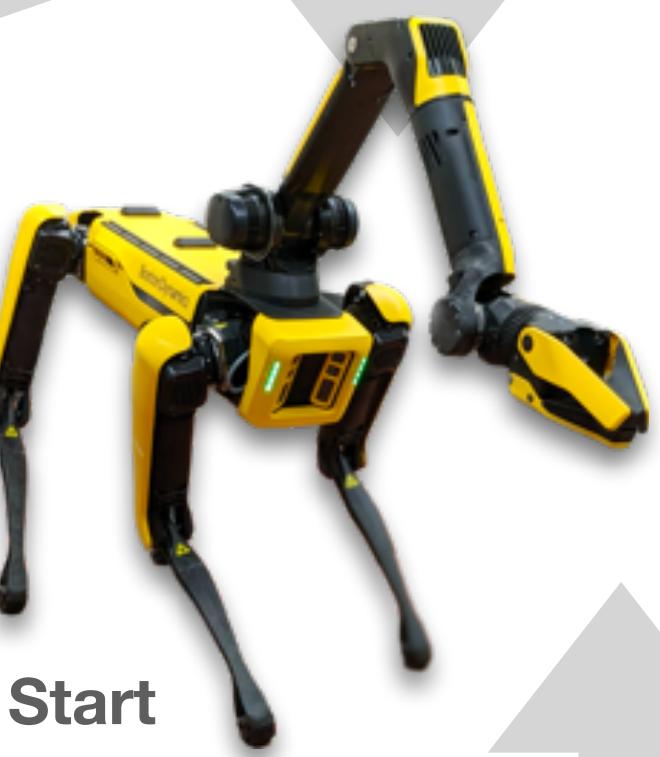


Lecture 13

Planning - V - Potential Fields



Course Logistics

- Quiz 11 was posted today and was due before the lecture.
- Project 3 was posted on 10/11 and is due on 10/25.
 - Please look into updates posted by Chahyon Ku on Ed discussions!



RRT Algorithm



RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD-RRT( $\mathcal{G}_{init}$ )  
1  $T \leftarrow \mathcal{G}_{init}$ ;  
2 for  $k = 1$  to  $K$  do  
3    $\theta_{rand} \leftarrow \text{RANDOM-CONFIG}()$   
4   EXTEND( $T$ ,  $\theta_{rand}$ );  
5 return  $T$ 
```

RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD-RRT( $q_{start}$ , K)
1  $T \leftarrow \text{INIT}(\mathbb{Q}_{free})$ 
2 for  $k = 1$  to  $K$  do
3    $q_{rand} \leftarrow \text{RANDOM-CONFIG}()$ 
4   EXTEND( $T$ ,  $q_{rand}$ )
5 return  $T$ 
```



Figure 3: The EXTEND operation.

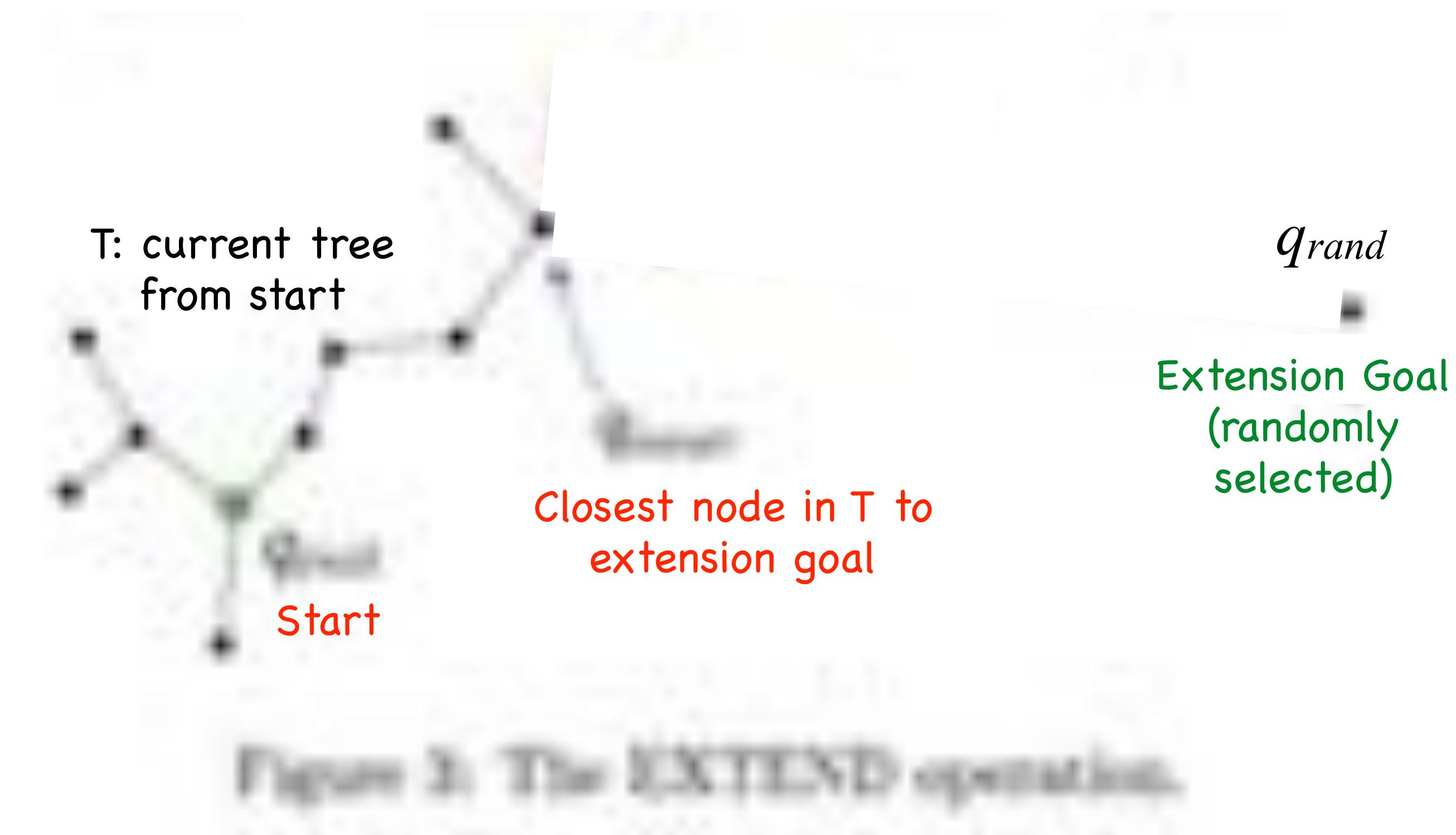
RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD-RRT( $q_{start}$ )
1  $T \leftarrow \text{init}(\text{graph})$ ;
2 for  $k = 1$  to  $K$  do
3    $q_{rand} \leftarrow \text{RANDOM-CONFIG}()$ ;
4   EXTEND( $T, q_{rand}$ );
5 return  $T$ 
```

```
EXTEND( $T, q$ )
1  $q_{near} \leftarrow \text{NEAREST-NEIGHBOR}(q, T)$ ;
2 if NEW-CONFIG( $q, q_{near}, q_{near}$ ) then
3    $T.add\text{-vertex}(q_{near})$ ;
4    $T.add\text{-edge}(q_{near}, q_{near})$ ;
5 if  $q_{near} = q$  then
6   Return Reached;
7 else
8   Return Admissible;
9 Return Dropped.
```

Extend graph towards a random configuration



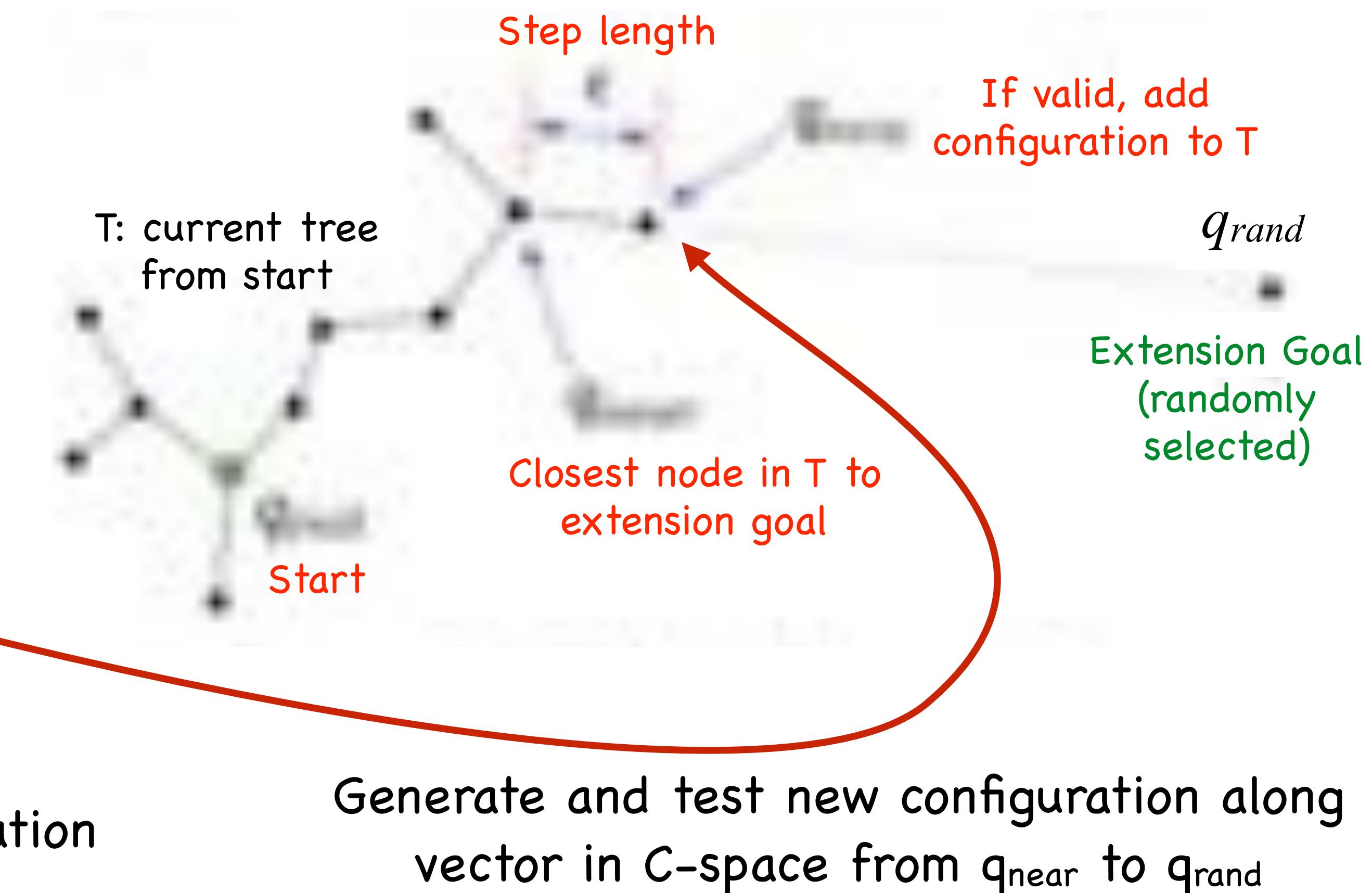
RRT Algorithm

Extend graph towards a random configuration and repeat

```
BUILD-RRT( $q_{start}$ )
1  $T \leftarrow \text{init}(q_{start})$ ;
2 for  $k = 1$  to  $K$  do
3    $q_{rand} \leftarrow \text{RANDOM-CONFIG}()$ ;
4   EXTEND( $T, q_{rand}$ );
5 return  $T$ 
```

```
EXTEND( $T, q$ )
1  $q_{near} \leftarrow \text{NEAREST-NEIGHBOR}(q, T)$ ;
2 if NEW-CONFIG( $q_{near}, q_{rand}$ ) then
3    $T$ .add vertex( $q_{near}$ );
4    $T$ .add edge( $q_{near}, q_{rand}$ );
5   if  $q_{rand} = q$  then
6     Return Reached;
7   else
8     Return Admissible;
9 return Trapped.
```

Extend graph towards a random configuration



RRT* Algorithm

RRT*

Algorithm 6: RRT*

```
1  $V \leftarrow \{x_{\text{init}}\}$ ;  $E \leftarrow \emptyset$ ;
2 for  $i = 1, \dots, n$  do
3    $x_{\text{rand}} \leftarrow \text{SampleFree}_c$ ;
4    $x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{rand}})$ ;
5    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$ ;
6   if  $\text{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$  then
7      $N_{\text{near}} \leftarrow \text{Near}(G = (V, E), x_{\text{near}}, \min\{\text{max}(\log(\text{card}(V)) / \text{card}(V))^{1/d}, \eta\})$ ;
8      $V \leftarrow V \cup \{x_{\text{new}}\}$ ;
9      $x_{\text{near}} \leftarrow x_{\text{nearest}}$ ;  $c_{\text{near}} \leftarrow \text{Cost}(x_{\text{nearest}}) + c(\text{Line}(x_{\text{nearest}}, x_{\text{new}}))$ ;
10    foreach  $x_{\text{near}} \in N_{\text{near}}$  do // Connect along a minimum-cost path
11      if  $\text{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{near}}$  then
12         $x_{\text{near}} \leftarrow x_{\text{new}}$ ;  $c_{\text{near}} \leftarrow \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}}))$ 
13     $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}})\}$ ;
14    foreach  $x_{\text{near}} \in N_{\text{near}}$  do // Rewire the tree
15      if  $\text{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}})) < \text{Cost}(x_{\text{near}})$ 
16      then  $x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}})$ ;
17       $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{near}}, x_{\text{new}})\}$ 
18
19 return  $G = (V, E)$ ;
```

RRT*

Algorithm 6: RRT*

```

1  $V \leftarrow \{x_{\text{init}}\}$ ;  $E \leftarrow \emptyset$ ;
2 for  $i = 1, \dots, n$  do
3    $x_{\text{rand}} \leftarrow \text{SampleFree}(\mathcal{X})$ ;
4    $x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{rand}})$ ;
5    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$  ;
6   if  $\text{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$  then
7      $N_{\text{near}} \leftarrow \text{Neax}(G = (V, E), x_{\text{new}}, \min\{\text{max}\{\log(\text{card}(V)) / \text{card}(V)\}^{1/d}, \eta\})$  ;
8      $V \leftarrow V \cup \{x_{\text{new}}\}$ ;
9      $x_{\text{near}} \leftarrow x_{\text{nearest}}$ ;  $c_{\text{near}} \leftarrow \text{Cost}(x_{\text{nearest}}) + c(\text{Line}(x_{\text{nearest}}, x_{\text{new}}))$ ;
10    foreach  $x_{\text{near}} \in N_{\text{near}}$  do // Connect along a minimum-cost path
11      if  $\text{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{near}}$  then
12         $x_{\text{near}} \leftarrow x_{\text{new}}$ ;  $c_{\text{near}} \leftarrow \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}}))$ 
13     $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}})\}$ ;
14    foreach  $x_{\text{near}} \in N_{\text{near}}$  do // Rewire the tree
15      if  $\text{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}})) < \text{Cost}(x_{\text{near}})$ 
16      then  $x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}})$ ;
17       $E \leftarrow (E \backslash \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{near}}, x_{\text{new}})\}$ 
18
19 return  $G = (V, E)$ ;

```

FIND x_{new}

FIND neighbors to x_{new} in G

ADD x_{new} to G

FIND edge to x_{new} from
neighbors with least cost
ADD that to G

REWIRE the edges in the
neighborhood if any least
cost path exists from the
root to the neighbors via x_{new}

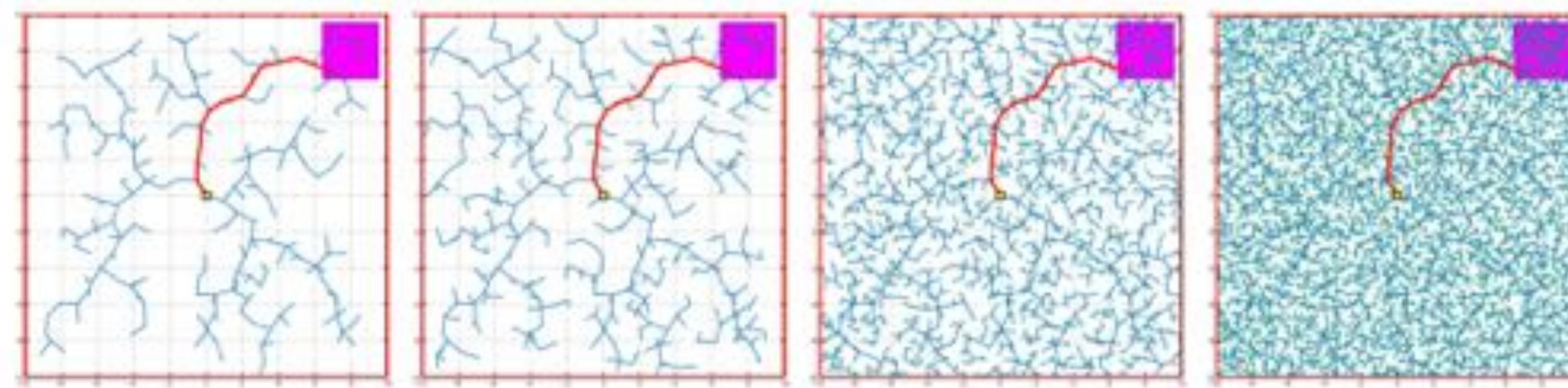
RRT*

- Asymptotically optimal
- Main idea:
 - Swap new point in as parent for nearby vertices who can be reached along shorter path through new point than through their original (current) parent

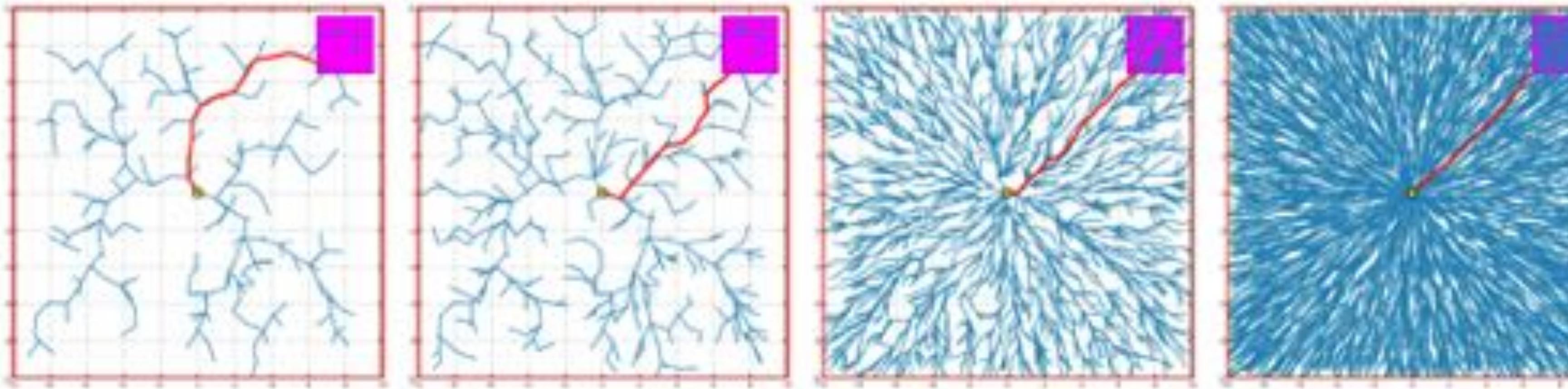
Demonstration - <https://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT/>

RRT*

RRT



RRT*

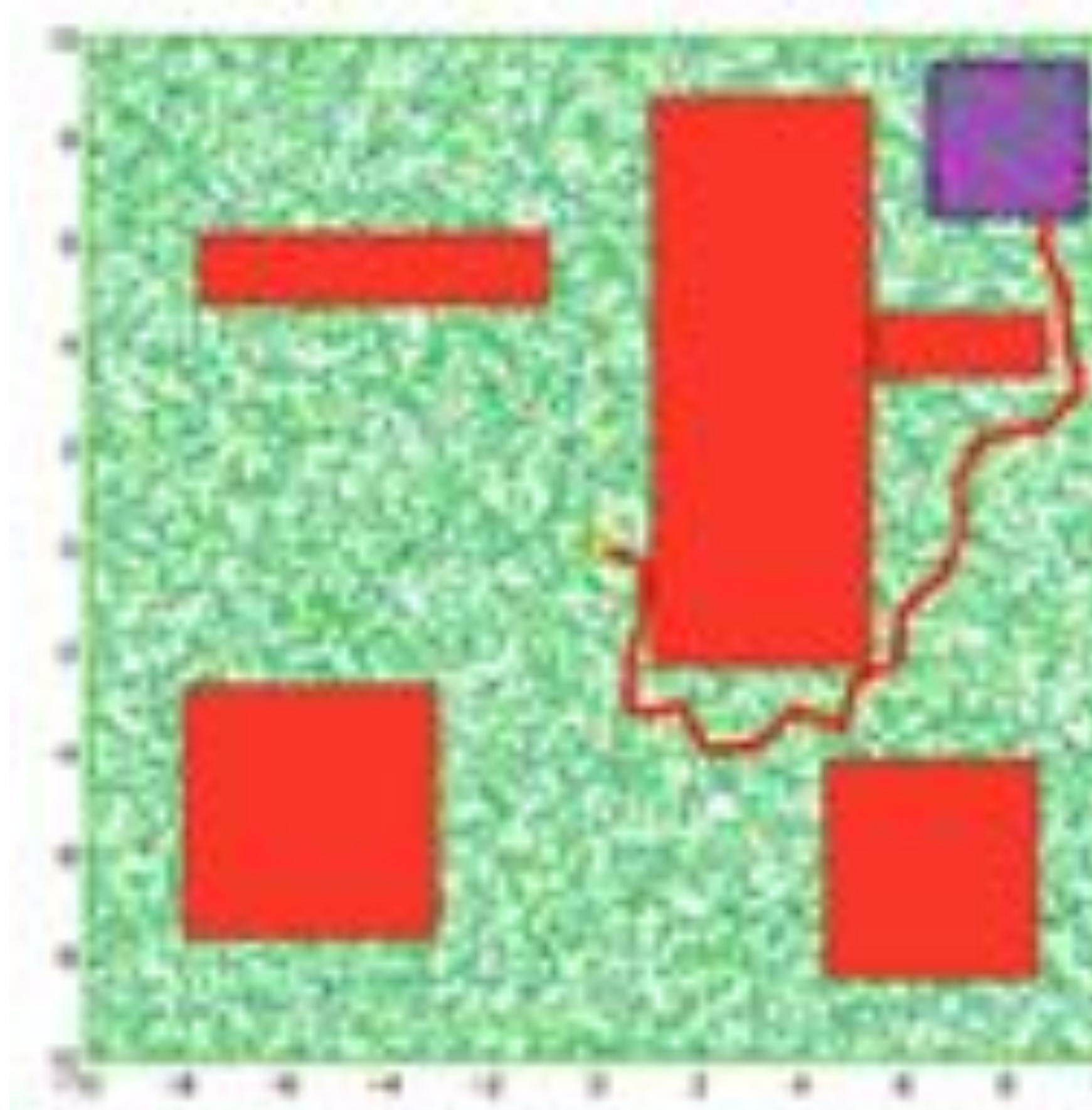


Source: Karaman and Frazzoli

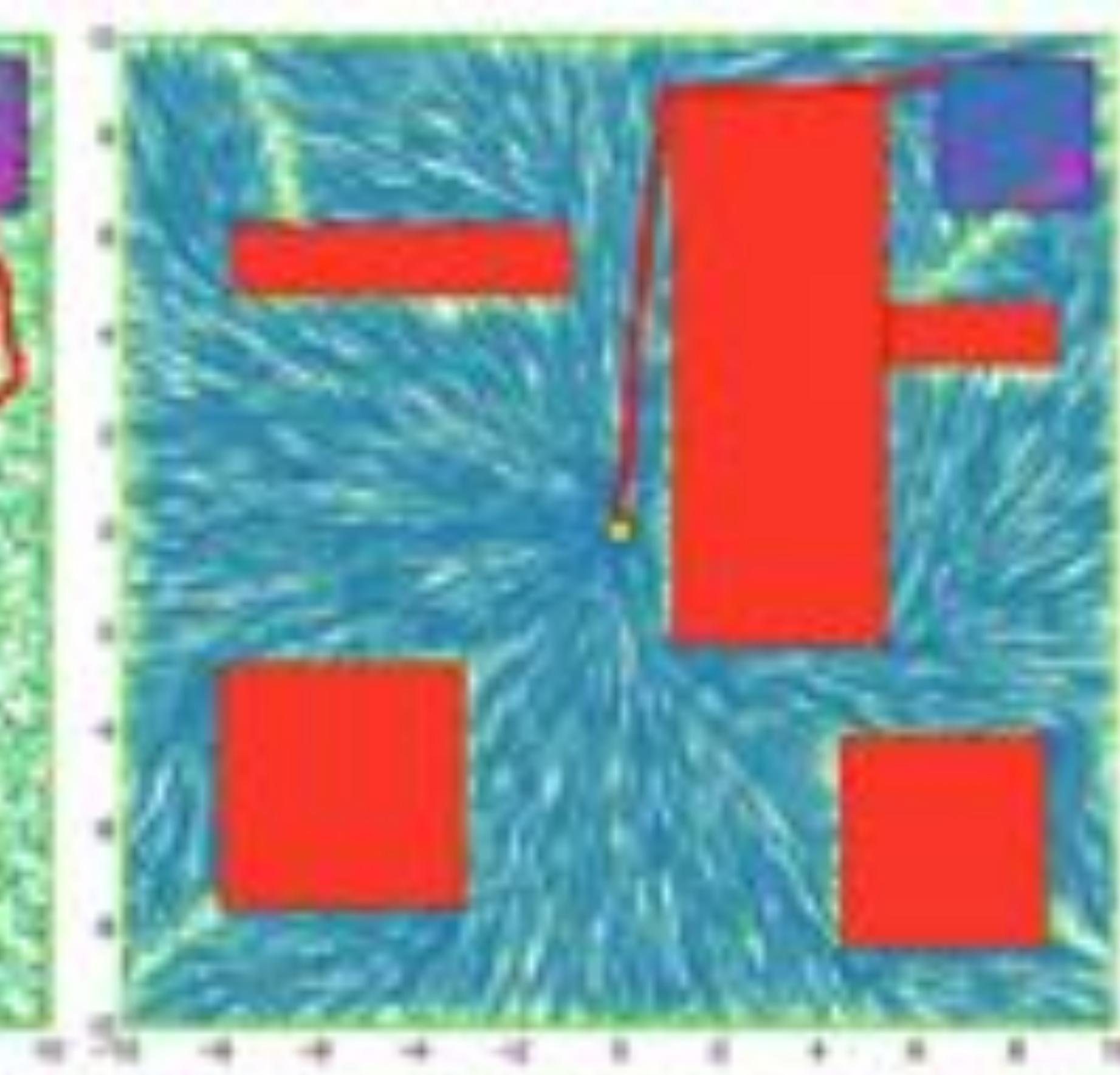


RRT*

RRT



RRT*



Source: Karaman and Frazzoli

Smoothing

Randomized motion planners tend to find not so great paths for execution: very jagged, often much longer than necessary.

- In practice: do smoothing before using the path
 - Shortcutting:
 - along the found path, pick two vertices x_{t_1}, x_{t_2} and try to connect them directly (skipping over all intermediate vertices)
 - Nonlinear optimization for optimal control
 - Allows to specify an objective function that includes smoothness in state, control, small control inputs, etc.

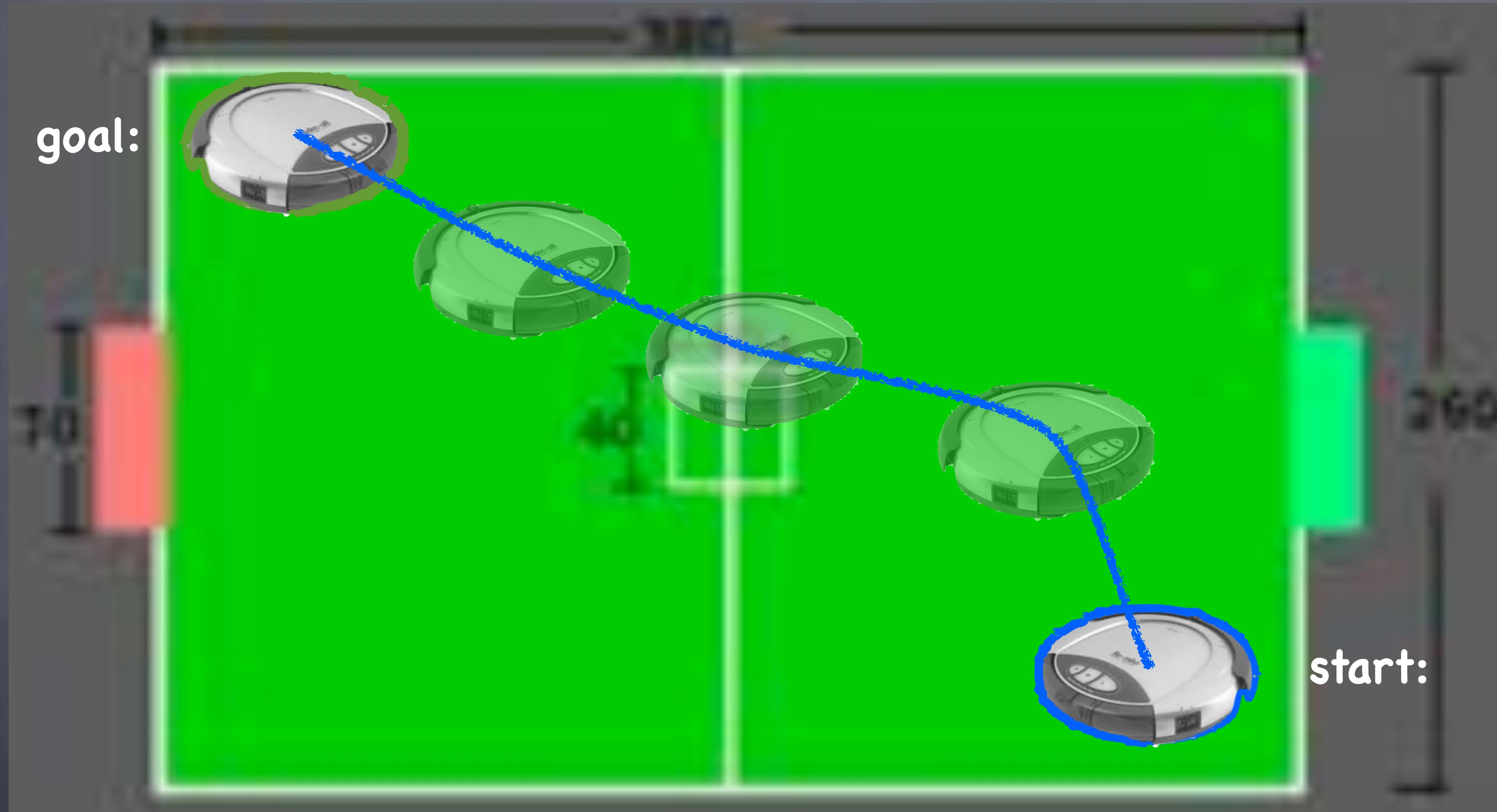


Approaches to motion planning

- 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)

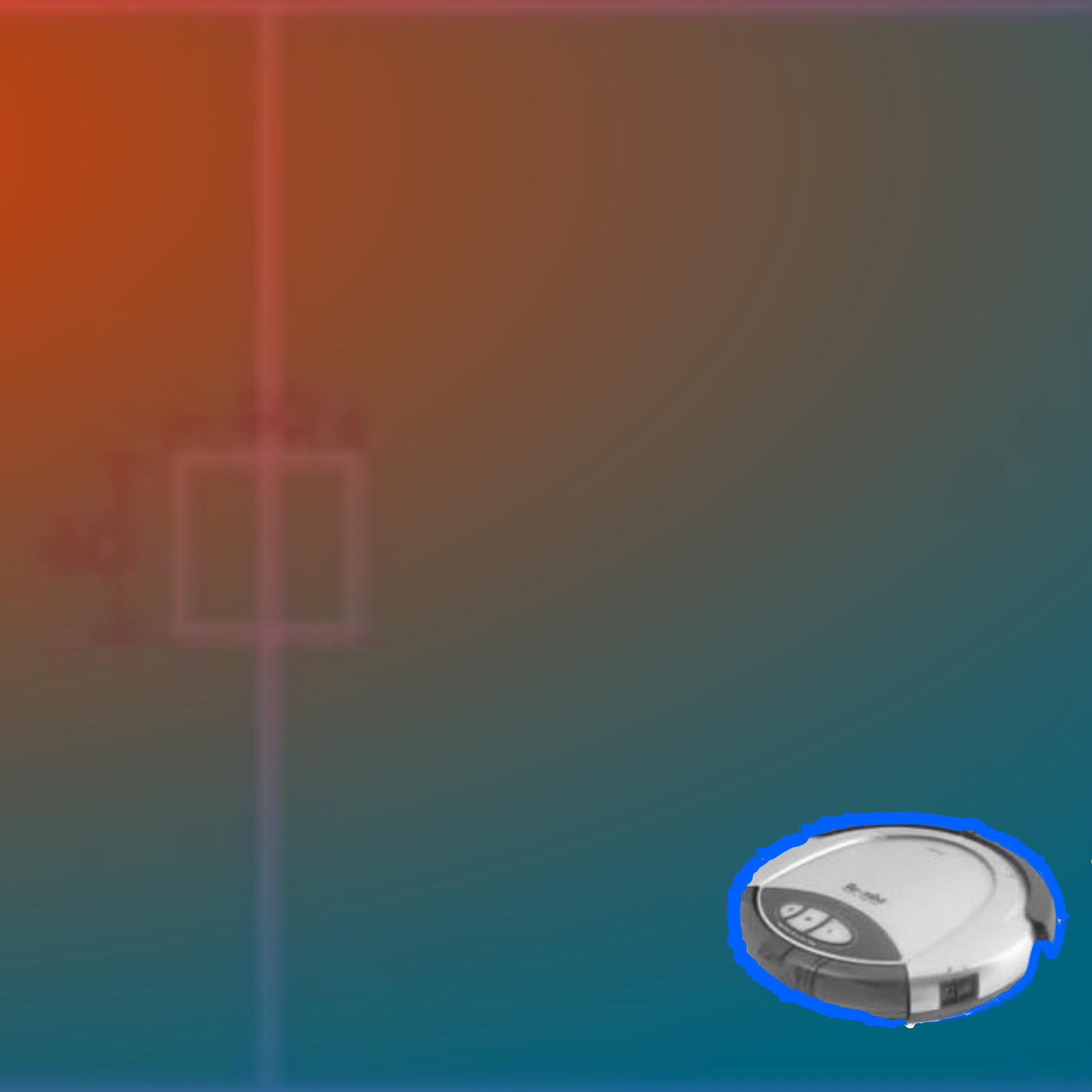


Potential field

(like a game of “warmer-colder”)

goal:

volcanic



start:
cold



Potential field

(like a game of “warmer-colder”)

goal:

volcanic



colder

start:
cold



Potential field

(like a game of “warmer-colder”)

goal:

volcanic



a little
warmer



colder

start:
cold

Potential field

(like a game of “warmer-colder”)

goal:
volcanic

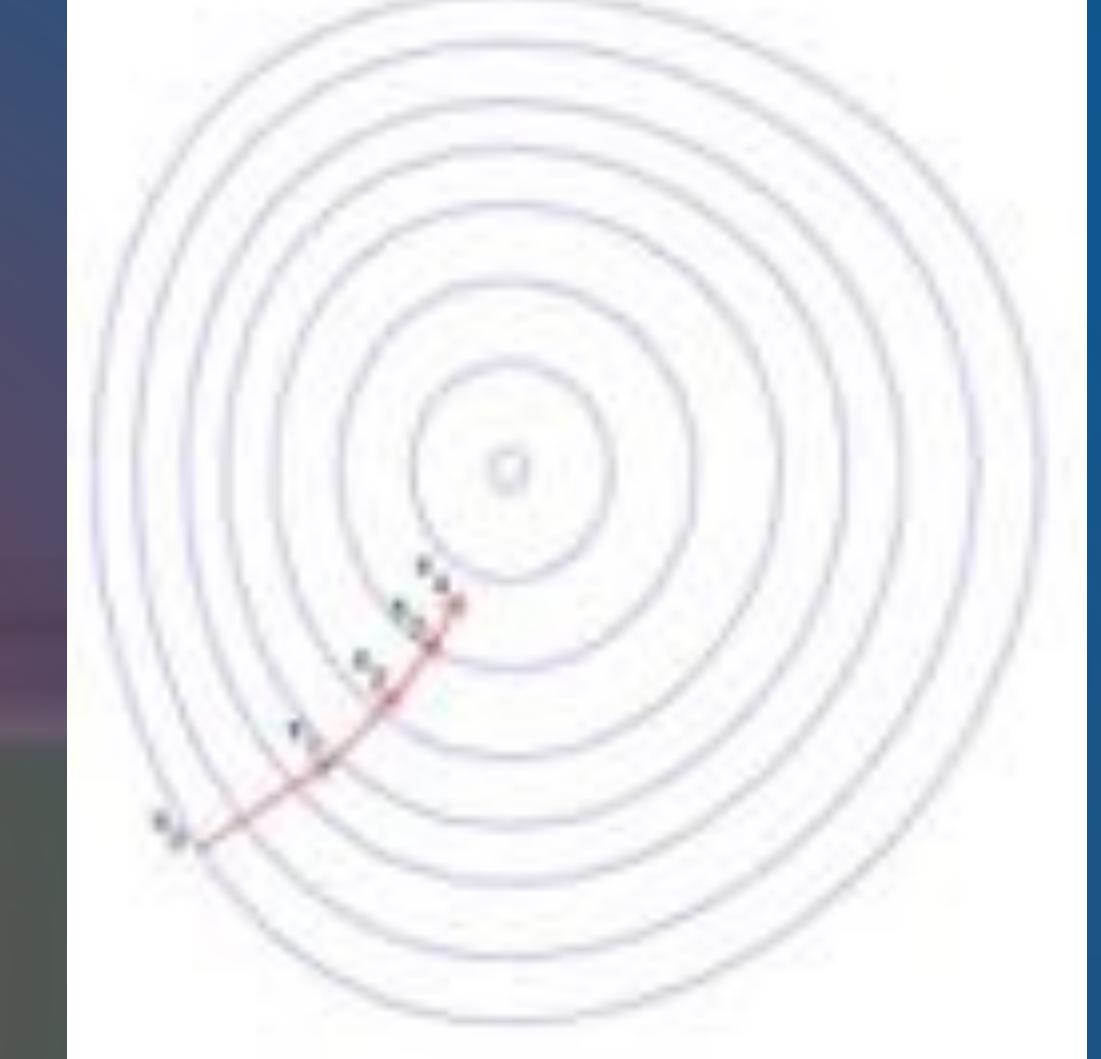


a little
warmer



colder

start:
cold



Gradient descent:
Energy potential
converges at goal

Potential field

(like a game of “warmer-colder”)

goal:
volcanic



on fire



heating up



a little
warmer



start:
cold



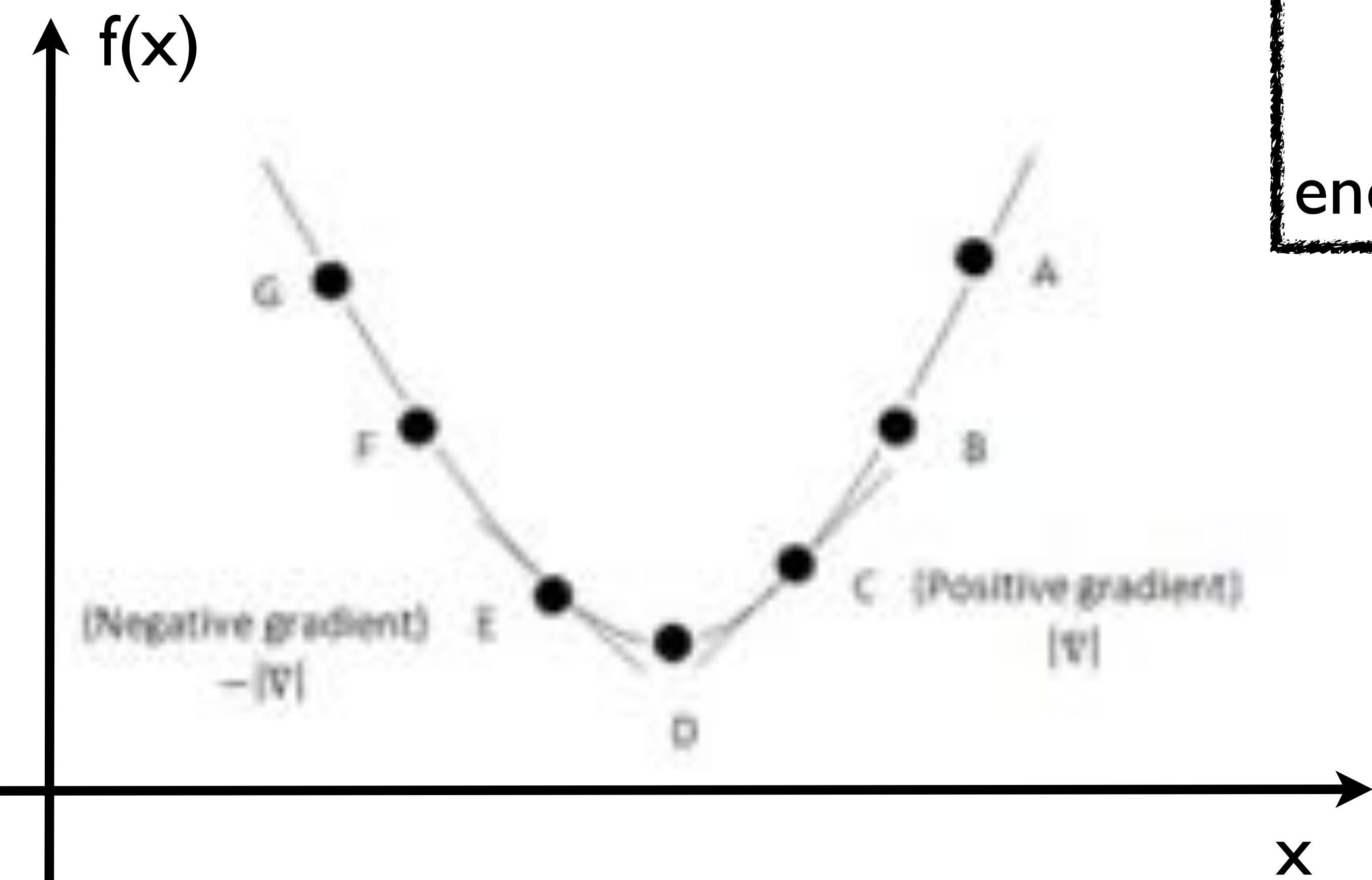
How do we define a potential field?

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 Algorithm:

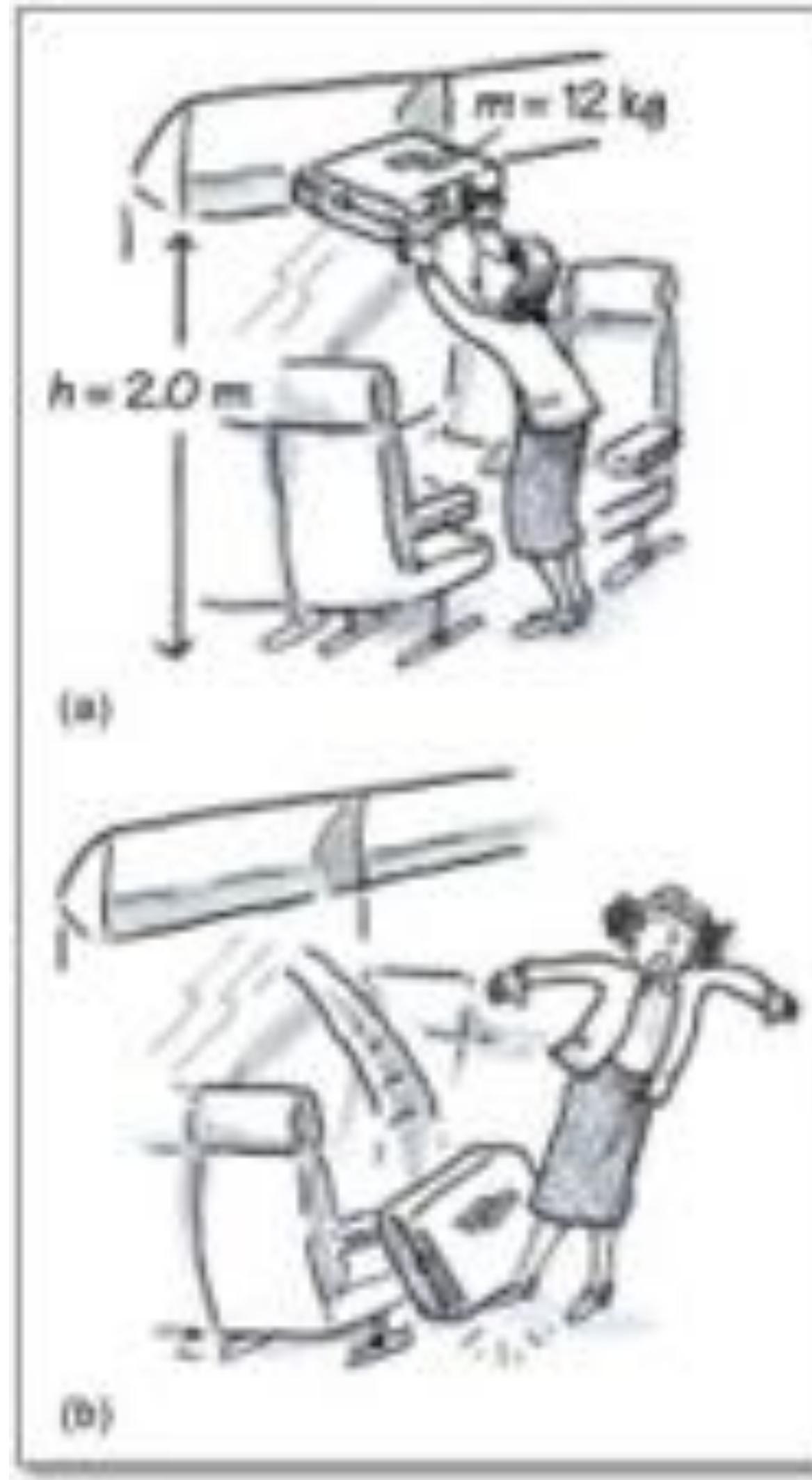
```
qpath[0] ← qstart
i ← 0
while (|| ∇U(q[i]) || > ε)
    qpath[i+1] ← qpath[i] - a ∇U(qpath[i])
    i ← i+1
end
```



Derivative assumed to be direction
of steepest ascent away from goal

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

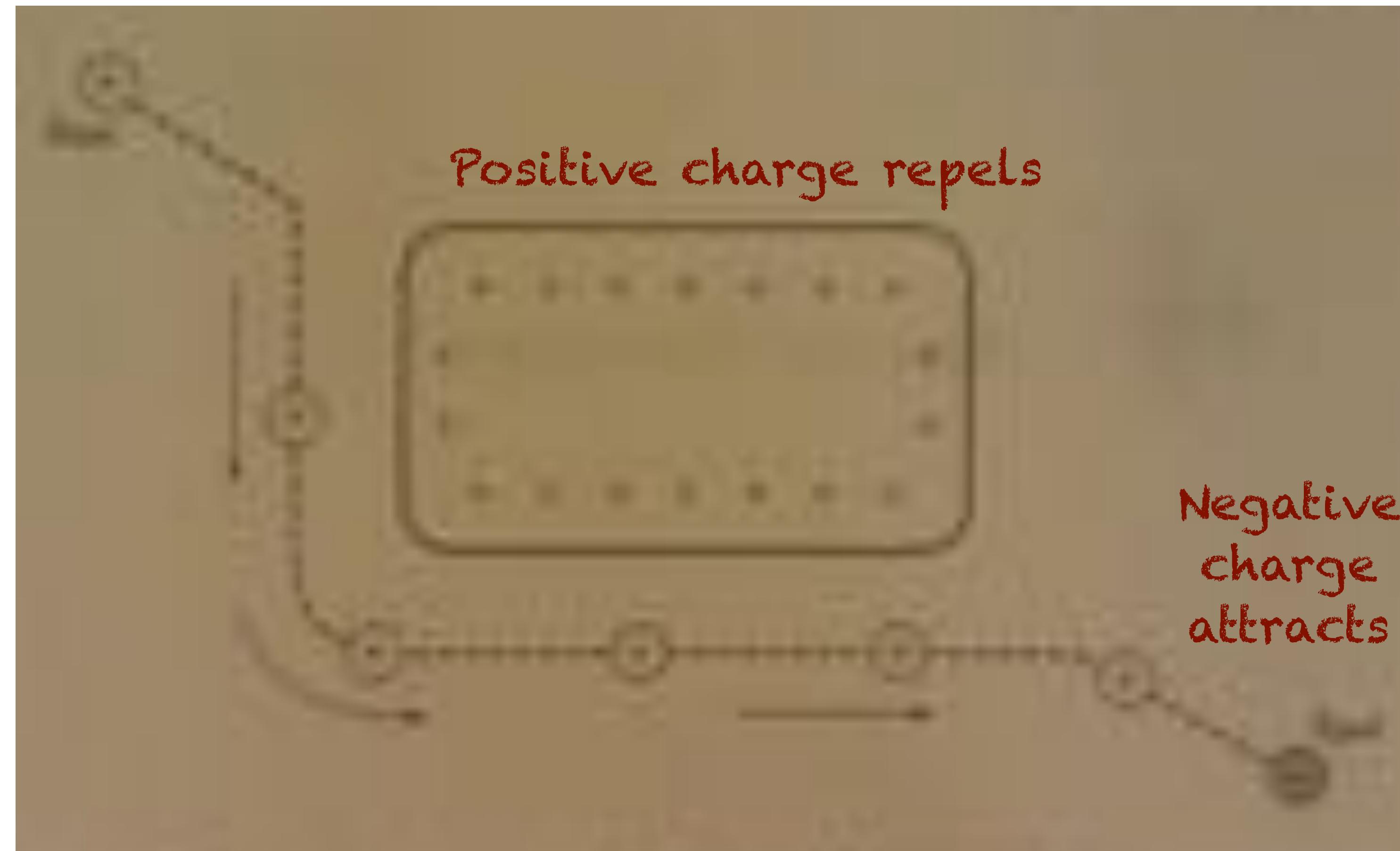
Potential Energy



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

Charged Particle Example

Positively charged particle follows potential energy to goal



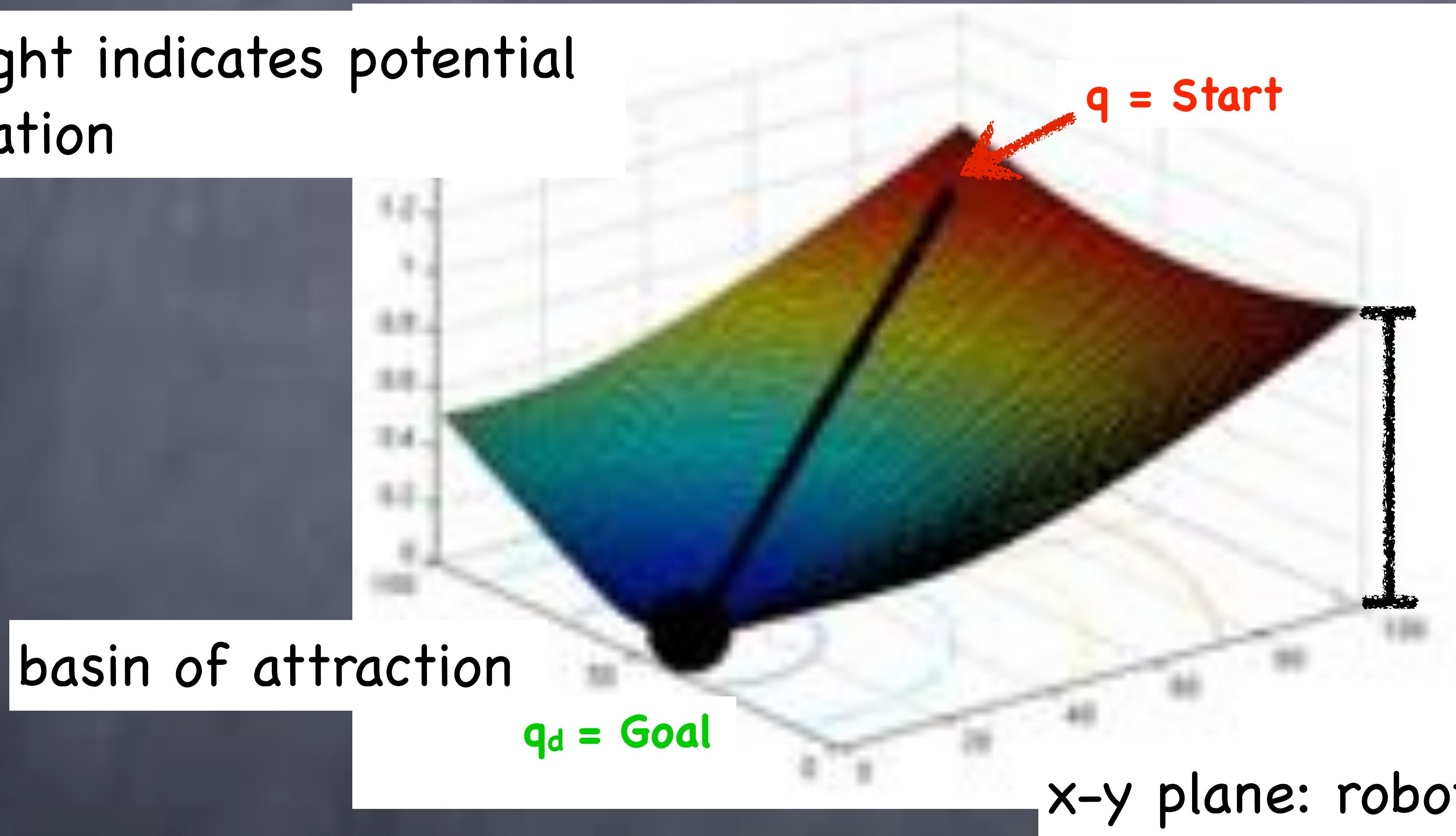
Convergent Potentials

let's call these "attractor landscapes"

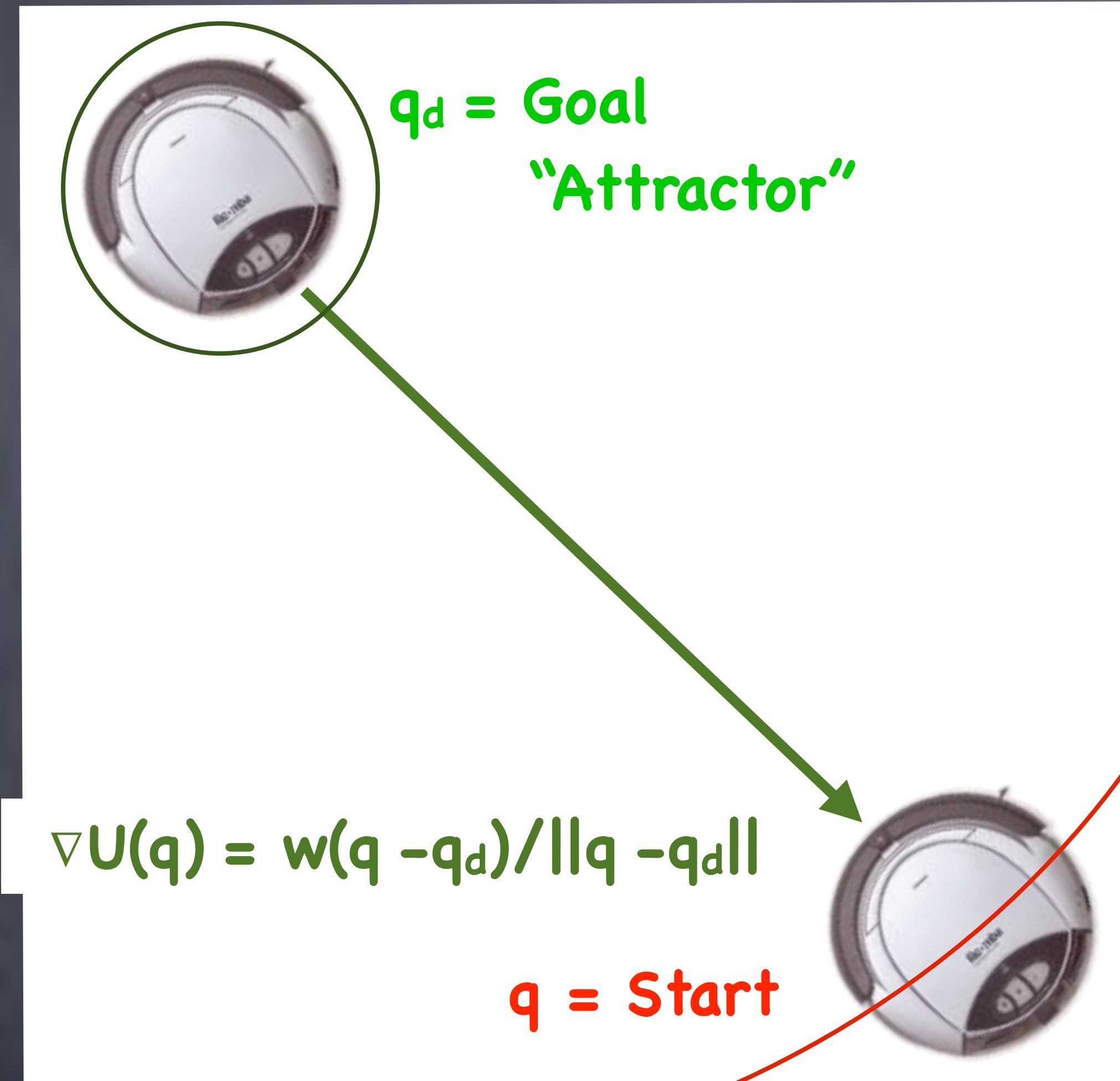


2D potential navigation

z : height indicates potential at location



top view



"Cone" Attractor

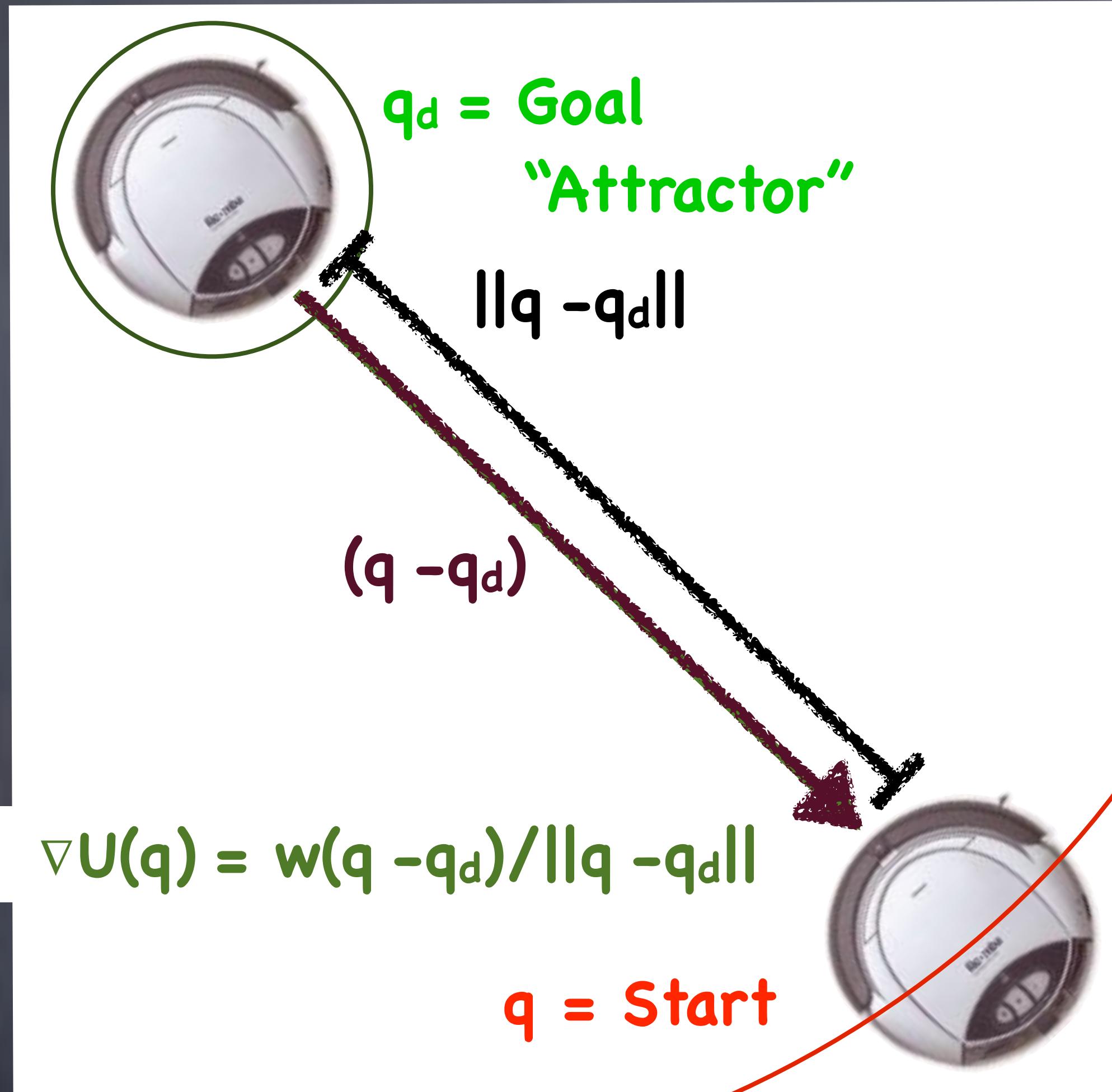
Start

Goal

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

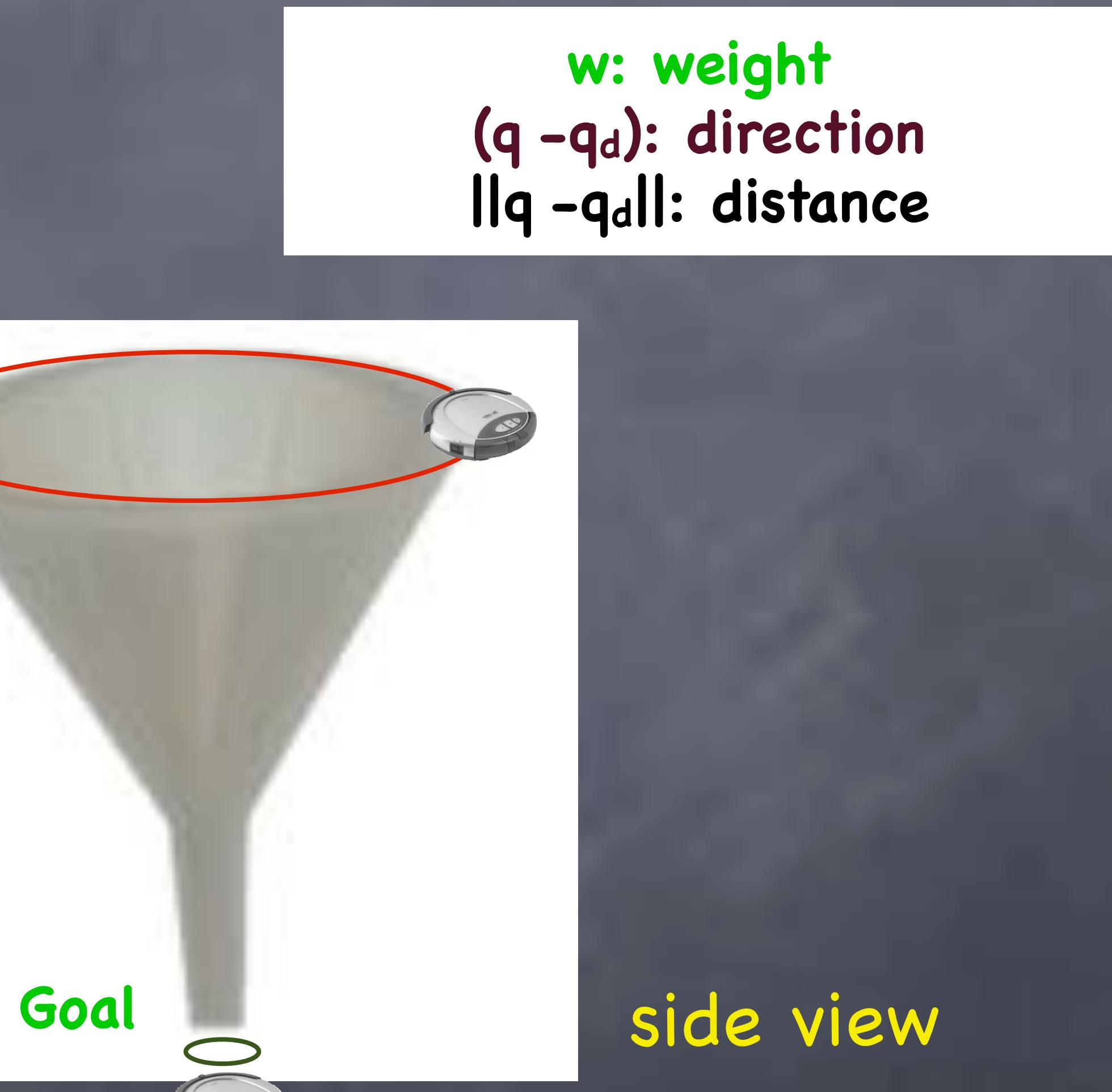
side view

top view



"Cone" Attractor

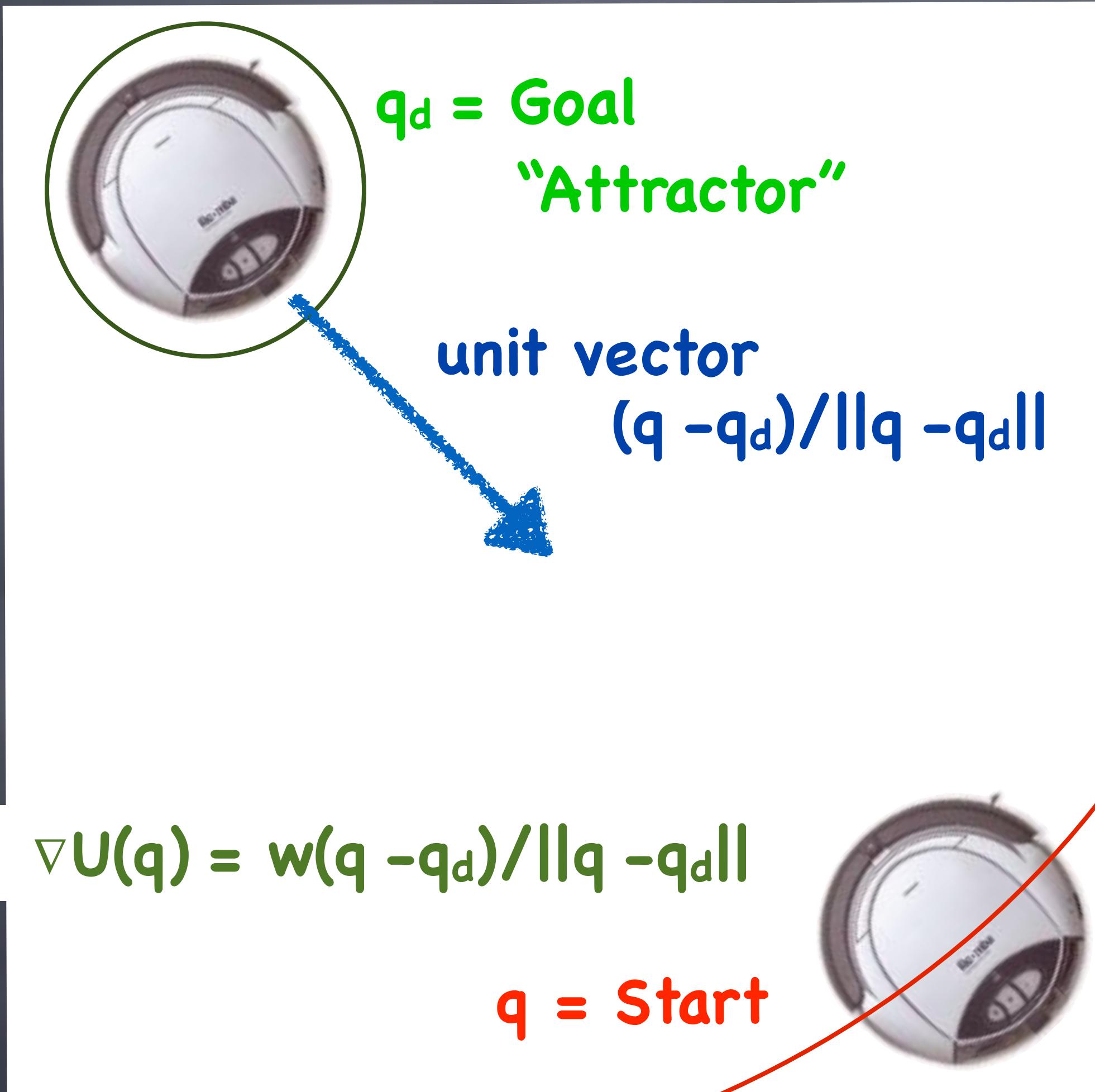
Start



side view



top view



"Cone" Attractor

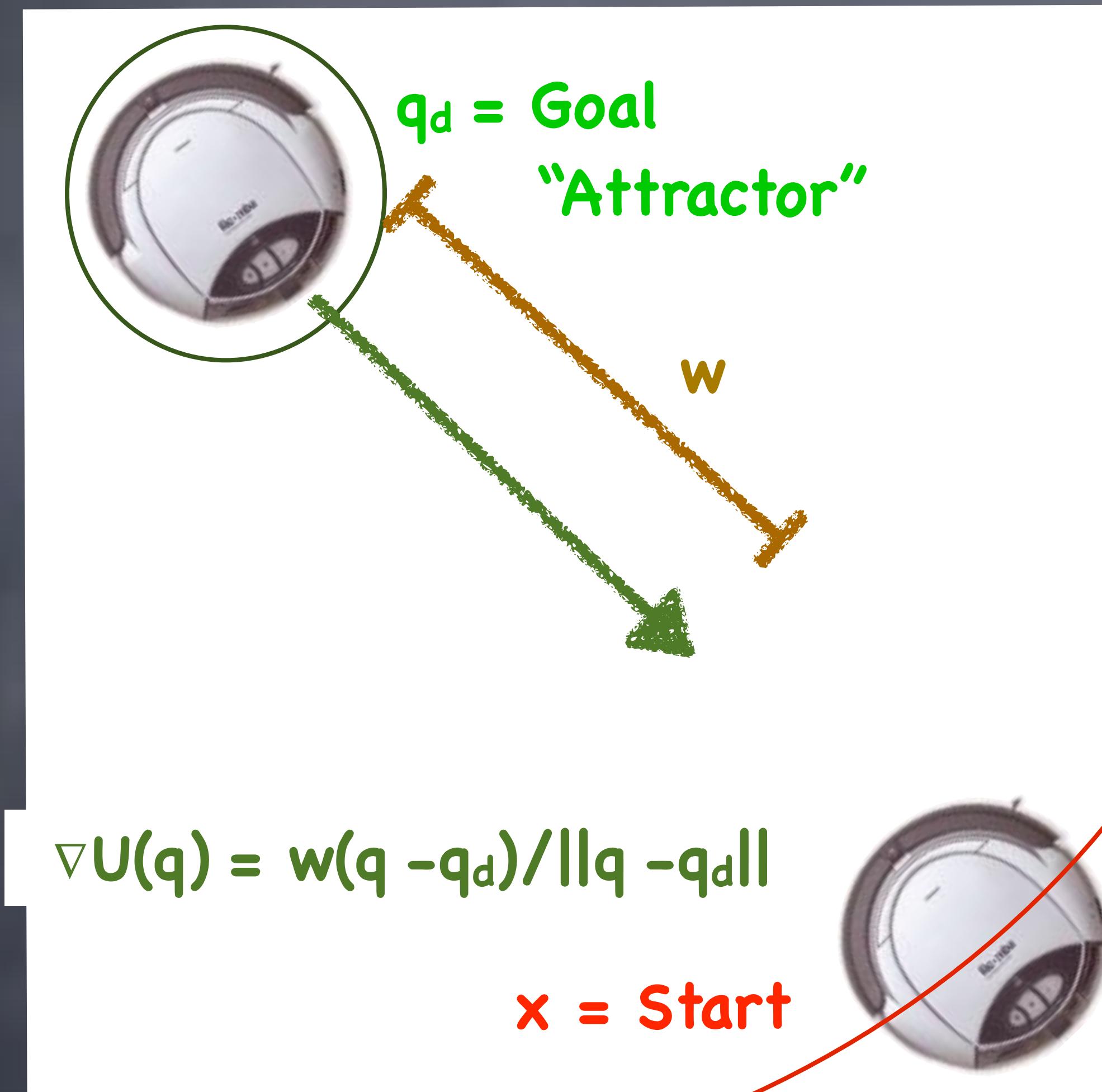
Start

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



side view

top view



"Cone" Attractor

Start

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



side view

Can we modulate the
range of a potential field?

top view



$q_d = \text{Goal}$

$q = \text{Start}$

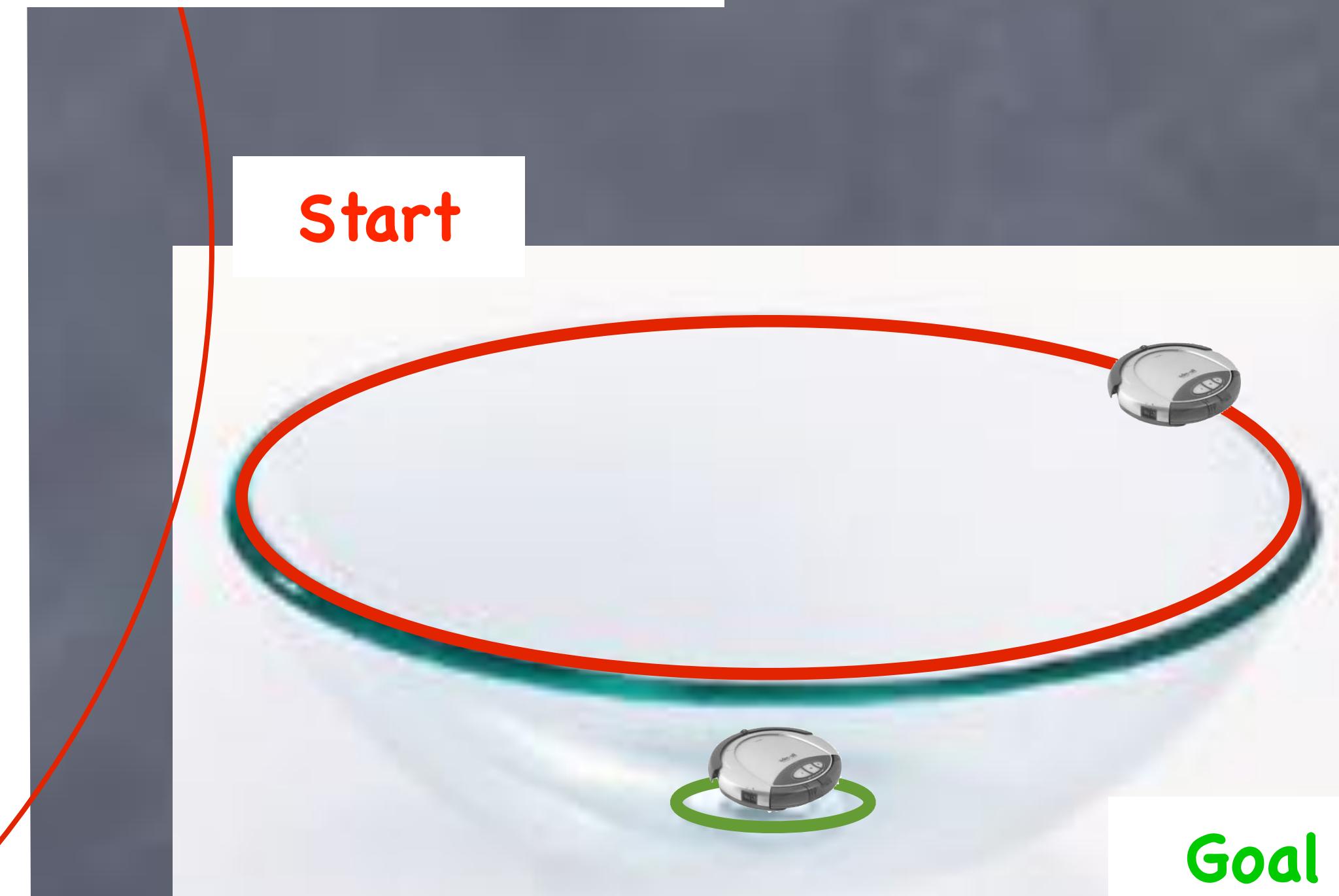
“Bowl” Attractor

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

Start

Goal

side view



$\exp(-d/w)$

weights the influence of a potential

$\exp(-d/.1)$

$\exp(-d/0.5)$

$\exp(-d/1)$

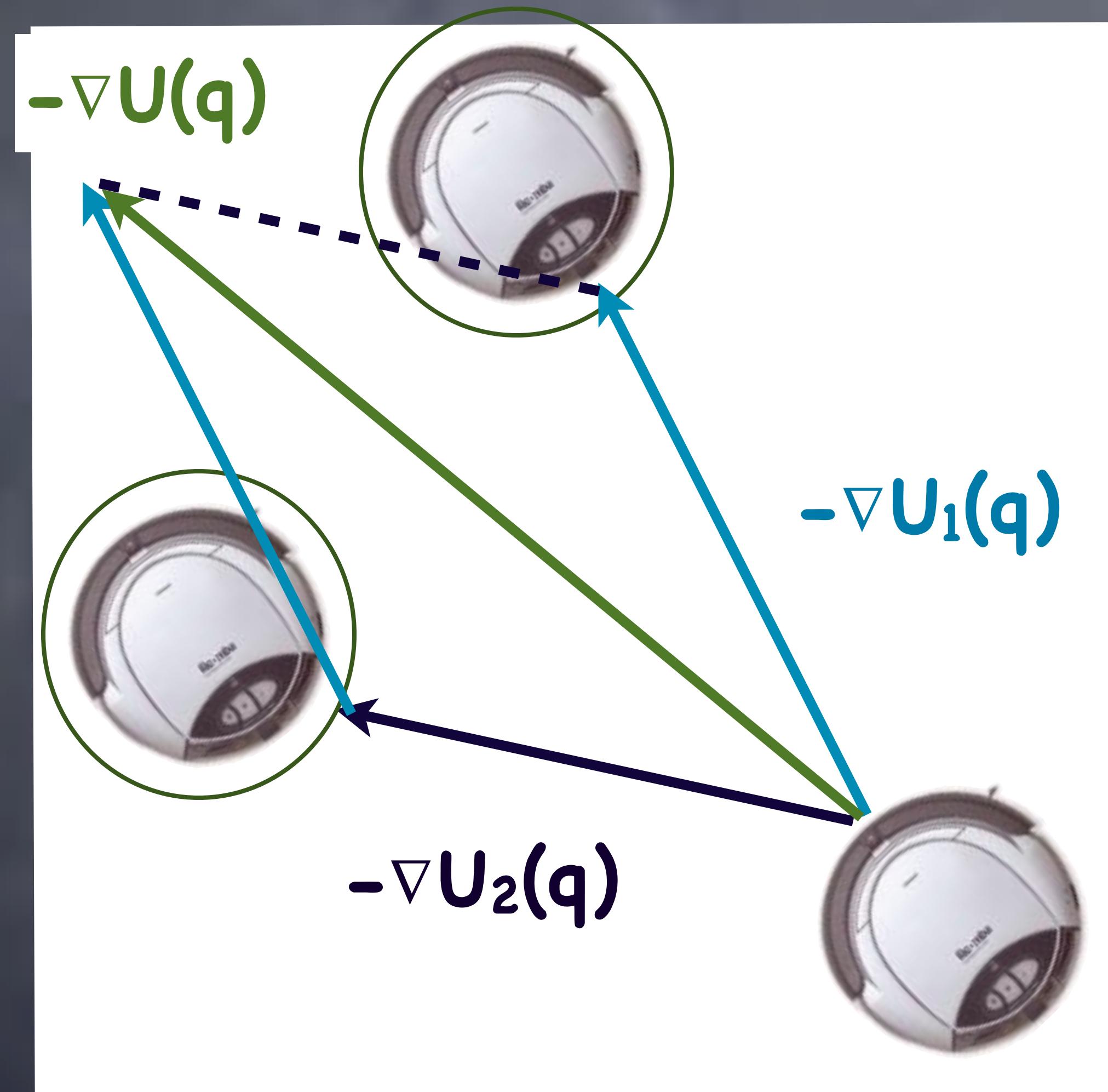
$\exp(-d/2)$

$\exp(-d/10)$

x-axis is d = distance

Can we combine
multiple potentials?

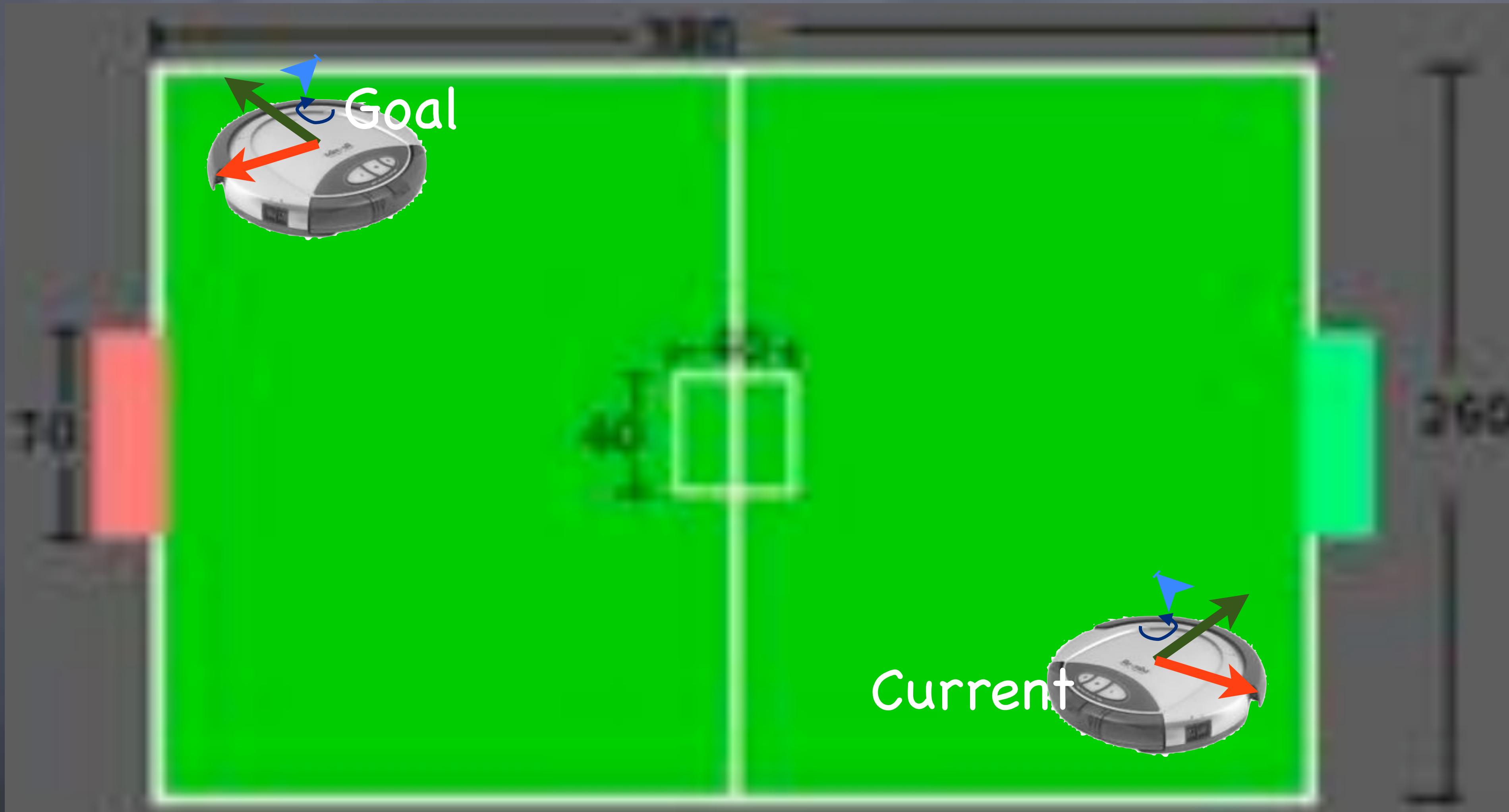
Multiple potentials



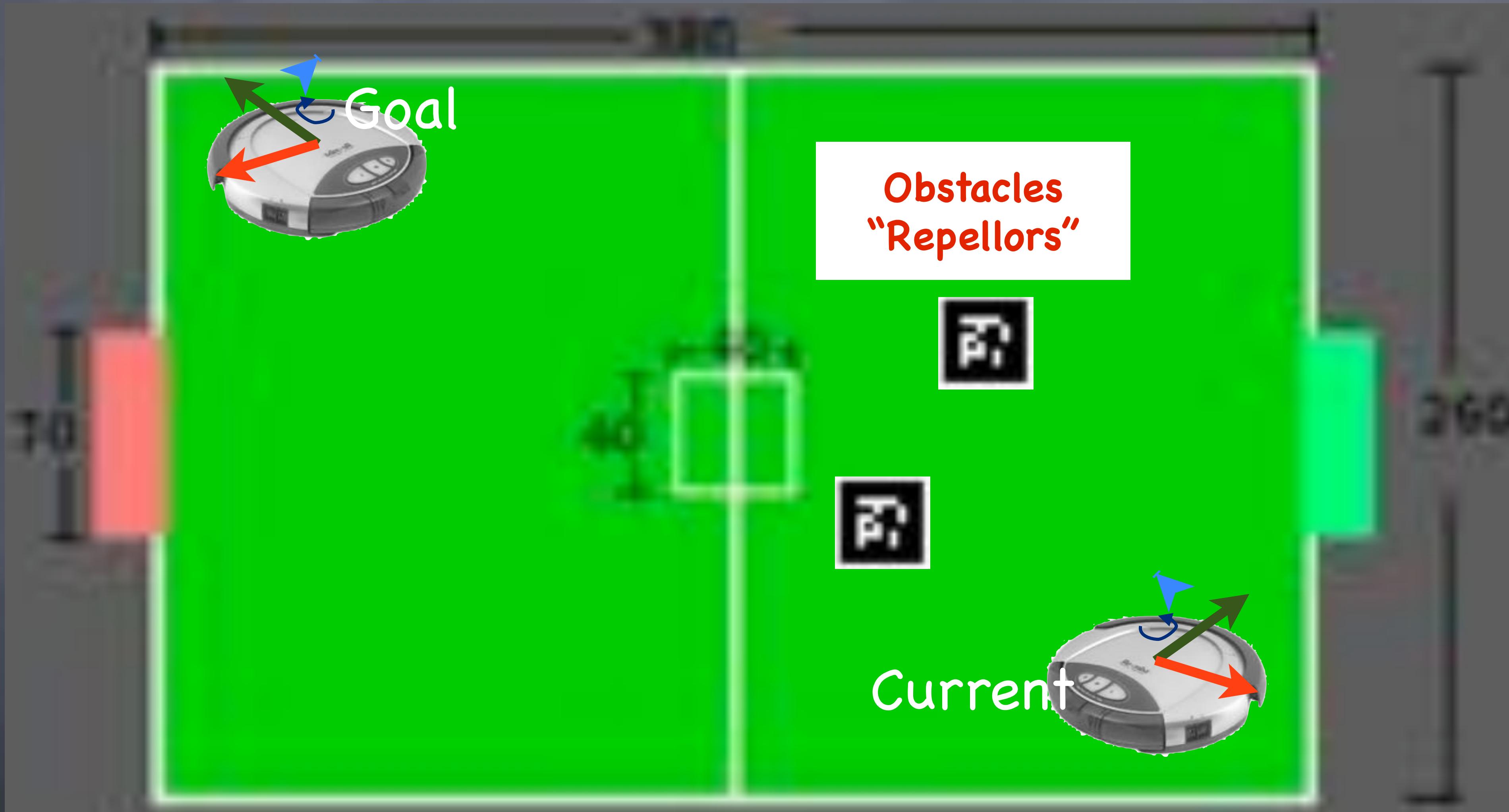
- Output of potential field is a vector
- Combine multiple potentials through vector summation

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

describe performance for this case

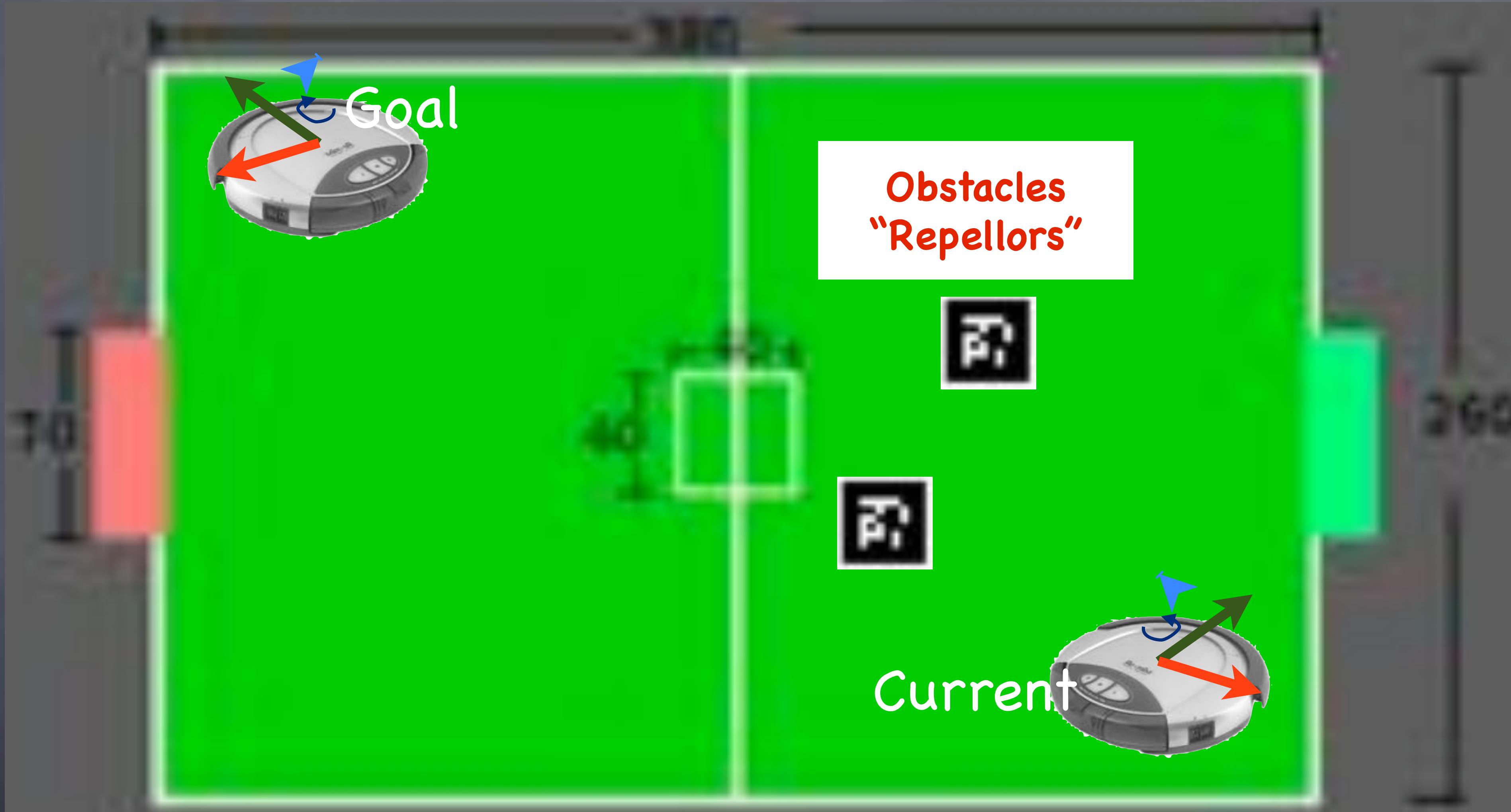


describe performance for this case



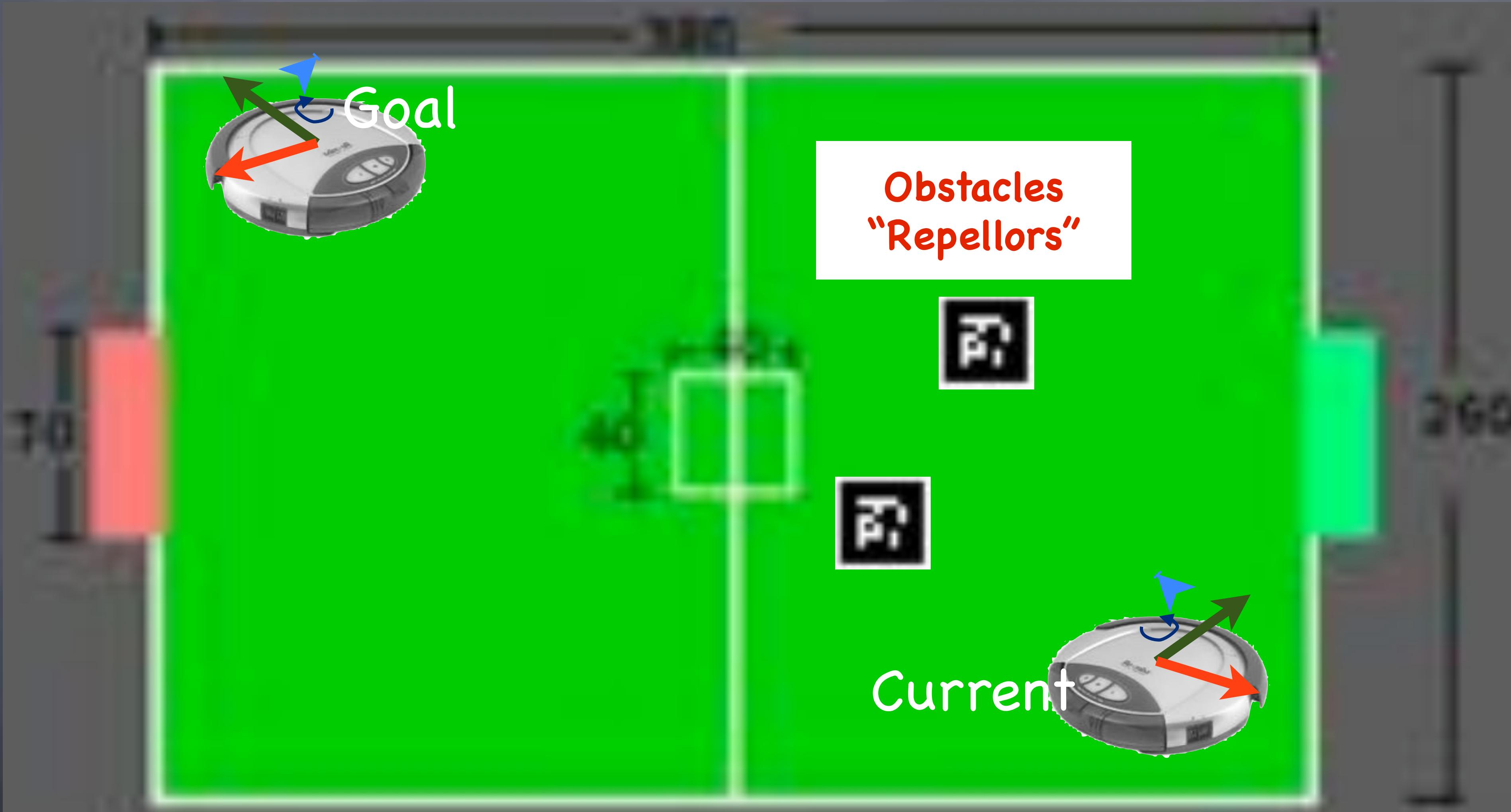
describe performance for this case

how do we deal with repellors?



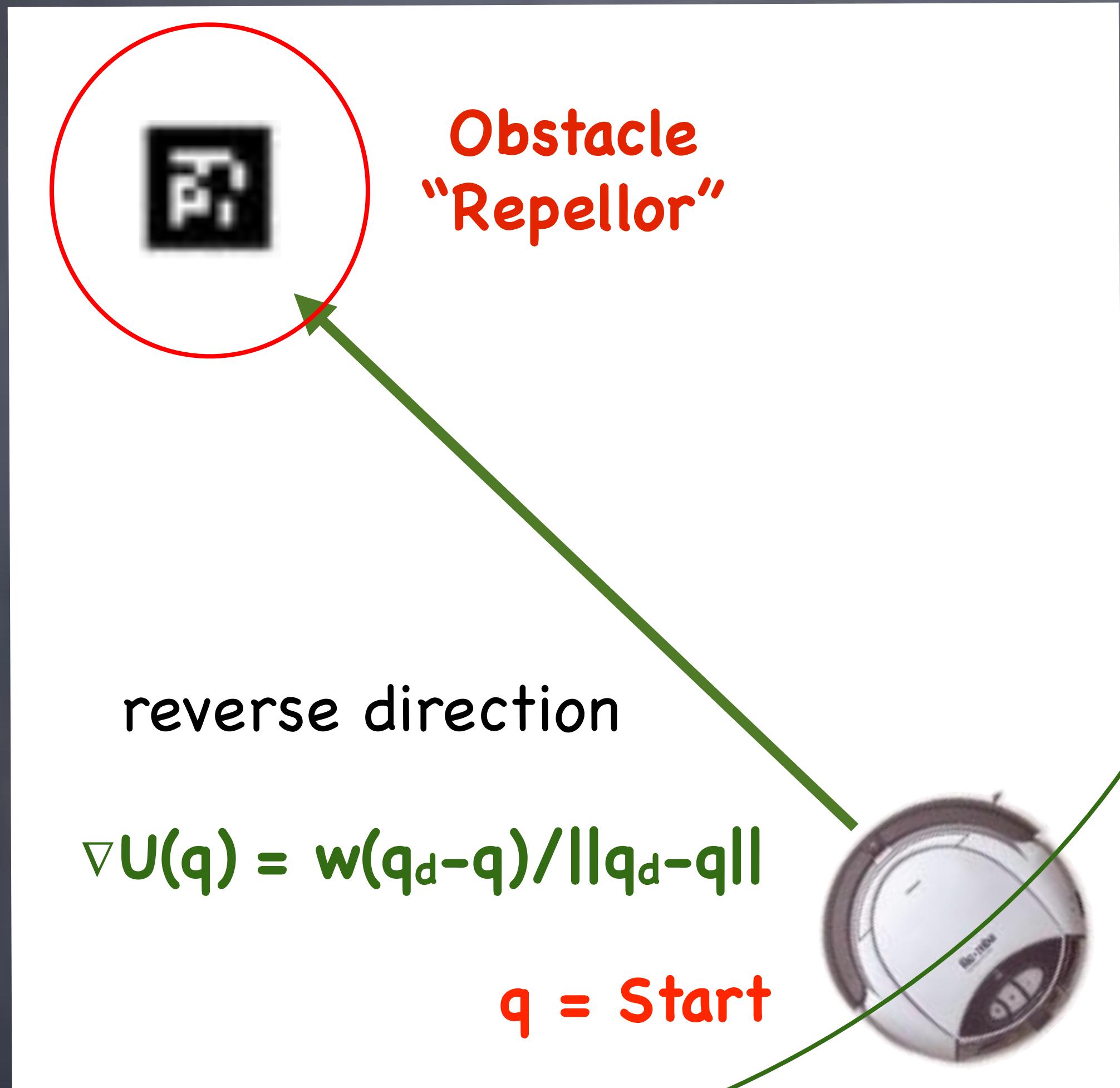
add sum of repulsive potentials

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

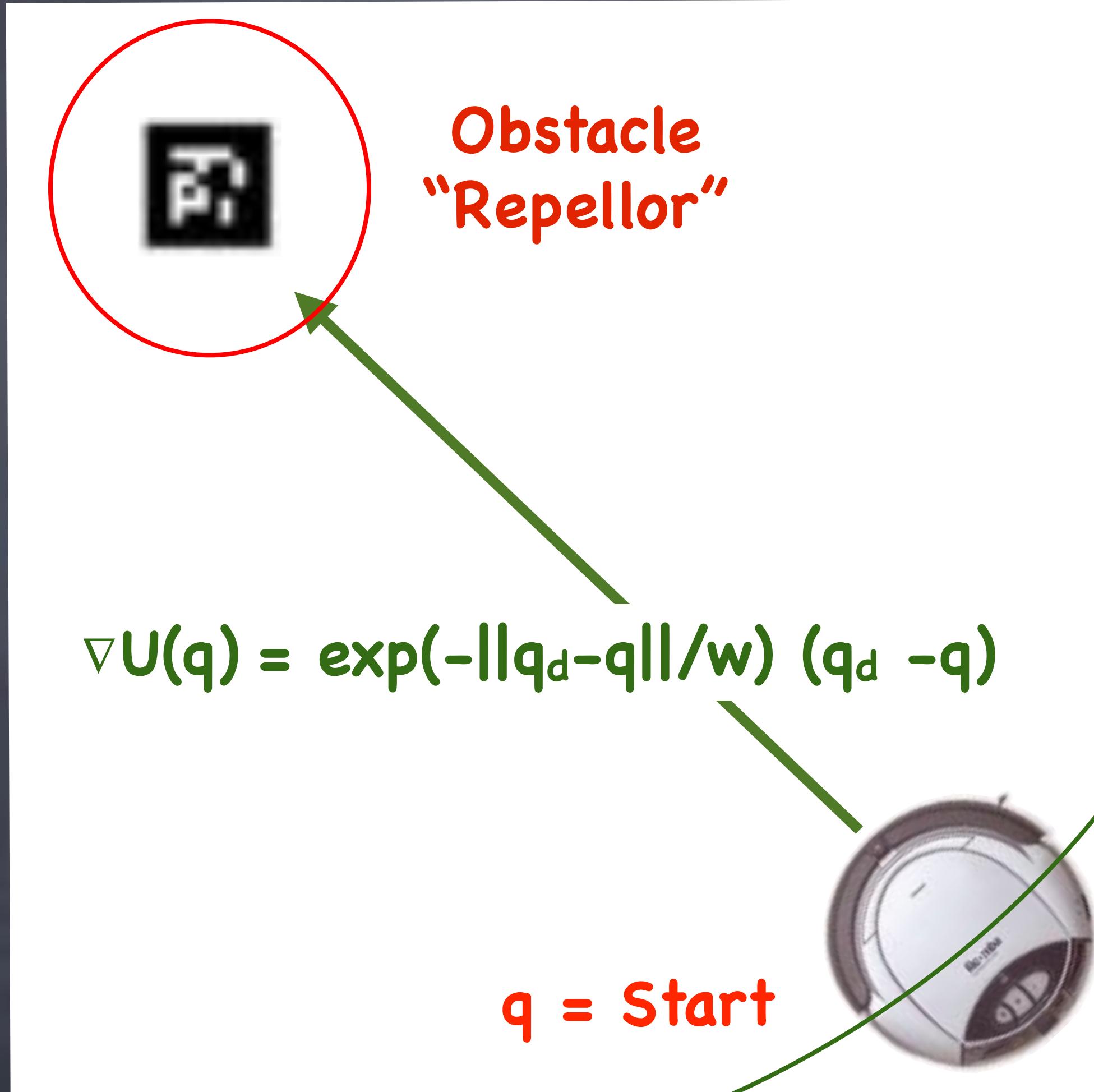


“Cone” Repellor

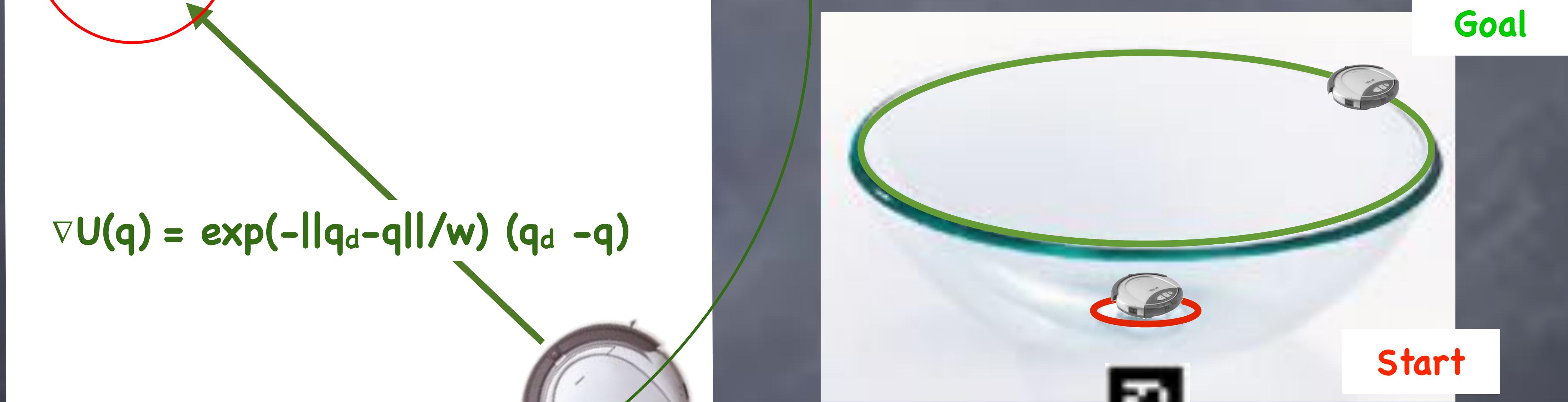
potential problems?



"Bowl" Repellor



top view



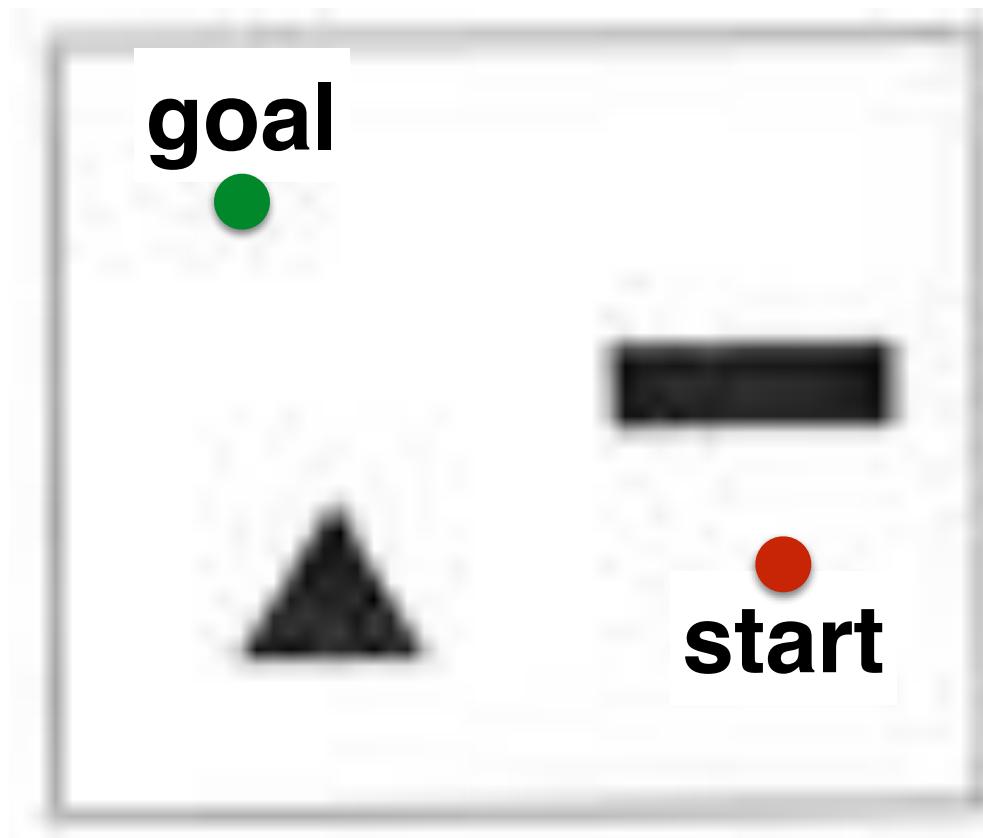
side view

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



2 Obstacle example

configuration space



attractor field



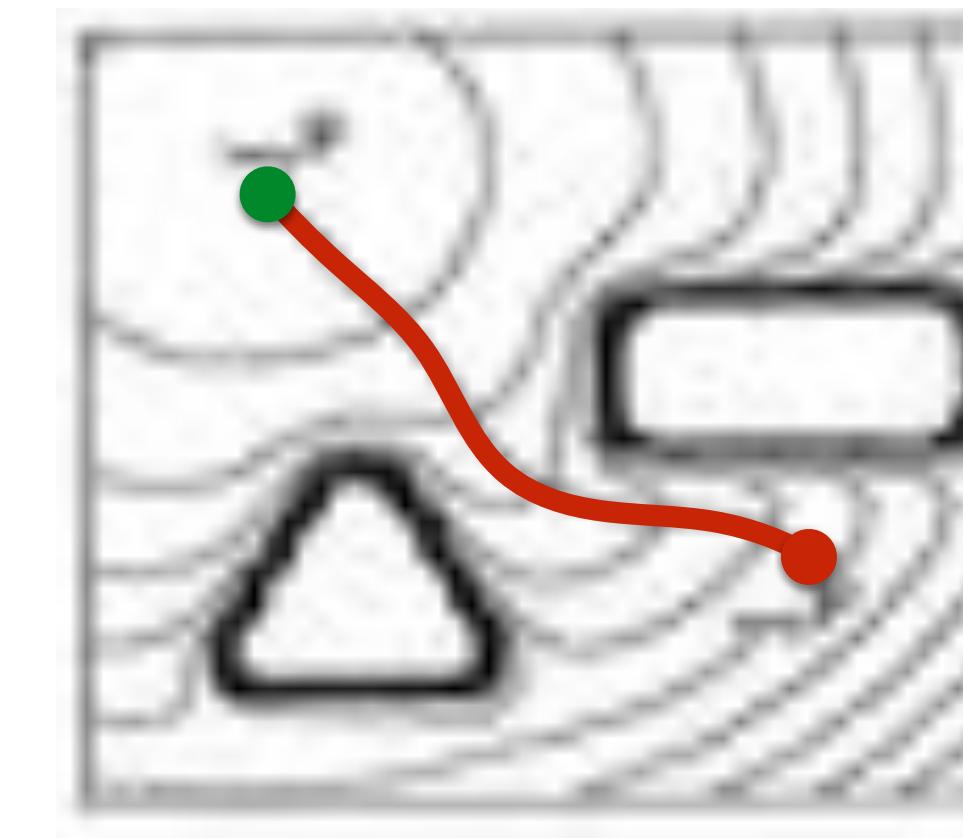
+



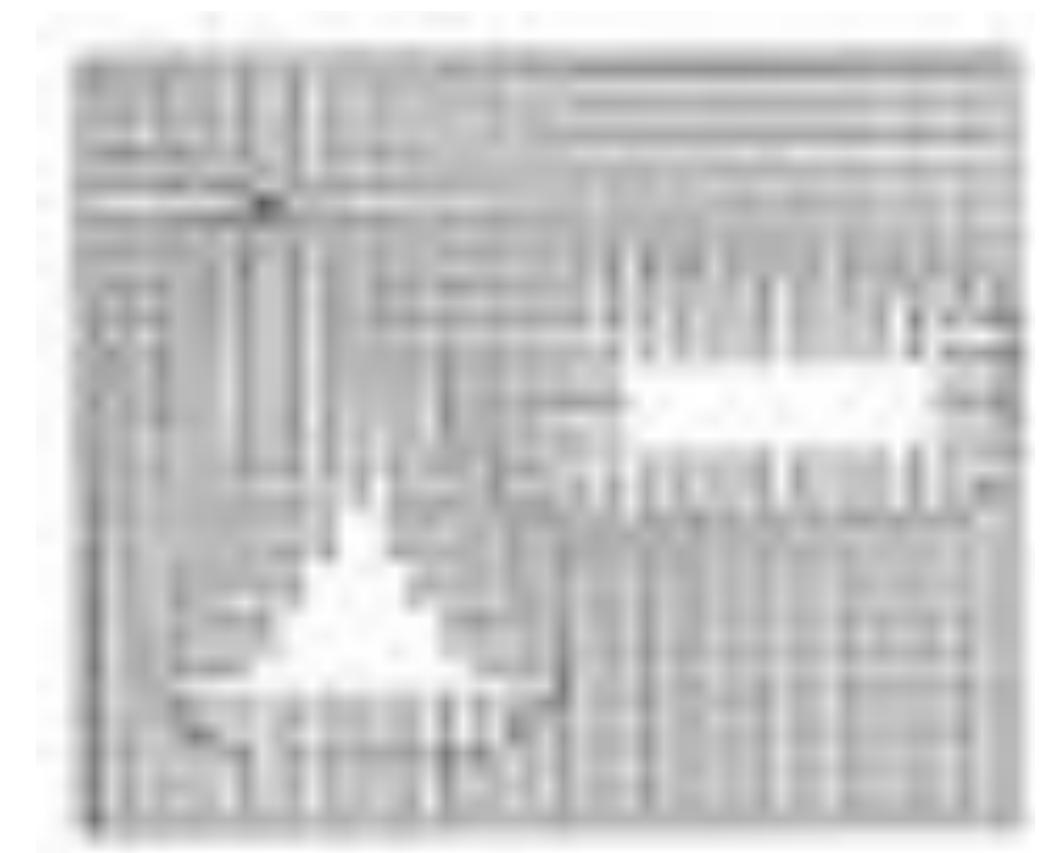
=



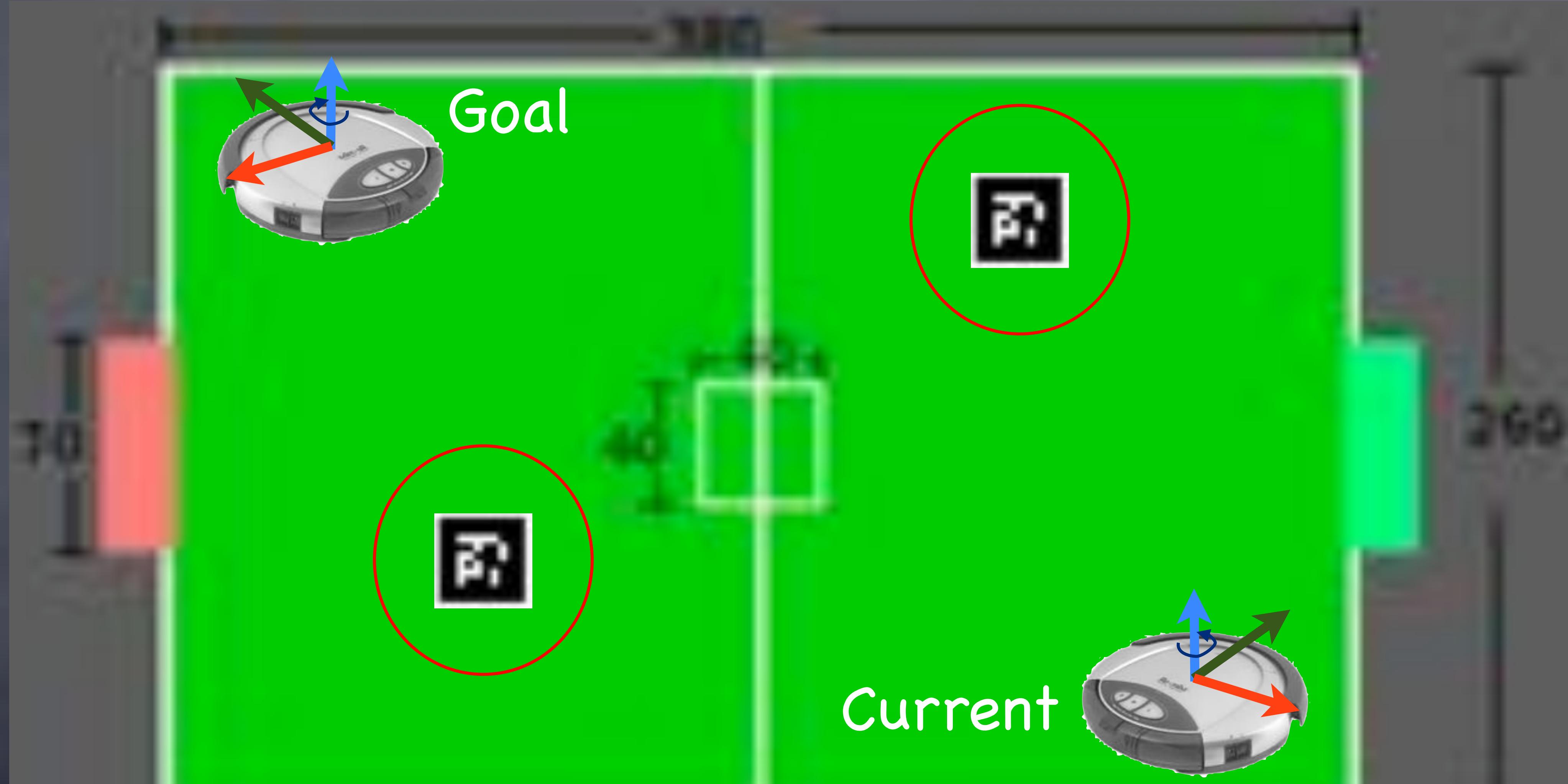
path from descent
on gradient field



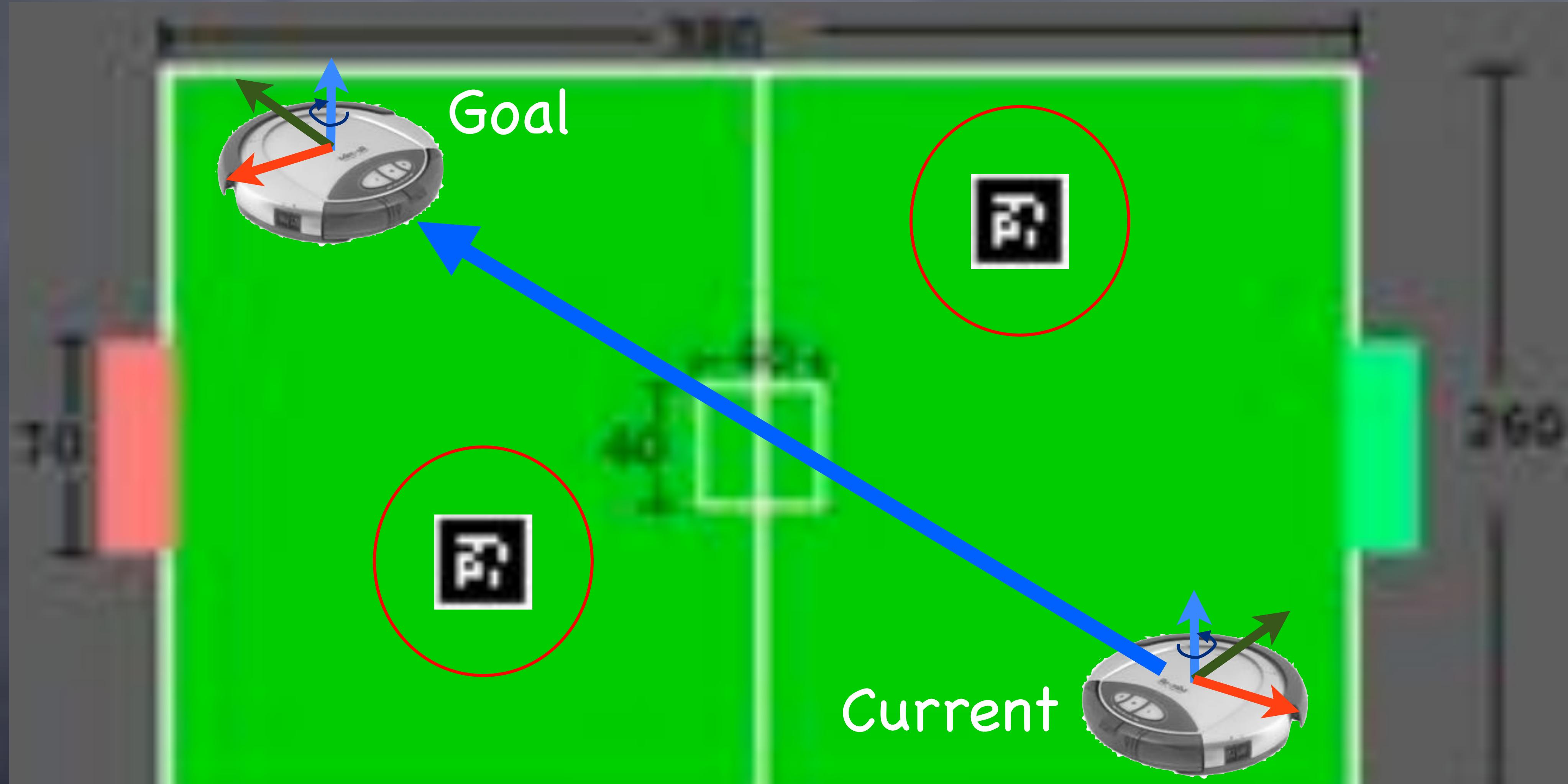
resulting
gradient field



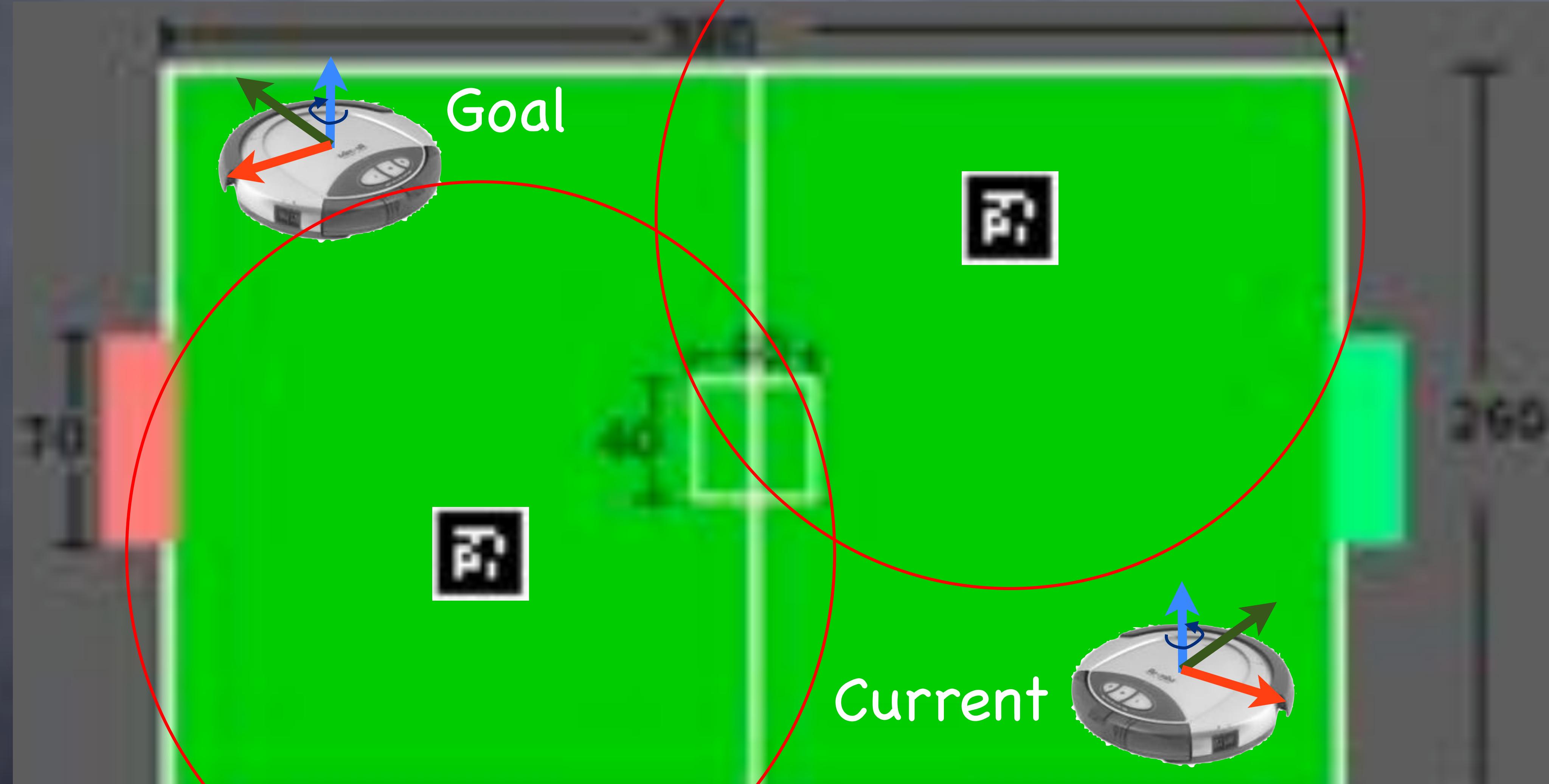
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



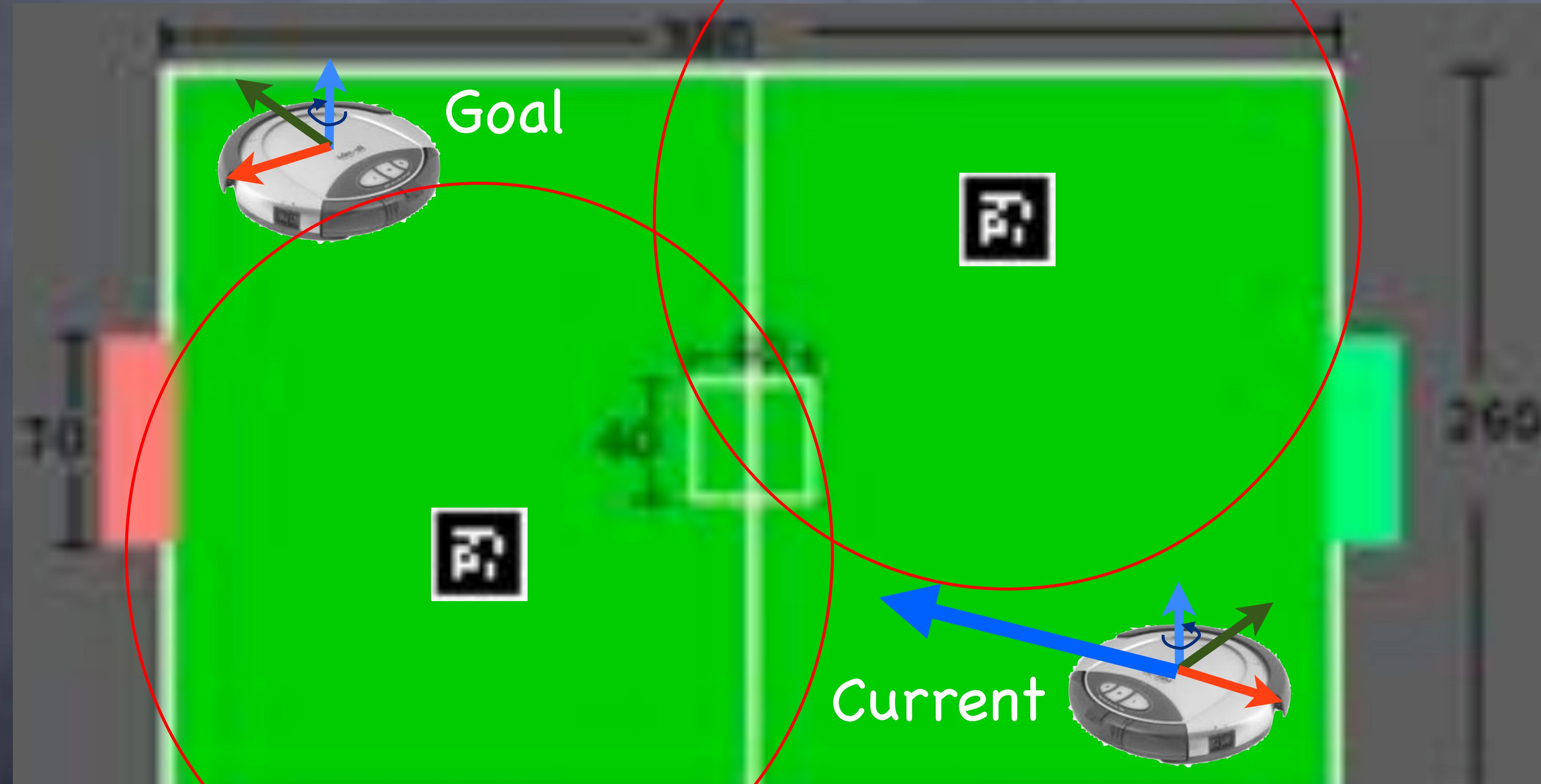
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



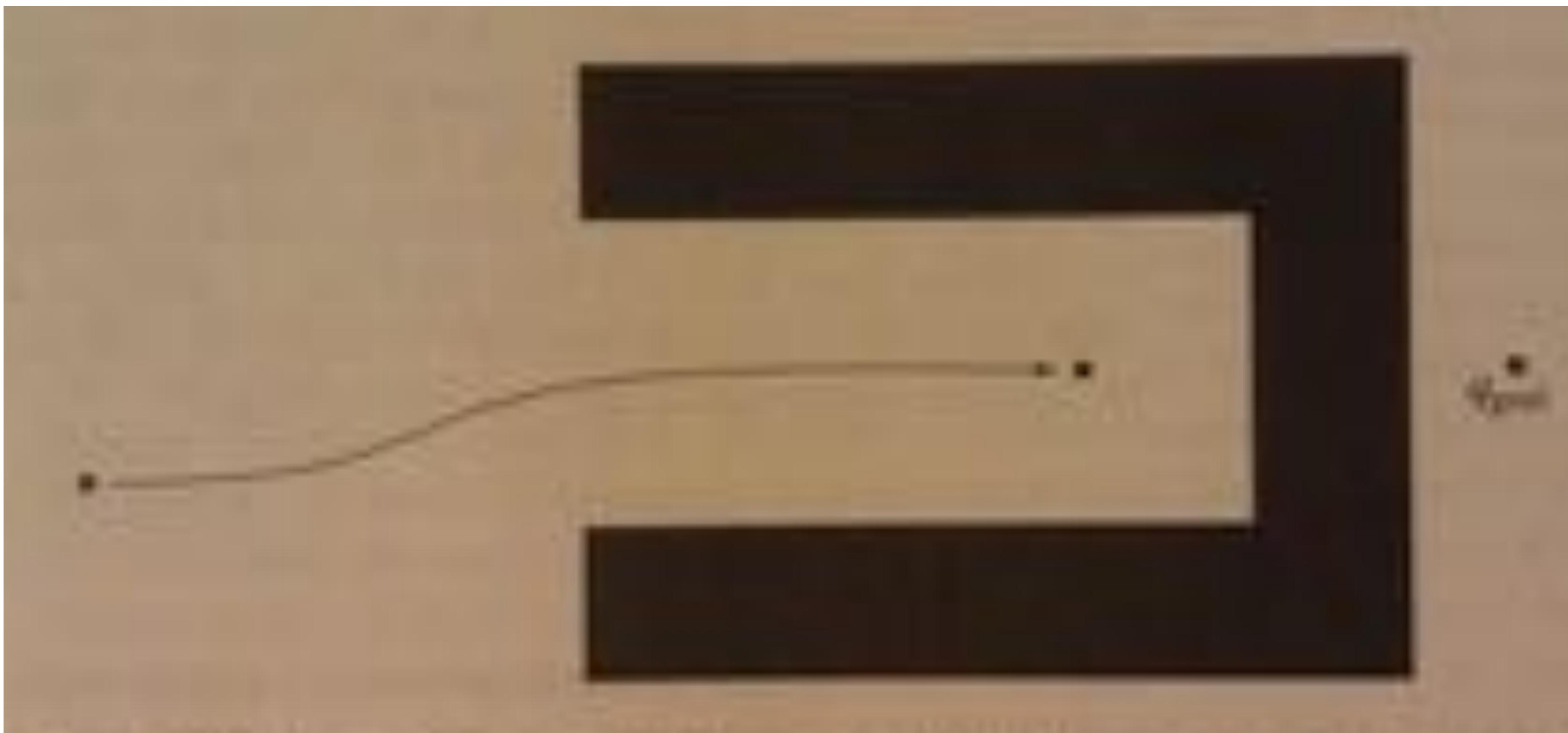
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



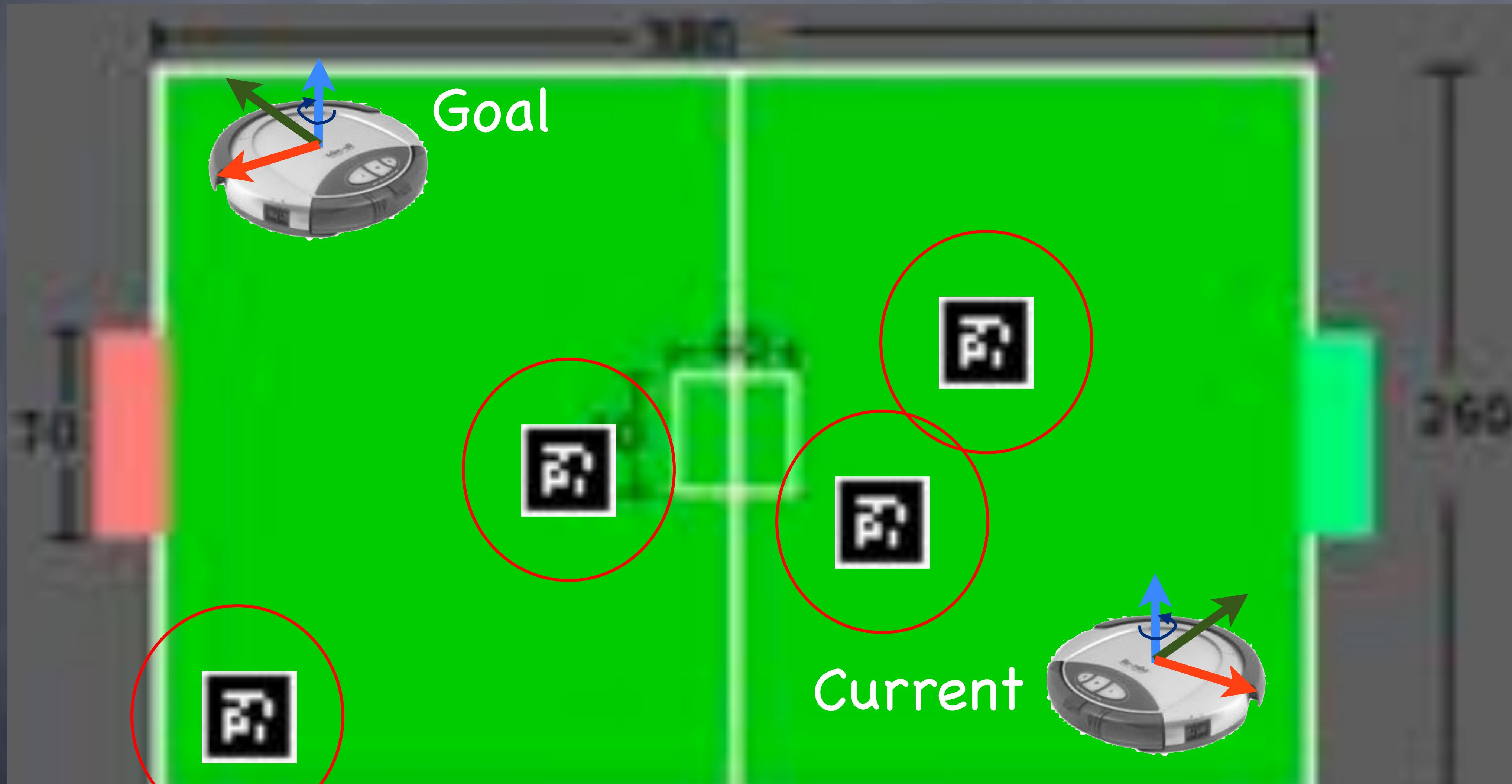
describe performance for this case
with cone attractor to goal and bowl repellors
with limited weight



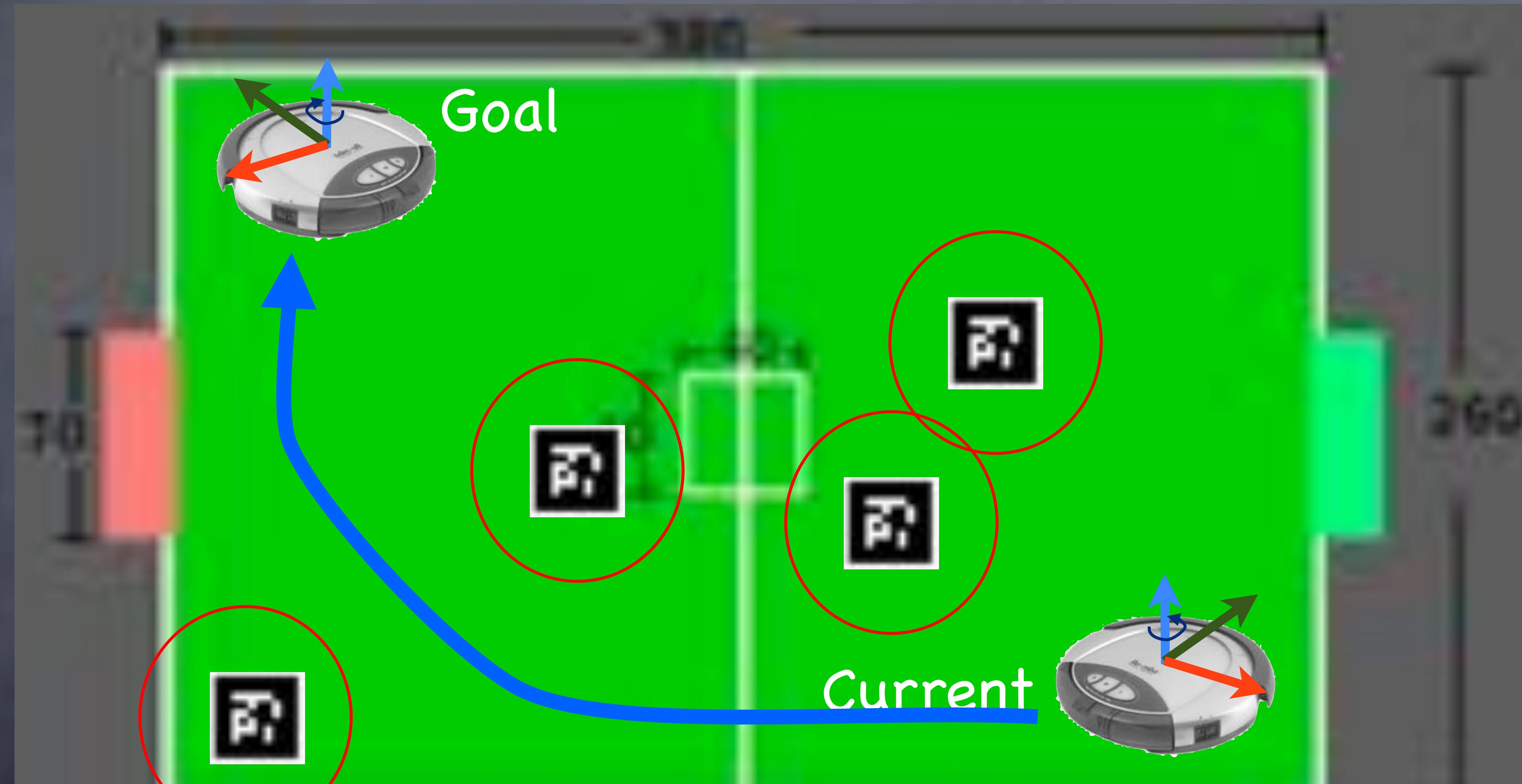
Local Minima



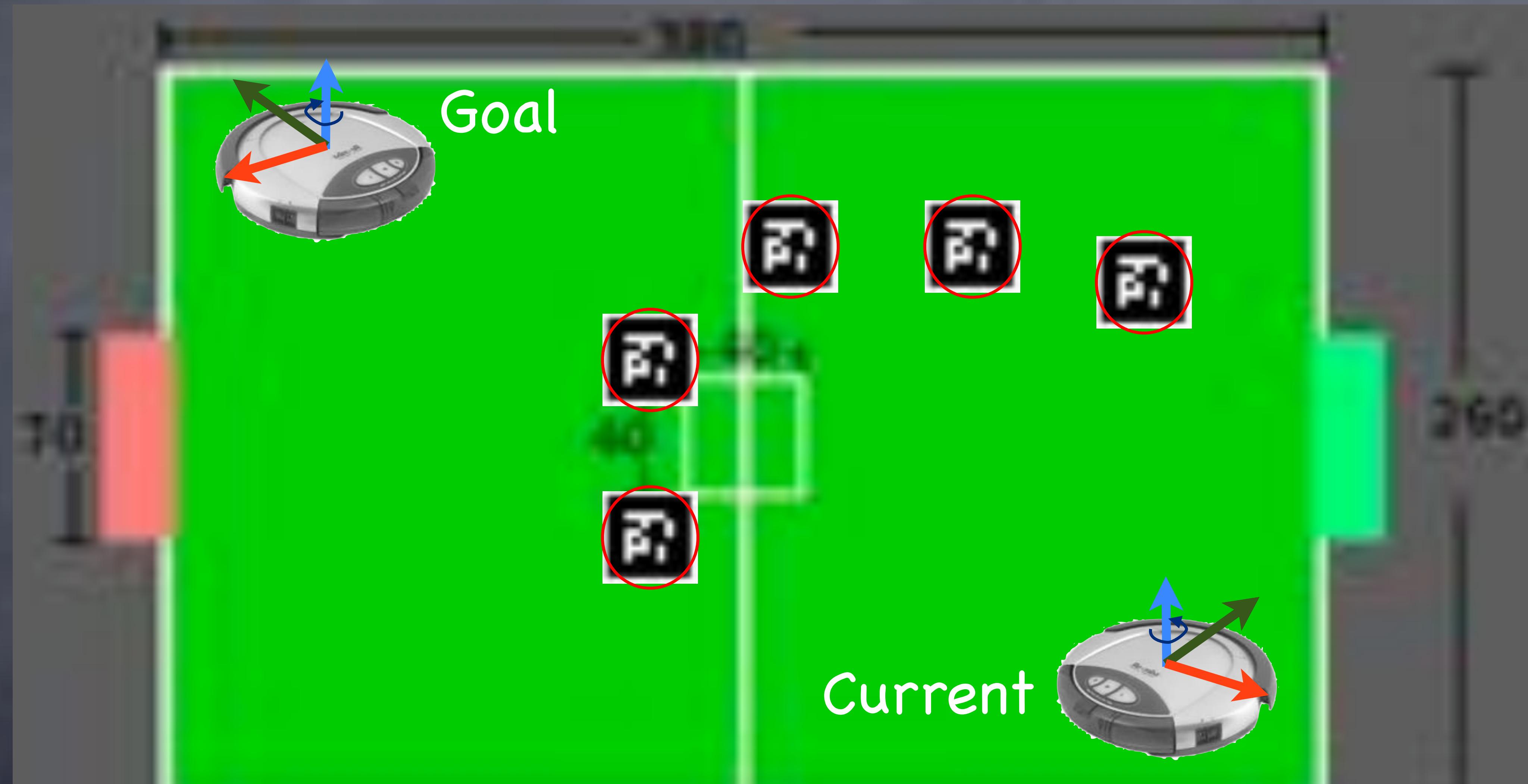
describe performance for this case



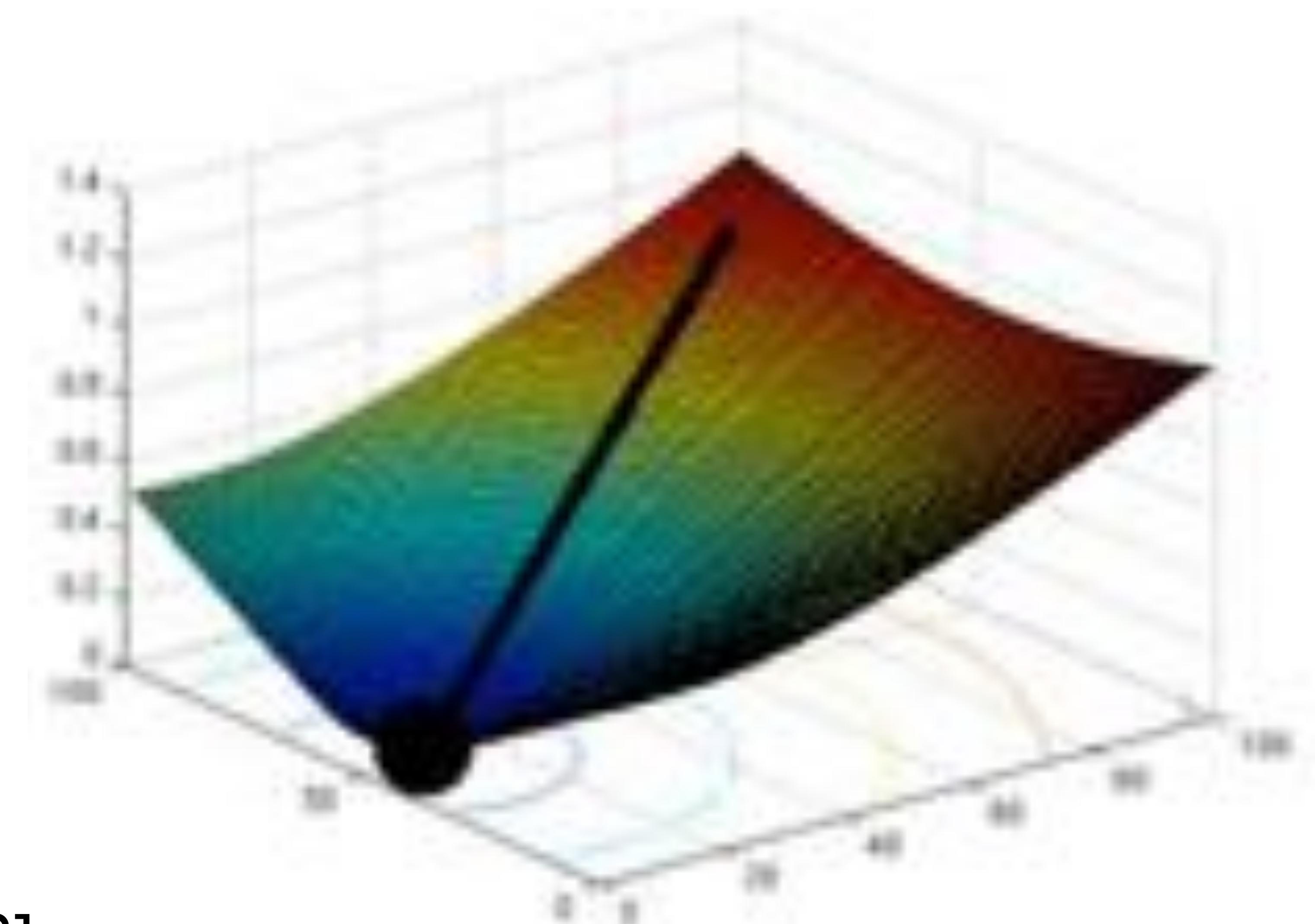
describe performance for this case



describe performance for this case



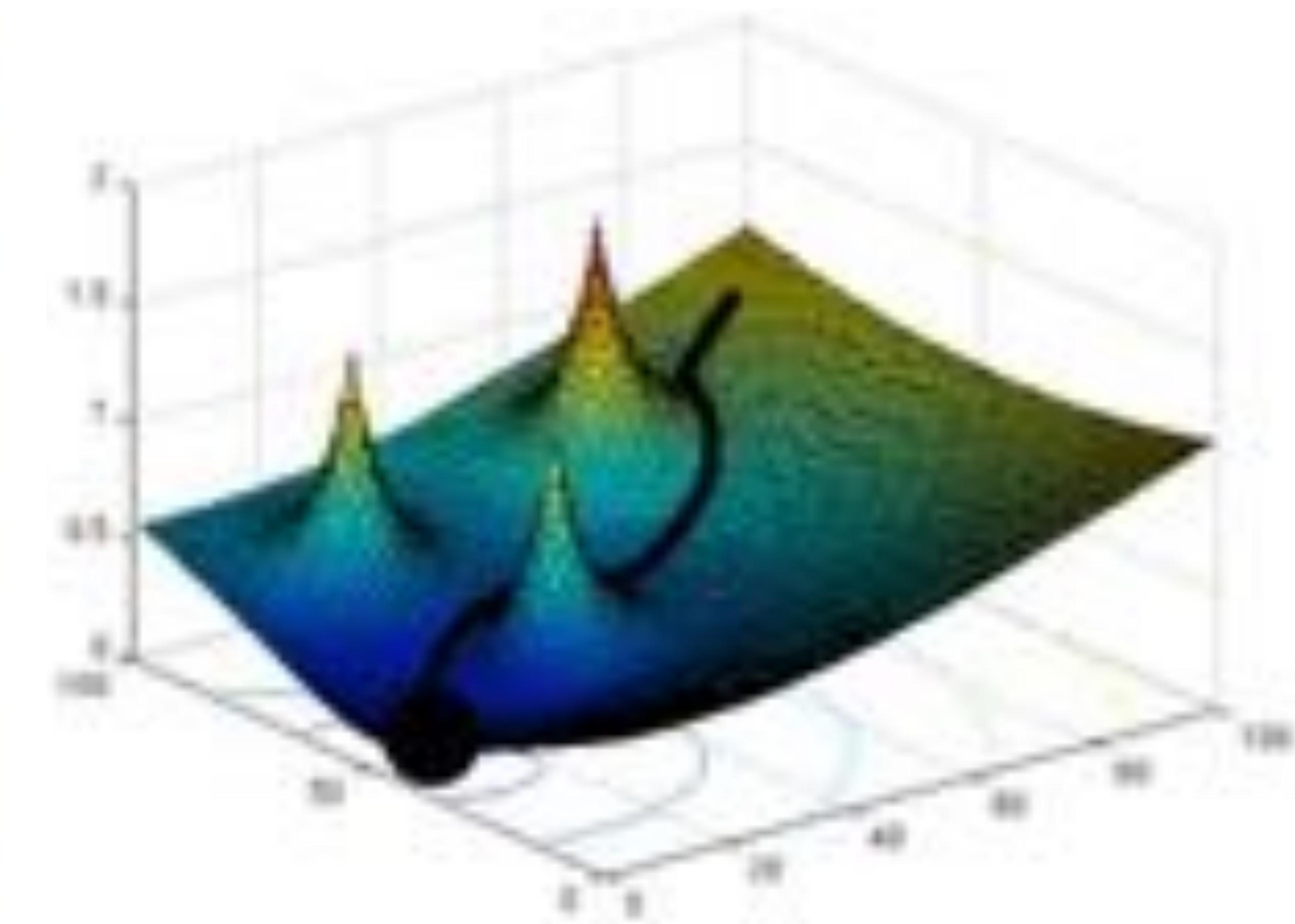
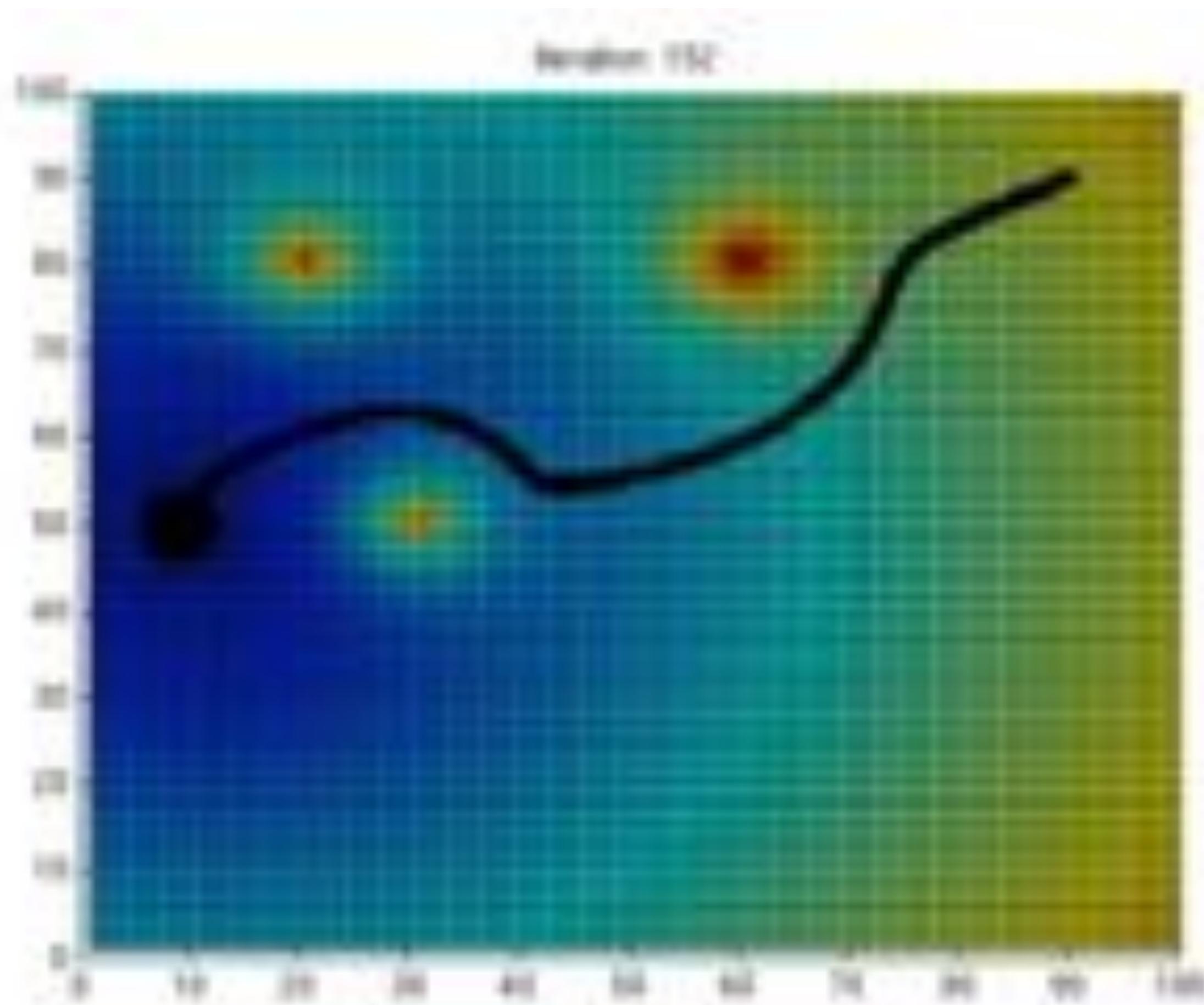
matlab example



pfield.m [I 5 8 12]



matlab example

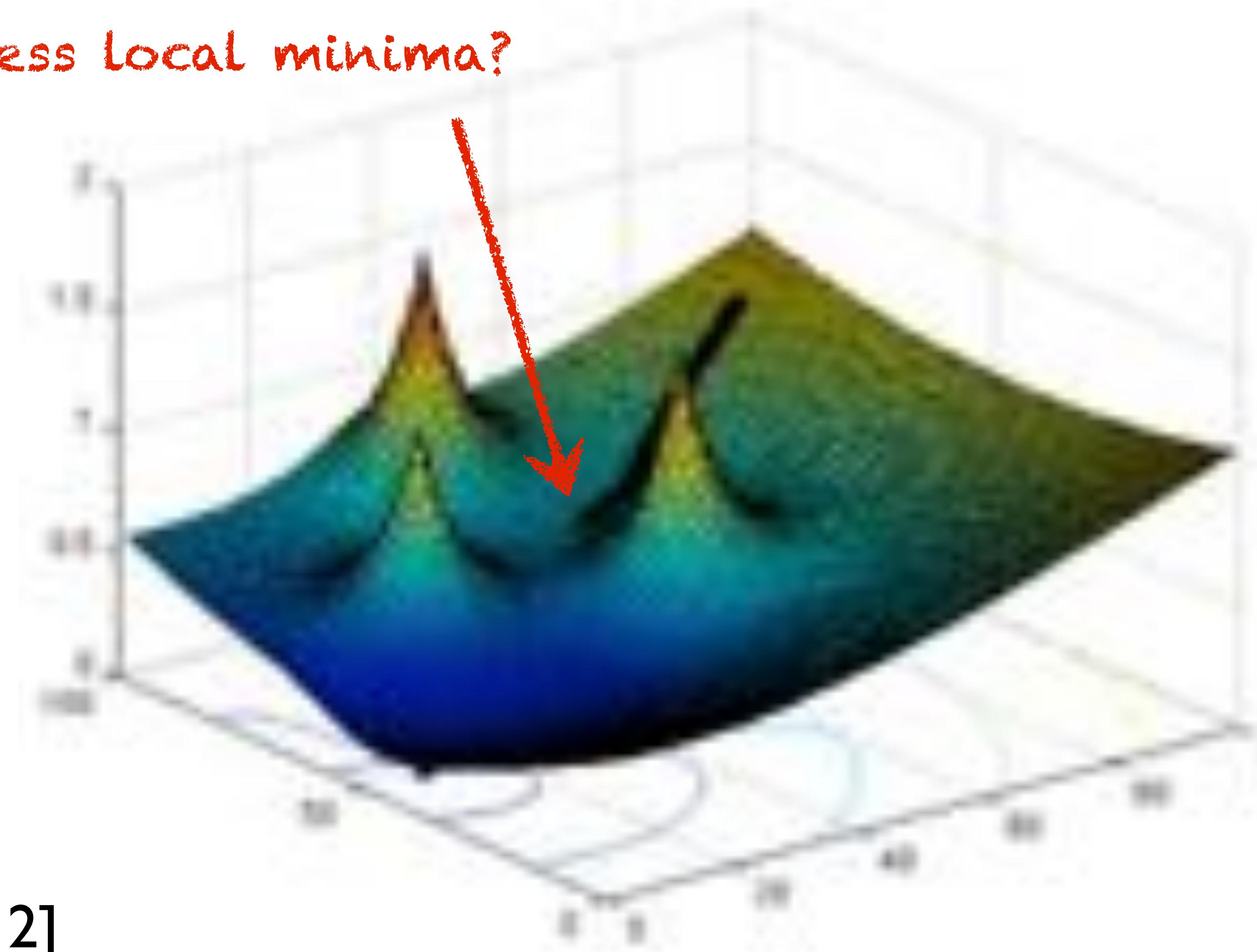


pfield.m [1 5 8 12]



matlab example

How to address Local minima?



pfield.m [I 5 8 12]



How can we get out of local minima?

How can we get out of local minima?

Go back to planning.

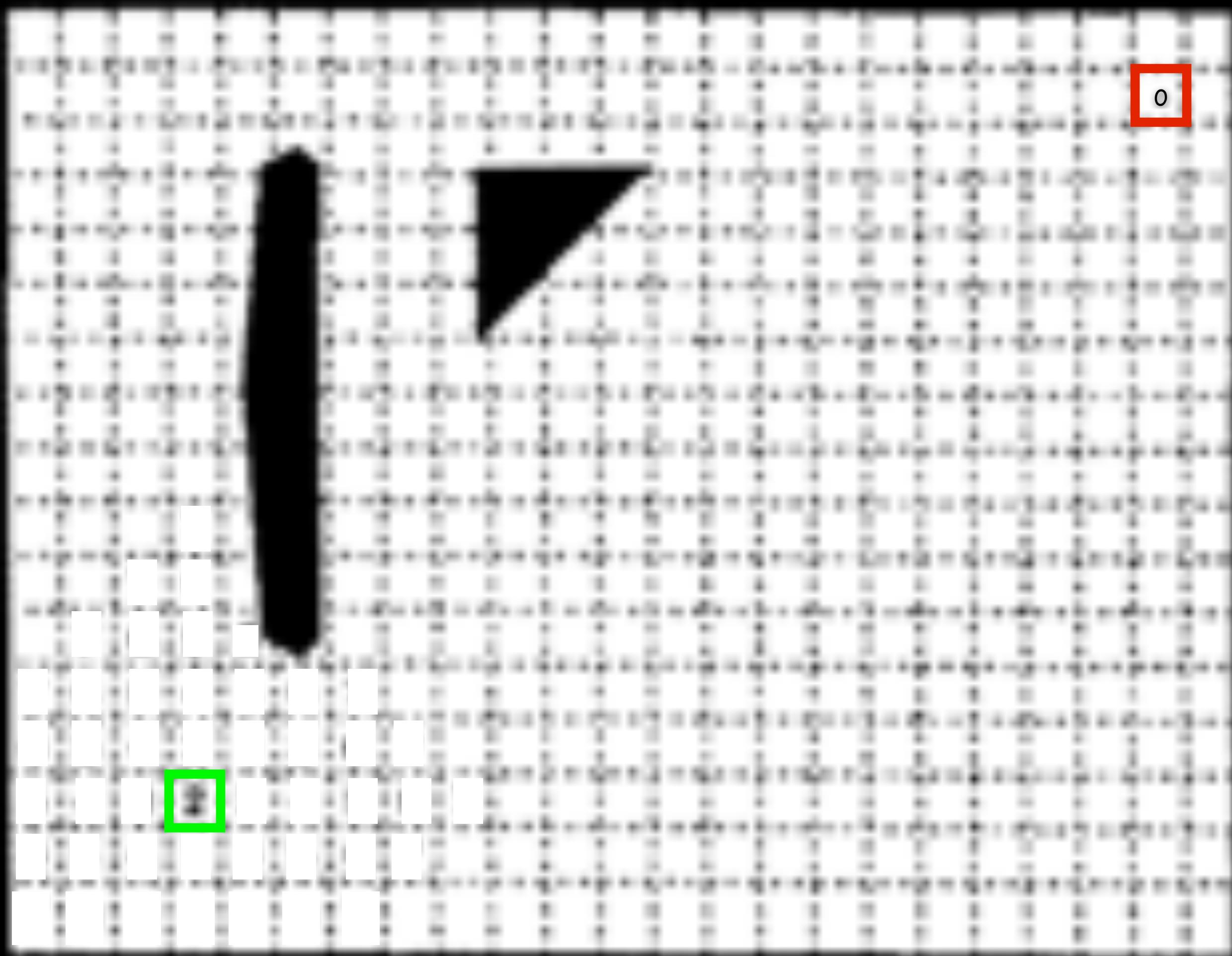


Wavefront Planning

- 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

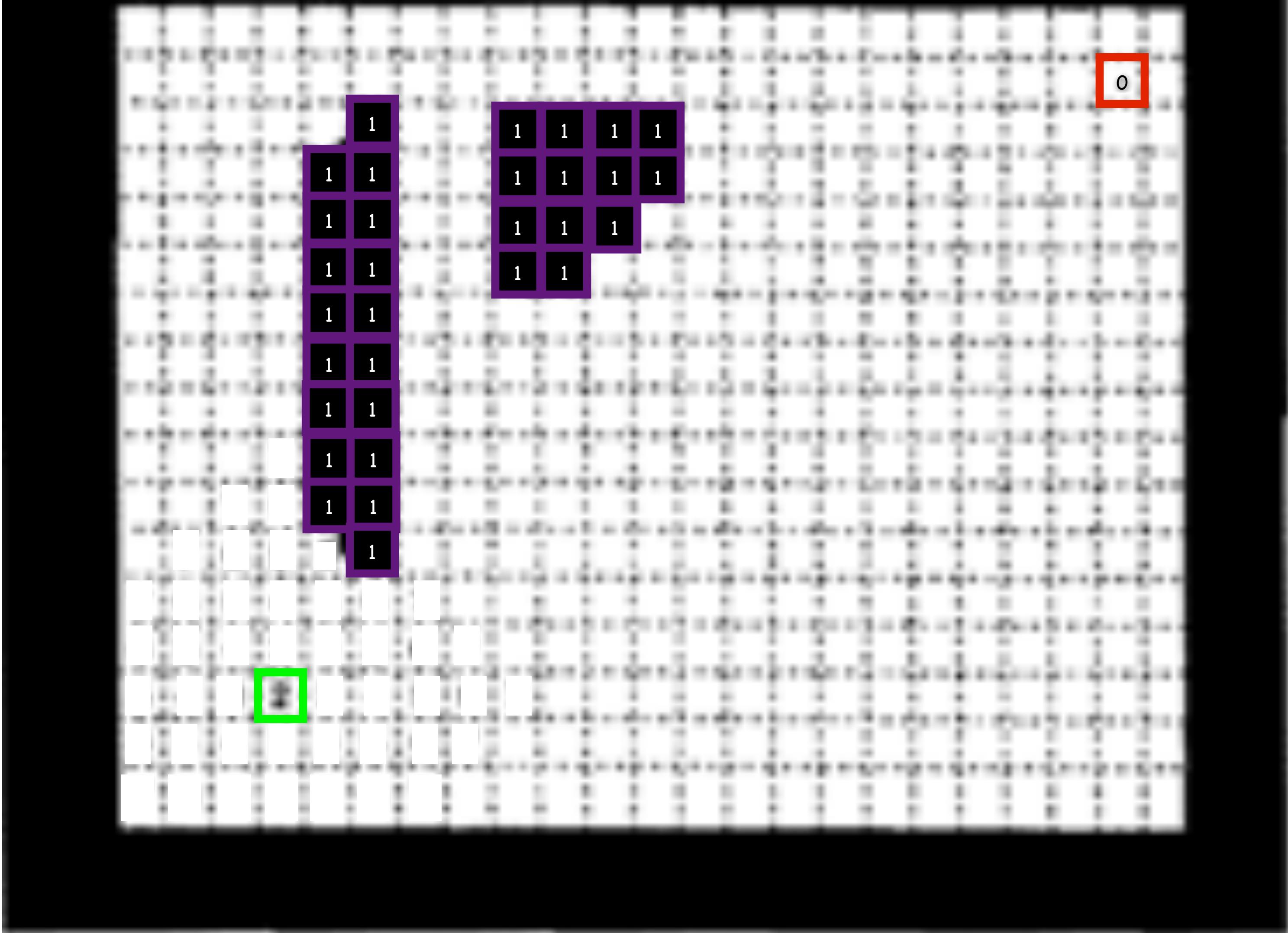


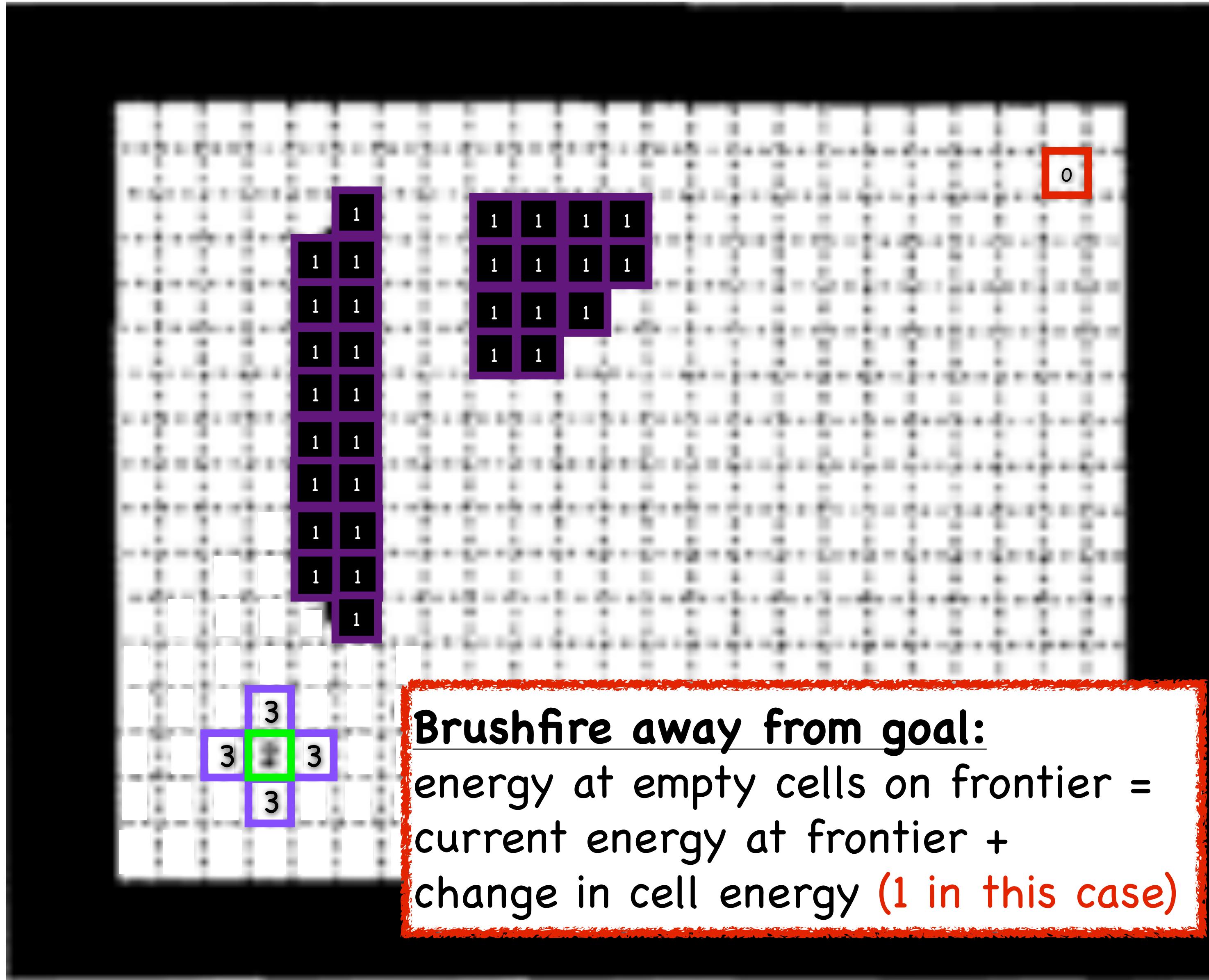
Start: mark with 0

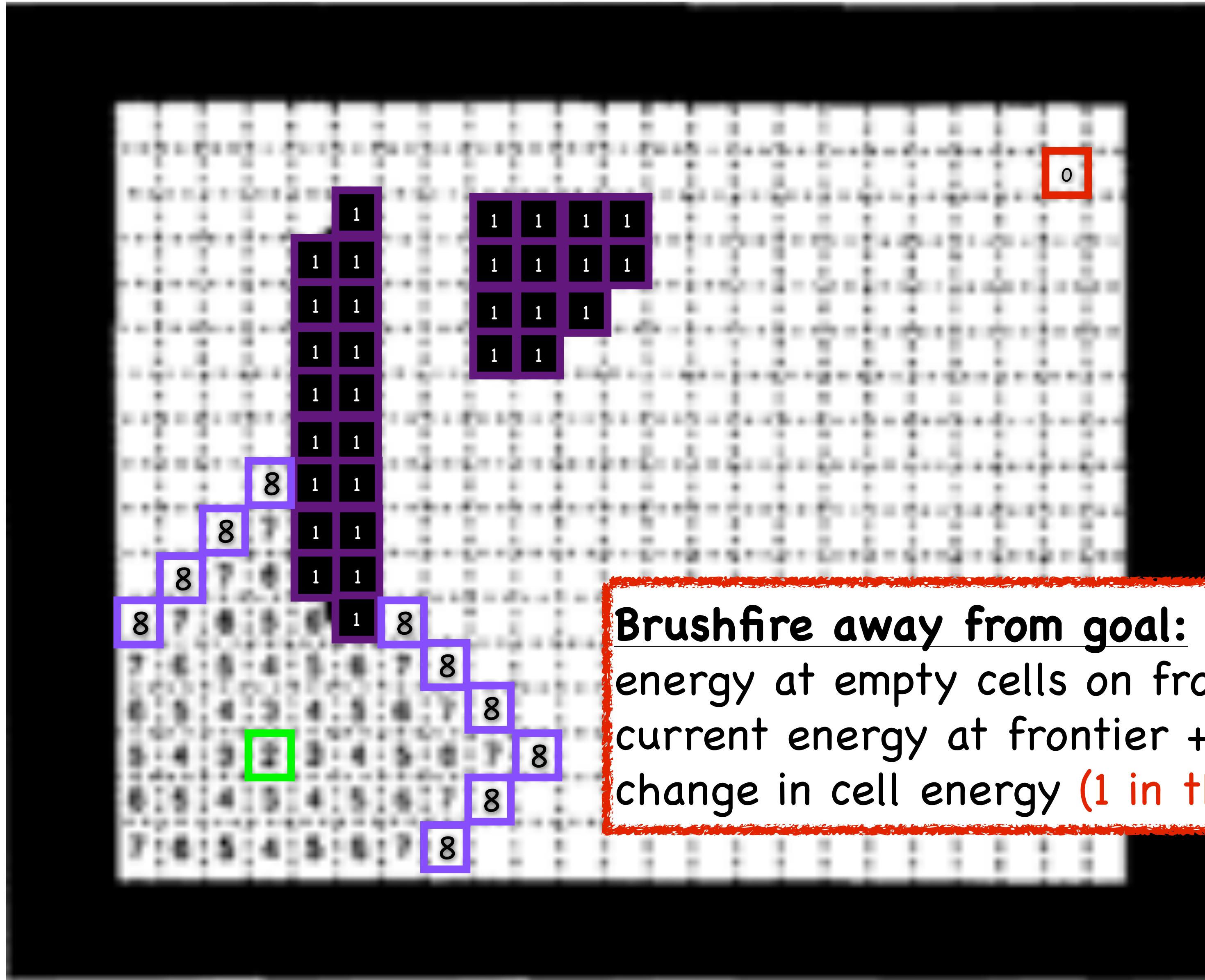


Goal: mark with 2

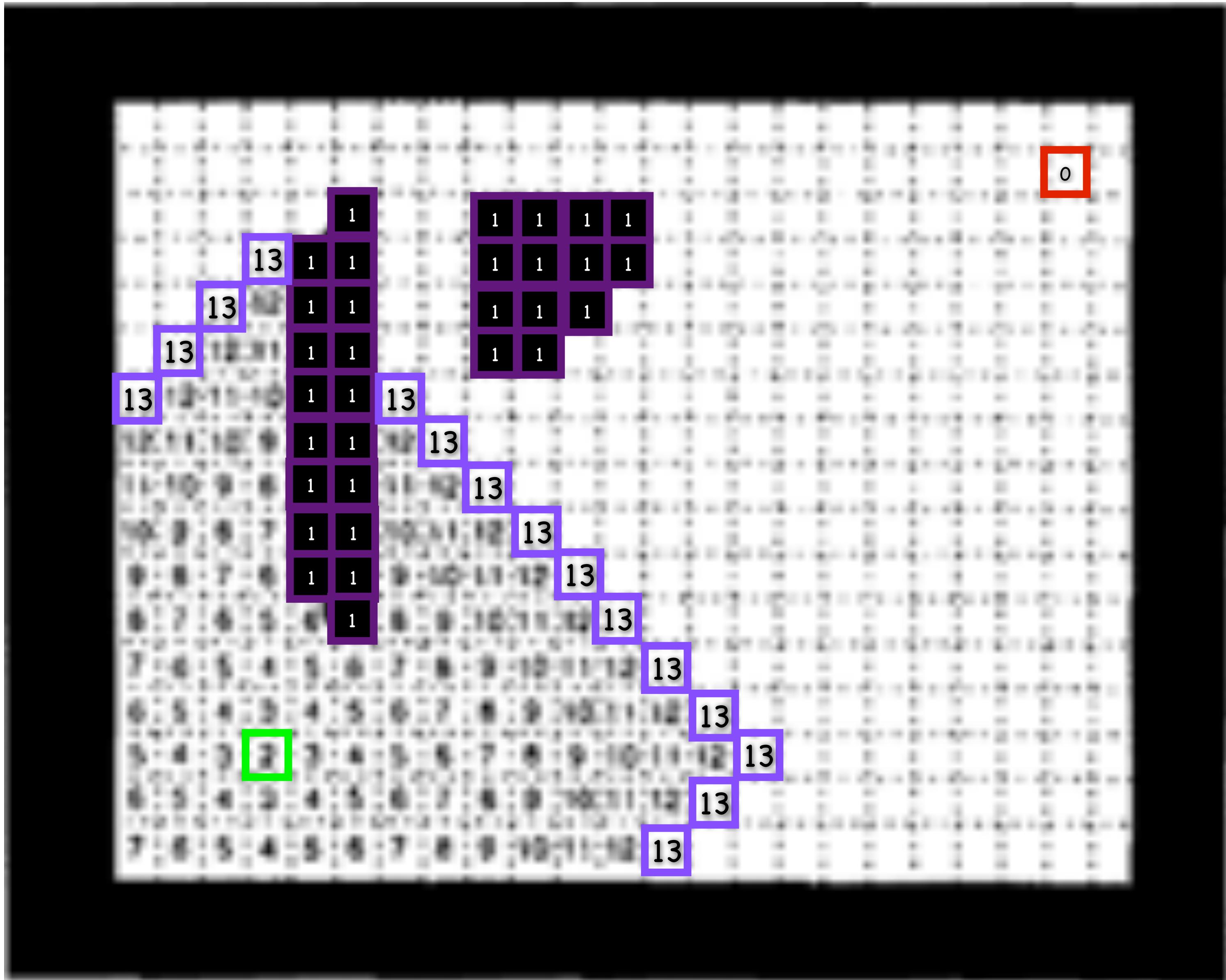
Obstacles: mark with 1

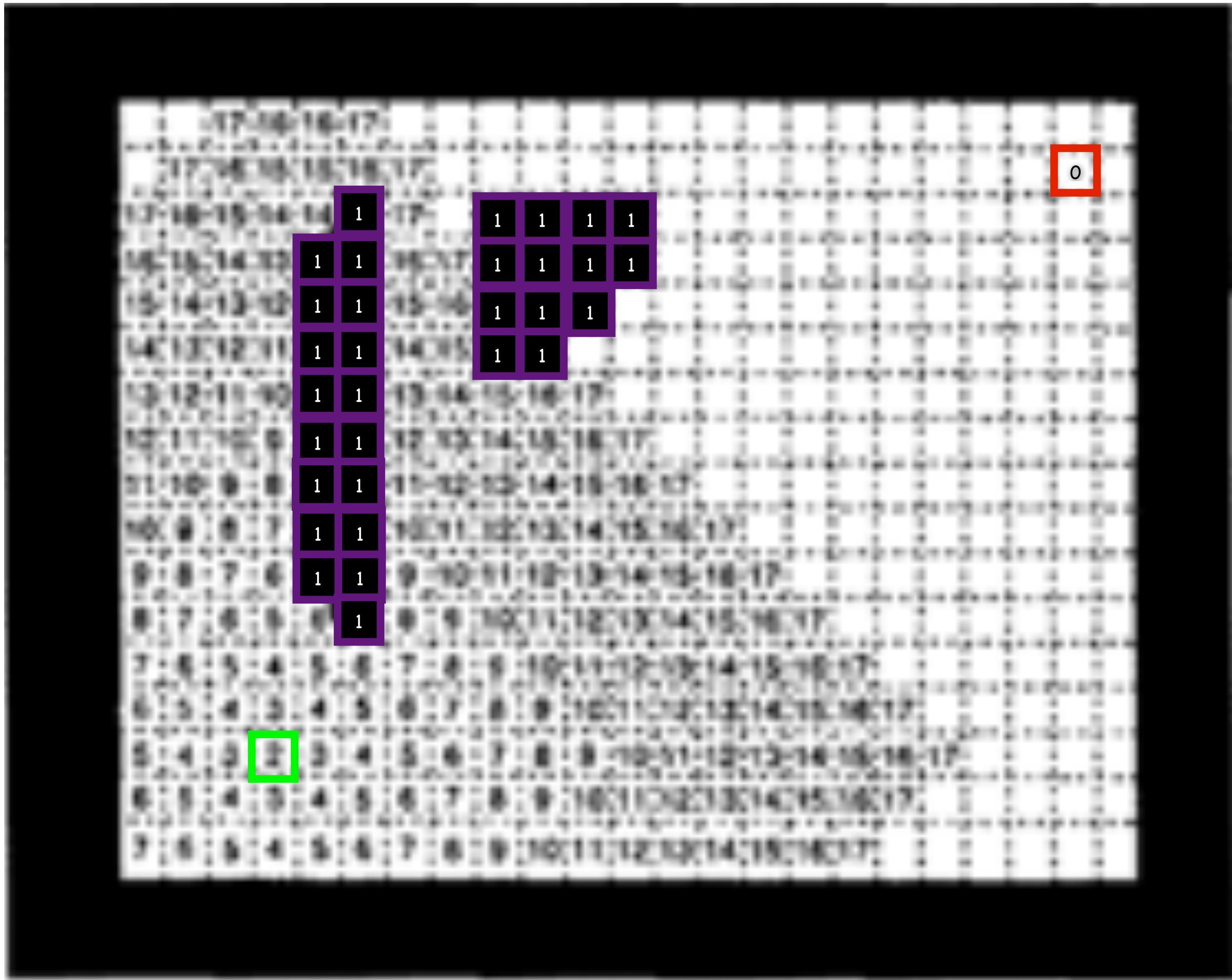


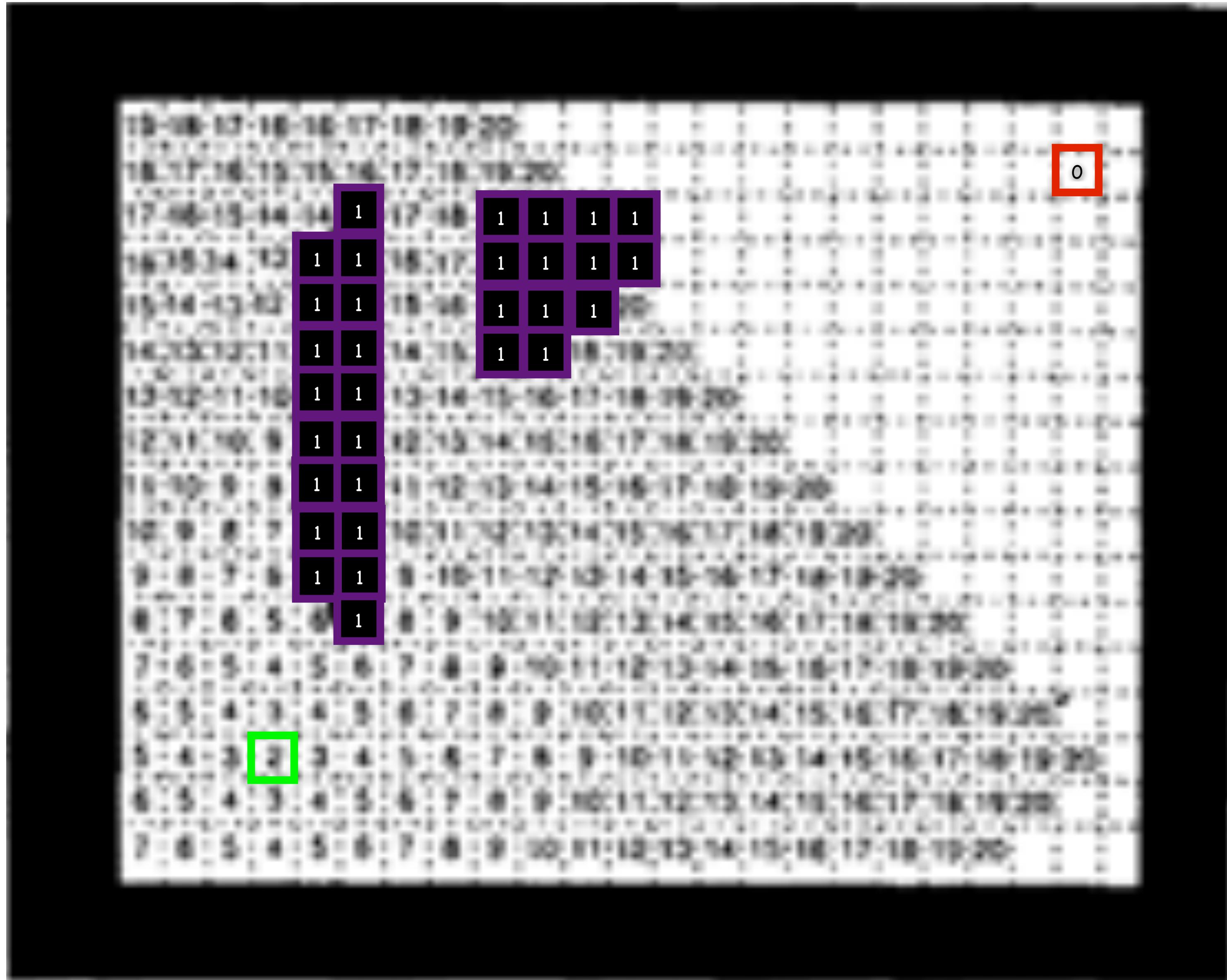


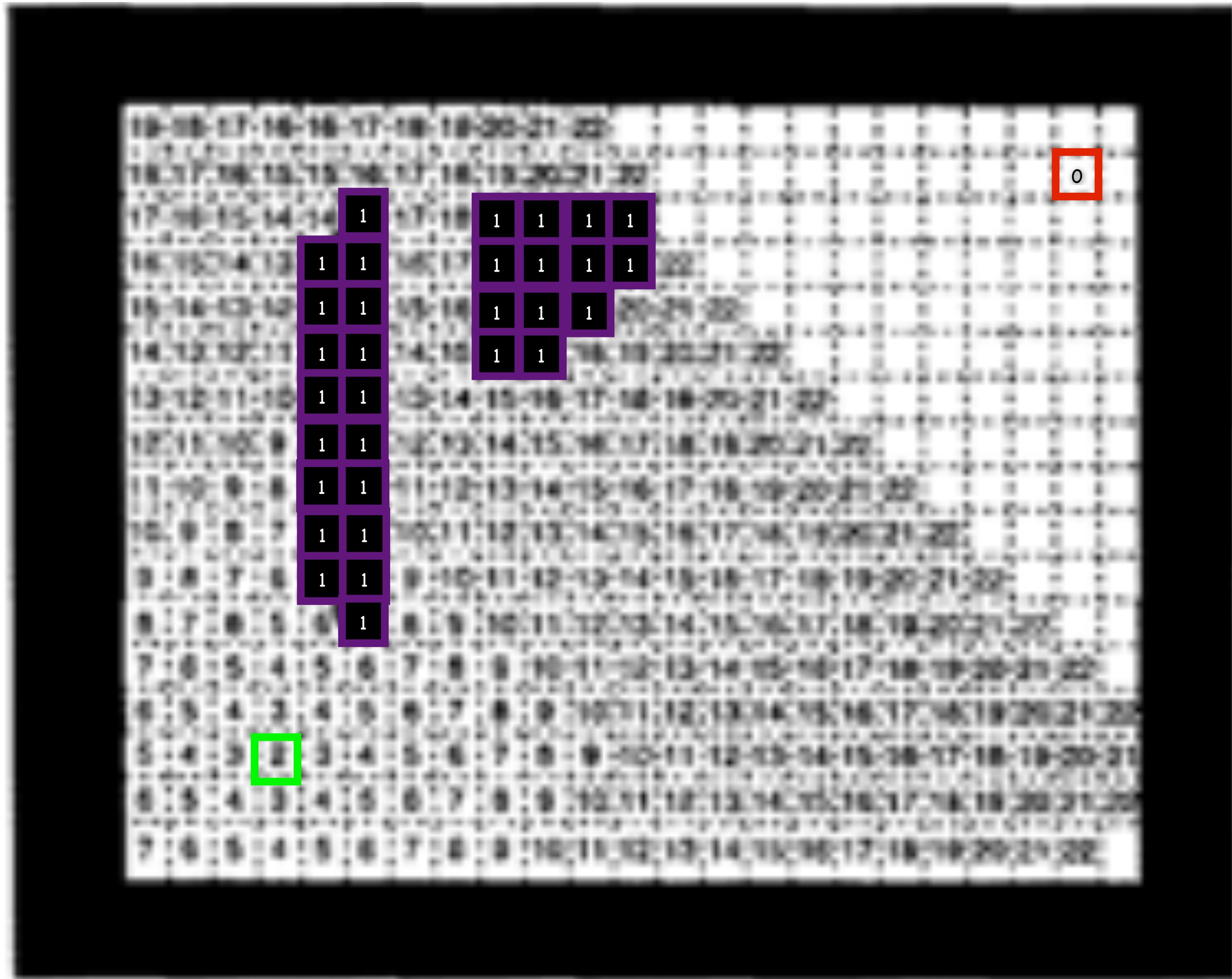


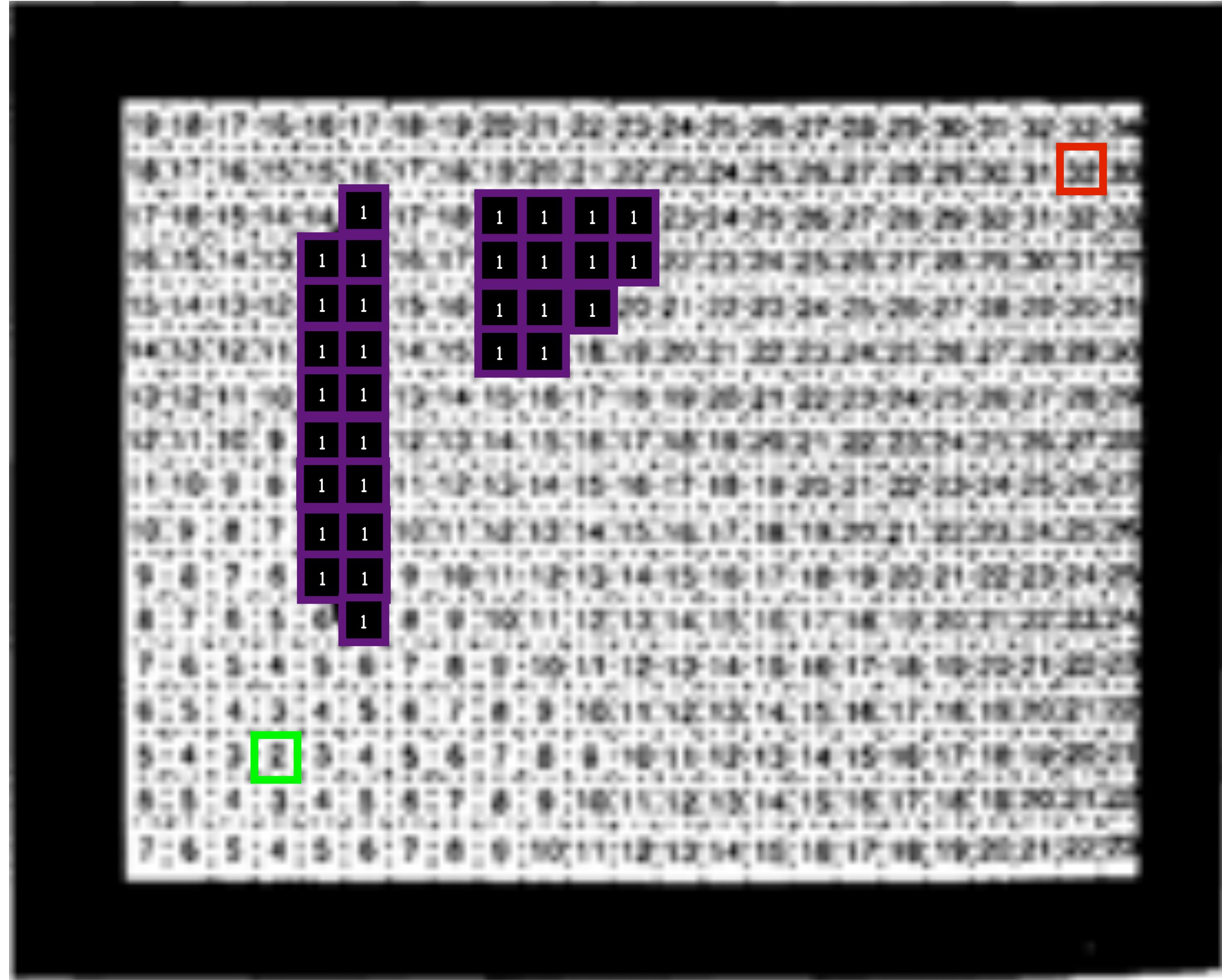
Brushfire away from goal:
energy at empty cells on frontier =
current energy at frontier +
change in cell energy (1 in this case)



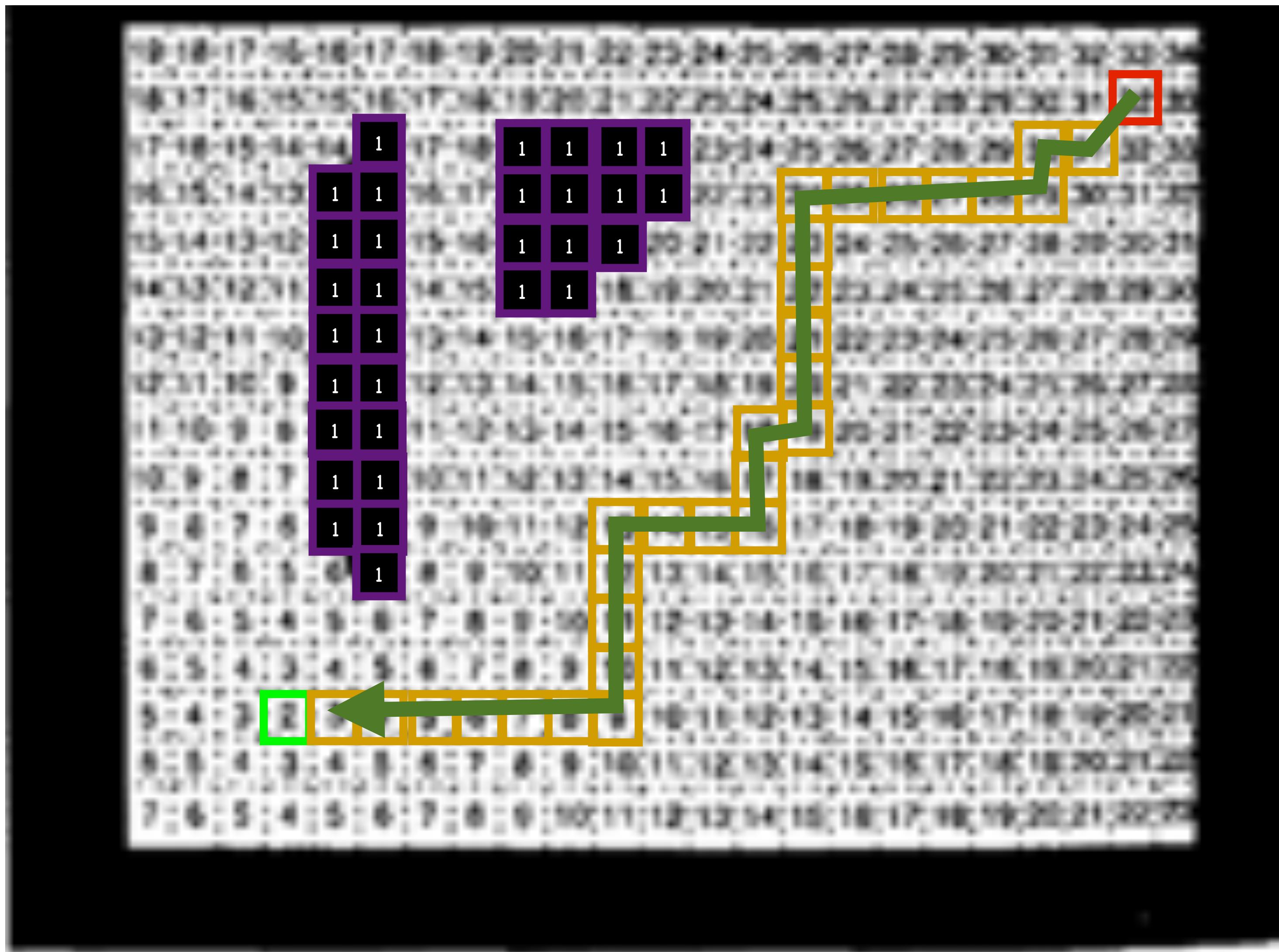






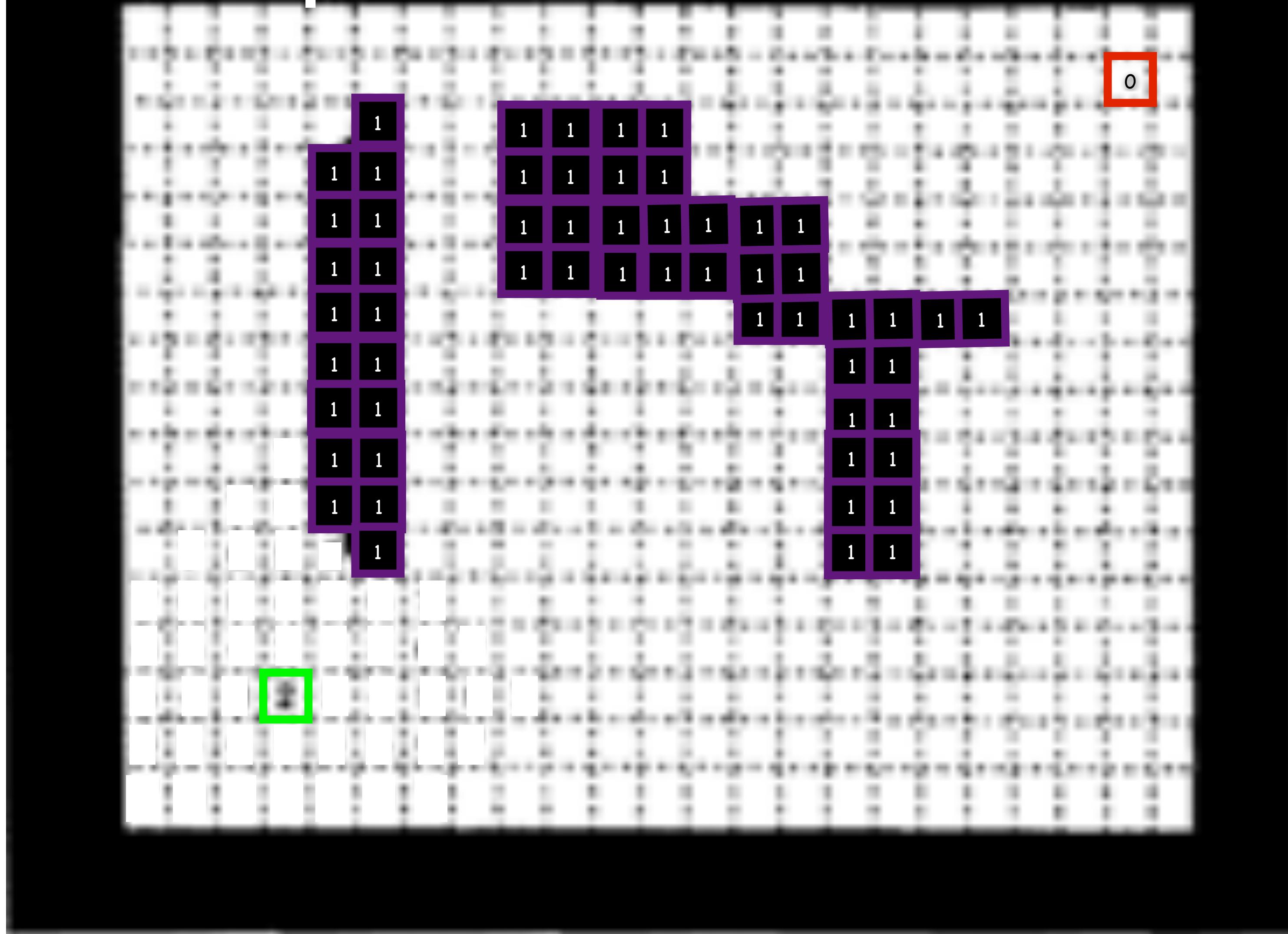


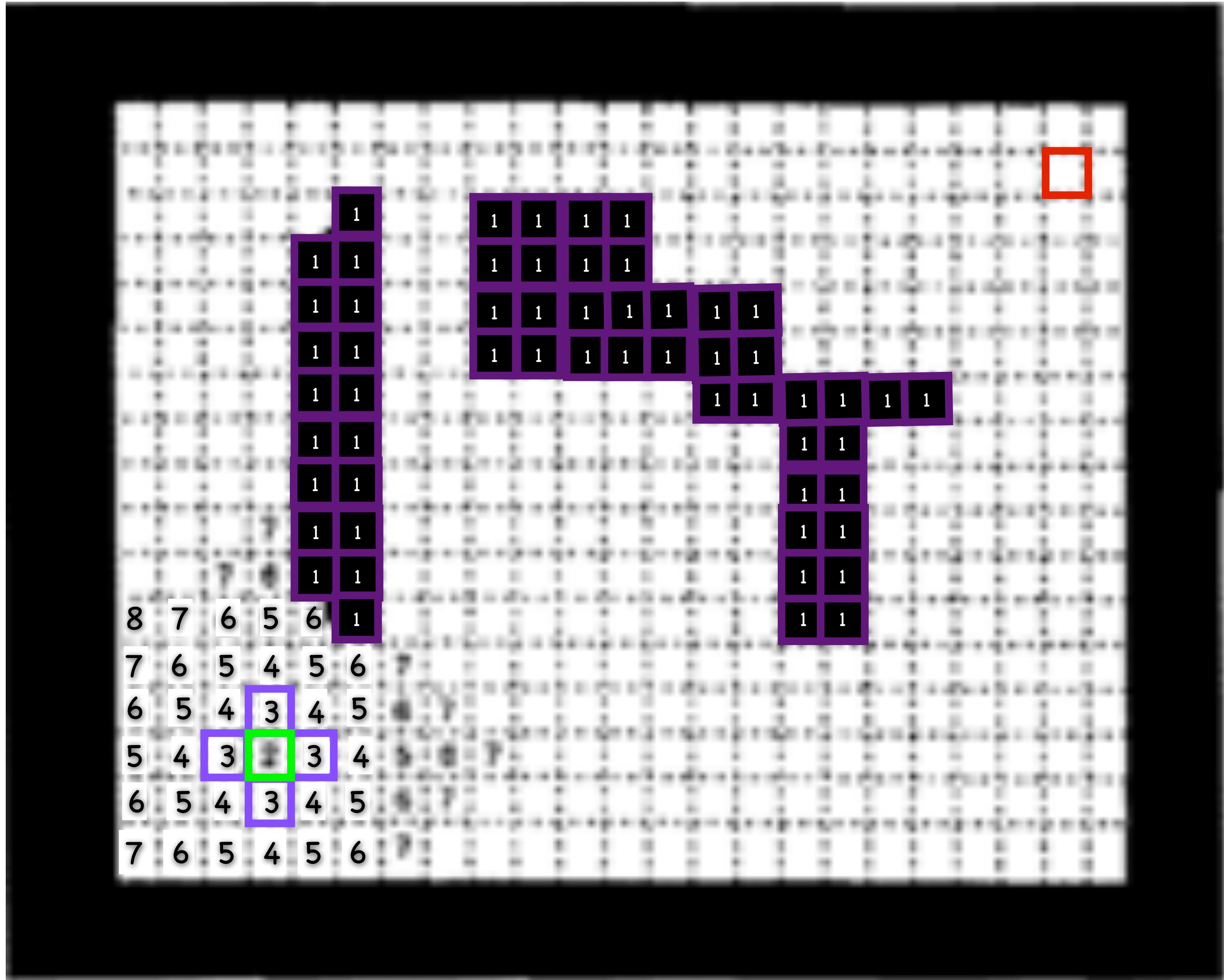
Once start reached,
follow brushfire potential to goal

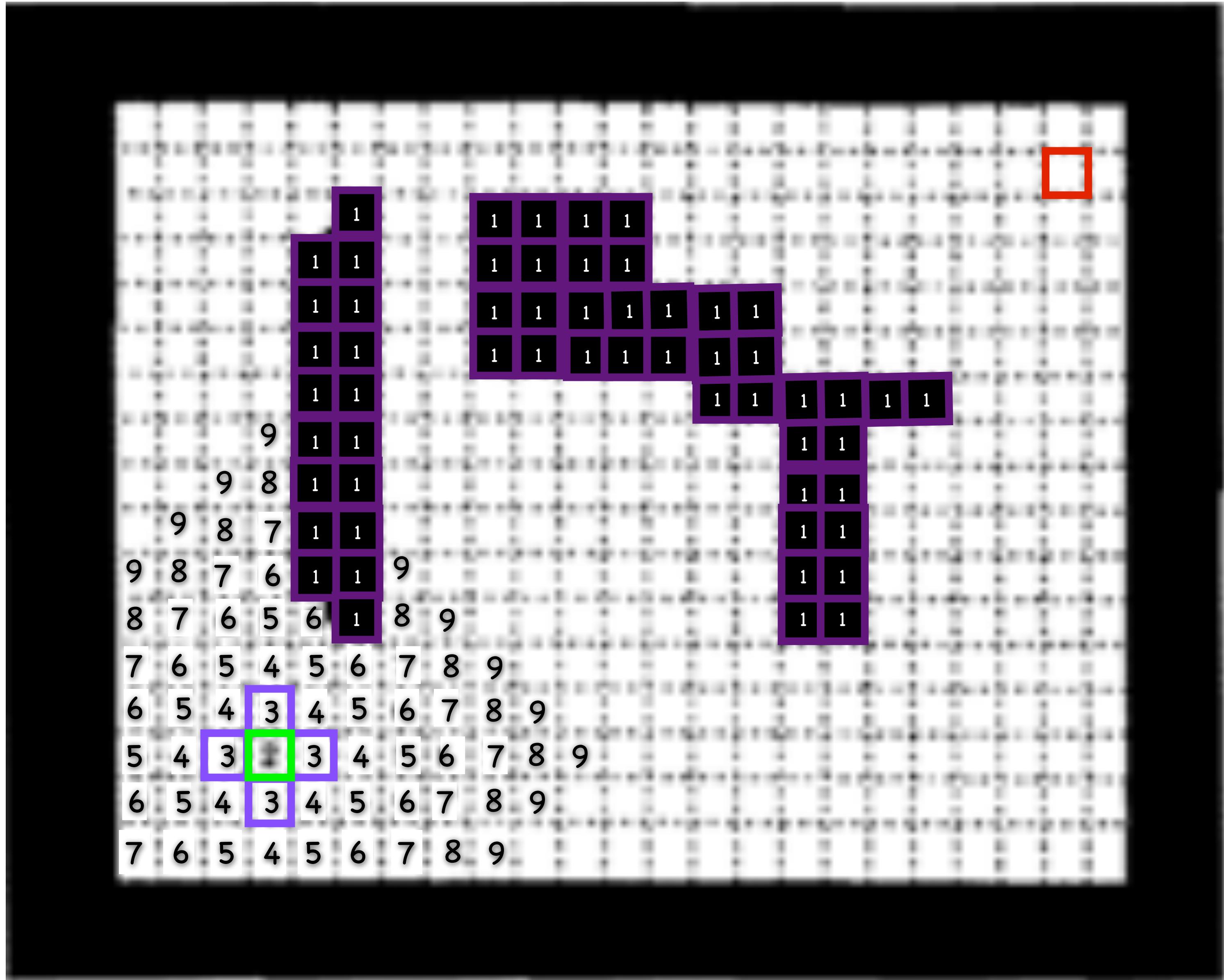


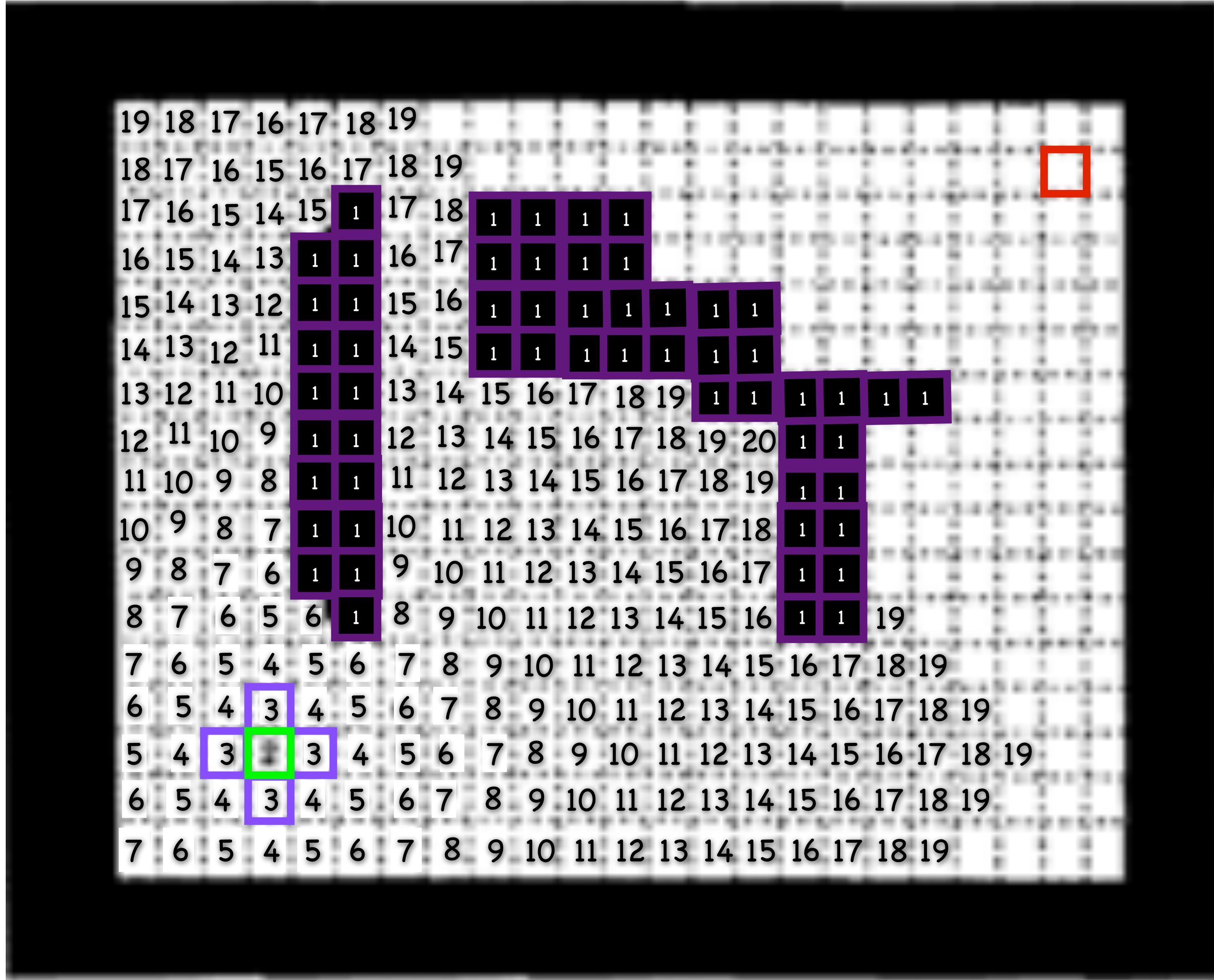
Example with Local Minima

Example with Local Minima









| | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 19 | 18 | 17 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | | | | | | |
| 18 | 17 | 16 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | | | | | |
| 17 | 16 | 15 | 14 | 15 | 1 | 17 | 18 | 1 | 1 | 1 | 1 | 25 | 26 | 27 | 28 | 29 | | | | | | |
| 16 | 15 | 14 | 13 | 1 | 1 | 16 | 17 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 | | | | | | | |
| 15 | 14 | 13 | 12 | 1 | 1 | 15 | 16 | 1 | 1 | 1 | 1 | 1 | 1 | 29 | 28 | 29 | | | | | | |
| 14 | 13 | 12 | 11 | 1 | 1 | 14 | 15 | 1 | 1 | 1 | 1 | 1 | 1 | 29 | 28 | 27 | 29 | | | | | |
| 13 | 12 | 11 | 10 | 1 | 1 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 | | |
| 12 | 11 | 10 | 9 | 1 | 1 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 1 | 1 | 23 | 24 | 25 | 26 | 27 | 28 |
| 11 | 10 | 9 | 8 | 1 | 1 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 22 | 23 | 24 | 25 | 26 | 27 |
| 10 | 9 | 8 | 7 | 1 | 1 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 1 | 1 | 21 | 22 | 23 | 24 | 25 | 26 |
| 9 | 8 | 7 | 6 | 1 | 1 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 1 | 1 | 20 | 21 | 22 | 23 | 24 | 25 |
| 8 | 7 | 6 | 5 | 6 | 1 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 1 | 1 | 19 | 20 | 21 | 22 | 23 | 24 |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 5 | 4 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 19 | 18 | 17 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | |
| 18 | 17 | 16 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | |
| 17 | 16 | 15 | 14 | 15 | 1 | 17 | 18 | 1 | 1 | 1 | 1 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 30 | 31 | 32 | 33 | |
| 16 | 15 | 14 | 13 | 13 | 1 | 1 | 16 | 17 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 | 30 | 31 | 30 | 29 | 30 | 31 | 32 |
| 15 | 14 | 13 | 12 | 12 | 1 | 1 | 15 | 16 | 1 | 1 | 1 | 1 | 1 | 1 | 30 | 31 | 30 | 29 | 28 | 29 | 30 | 31 | |
| 14 | 13 | 12 | 11 | 11 | 1 | 1 | 14 | 15 | 1 | 1 | 1 | 1 | 1 | 1 | 31 | 30 | 29 | 28 | 27 | 28 | 29 | 30 | |
| 13 | 12 | 11 | 10 | 10 | 1 | 1 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 |
| 12 | 11 | 10 | 9 | 9 | 1 | 1 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 1 | 1 | 23 | 24 | 25 | 26 | 27 | 28 |
| 11 | 10 | 9 | 8 | 8 | 1 | 1 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 22 | 23 | 24 | 25 | 26 | 27 |
| 10 | 9 | 8 | 7 | 7 | 1 | 1 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 1 | 1 | 21 | 22 | 23 | 24 | 25 | 26 |
| 9 | 8 | 7 | 6 | 6 | 1 | 1 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 1 | 1 | 20 | 21 | 22 | 23 | 24 | 25 |
| 8 | 7 | 6 | 5 | 6 | 1 | 1 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 1 | 1 | 19 | 20 | 21 | 22 | 23 | 24 |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | |
| 5 | 4 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | | |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | |

local minima avoided

| | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 19 | 18 | 17 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| 18 | 17 | 16 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 17 | 16 | 15 | 14 | 15 | 1 | 17 | 18 | 1 | 1 | 1 | 1 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 30 | 31 | 32 | 33 |
| 16 | 15 | 14 | 13 | 13 | 1 | 1 | 16 | 17 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 | 30 | 31 | 30 | 29 | 30 | 31 |
| 15 | 14 | 13 | 12 | 12 | 1 | 1 | 15 | 16 | 1 | 1 | 1 | 1 | 1 | 1 | 30 | 31 | 30 | 29 | 28 | 29 | 30 | 31 |
| 14 | 13 | 12 | 11 | 11 | 1 | 1 | 14 | 15 | 1 | 1 | 1 | 1 | 1 | 1 | 31 | 30 | 29 | 28 | 27 | 28 | 29 | 30 |
| 13 | 12 | 11 | 10 | 10 | 1 | 1 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 1 | 1 | 1 | 26 | 27 | 28 | 29 |
| 12 | 11 | 10 | 9 | 9 | 1 | 1 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 1 | 1 | 23 | 24 | 25 | 26 | 27 |
| 11 | 10 | 9 | 8 | 8 | 1 | 1 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1 | 1 | 22 | 23 | 24 | 25 | 26 |
| 10 | 9 | 8 | 7 | 7 | 1 | 1 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 1 | 1 | 21 | 22 | 23 | 24 | 25 |
| 9 | 8 | 7 | 6 | 6 | 1 | 1 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 1 | 1 | 20 | 21 | 22 | 23 | 24 |
| 8 | 7 | 6 | 5 | 6 | 1 | 1 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 1 | 1 | 19 | 20 | 21 | 22 | 23 |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 5 | 4 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

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



Next Lecture

Planning - VI - Collision Detection

