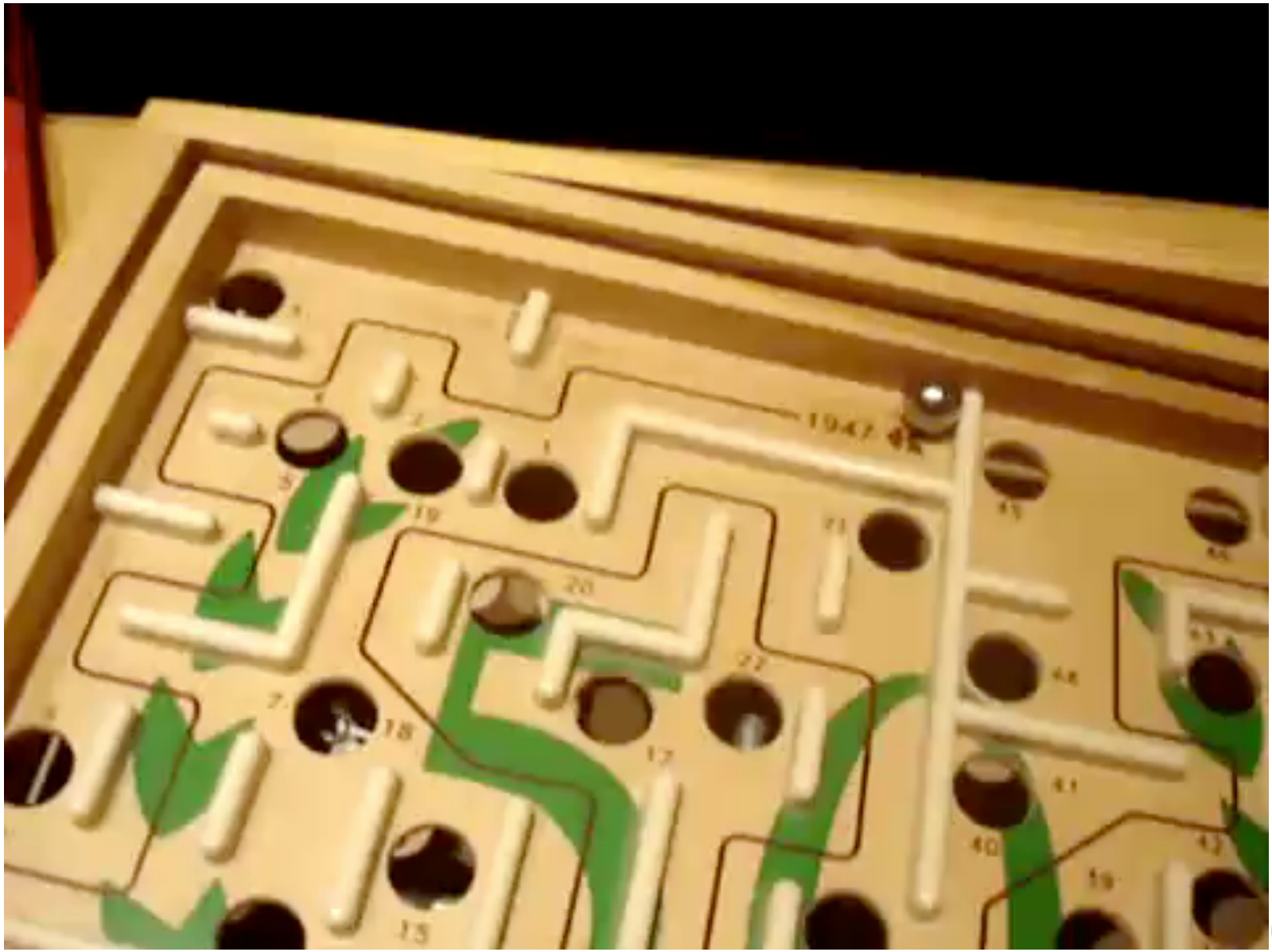
Potential fields

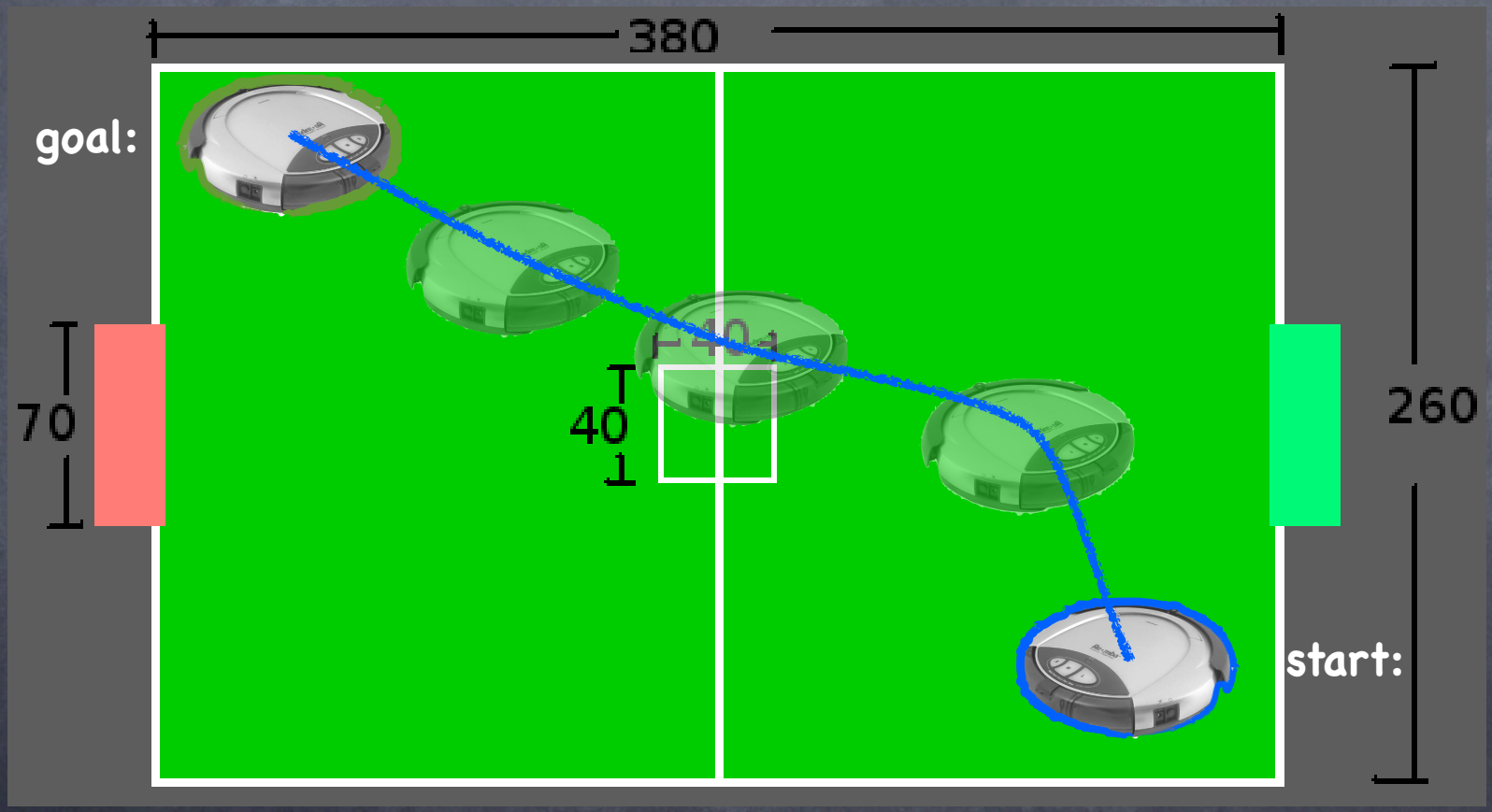
follow your potential

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**

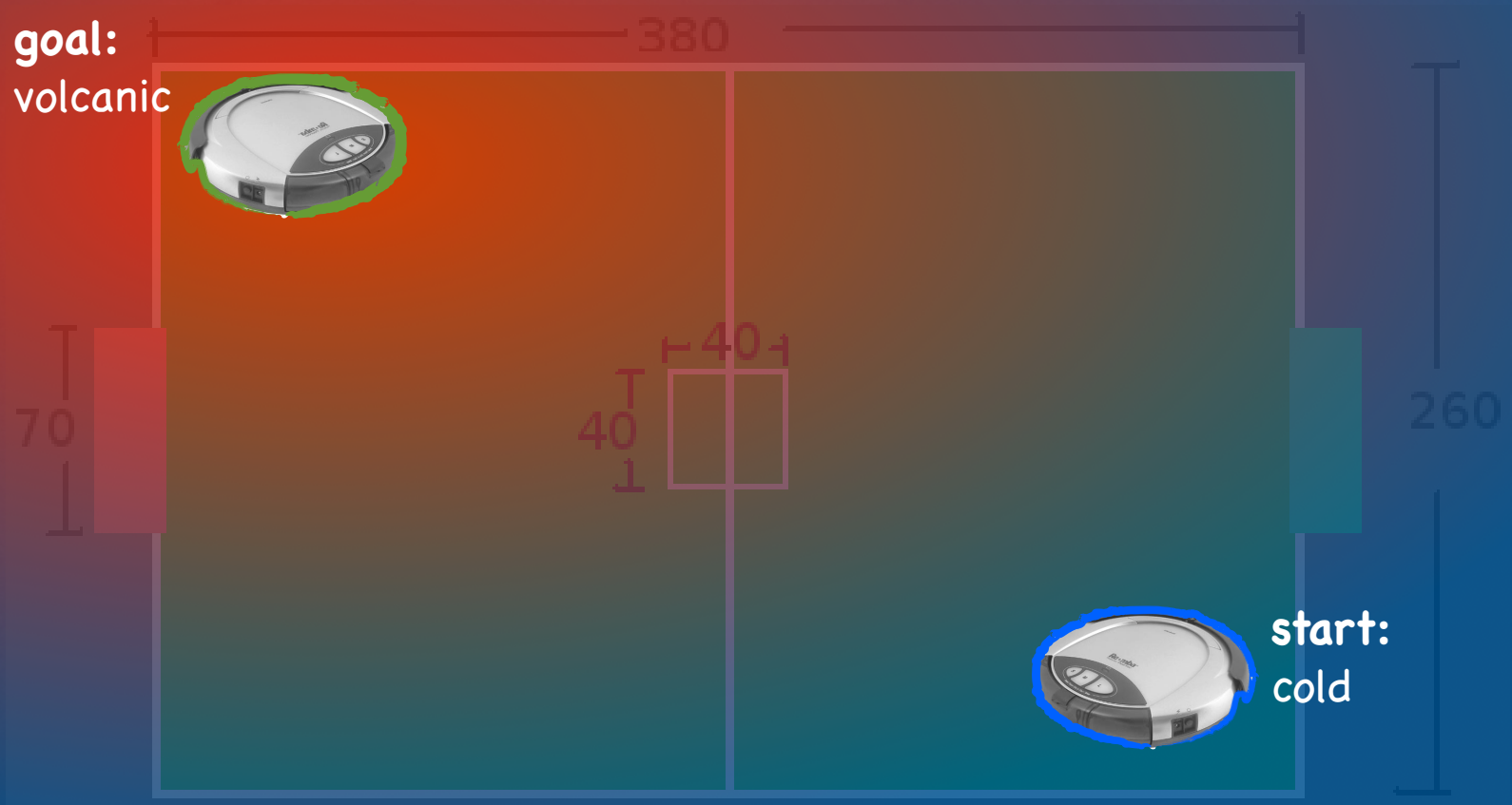


Navigation (again)



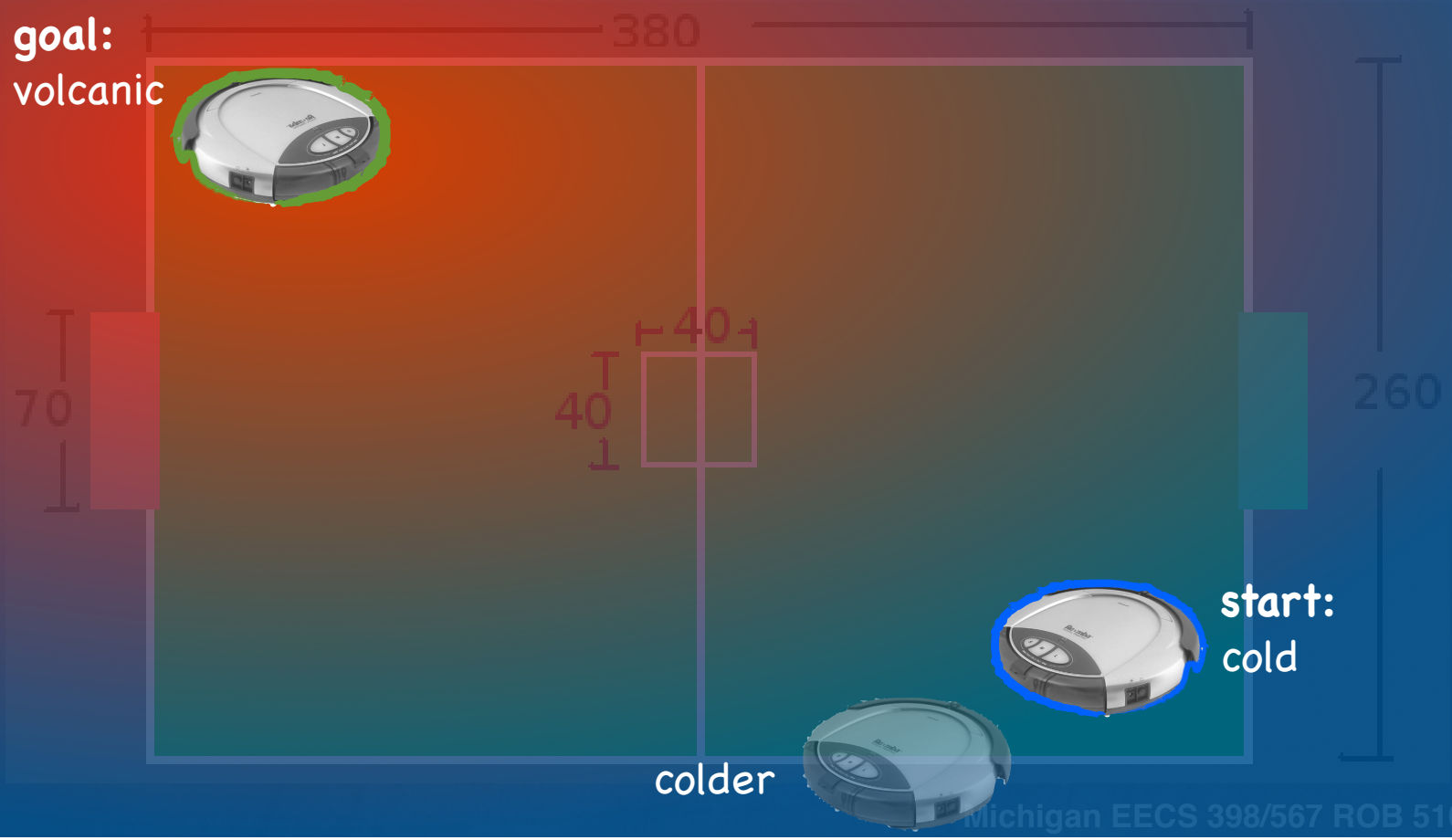
Potential field

(like a game of "warmer-colder")



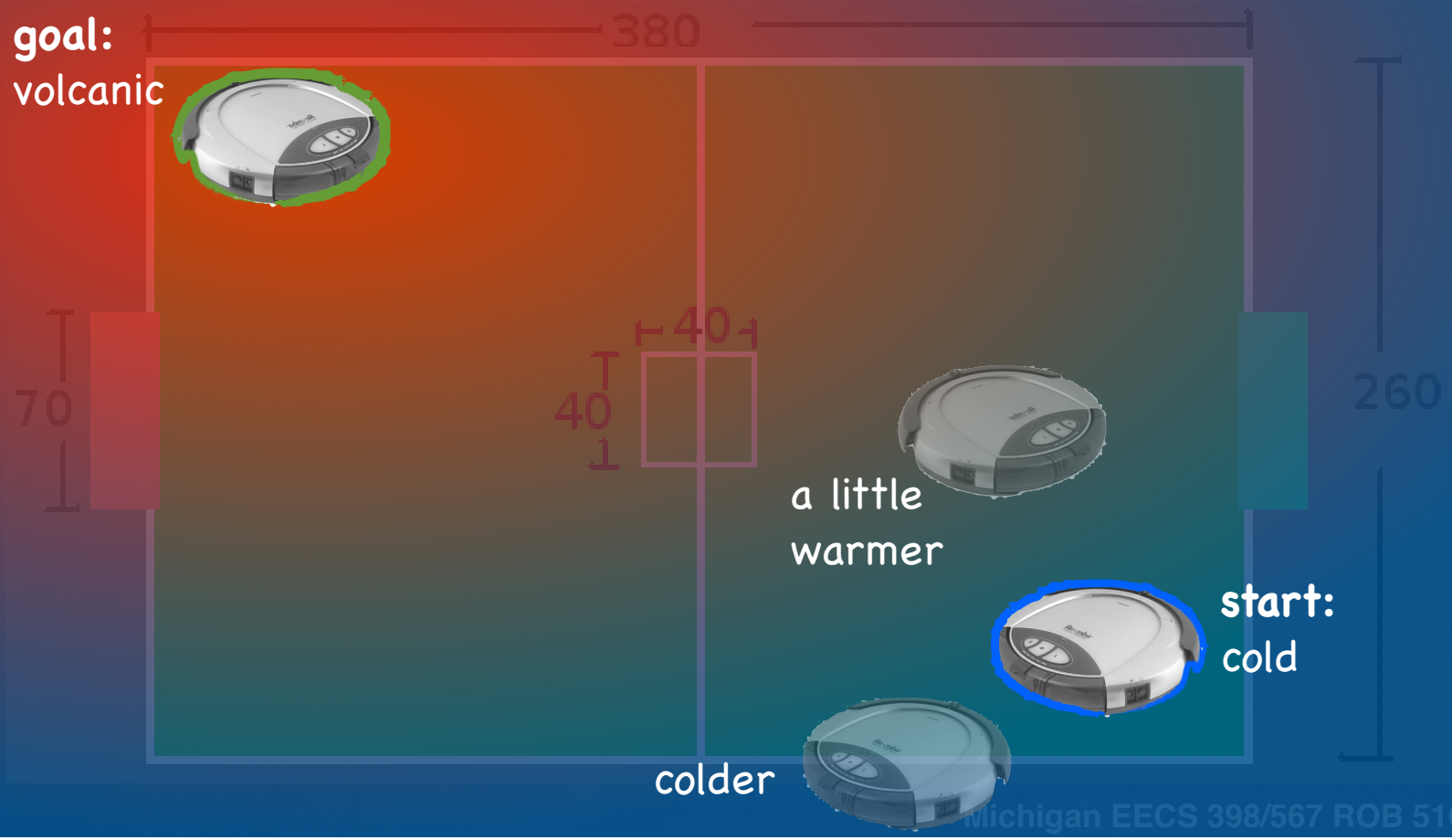
Potential field

(like a game of "warmer-colder")



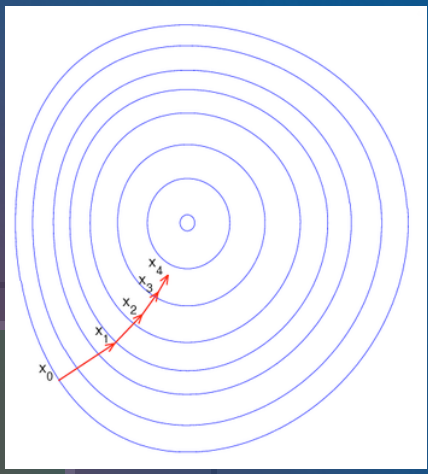
Potential field

(like a game of "warmer-colder")



Potential field

(like a game of "warmer-colder")



Gradient descent:
Energy potential converges at goal

goal:
volcanic

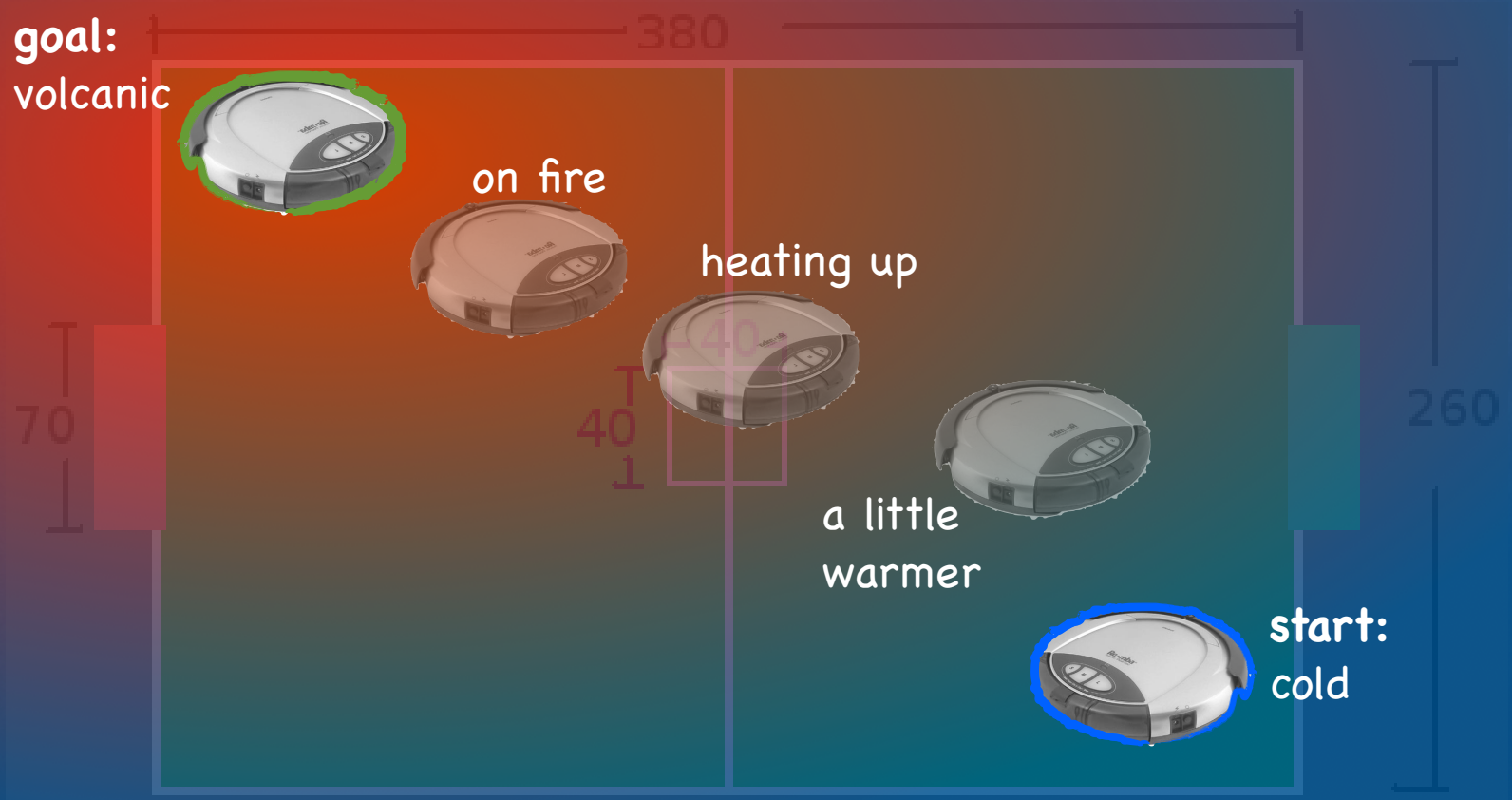
a little warmer

start:
cold

colder

Potential field

(like a game of "warmer-colder")



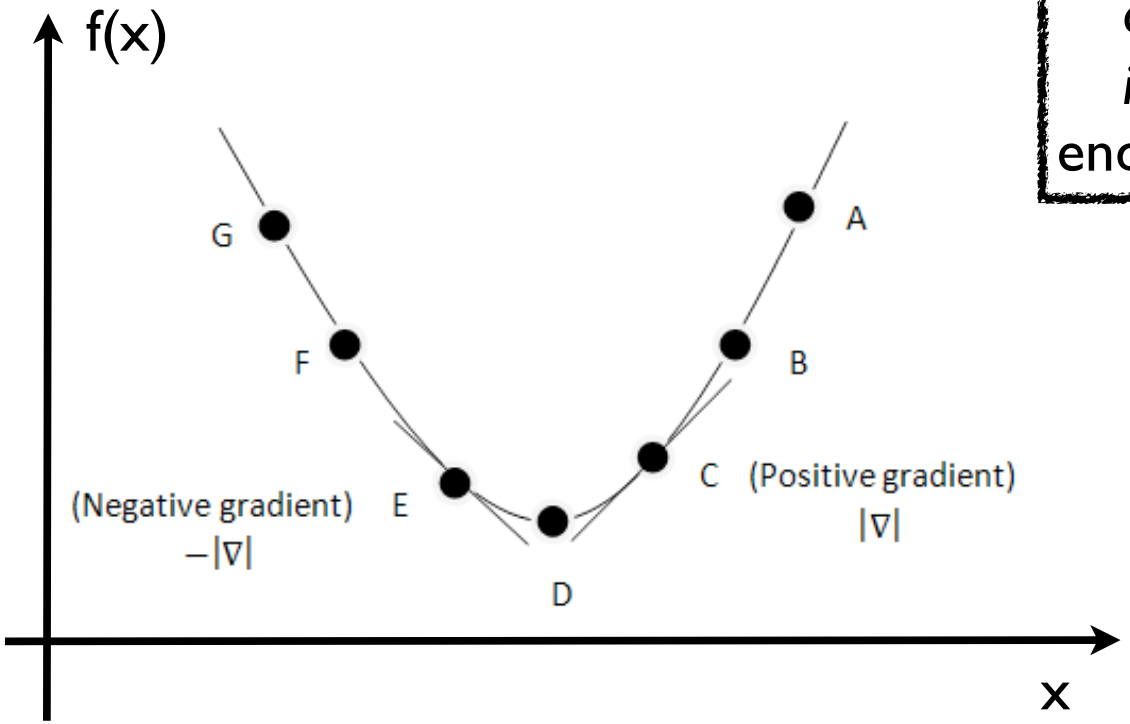
How do we define a potential field?

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

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

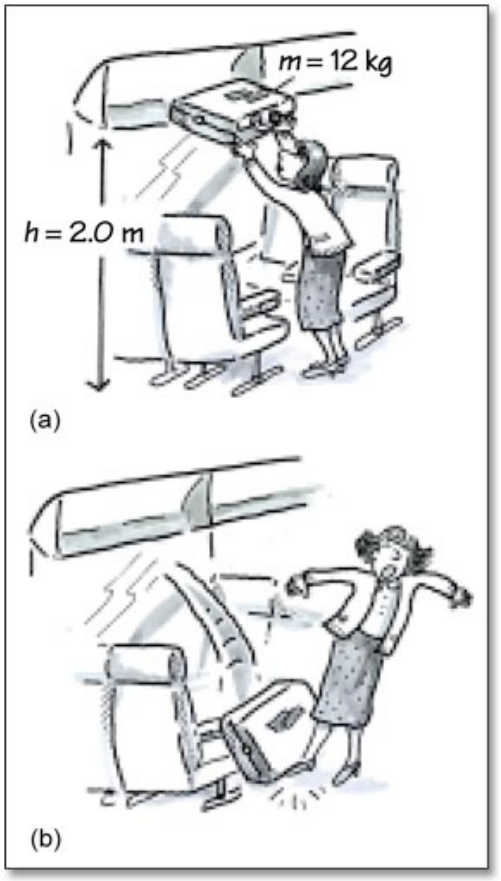
  $i \leftarrow i+1$

end

Derivative assumed to be direction of steepest ascent away from goal

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

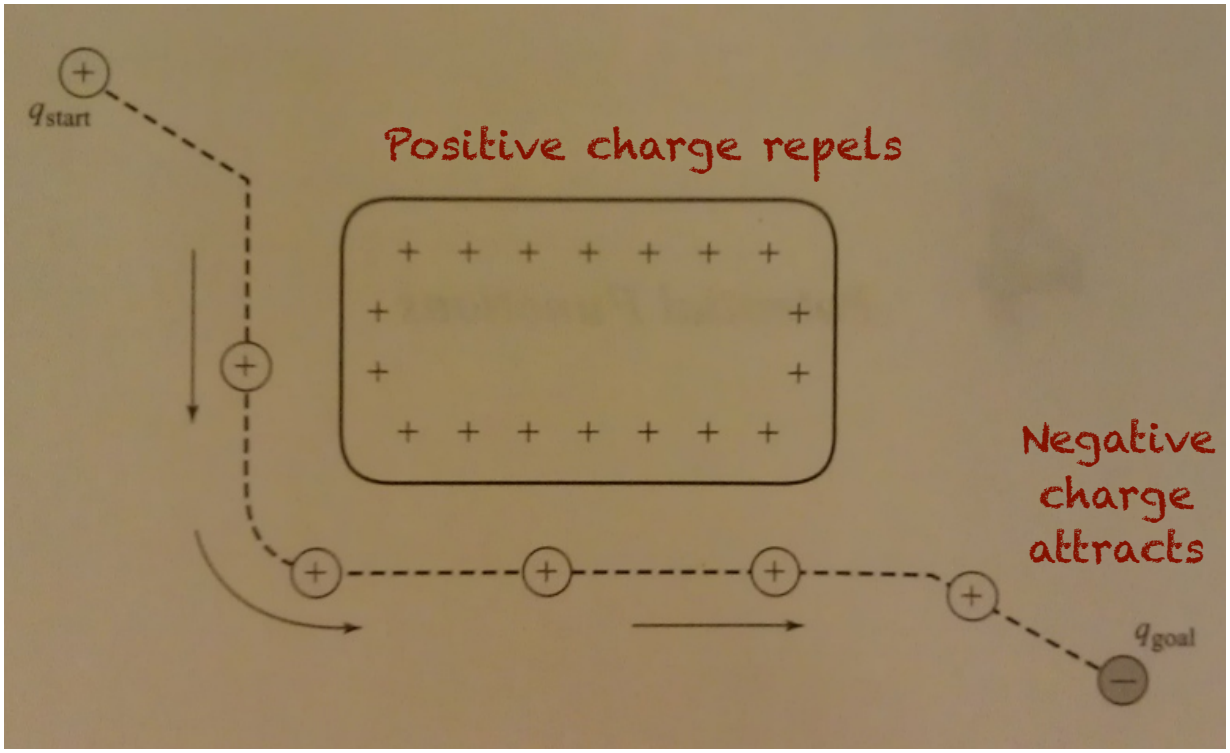
Potential Energy



- 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



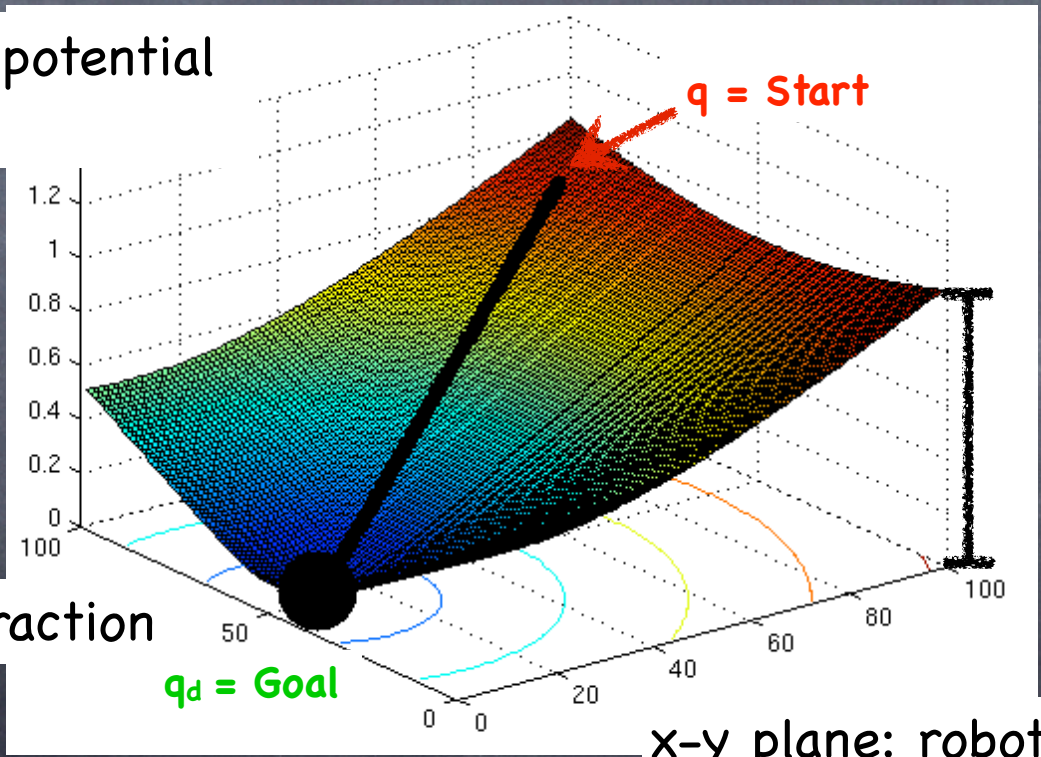
Convergent Potentials

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

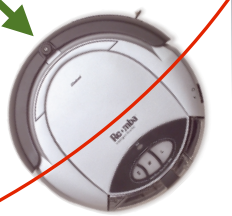
"Cone" Attractor

q_d = Goal

"Attractor"

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

q = Start



top view

w: weight

$(q - q_d)$: direction

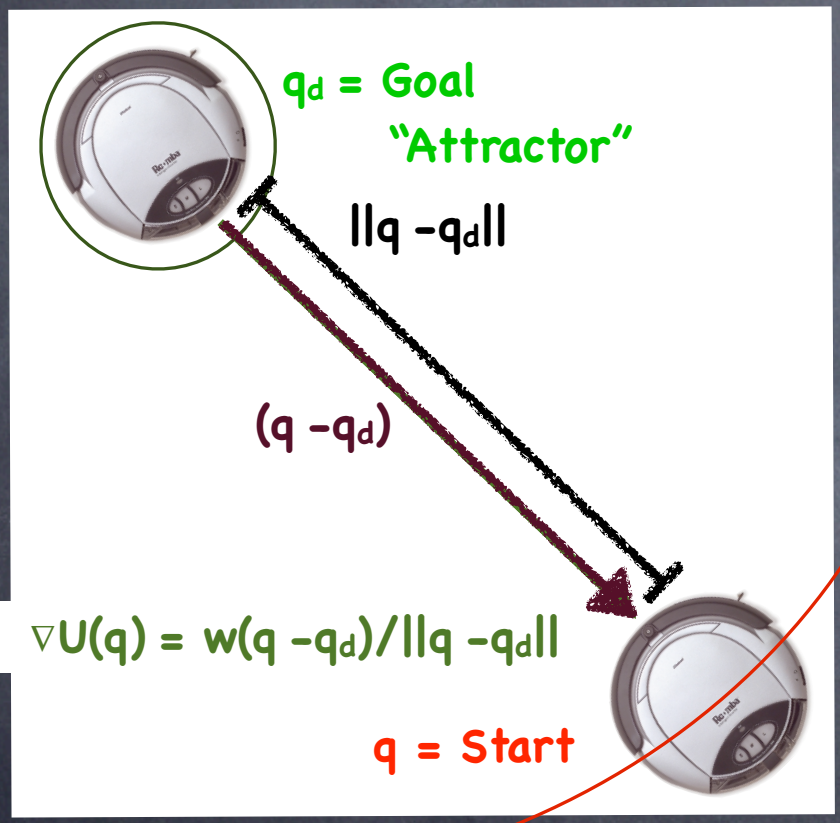
$\|q - q_d\|$: distance

Start

Goal

side view

"Cone" Attractor

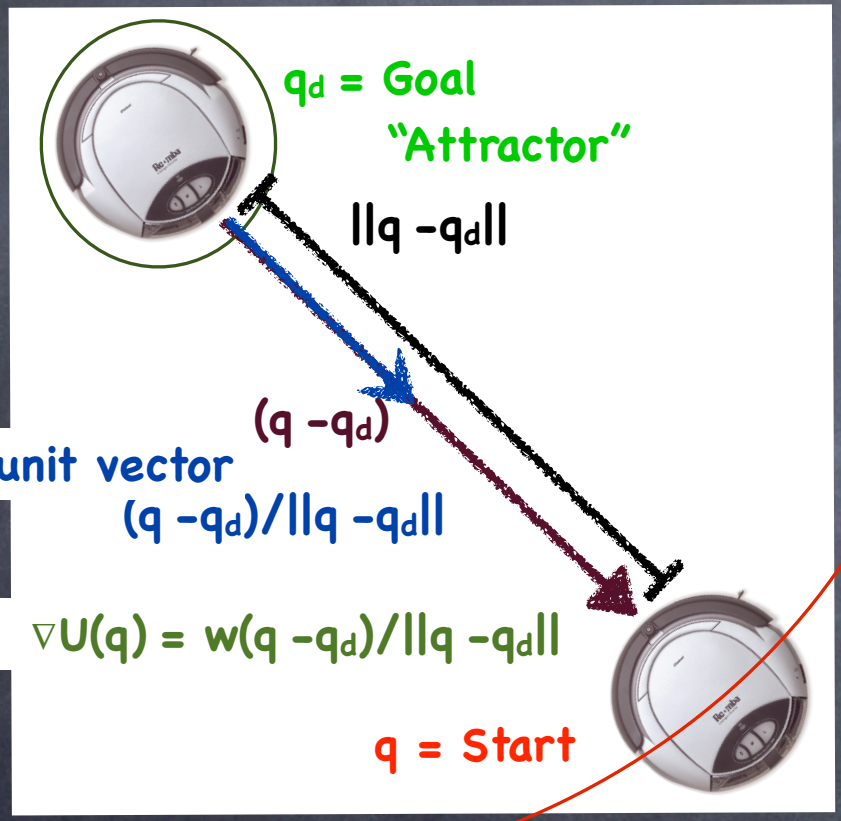


top view

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

side view

"Cone" Attractor

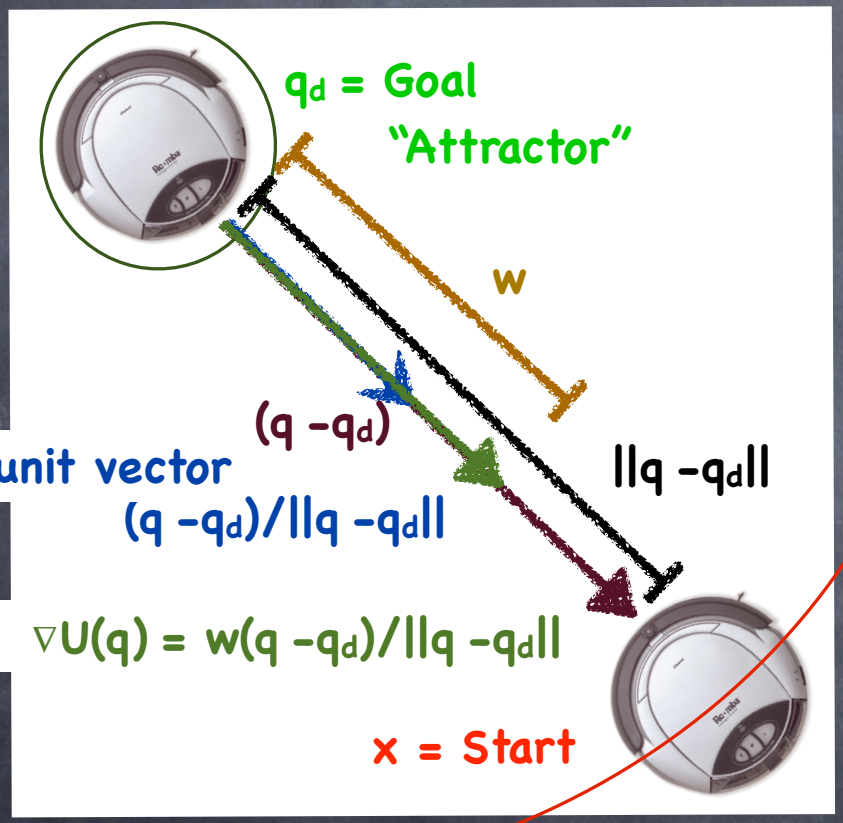


top view

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

side view

"Cone" Attractor



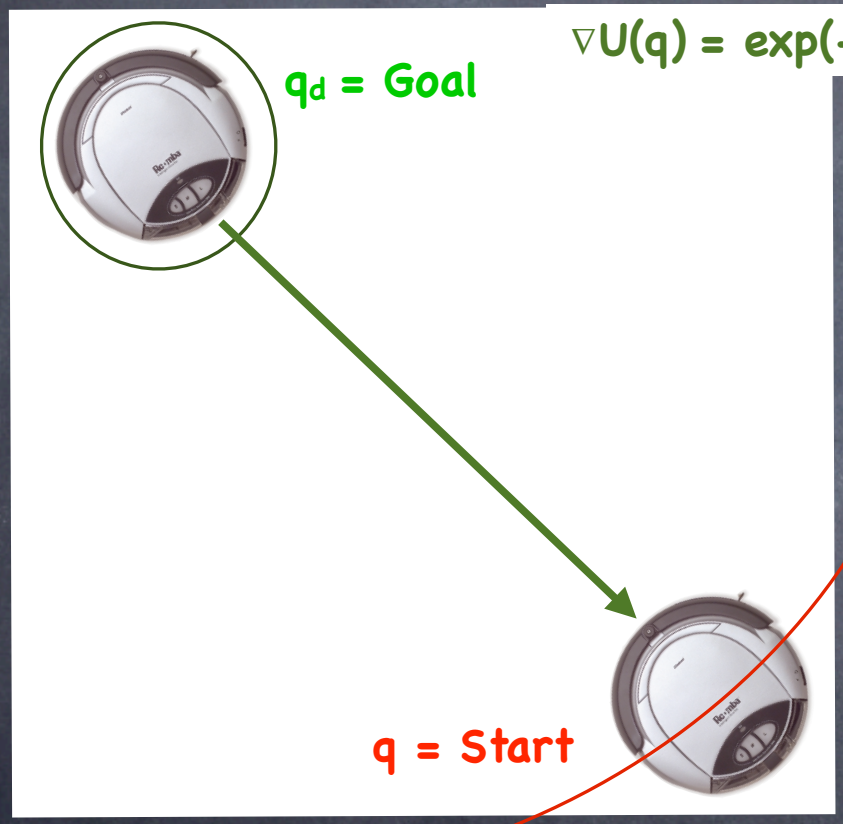
top view

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

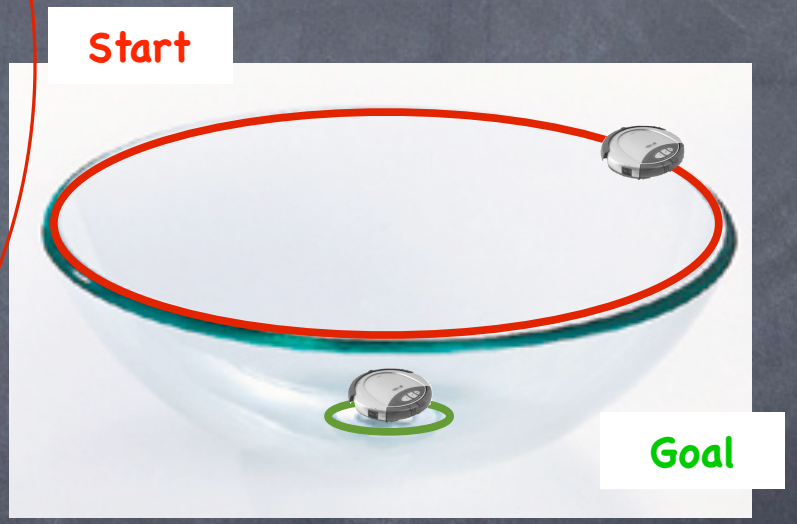
side view

"Bowl" Attractor

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



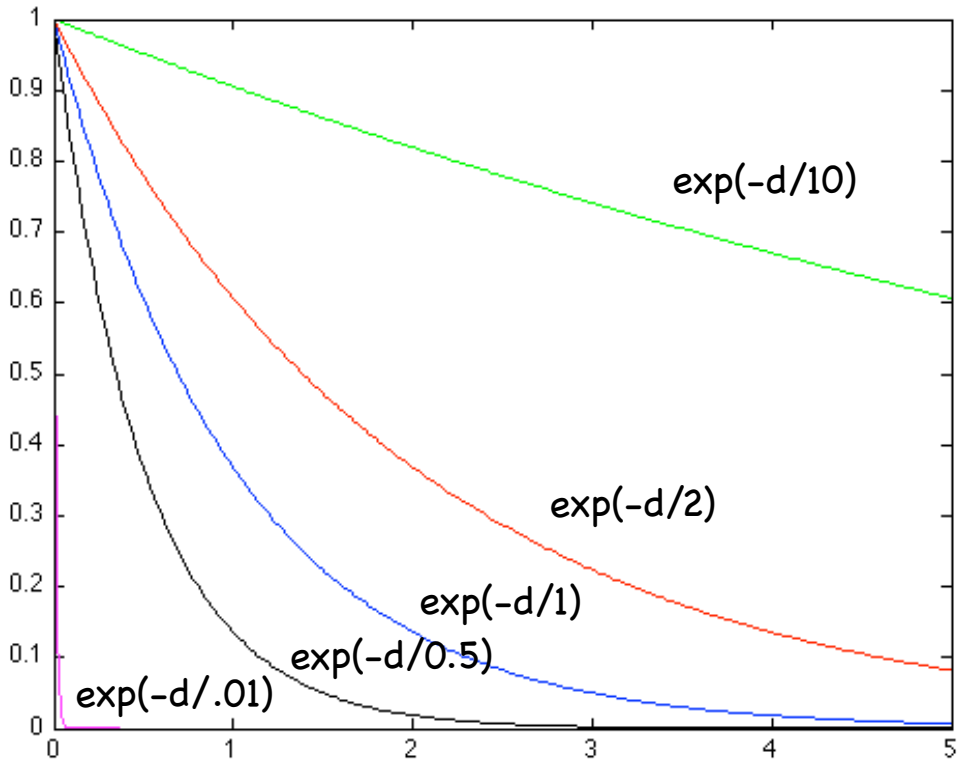
top view



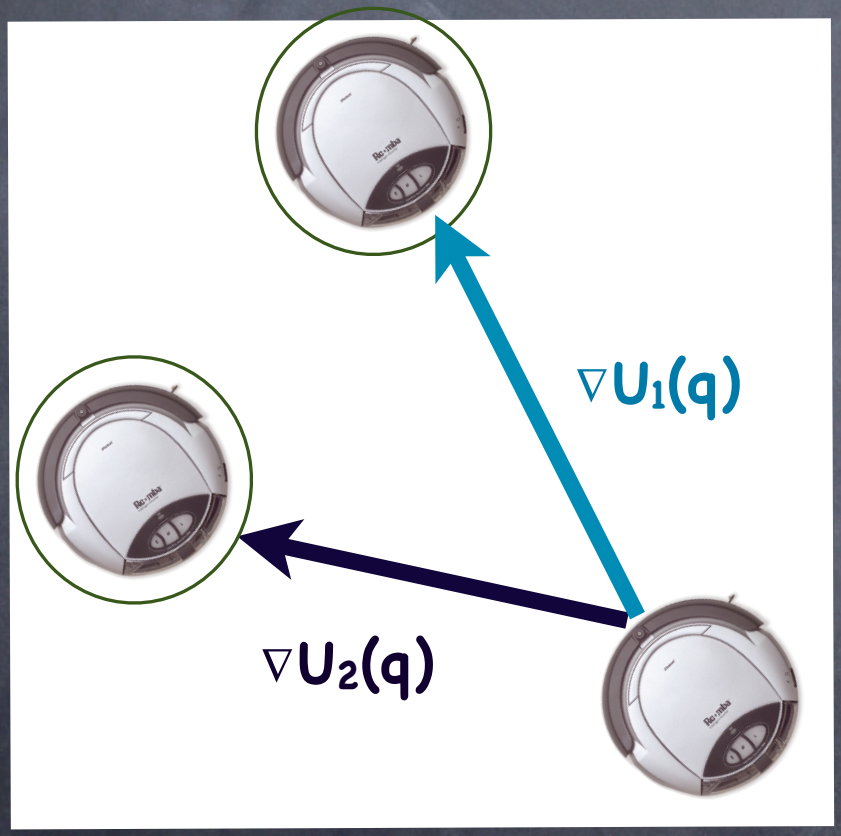
side view

exp(-d/w)

exp(-d/w) : influence of potential

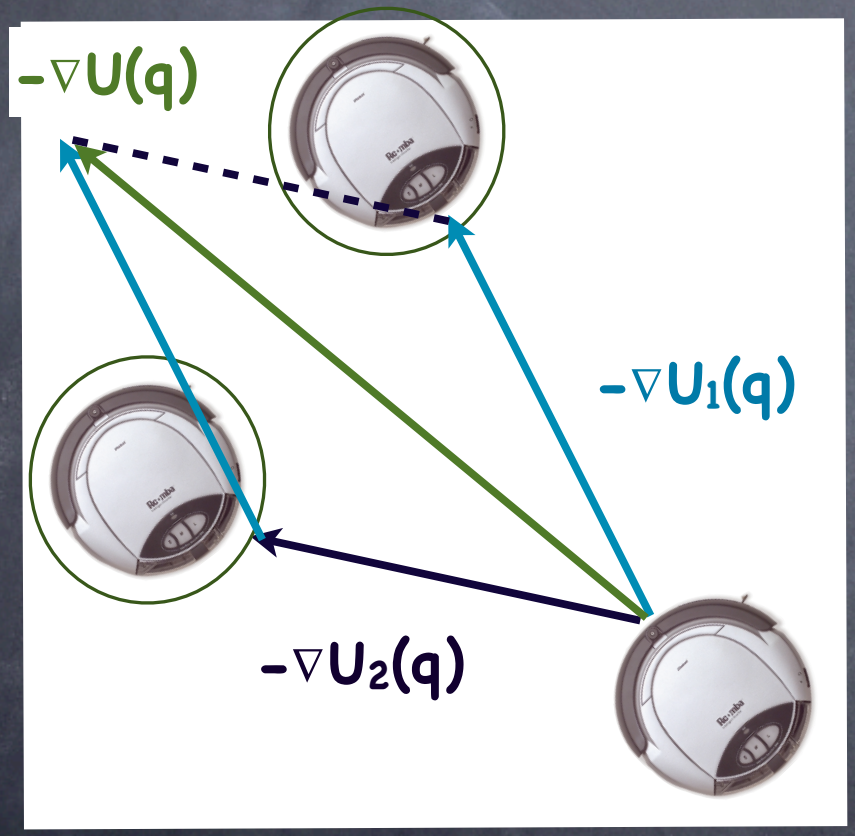


Multiple potentials



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

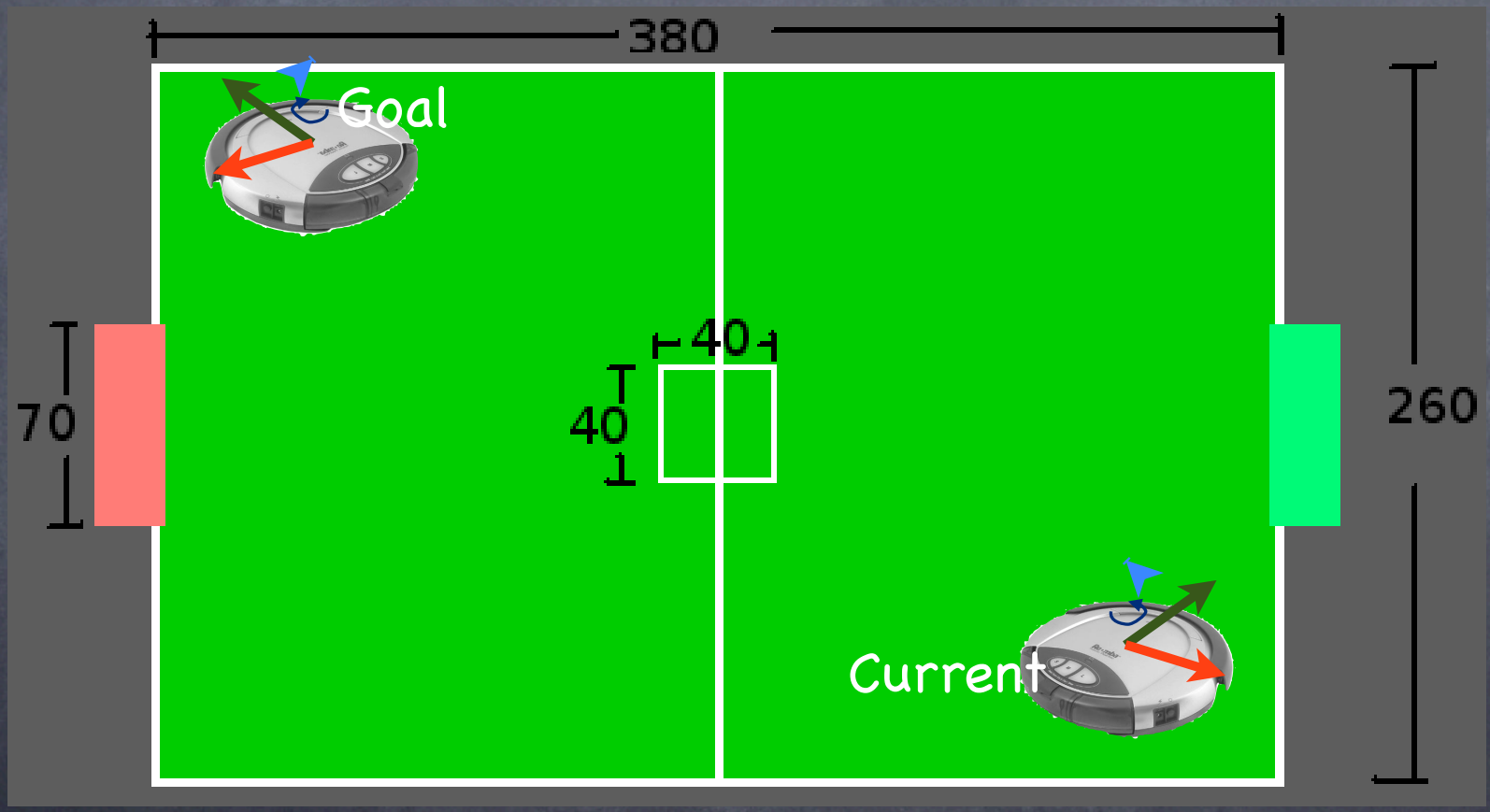
Multiple potentials



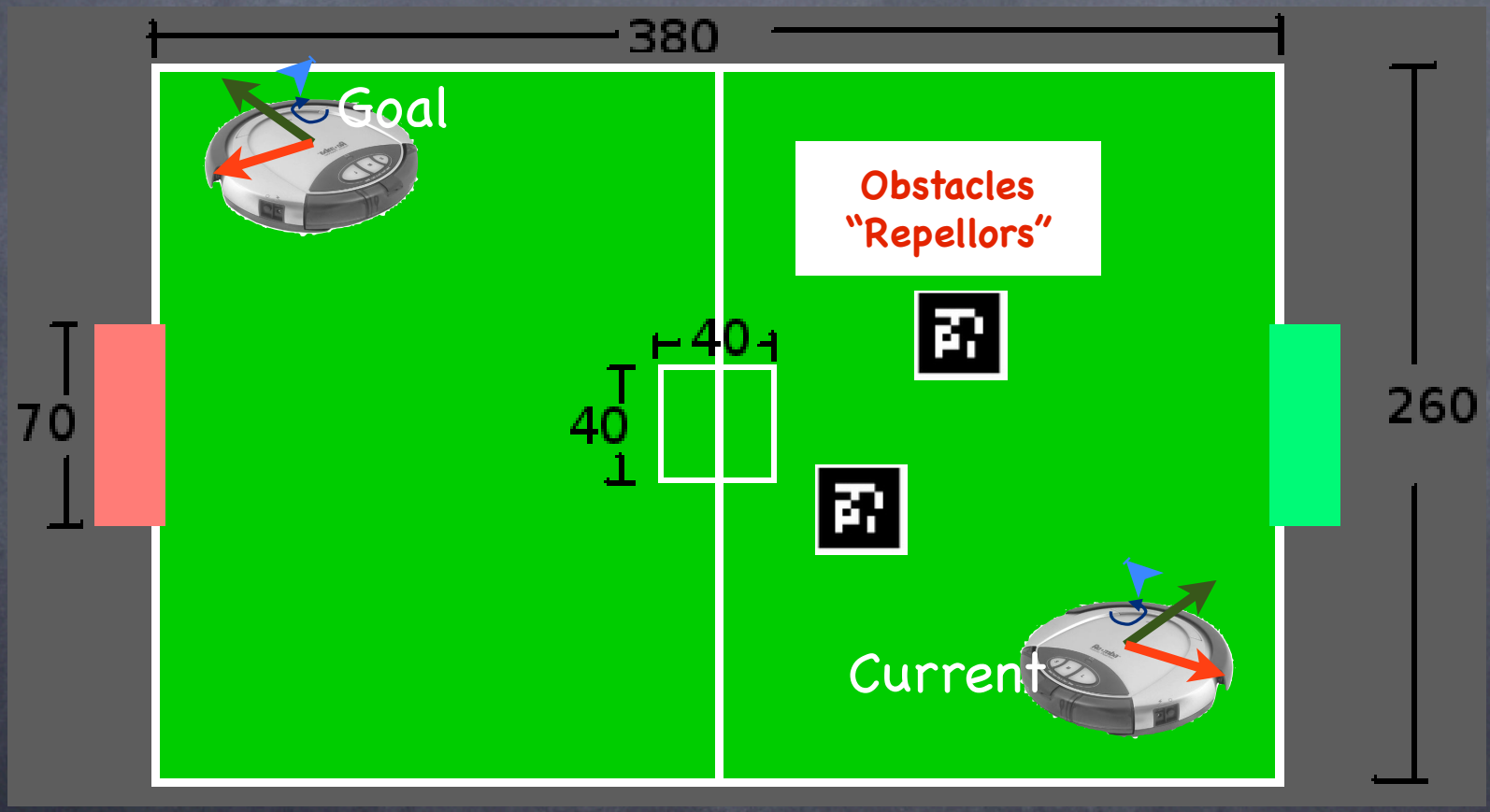
- Output of potential field is a vector
- Combine multiple potentials through vector summation

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

describe performance for this case

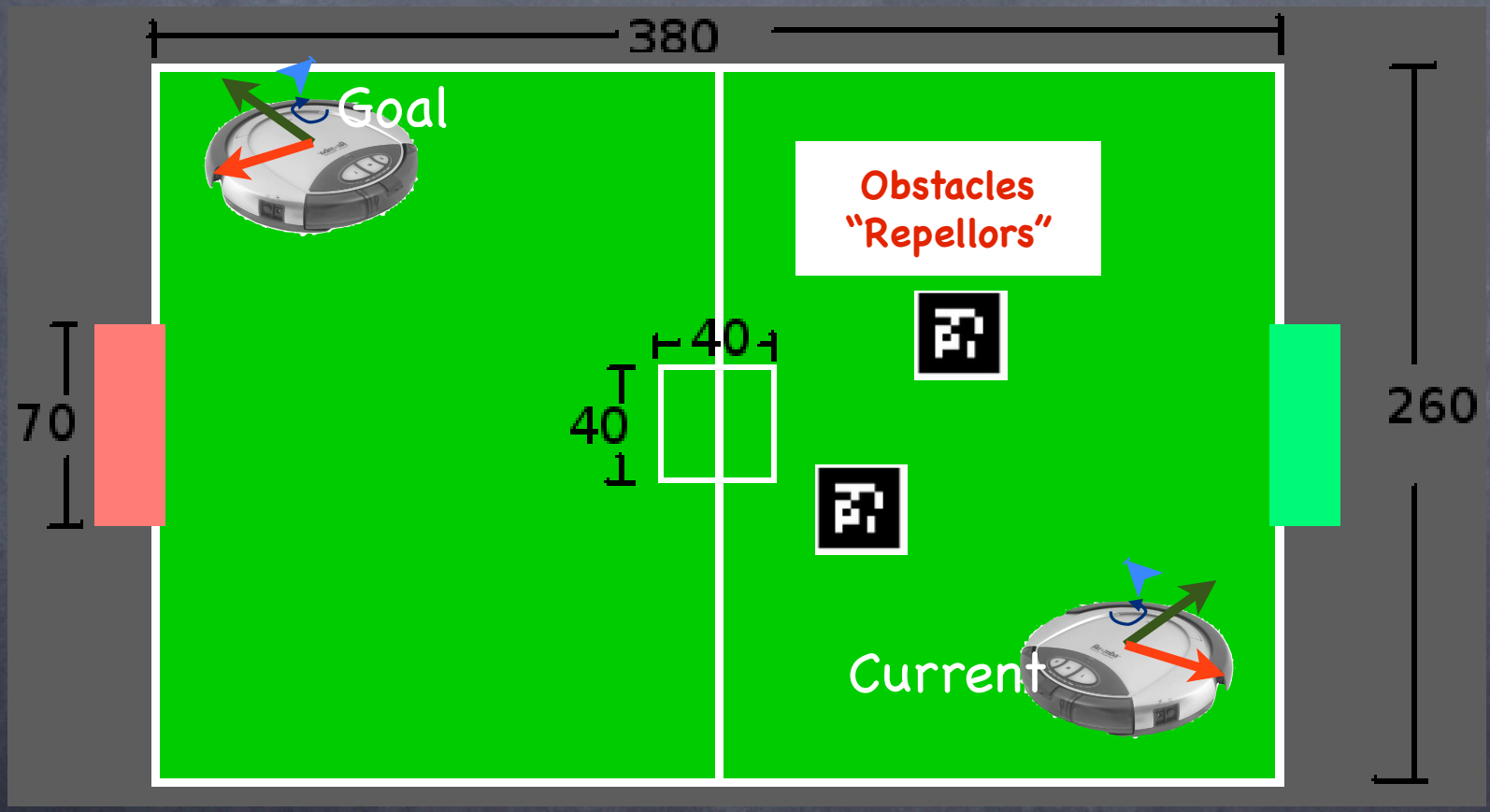


describe performance for this case



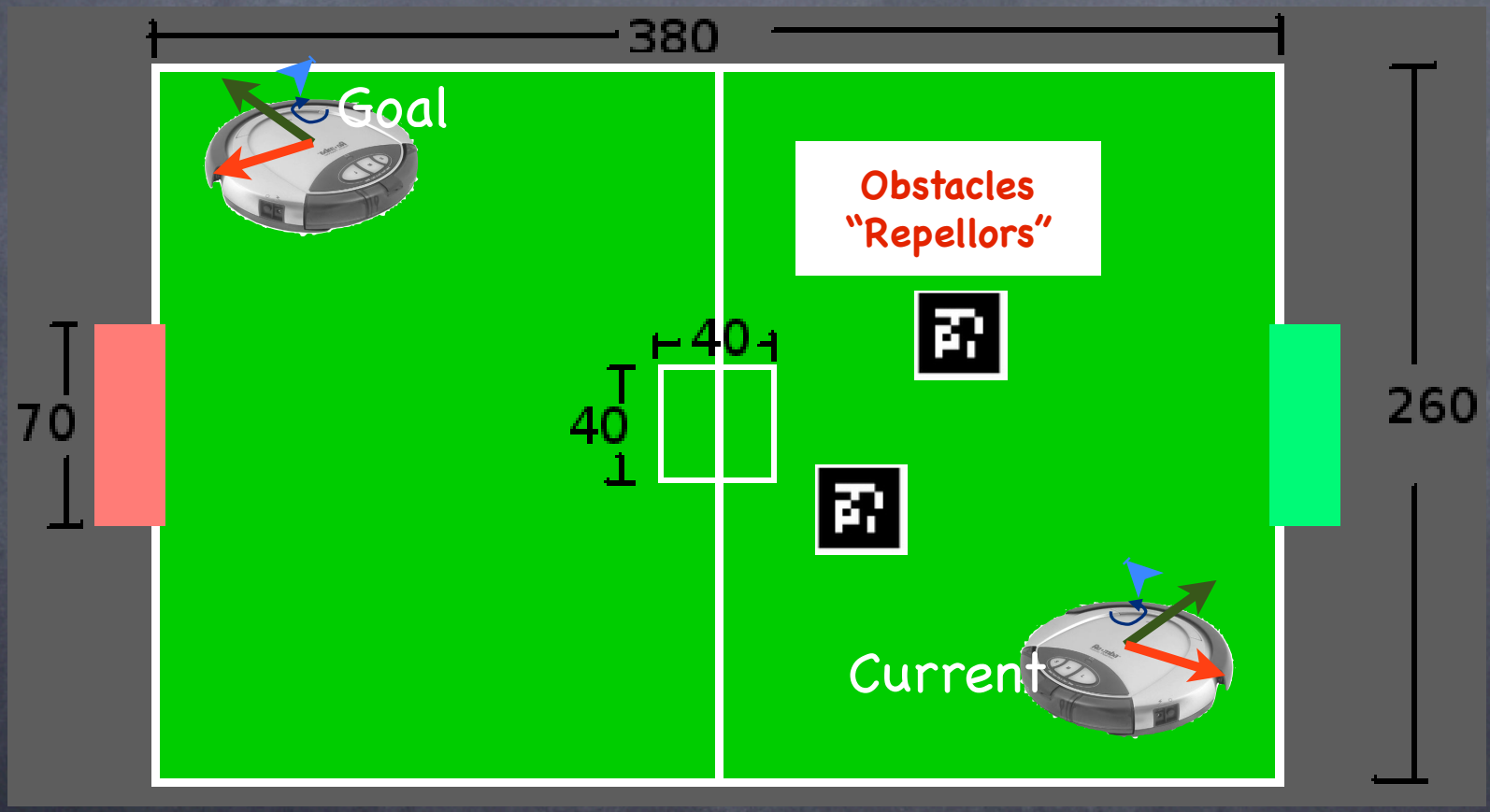
describe performance for this case

how do we deal with repellors?



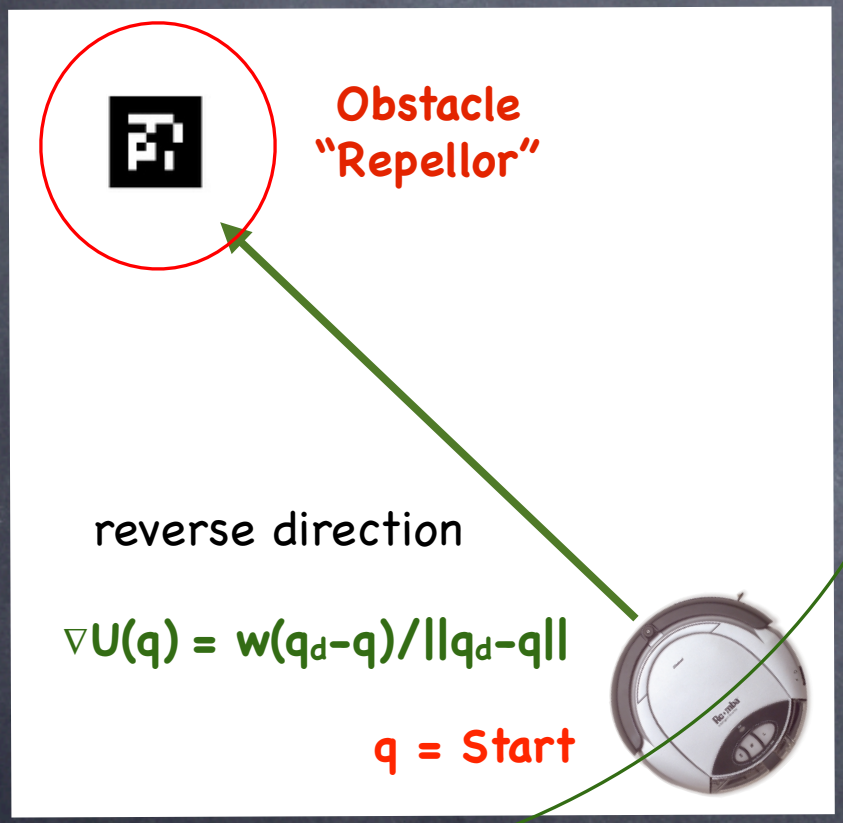
add sum of repulsive potentials

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



"Cone" Repellor

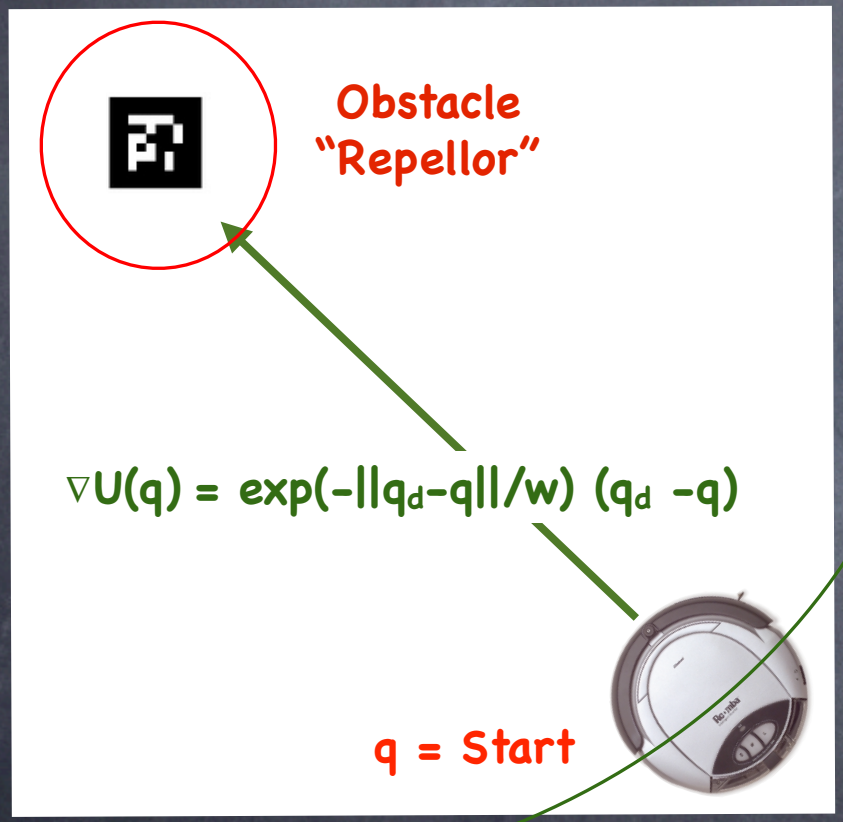
potential problems?



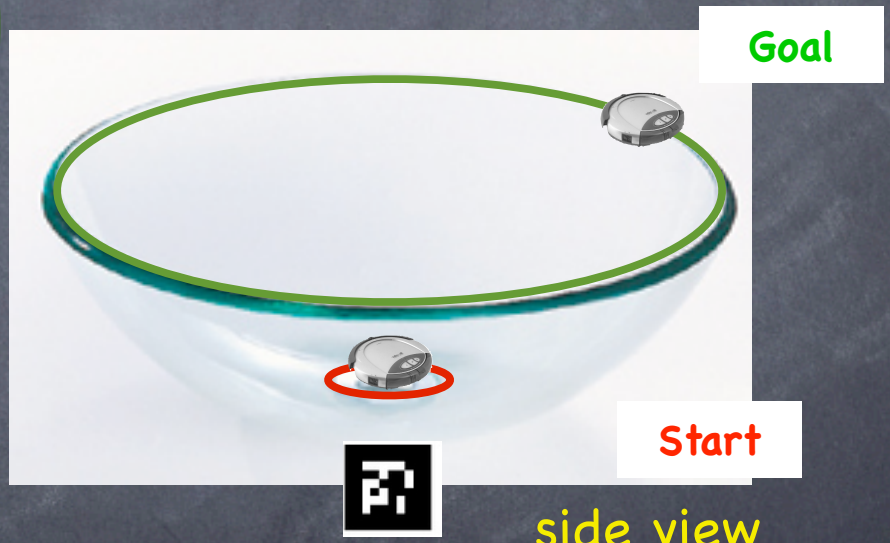
top view

side view

"Bowl" Repellor

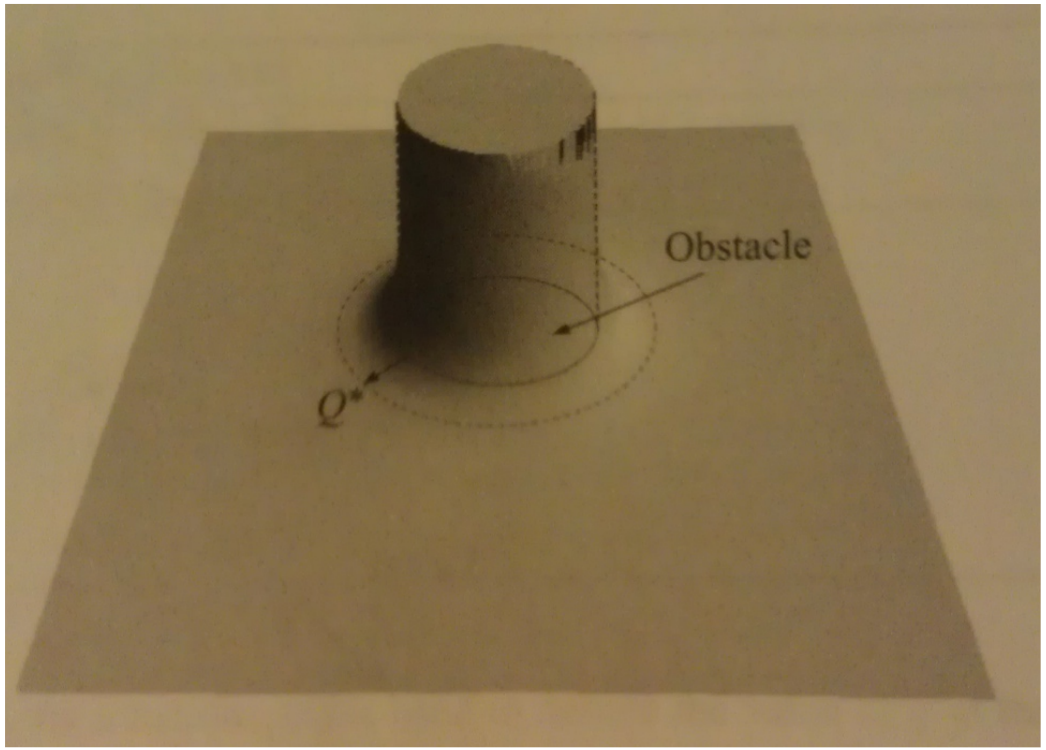


top view

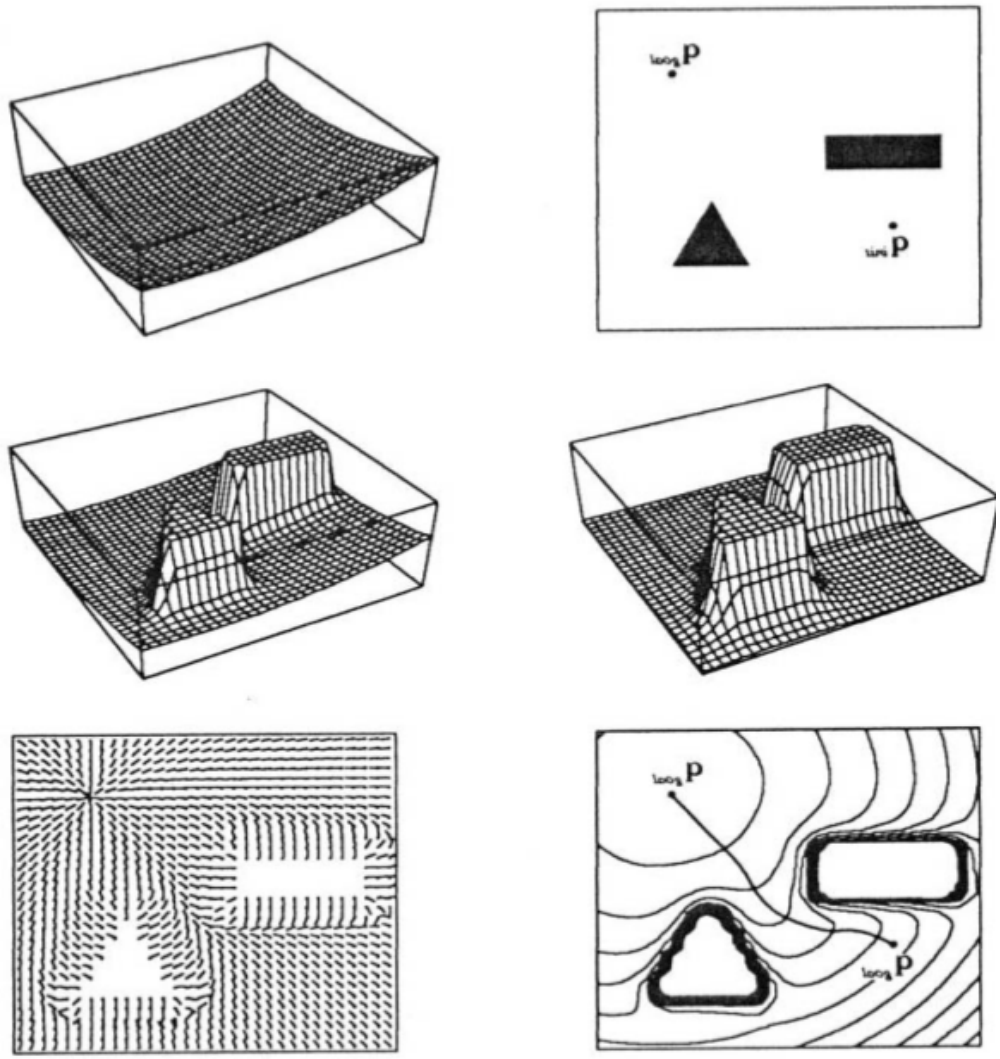


side view

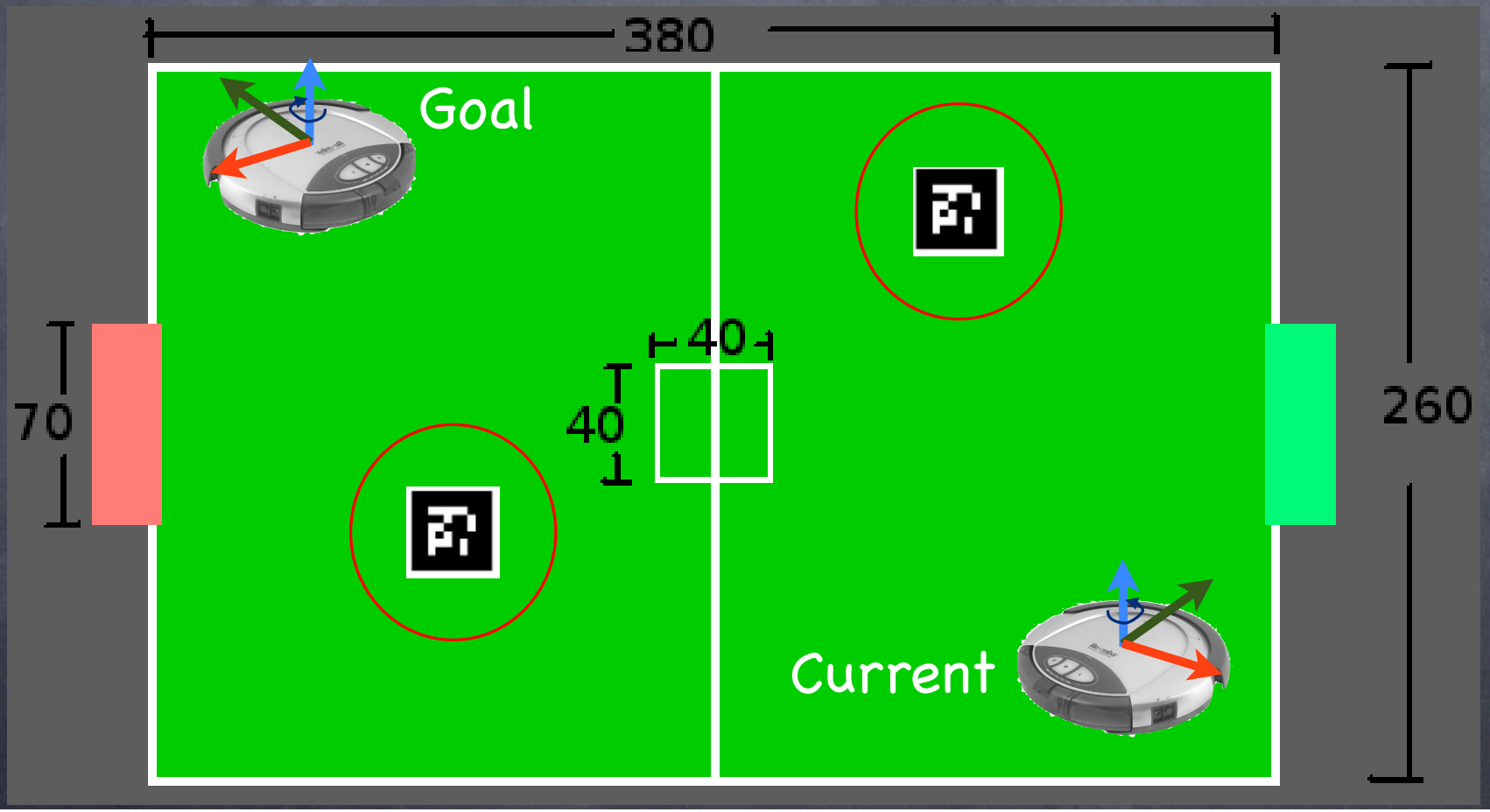
repellor should only have local influence,
repelling only around boundary improves path



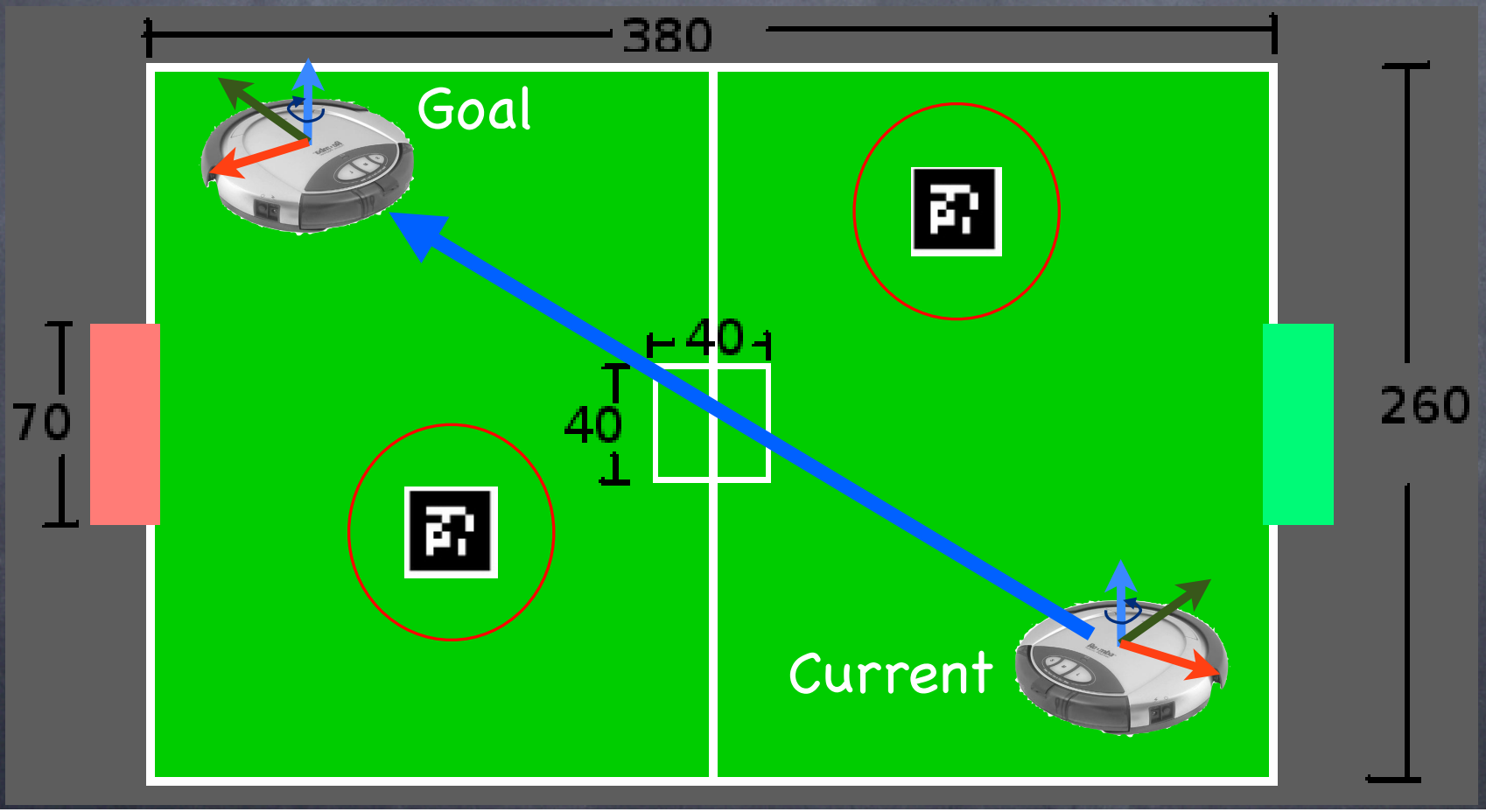
2 Obstacle example



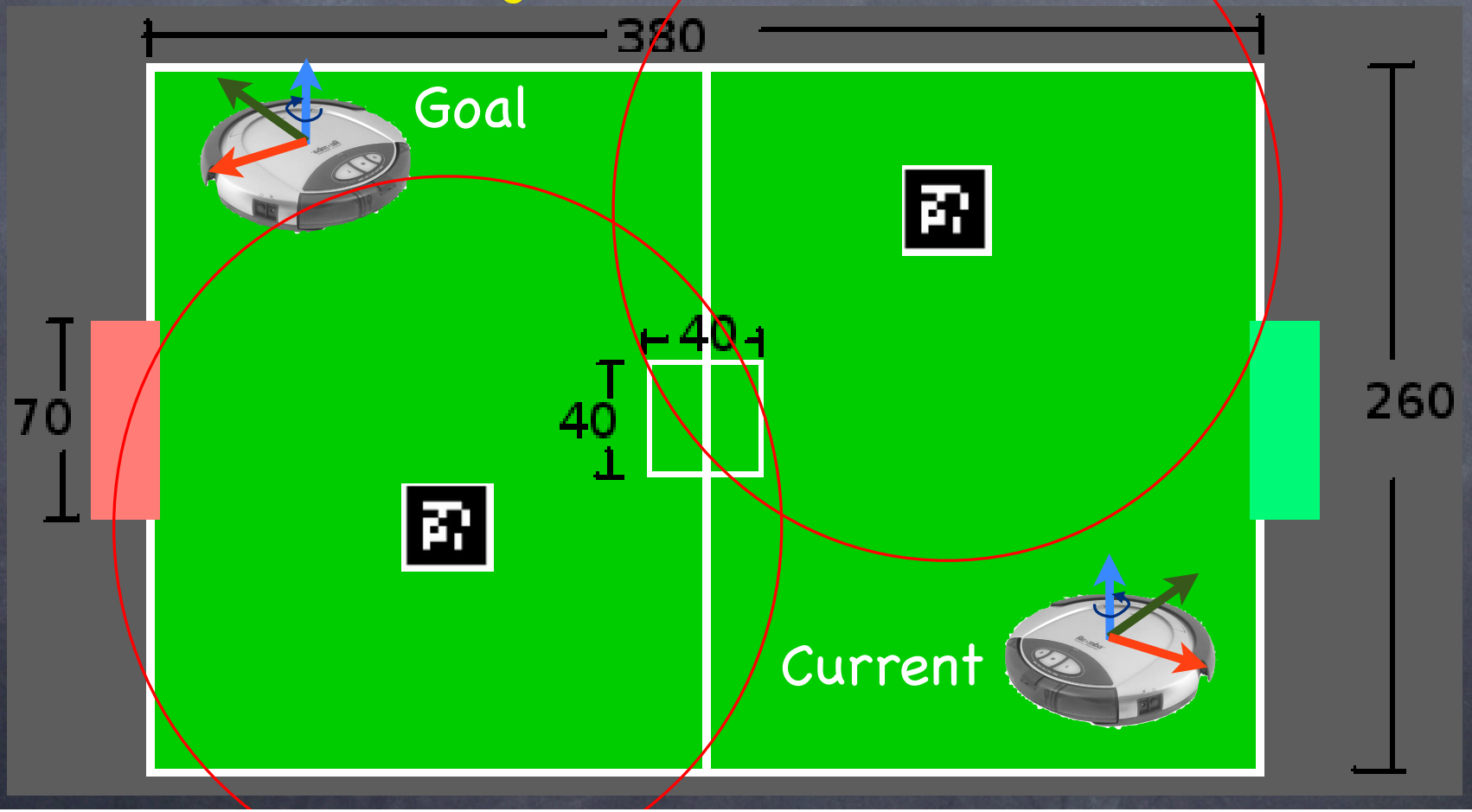
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



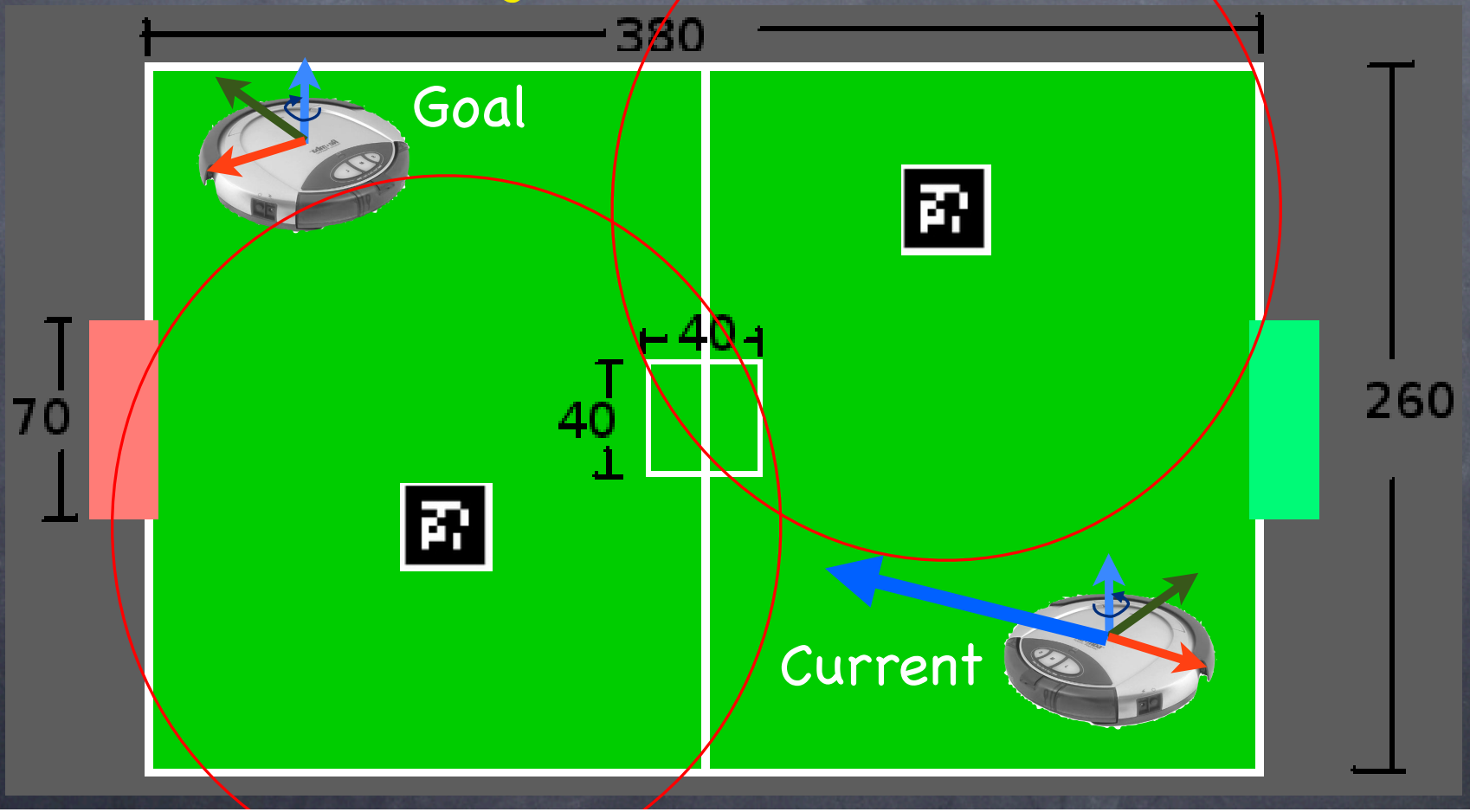
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



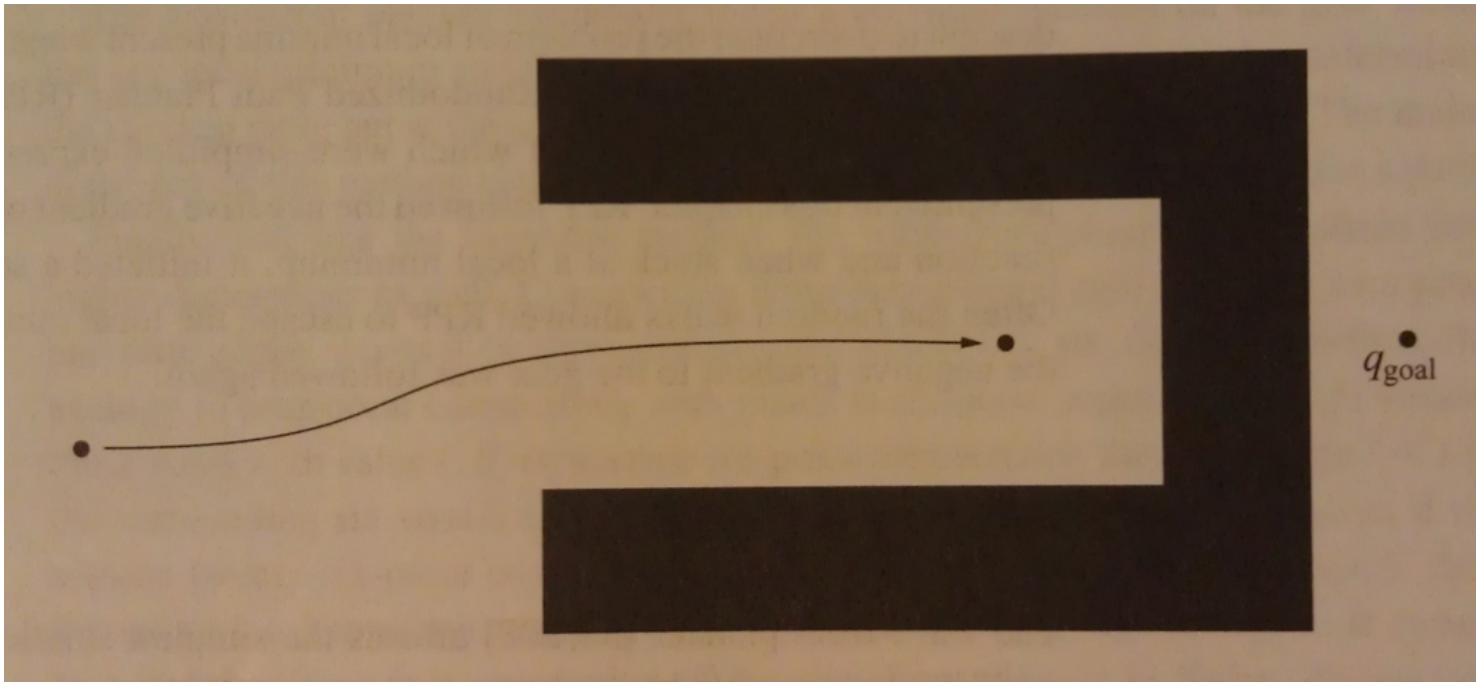
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



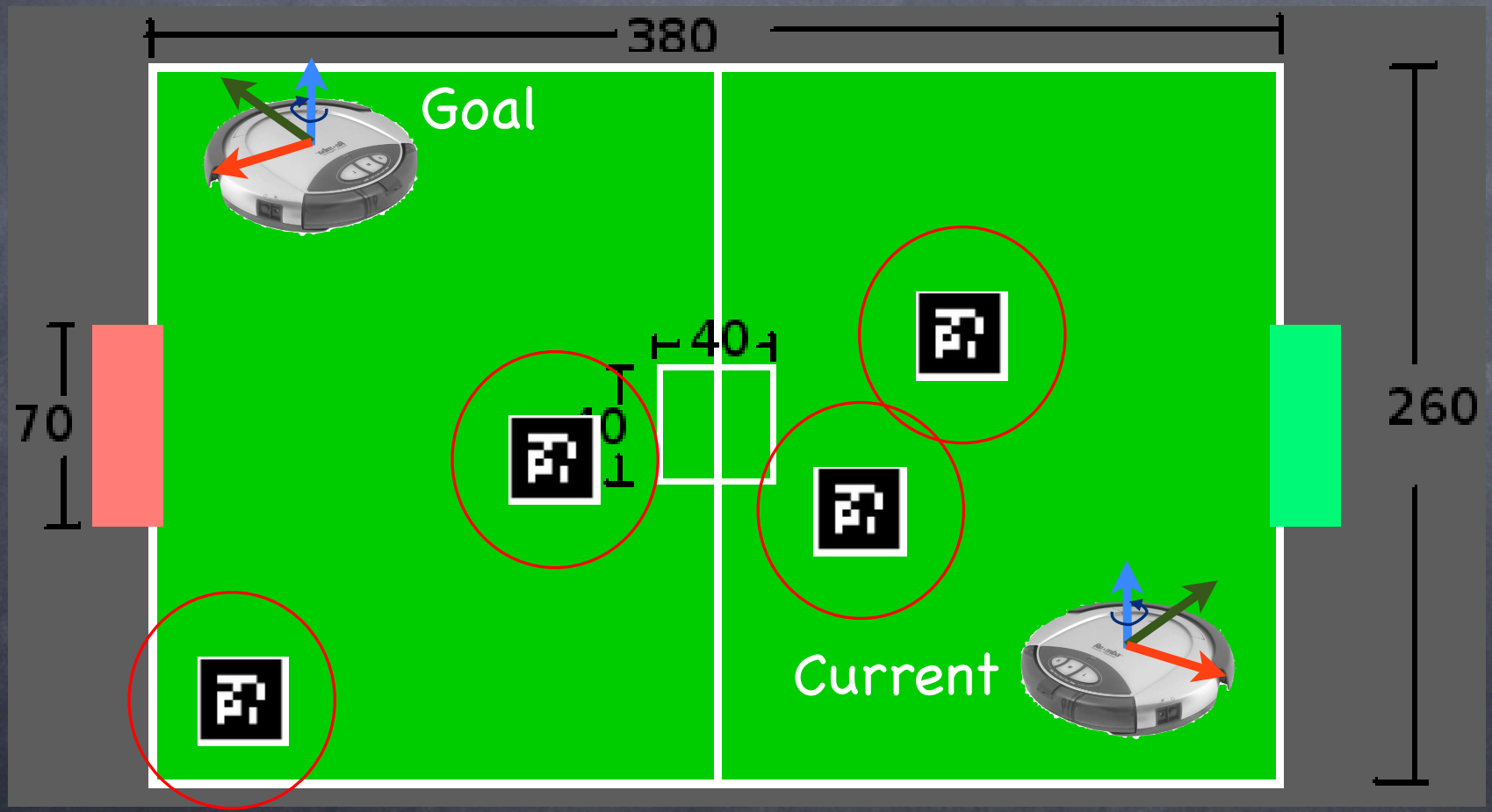
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



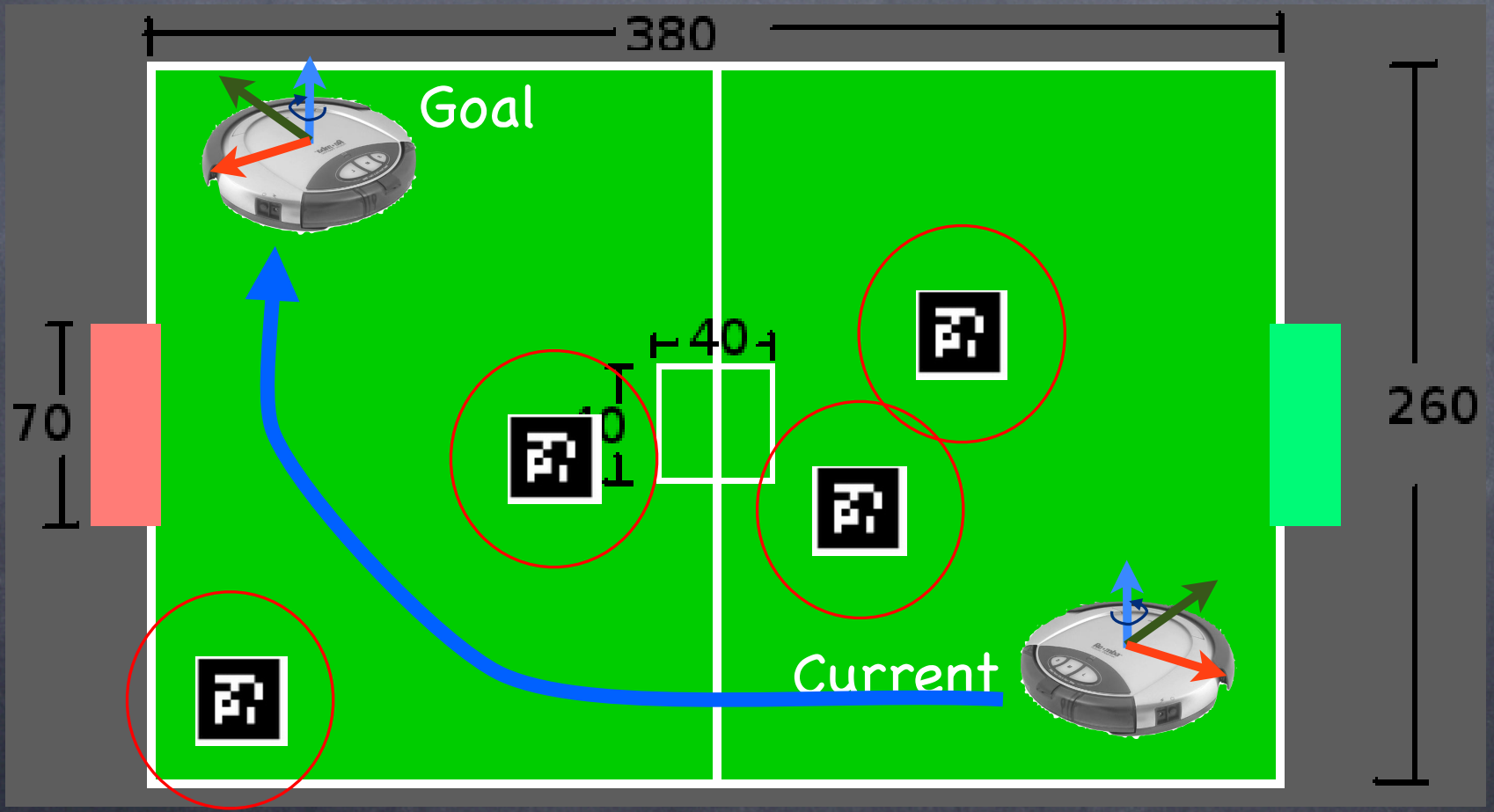
Local Minima



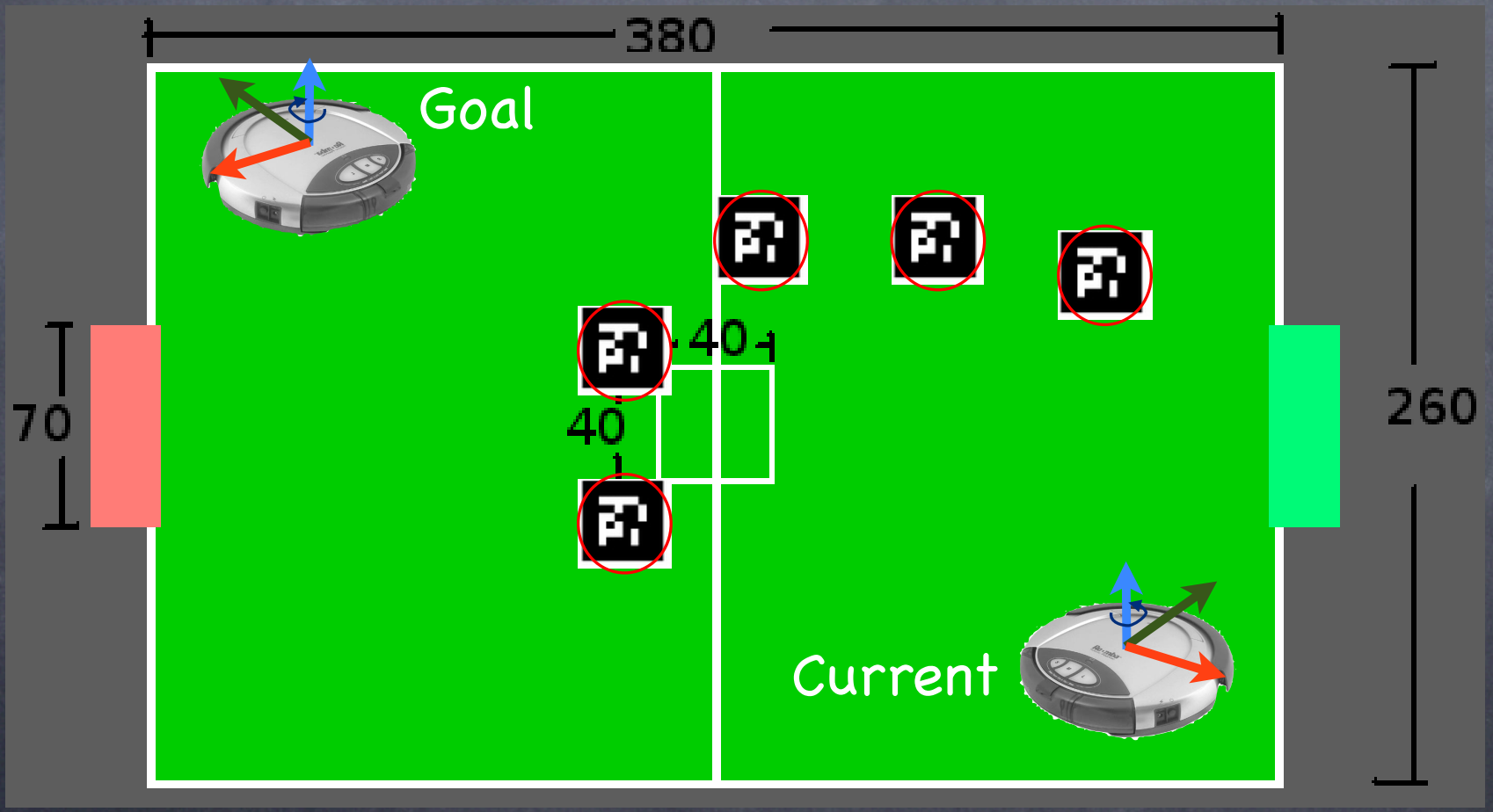
describe performance for this case



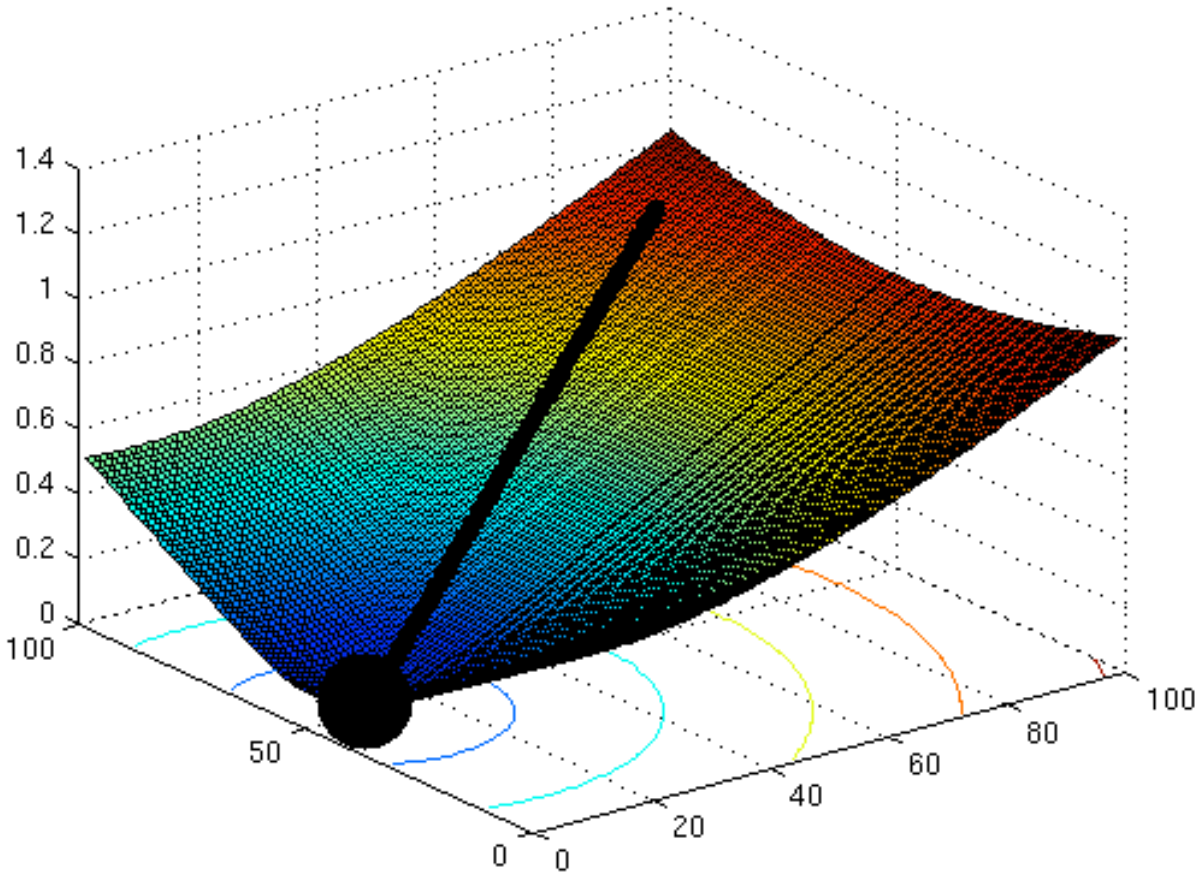
describe performance for this case



describe performance for this case

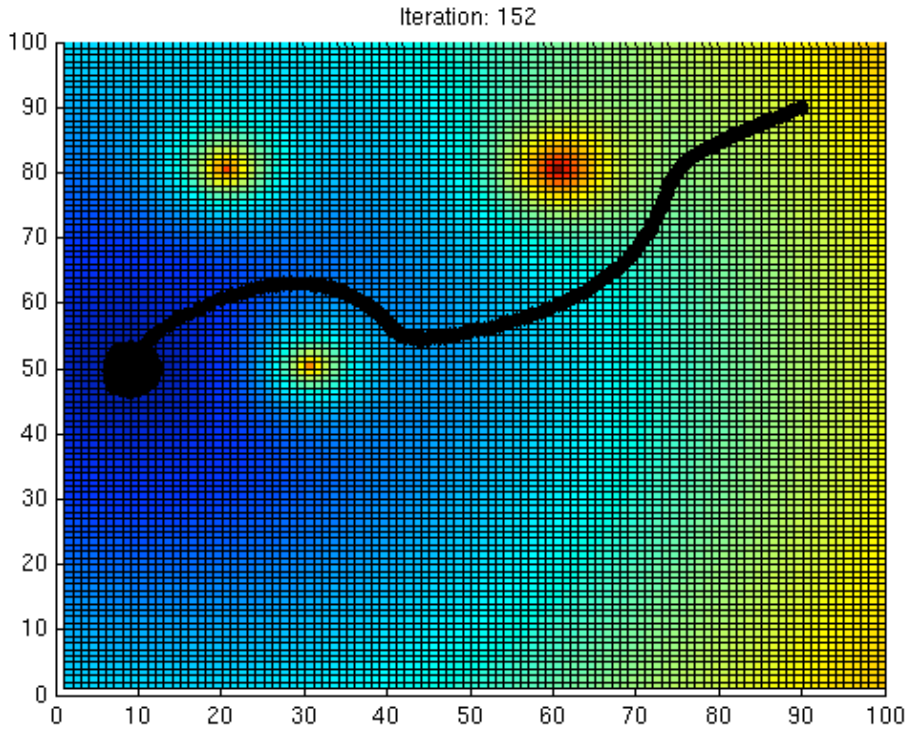


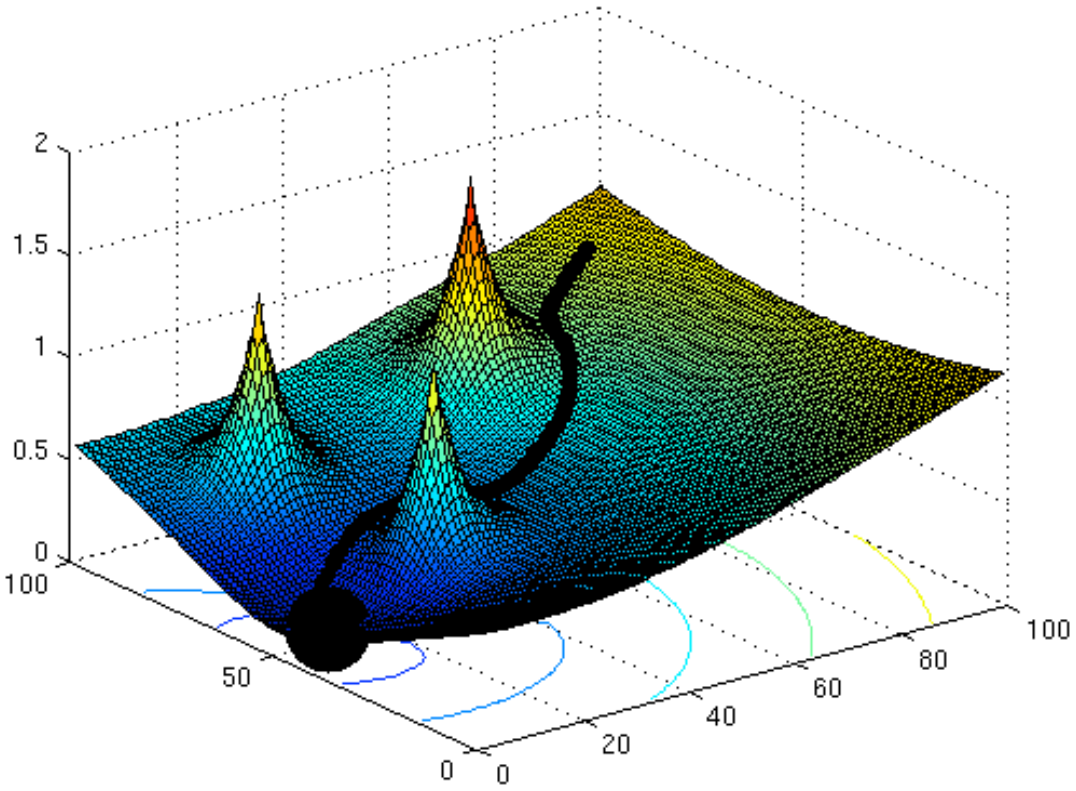
matlab example



pfield.m [1 5 8 12]

matlab example

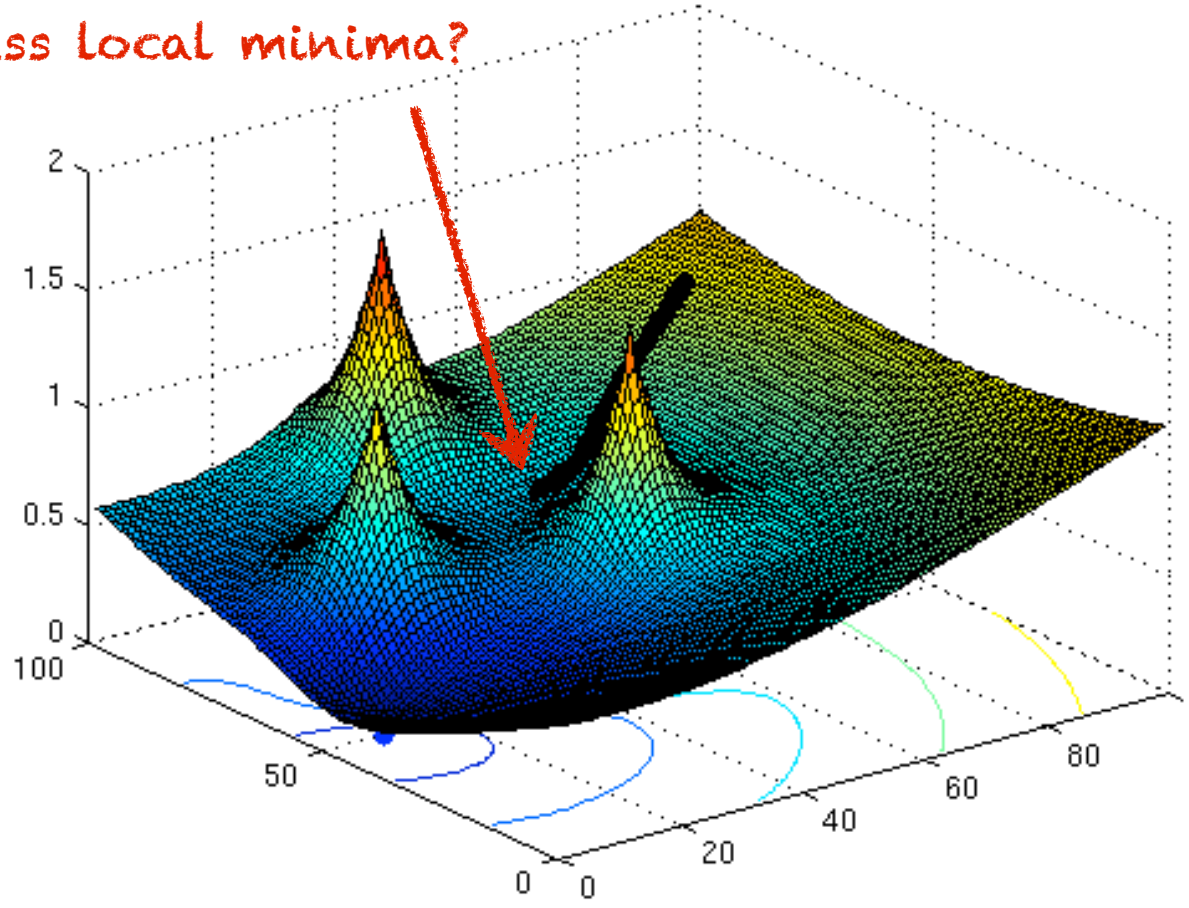




pfield.m [1 5 8 12]

matlab example

How to address local minima?



pfield.m [1 5 8 12]

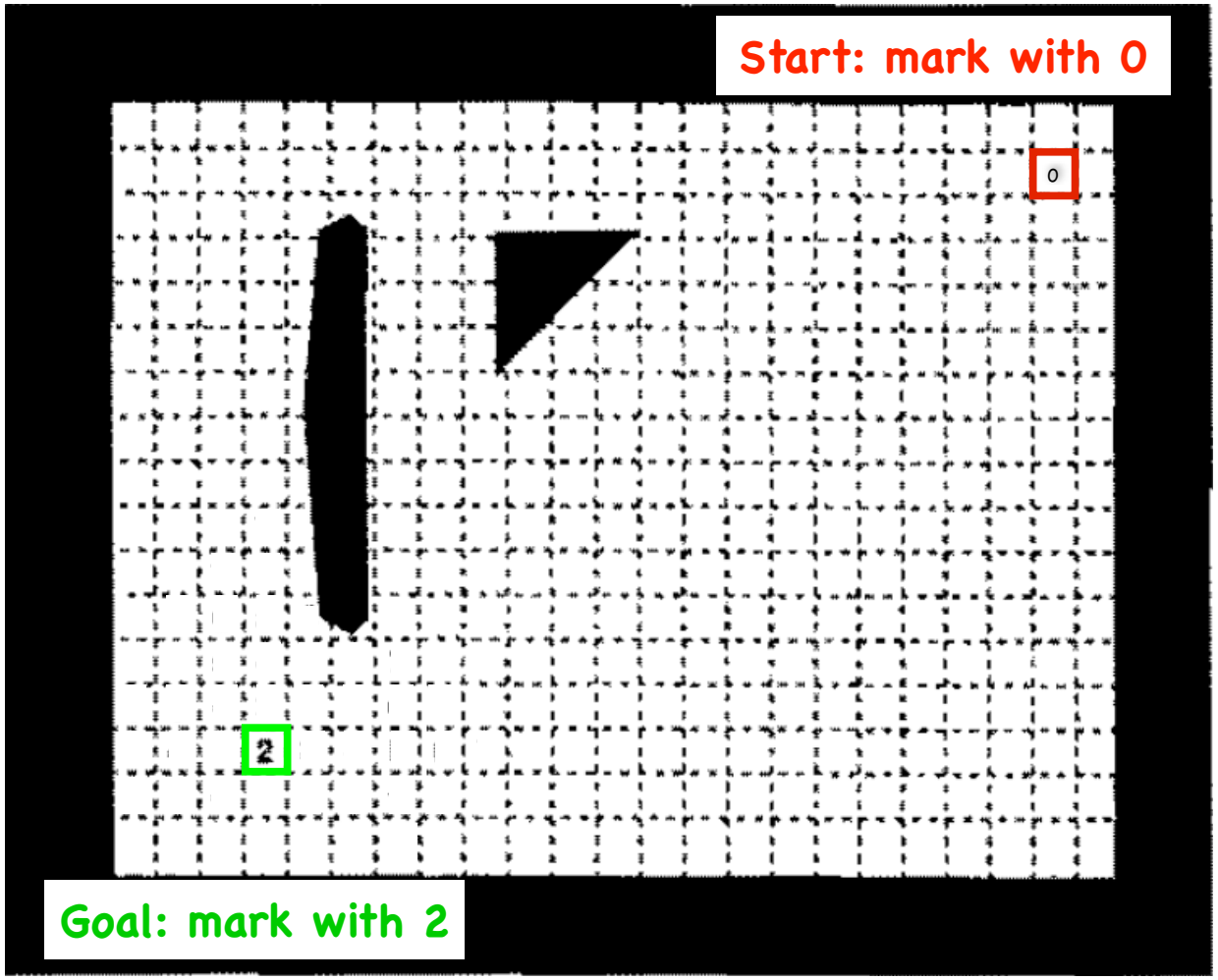
How can we get out of local minima?

How can we get out of local minima?

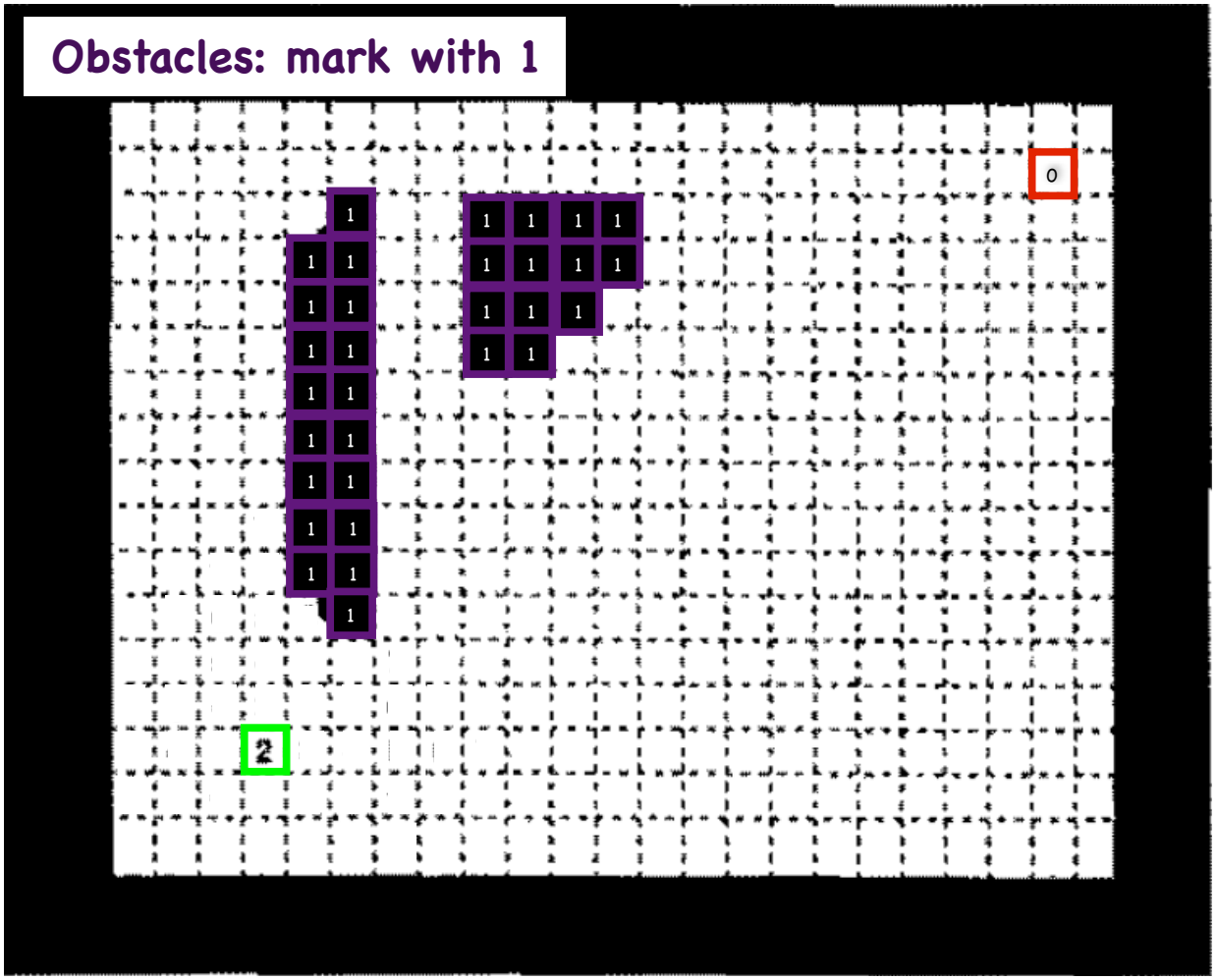
Go back to planning.

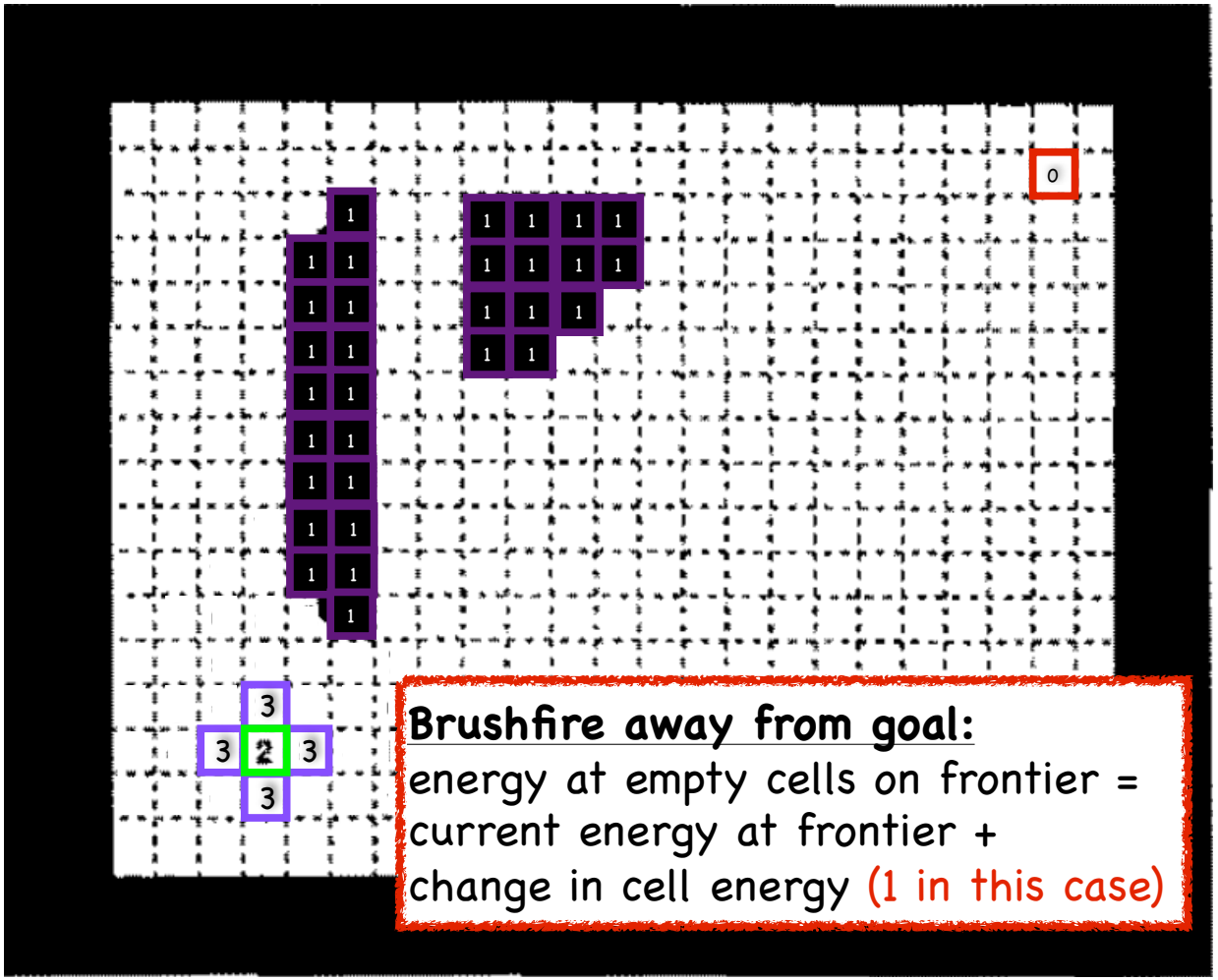
Wavefront Planning

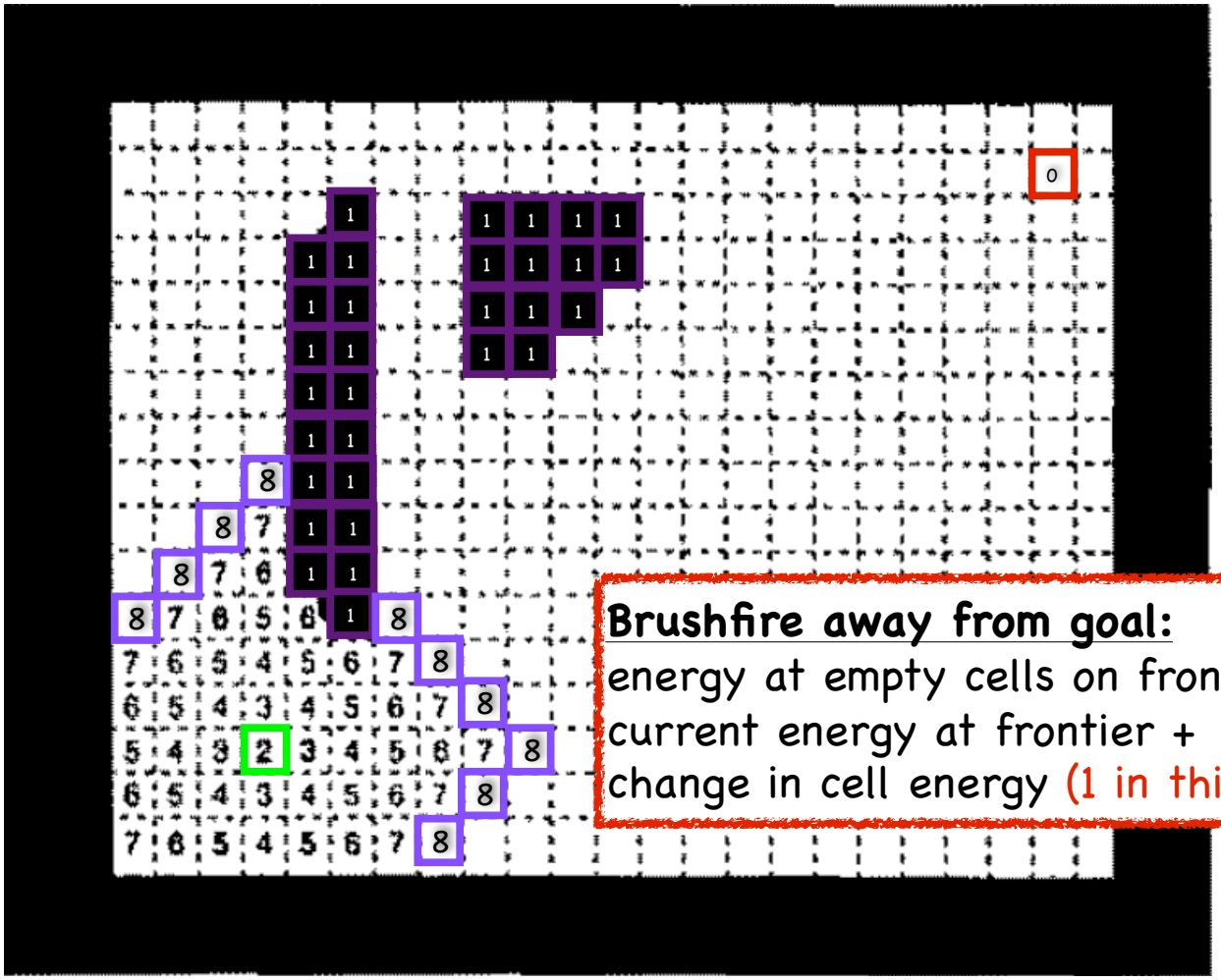
- Discretize potential field into grid
  - Cells store cost to goal with respect to potential field
  - Computed by Brushfire algorithm (essentially BFS)
- Grid search to find navigation path to goal

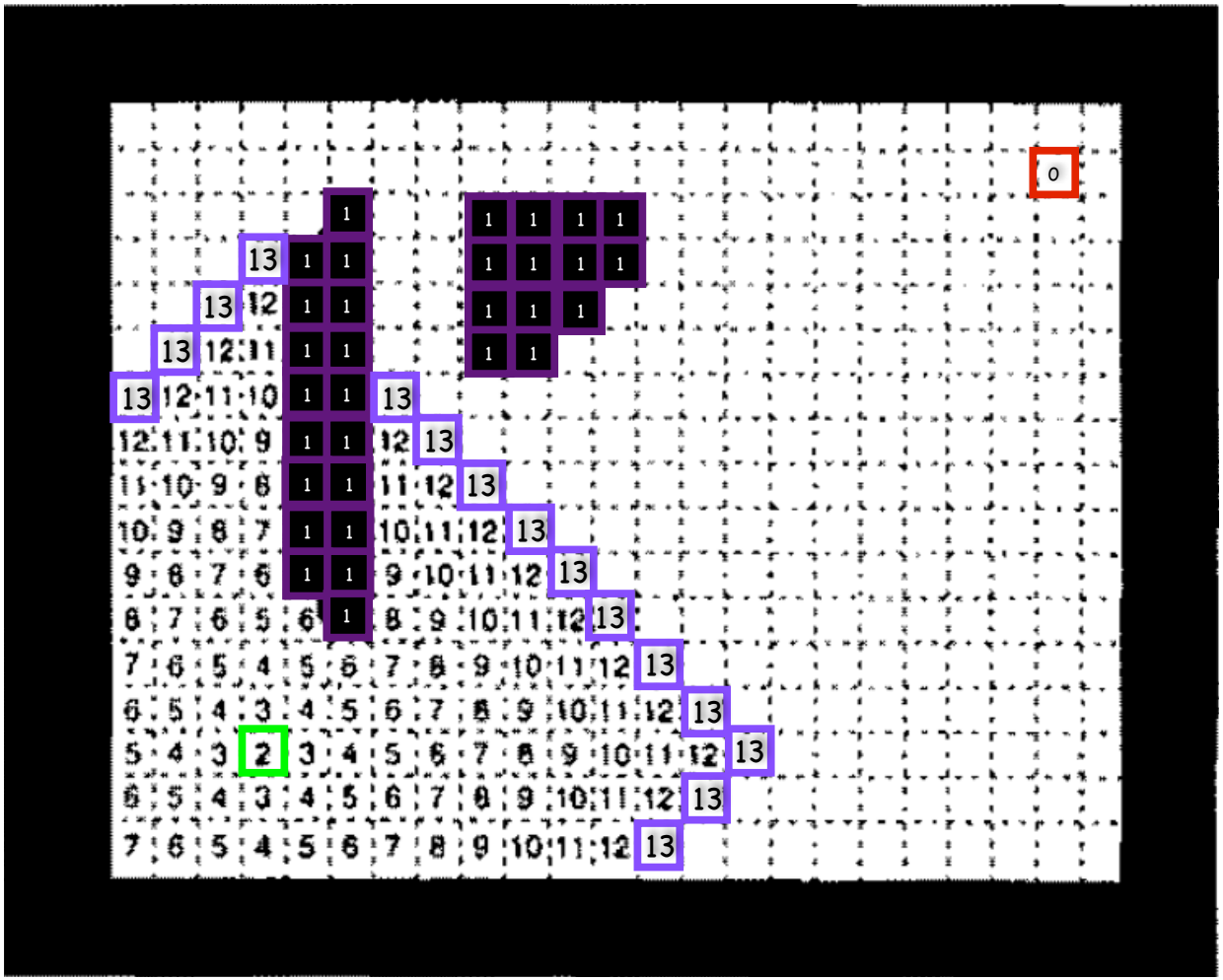

Start: mark with 0
0
2
Goal: mark with 2

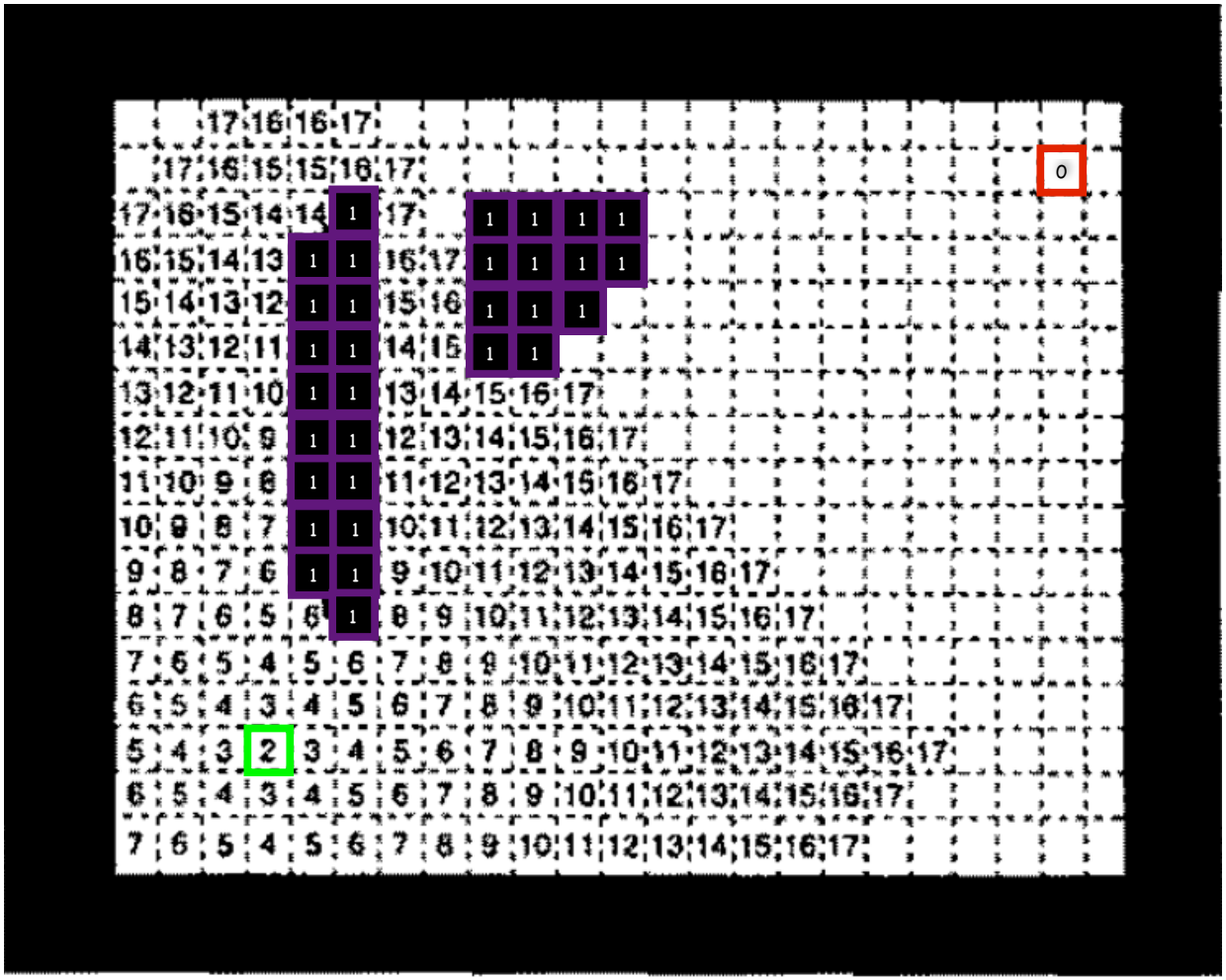
Obstacles: mark with 1

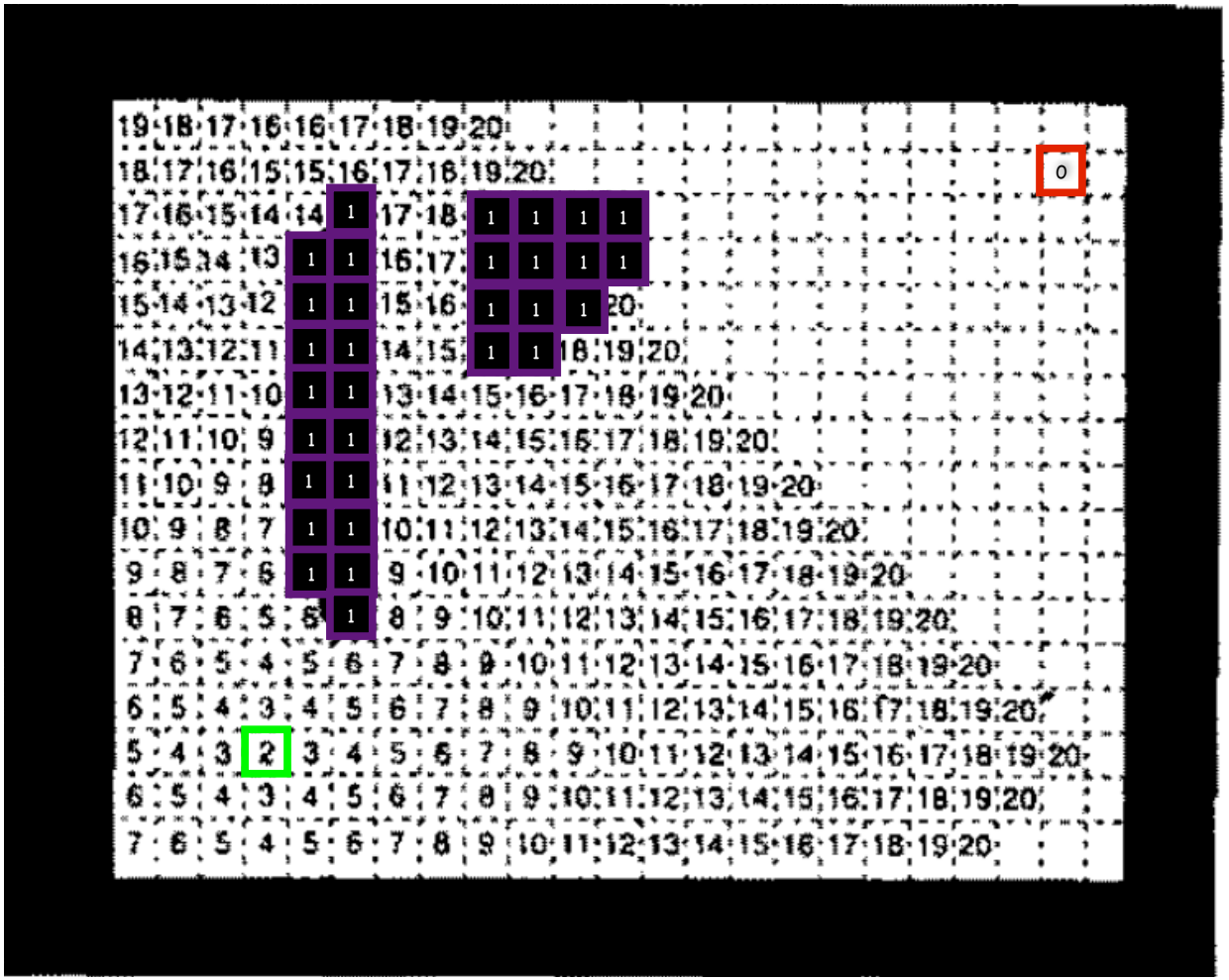


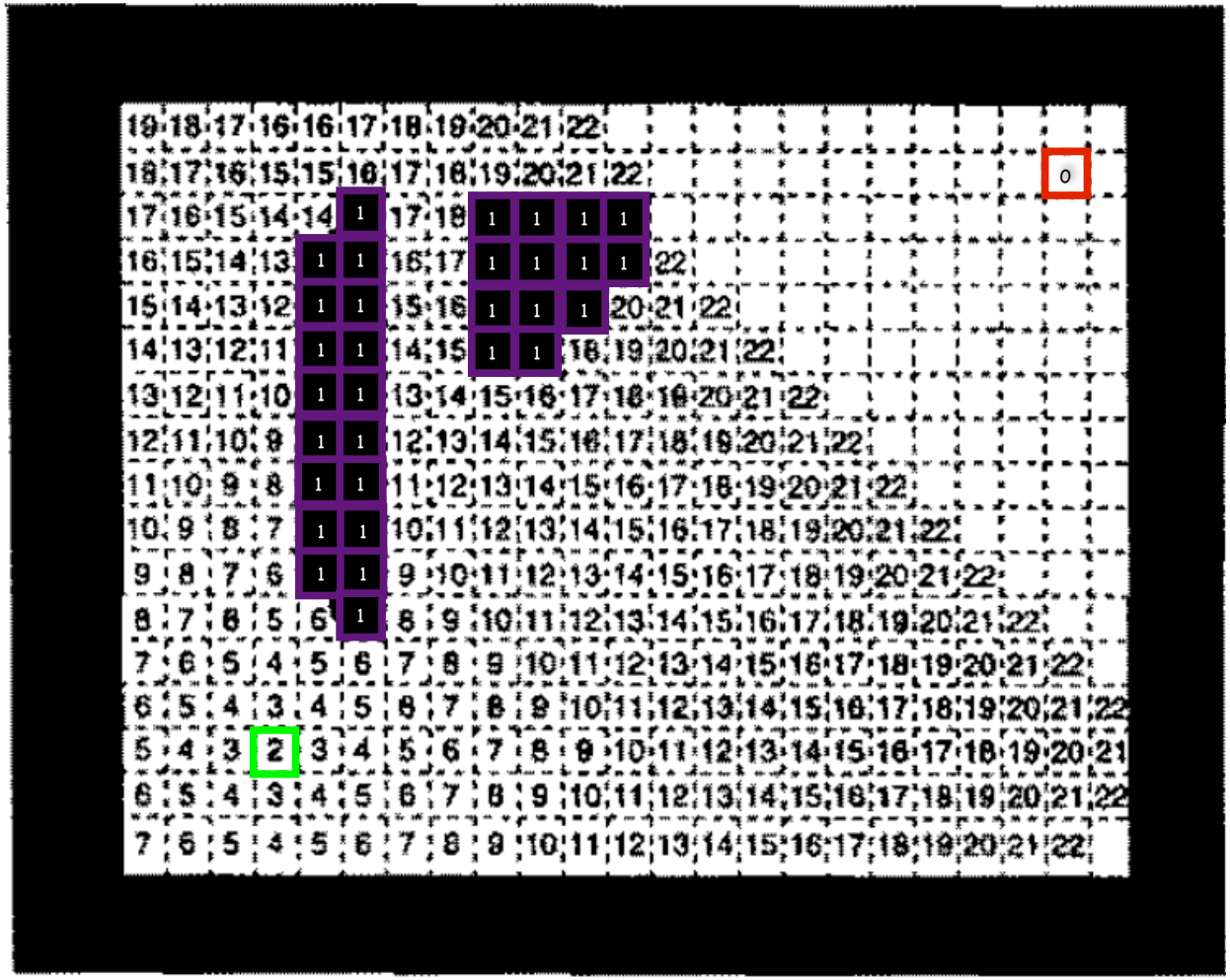

0
1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1
1 1 1 1
1 1 1 1
1 1 1
1 1
3
3 2 3
3
Brushfire away from goal:
energy at empty cells on frontier =
current energy at frontier +
change in cell energy (1 in this case)

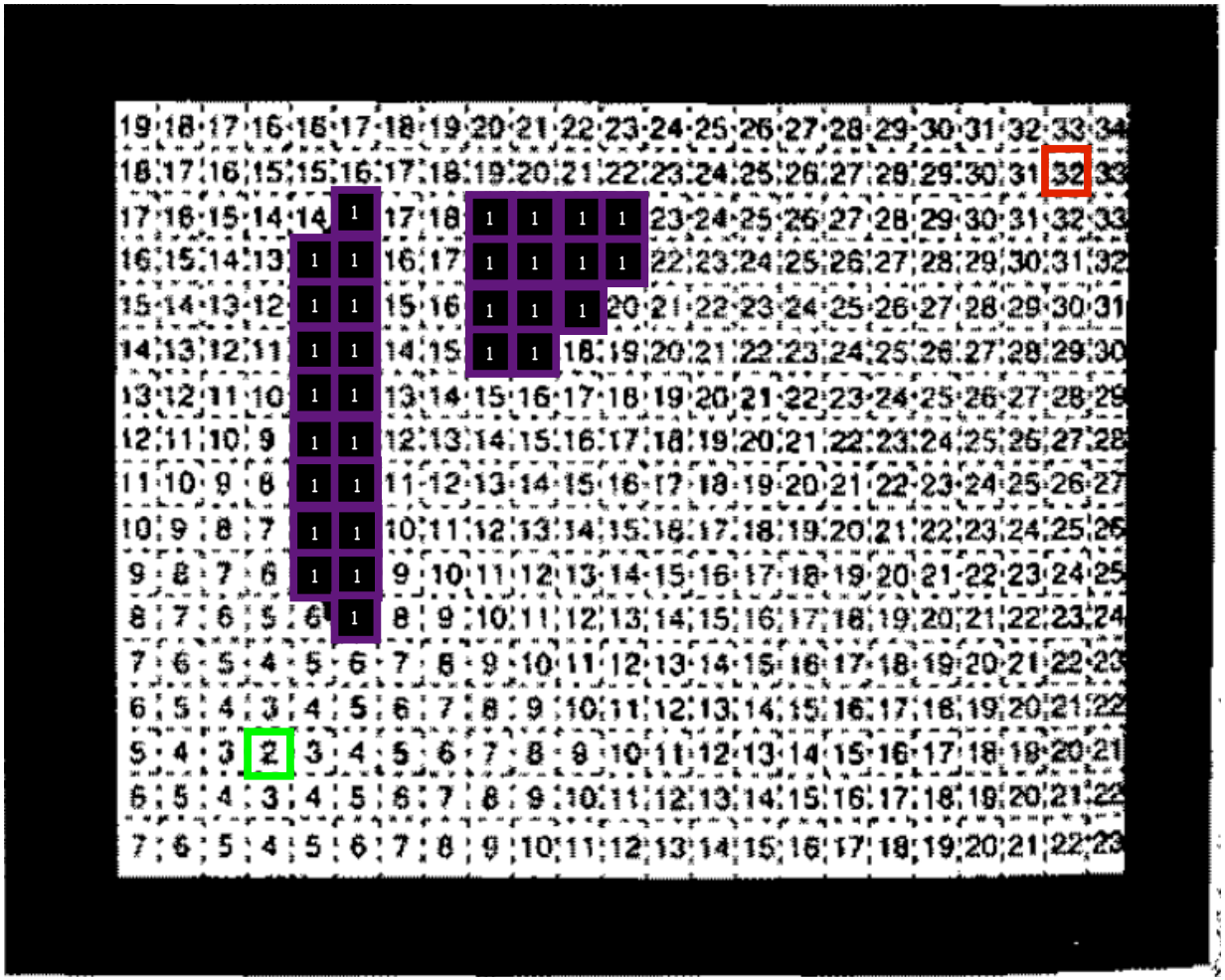

0
Brushfire away from goal:
energy at empty cells on frontier =
current energy at frontier +
change in cell energy (1 in this case)


0
1
13
2

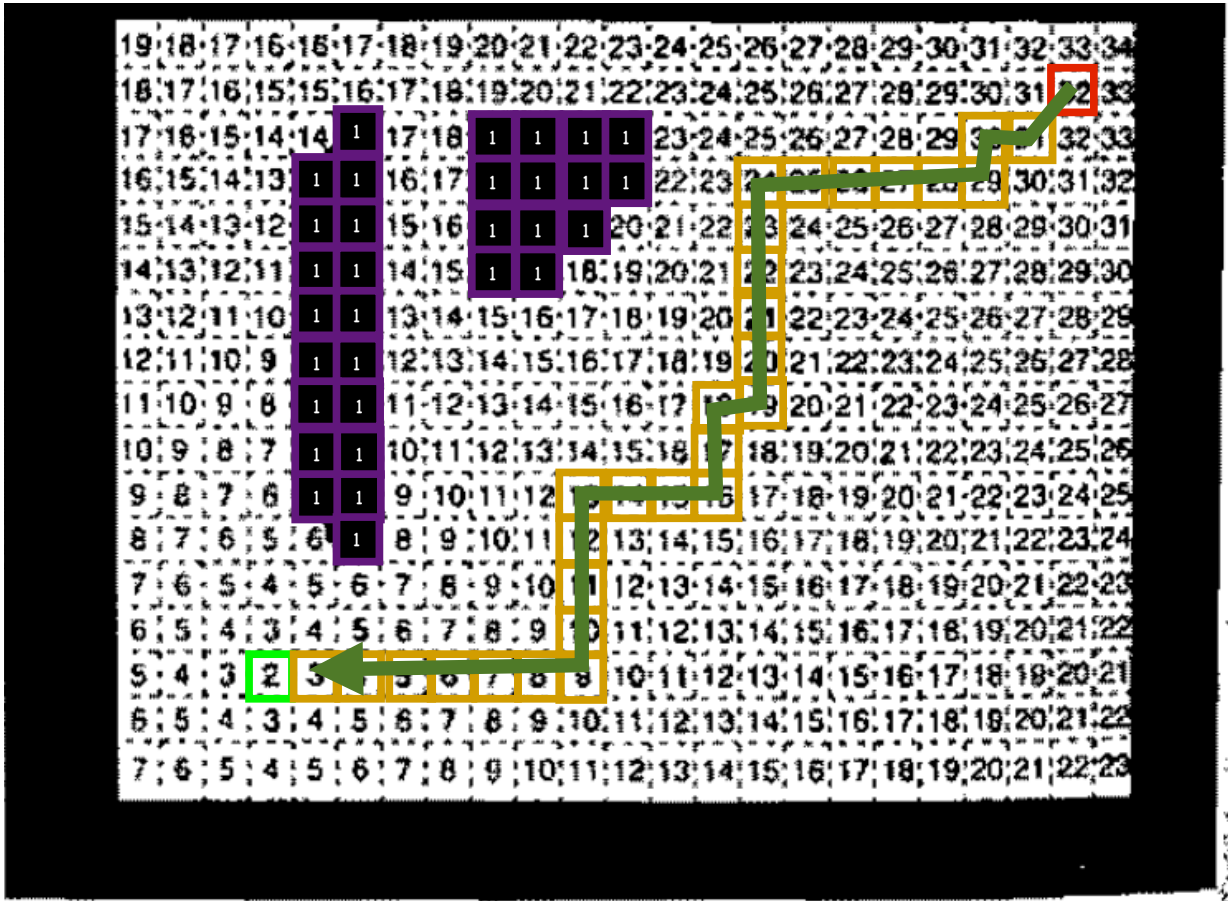

0
2


0
2


0
2

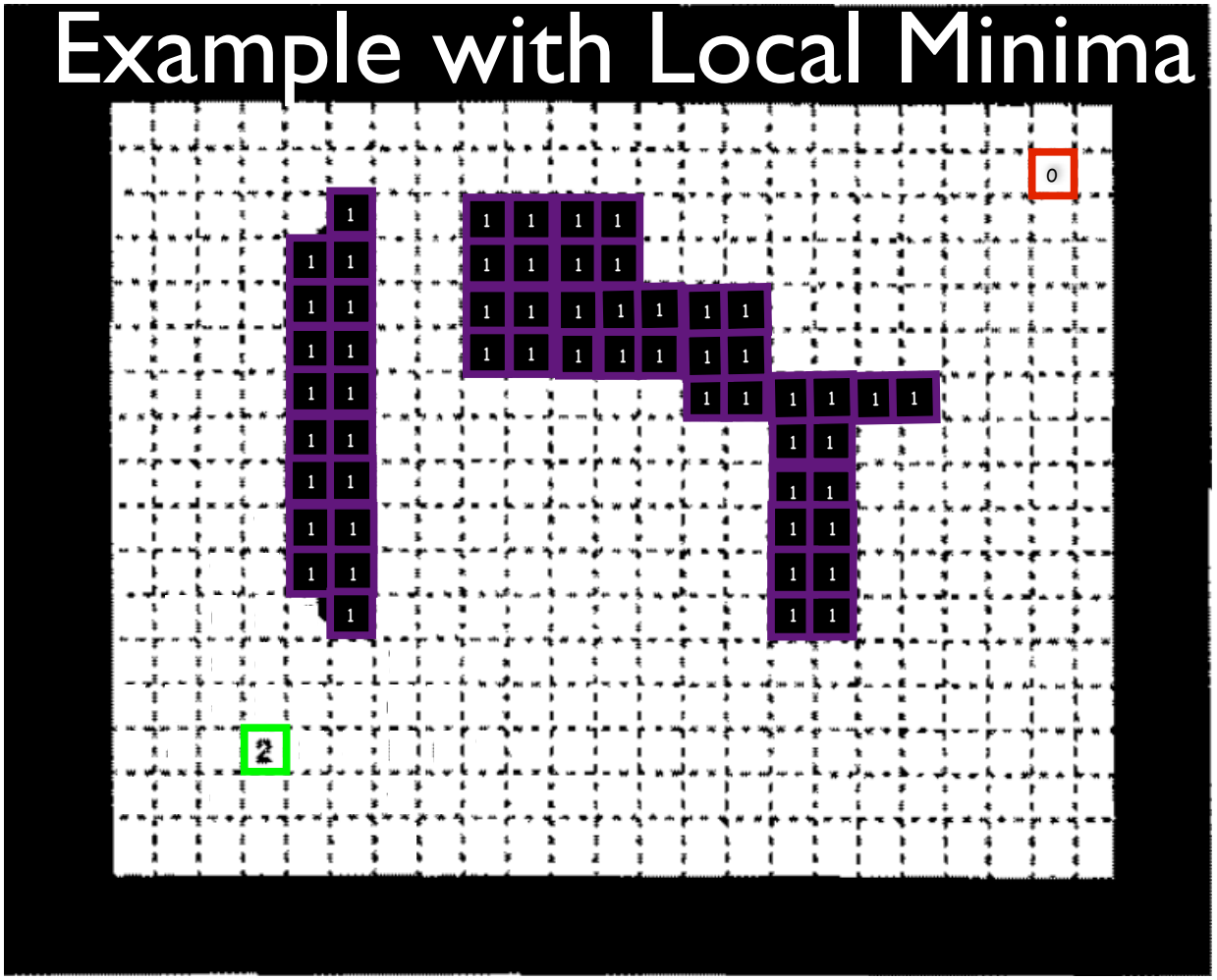


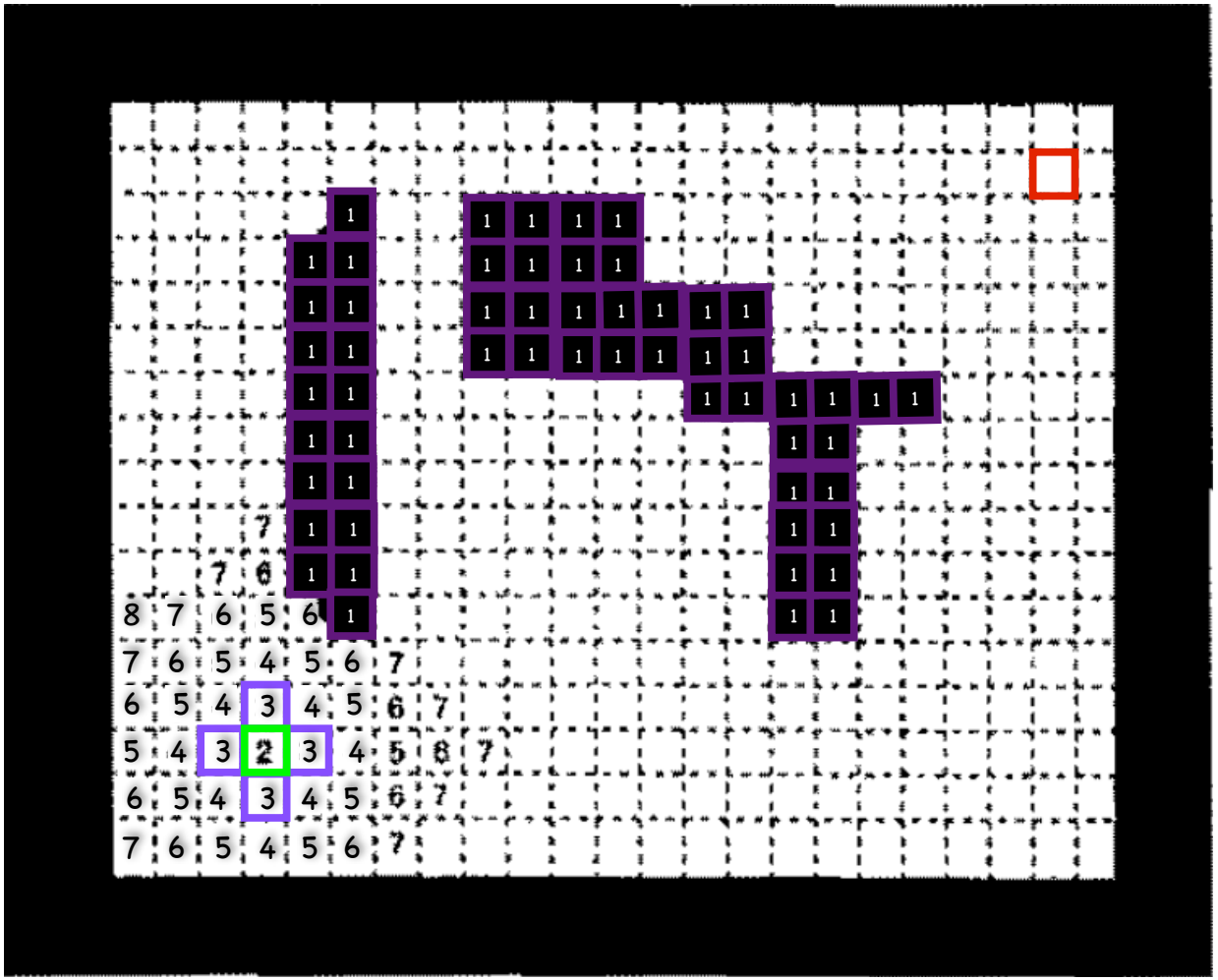
Once start reached, follow brushfire potential to goal

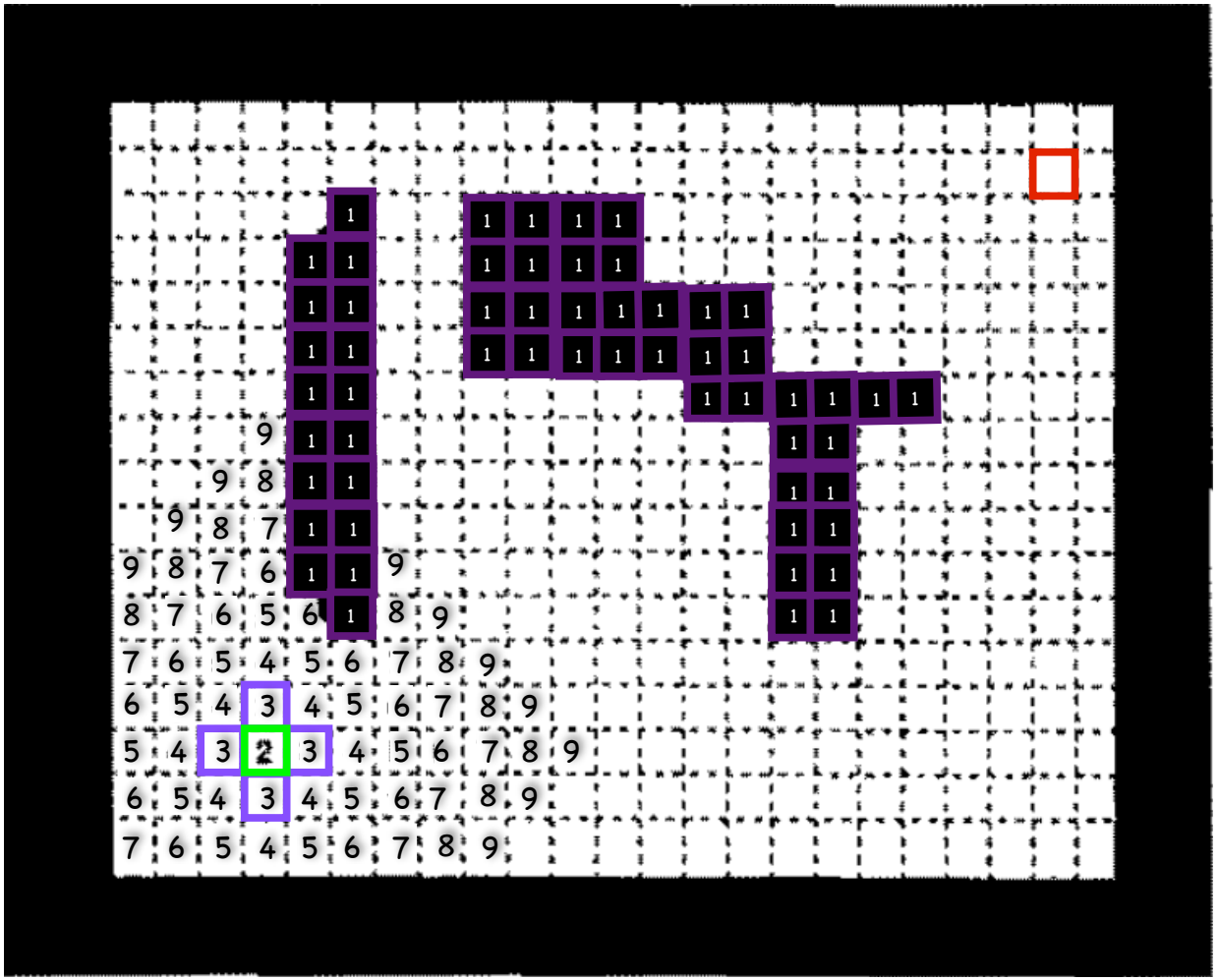


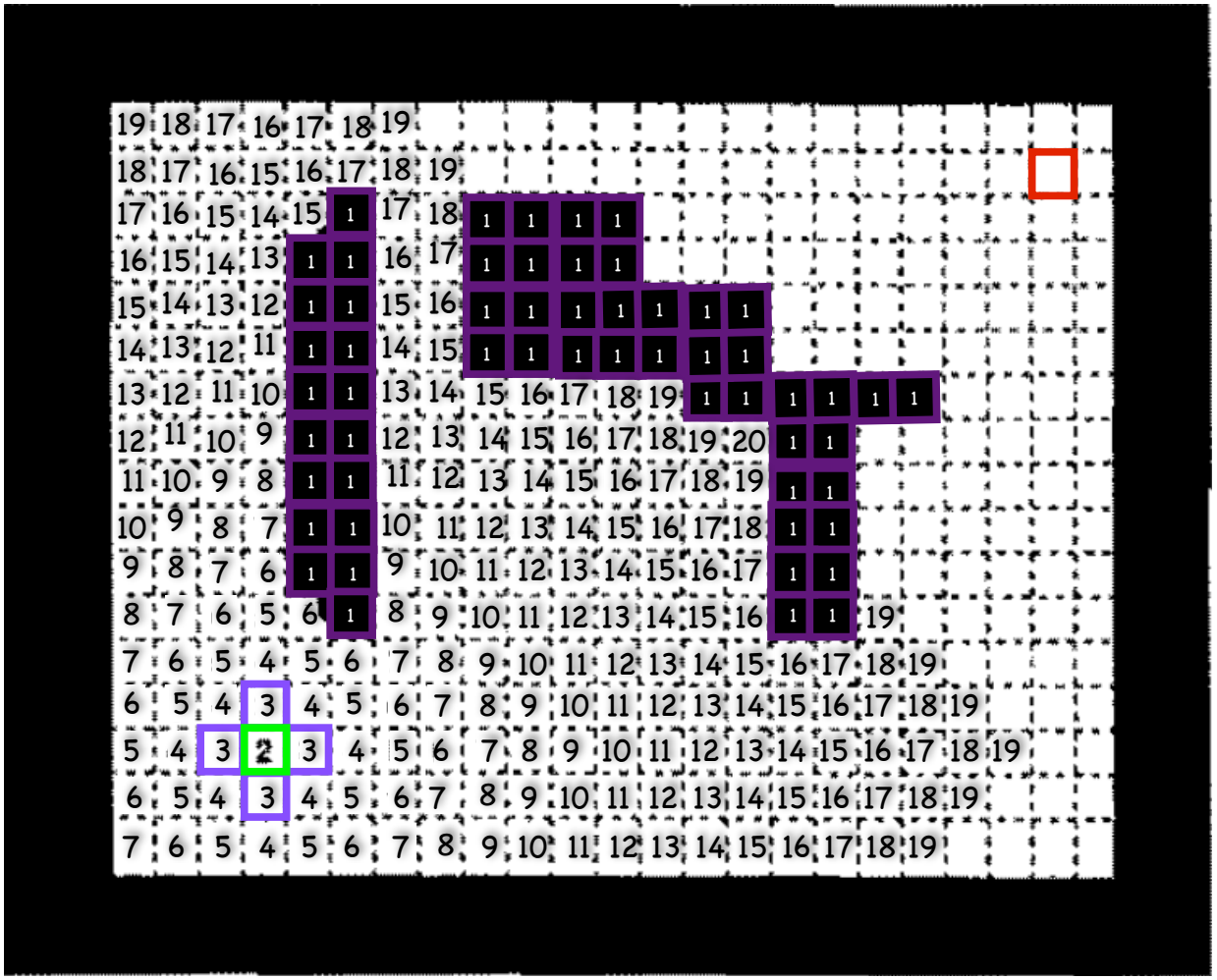
Example with Local Minima

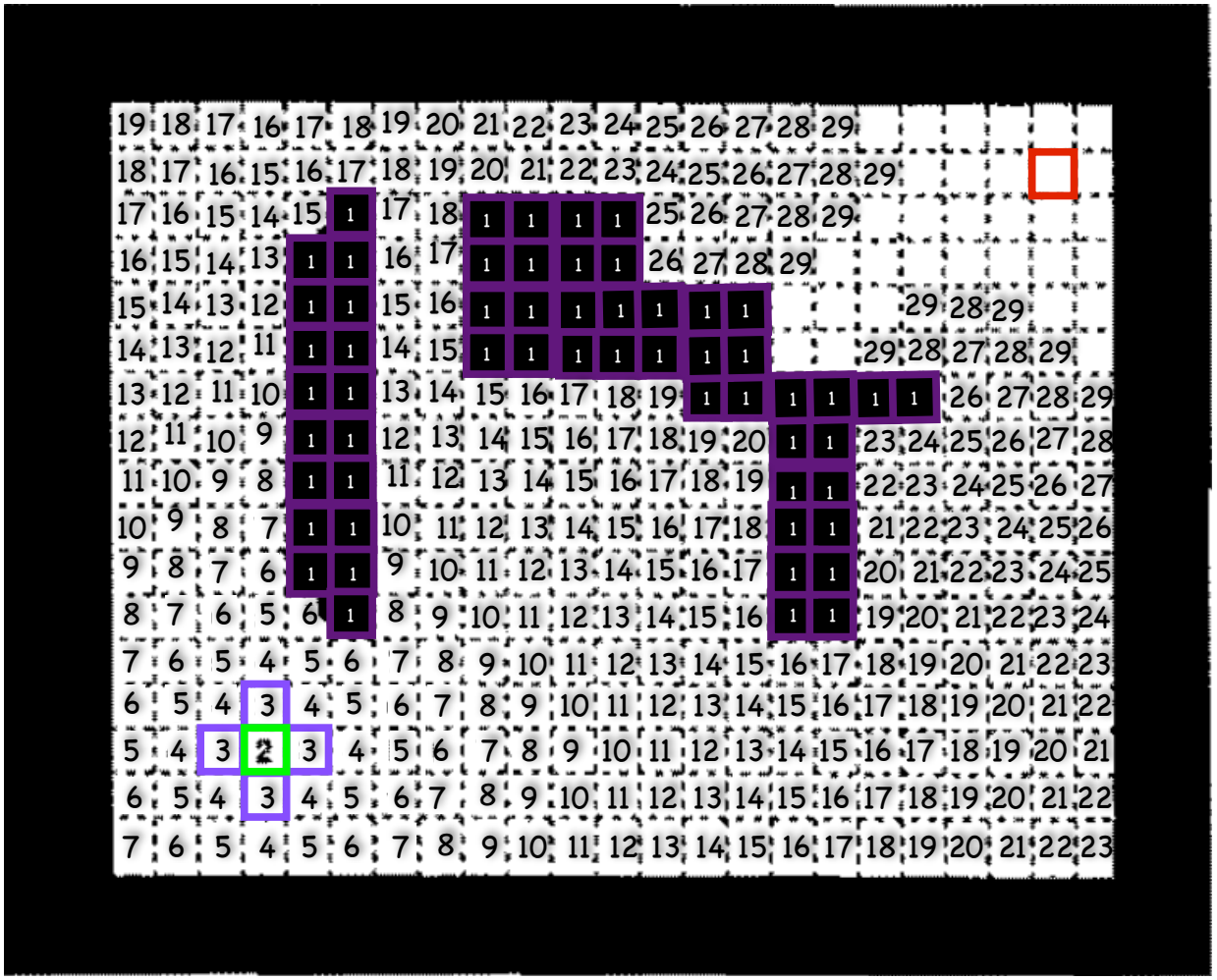
Example with Local Minima

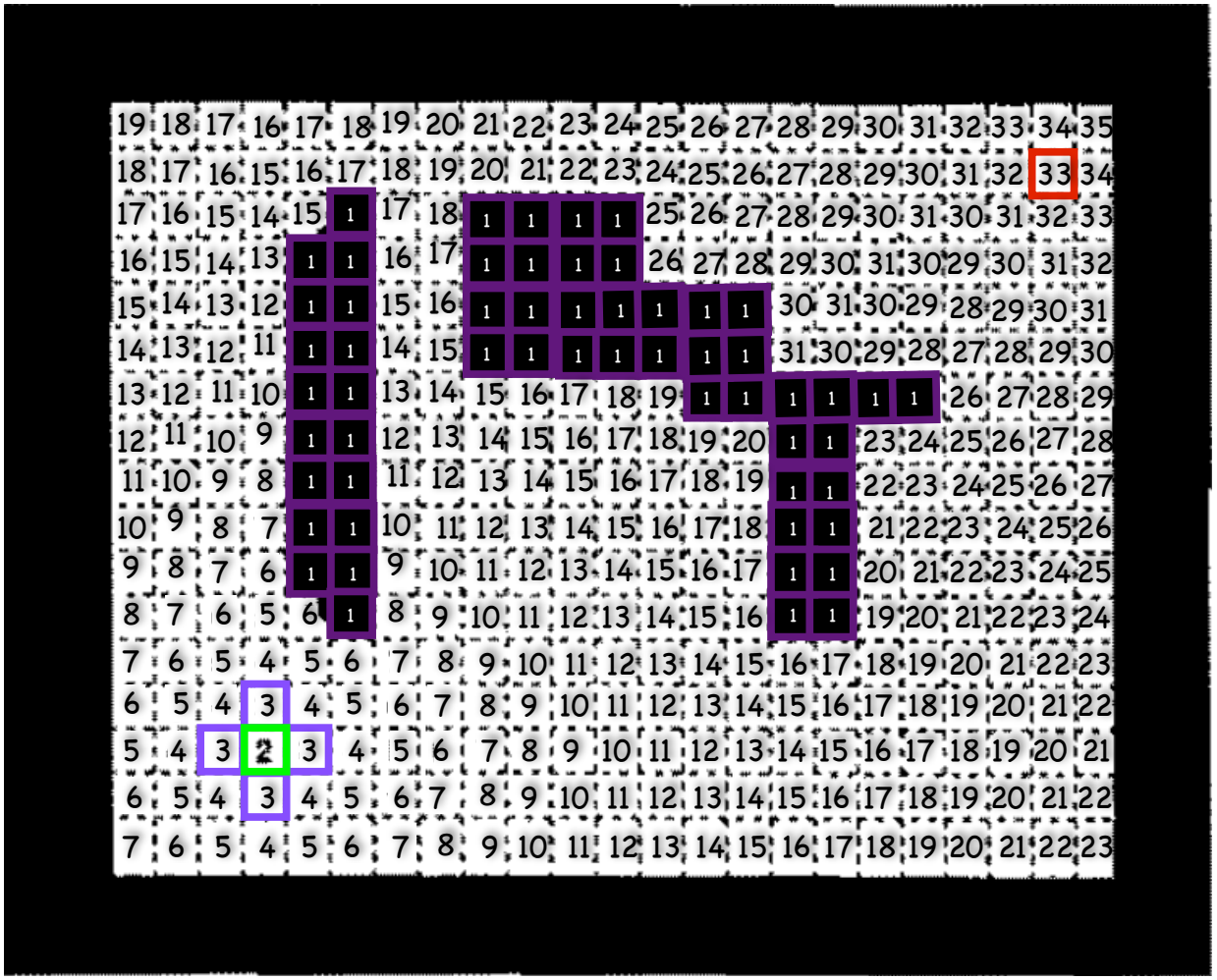


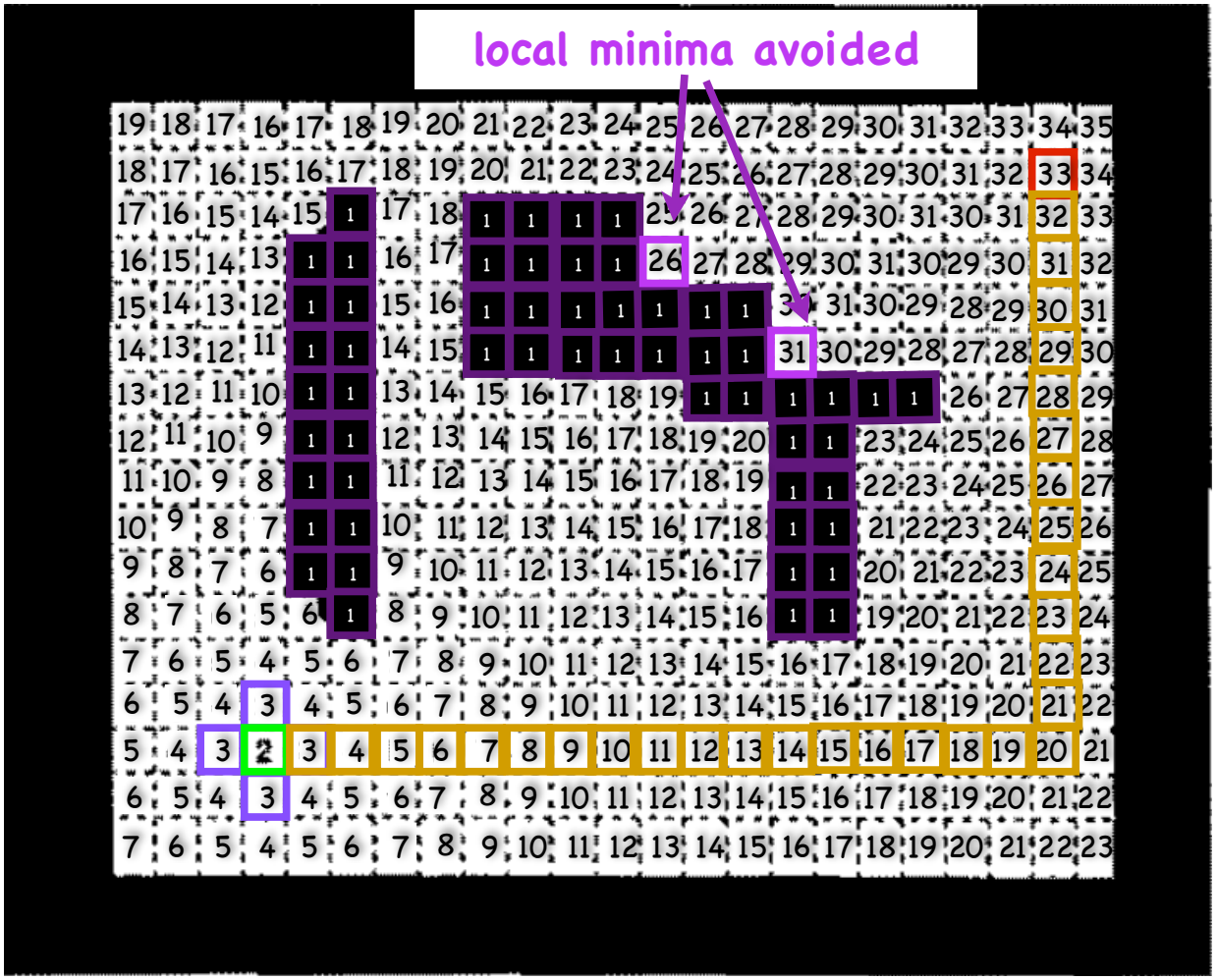




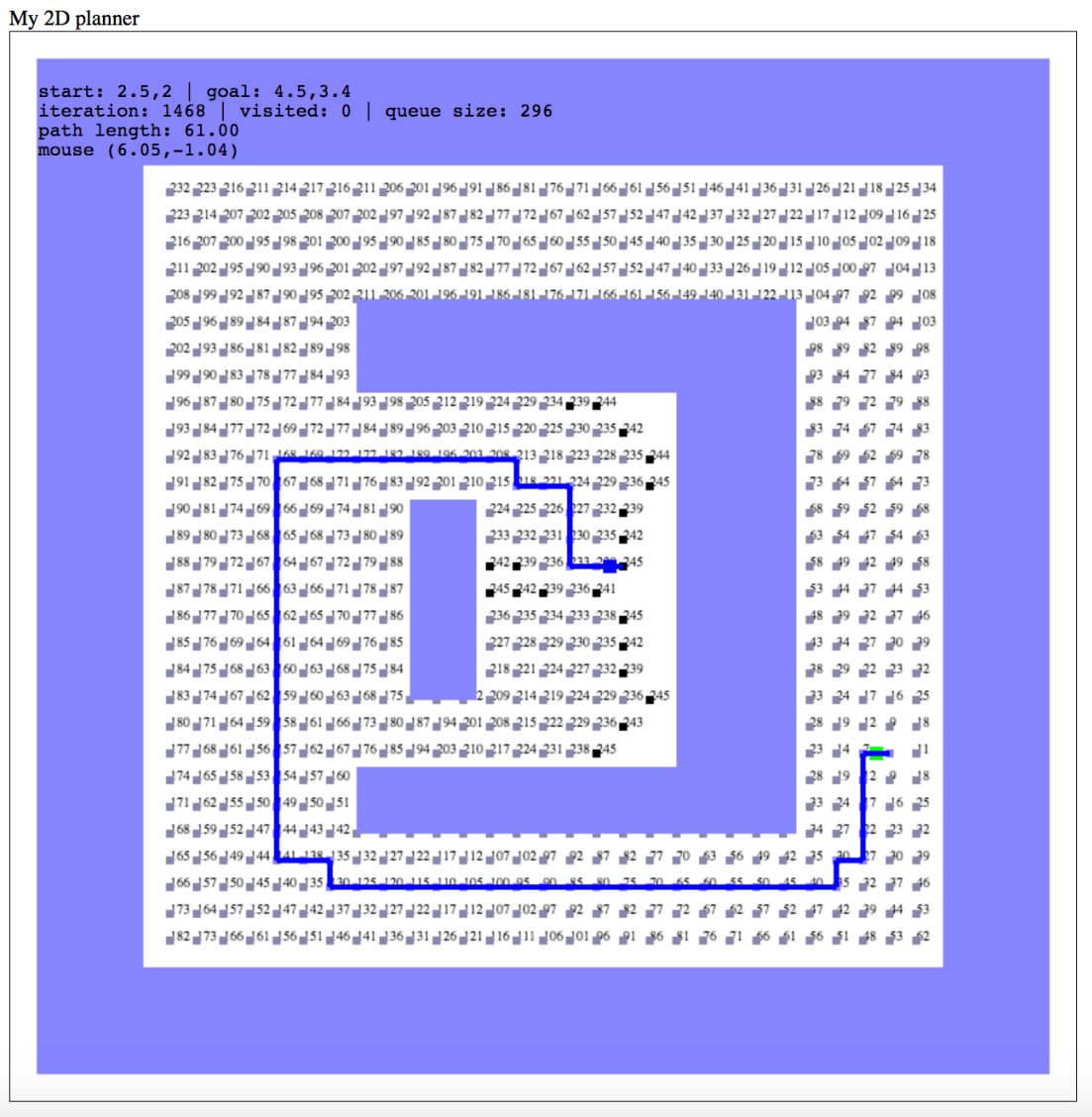







local minima avoided

Maithili's wavefront planner



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

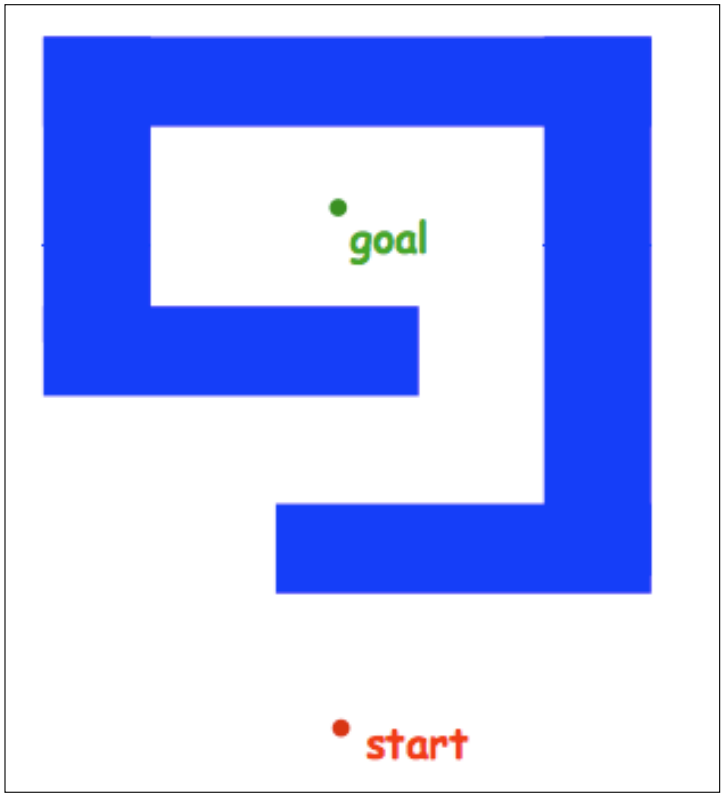
Grid Search/Wavefront
- Complete
- General C-spaces
- Limited dimensionality

Random walk
- Will find path eventually

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

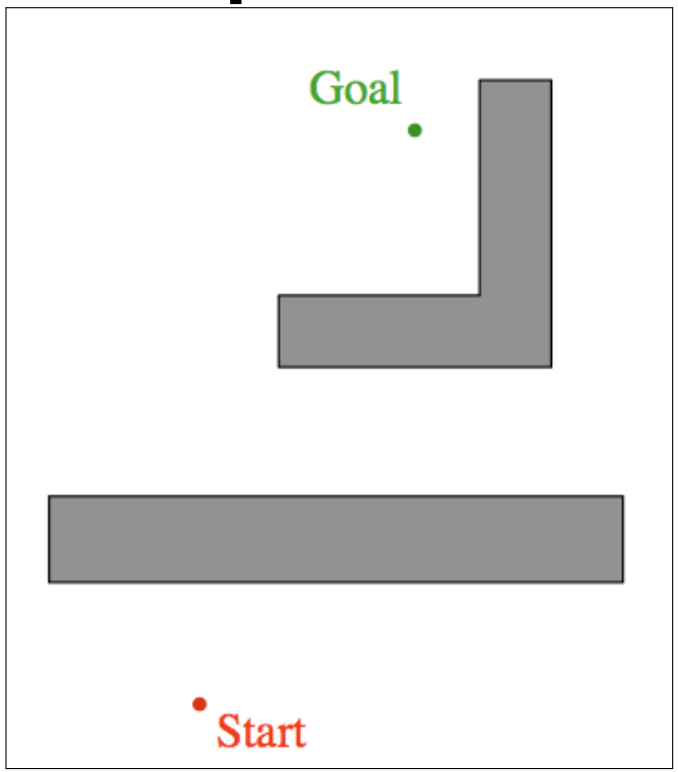
Compare: BugX examples

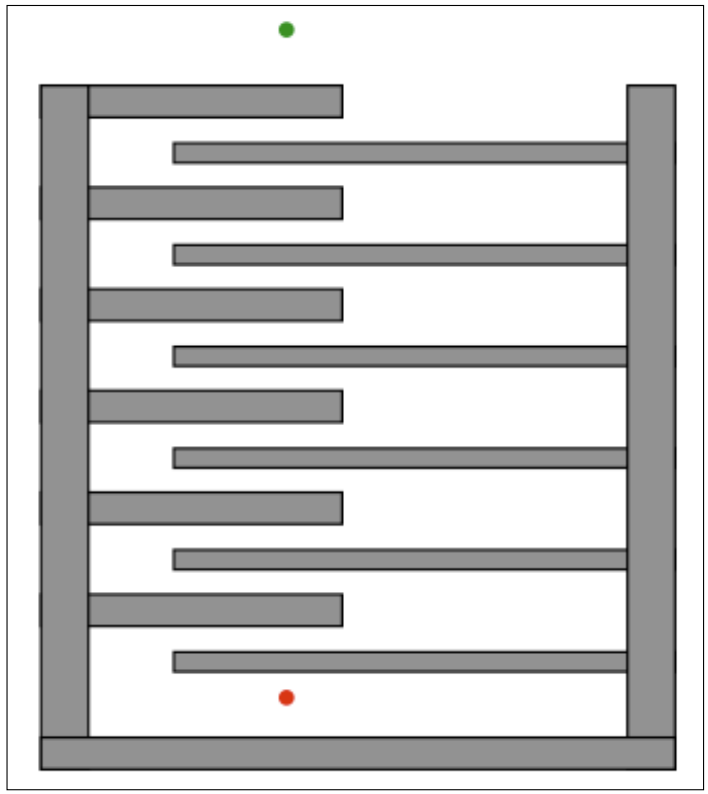
Describe likely paths from potential field and wavefront planners



Compare: BugX examples

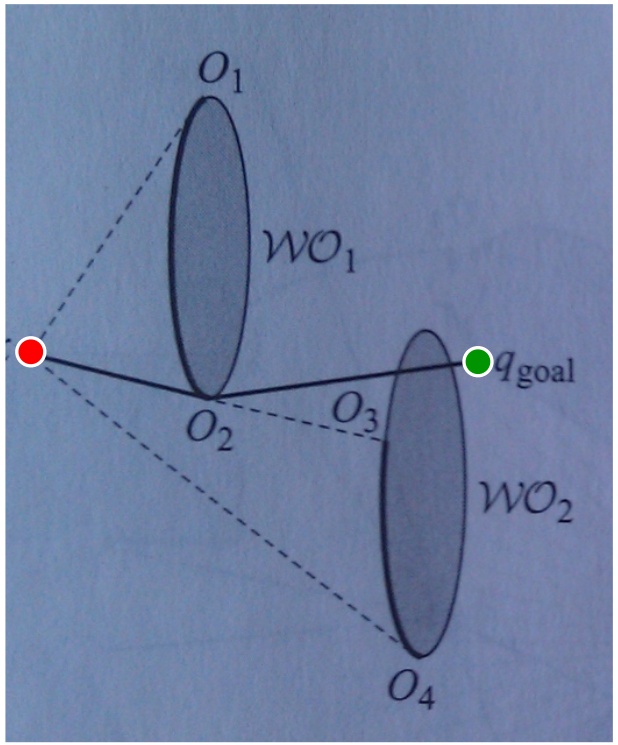
Describe likely paths from potential field and wavefront planners

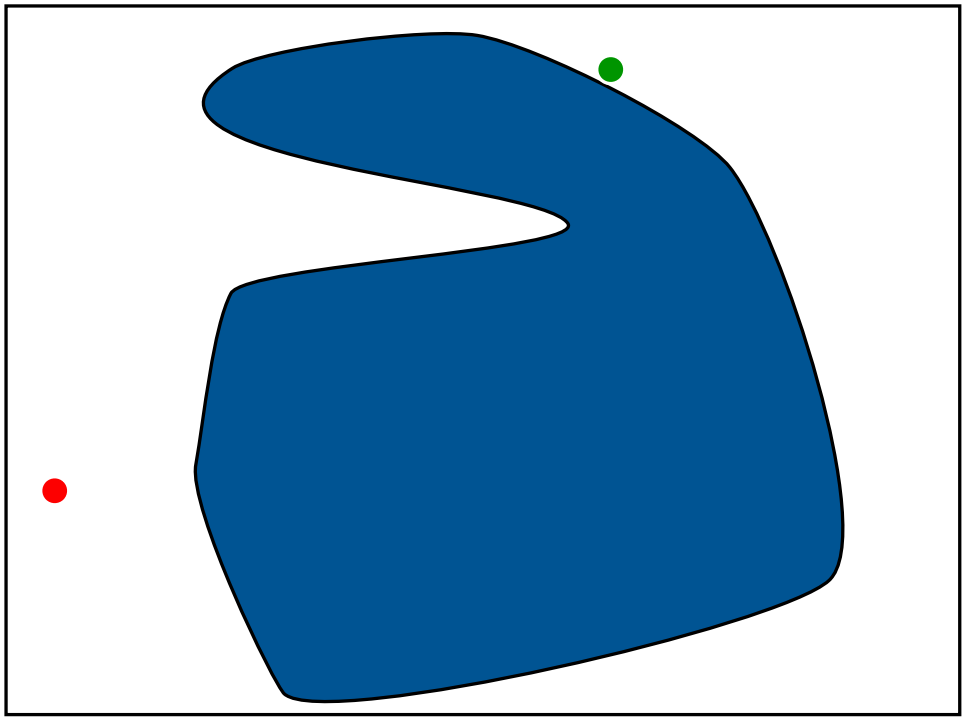




Compare: BugX examples

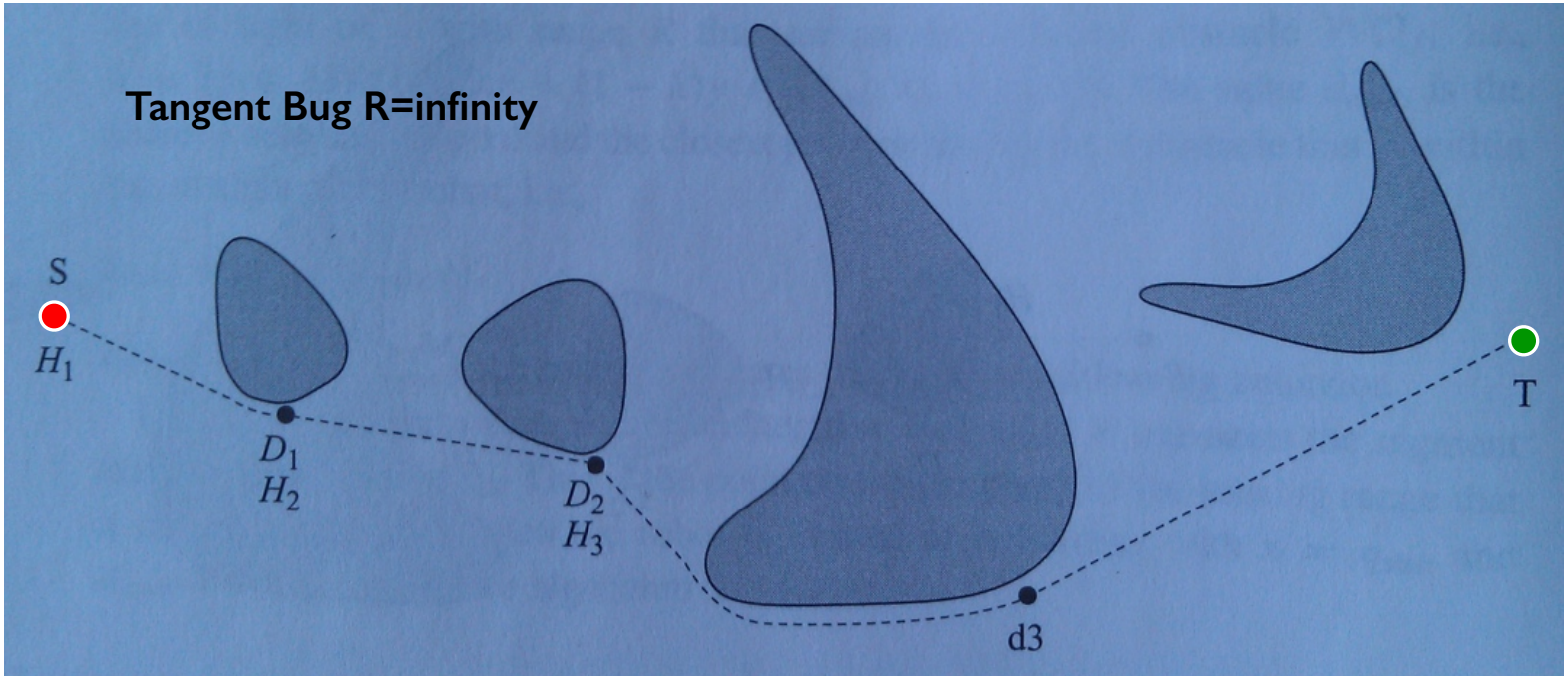
Describe likely paths from potential field and wavefront planners





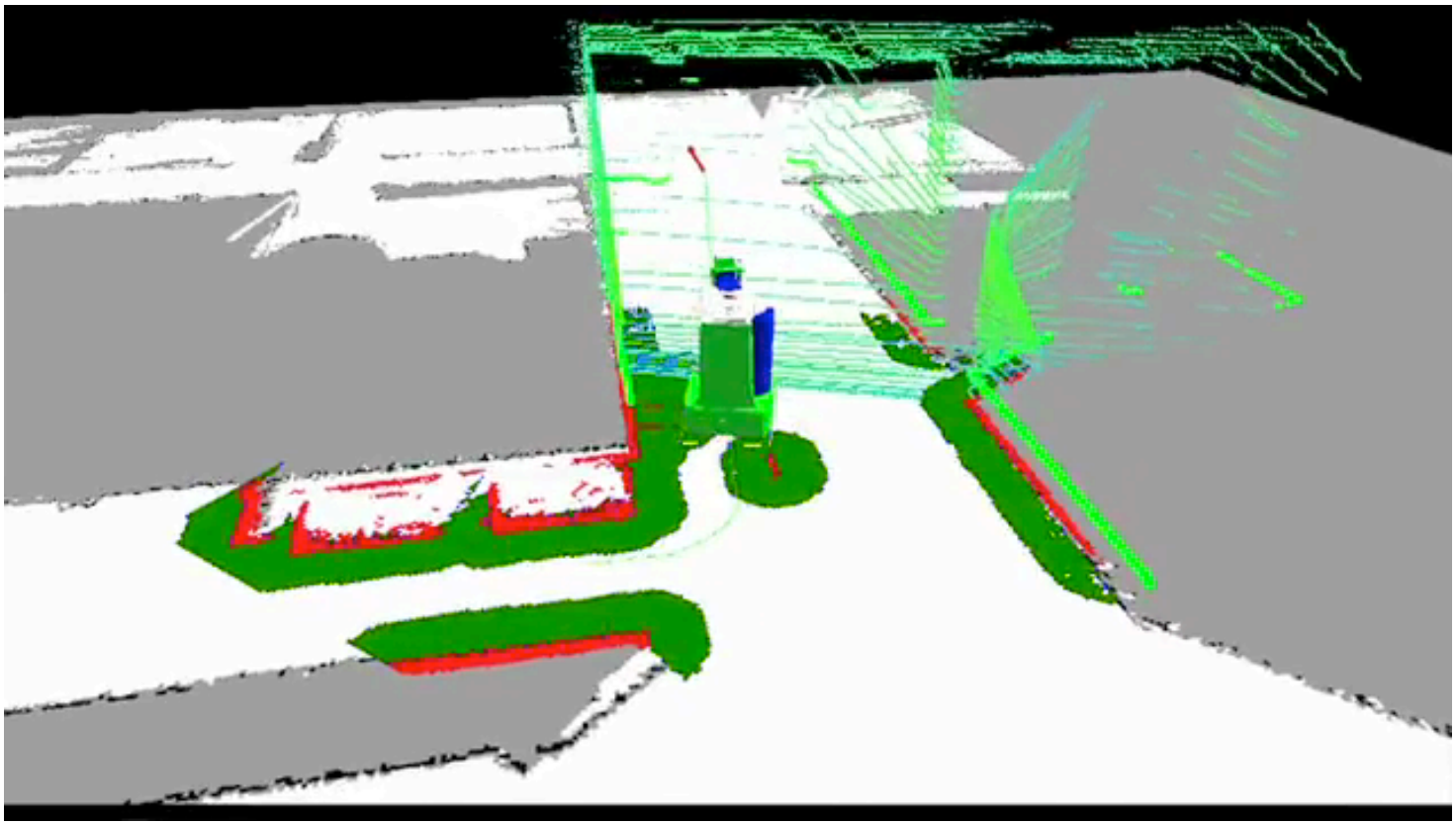
Compare: BugX examples

Describe likely paths from potential field and wavefront planners



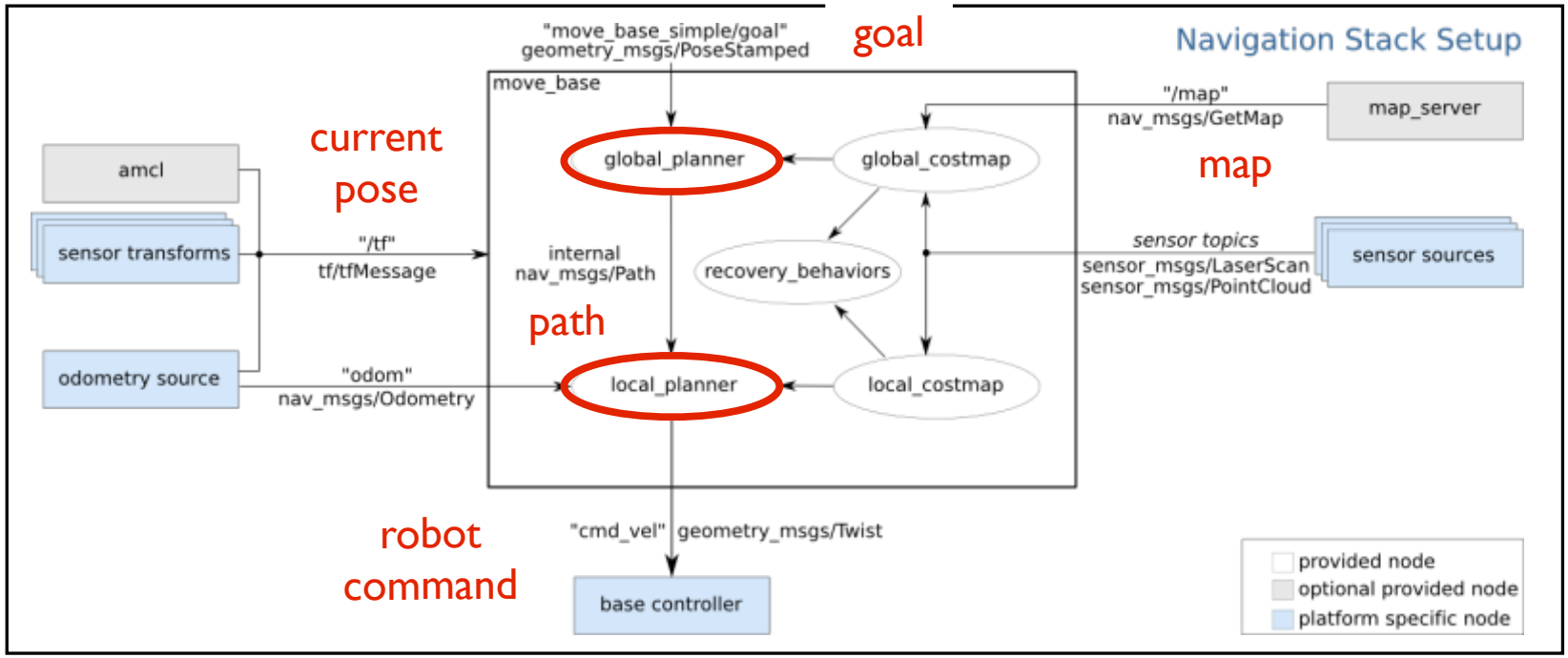
Navigation in ROS

ROS Navigation Stack



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

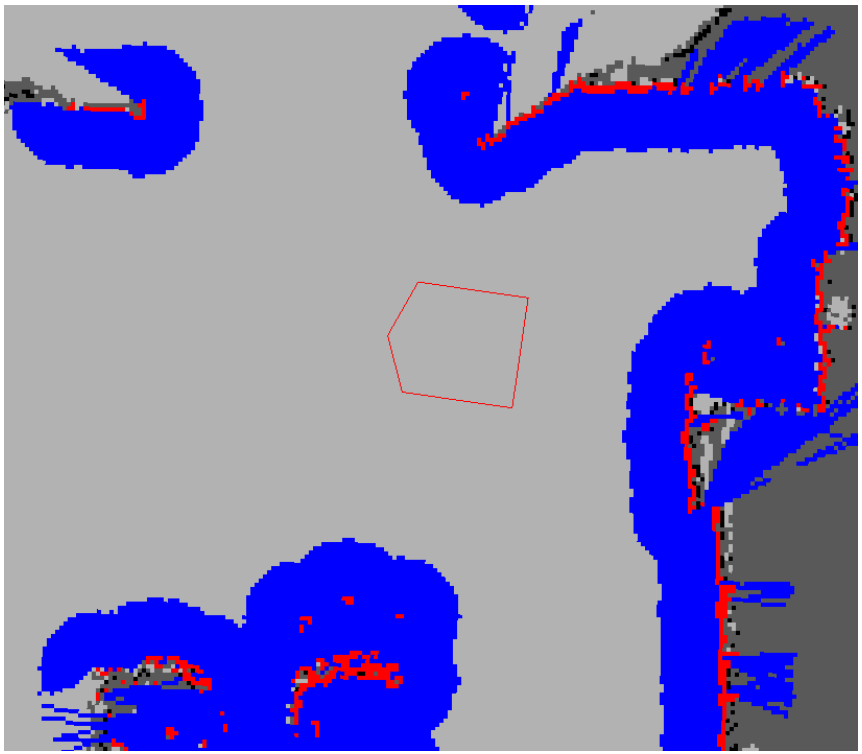
move_base



- Core of nav_stack for planning and motion control

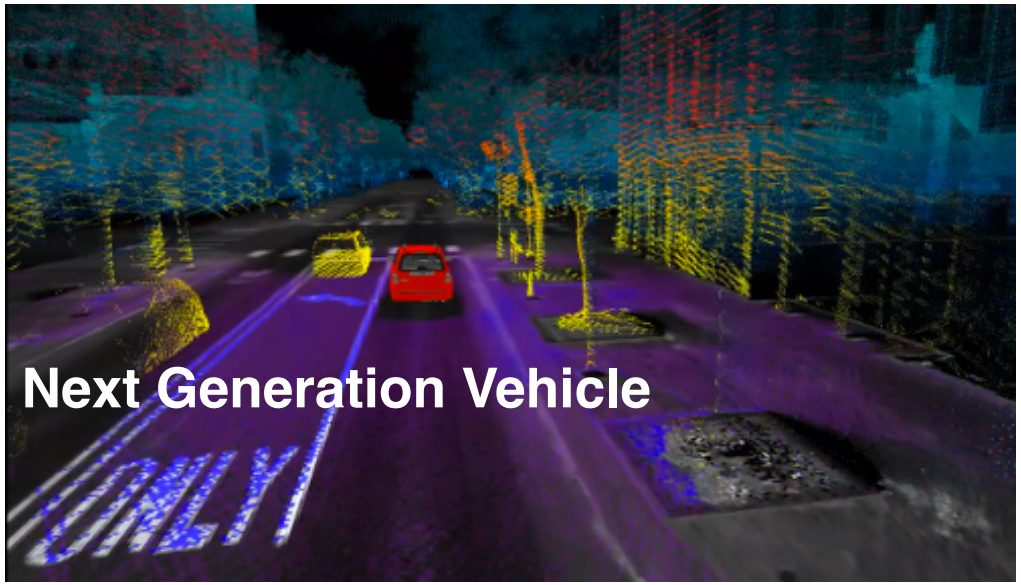
Costmap in ROS

- Treat as a Minkowski sum over the given map



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

Navigation in LCM

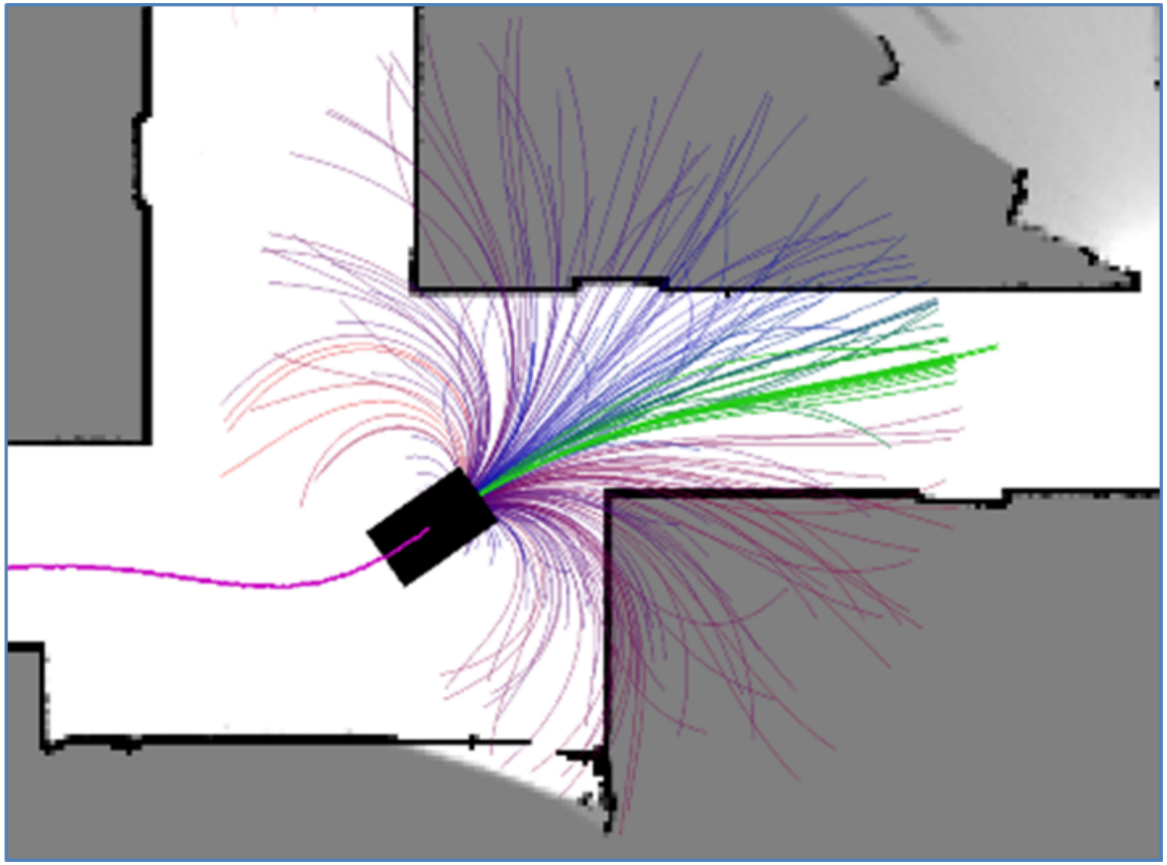

Next Generation Vehicle


BotLab EECS 467 / ROB 550


MagicBots

[Park, Johnson, Kuipers 2012]

Vulcan Navigation



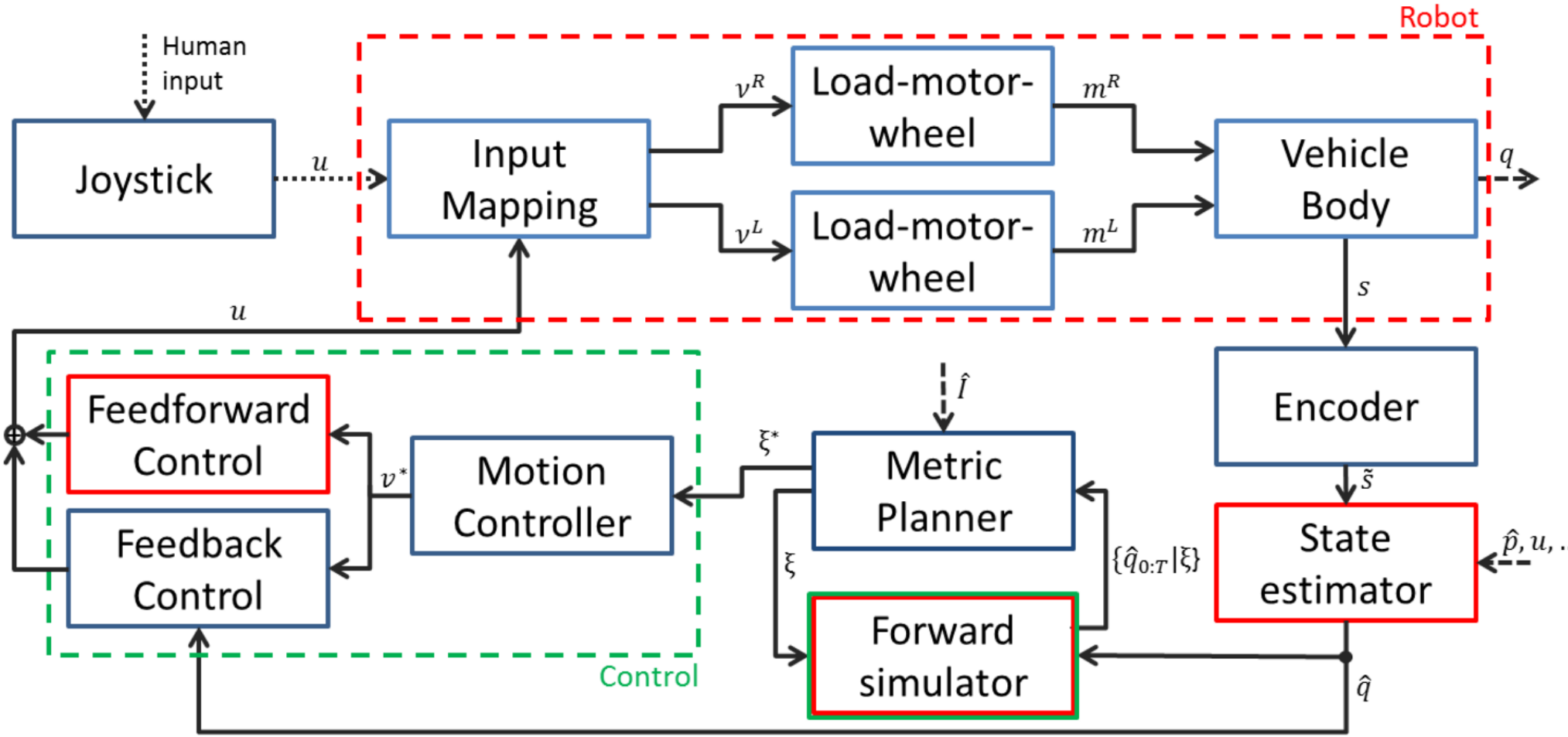
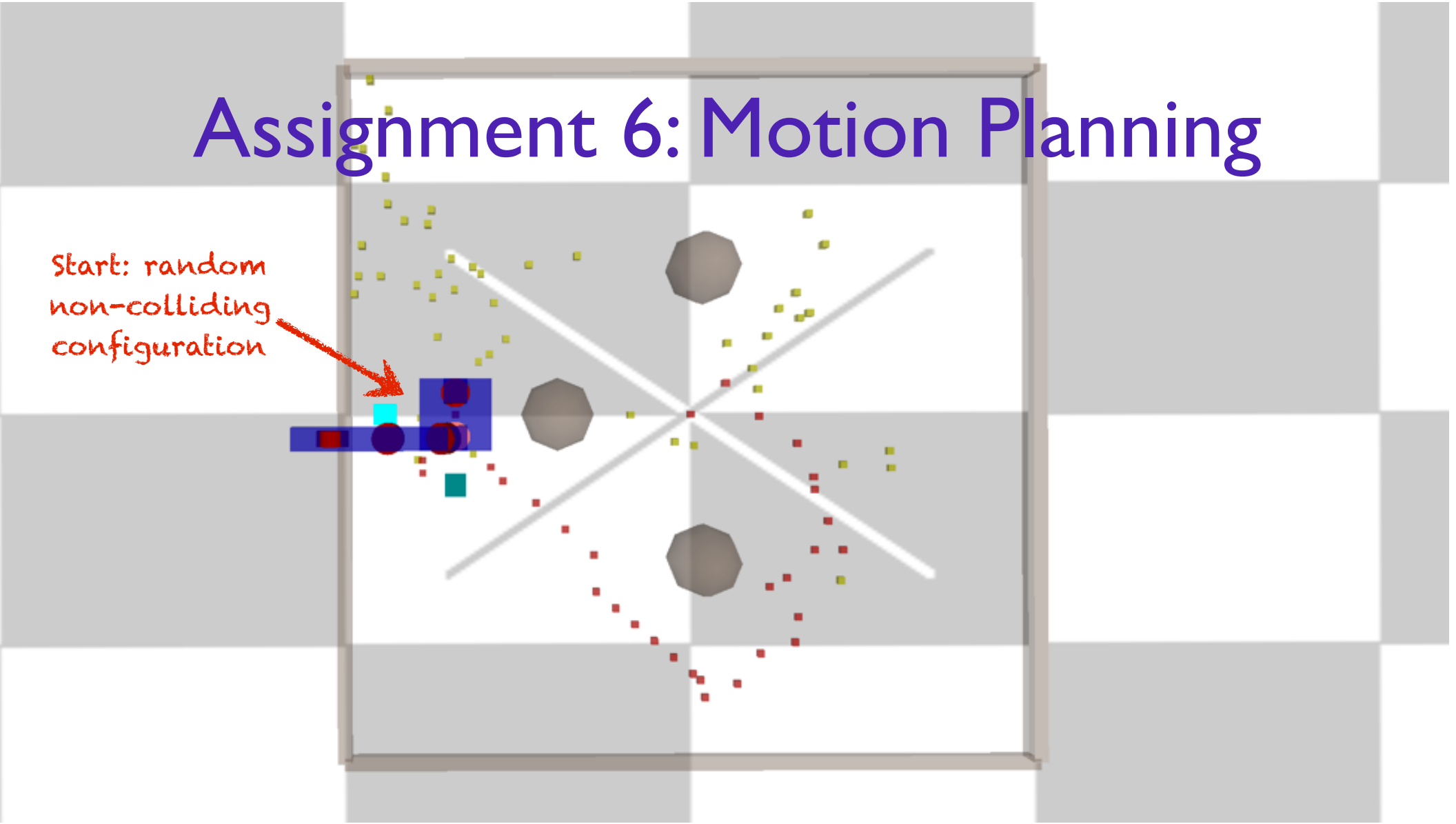


Figure A.1: Vulcan system diagram.

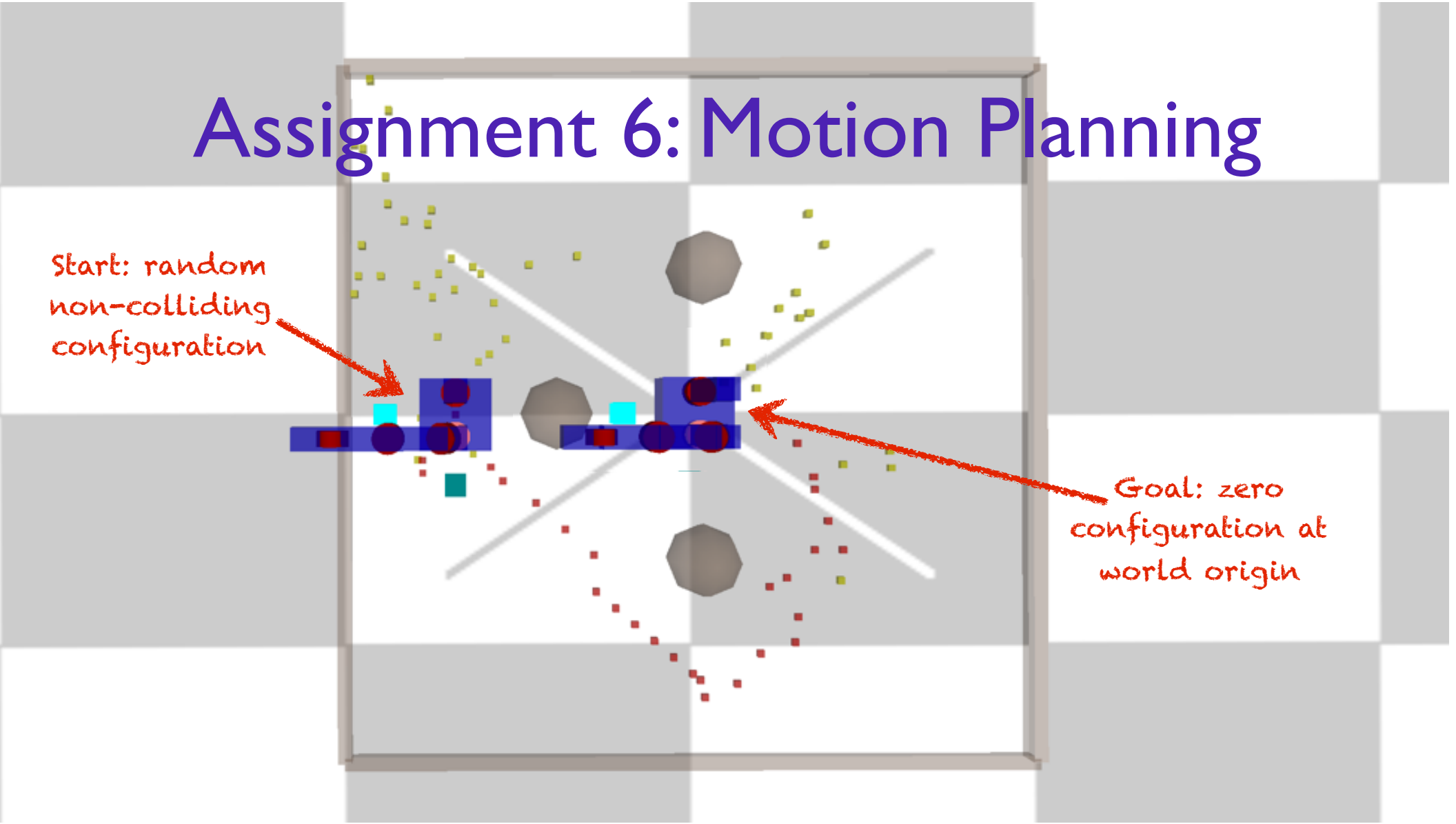
Assignment 6: Motion Planning

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

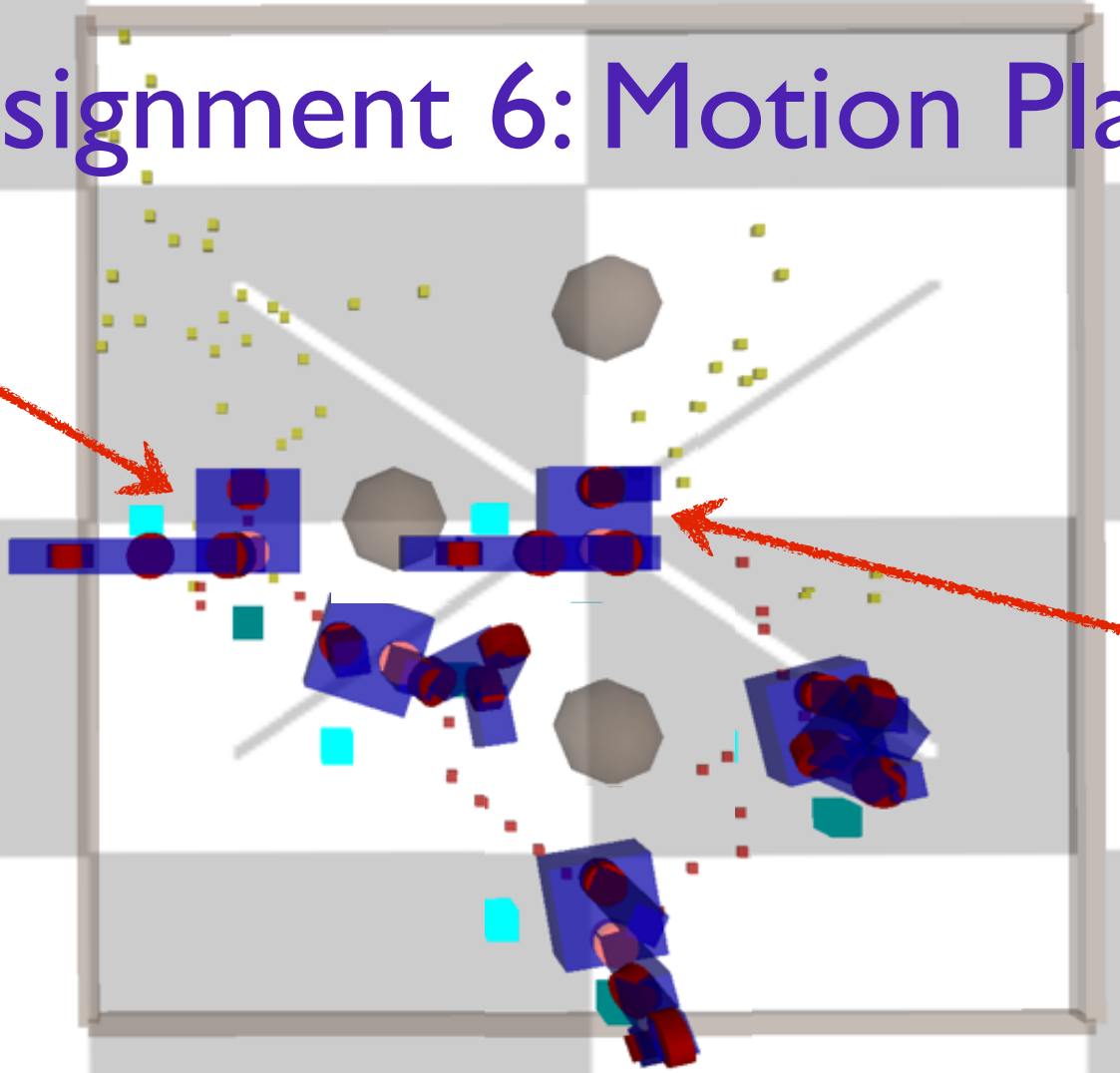
Assignment 6: Motion Planning



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

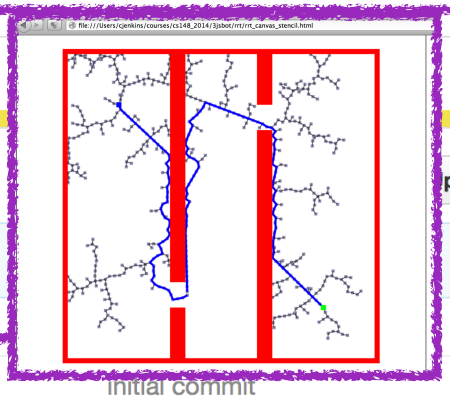
ohseejay / **kineval-stencil-fall16**

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

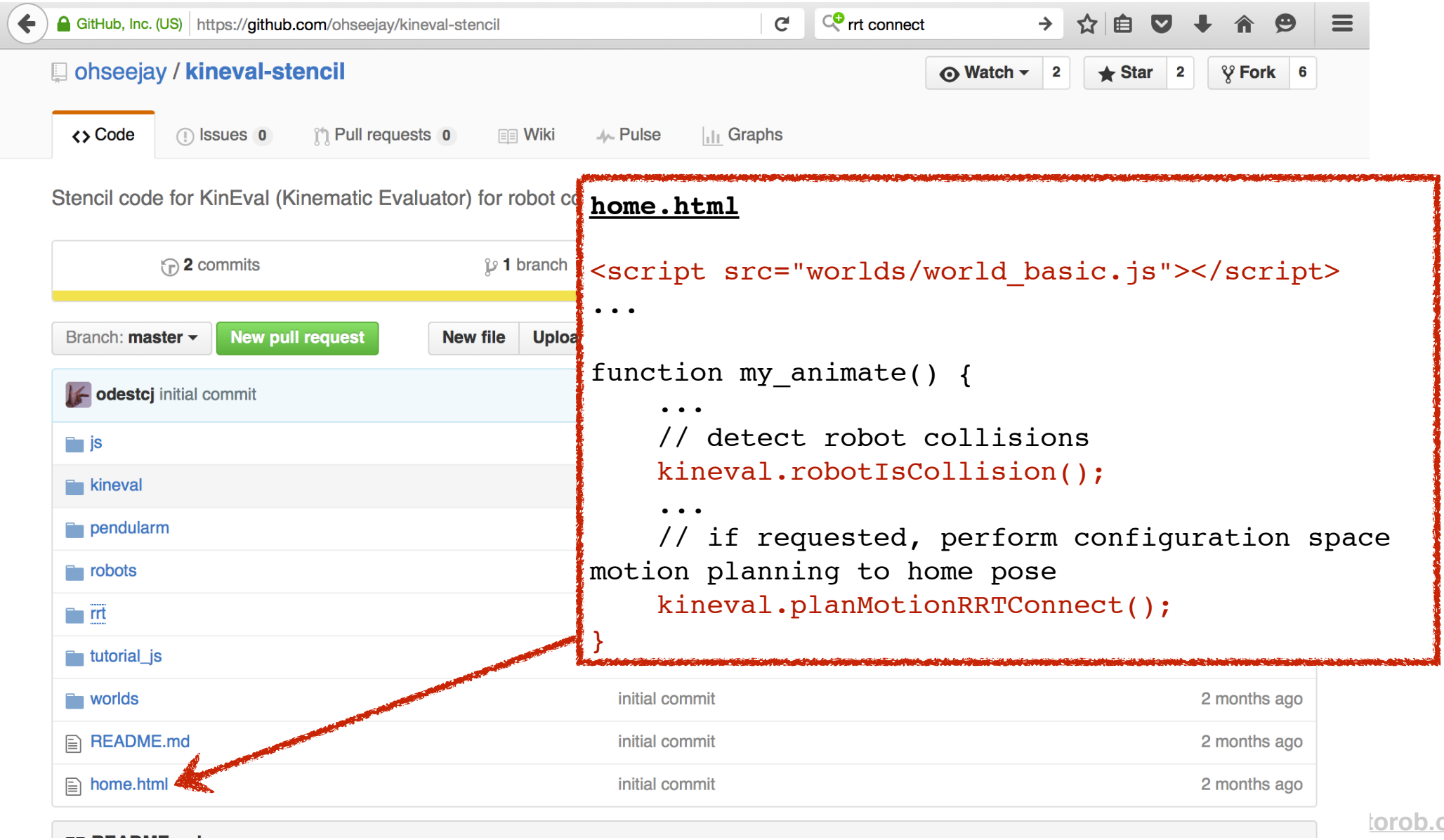
Branch: master New pull request pload files Find file Clone or download

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

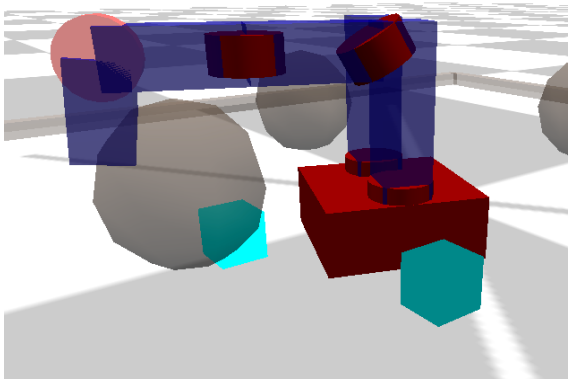
rob.org


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

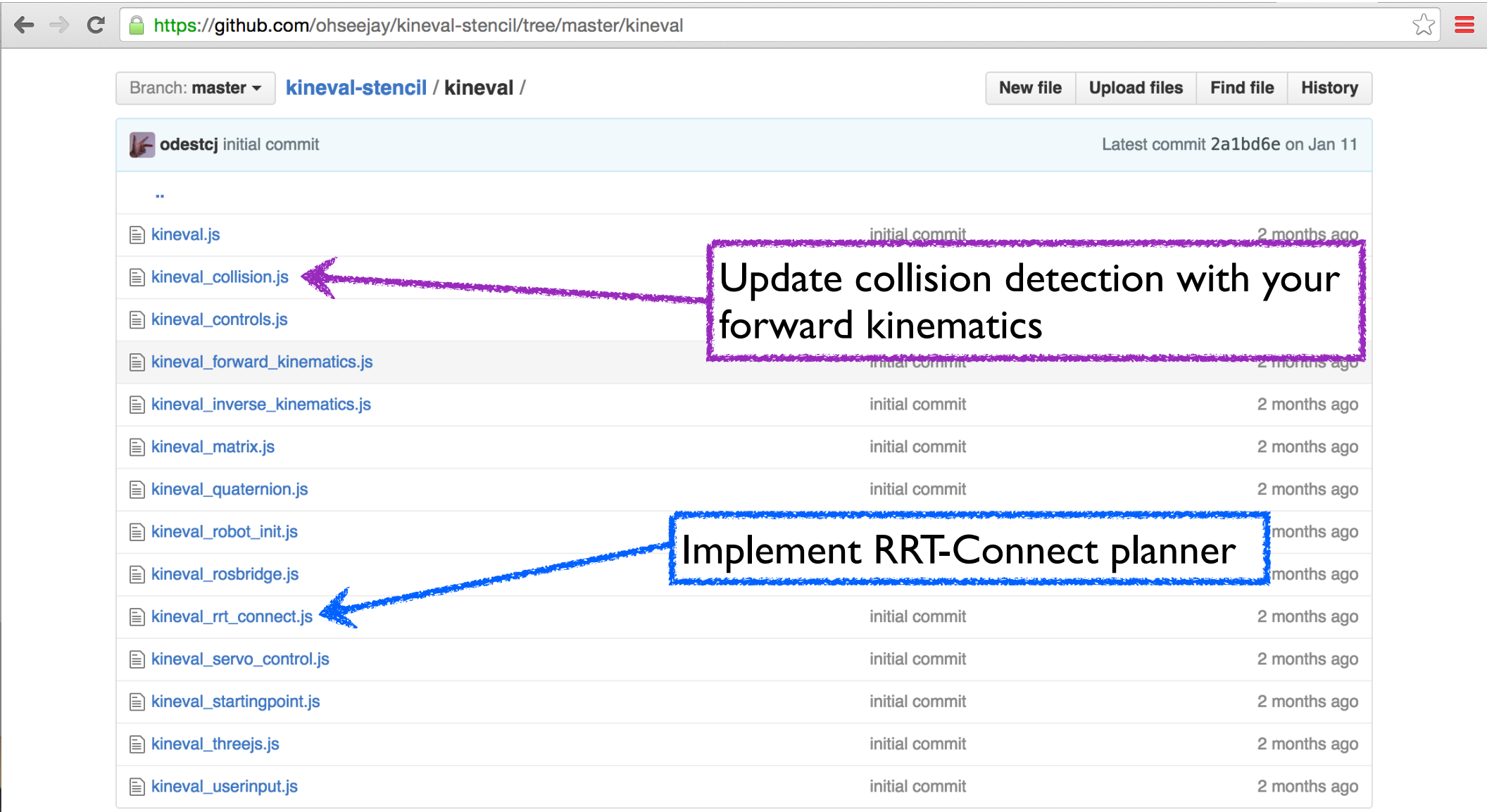
home.html

```
<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);

}
```

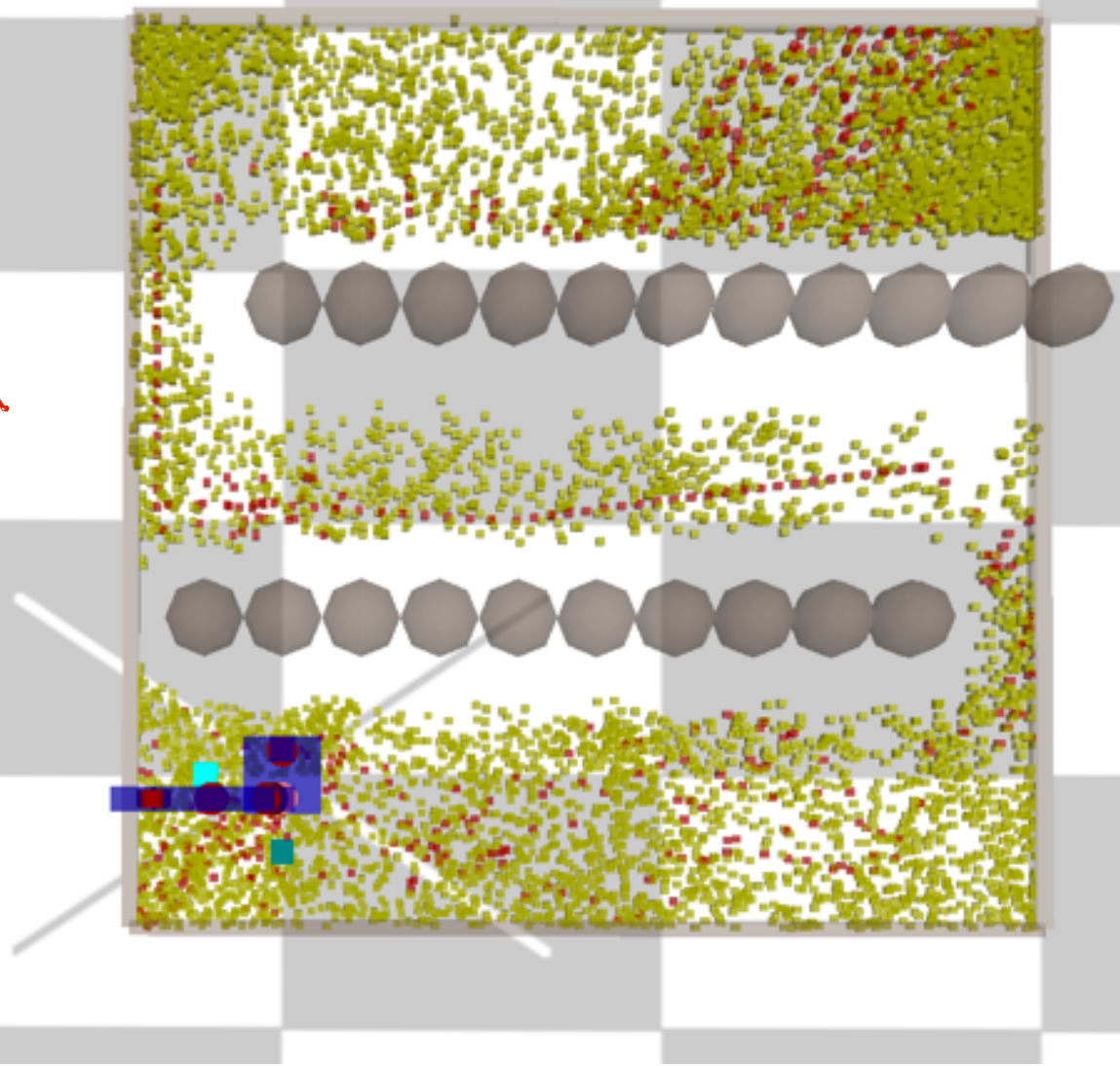
"electric fence" world boundary detection is provided

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!




Next Lecture:
Collision Detection