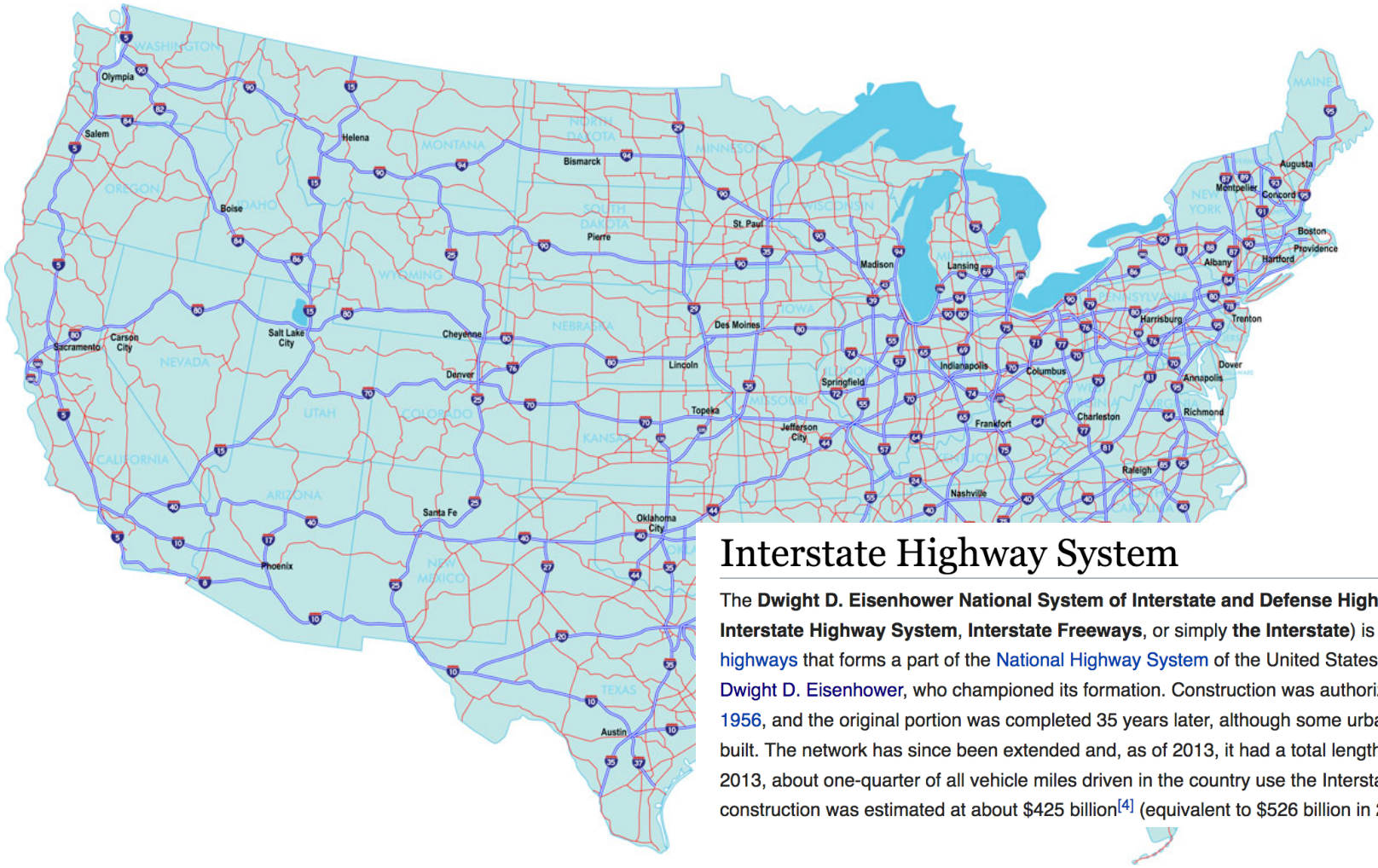

Olympia
Salem
Helena
Bismarck
Boise
Pierre
St. Paul
Madison
Lansing
Augusta
Montpelier
Concord
Boston
Providence
Albany
Hartford
Harrisburg
Trenton
Dover
Annapolis
Richmond
Sacramento
Carson City
Salt Lake City
Cheyenne
Denver
Des Moines
Lincoln
Topeka
Jefferson City
Springfield
Indianapolis
Columbus
Frankfort
Charleston
Raleigh
Nashville
Columbia
Atlanta
Montgomery
Jackson
Tallahassee
Baton Rouge
Little Rock
Oklahoma City
Santa Fe
Phoenix
Austin
Milwaukee
Chicago
University of Michigan
Toledo



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]).

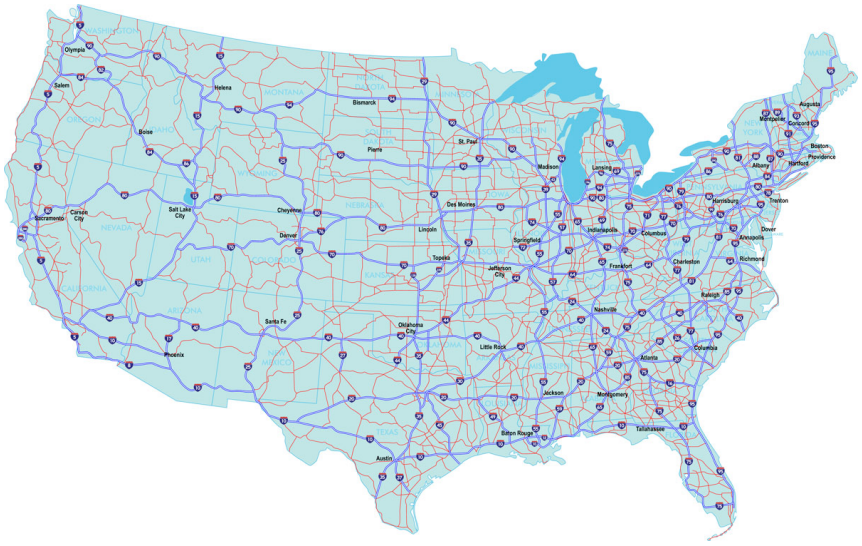

CHINA'S PROPOSED
BELT AND ROAD INITIATIVE
Europe
Europe
Central Asia
CHINA
Persian Gulf
Mediterranean Sea
Western Asia
South Asia
South China Sea
Southeast Asia
South Pacific Ocean
Indian Ocean



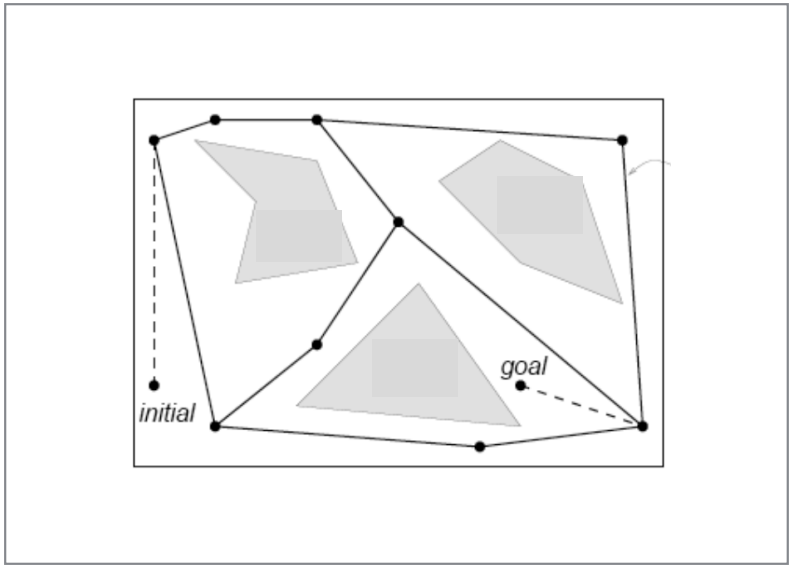
Interstate Highway System

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

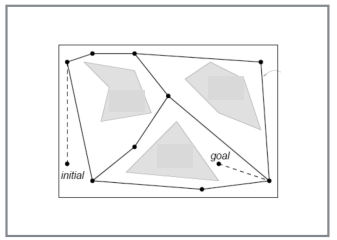


Roadmap over geolocations

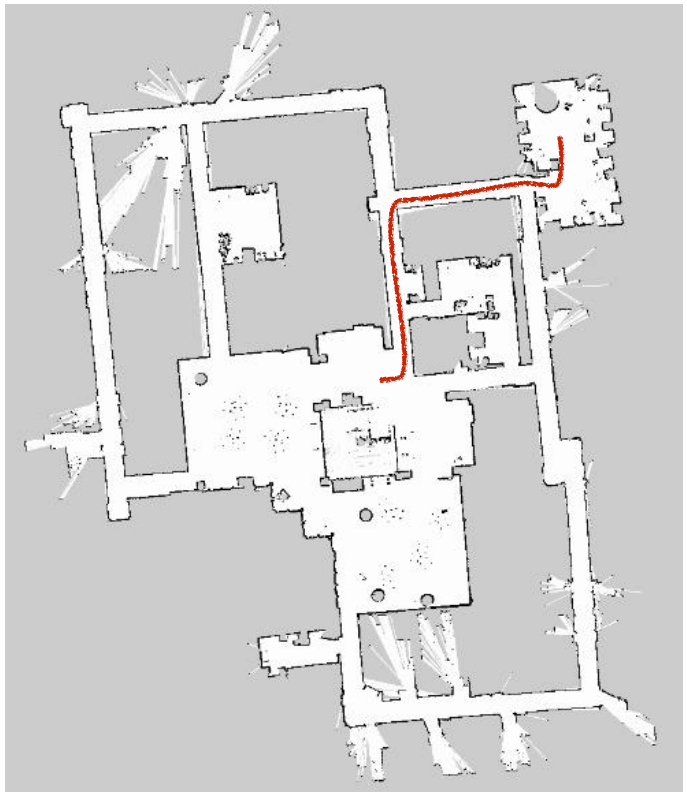


Roadmap over robot configurations

Roadmaps

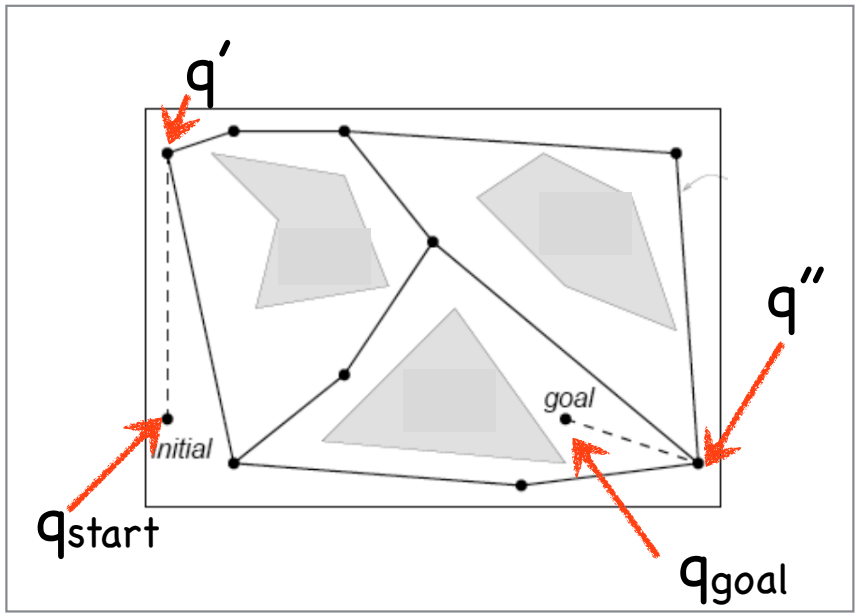


- Graph search assumed C-space as a fixed uniform grid
  - finite set of discretized cells
- How does this scale beyond planar navigation?
  - curse of dimensionality
- Roadmaps are a more general notion of graphs in C-space



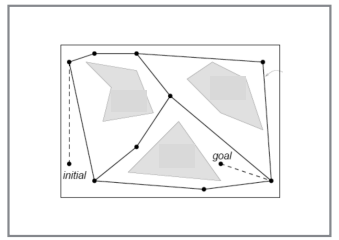
Roadmap Definition

- 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

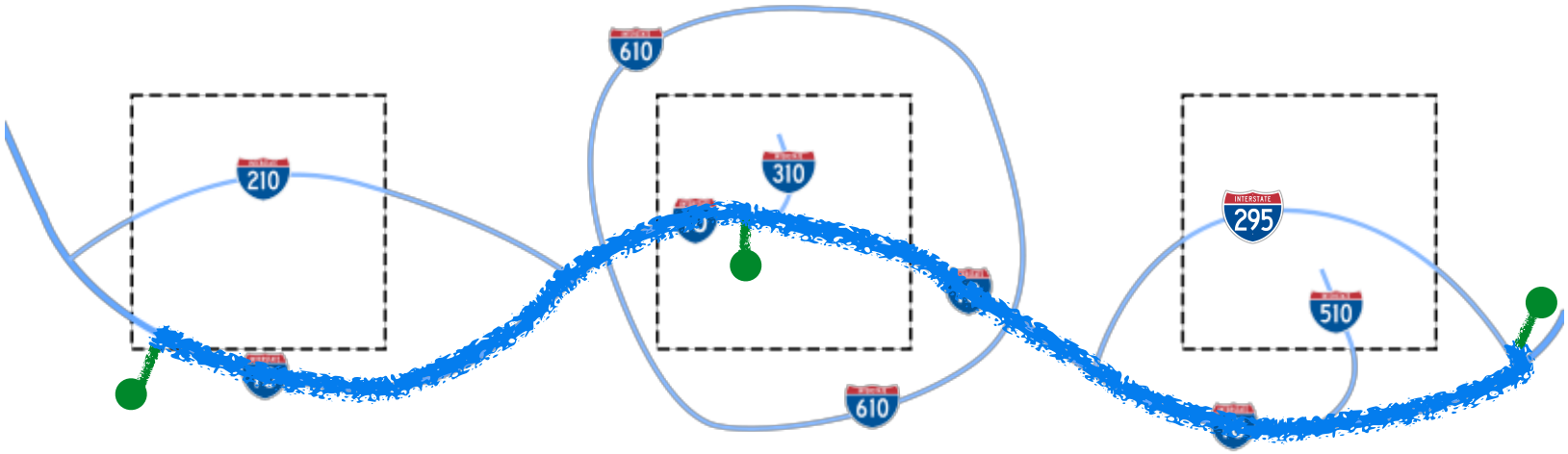


- Roadmap properties:
  - Accessibility: There is a path from $q_{start} \in Q_{free}$ to some $q' \in RM$
  - Departability: There is a path from $q'' \in RM$ to $q_{goal} \in Q_{free}$
  - Connectivity: there exists a path in *RM* between *q'* and *q''*

What cities? What universities?



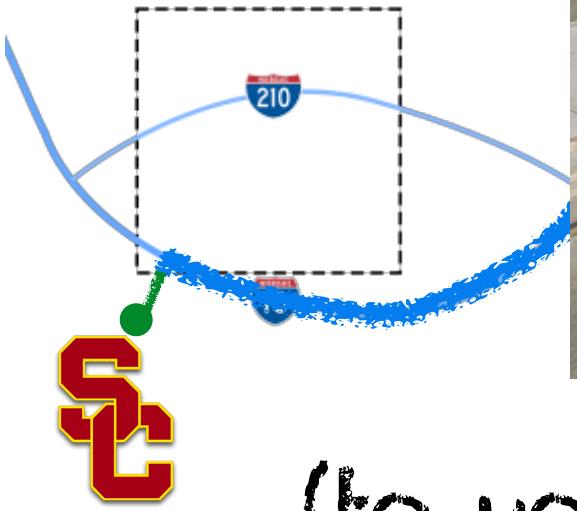
??????? ??????? ????????



(to your best approximation)

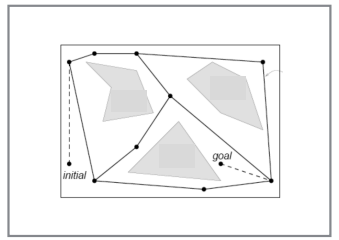
What cities? W

"Los Angeles"

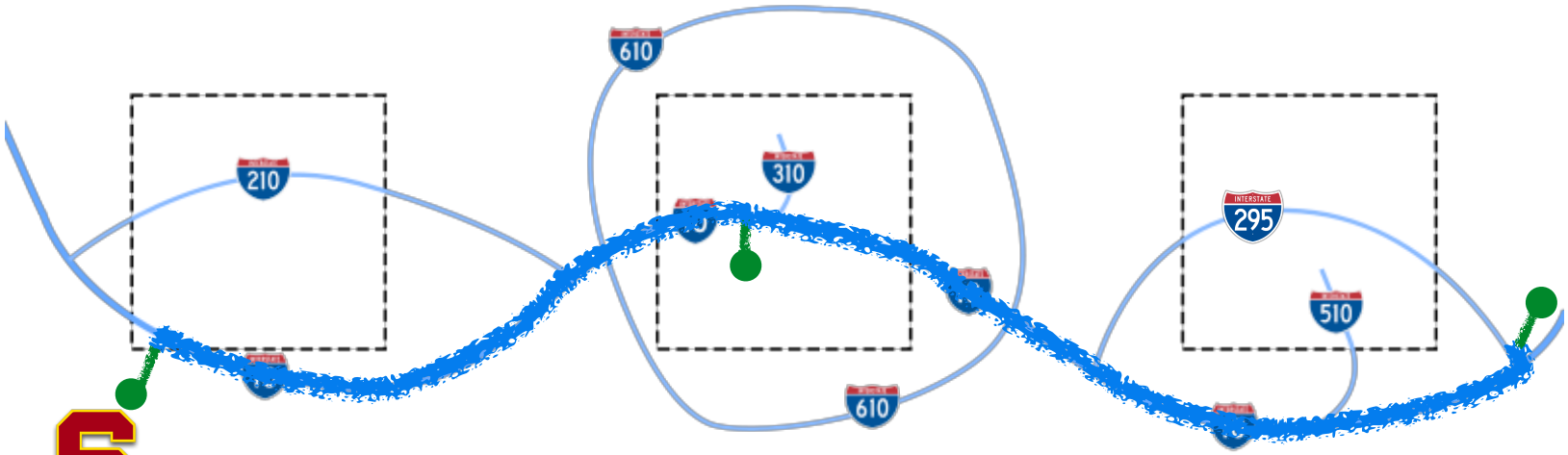


(to your best approximation)

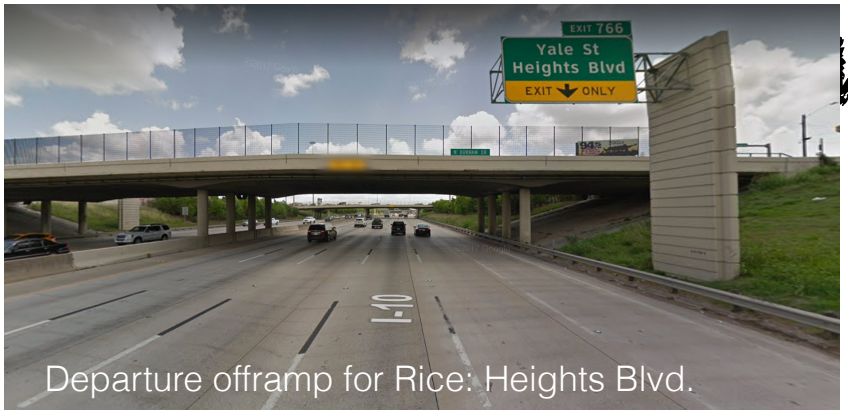
What cities? What universities?



"Los Angeles" ??????? ????????

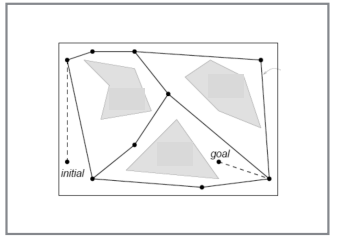


(to your best approximation)

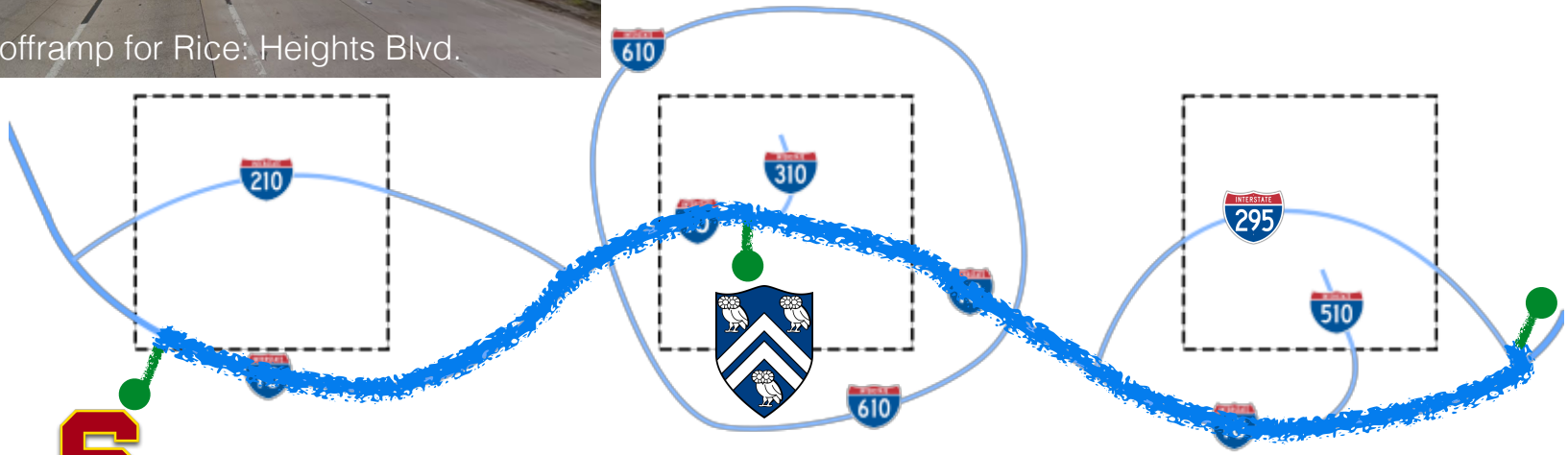


Departure offramp for Rice: Heights Blvd.

hat universities?

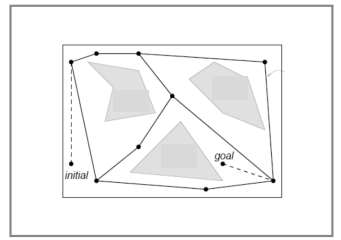


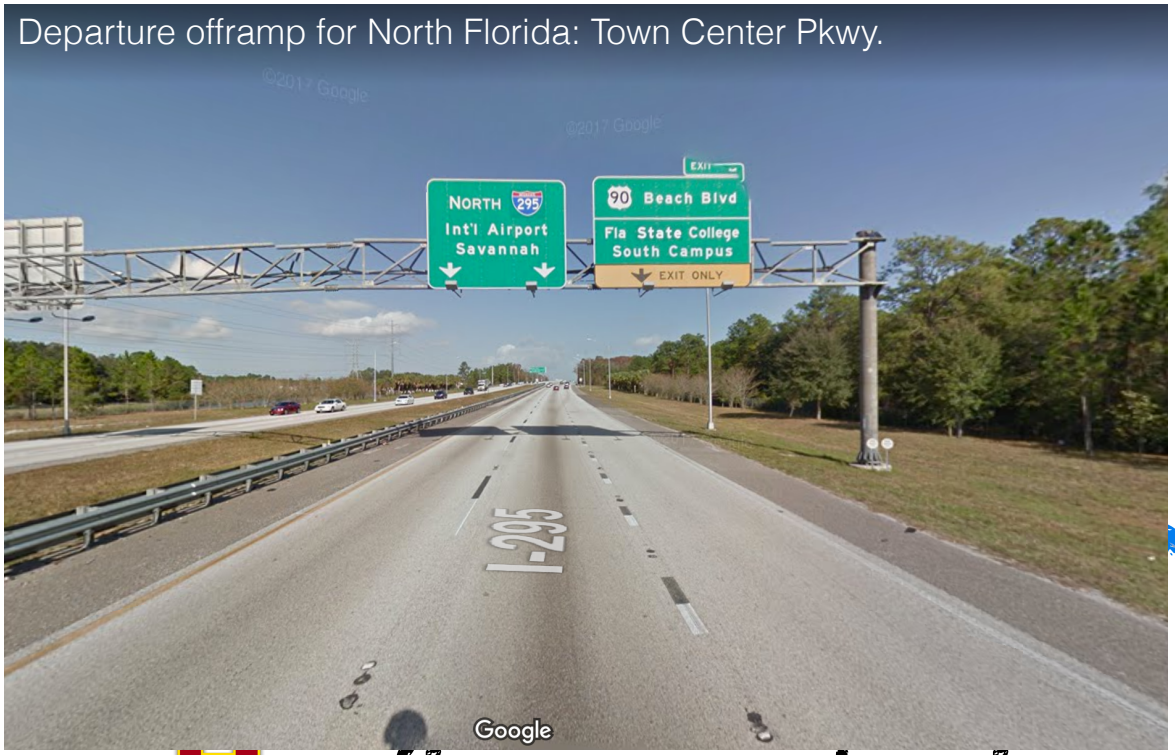
"Houston" ????????



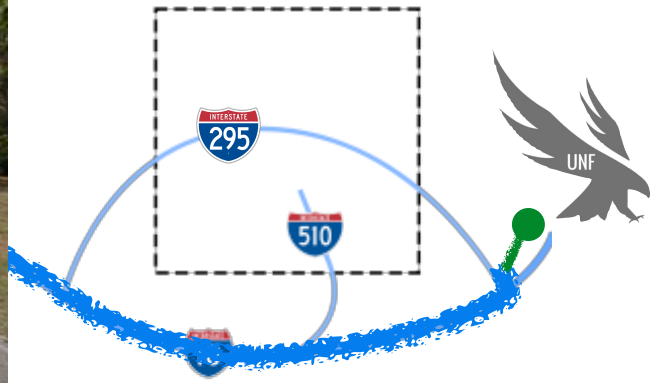
(to your best approximation)

What cities? What universities?





"Jacksonville"

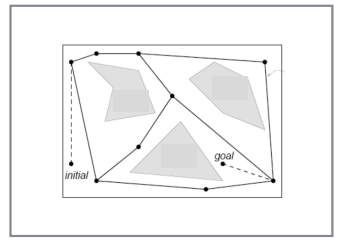


(to your best approximation)


Departure offramp for Michigan: State St.
EXIT 177
State St
1 MILE
I-94

Better departure offramp for Michigan: Plymouth Rd.

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 assume 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
- Voronoi
- Silhouette Method

Probabilistic:

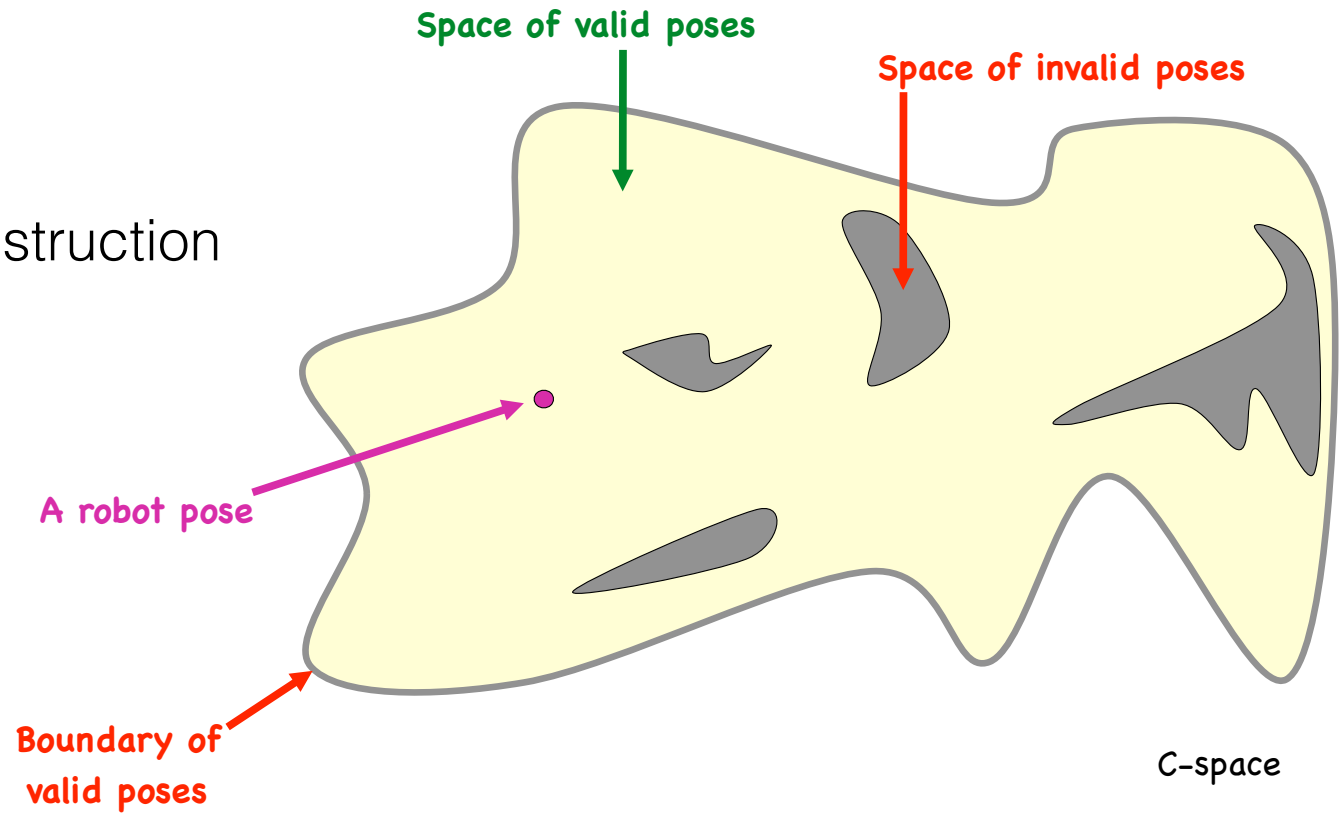
C-space sampling

- PRM (multi-query)
- RRT (single query)

[Kavraki, Svetska, Latombe, Overmars, 95]

Probabilistic road maps

- Two phases
  - Roadmap construction
  - Path Query

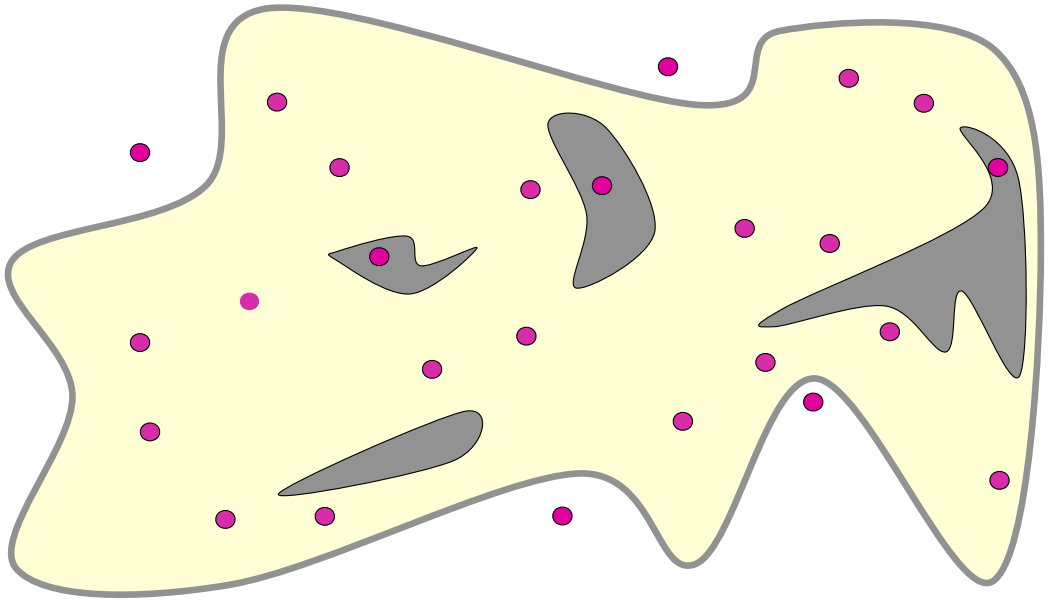


PRM: construction phase

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

2) Eliminate invalid poses

3) Connect neighboring poses

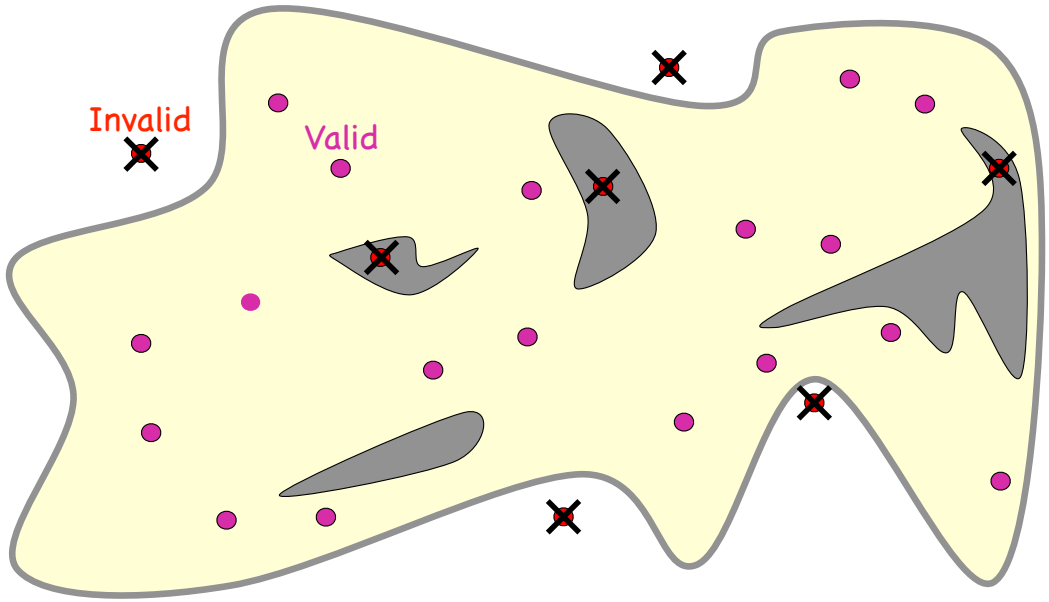


PRM: construction phase

1) Select N sample poses at random

2) **Eliminate invalid poses**

3) Connect neighboring poses

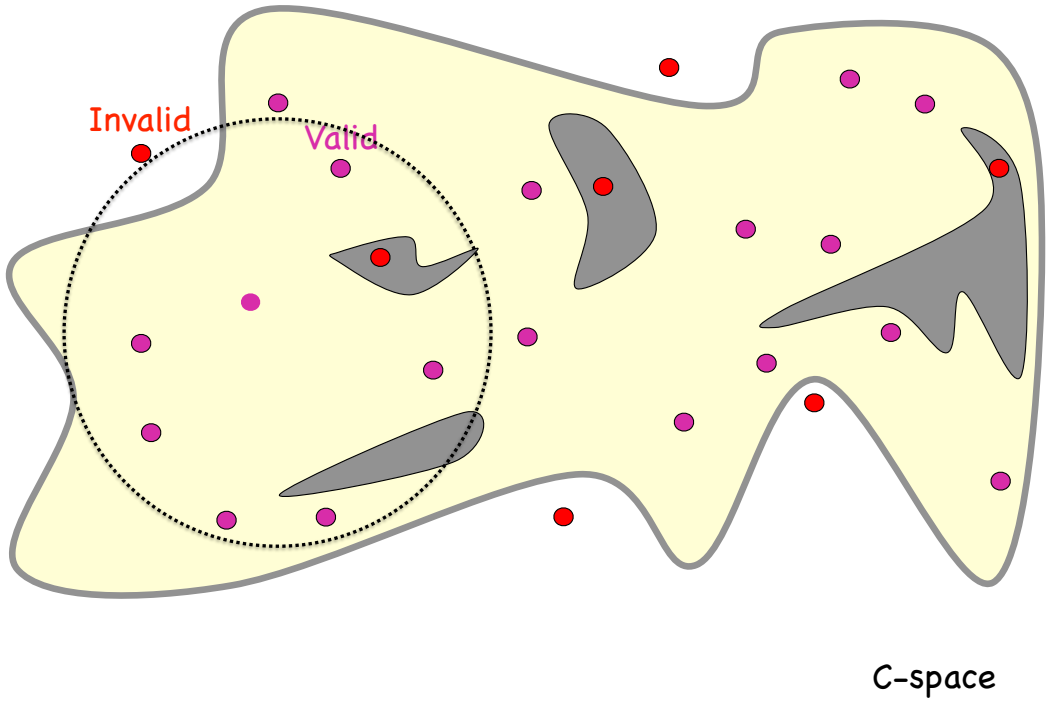


PRM: construction phase

1) Select N sample poses at random

2) Eliminate invalid poses

3) **Connect neighboring poses**

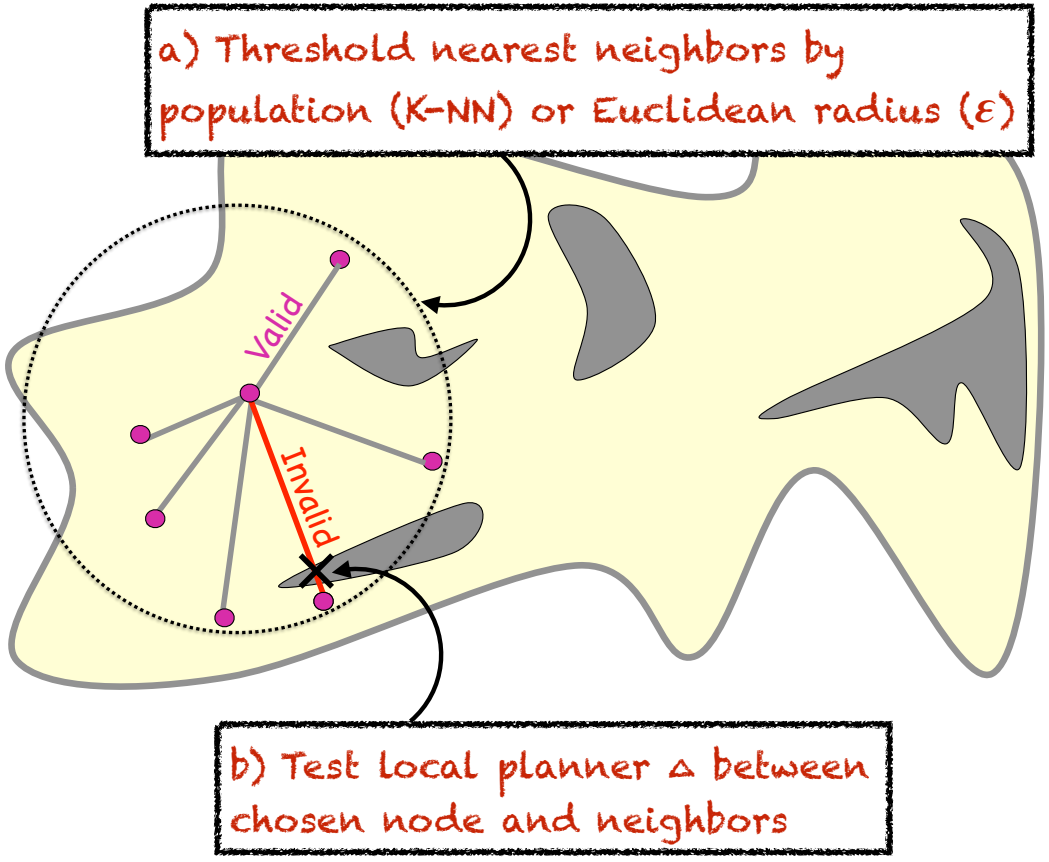


PRM: construction phase

1) Select N sample poses at random

2) Eliminate invalid poses

3) **Connect neighboring poses**



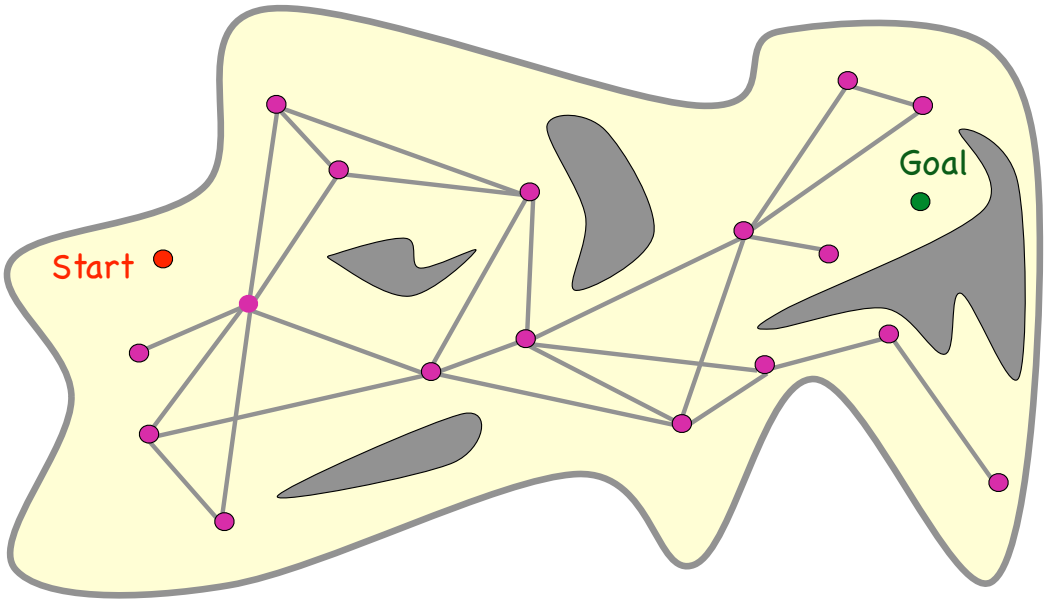
PRM: query phase

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



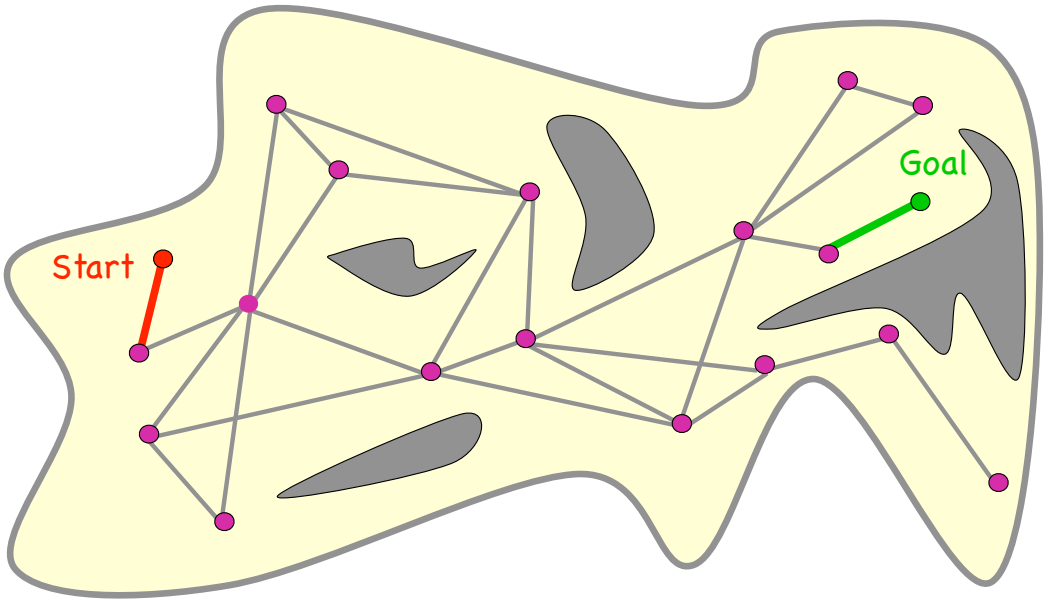
PRM: query phase

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



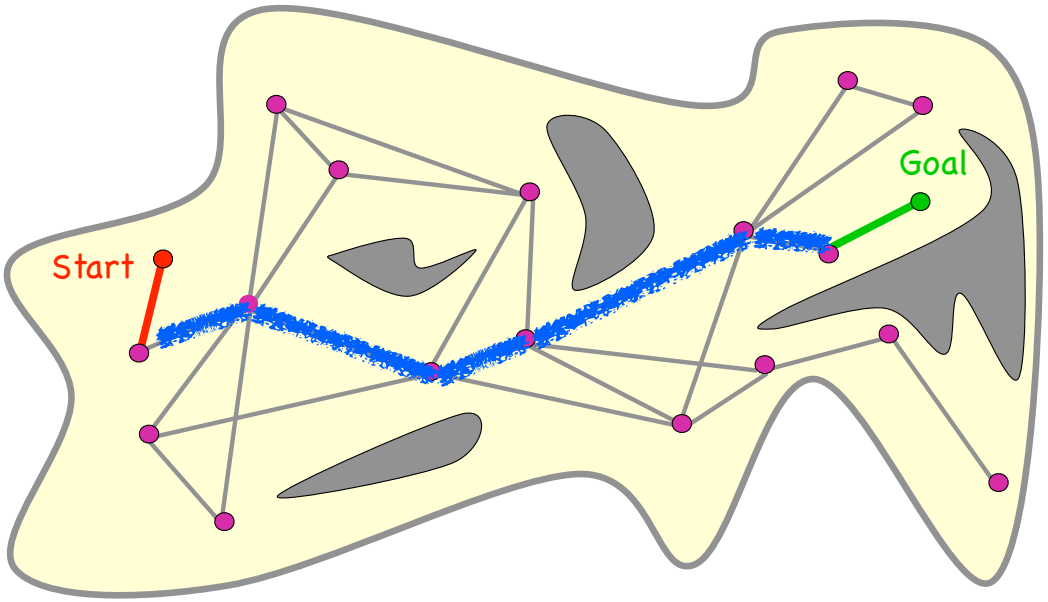
PRM: query phase

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

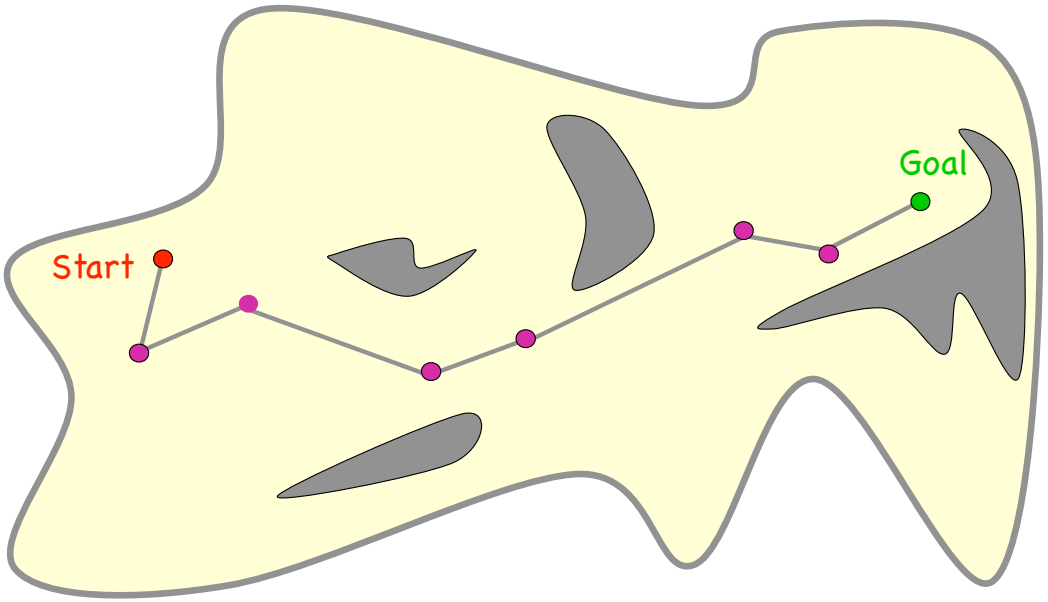
PRM: query phase

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**



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)

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
```

```
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;
```

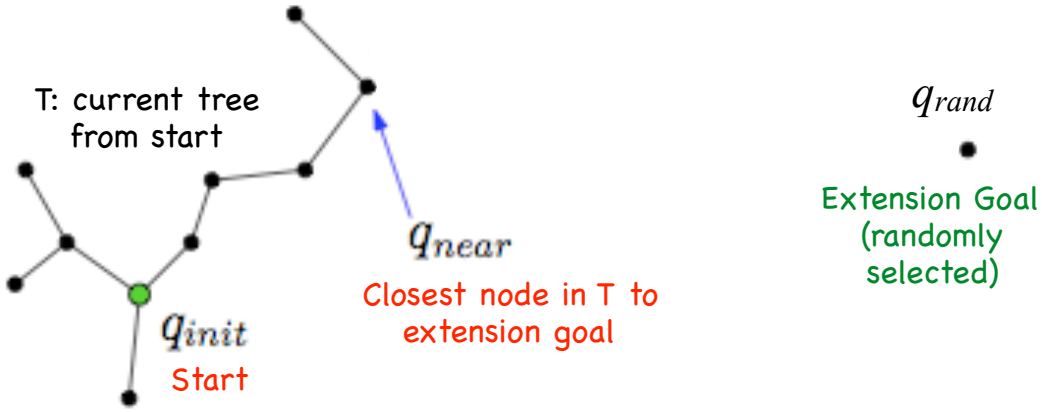


Figure 3: The EXTEND operation.

Extend graph towards a random configuration

[Kuffner, LaValle 2000]

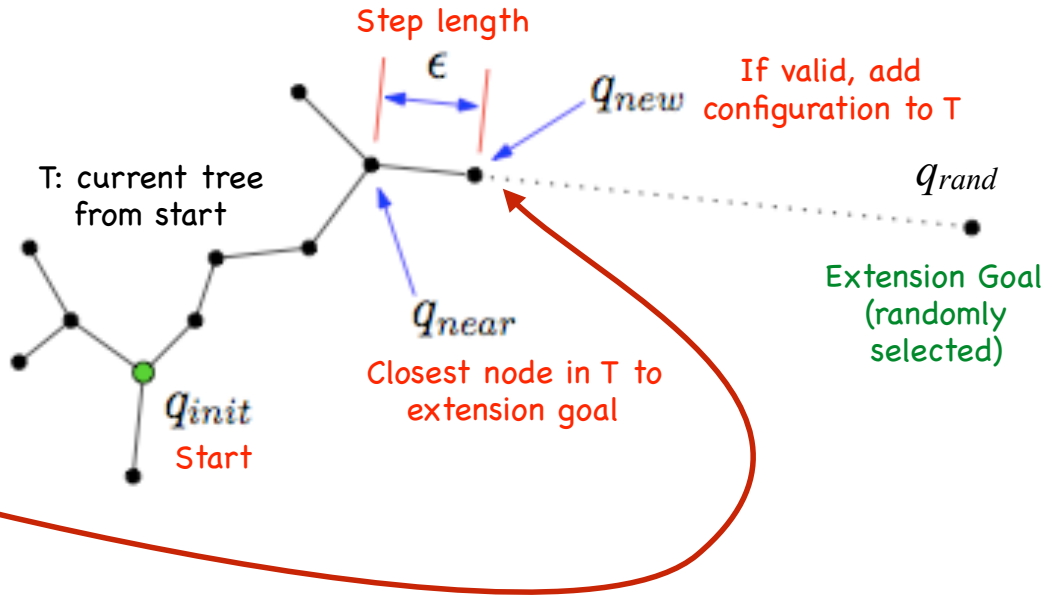
RRT Algorithm

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
```

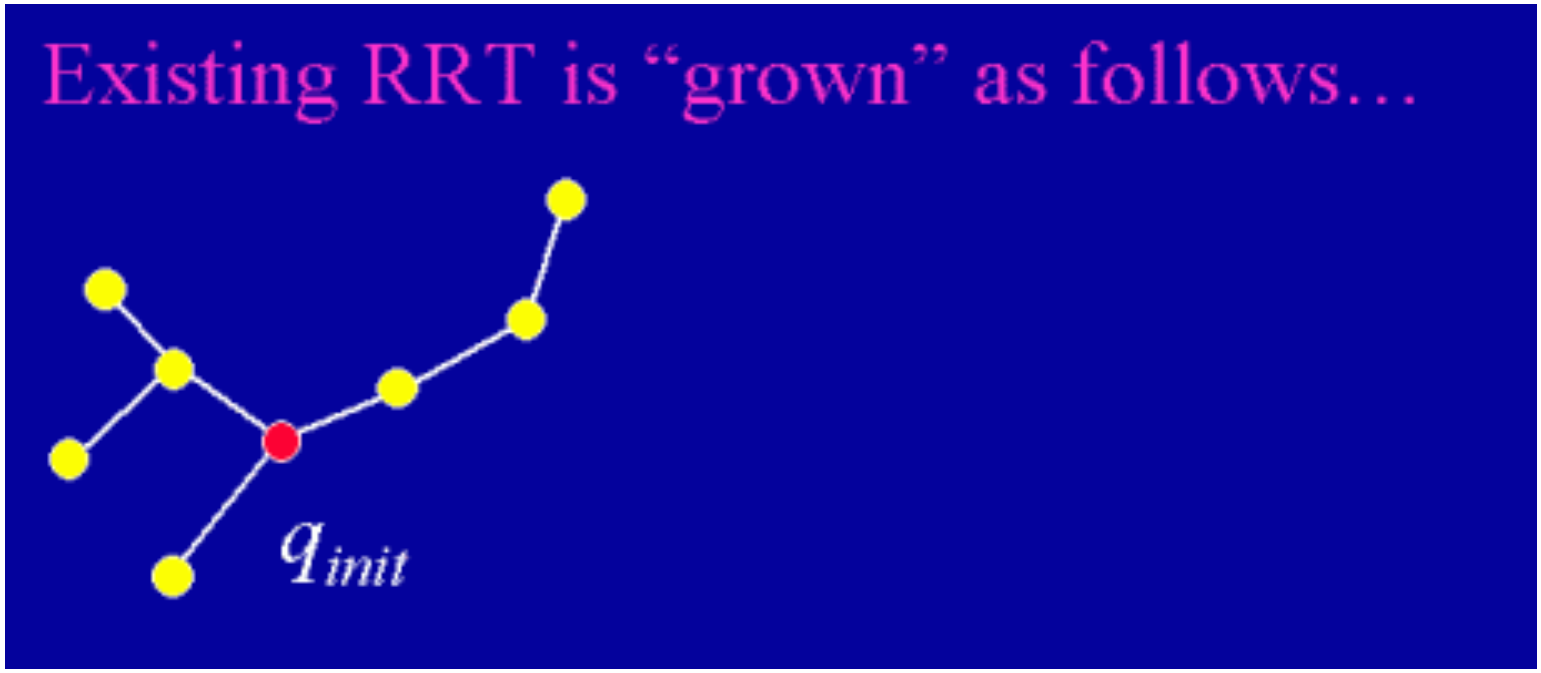
```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

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

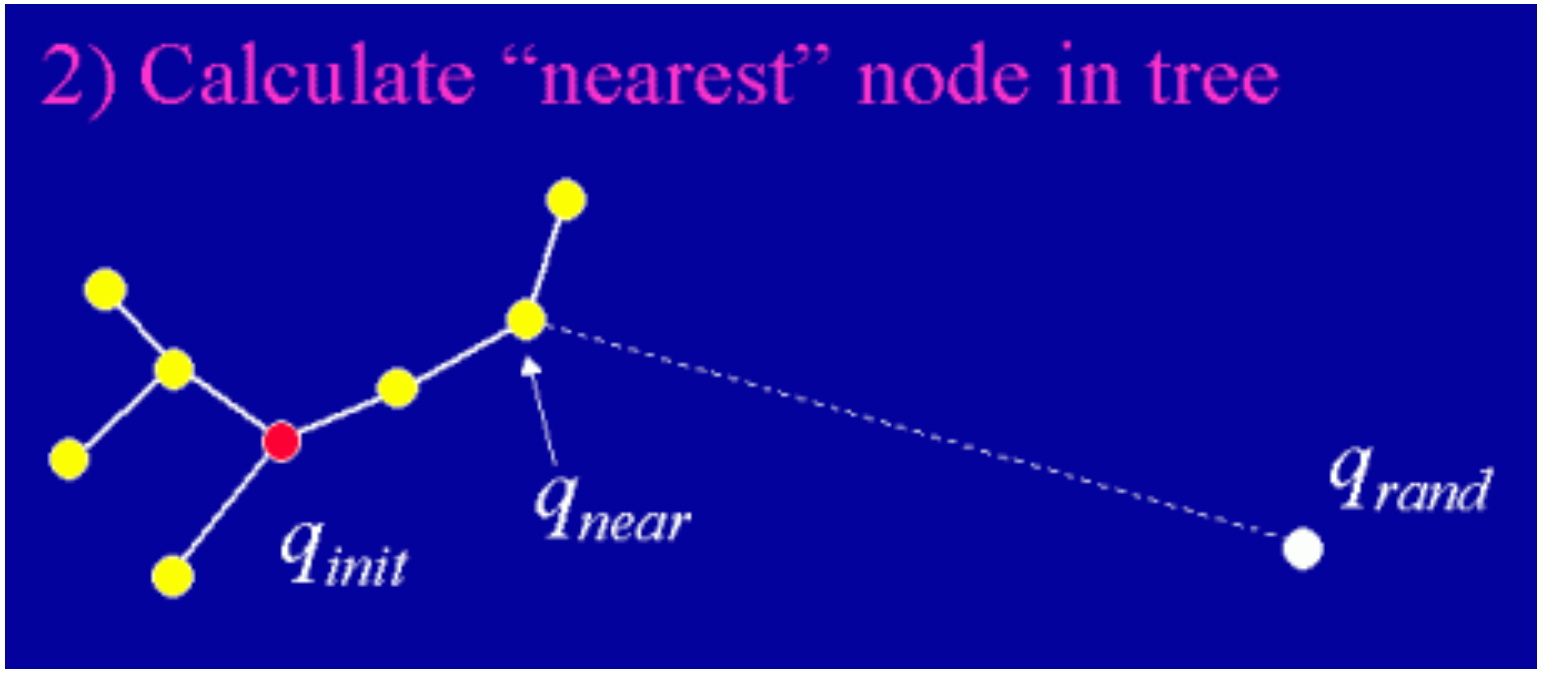


http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation

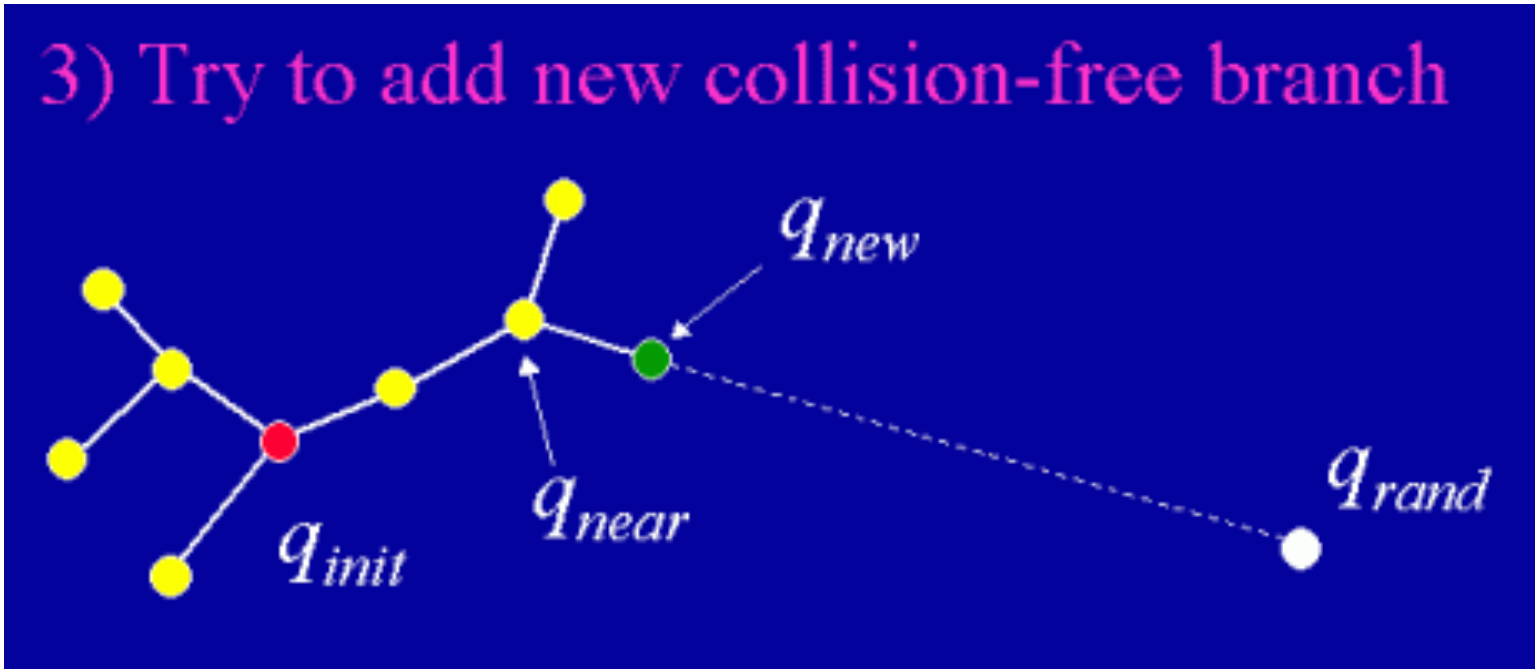
http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation



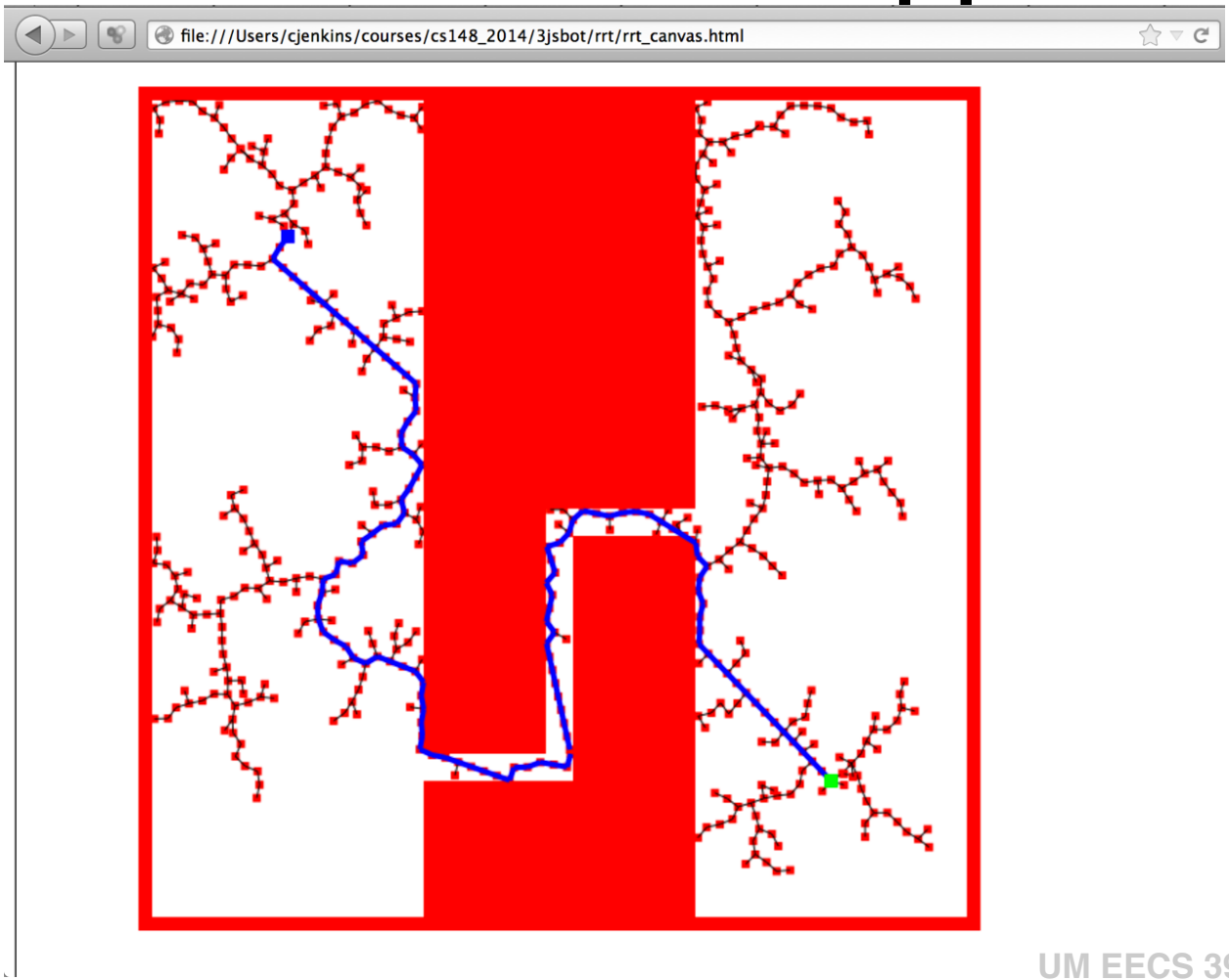
http://www.kuffner.org/james/plan/algorithm.php

RRT Extend animation



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

Let's see what happens

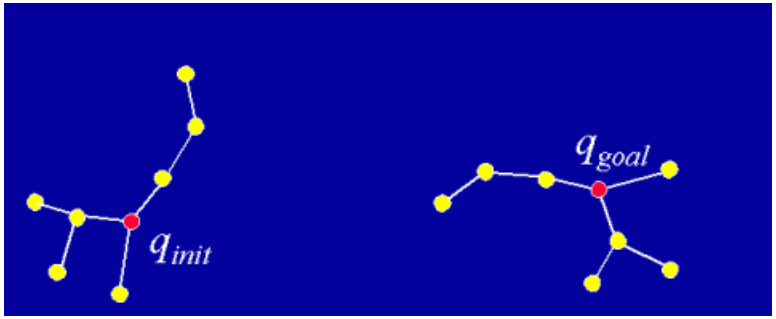


RRT Connect

[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

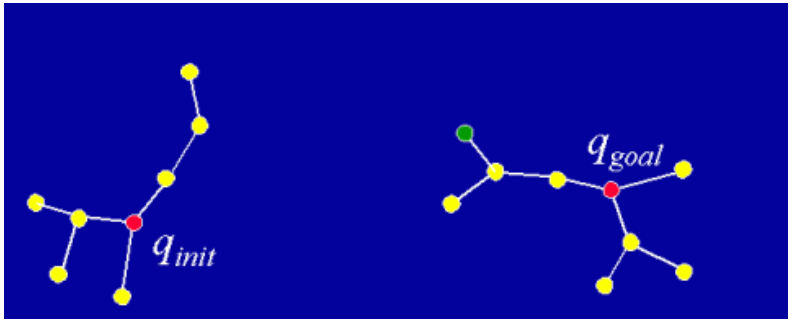
[Kuffner, LaValle 2000]

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

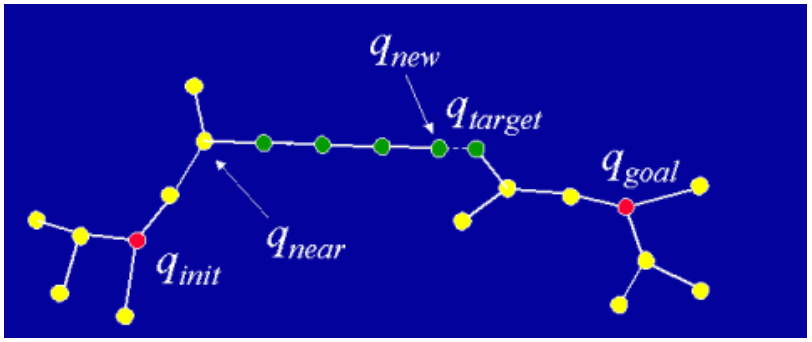
[Kuffner, LaValle 2000]

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

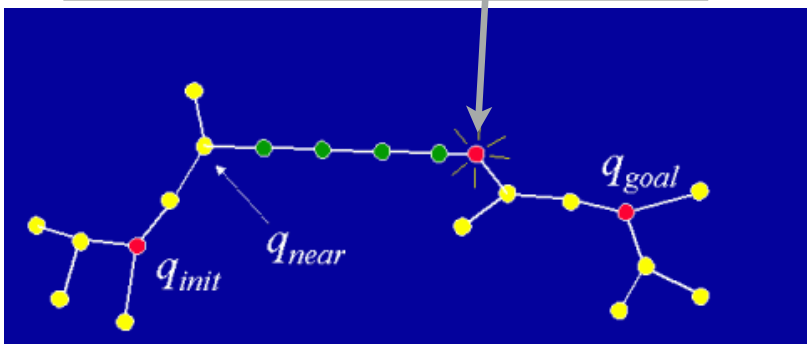
RRT Connect

[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
```

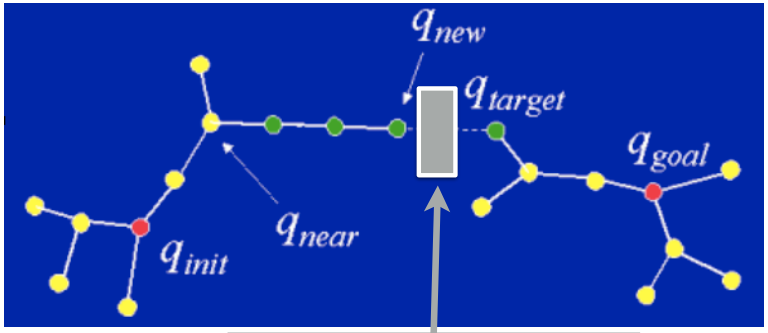
```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

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;
```

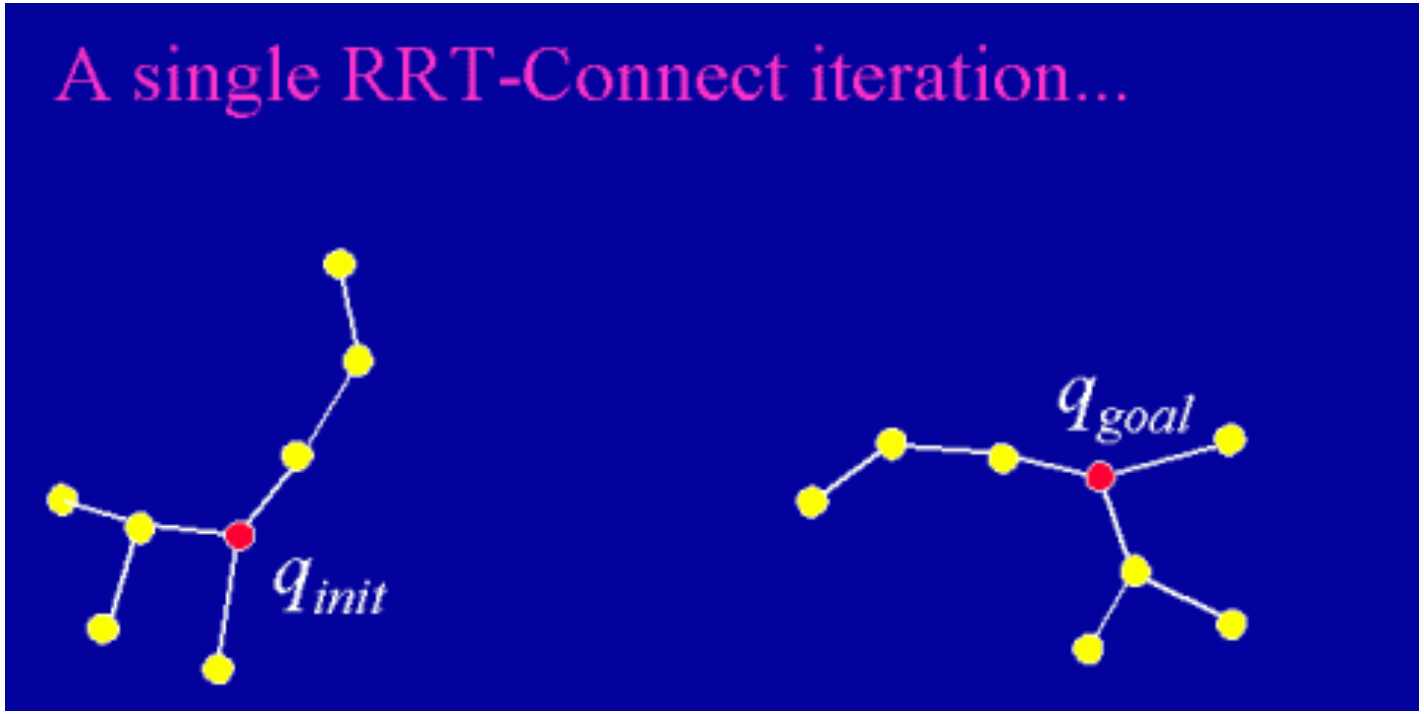
2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

3) reverse roles for trees A and B and repeat



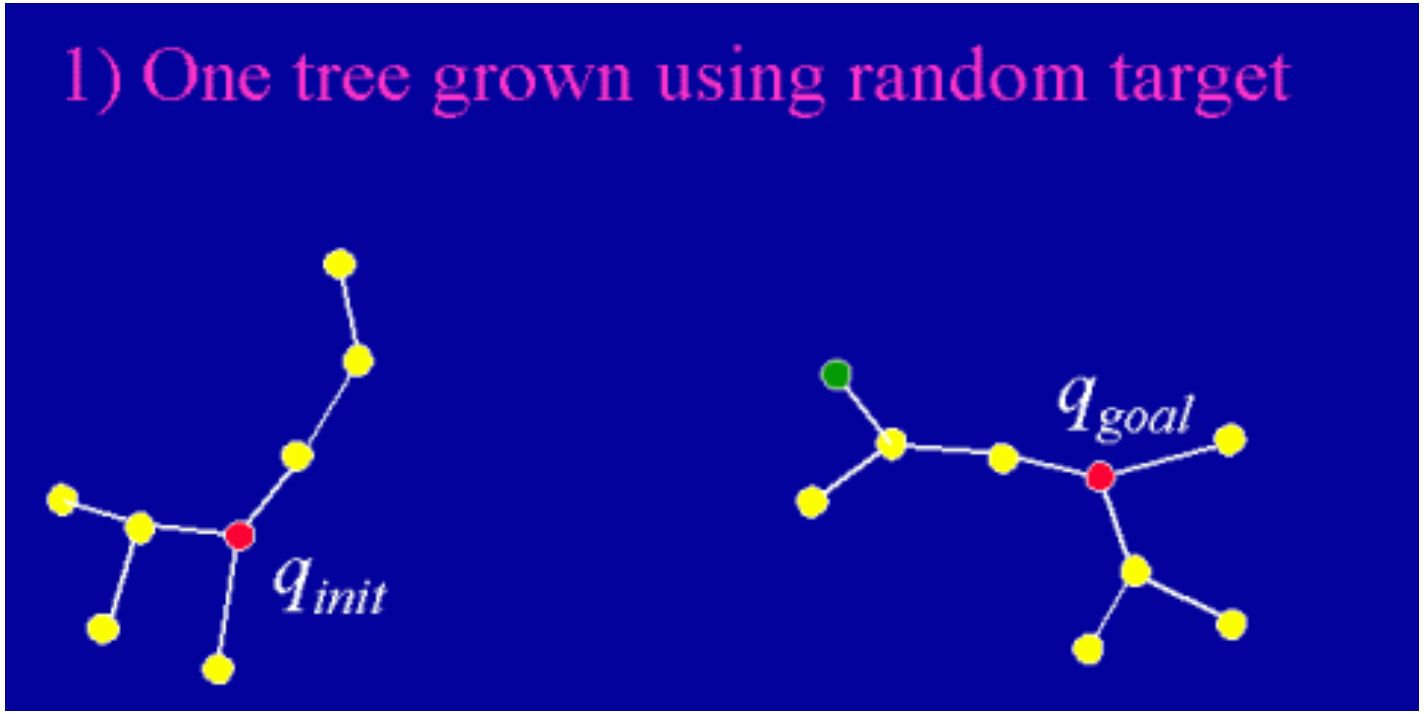
collision encountered

RRT-Connect animation



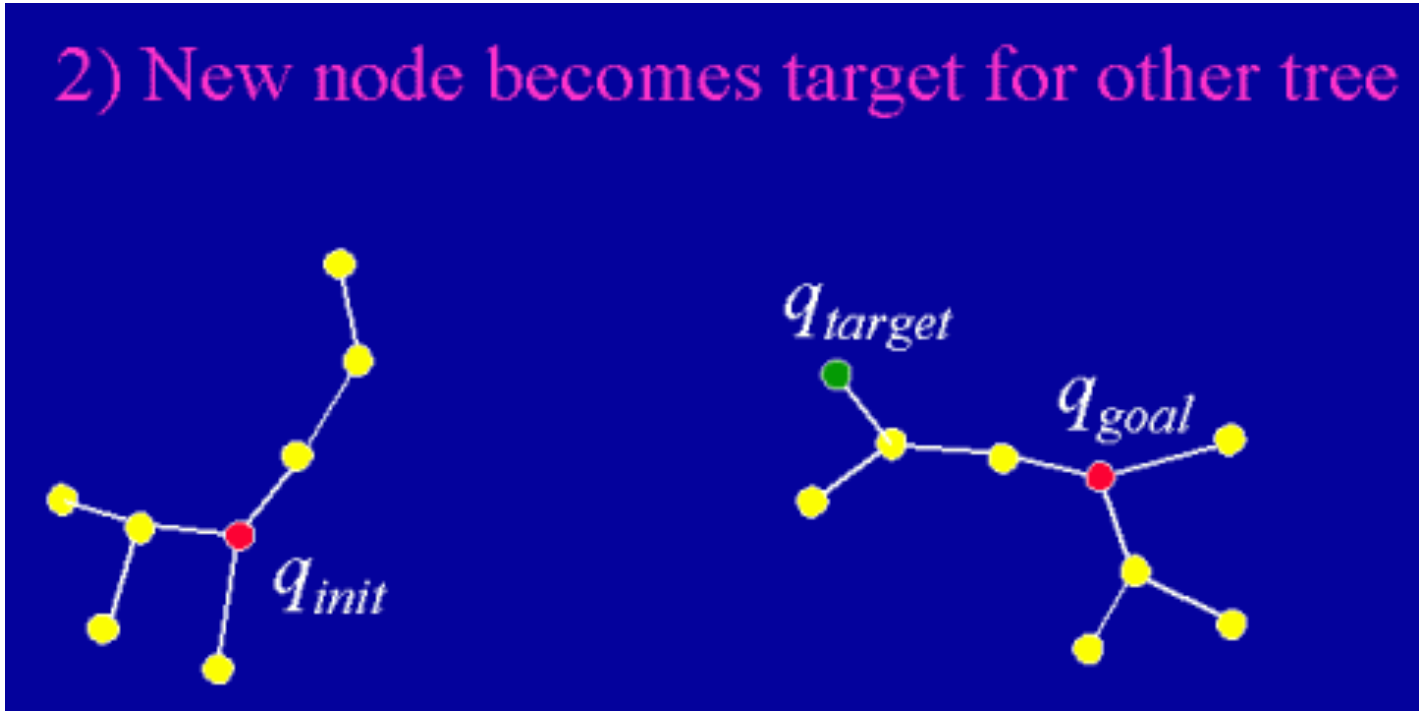
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



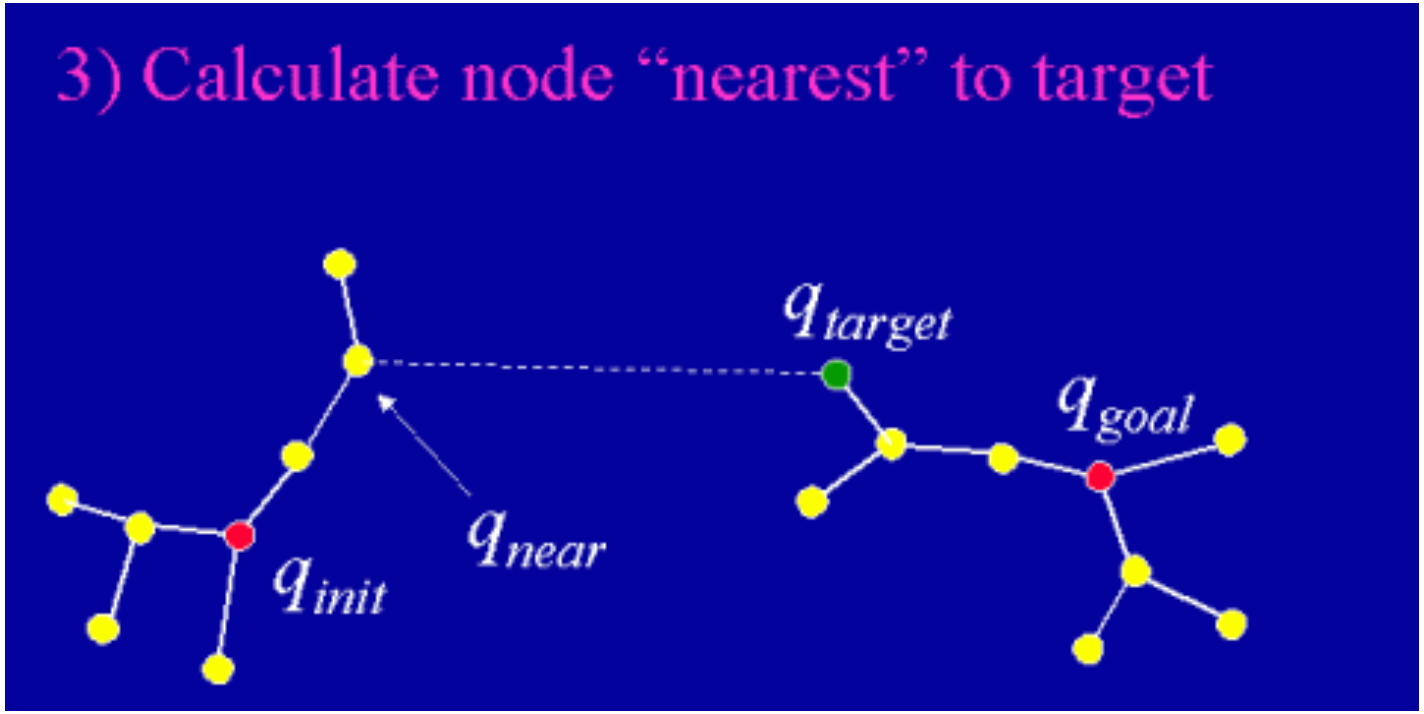
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



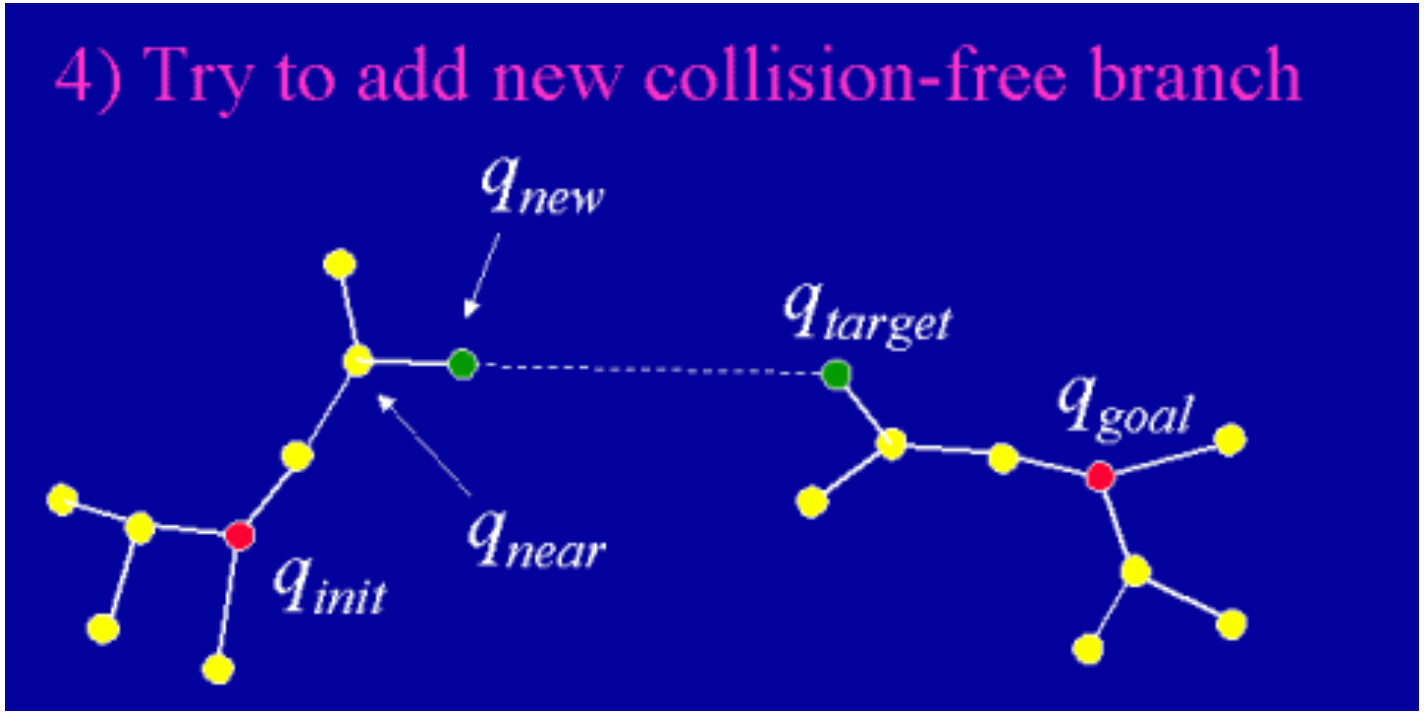
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



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

RRT-Connect animation

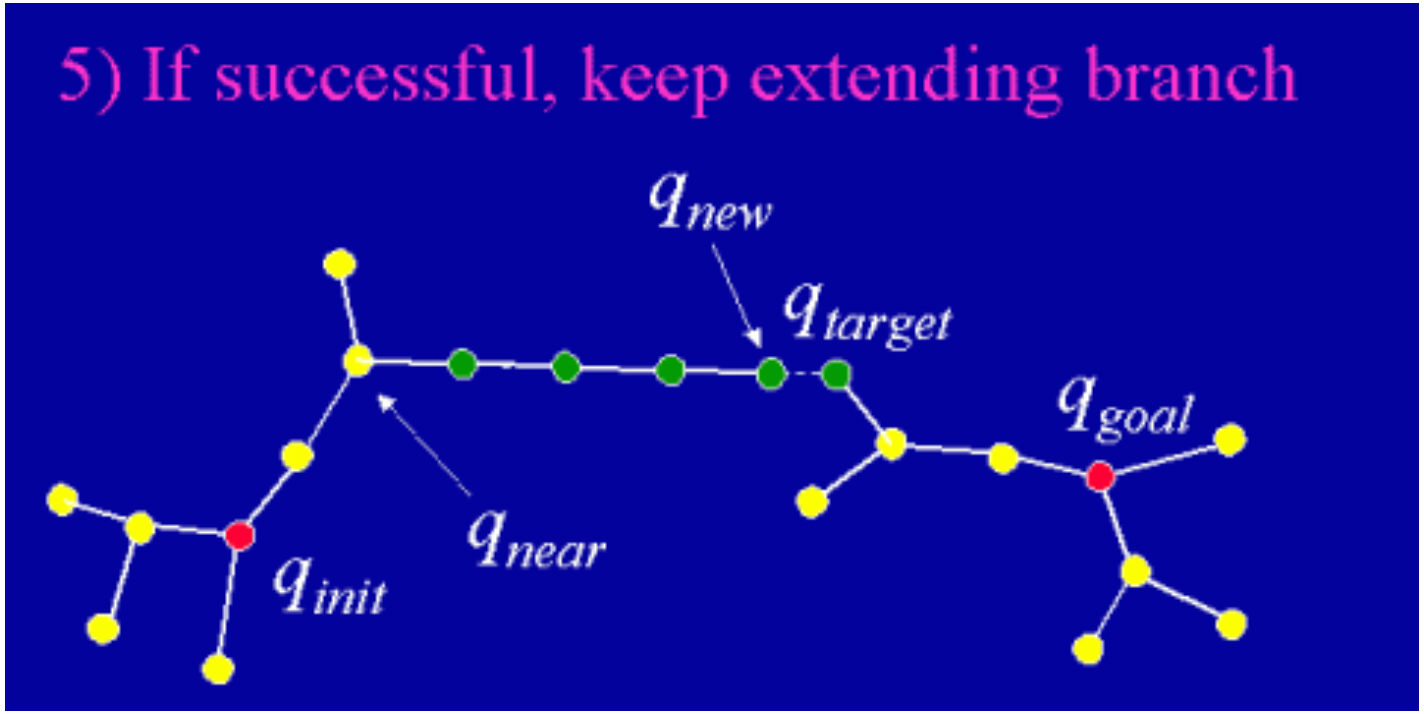


http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation

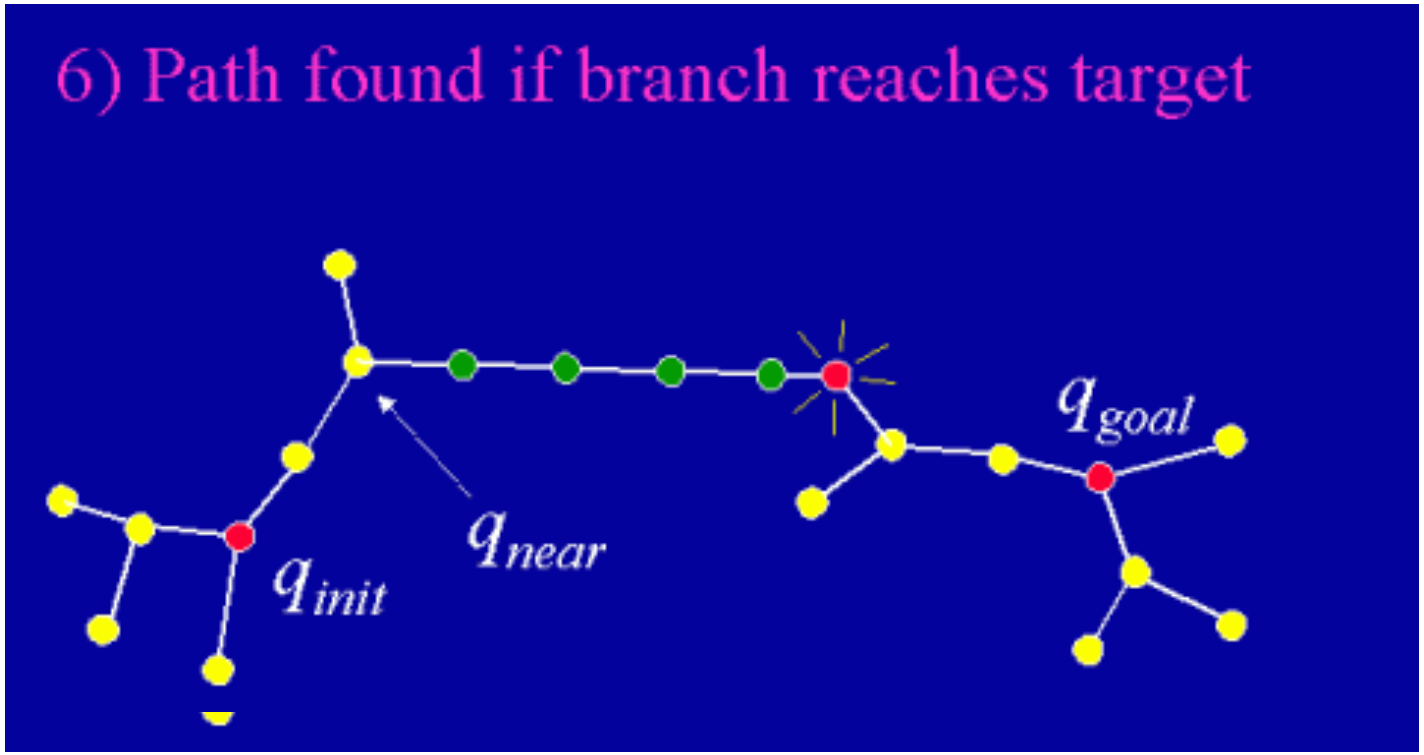
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



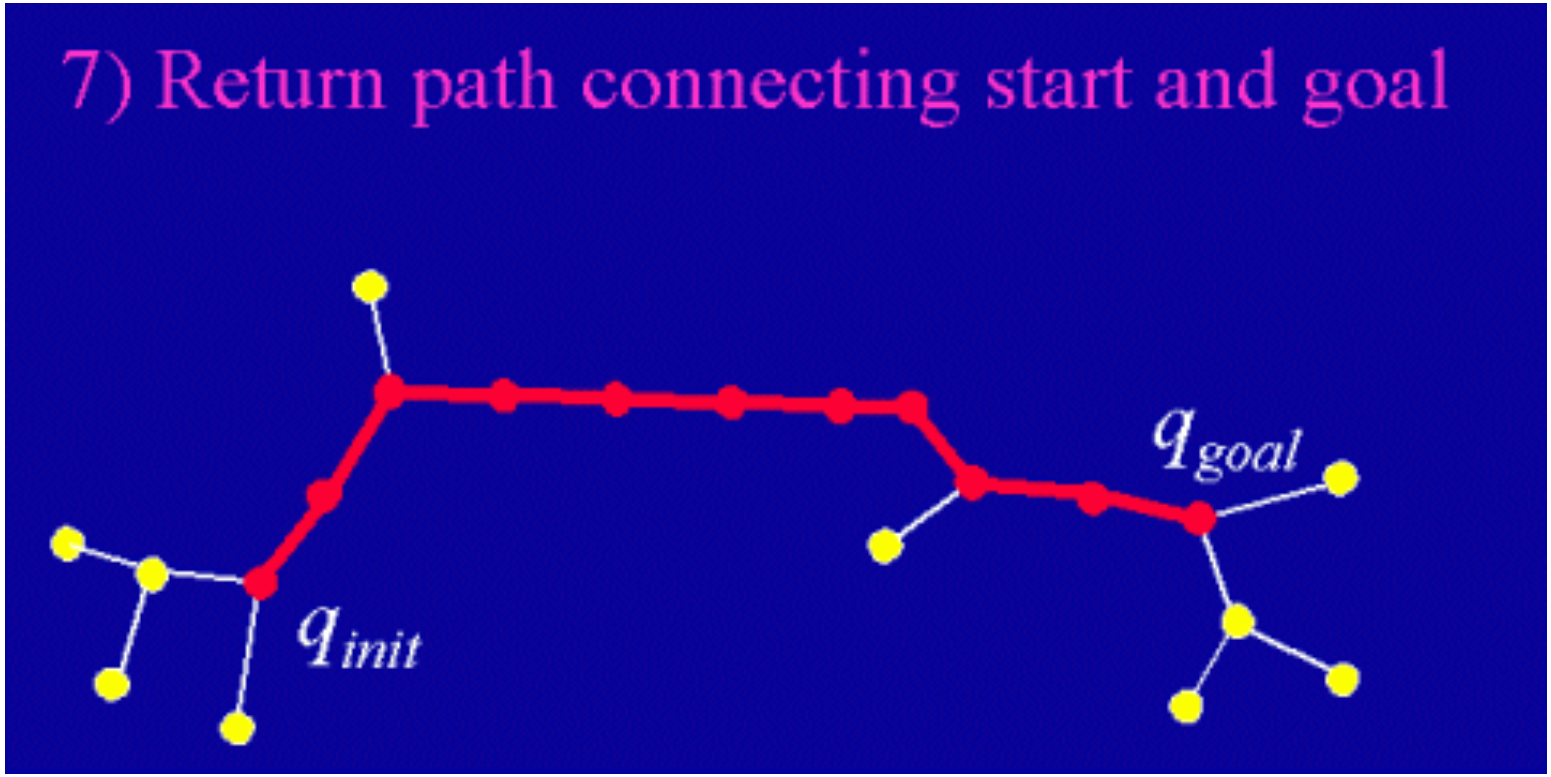
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



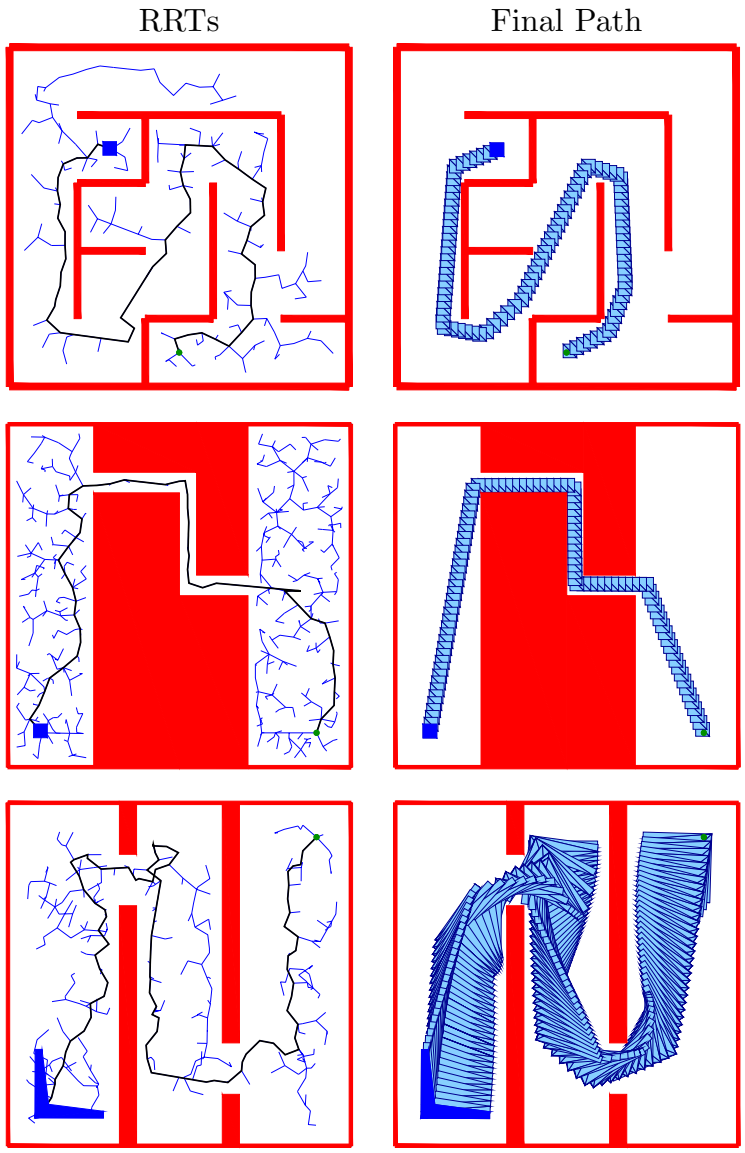
http://www.kuffner.org/james/plan/algorithm.php

RRT-Connect animation



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

Examples of RRT-Connect

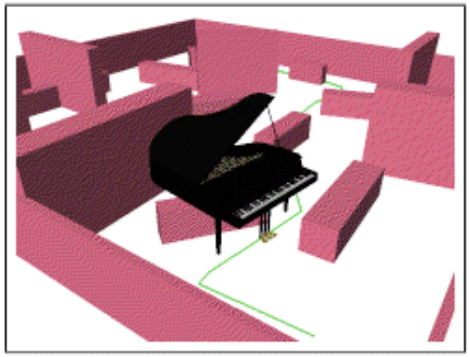


2 DOF maze

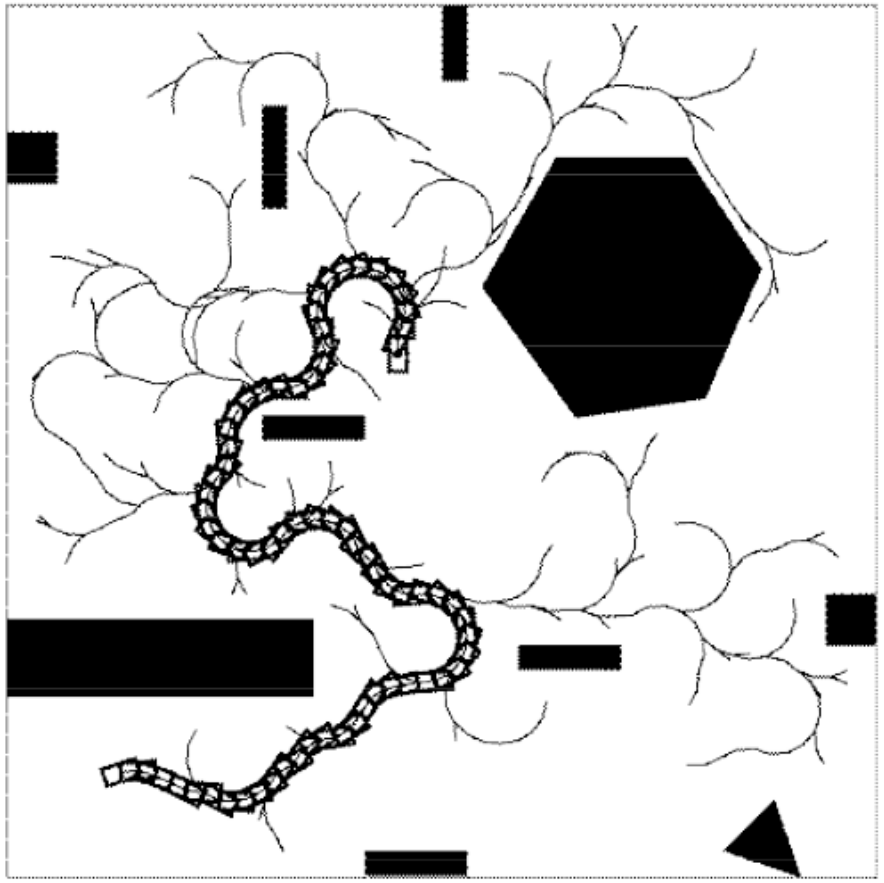
2 DOF single passway

3 DOF single passway
(with non-point geometry)

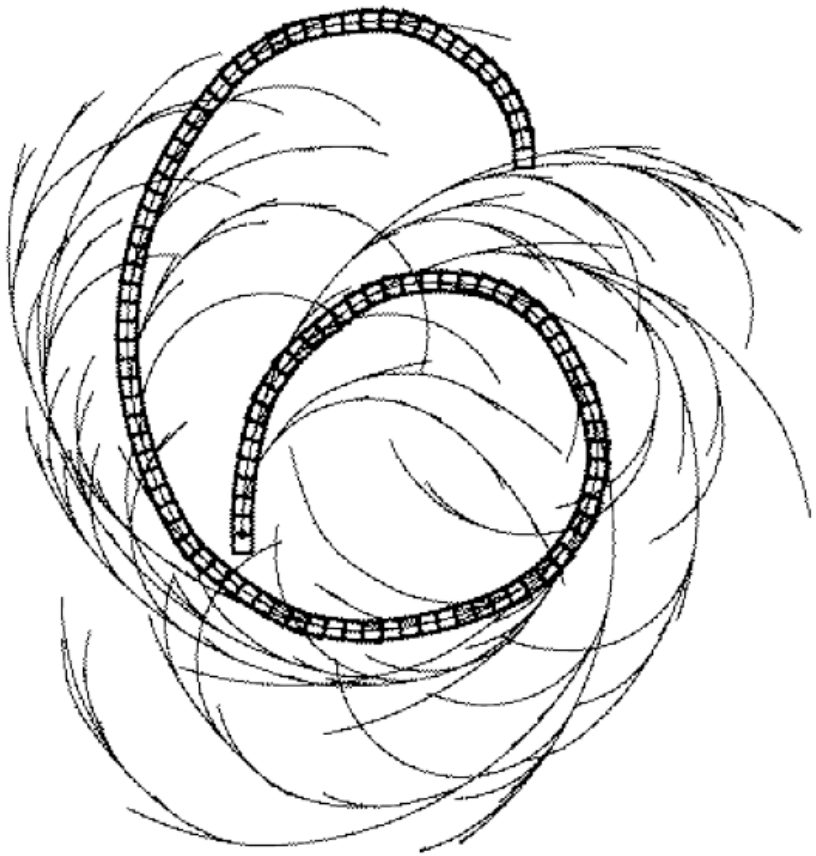
Moving a Piano



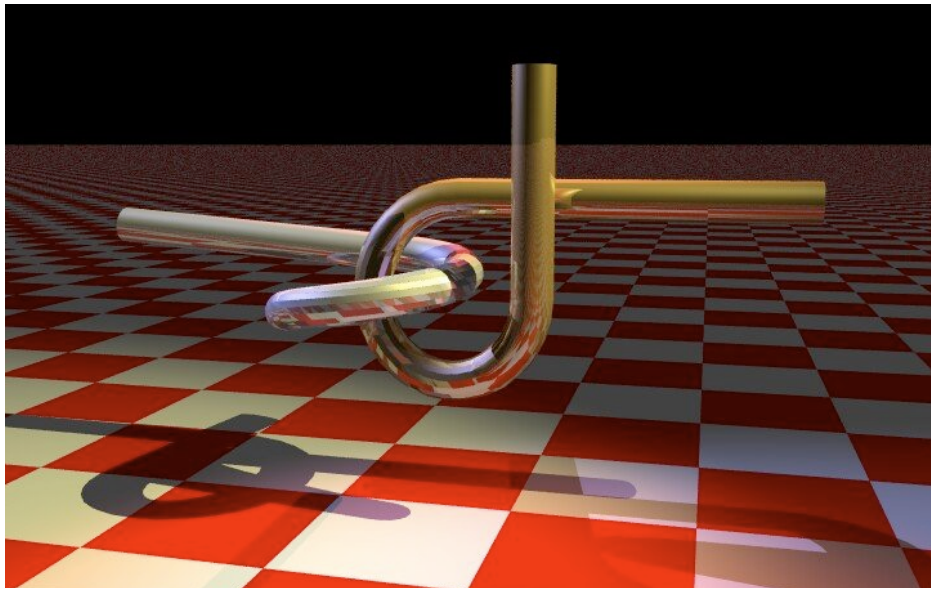
A Car-Like Robot

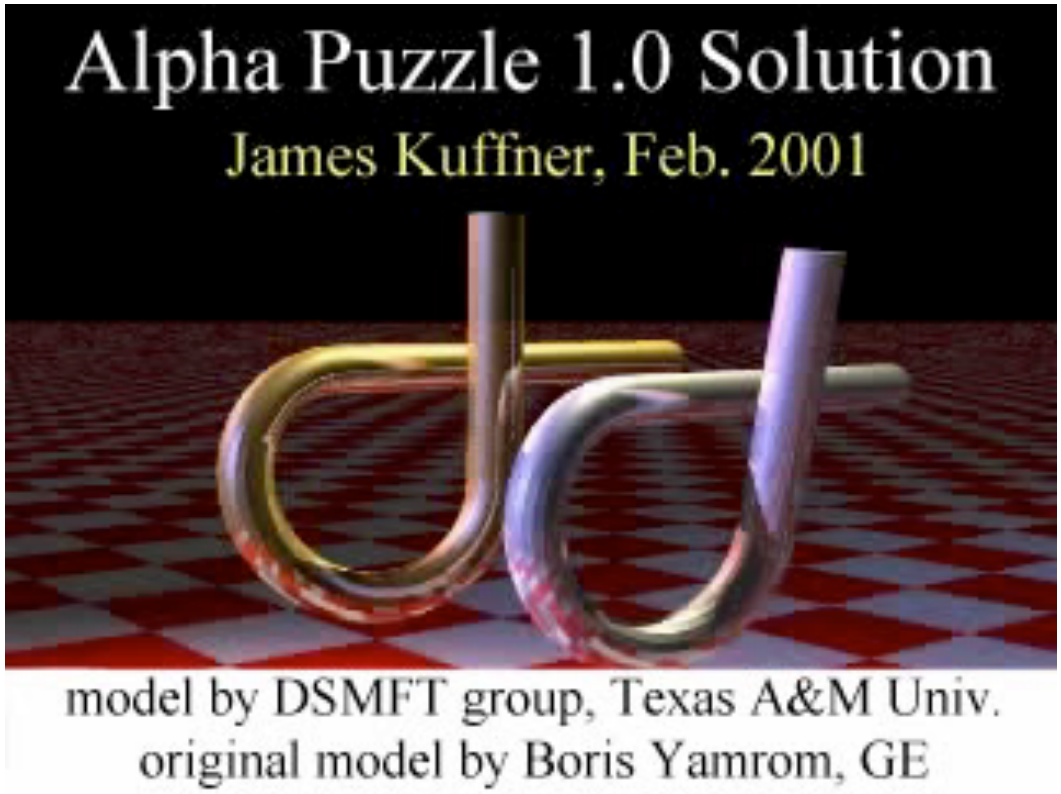


A Right-Turn Only Car

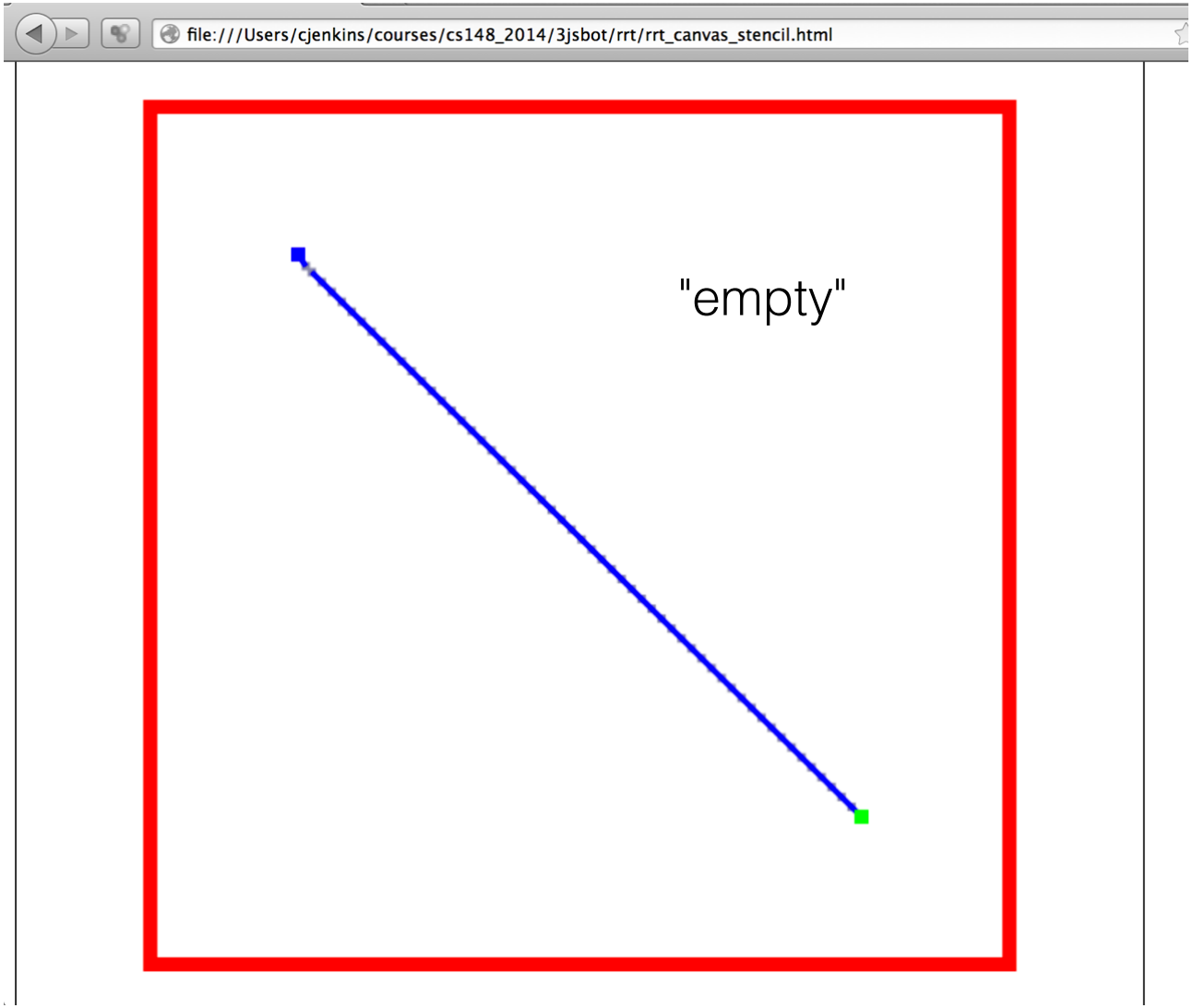


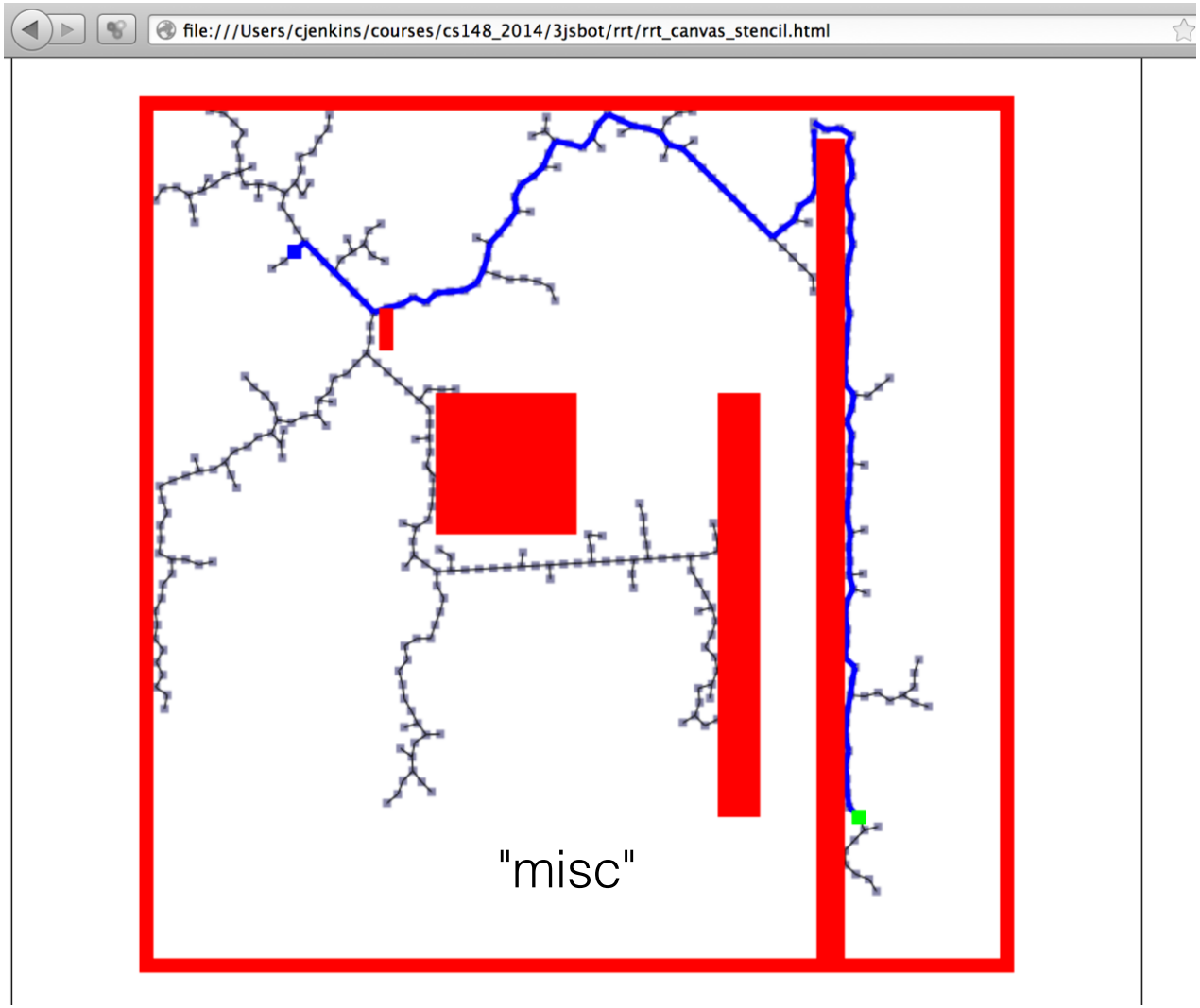
Alpha Puzzle

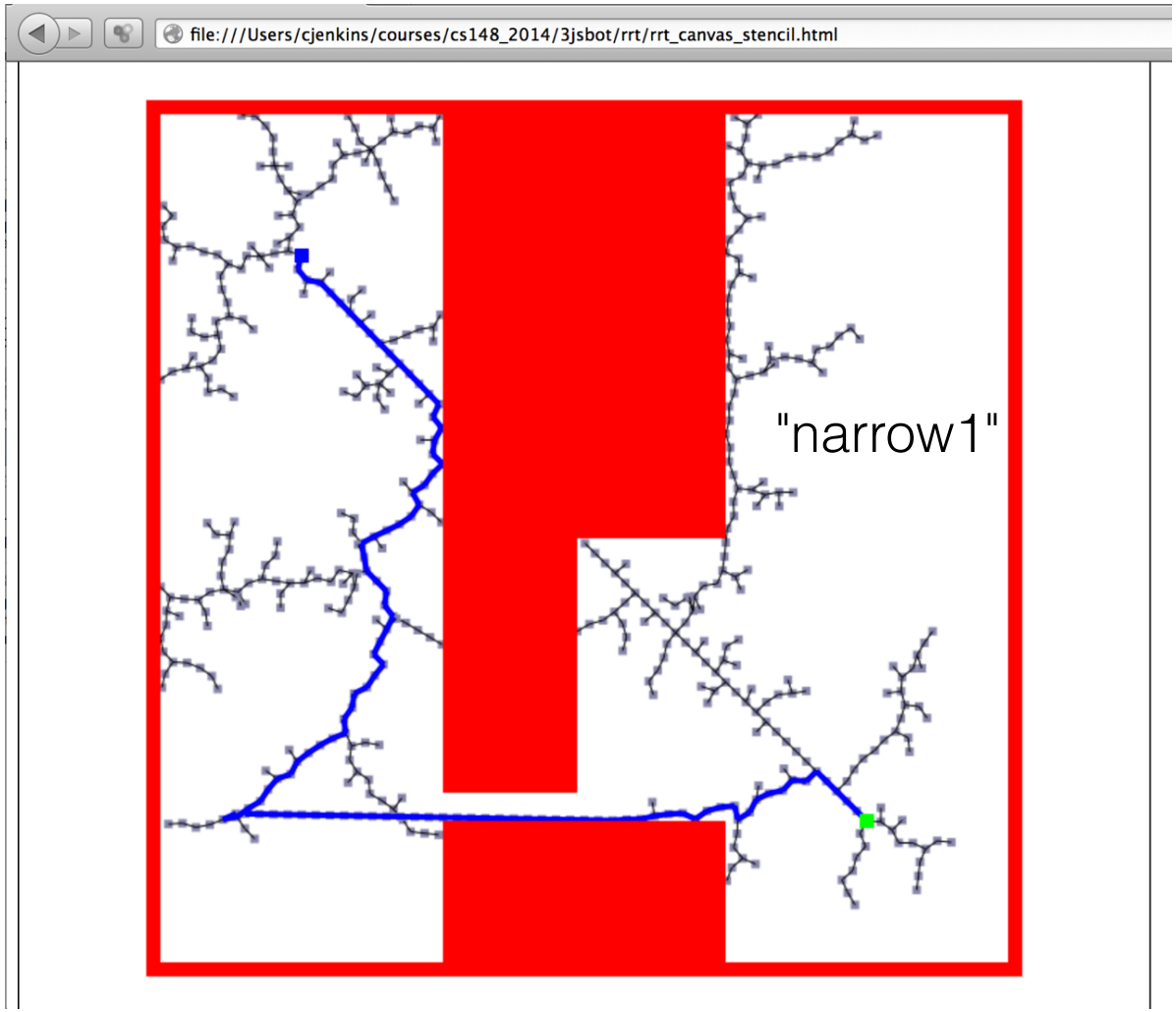


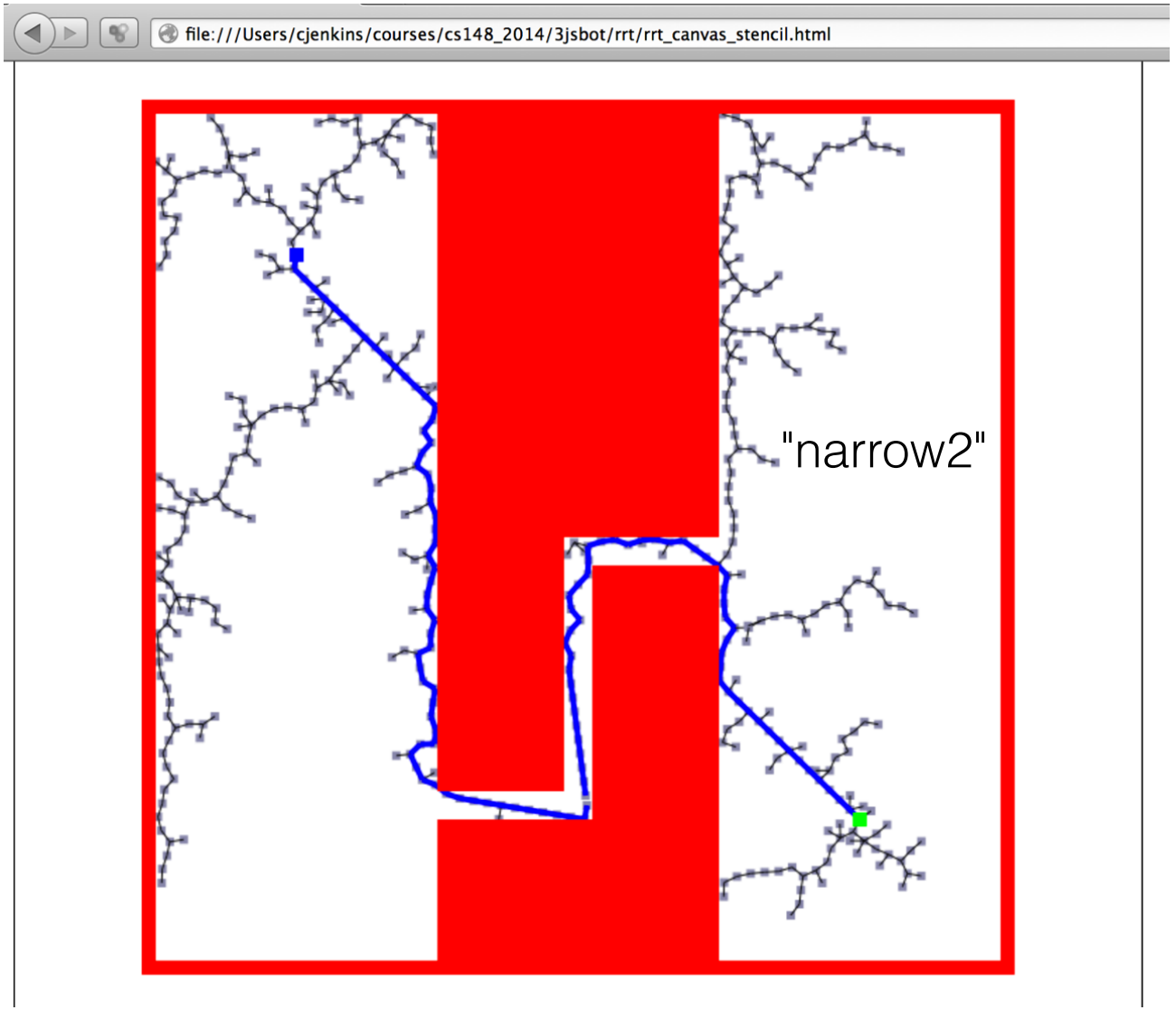


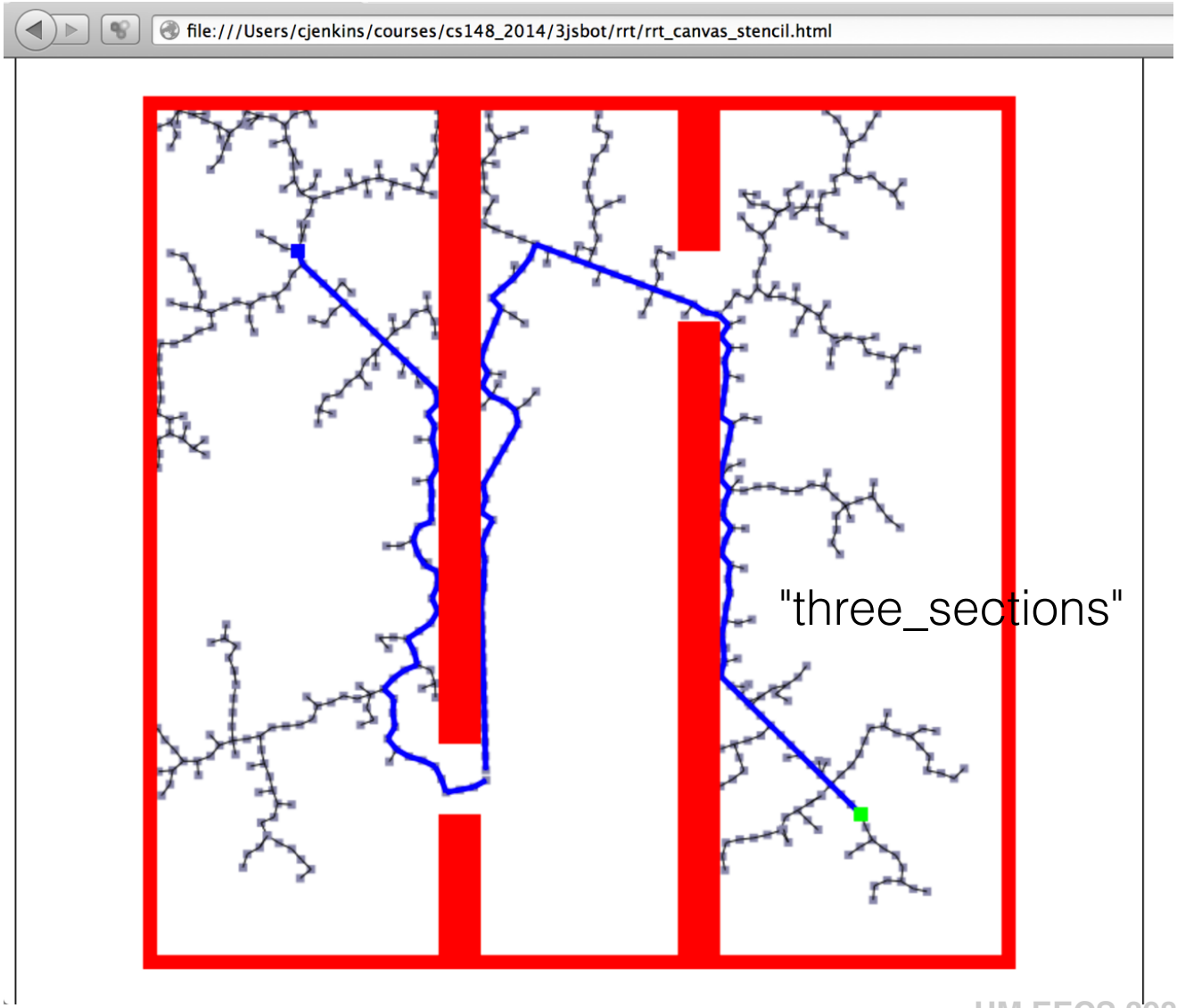
Canvas Stencil Examples


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

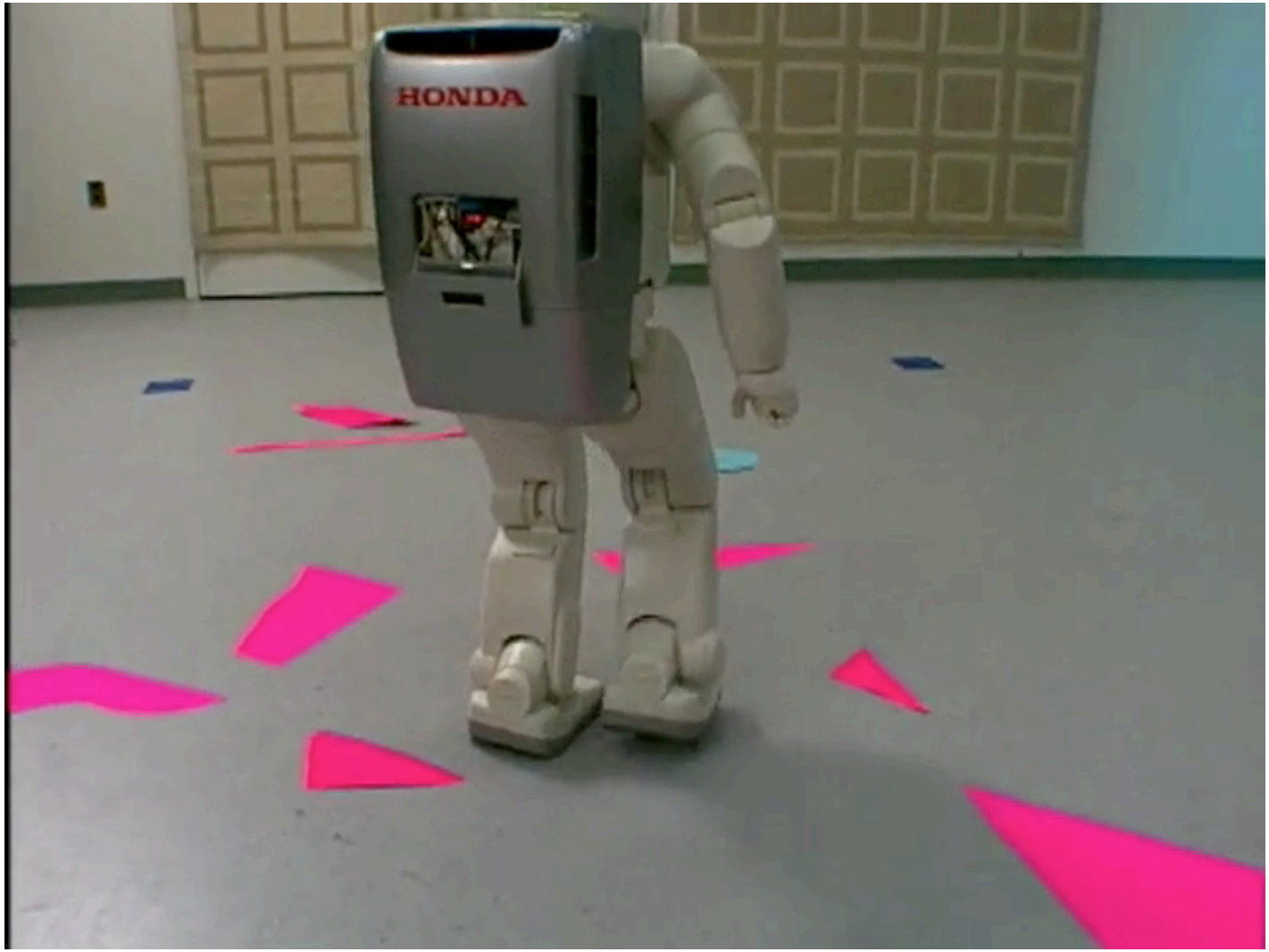

file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"misc"


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"

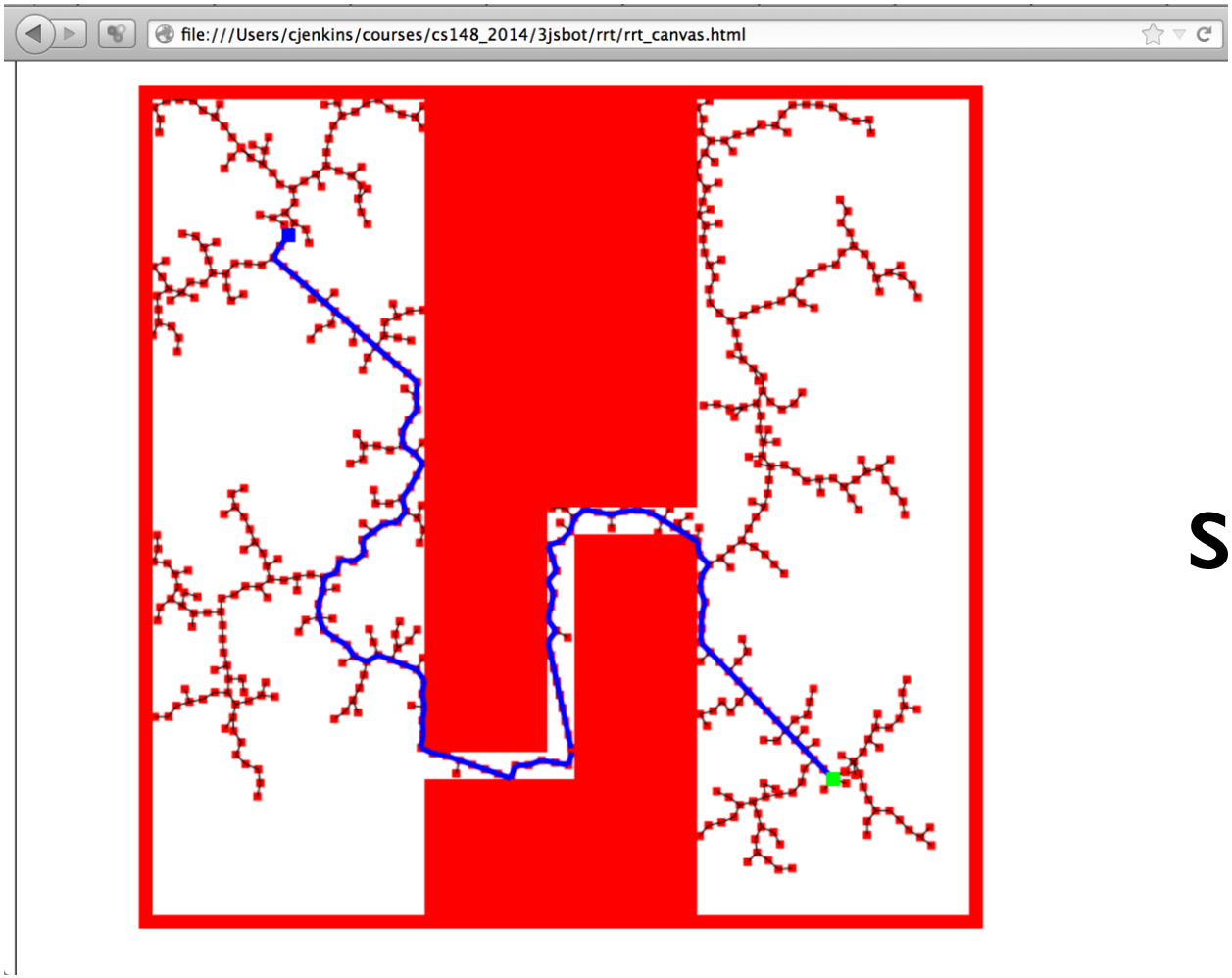

file:///Users/cjenkins/courses/cs148_2014/3jsbot/rrt/rrt_canvas_stencil.html
"three_sections"

“We’ve made robot history”



Kuffner/Asimo Discovery Channel feature - http://vimeo.com/9911572

RRTs can take a lot of time...



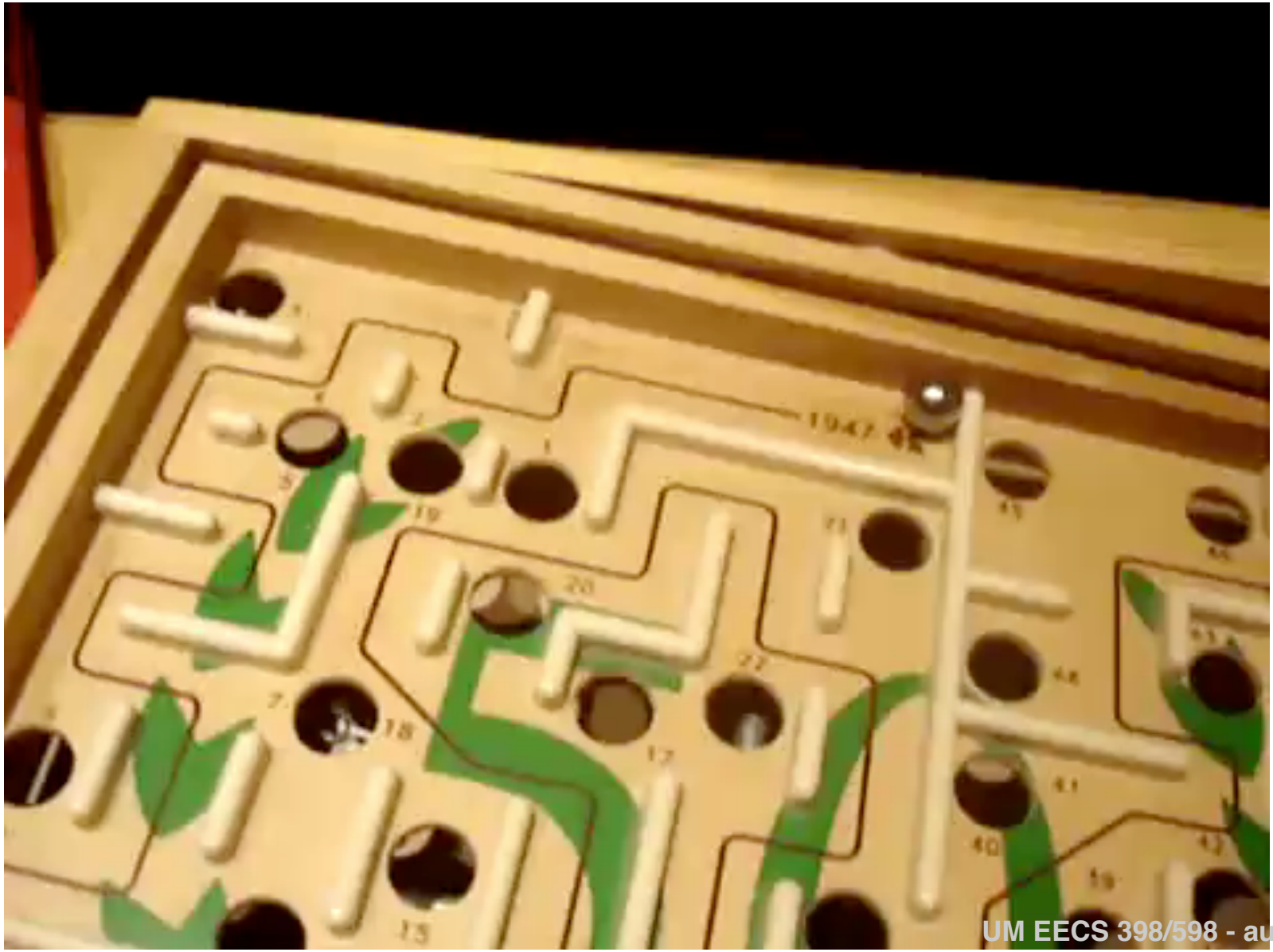
Is there a simpler way?

Potential fields

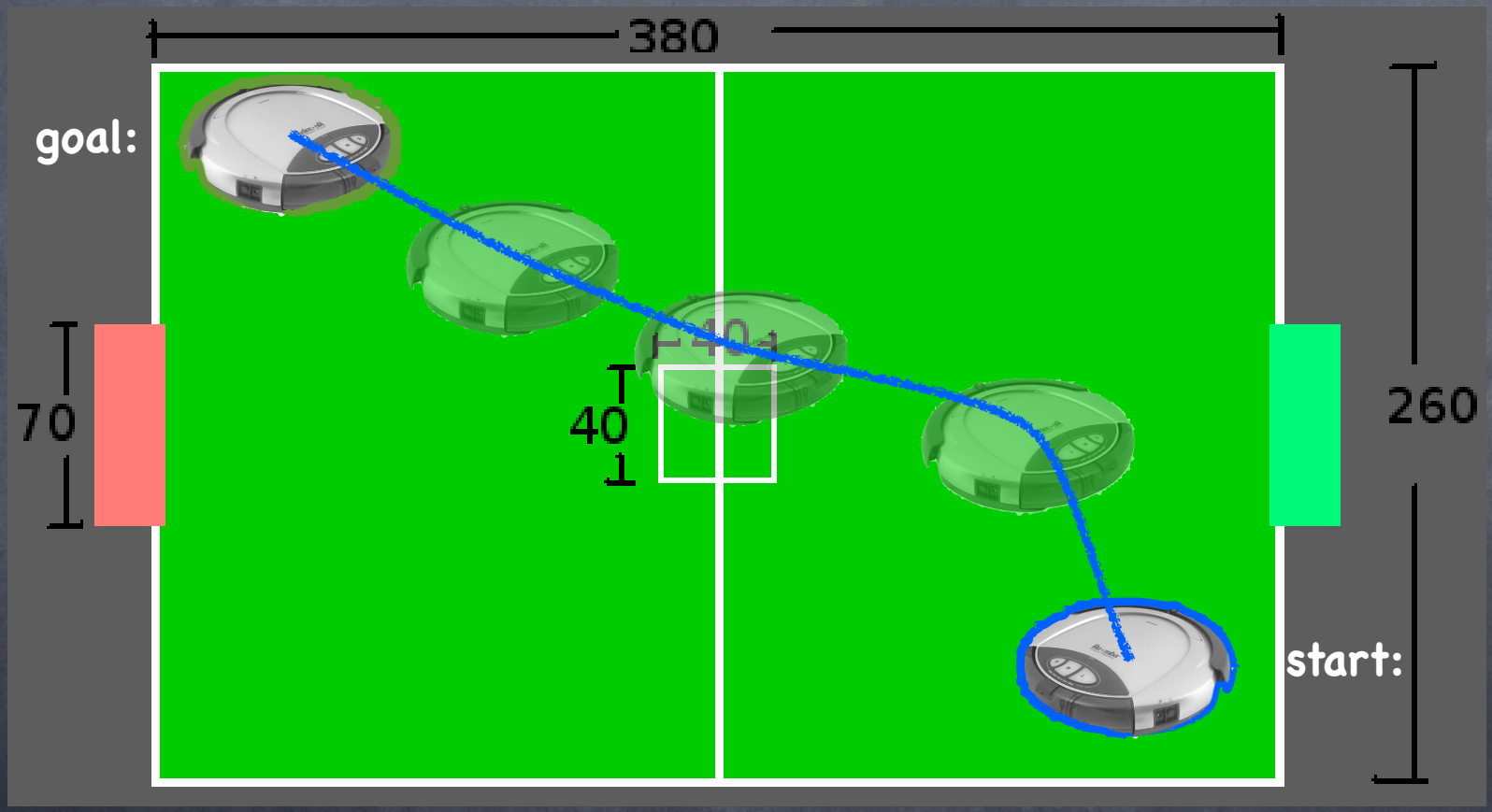
follow your potential

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**

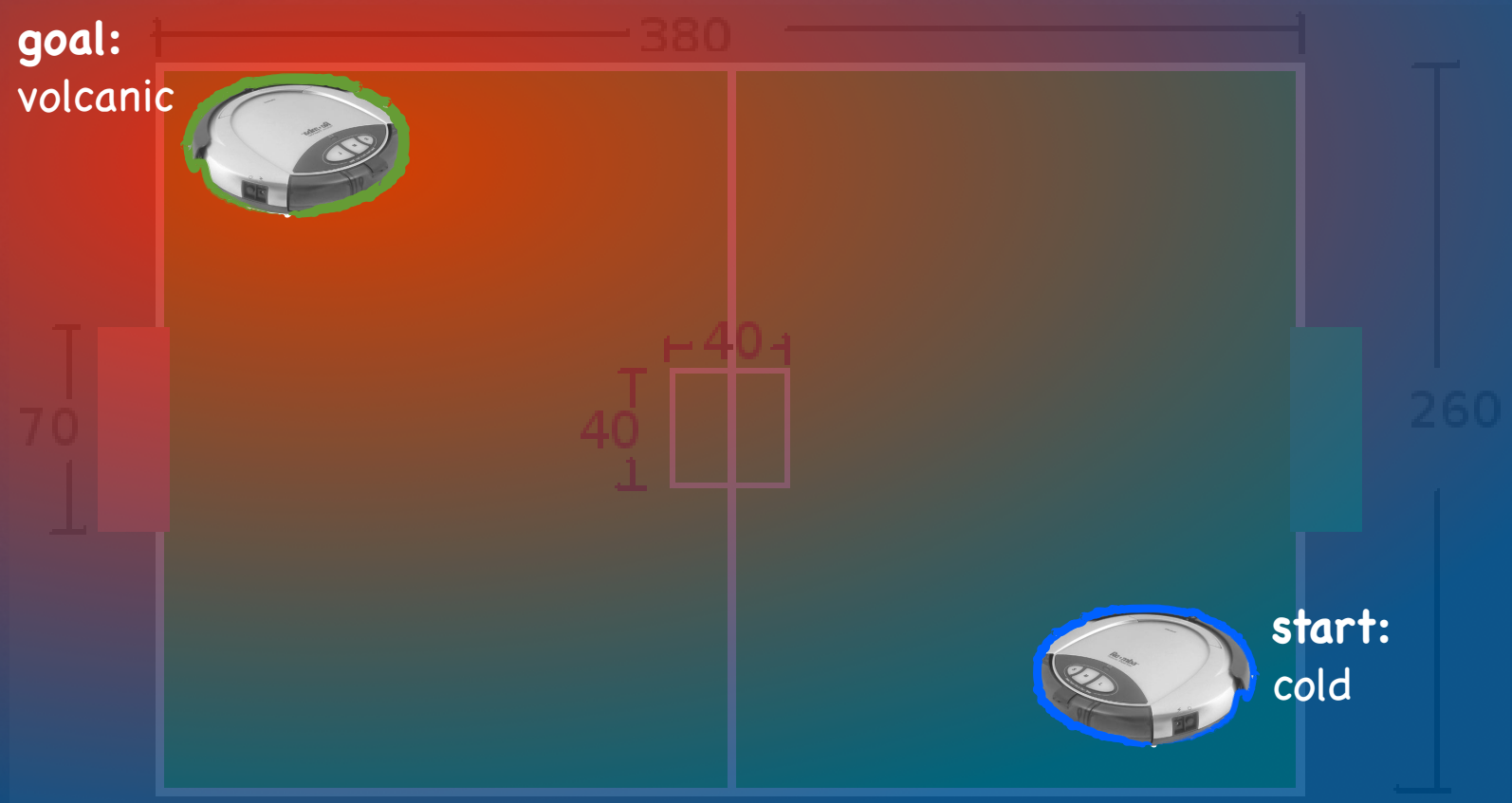

1947
UM EECS 398/598 - autorob.github.io

Navigation (again)



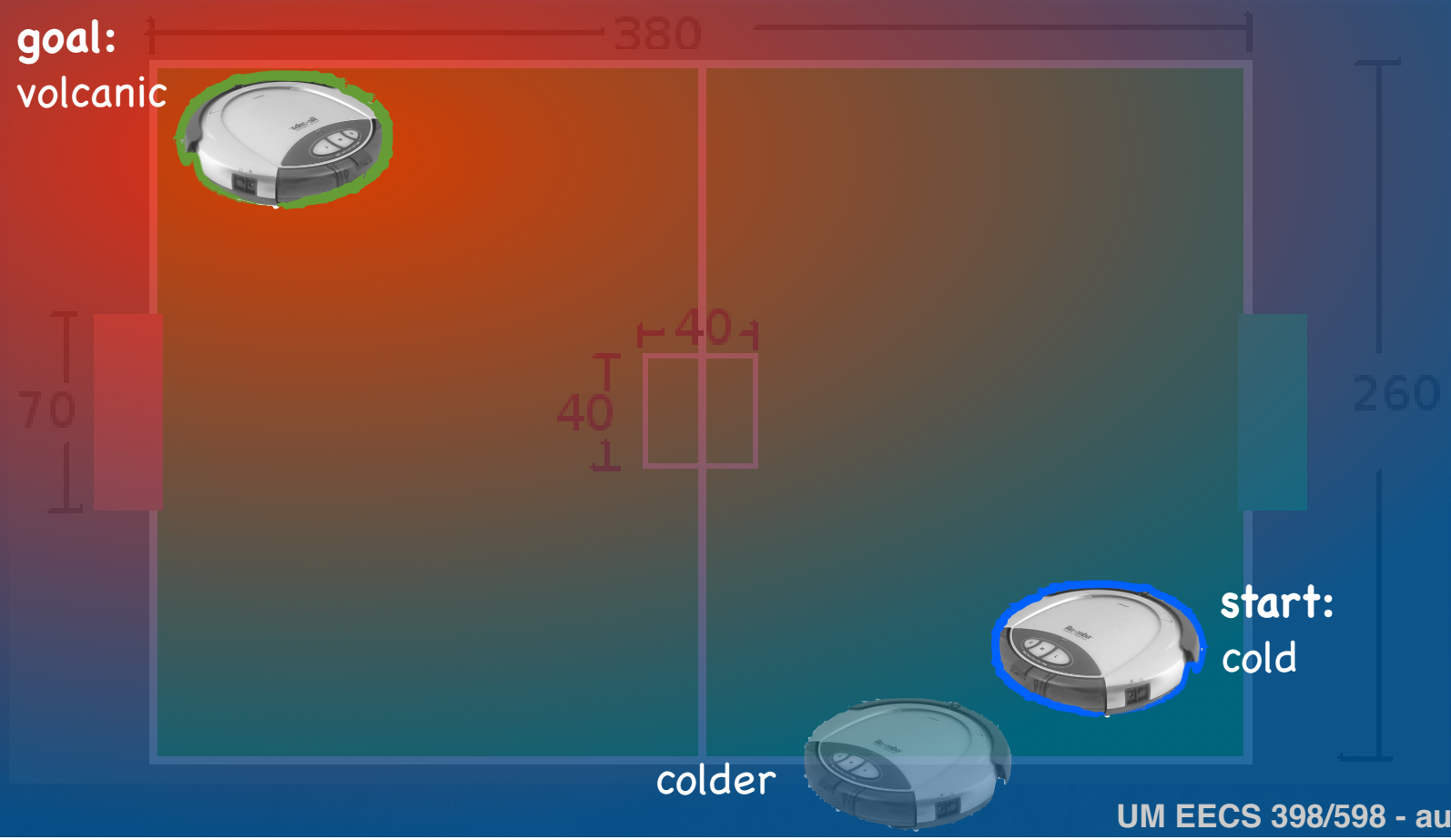
Potential field

(like a game of "warmer-colder")



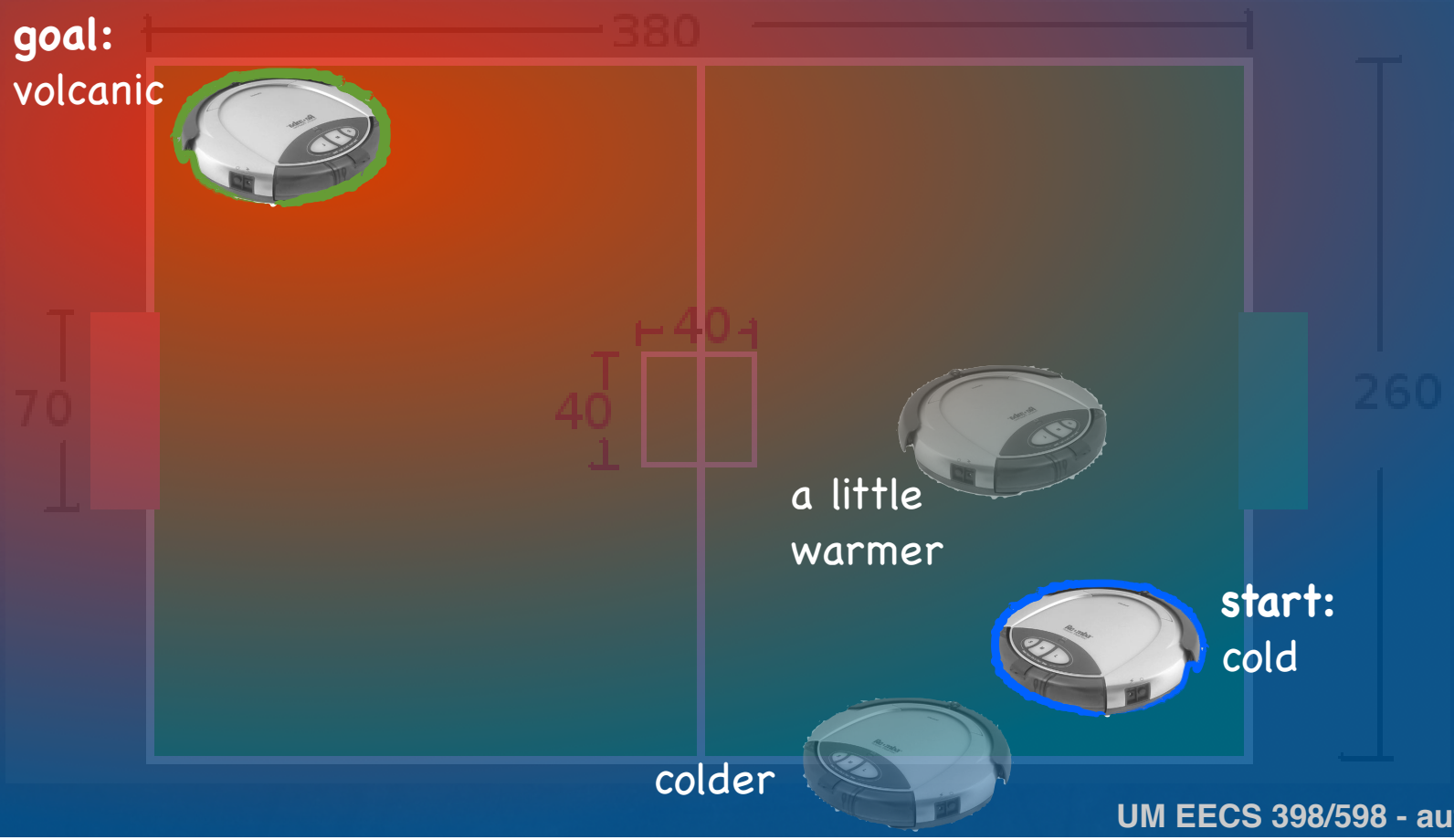
Potential field

(like a game of "warmer-colder")



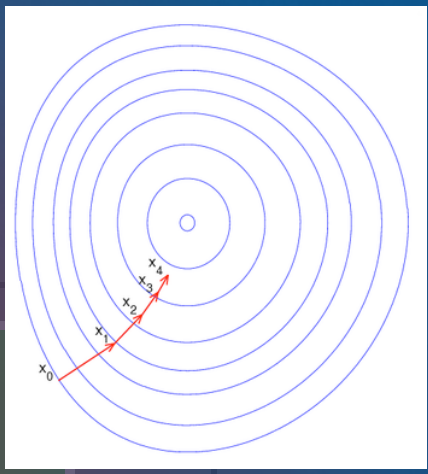
Potential field

(like a game of "warmer-colder")



Potential field

(like a game of "warmer-colder")



Gradient descent:
Energy potential converges at goal

goal:
volcanic

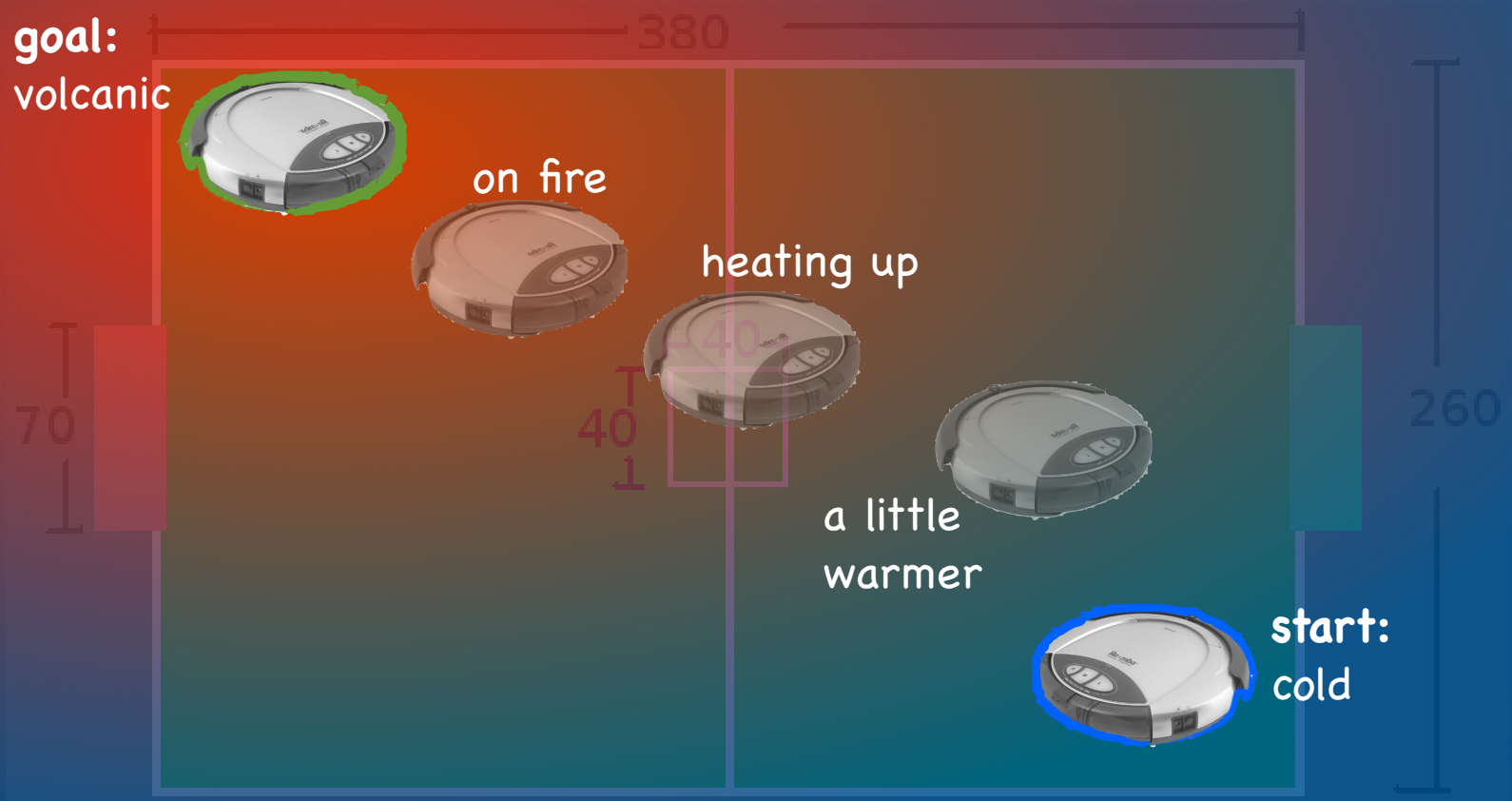
a little warmer

start:
cold

colder

Potential field

(like a game of "warmer-colder")



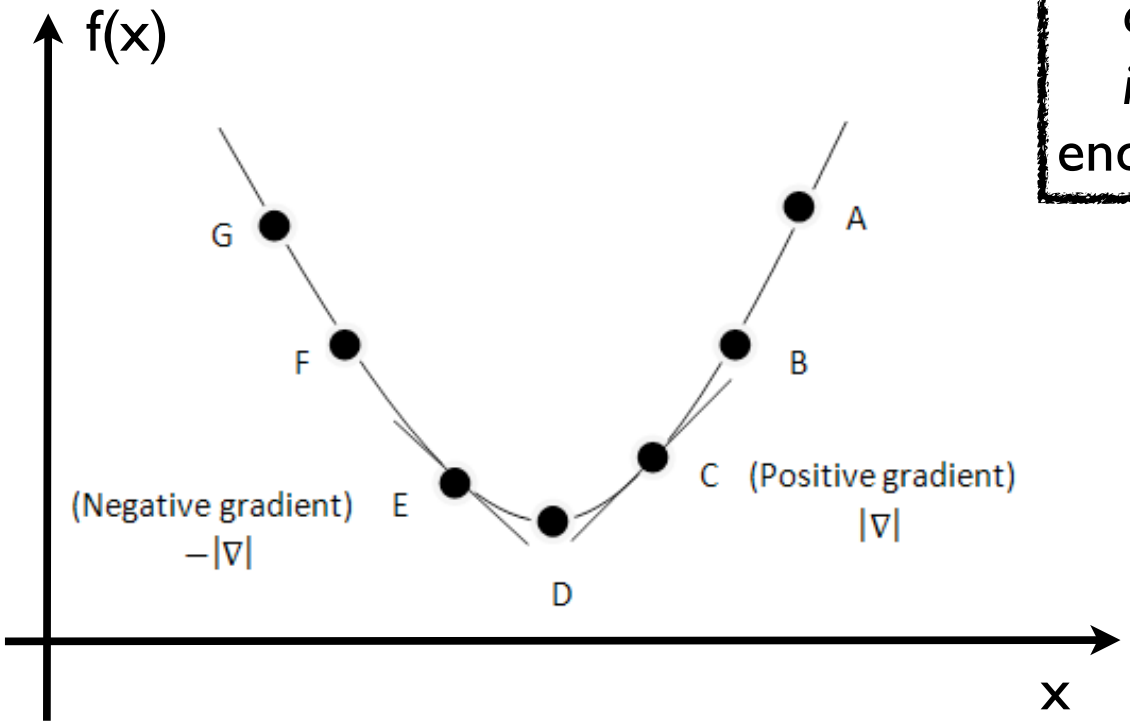
How do we define a potential field?

Potential Field

- A potential field is a differentiable function $U(q)$ that maps configurations to scalar "energy" value
- At any q, gradient $\nabla U(q)$ is the vector that maximally increases U
- At goal q_{goal}, energy is minimized such that $\nabla U(q_{goal}) = 0$
- Navigation by descending field $-\nabla U(q)$ *to goal*

Gradient descent

From Wikipedia, the free encyclopedia



Gradient Descent Algorithm:

$q_{path}[0] \leftarrow q_{start}$

$i \leftarrow 0$

while $(||\nabla U(q[i])|| > \varepsilon)$

  $q_{path}[i+1] \leftarrow q_{path}[i] - \alpha \nabla U(q_{path}[i])$

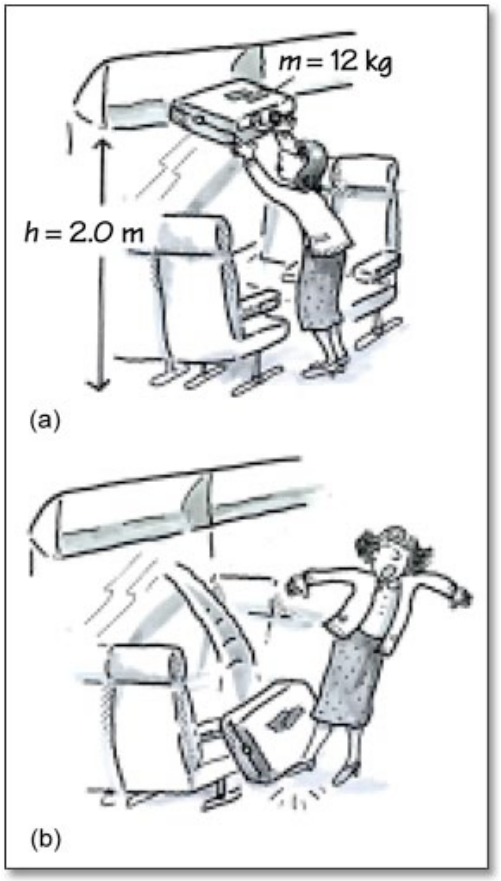
  $i \leftarrow i+1$

end

Derivative assumed to be direction of steepest ascent away from goal

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n)$$

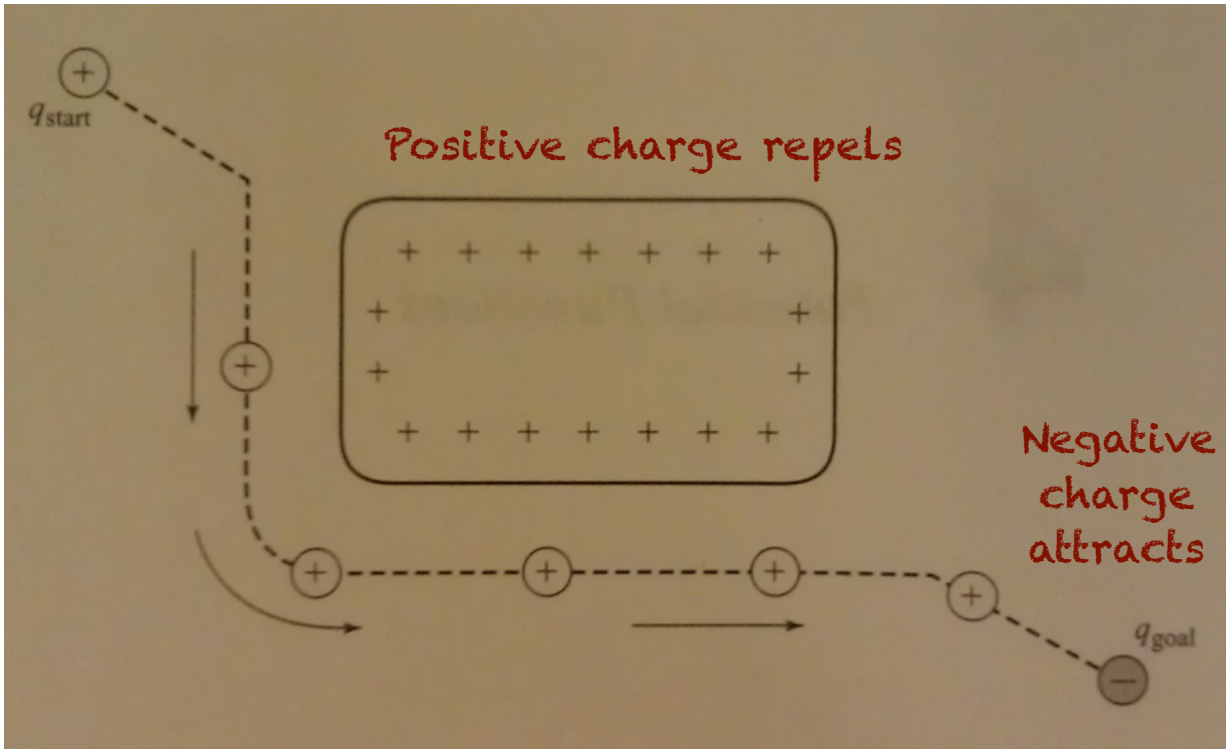
Potential Energy



- Energy stored in a physical system
- Kinetic motion caused by system moving to lower energy state
- For objects acting only w.r.t. gravity
- potential_energy = mass*height*gravity

Charged Particle Example

Positively charged particle follows potential energy to goal



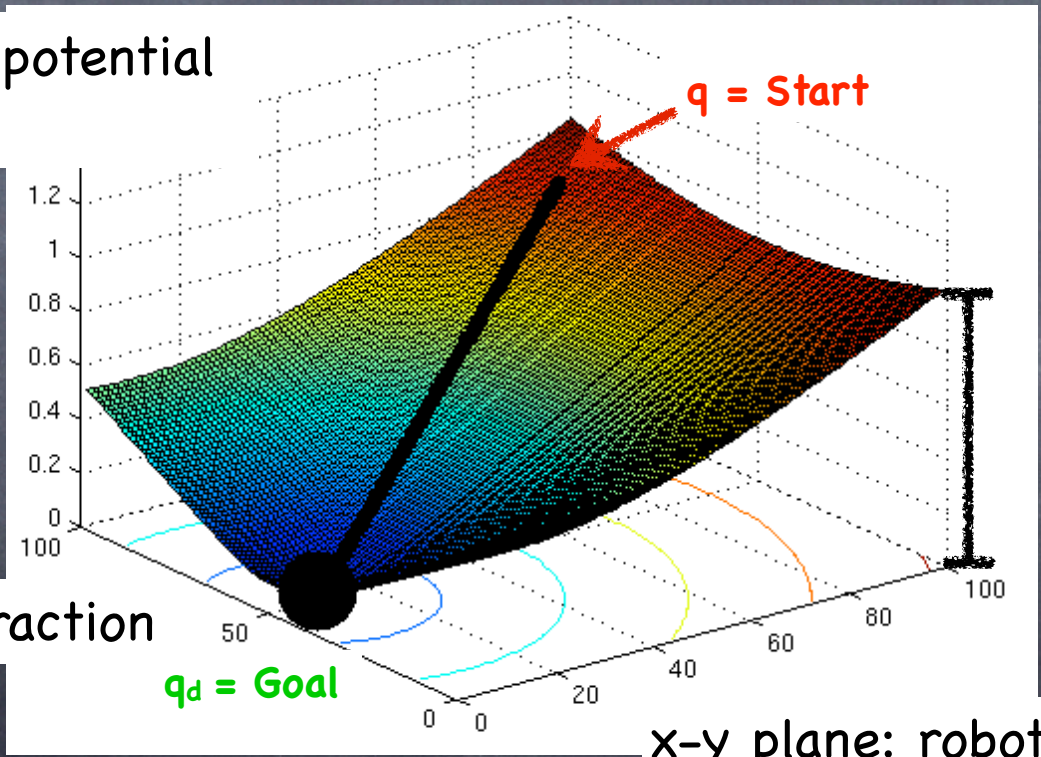
Convergent Potentials

let's call these "attractor landscapes"

basin of attraction

2D potential navigation

z: height indicates potential at location



basin of attraction

x-y plane: robot position

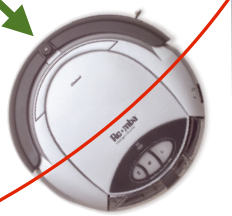
"Cone" Attractor

q_d = Goal

"Attractor"

$\nabla U(q) = w(q - q_d)/\|q - q_d\|$

q = Start



top view

w: weight

$(q - q_d)$: direction

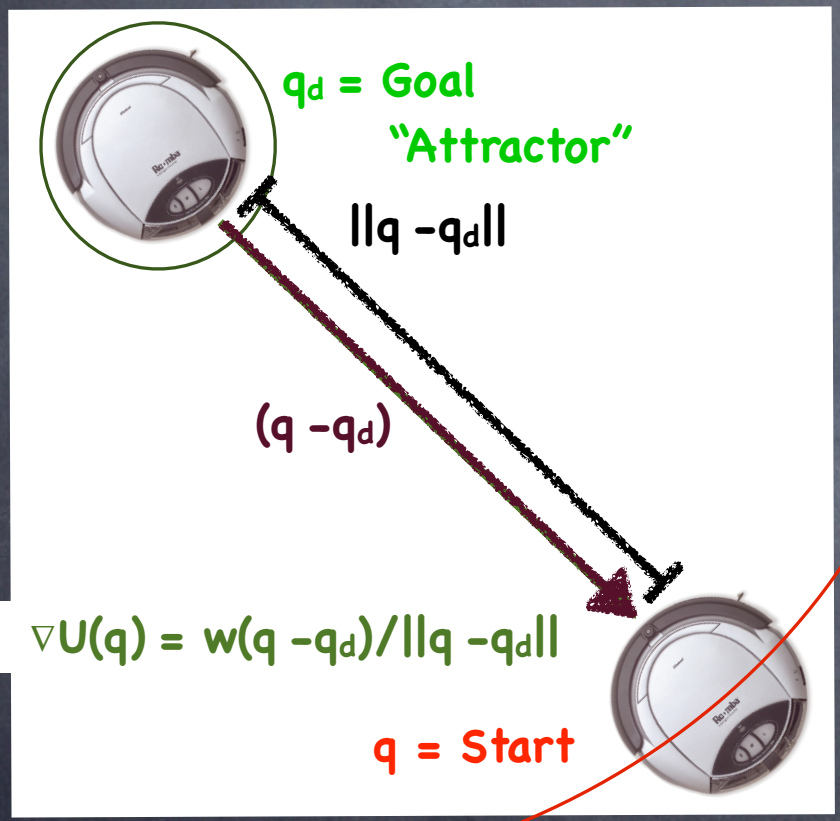
$\|q - q_d\|$: distance

Start

Goal

side view

"Cone" Attractor

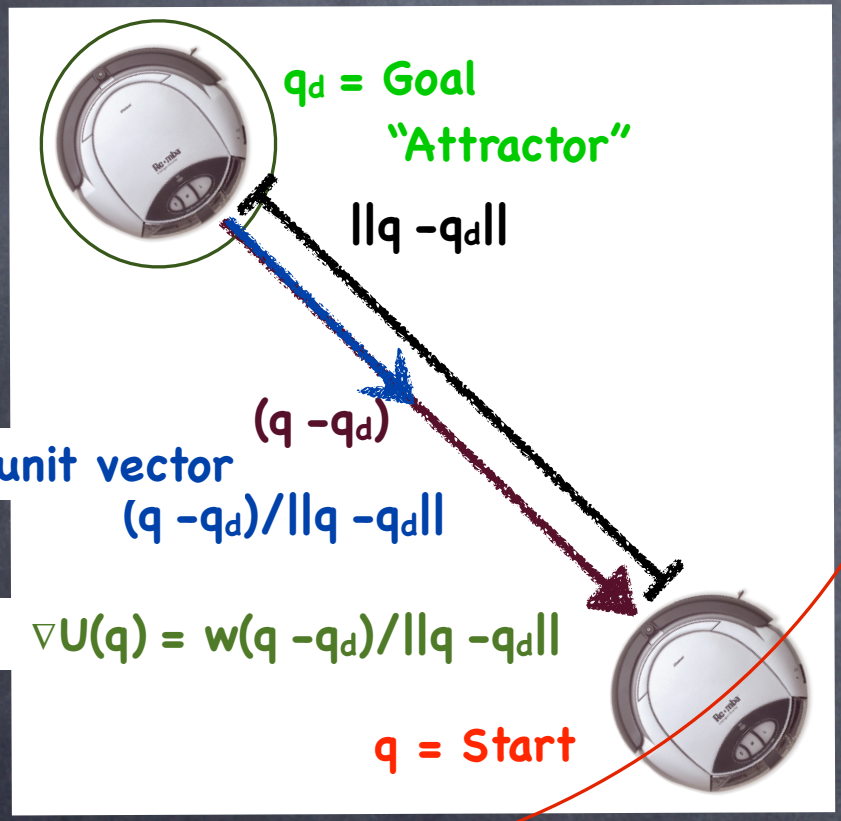


top view

w: weight
$(q - q_d)$: direction
$\|q - q_d\|$: distance

side view

"Cone" Attractor

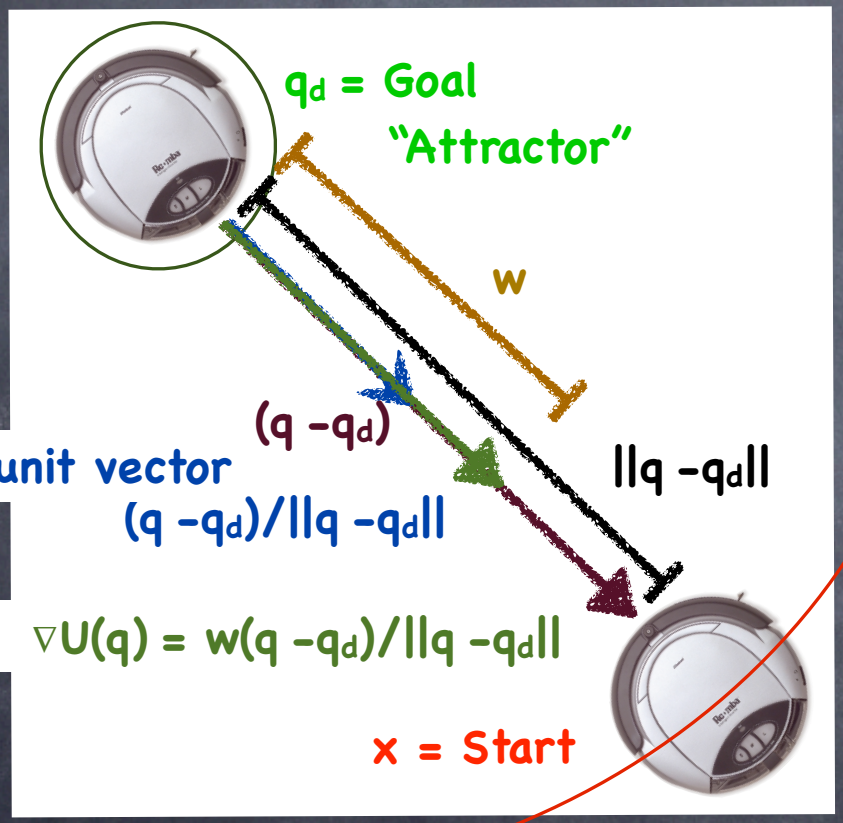


top view

w: weight
$(q_d - q)$: direction
$\|q_d - q\|$: distance

side view

"Cone" Attractor



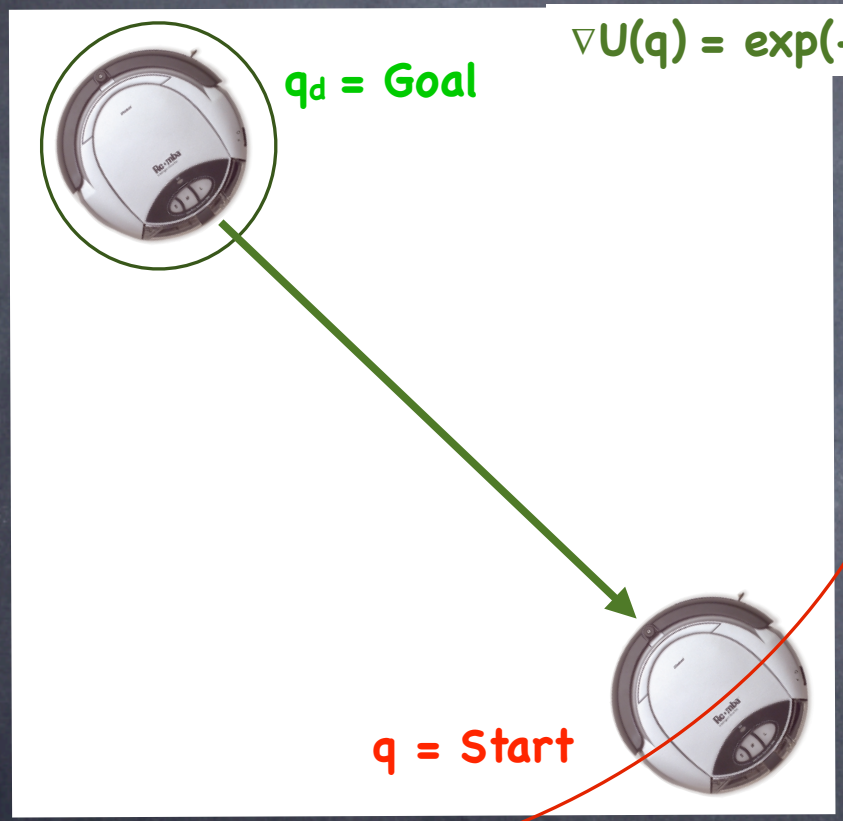
top view

w: weight (< 1)
$(q - q_d)$: direction
$\|q - q_d\|$: distance

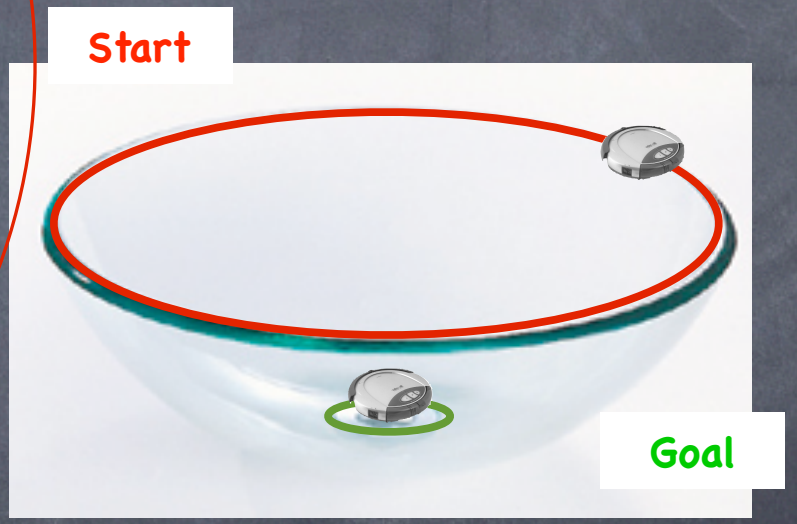
side view

"Bowl" Attractor

$\nabla U(q) = \exp(-\|q - q_d\|/w)\,(q - q_d)$



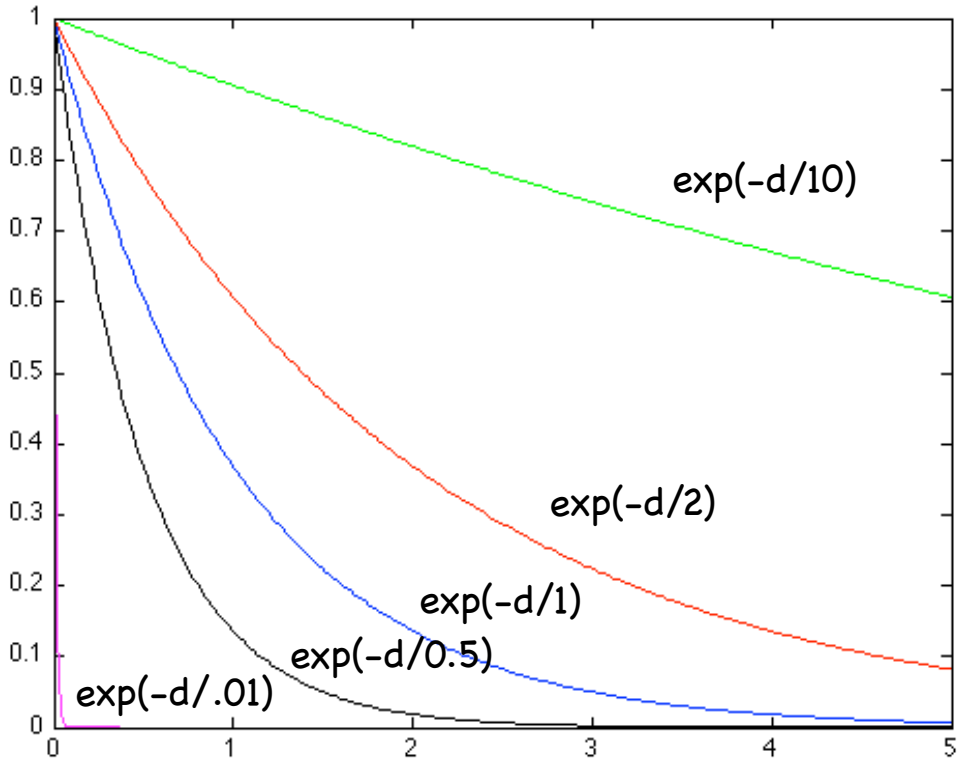
top view



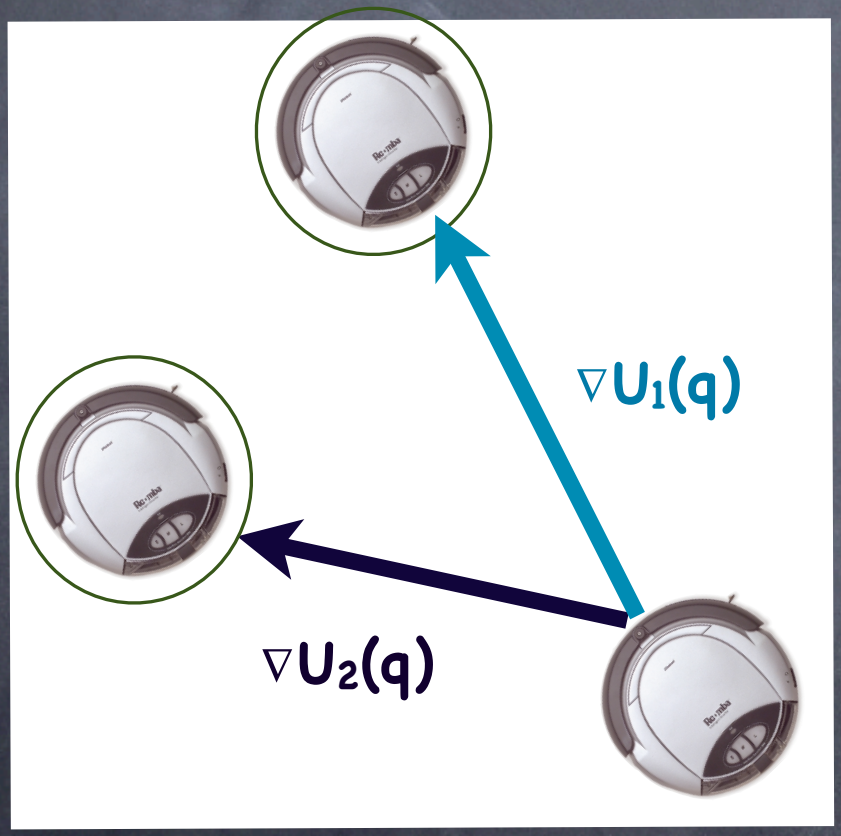
side view

exp(-d/w)

exp(-d/w) : influence of potential

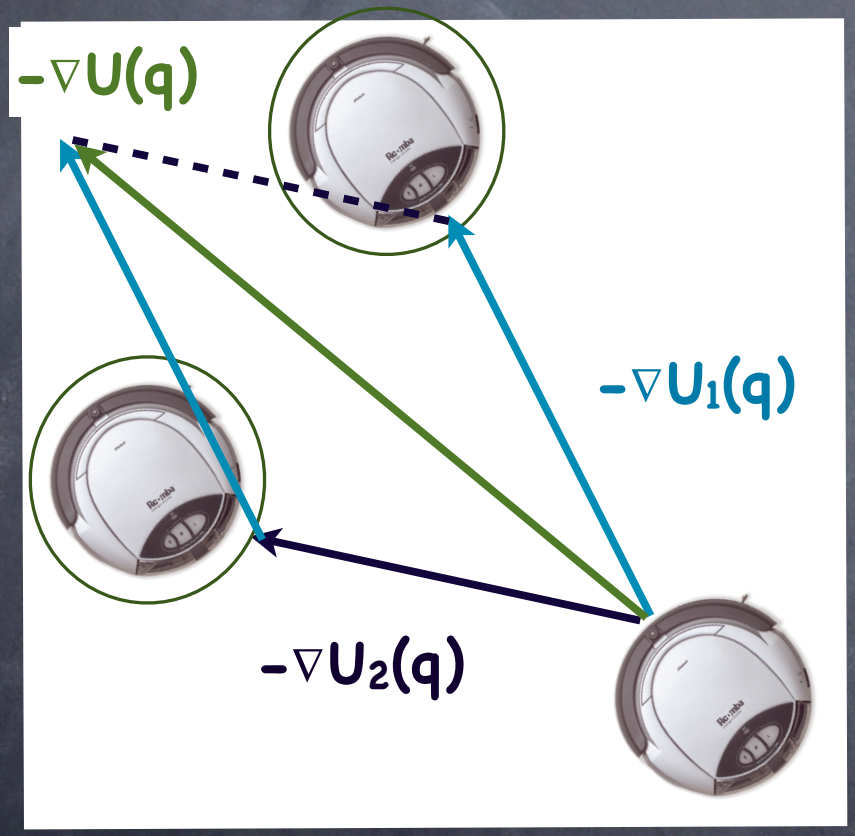


Multiple potentials



- Output of potential field is a vector
- How to combine or select between multiple potentials?

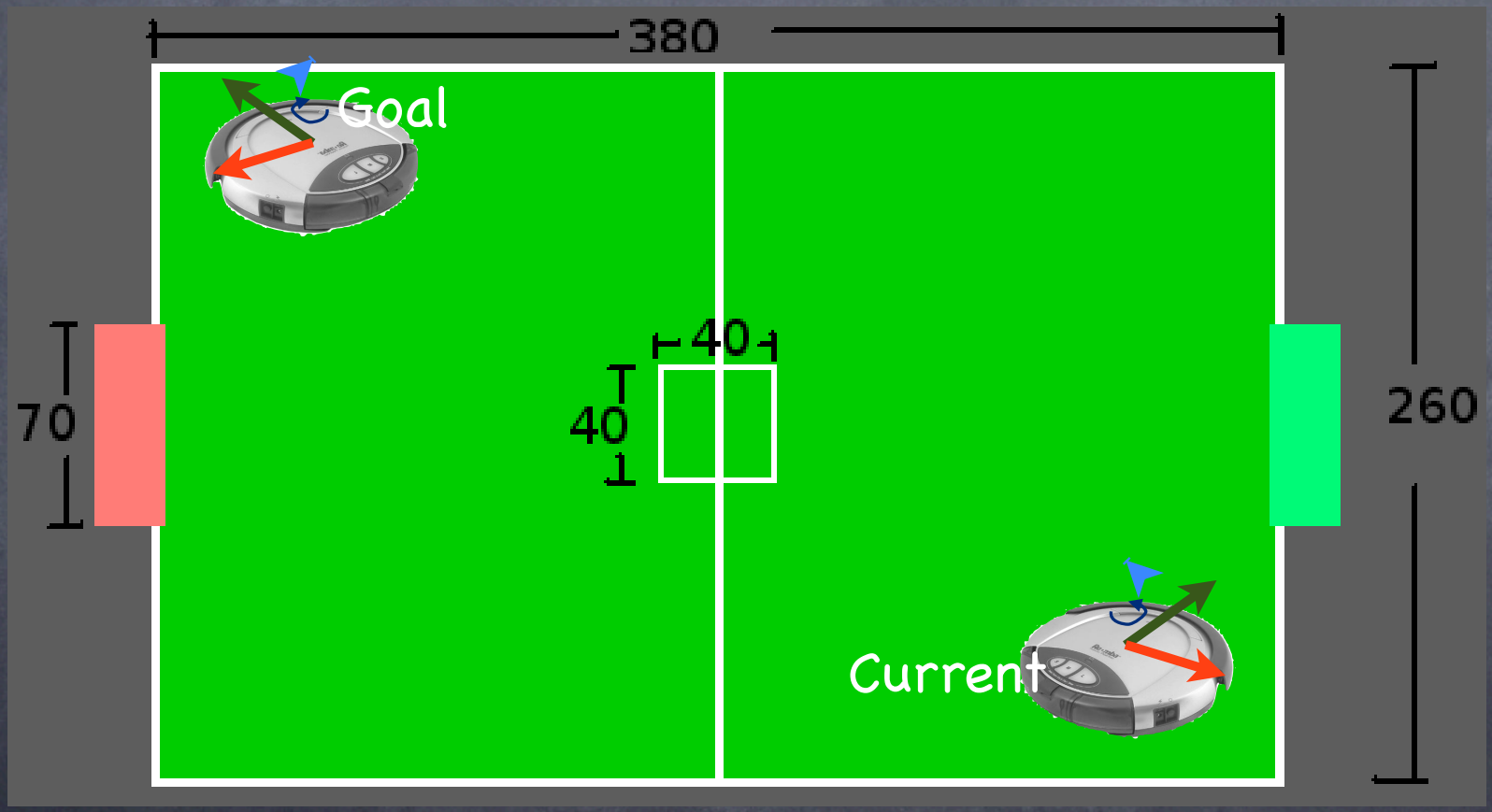
Multiple potentials



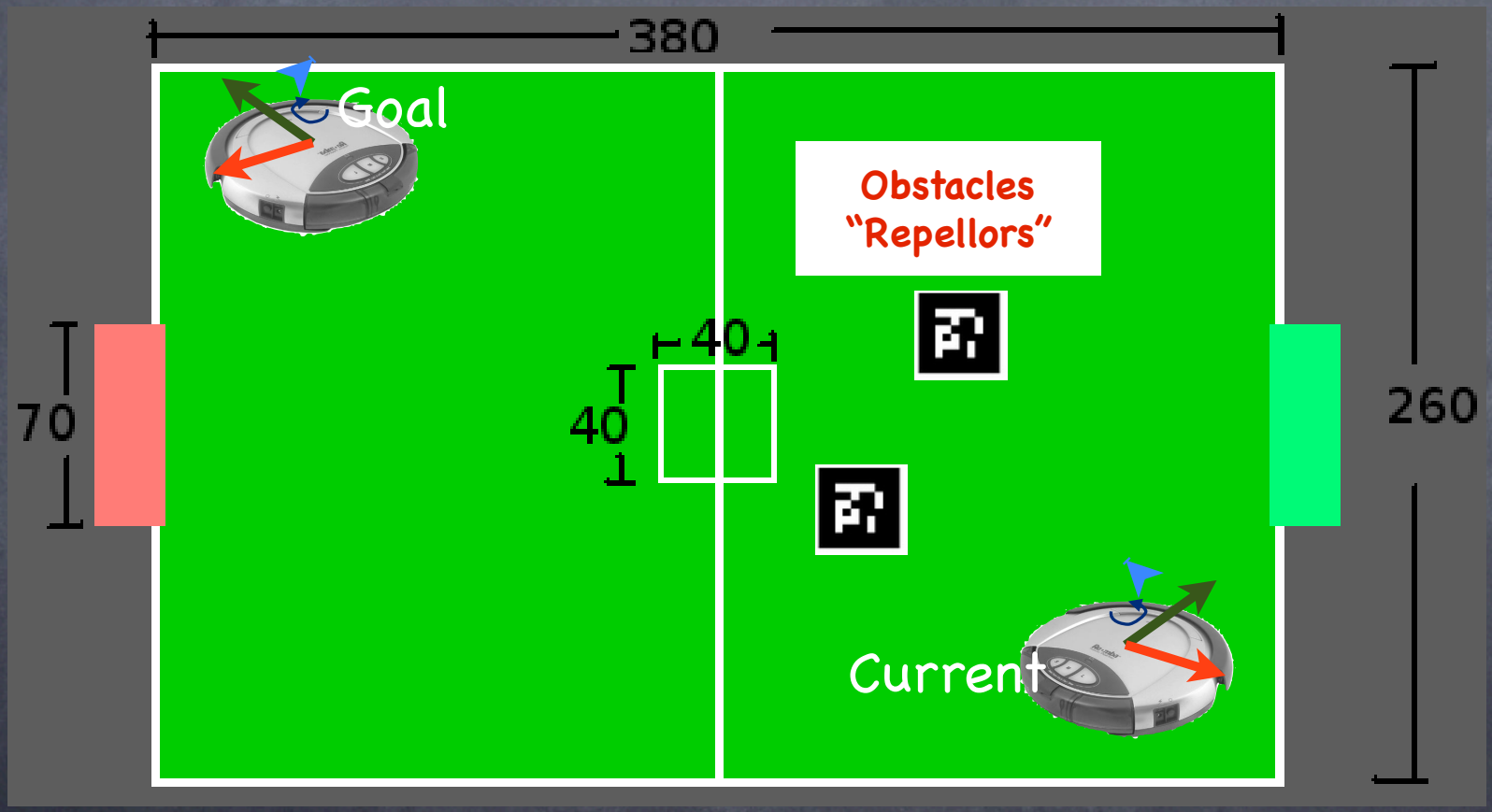
- Output of potential field is a vector
- Combine multiple potentials through vector summation

$$U(q) = \Sigma_i U_i(q)$$

describe performance for this case

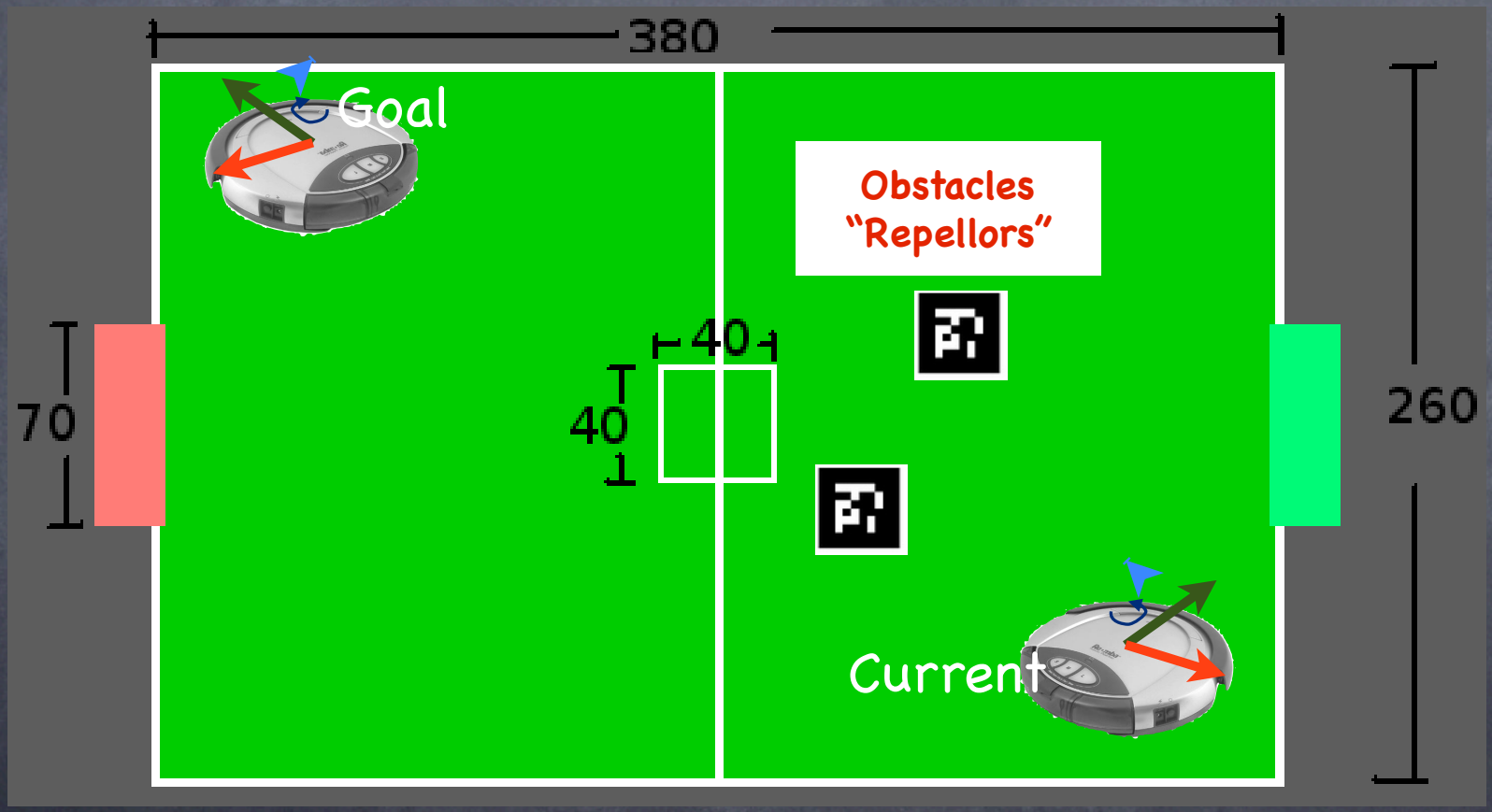


describe performance for this case



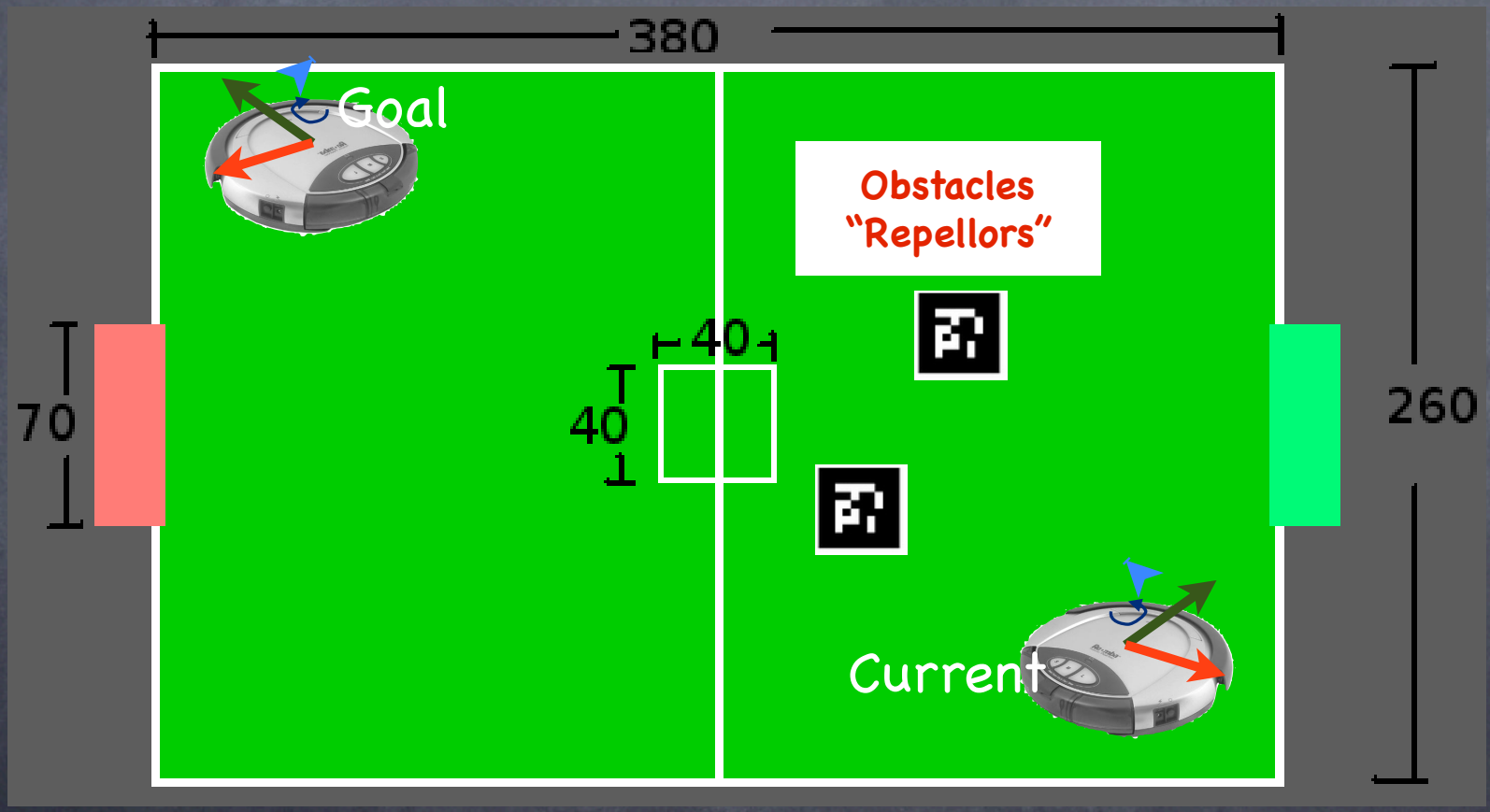
describe performance for this case

how do we deal with repellors?



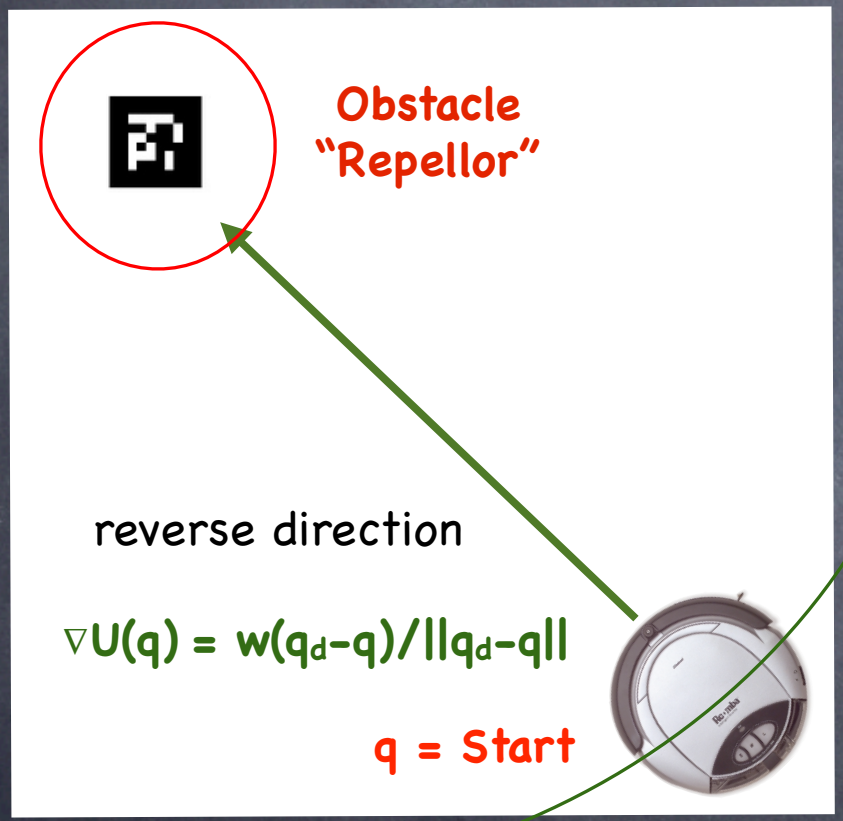
add sum of repulsive potentials

$$U(q) = U_{attracts}(q) + U_{repellors}(q)$$



"Cone" Repellor

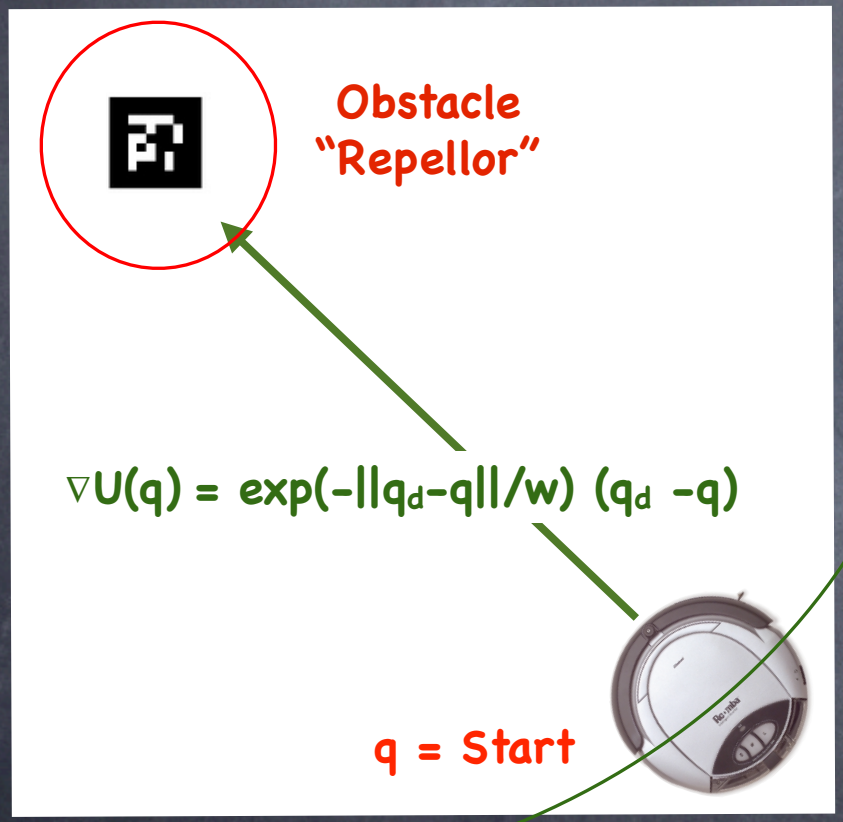
potential problems?



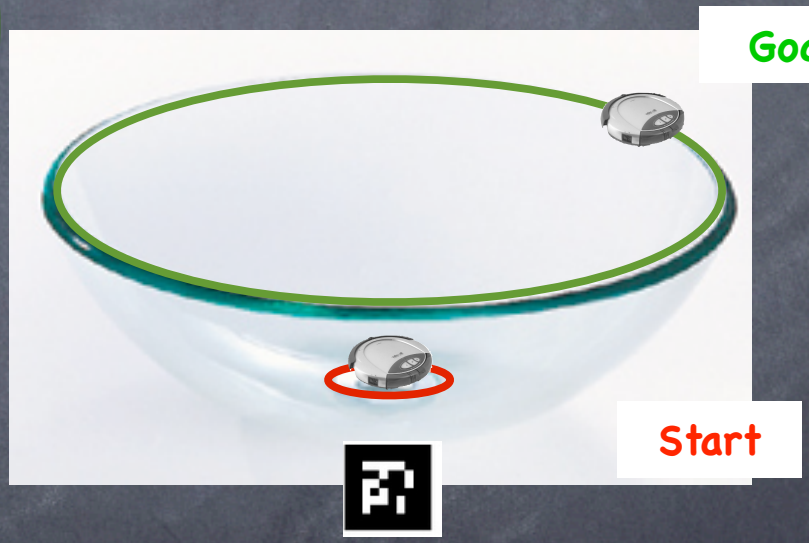
top view

side view

"Bowl" Repellor

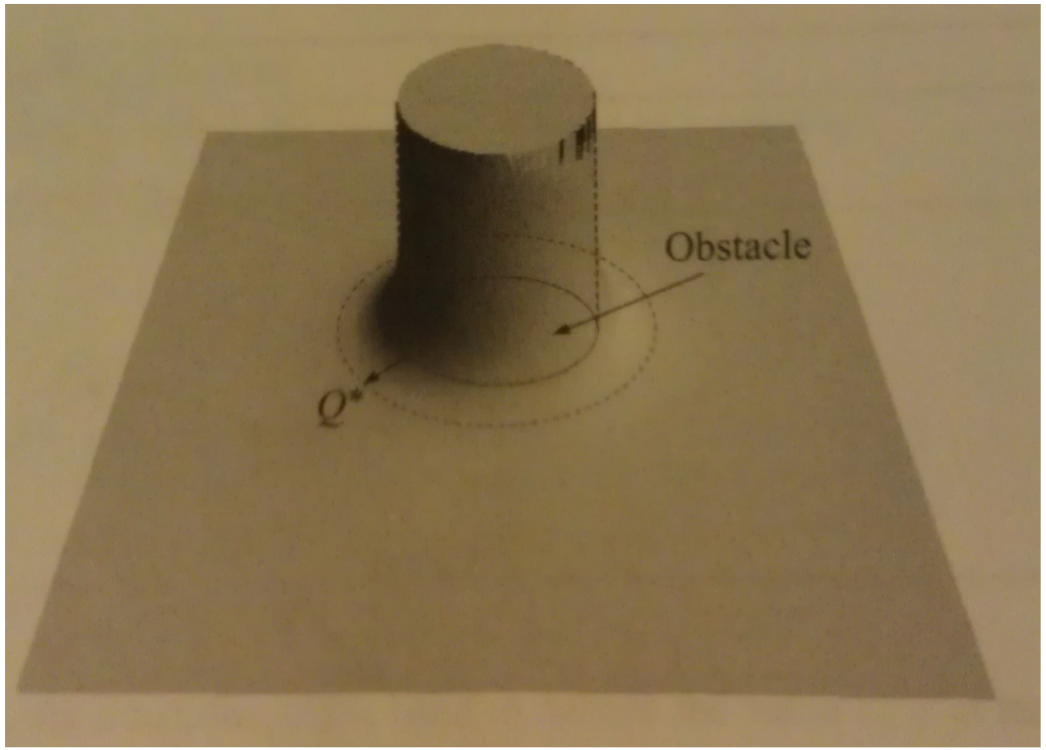


top view

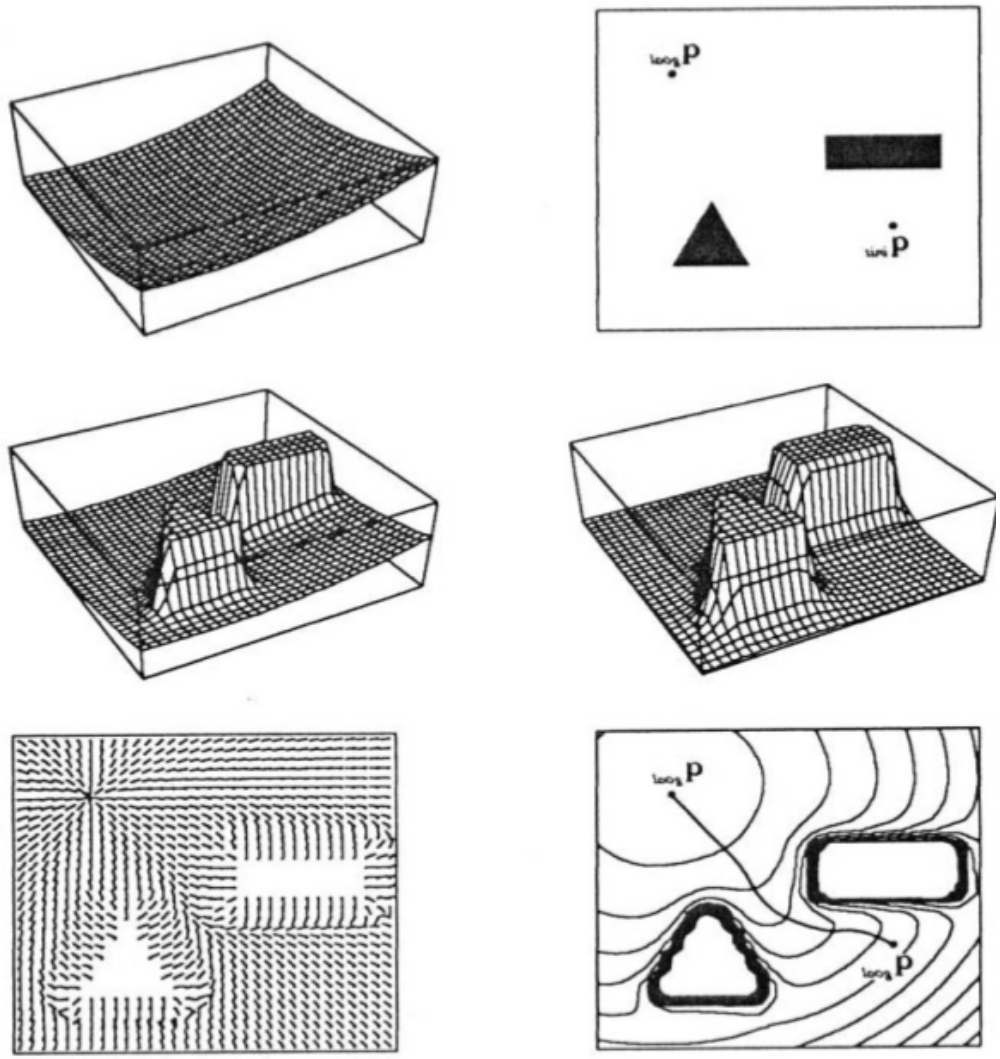


side view

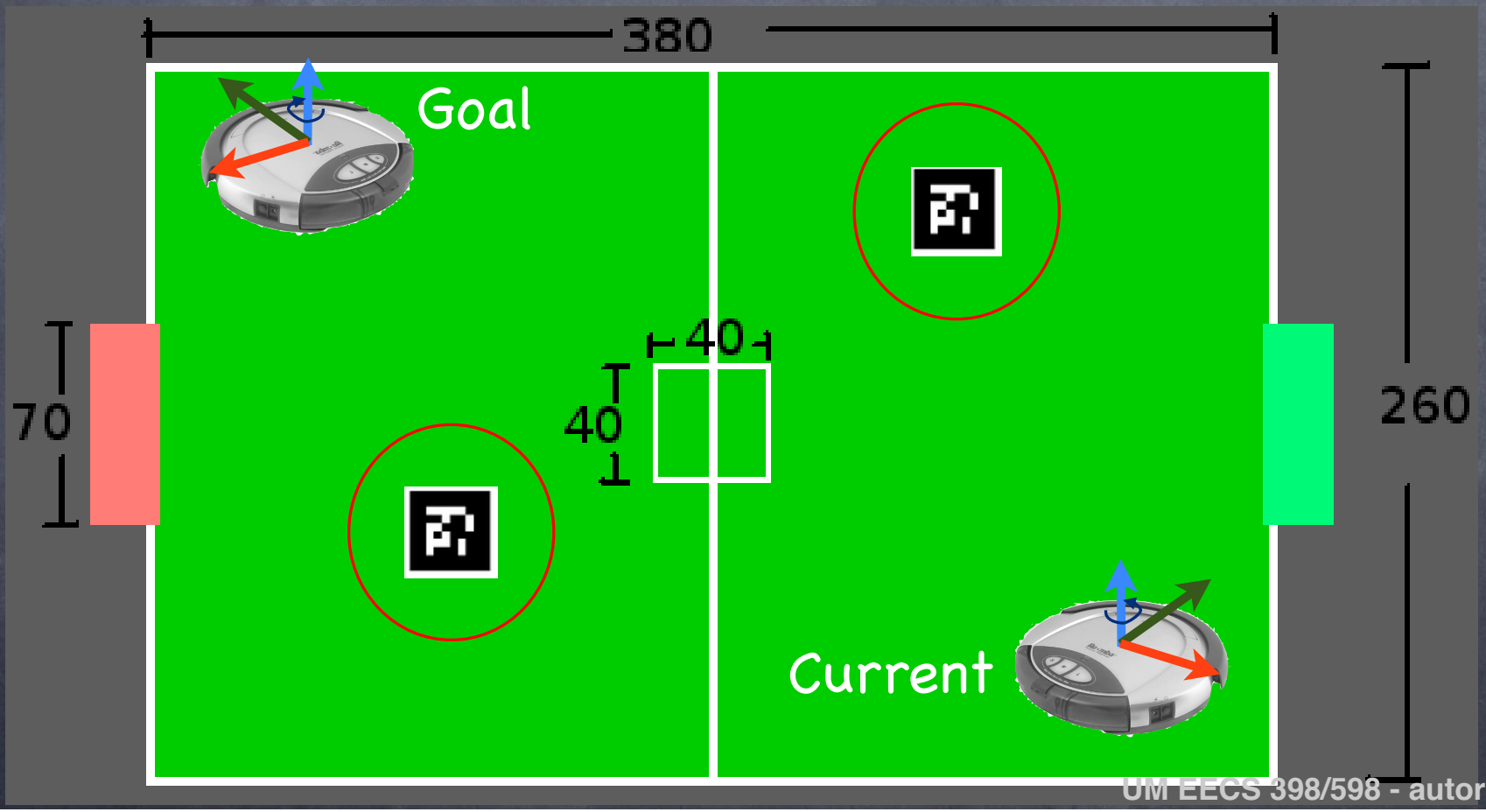
repellor should only have local influence,
repelling only around boundary improves path



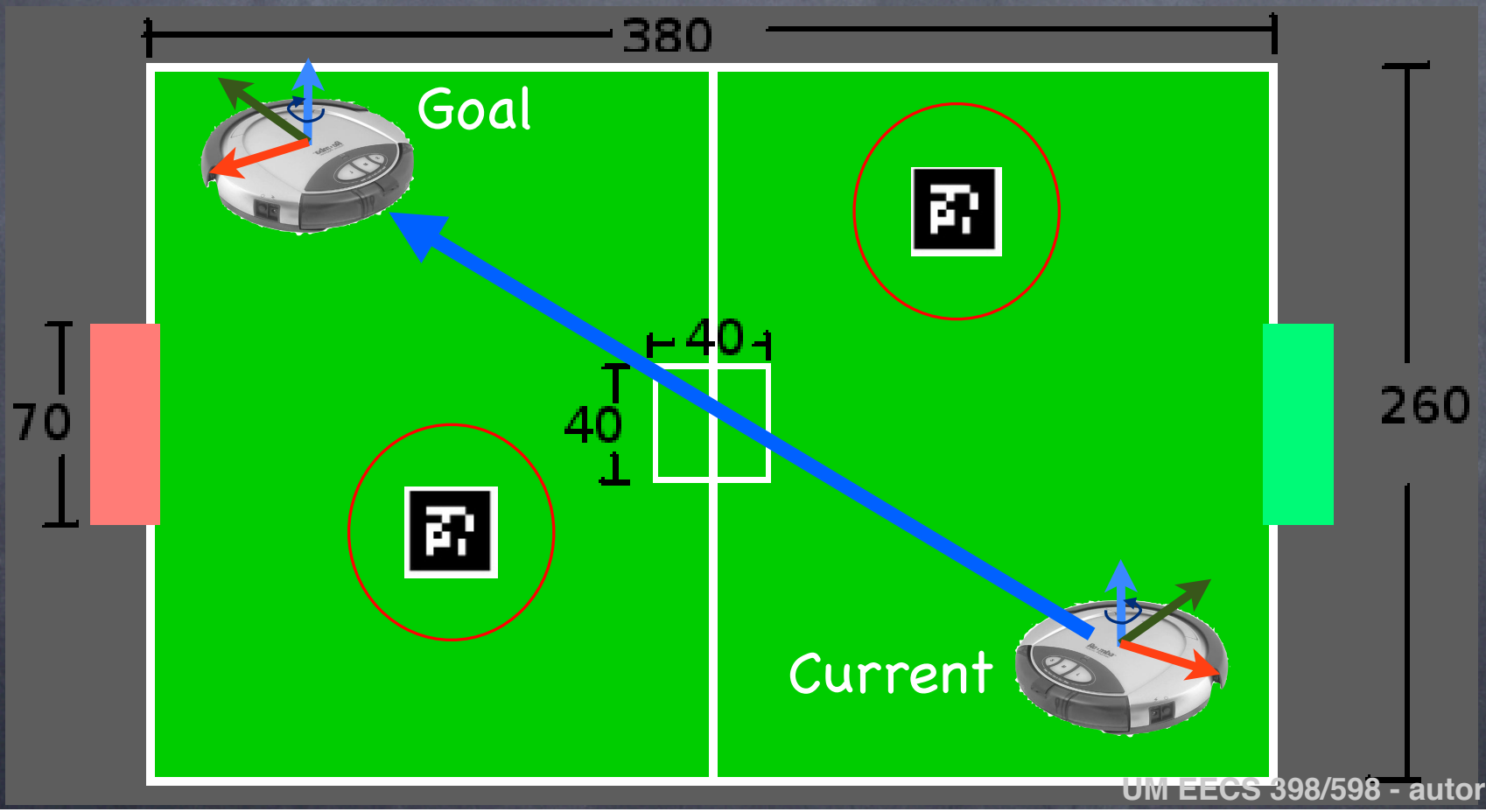
2 Obstacle example



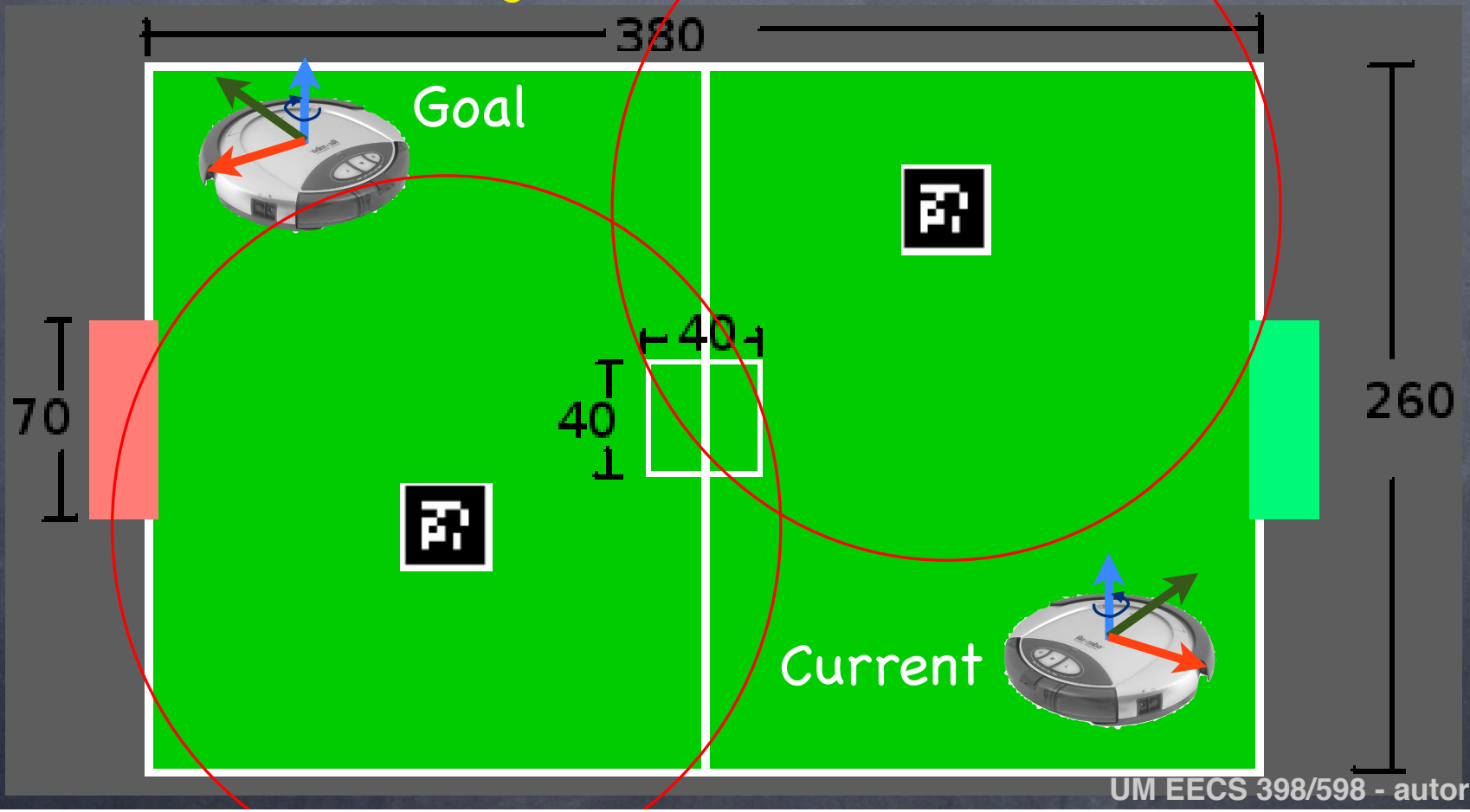
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



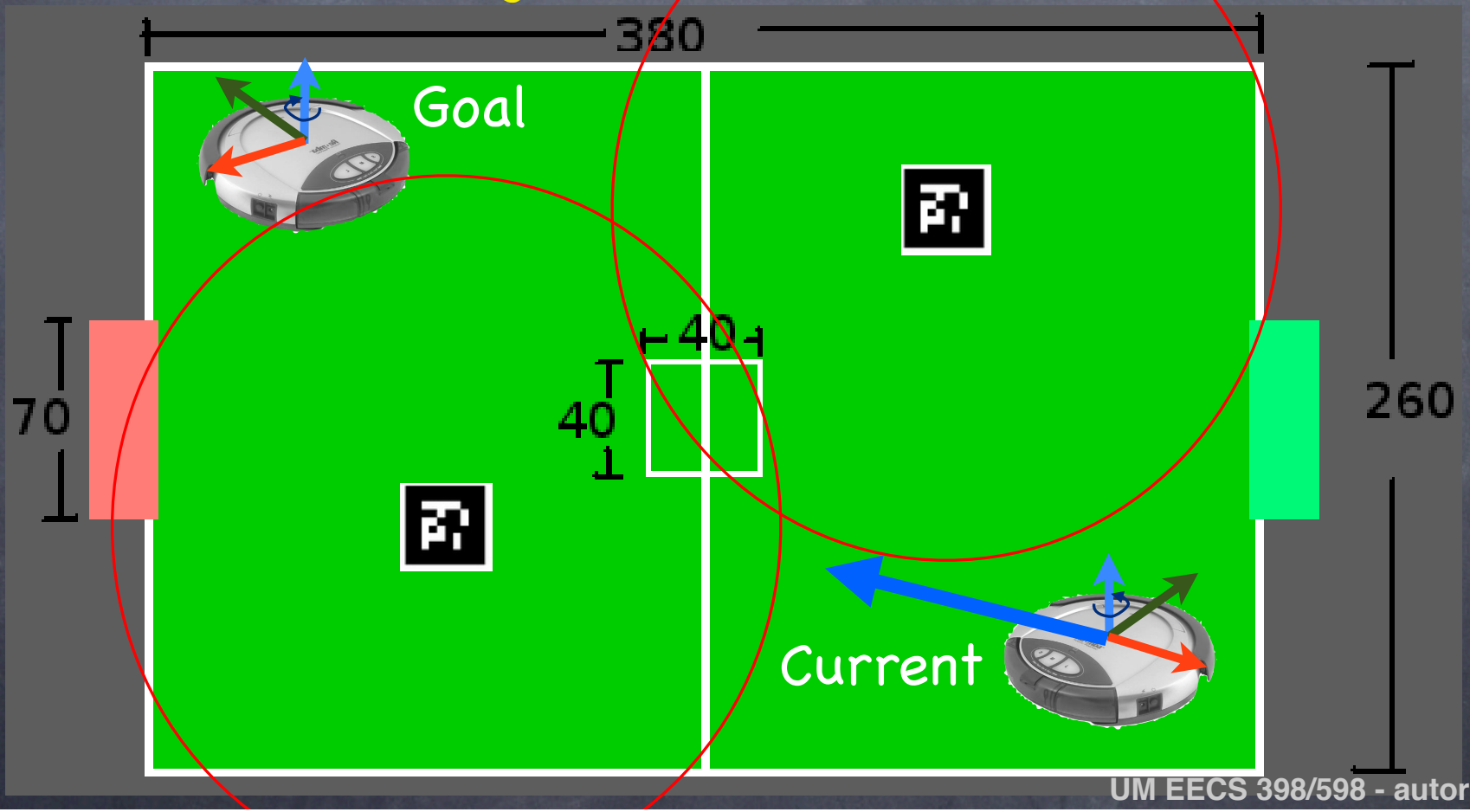
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



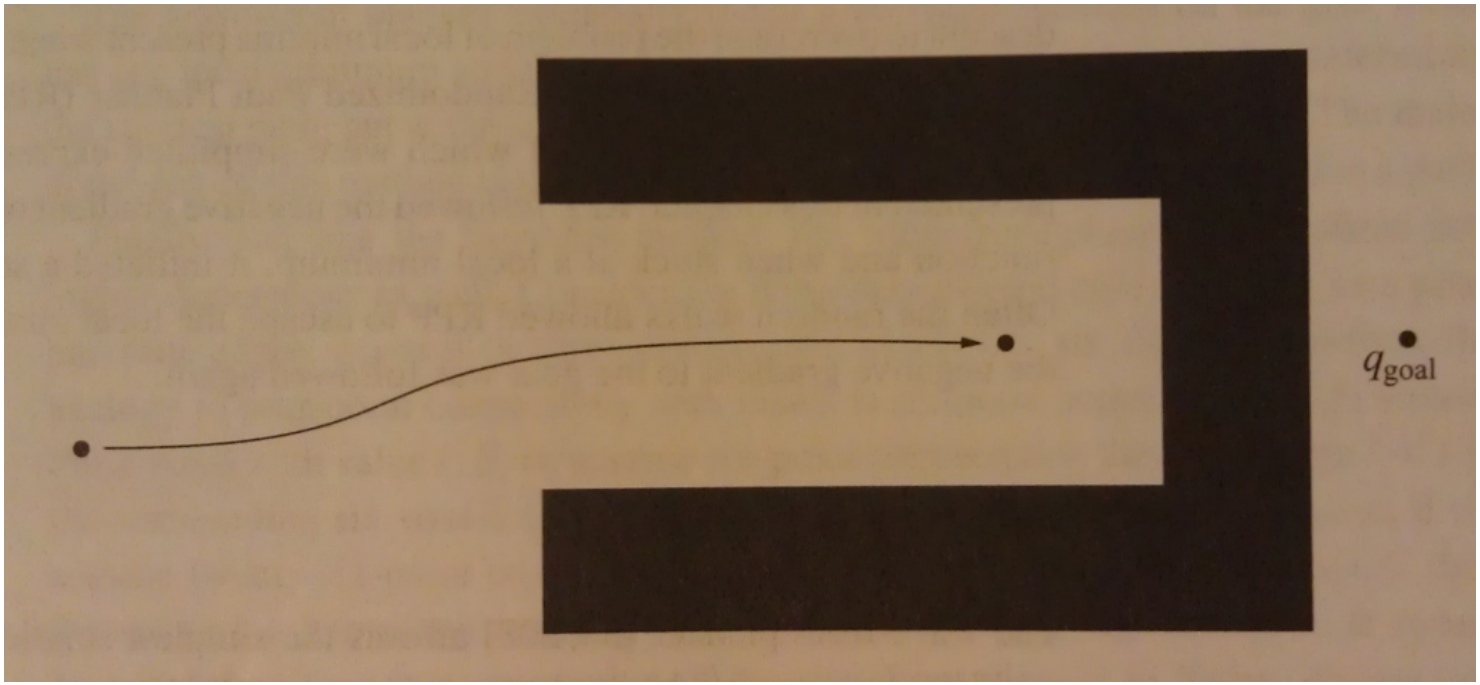
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



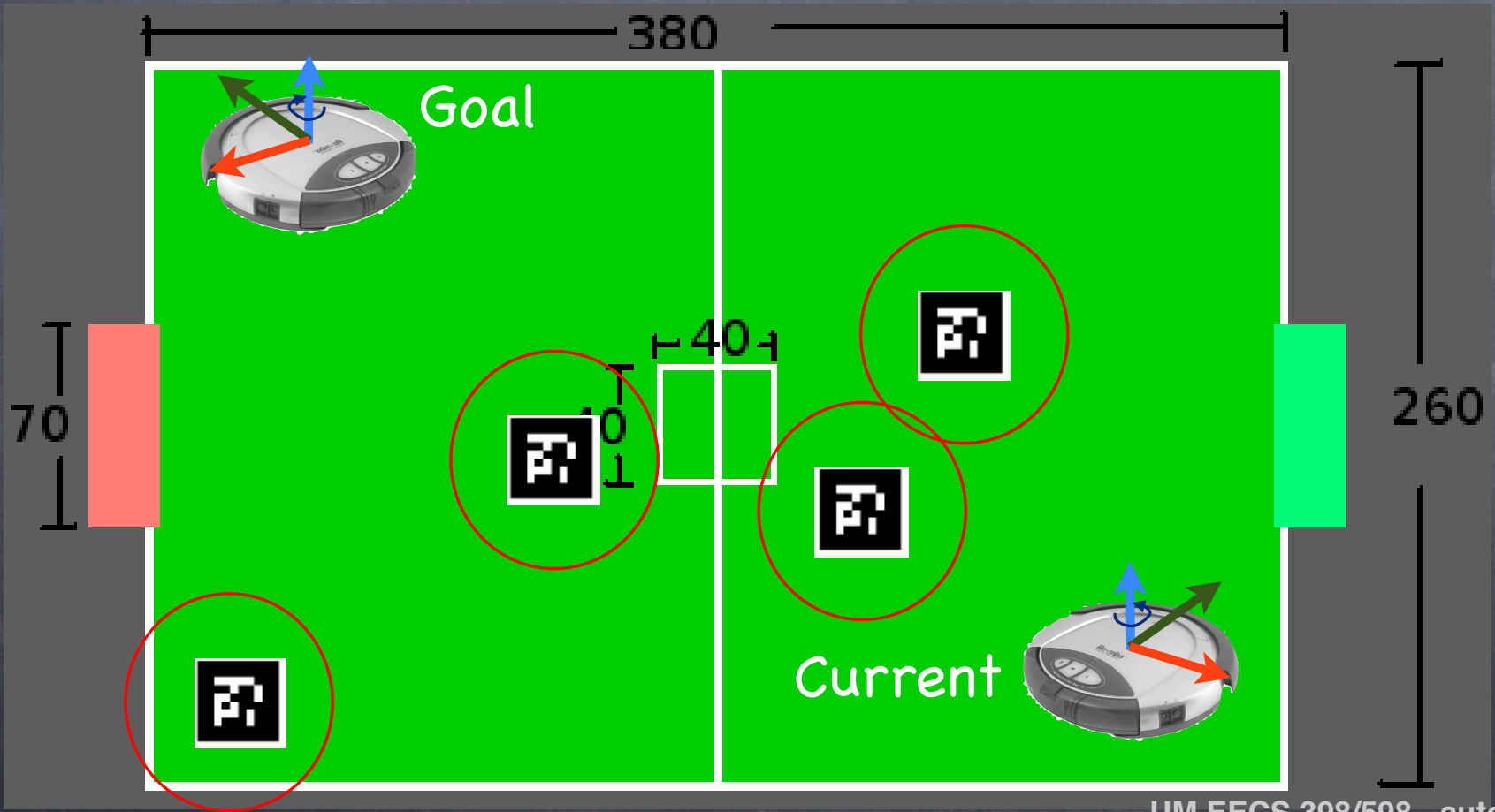
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



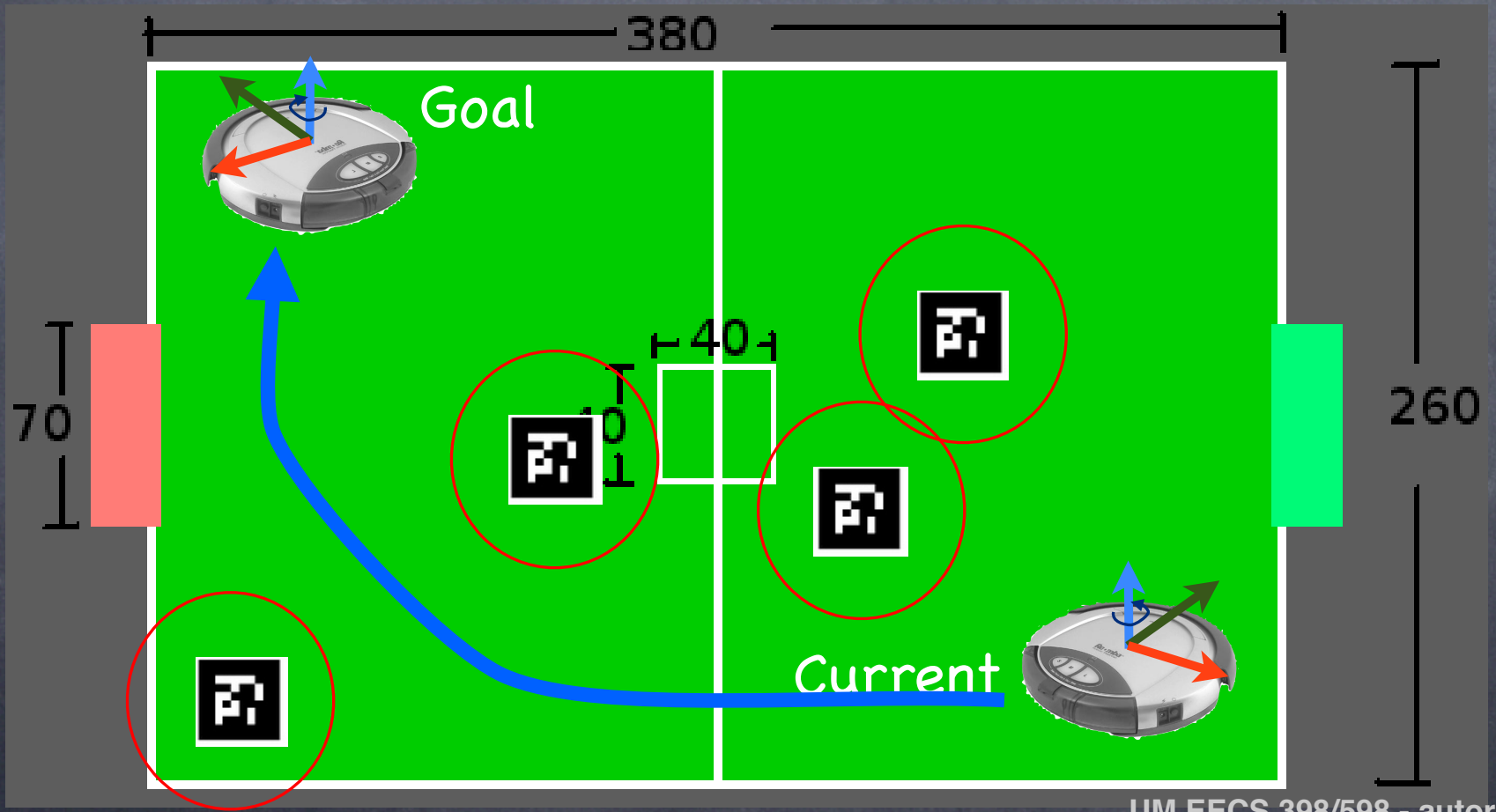
Local Minima



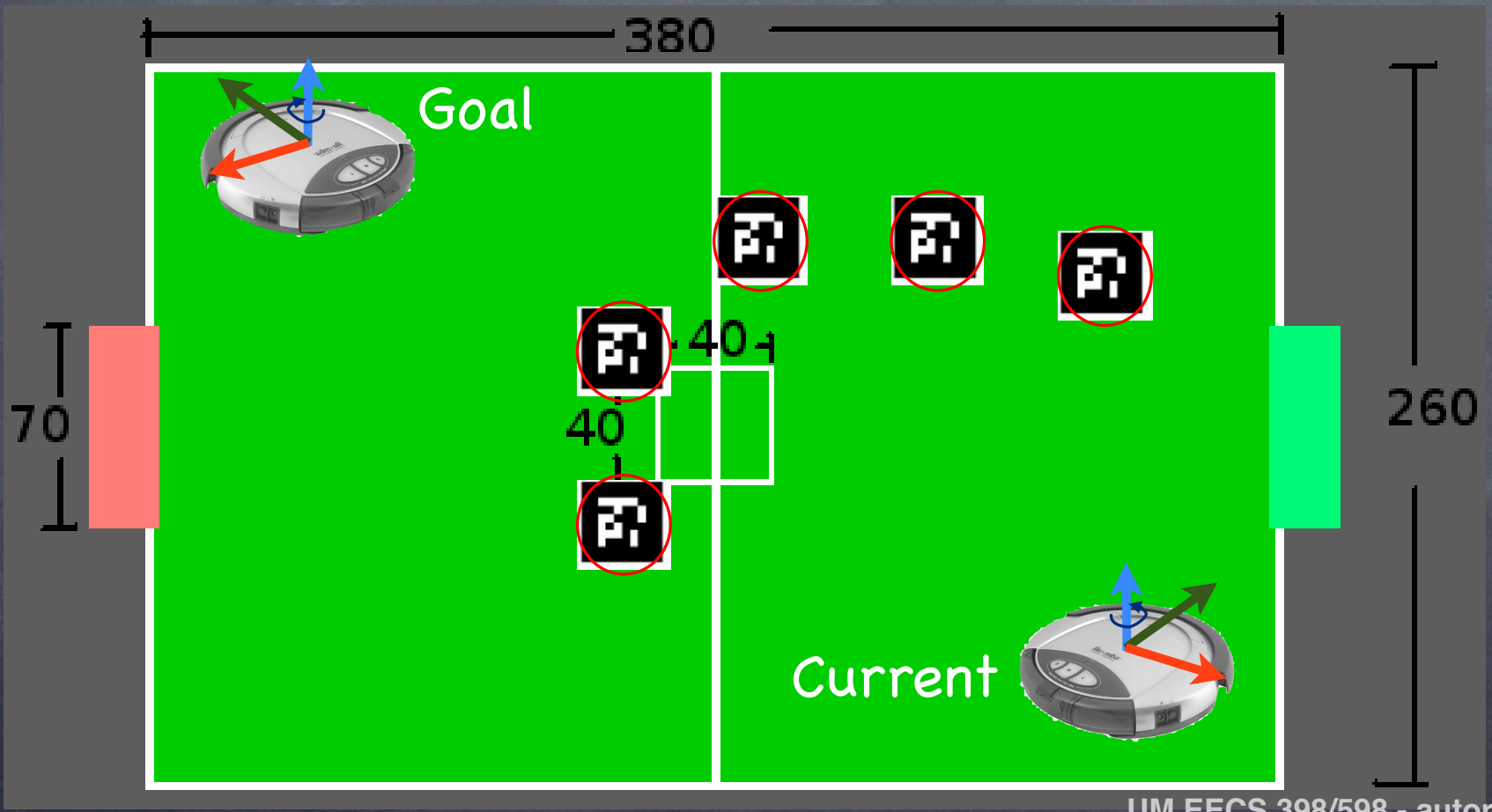
describe performance for this case



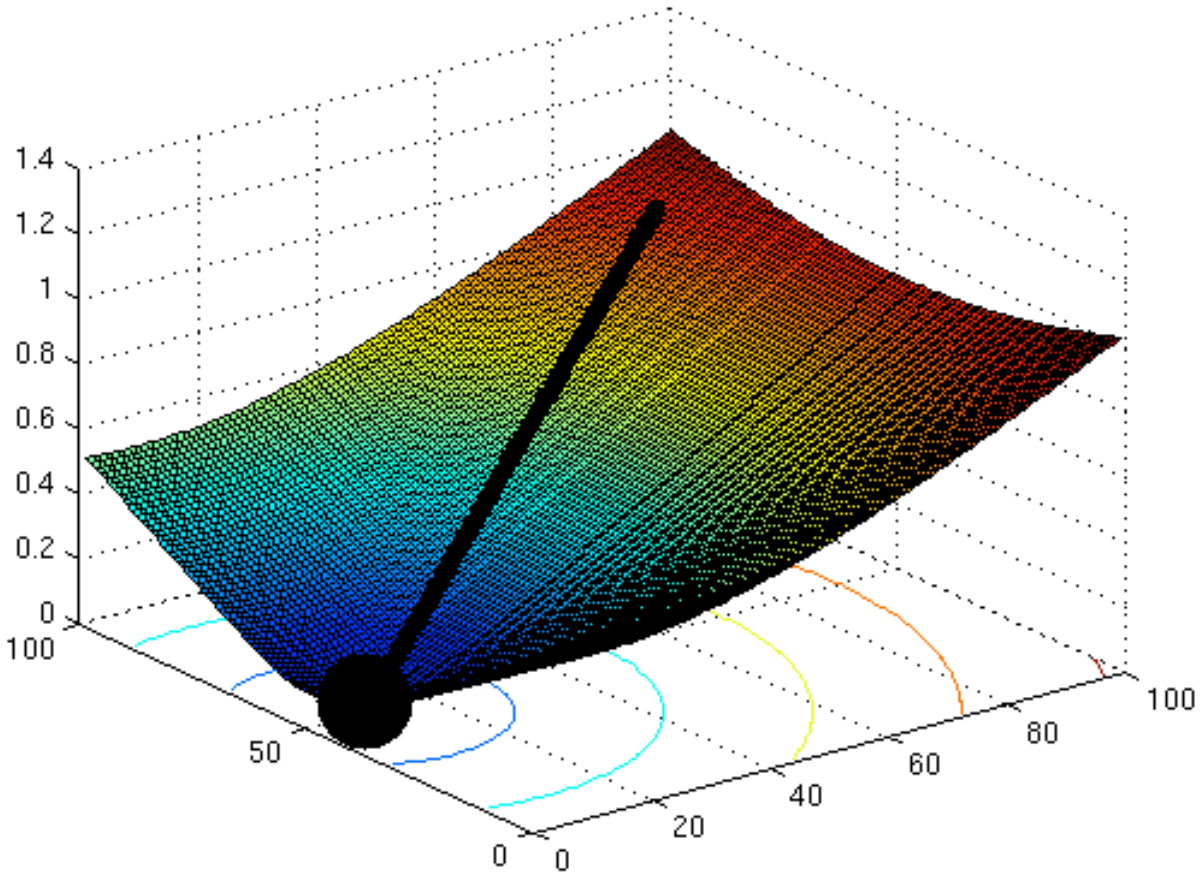
describe performance for this case



describe performance for this case

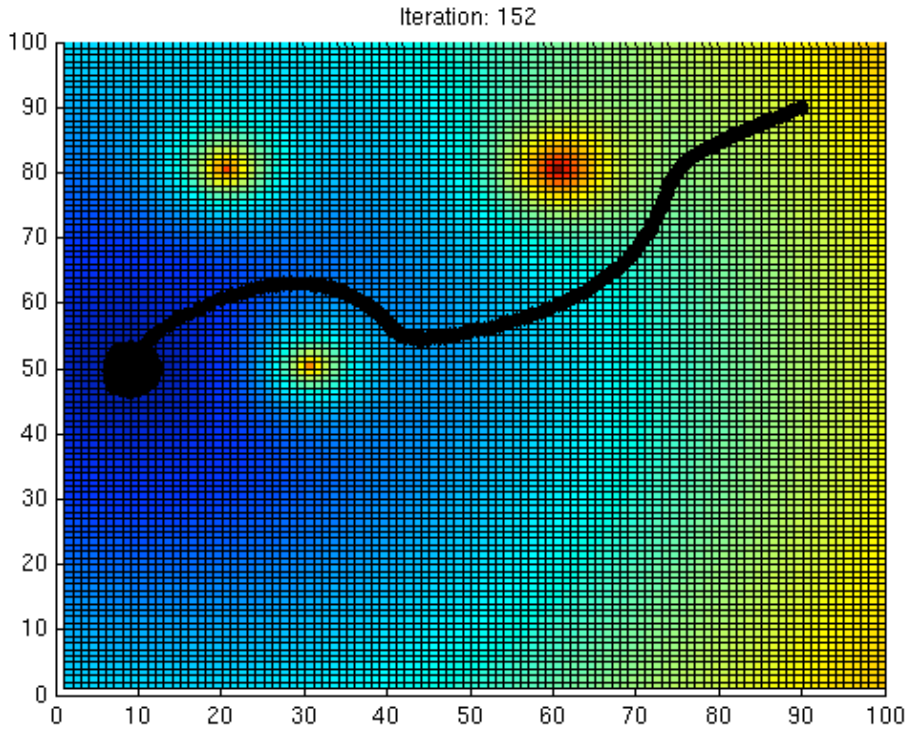


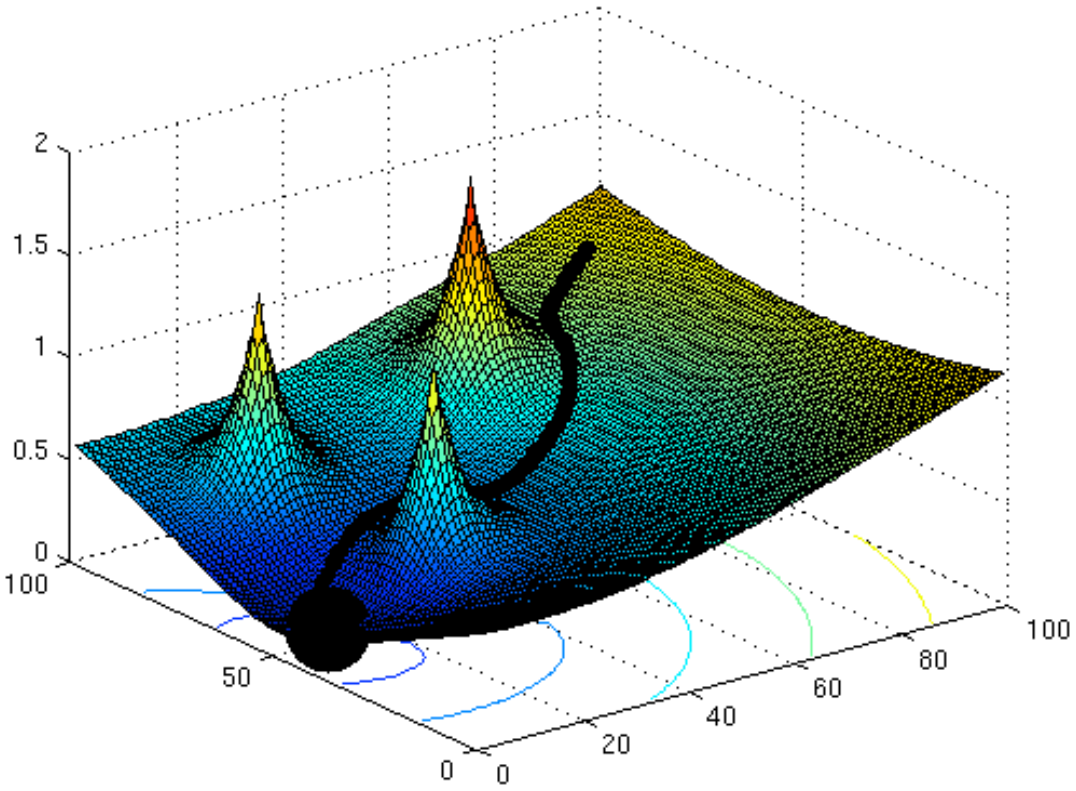
matlab example



pfield.m [1 5 8 12]

matlab example

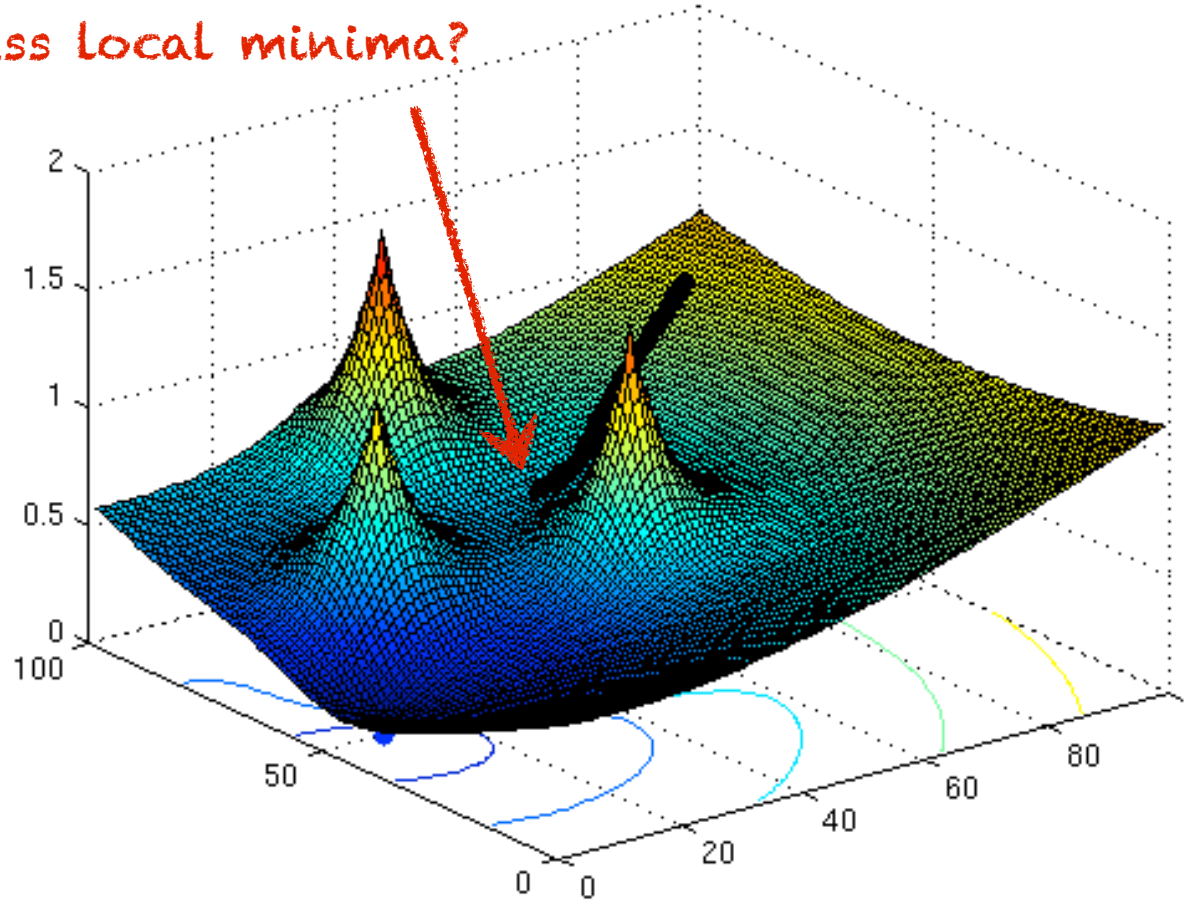




pfield.m [1 5 8 12]

matlab example

How to address local minima?



pfield.m [1 5 8 12]

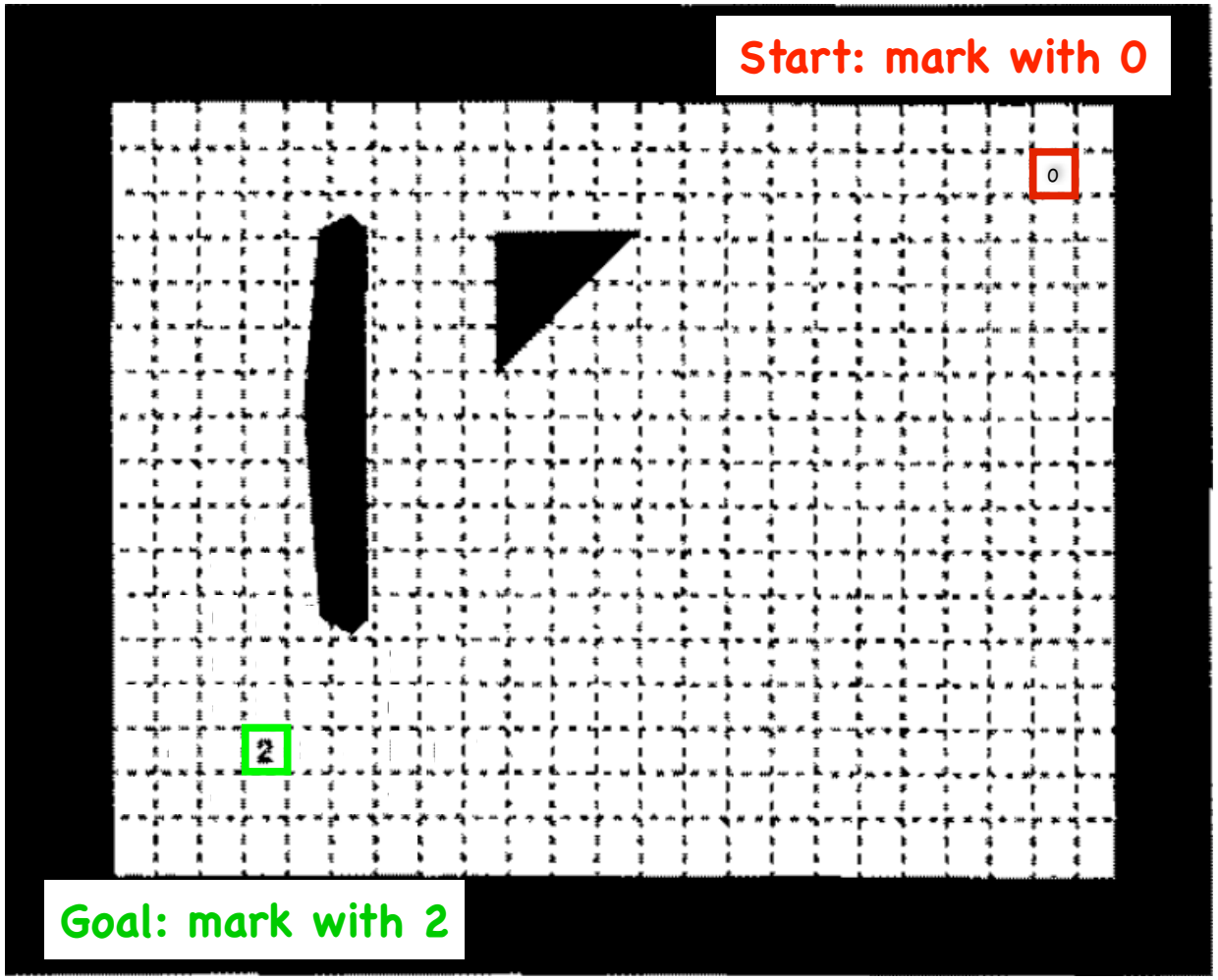
How can we get out of local minima?

How can we get out of local minima?

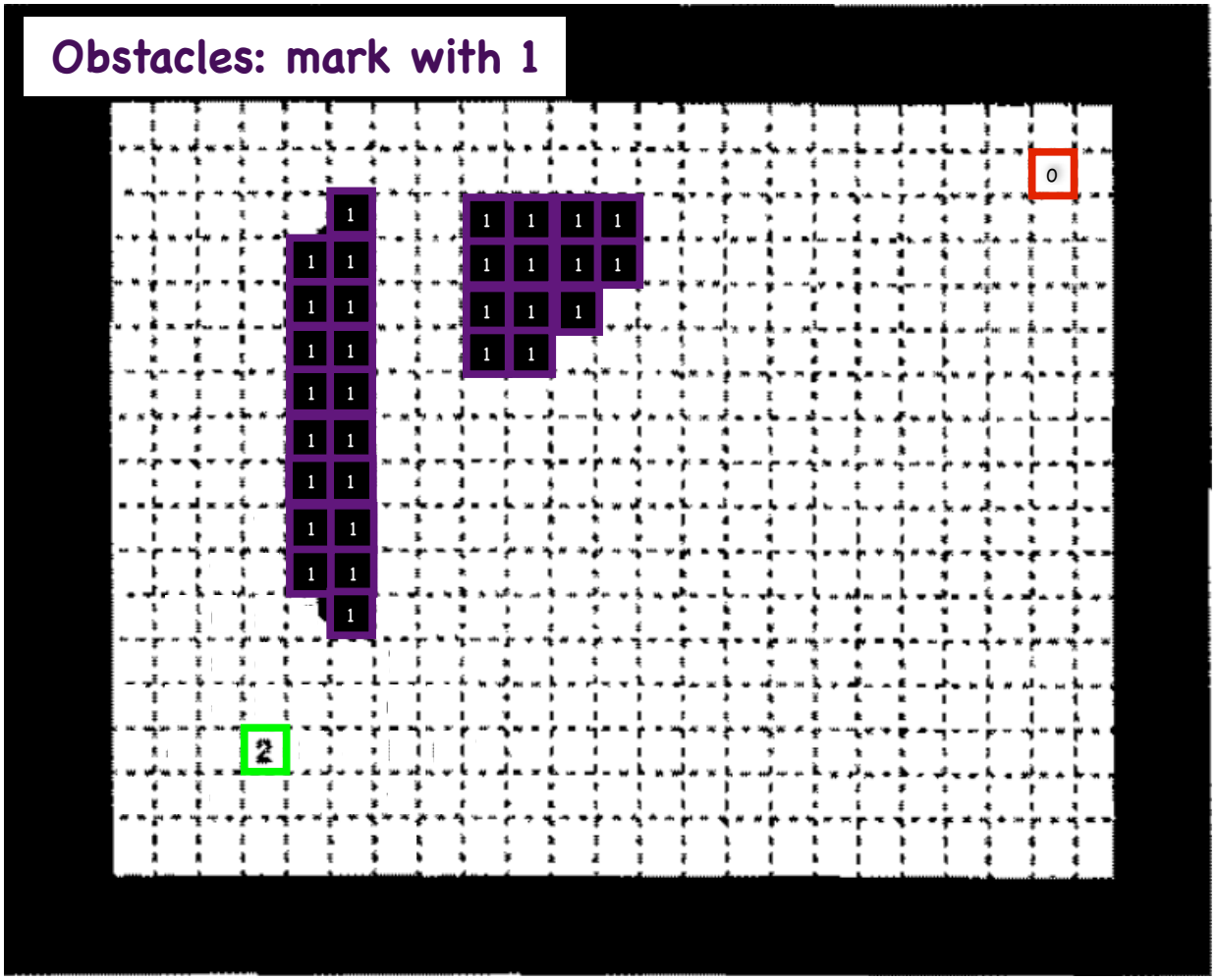
Go back to planning.

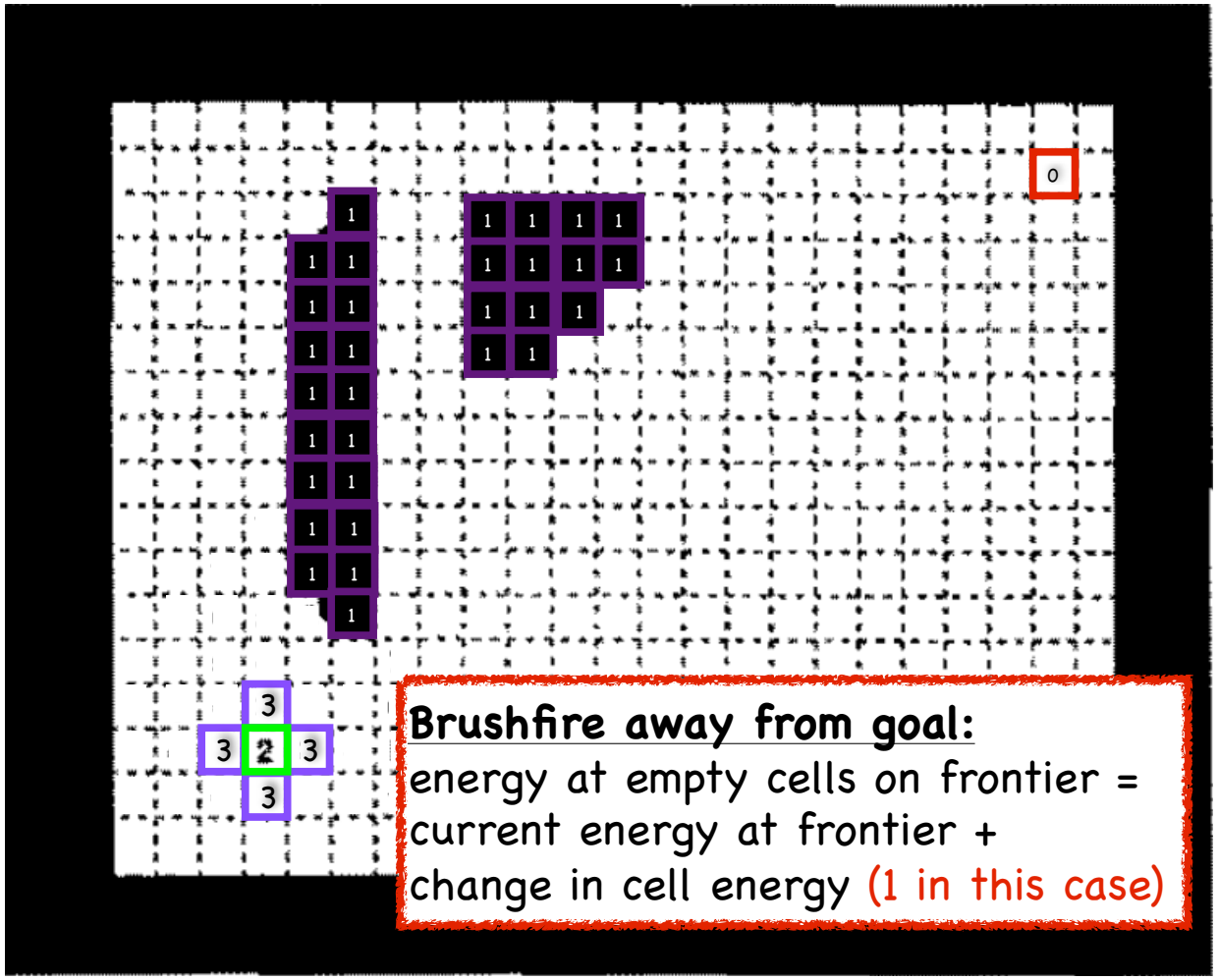
Wavefront Planning

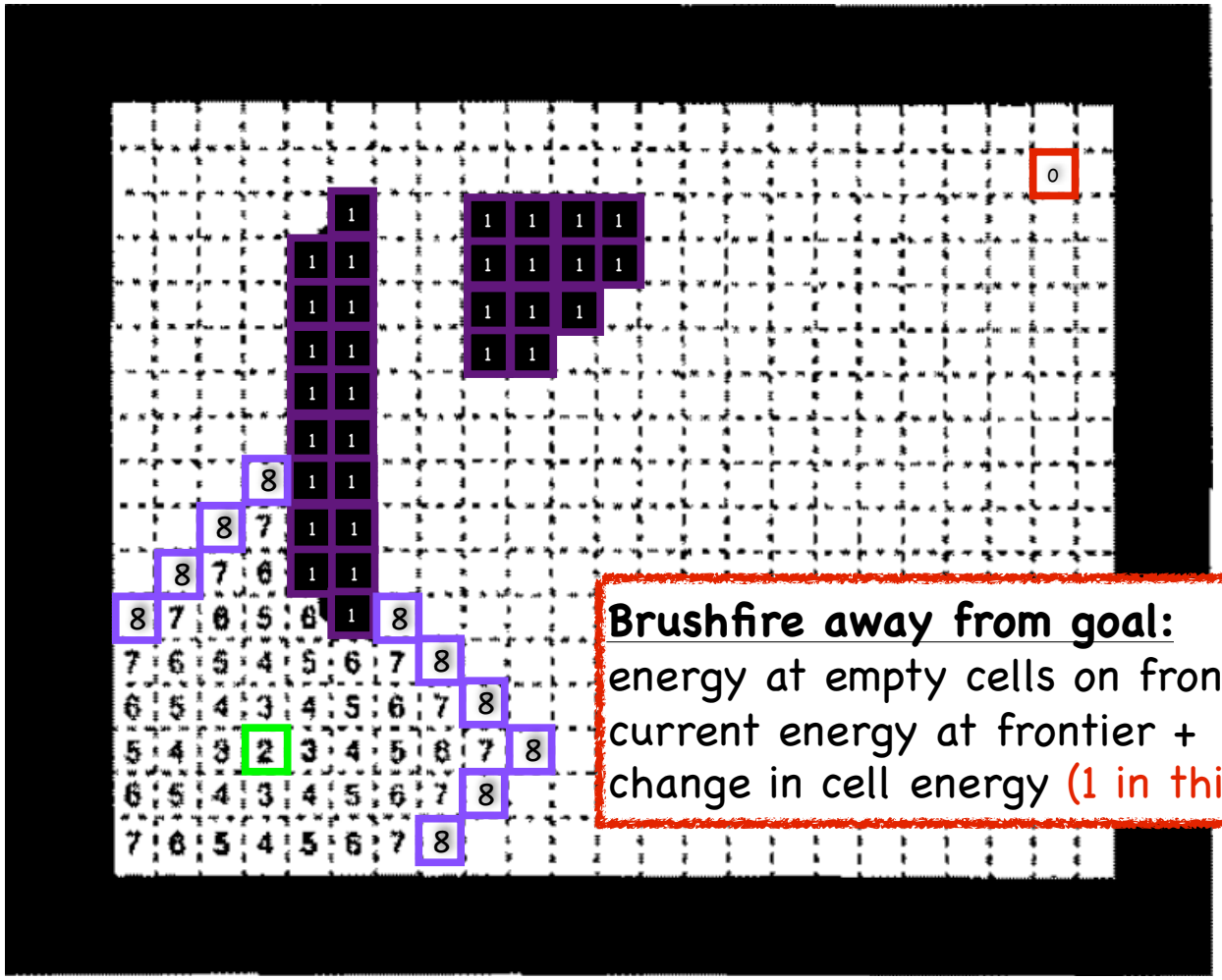
- Discretize potential field into grid
  - Cells store cost to goal with respect to potential field
  - Computed by Brushfire algorithm (essentially BFS)
- Grid search to find navigation path to goal

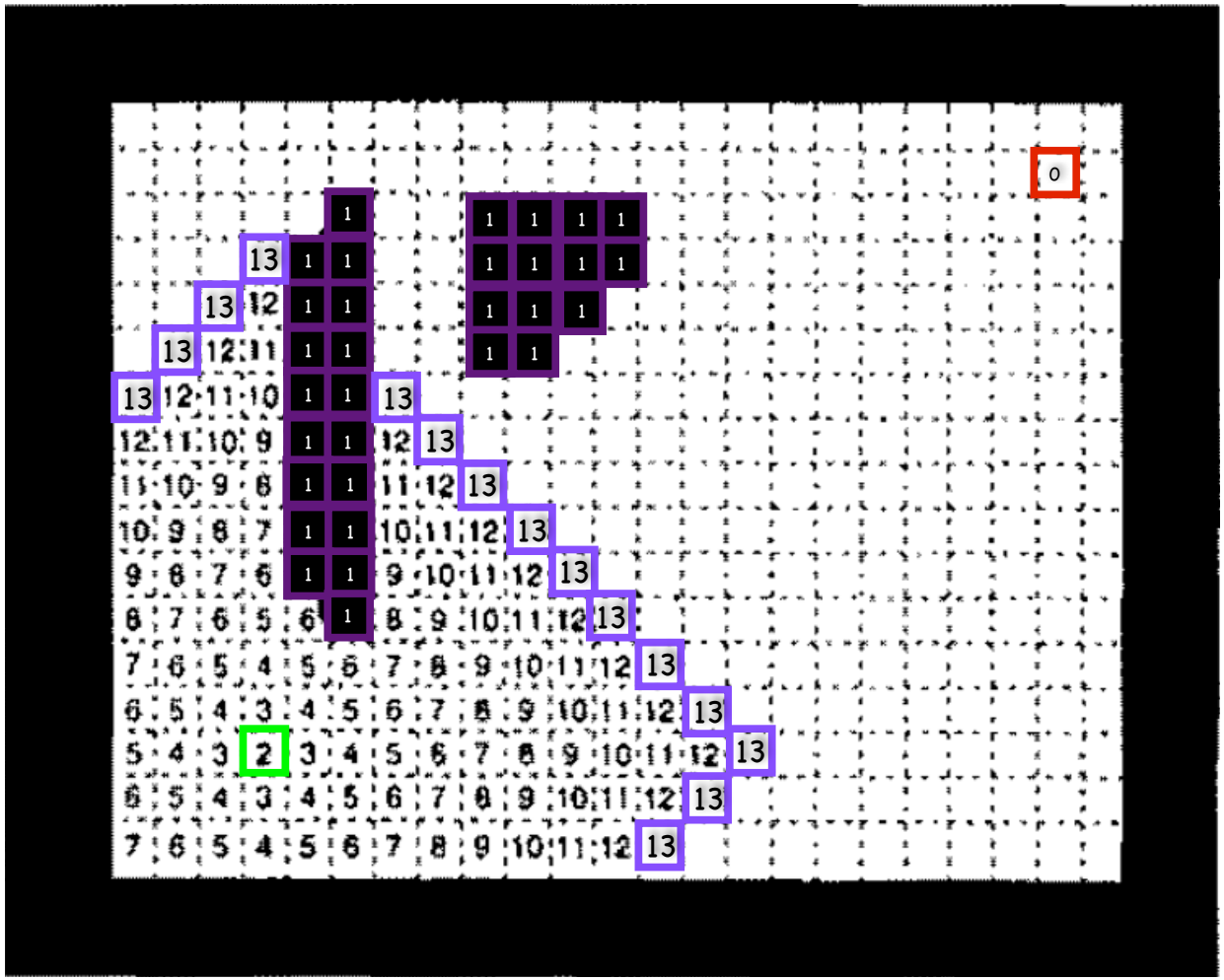

Start: mark with 0
0
2
Goal: mark with 2

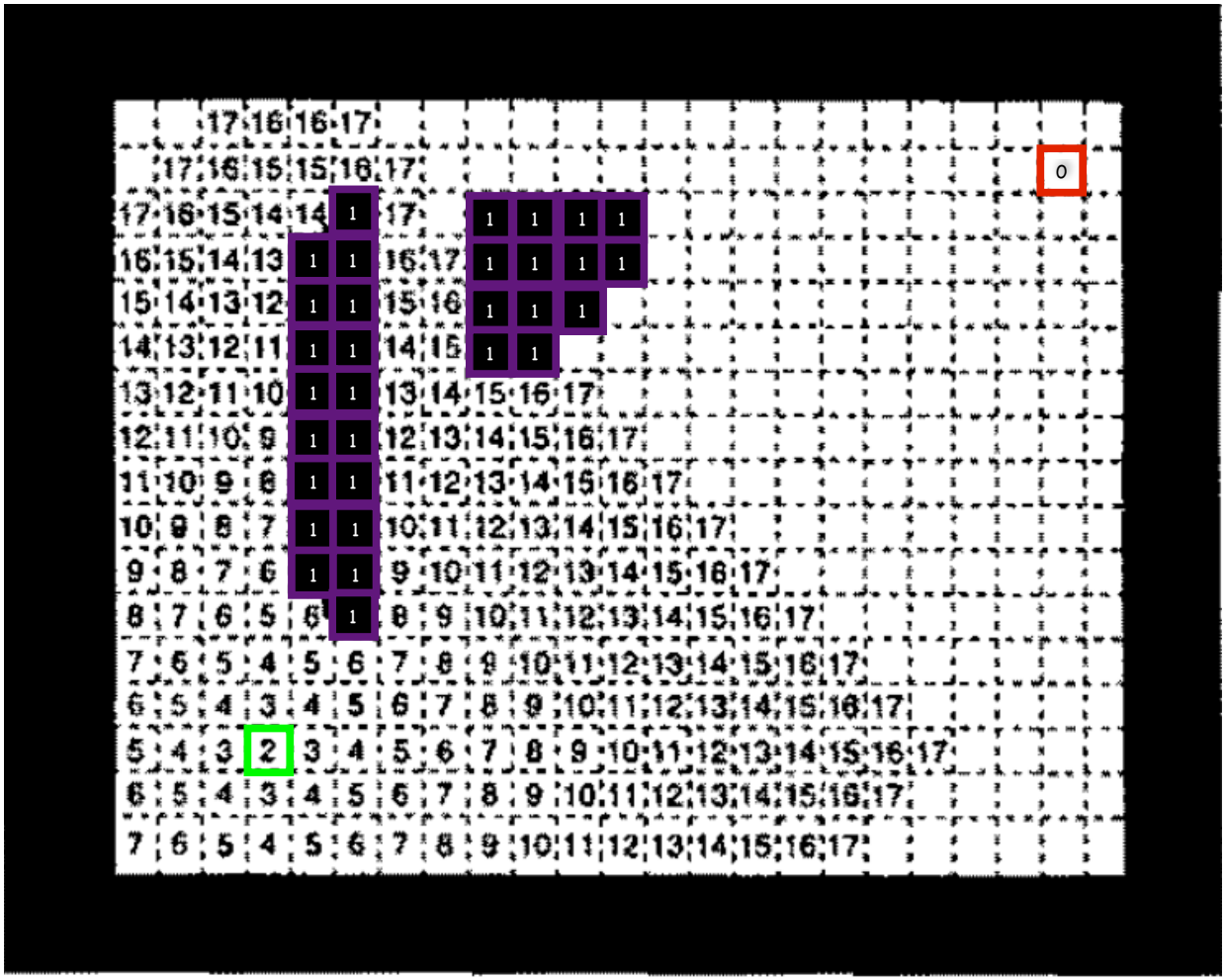
Obstacles: mark with 1

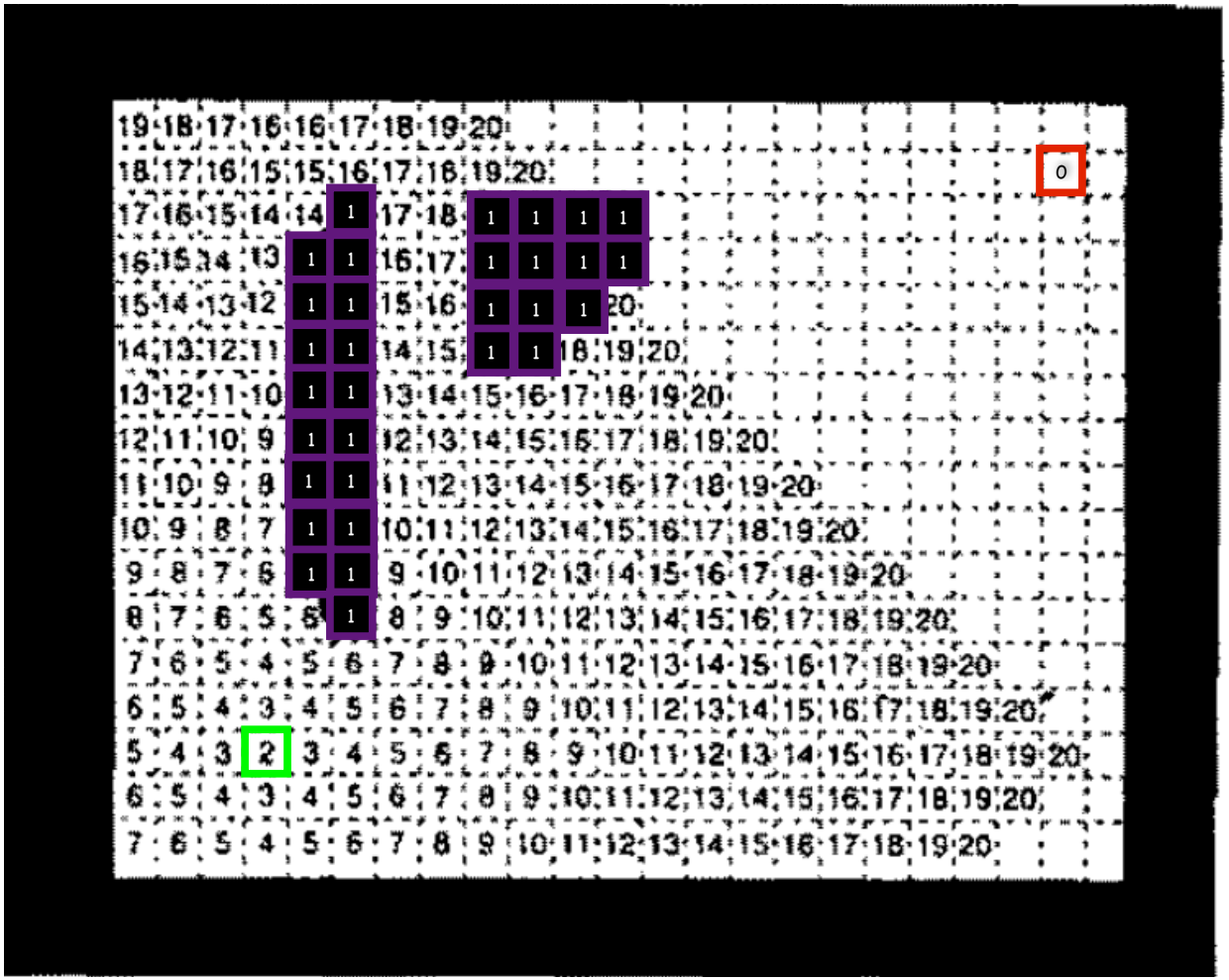


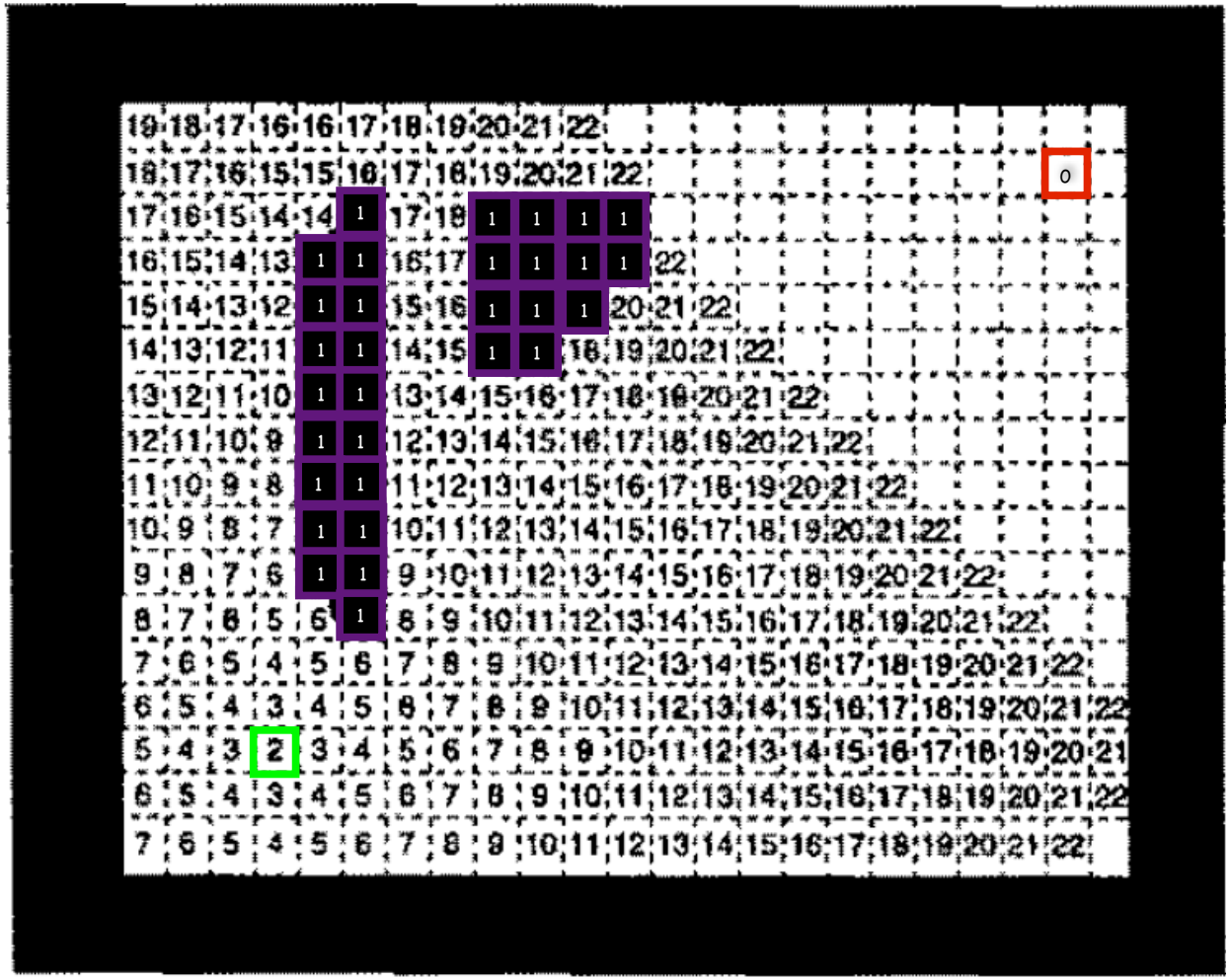

0
1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1
1 1 1 1
1 1 1 1
1 1 1
1 1
3
3 2 3
3
Brushfire away from goal:
energy at empty cells on frontier =
current energy at frontier +
change in cell energy (1 in this case)

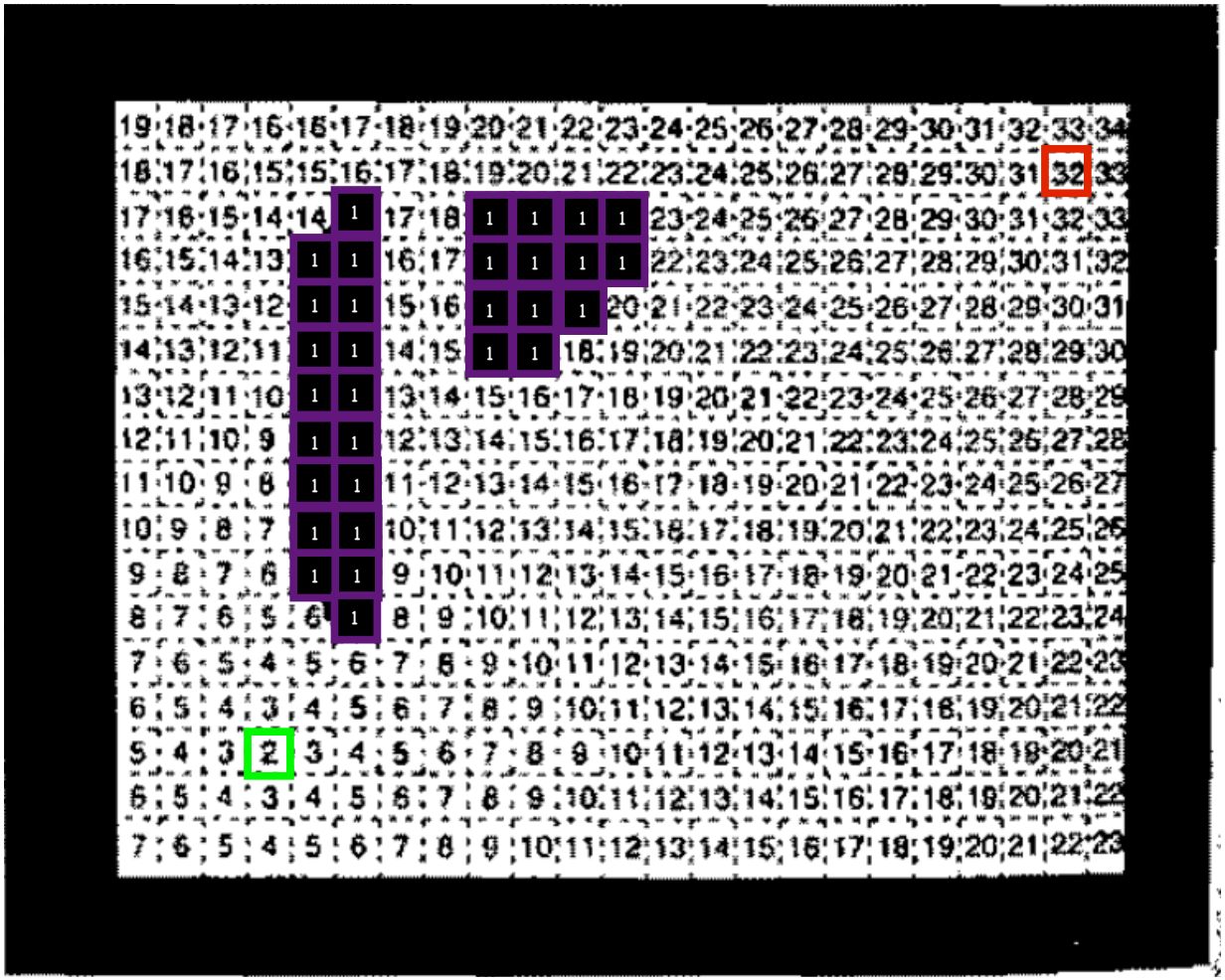

0
Brushfire away from goal:
energy at empty cells on frontier =
current energy at frontier +
change in cell energy (1 in this case)


0
1
13
2

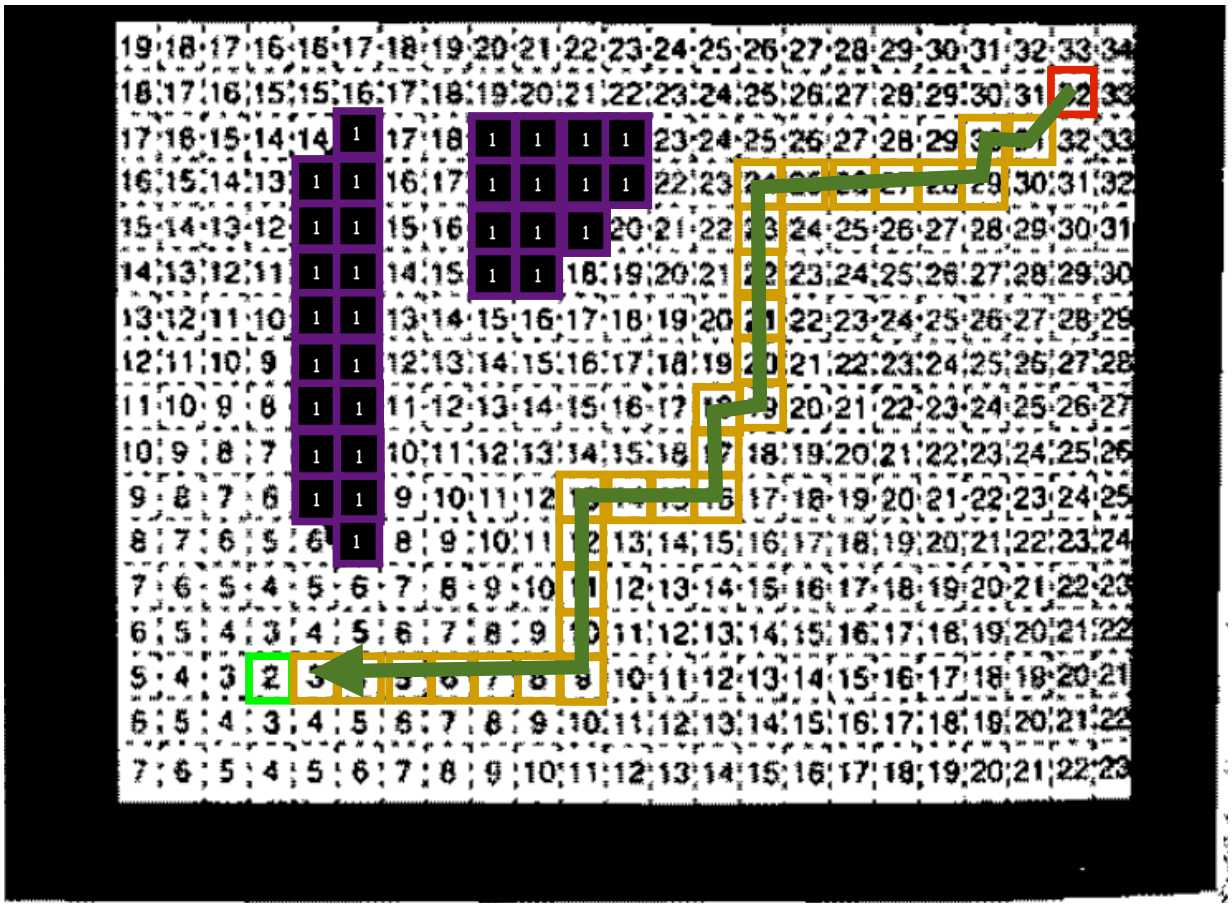

0


0
2


0
2

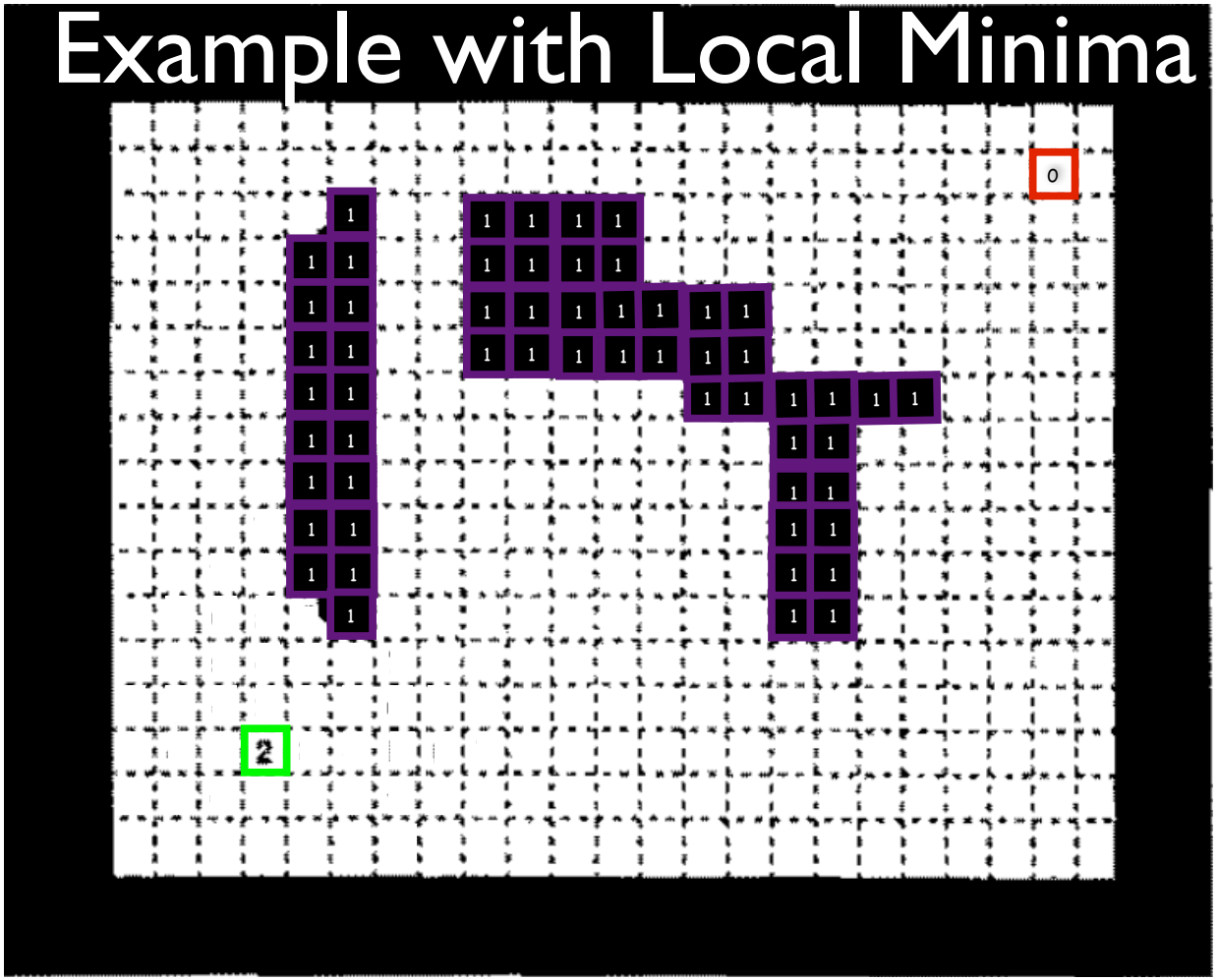


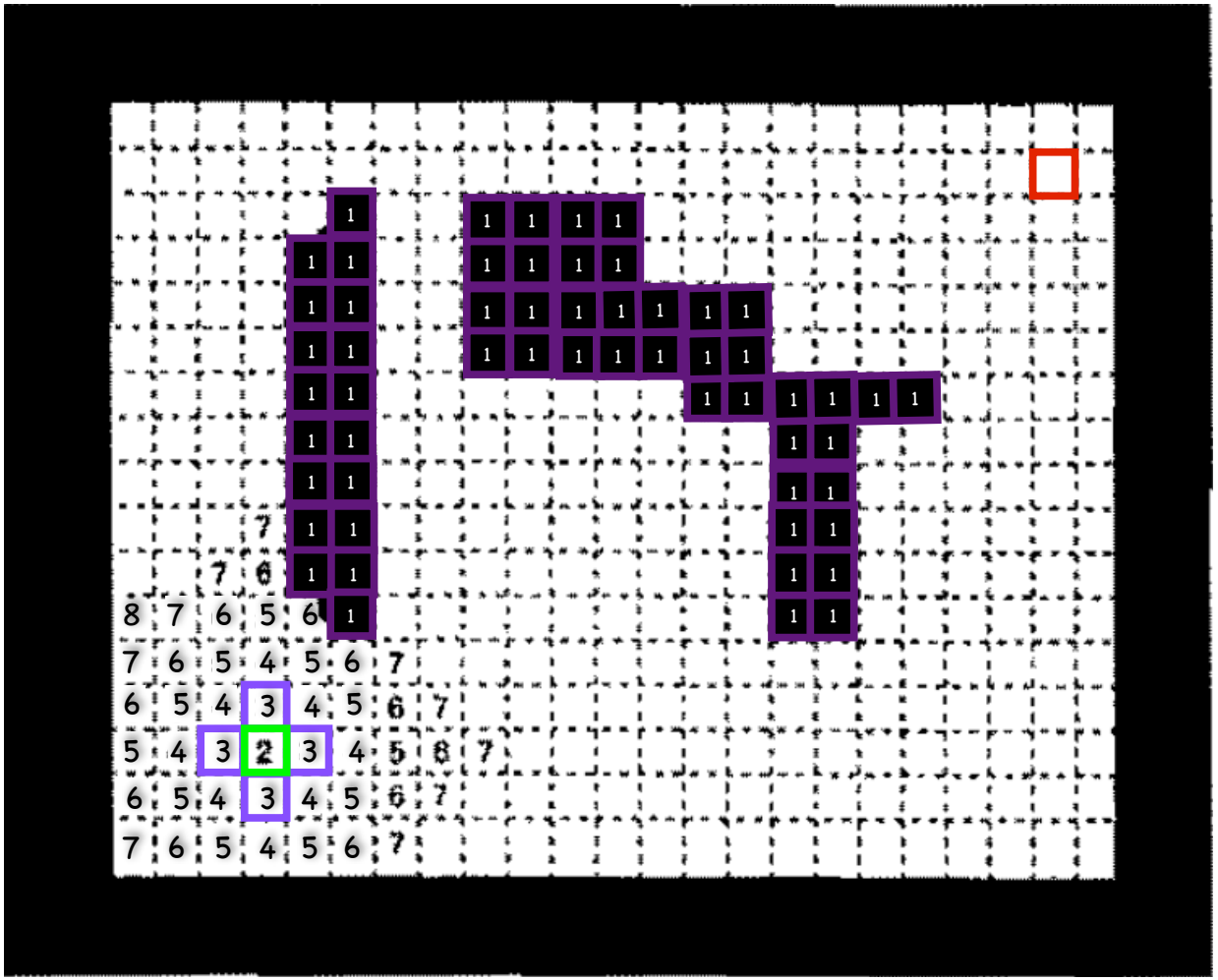
Once start reached, follow brushfire potential to goal

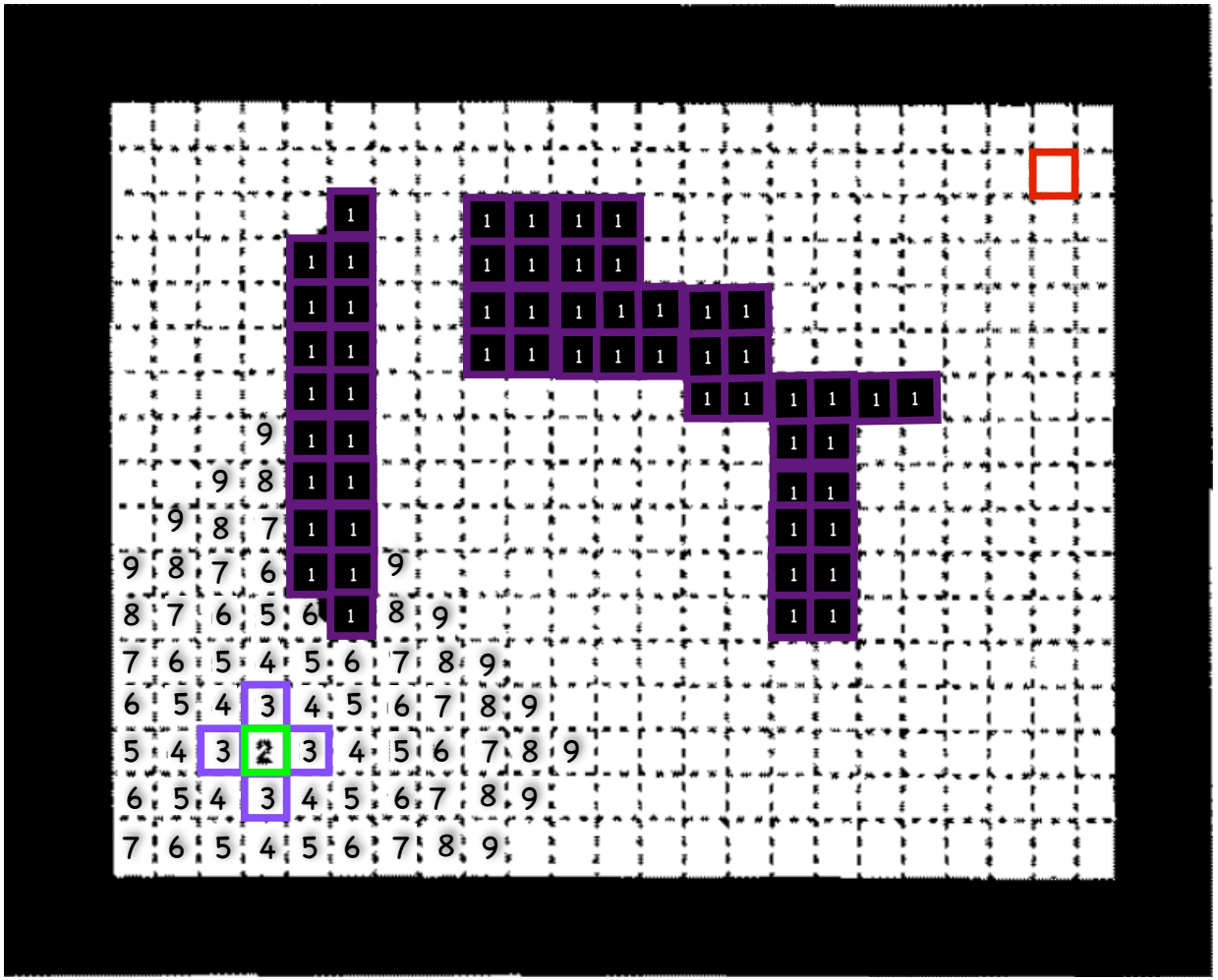


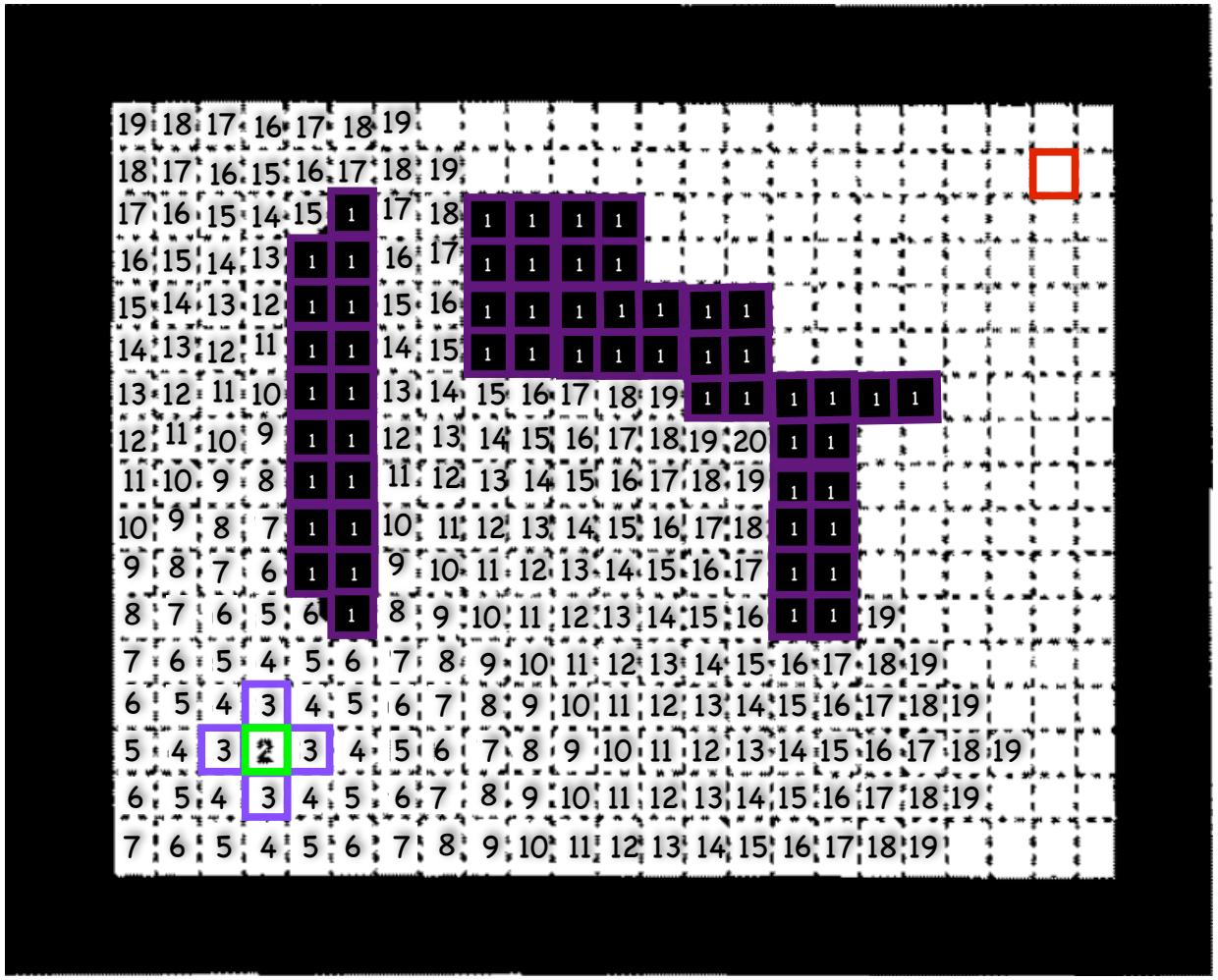
Example with Local Minima

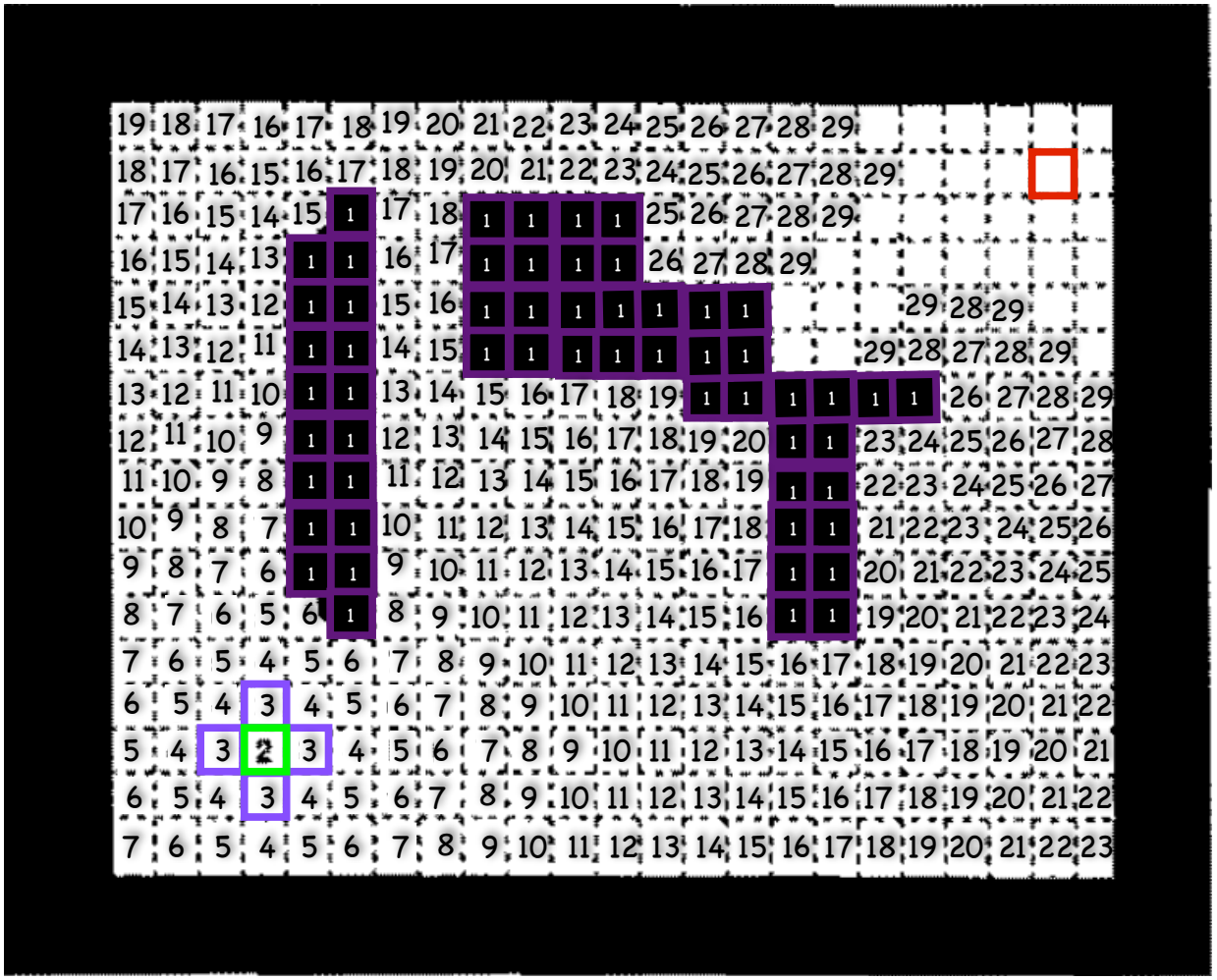
Example with Local Minima

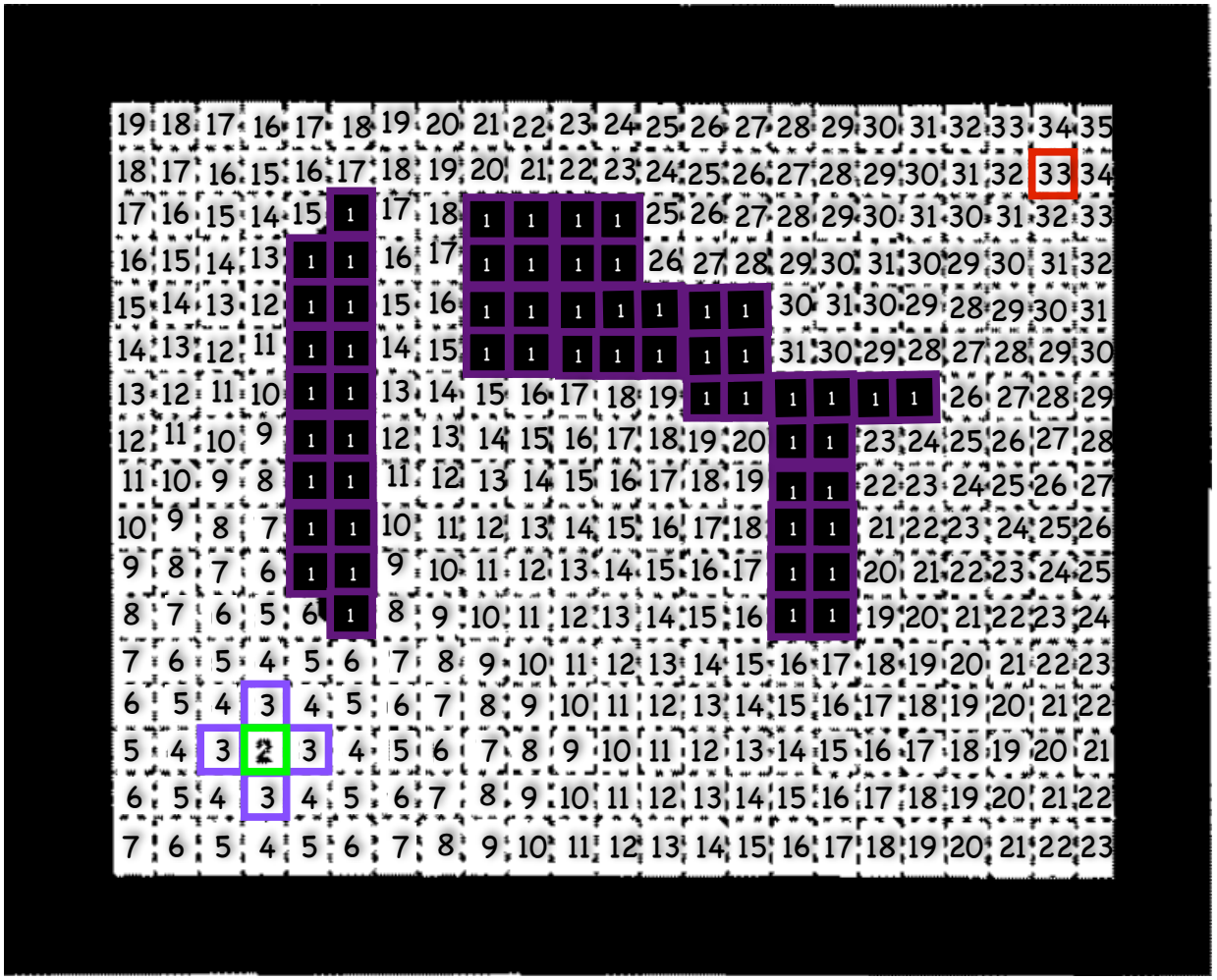


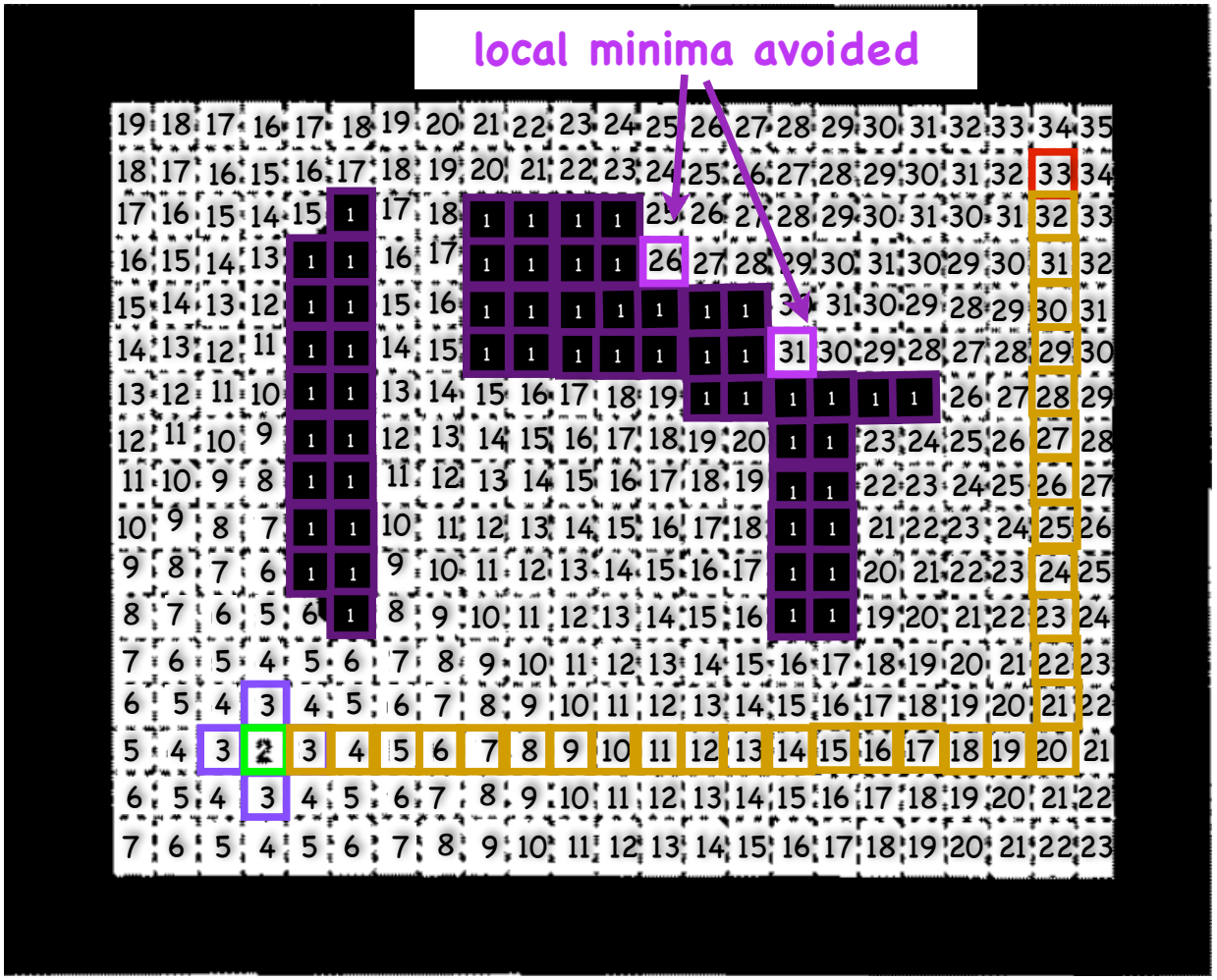











local minima avoided

Can we extend potential fields for arm navigation?

Potential Fields for Robot Arm

- Define endeffector goal as the attractive potential with cone potential
- Define repulsive potentials wrt. collision objects
  - Select points on robot links with “bowl” potential from nearest object
- Use manipulator Jacobian to transform potential at each point into velocites at robot joints
- Weighted sum of transformed velocities to generate control

Navigation Recap

Navigation Recap

Bug X
- Complete
- Non-optimal
- Planar

Subsumption and FSMs
- Fast but not adaptive
- Emphasis on good design

Potential Fields
- Complete in special cases
- Non-optimal
- General C-spaces
- Scales w/dimensionality

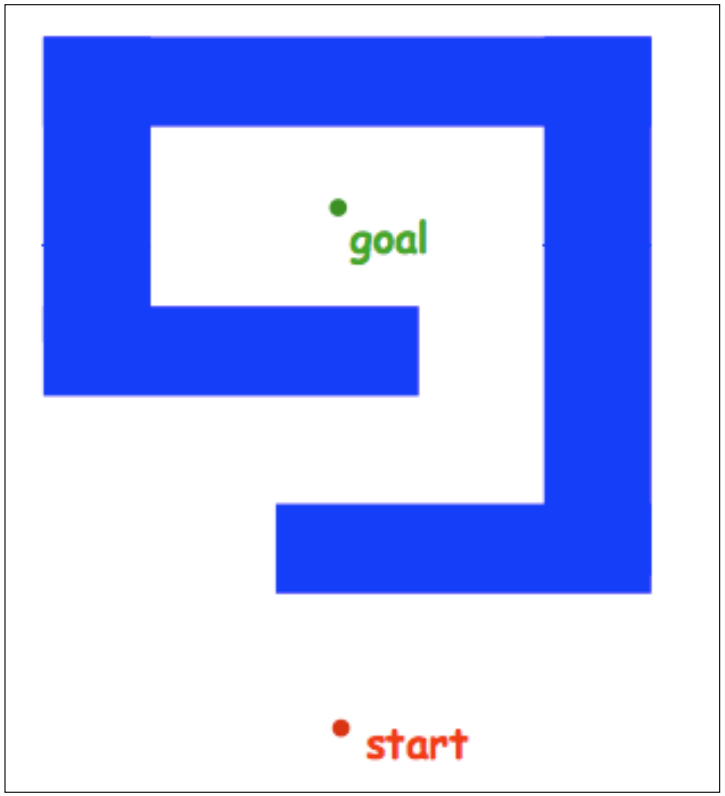
Grid Search/Wavefront
- Complete
- General C-spaces
- Limited dimensionality

Random walk
- Will find path eventually

Sampling roadmaps/RRT
- Probabilistically complete
- General
- Tractable (with good sampling)
- Scales w/dimensionality
- Not necessarily optimal

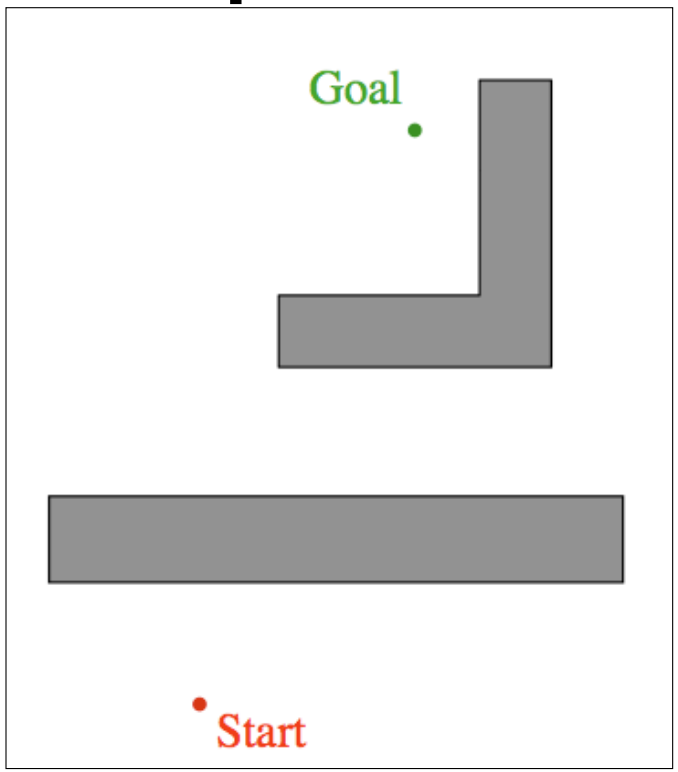
Compare: BugX examples

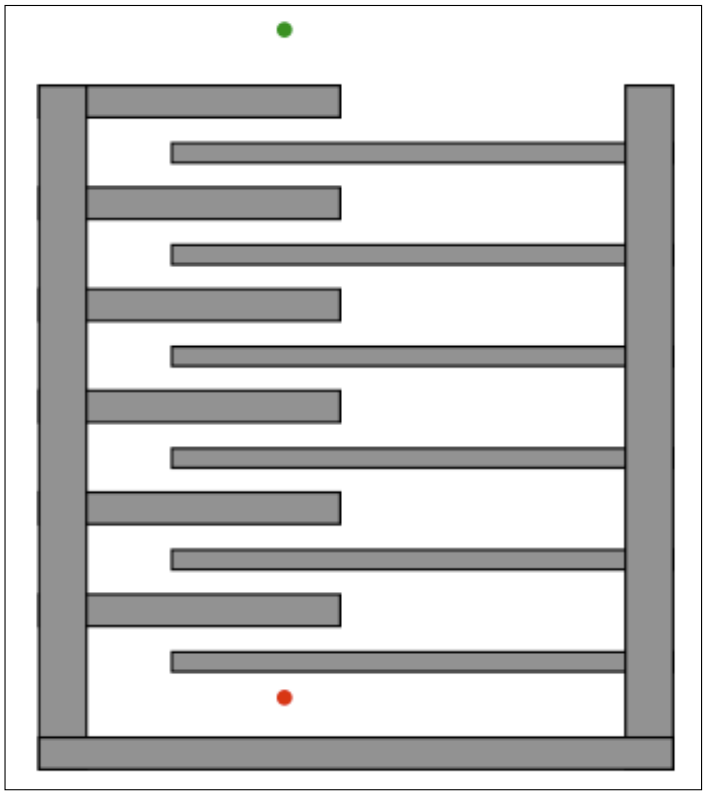
Describe likely paths from potential field and wavefront planners



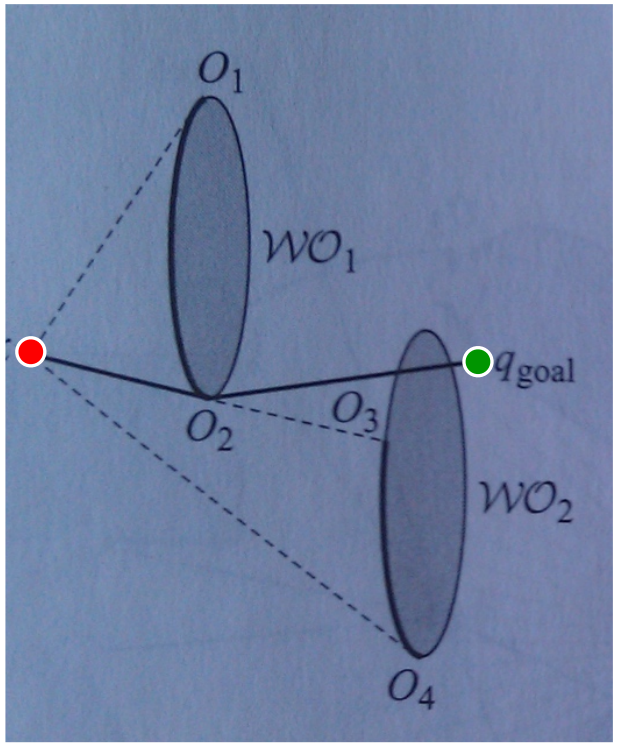
Compare: BugX examples

Describe likely paths from potential field and wavefront planners

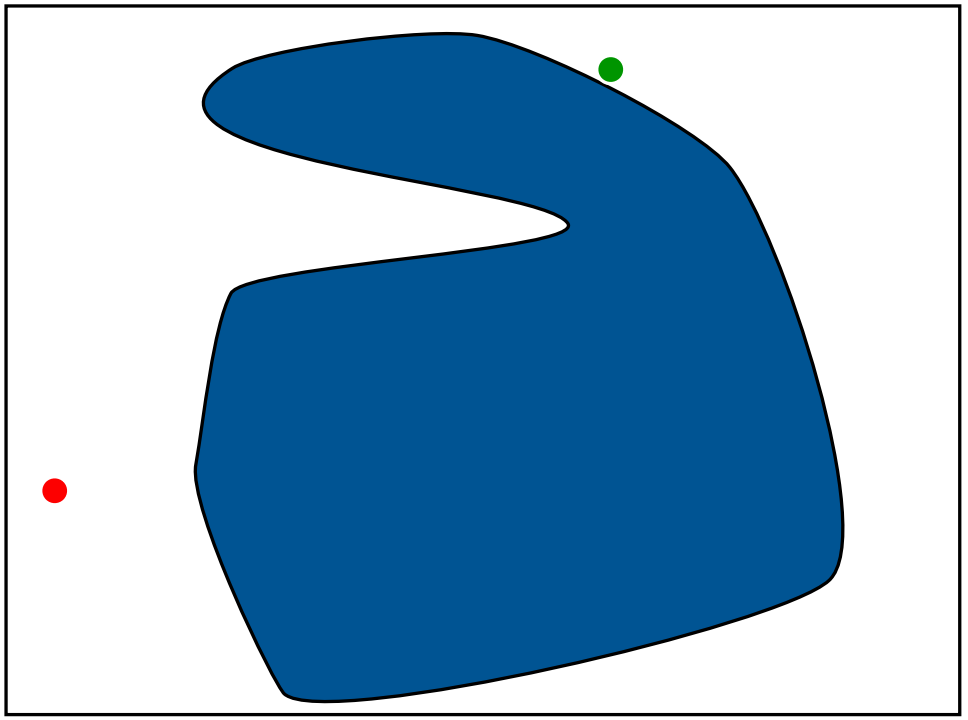




Compare: BugX examples

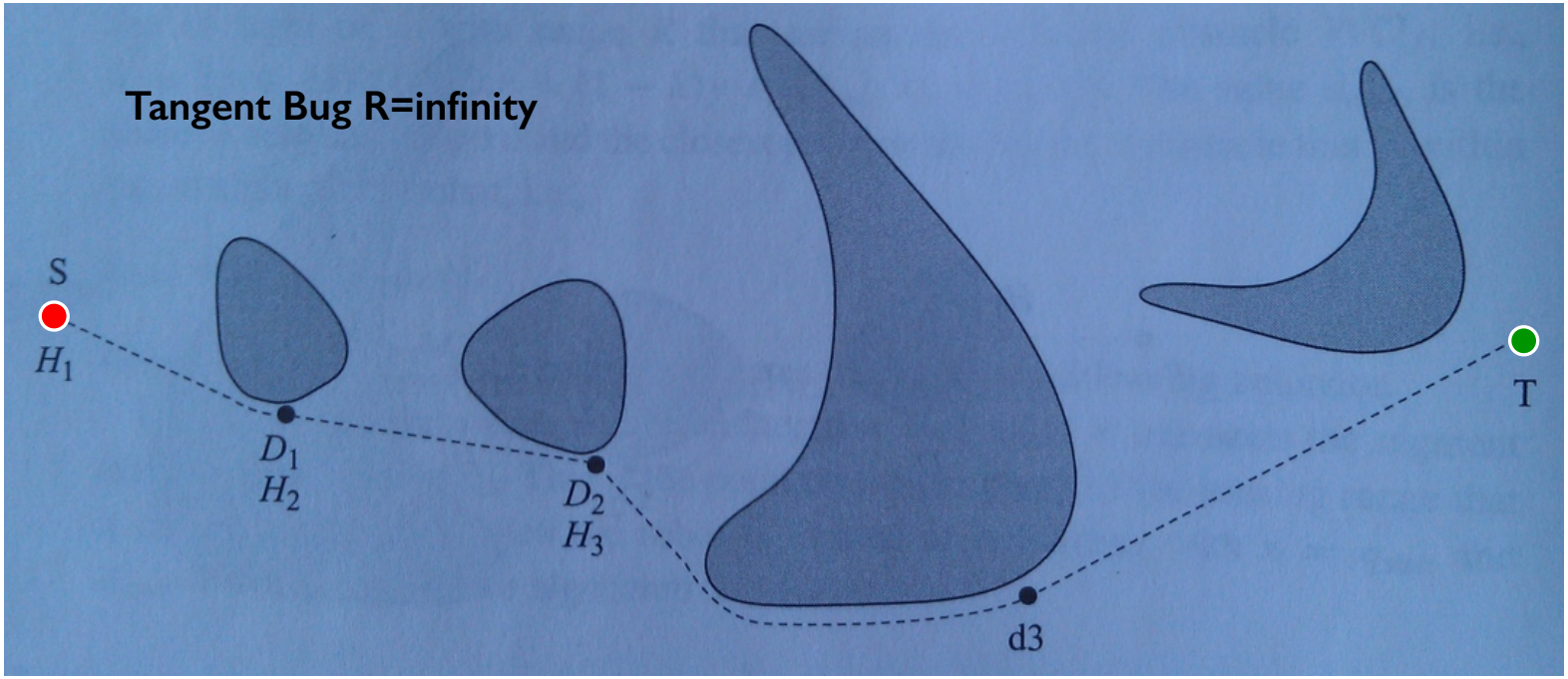


Describe likely paths from potential field and wavefront planners



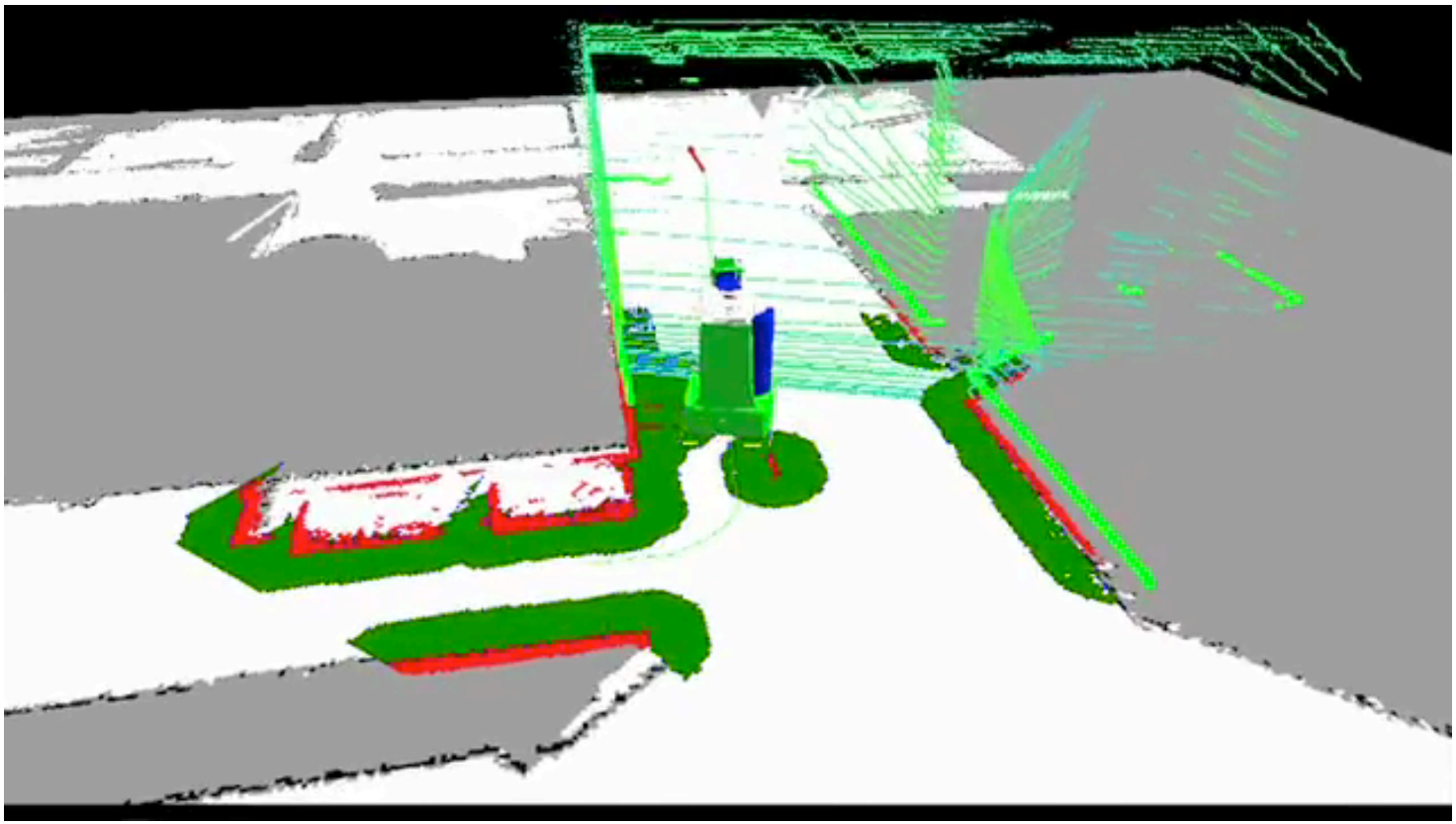
Compare: BugX examples

Describe likely paths from potential field and wavefront planners



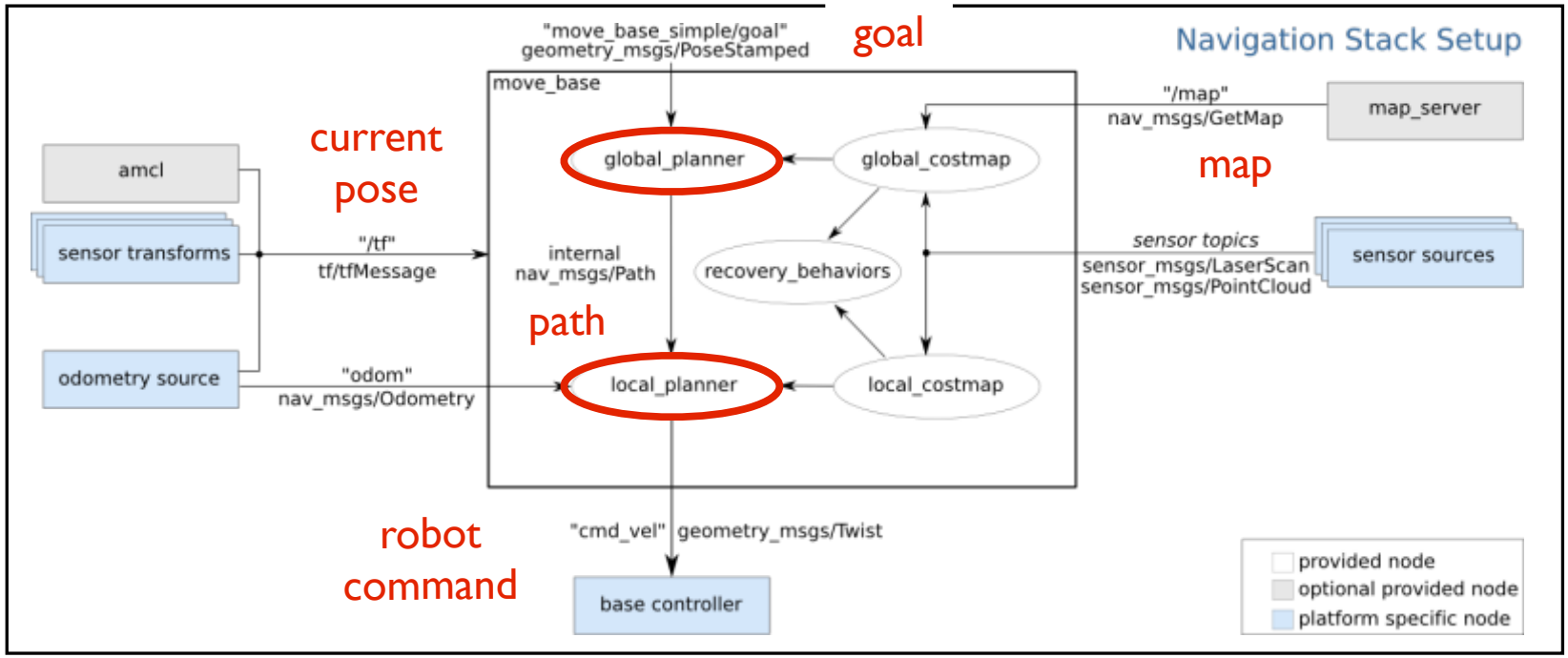
Navigation in ROS

ROS Navigation Stack



http://www.ros.org/wiki/navigation

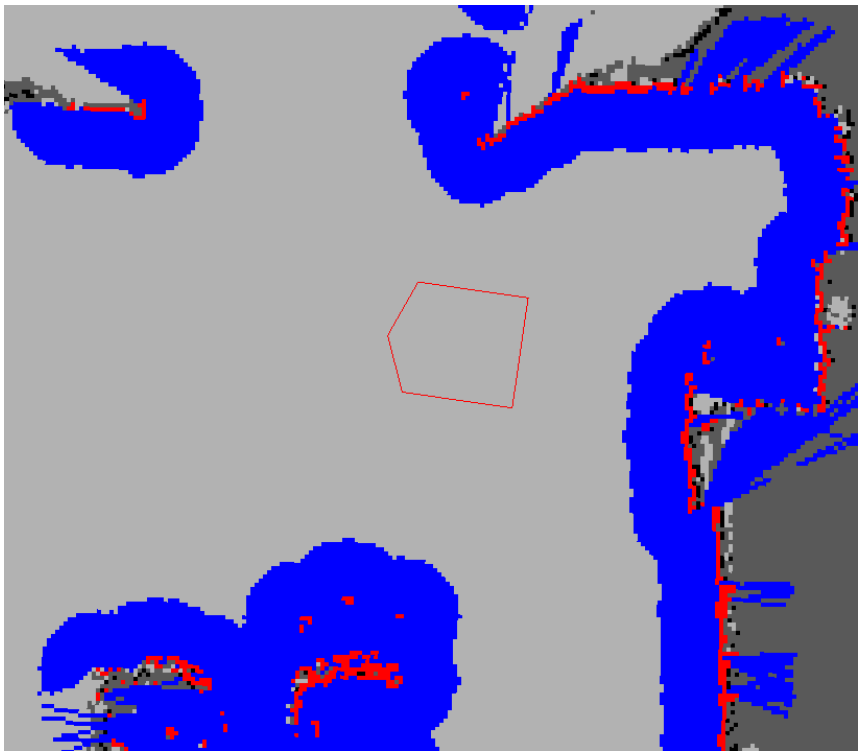
move_base



- Core of nav_stack for planning and motion control

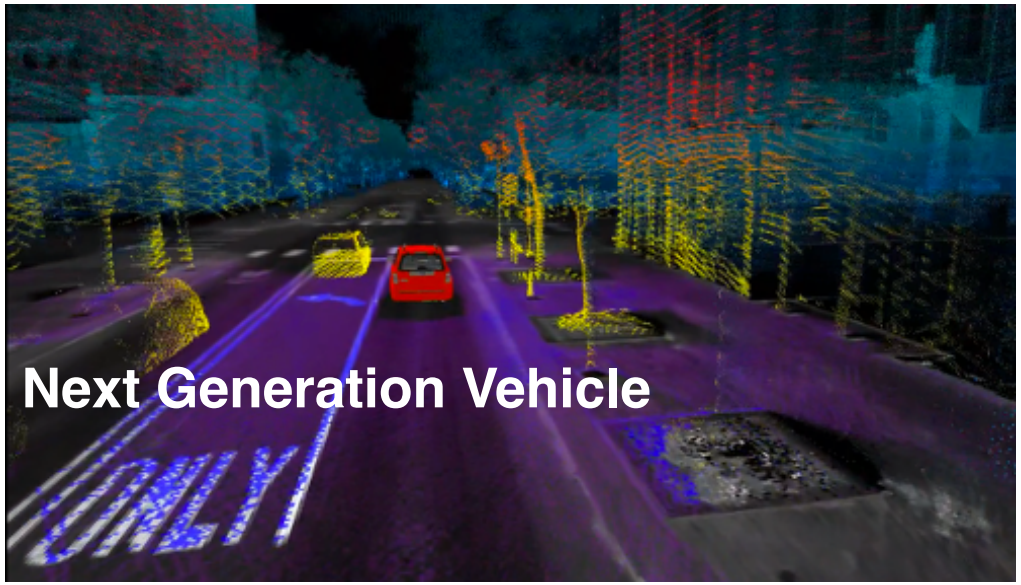
Costmap in ROS

- Treat as a Minkowski sum over the given map



http://www.ros.org/wiki/costmap_2d

Navigation in LCM

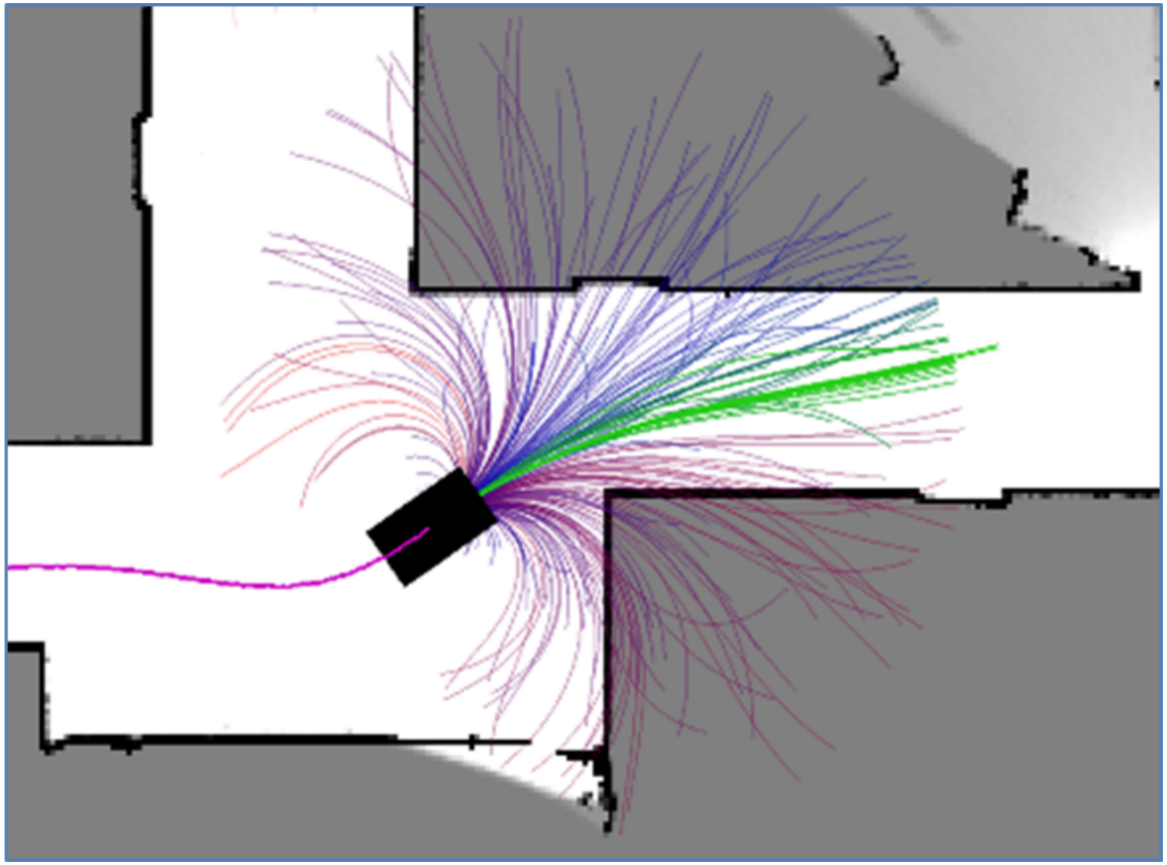

Next Generation Vehicle


BotLab EECS 467 / ROB 550


MagicBots

[Park, Johnson, Kuipers 2012]

Vulcan Navigation



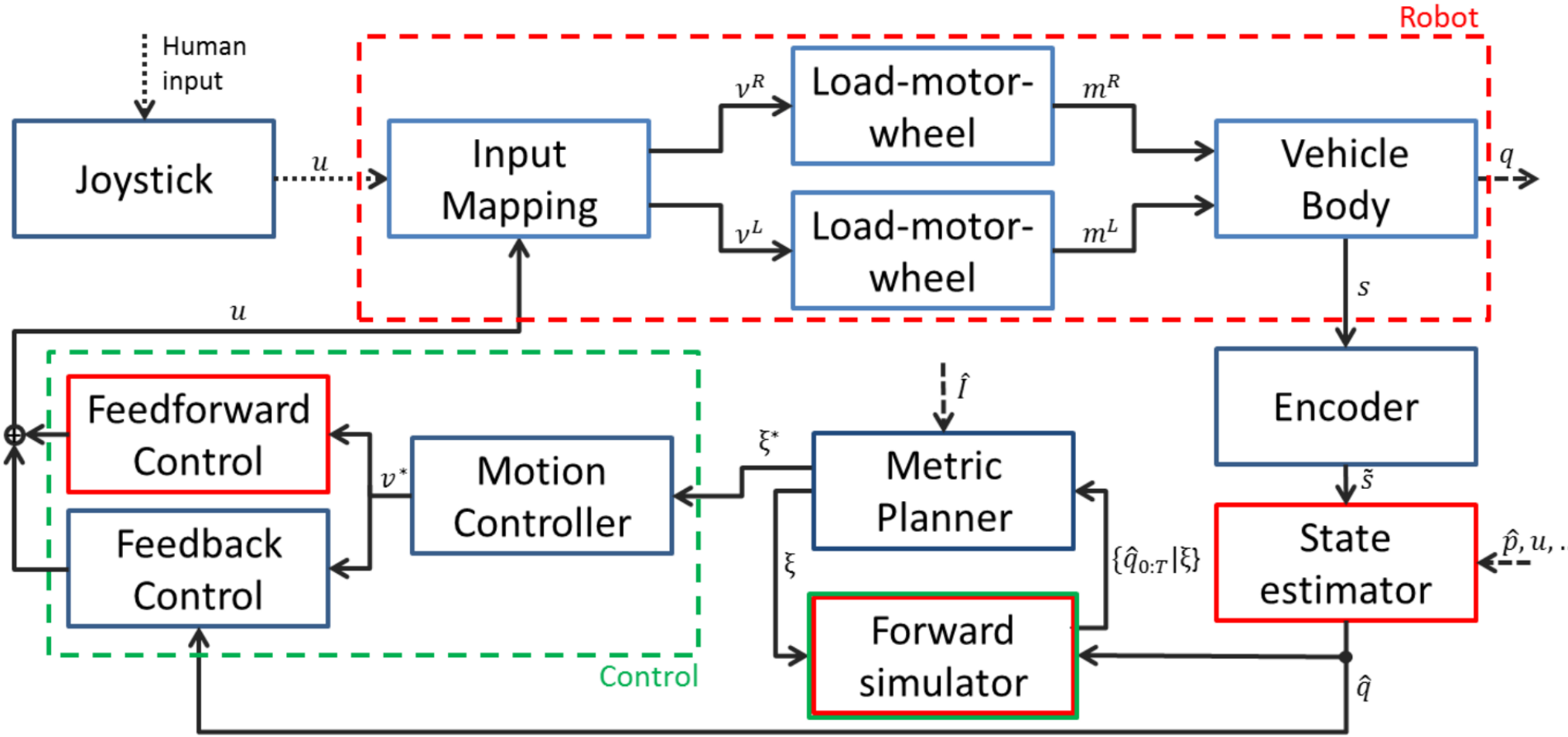
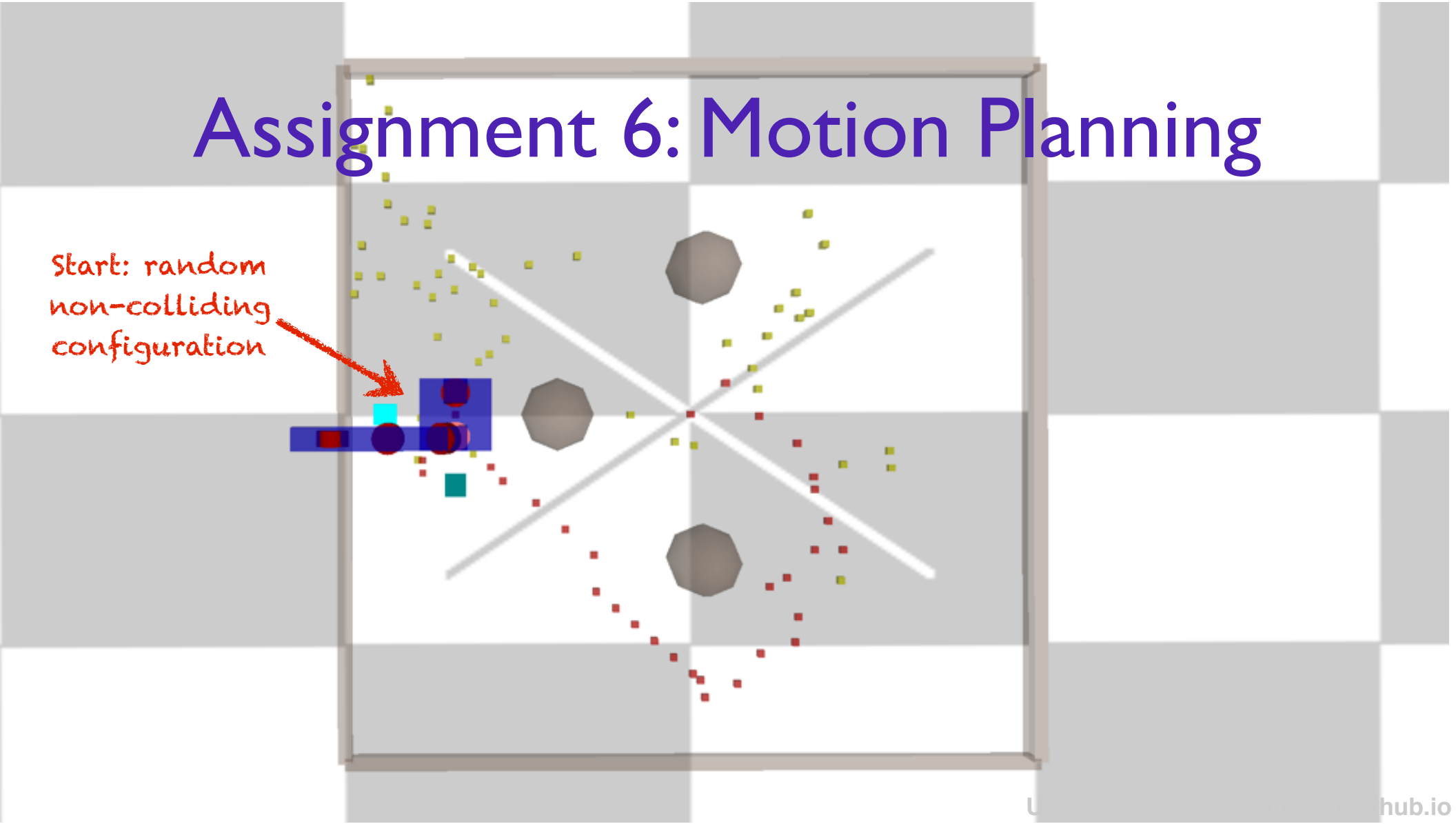


Figure A.1: Vulcan system diagram.

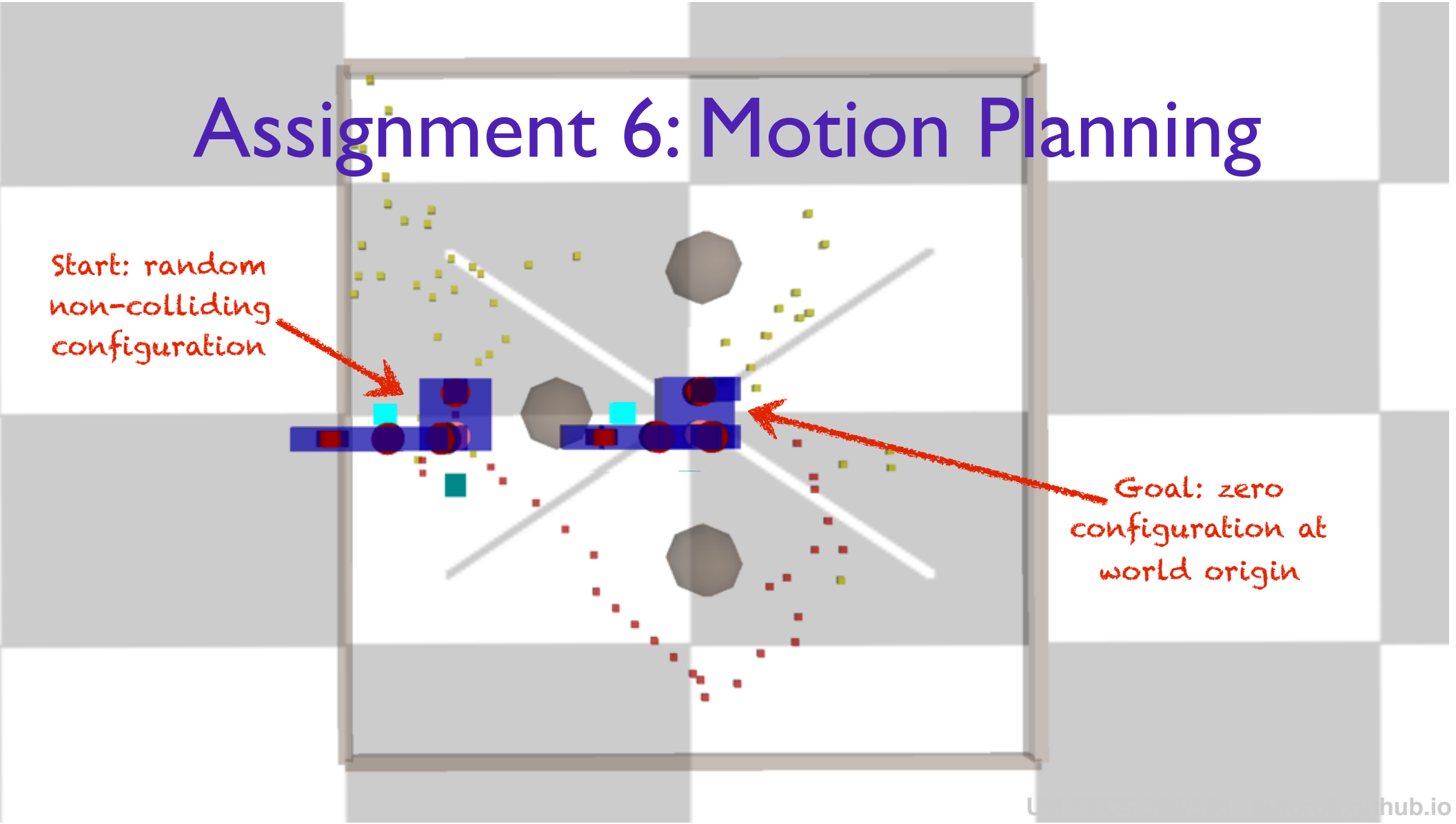
Assignment 6: Motion Planning

- Generate a collision free motion plan to the world origin and zero joint angle configuration

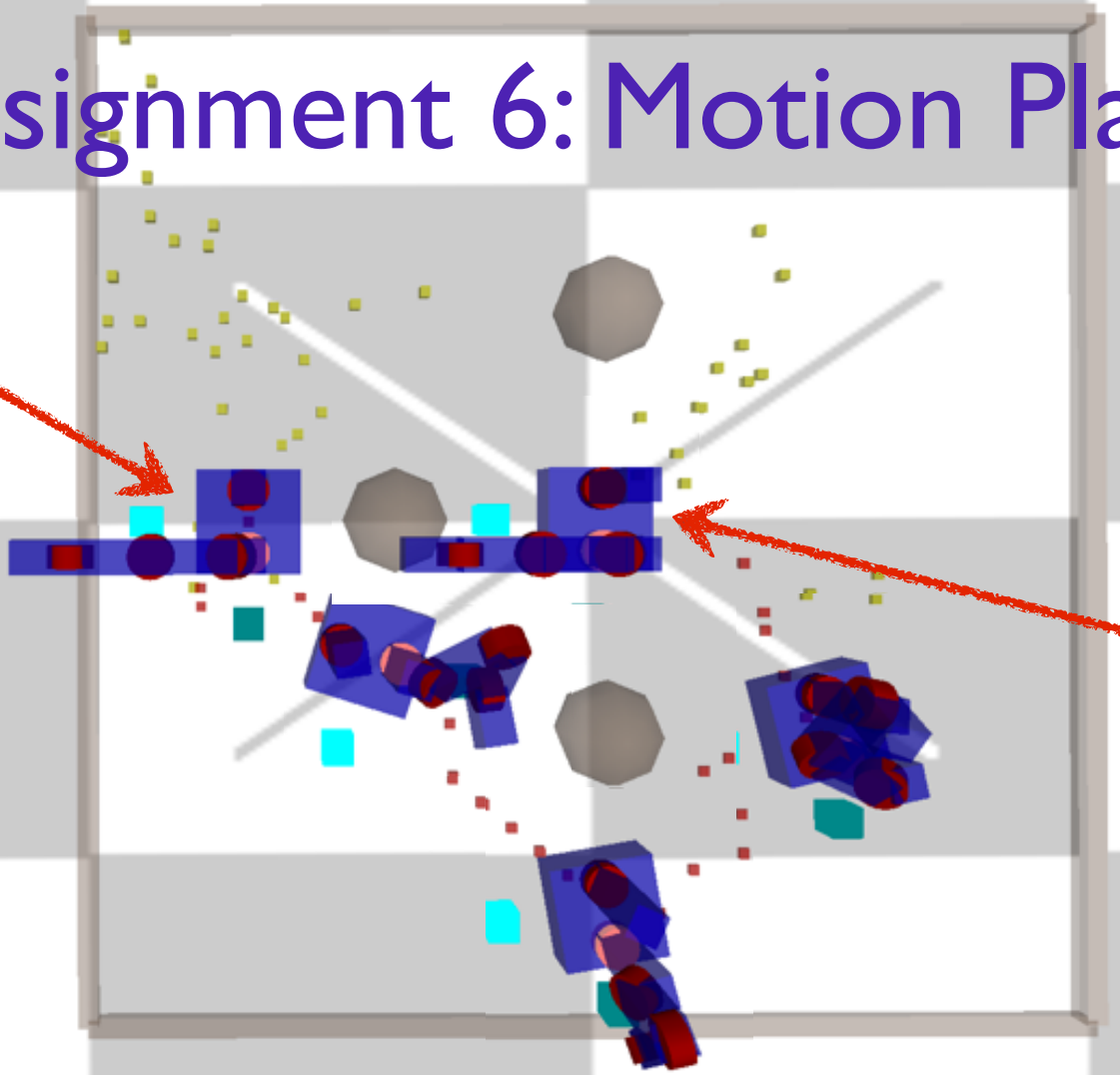
Assignment 6: Motion Planning



Assignment 6: Motion Planning



Assignment 6: Motion Planning



Start: random non-colliding configuration

Goal: zero configuration at world origin

Generate collision-free motion plan

GitHub, Inc. (US) https://github.com/ohseejay/kineval-stencil-fall16 rrt connect

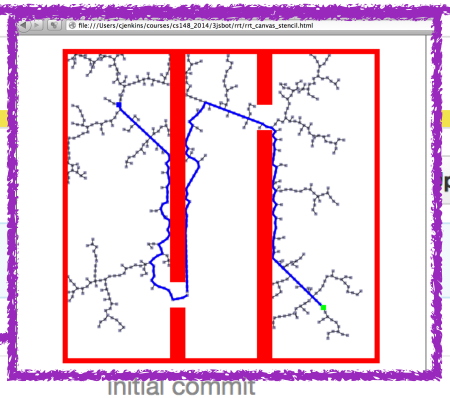
ohseejay / **kineval-stencil-fall16**

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0

Stencil code for KinEval (Kinematic Evaluator) for robo

Recommended starting point:
Update code stencil from Path Planning project



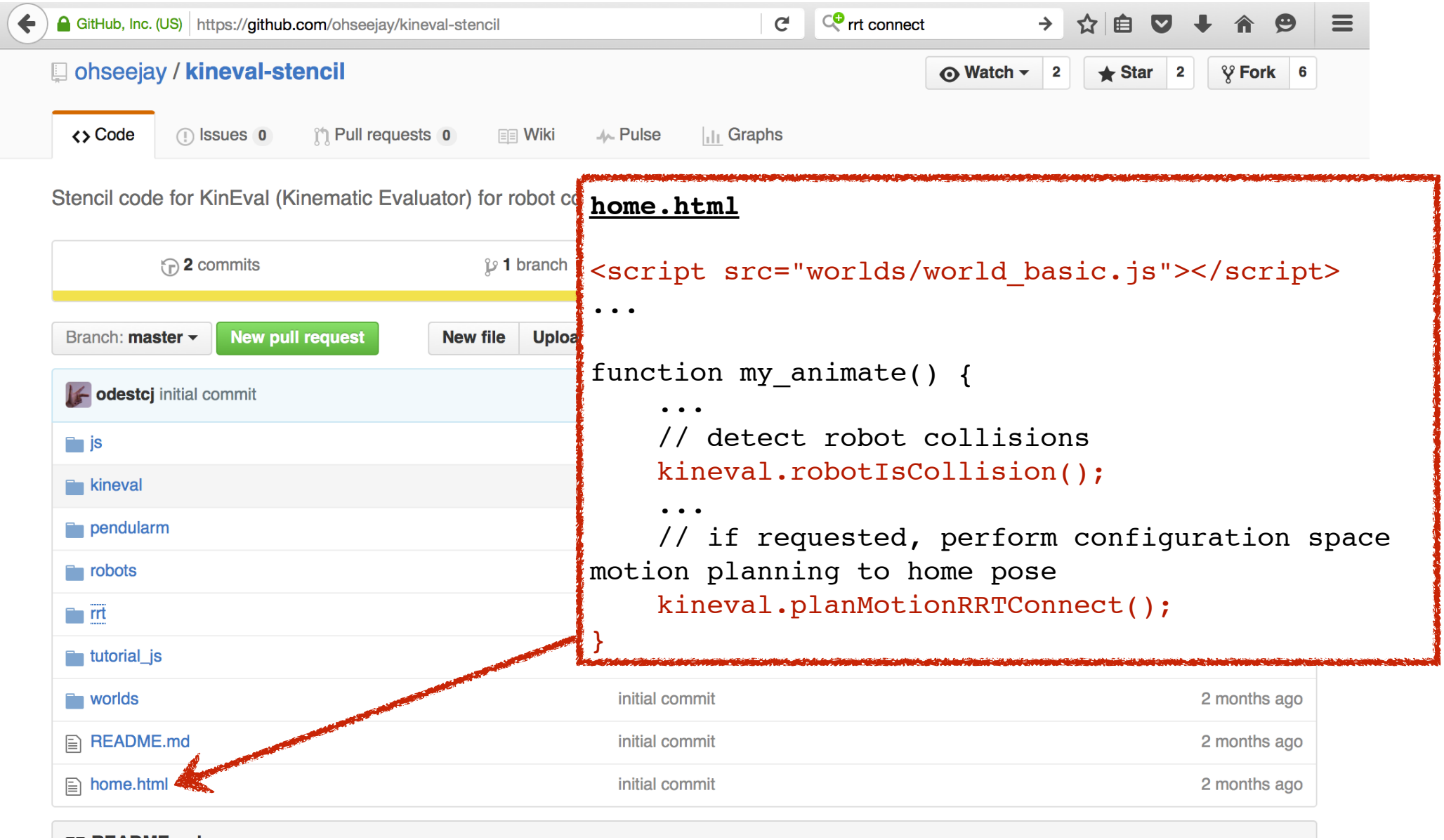
2 commits 1 branch 2 contributors

Branch: master New pull request pload files Find file Clone or download

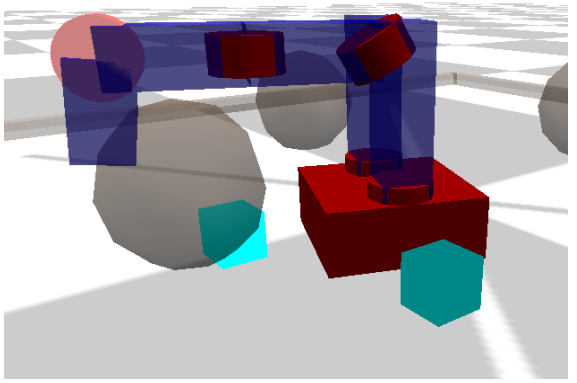
odestcj initial commit Latest commit 5fd521e 26 days ago

js		26 days ago
kineval	initial commit	26 days ago
project_pathplan	initial commit	26 days ago
project_pendularm		
robots		
tutorial_heapsort		
tutorial_js	initial commit	26 days ago
worlds	initial commit	26 days ago
README.md	initial commit	26 days ago
home.html	initial commit	26 days ago

various 3D worlds for testing robots included by:
home.html?world=worlds/world_local_minima.js


GitHub, Inc. (US)
https://github.com/ohseejay/kineval-stencil
rrt connect
ohseejay / kineval-stencil
Watch 2
Star 2
Fork 6
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Stencil code for KinEval (Kinematic Evaluator) for robot c
2 commits
1 branch
Branch: master
New pull request
New file
Uploa
odestcj initial commit
js
kineval
pendularm
robots
rrt
tutorial_js
worlds
initial commit
2 months ago
README.md
initial commit
2 months ago
home.html
initial commit
2 months ago
home.html
<script src="worlds/world_basic.js"></script>
...
function my_animate() {
...
// detect robot collisions
kineval.robotIsCollision();
...
// if requested, perform configuration space motion planning to home pose
kineval.planMotionRRTConnect();
}

world file can be alternatively loaded by a script tag



home.html

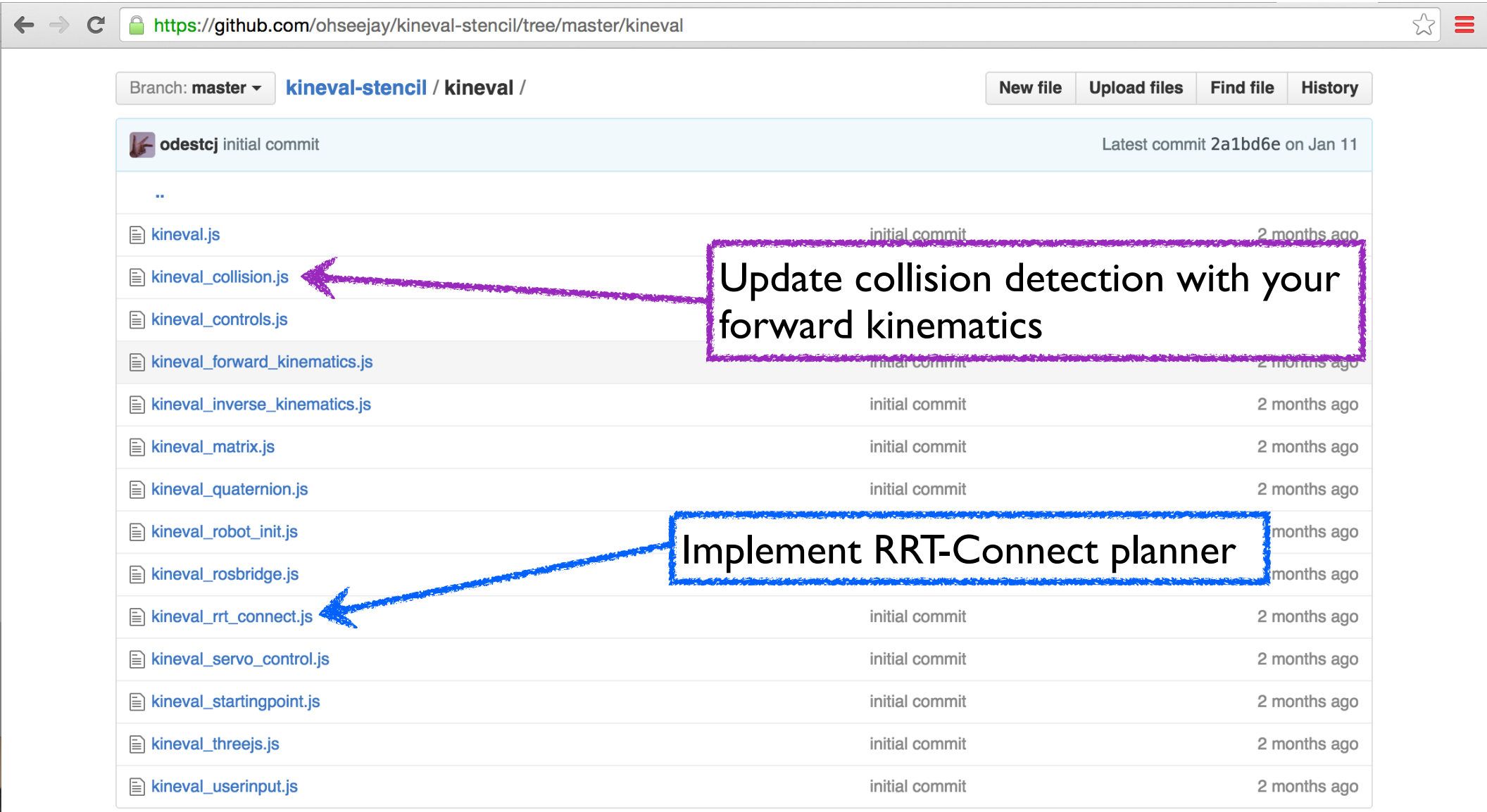
```
<script src="worlds/world_basic.js"></script>
...

function my_animate() {
    ...
    // detect robot collisions
    kineval.robotIsCollision();
    ...
    // if requested, perform configuration space motion planning to home pose
    kineval.planMotionRRTConnect();

}
```

detect if current configuration is in collision (colliding link turns red)

iterate motion planner


https://github.com/ohseejay/kineval-stencil/tree/master/kineval
Branch: master
kineval-stencil / kineval /
New file
Upload files
Find file
History
odestcj initial commit
Latest commit 2a1bd6e on Jan 11
..
kineval.js initial commit 2 months ago
kineval_collision.js
kineval_controls.js
kineval_forward_kinematics.js
kineval_inverse_kinematics.js initial commit 2 months ago
kineval_matrix.js initial commit 2 months ago
kineval_quaternion.js initial commit 2 months ago
kineval_robot_init.js
kineval_rosbridge.js
kineval_rrt_connect.js initial commit 2 months ago
kineval_servo_control.js initial commit 2 months ago
kineval_startingpoint.js initial commit 2 months ago
kineval_threejs.js initial commit 2 months ago
kineval_userinput.js initial commit 2 months ago
Update collision detection with your forward kinematics
Implement RRT-Connect planner

kineval_collision.js

input: q (robot configuration)
output: false (for no collision)
or name of link in collision

```
kineval.poseIsCollision = function robot_collision_test(q) {
    // perform collision test of robot geometry against planning world

    // test base origin (not extents) against world boundary extents
    if ((q[0]<robot_boundary[0][0])||(q[0]>robot_boundary[1][0])||
        (q[2]<robot_boundary[0][2])||(q[2]>robot_boundary[1][2]))
        return robot.base;

    // traverse robot kinematics to test each body for collision
    // STENCIL: implement forward kinematics for collision detection
    return robot_collision_forward_kinematics(q);

}
```

"electric fence" world boundary detection is provided

Uncomment this call; and
Implement this function to traverse forward kinematics for collisions;
Use provided link collision function for AABB test of each link

kineval_rrt_connect.js

```
kineval.robotRRTPlannerInit = function robot_rrt_planner_init() {

   // set start (q_init) and goal (q_goal) configurations

}

function robot_rrt_planner_iterate() {

    var i;
    rrt_alg = 1;  // 0: basic rrt (OPTIONAL), 1: rrt_connect (REQUIRED)

    if (rrt_iterate && (Date.now()-cur_time > 10)) {
        cur_time = Date.now();

        // implement one planning iteration here with calls to rrt_extend, etc.

        // if plan found, store configuration sequence in kineval.motion_plan
        //   kineval.motion_plan[kineval.motion_plan_traversal_index]
    }
}
```

make sure to test against all provided worlds!