

Lecture 22

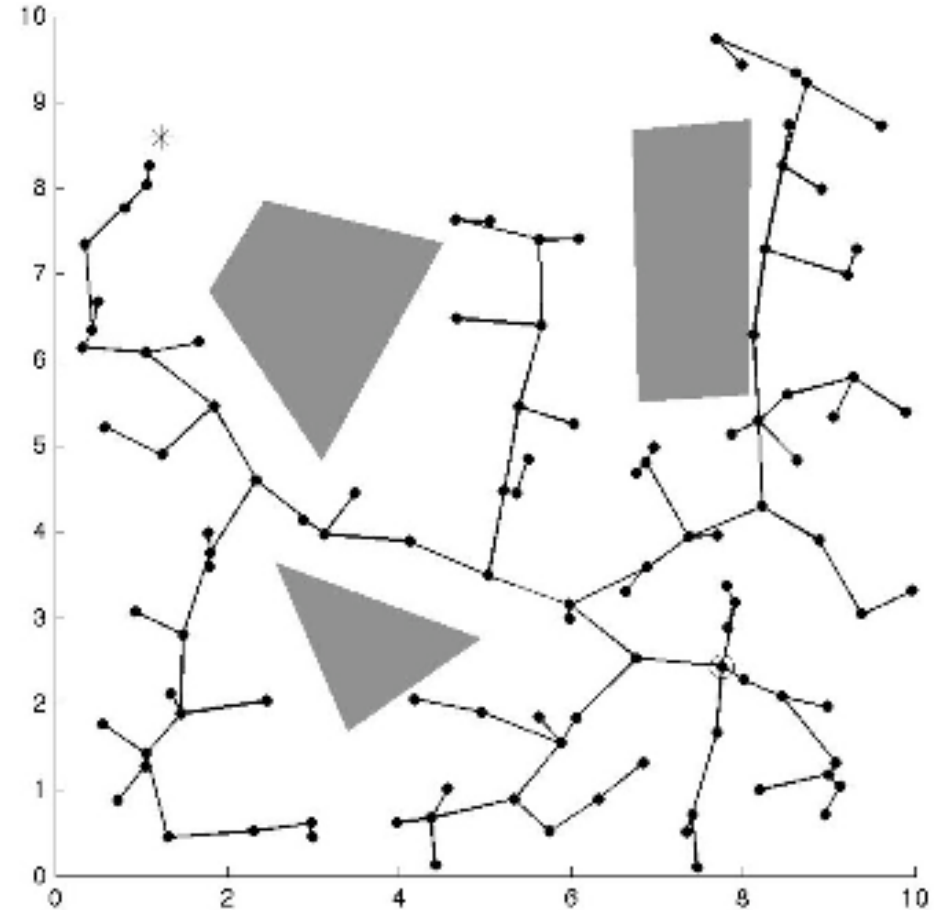
Probabilistic Path Planning: PRM

CS 3630



Recap: Rapidly-Exploring Random Tree (RRT)

- Searches for a path from the initial configuration to the goal configuration by expanding a search tree
- For each step,
 - The algorithm samples a target configuration and expands the tree towards it.
 - The sample can either be a random configuration or the goal configuration itself, depends on the probability value defined by the user.



General Types of approaches

Sampling-based methods typically fall into two categories:

	Single Query to a Given Goal	
Steps	Construct roadmap as you search for the goal	
Canonical Algorithm	Rapidly Exploring Random Tree (RRT)	
Strengths	No pre-processing Highly adaptive to dynamic environments	
Drawbacks	No memory of past searches, time-consuming search process	

General Types of approaches

Sampling-based methods typically fall into two categories:

	Single Query to a Given Goal	Multiple Queries to the Same Goal
Steps	Construct roadmap as you search for the goal	1. Construct roadmap 2. Search for the goal
Canonical Algorithm	Rapidly Exploring Random Tree (RRT)	Probabilistic Roadmap (PRM)
Strengths	No pre-processing Highly adaptive to dynamic environments	Fast search times
Drawbacks	No memory of past searches, time-consuming search process	Changes in the environment require re-processing the roadmap

General Types of approaches



	Single Query to a Given Goal	Multiple Queries to the Same Goal
Steps	Construct roadmap as you search for the goal	<ol style="list-style-type: none">1. Construct roadmap2. Search for the goal
Canonical Algorithm	Rapidly Exploring Random Tree (RRT)	Probabilistic Roadmap (PRM)
Strengths	No pre-processing Highly adaptive to dynamic environments	Fast search times
Drawbacks	No memory of past searches, time-consuming search process	Changes in the environment require re-processing the roadmap

Probabilistic Road Map (PRM)

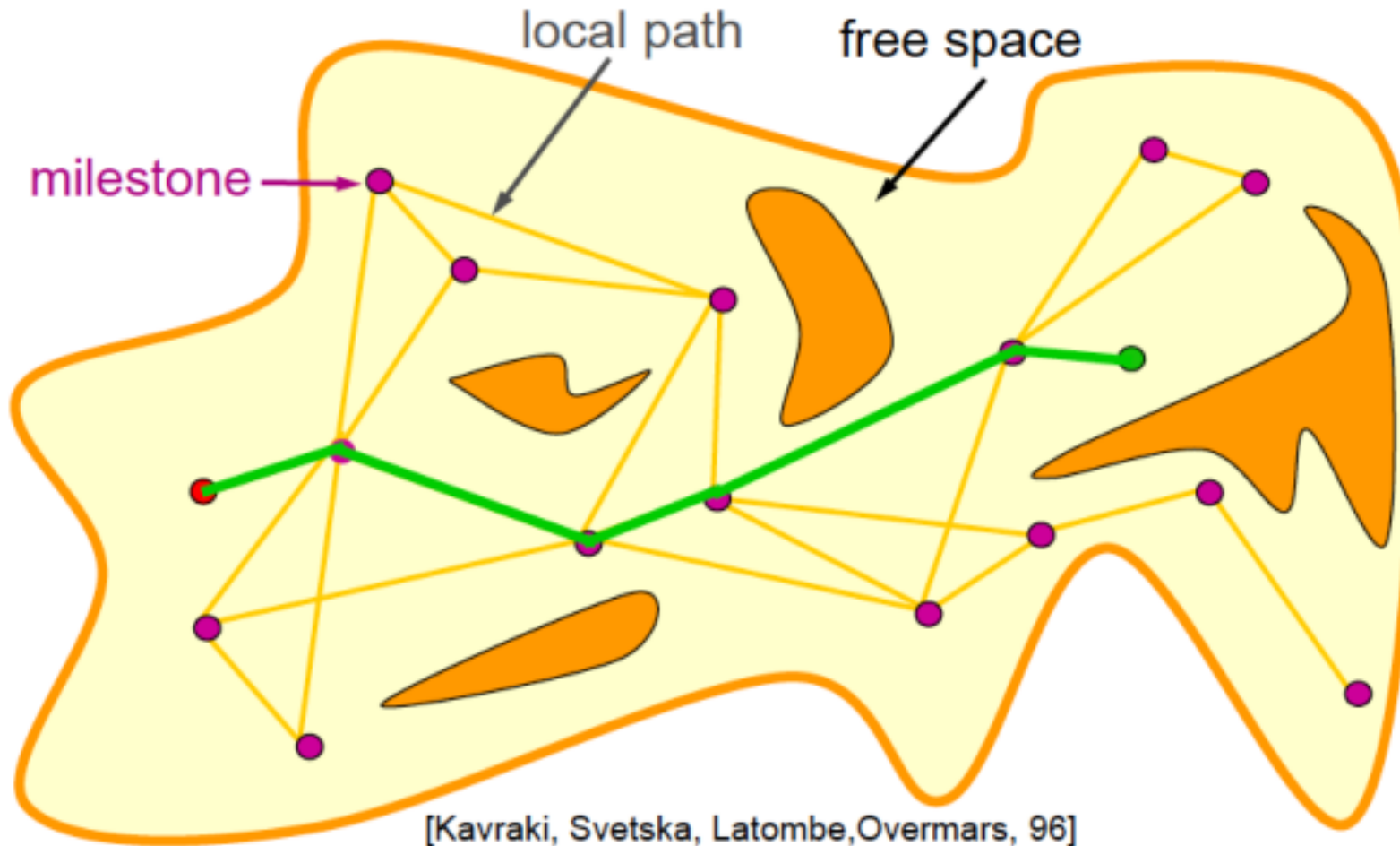
- Probabilistic Roadmap methods proceed in two phases:

1. Preprocessing Phase – to construct the roadmap G

2. Query Phase – to search given q_{init} and q_{goal}

The roadmap is an *undirected* graph $G = (N, E)$. The nodes in N are a set of configurations of the robot chosen over C -free. The edges in E correspond to feasible straight-line paths.

Probabilistic Road Map (PRM): an example



PRM Assumptions

- **Core assumption:** static obstacles
 - We can precompute a graph and use it repeatedly for many queries
- Examples:
 - Navigation in static environments
 - Robot manipulator in a workcell
- Advantages:
 - Amortize the cost of planning over many queries
 - Probabilistically complete



PRM with Uniform sampling

Input: geometry of the moving object & obstacles

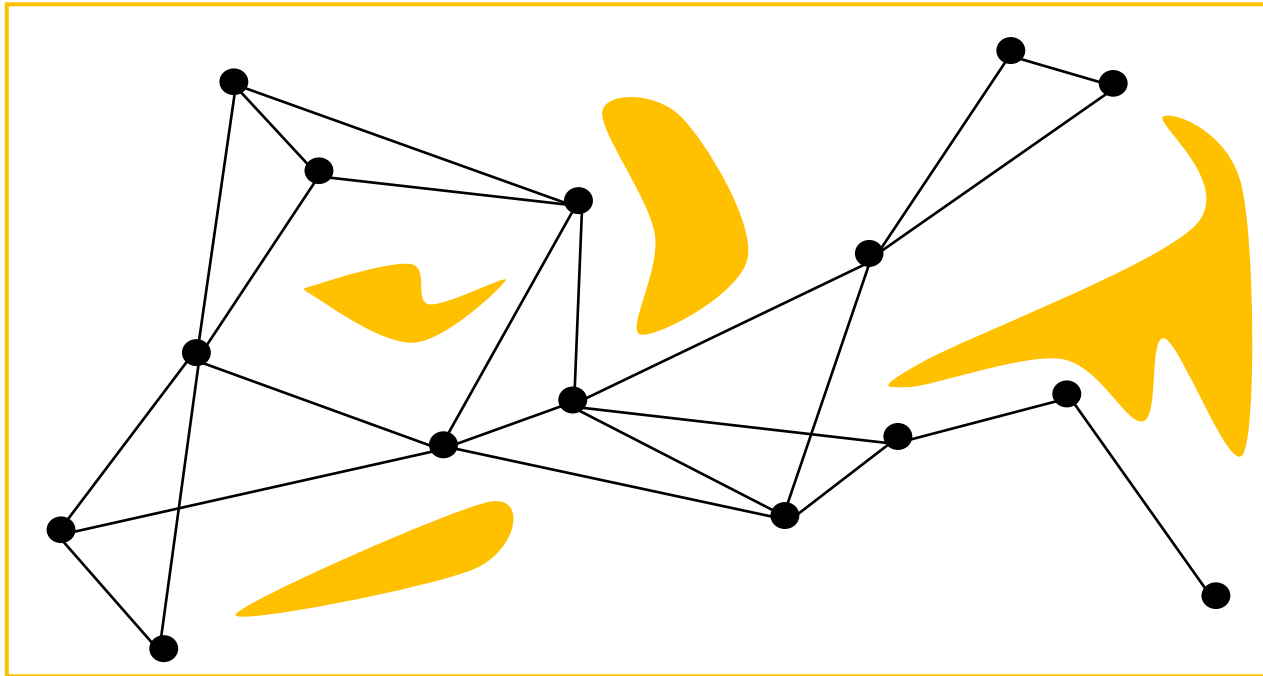
Output: roadmap $G = (V, E)$

```
1:  $V \leftarrow \emptyset$  and  $E \leftarrow \emptyset$ .
2: repeat
3:    $q \leftarrow$  a configuration sampled uniformly at random from  $C$ .
4:   if CLEAR( $q$ ) then
5:     Add  $q$  to  $V$ .
6:      $N_q \leftarrow$  a set of nodes in  $V$  that are close to  $q$ .
6:     for each  $q' \in N_q$ , in order of increasing  $d(q, q')$ 
7:       if LINK( $q', q$ ) then
8:         Add an edge between  $q$  and  $q'$  to  $E$ .
```

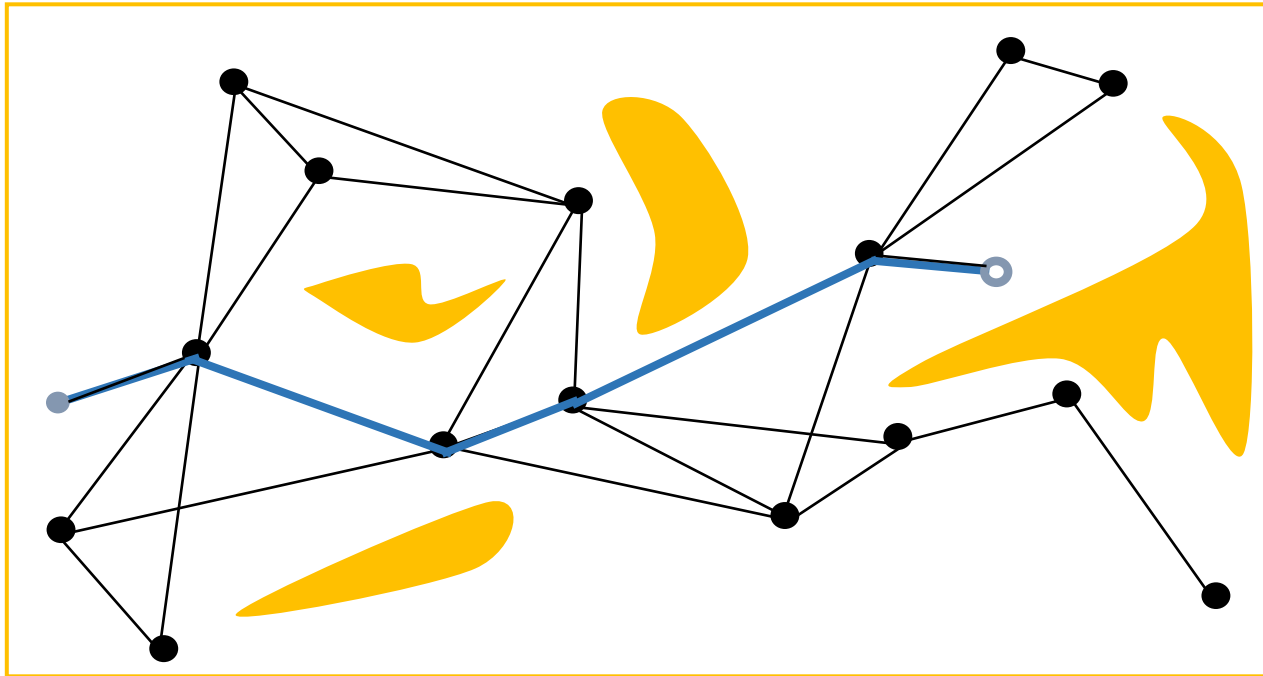
Probabilistic Road Map (PRM)



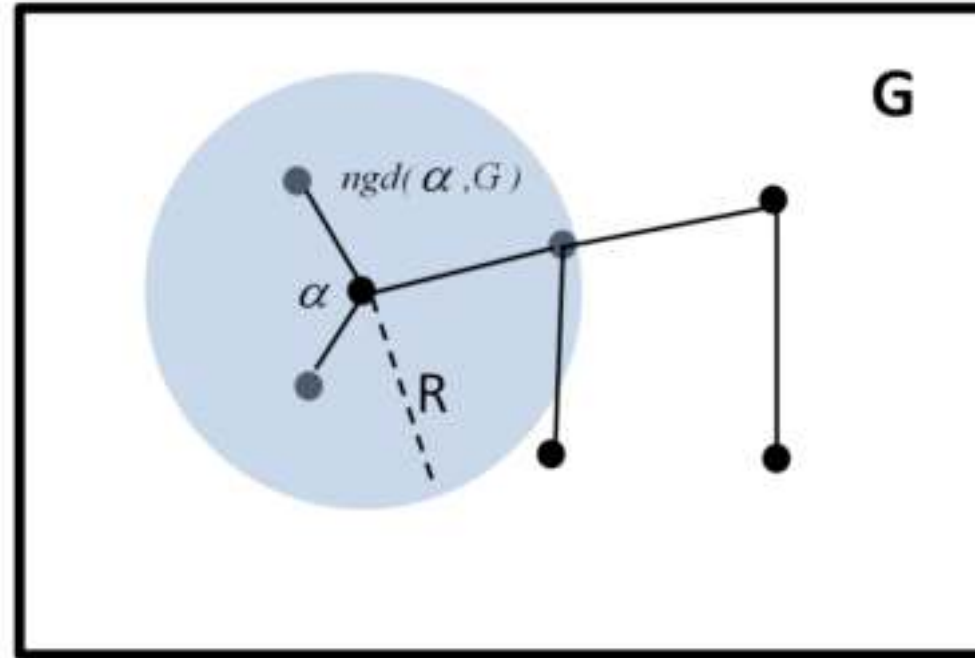
Probabilistic Road Map (PRM) – Phase 1



Probabilistic Road Map (PRM) – Phase 2



Connecting a new sample

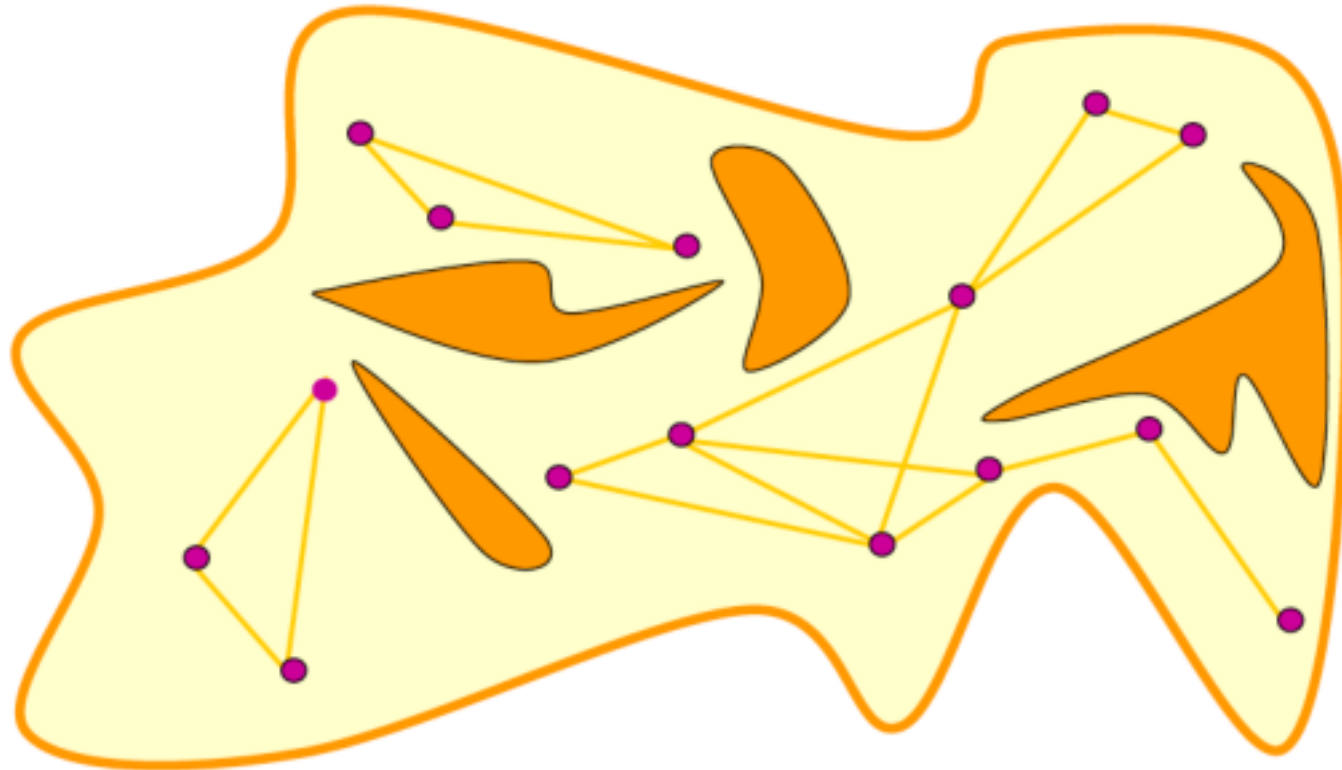


Probabilistic Road Map (PRM)

- Samples can be generated either randomly or using other “clever” methods. The biggest advantage for random sampling is that there is no bias.
- The distance of neighbor R is the number that tells you how far you think you can connect to configurations by straight lines.
- **Advantage:** it works for any q_{init} and q_{goal} , repeatedly
- **Disadvantage:** it spends a lot of time on preprocessing.

A Potential Challenge

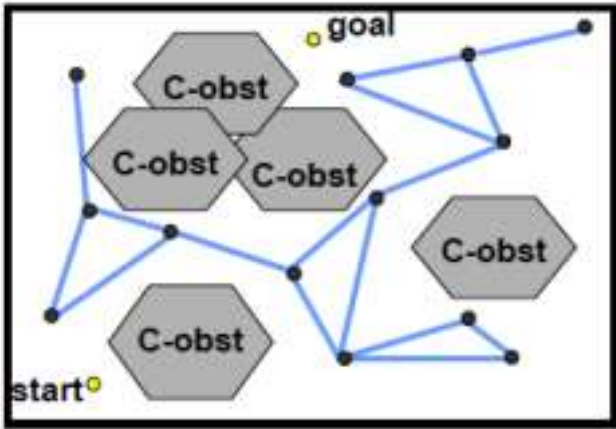
Many small, unconnected graph clusters



PRM variants

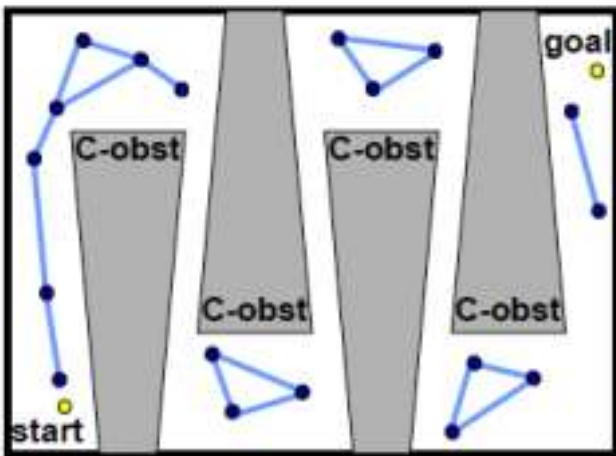
- There are (very) many...
- **Lazy PRM:**
 - Create a dense PRM without ANY collision checking
 - When you have q_{init} and q_{goal} :
 - Find $q_{init} \rightarrow s_1 \rightarrow s_2 \rightarrow q_{goal}$ (a feasible path)
 - Check only the edges in the returned path for collisions, remove any edges with collisions.

PRM summary



- Pros:

- Probabilistically complete
- Fast queries given a lot of preprocessing
- Many success stories



- Cons:





- Struggle with narrow passages (variants developed to help address this)
- No optimality or completeness

General Types of approaches

Sampling-based methods typically fall into two categories:

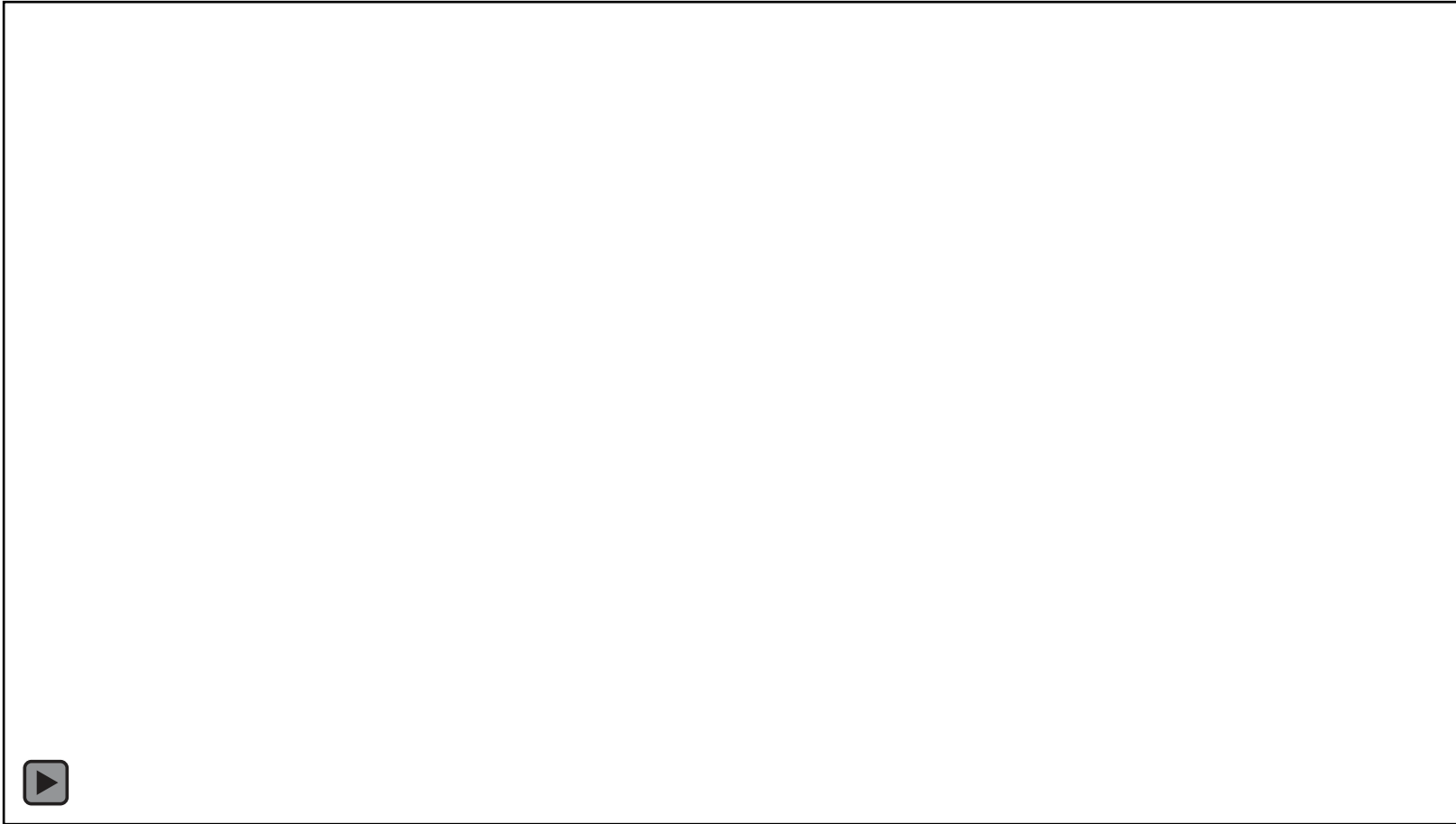
	Single Query to a Given Goal	Multiple Queries to the Same Goal
Steps	Construct roadmap as you search for the goal	1. Construct roadmap 2. Search for the goal
Canonical Algorithm	Rapidly Exploring Random Tree (RRT)	Probabilistic Roadmap (PRM)
Strengths	No pre-processing Highly adaptive to dynamic environments	Fast search times
Drawbacks	No memory of past searches, time-consuming search process	Changes in the environment require re-processing the roadmap

Comparison of RRT and PRM

		Run Time [s]	Path Length	Smoothness	Clearance
Standard 	RRT	6.23	1353	33	10
	PRM	1.04	1238	35	10
Cluttered 	RRT	5.33	1944	35	4
	PRM	1.93	2220	59	4
Narrow Passage 	RRT	8.94	4292	29	3
	PRM	2.1	3315	70	3
Spiral 	RRT	25.81	7017	46	0
	PRM	53.21	4424	54	0

Examples:

1. RRT
2. RRT March
3. RRT*
4. PRM



Summary

- Both RRT and PRM are examples of **sampling-based algorithms** that are **probabilistically complete**
- **Definition:** A path planner is *probabilistically complete* if, given a solvable problem, the probability that the planner solves the problem goes to 1 as time goes to infinity.

Is all this planning necessary? Can we get away
with a simpler method?

Optimal Paths Using the Value Function

One approach to path planning:

- Place high reward at the goal configuration.
- Place negative reward along obstacle edges.
- Compute the value function.
- Follow the gradient of the value function from the current configuration to the goal.

We've already seen a version of this for our logistics robot, but things are a bit different now:

- For path planning, we often consider deterministic actions (i.e., no need for expected values).
- Including uncertainty in the motion model leads to more robust plans.

Reminder about Value Iteration

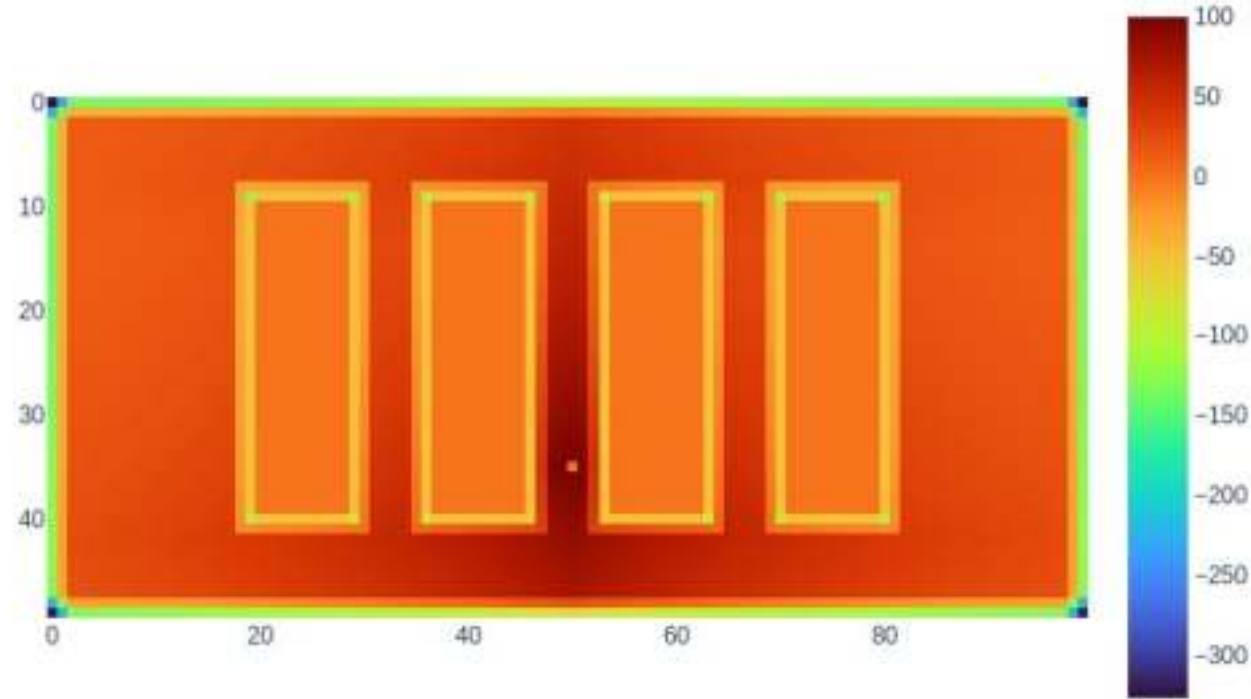
Recall the Value Iteration Equation:

$$V^{k+1}(x) \leftarrow \max_a \left\{ \bar{R}(x, a) + \gamma \sum_{x'} P(x'|x, a) V^k(x') \right\}$$

- $\bar{R}(x, a)$ is the expected reward for applying action a in state x .
 - $P(x'|x, a)$ is the motion model.
 - $V^k(x')$ is the approximation to the optimal value function at iteration k .
- For classical path planning, we ignore uncertainty, so that $P(x'|x, a) = 1$ for the deterministic outcome, and $P(x'|x, a) = 0$ for all other outcomes.
- If we add a bit of uncertainty, we increase robustness of the plan (e.g., decrease the probability that the robot might accidentally collide with an obstacle).

Value Iteration in the Warehouse

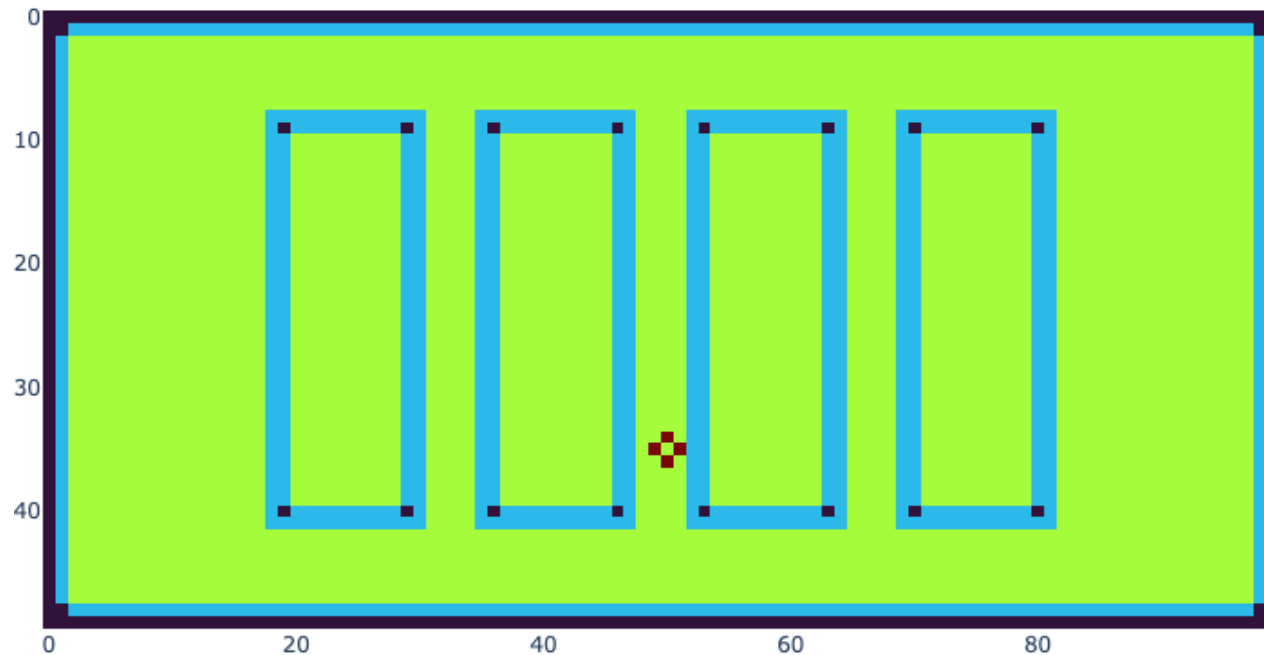
- In our warehouse example, we set the reward to 100 for the goal, and to -50 for obstacle edges, and used value iteration to compute the value function:



- To move from any point in the warehouse toward the goal, simply follow the gradient of the value function!

Value Iteration at Work

- *Value iteration is expensive.*
- At each iteration, we must compute N updates (one for each possible state).
- For the warehouse, we have $N = 50 \times 100 = 5000$.
- Many problems have much larger state spaces, which makes value iteration impractical (and possibly intractable).



➤ *At each iteration, every grid cell updates its estimate of the optimal value function.*

Path Planning with Artificial Potential Fields

With so many slides and ideas from so many people,
including:

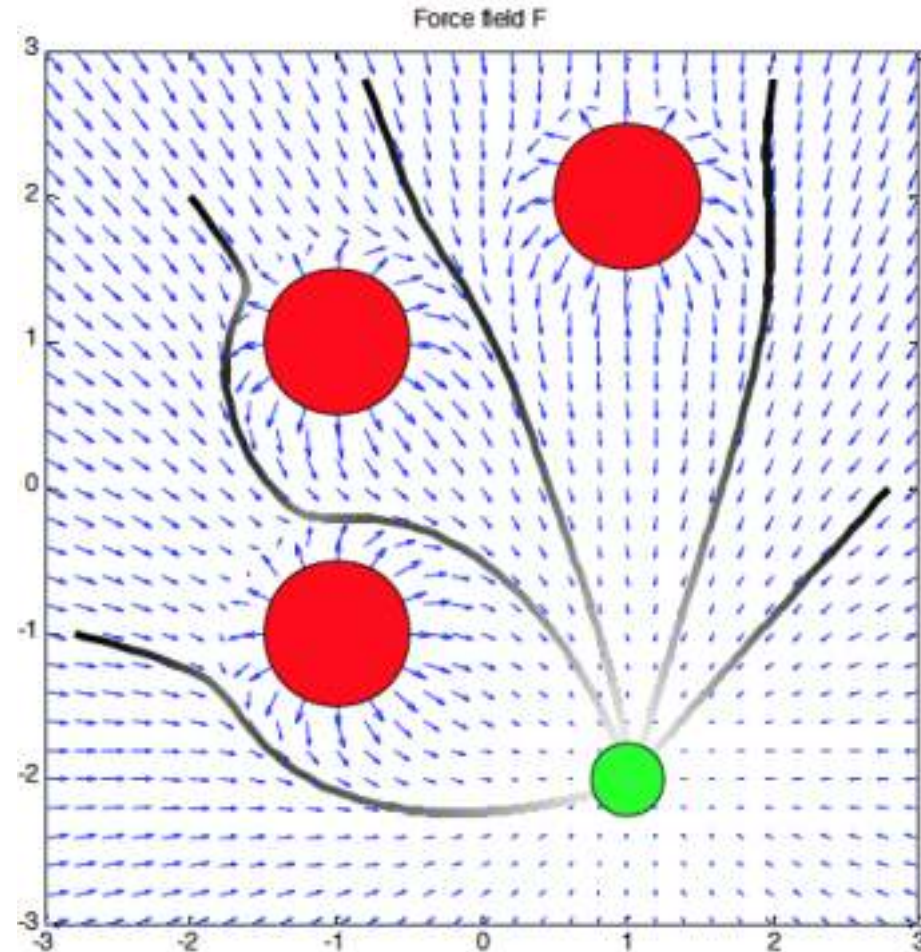
Howie Choset, Nancy Amato, David Hsu,
Sonia Chernova, Steve LaValle, James Kuffner,
Greg Hager, Ji Yeong Lee

Planning with Potential Fields

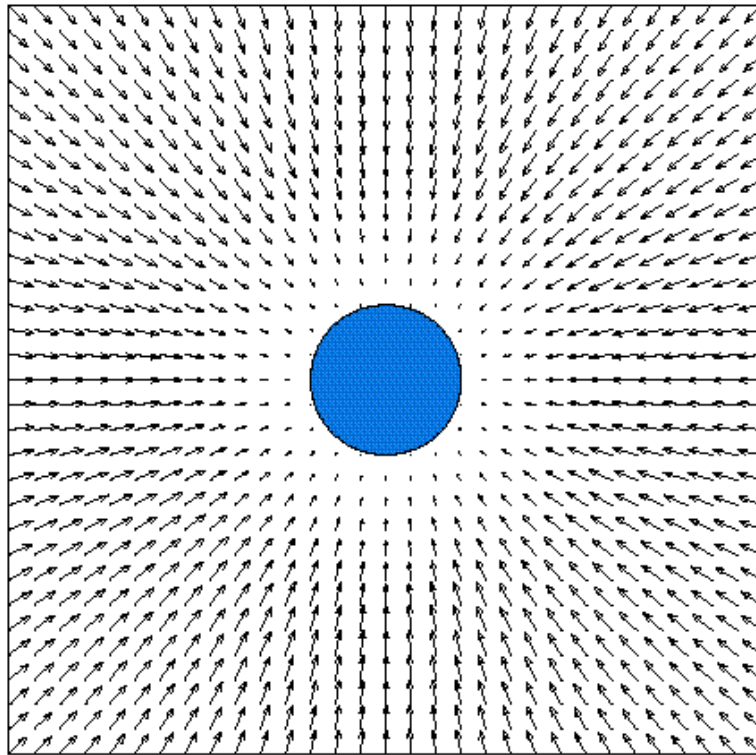
- ***Don't compute the value function at every cell in the workspace.***
- **Instead**, try to develop a nice, closed-form expression that gives similar information to the value function.
 - A single global minimum at the goal (use gradient descent to find the goal)
 - Infinite cost at the edges of obstacles (to prevent collisions).
- In practice, we won't be able to construct such functions, but we can often construct pretty good approximations that work well for local path planning (i.e., when the distance to the goal is not so great).
- Artificial Potential Fields are a popular approach.

Potential fields

- Consider the robot a particle in a potential field
- Goal serves as an attractor, generating potential field function U_{att}
- Obstacles repel the particle, each generating potential field function U_{rep}



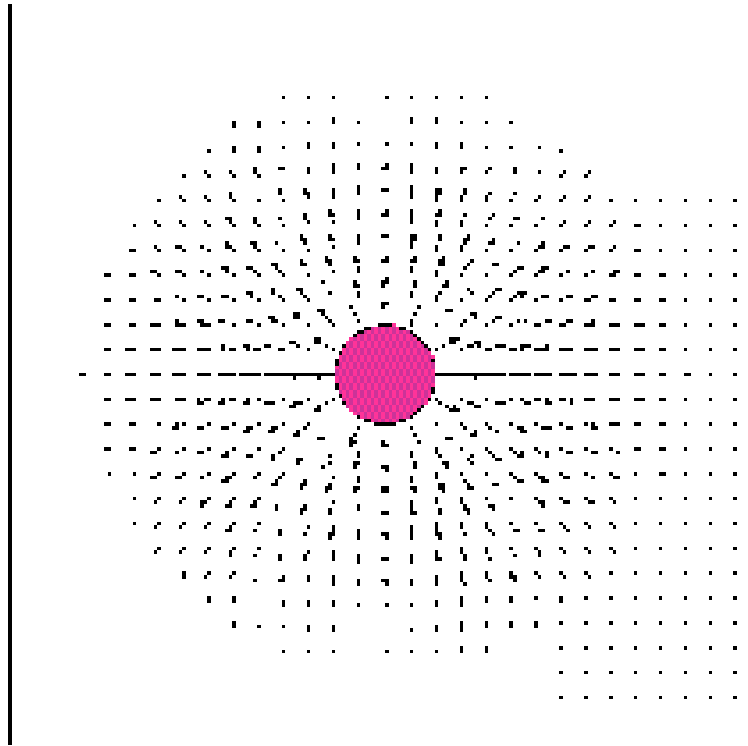
Attractive Potential Field



This vector field is defined by the negative gradient of the attractive potential function:

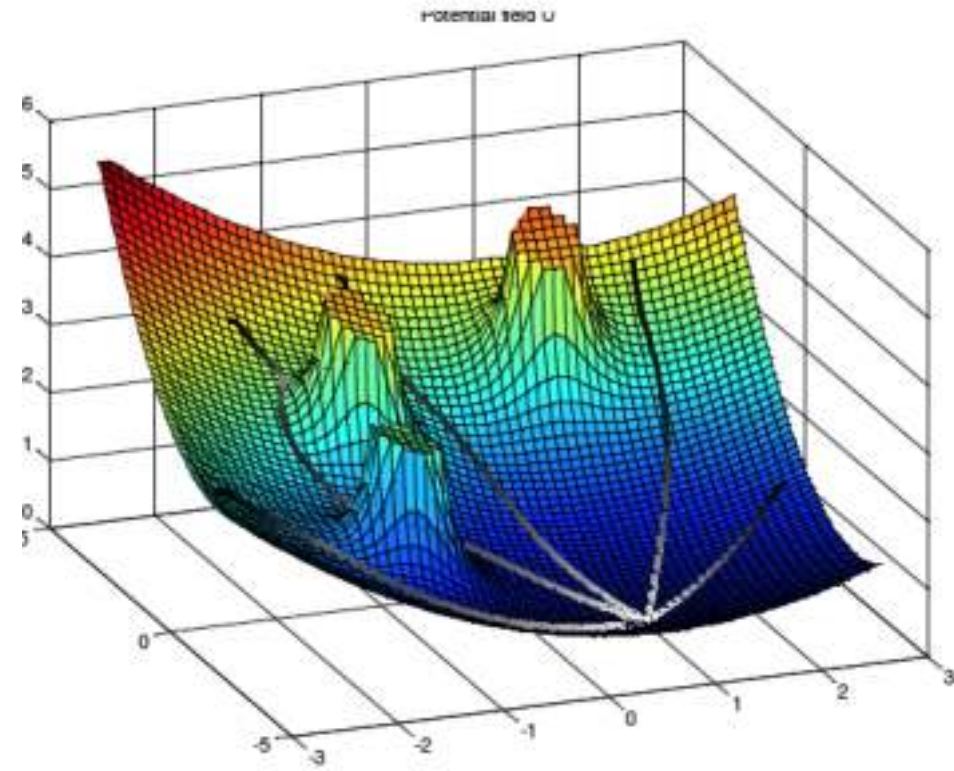
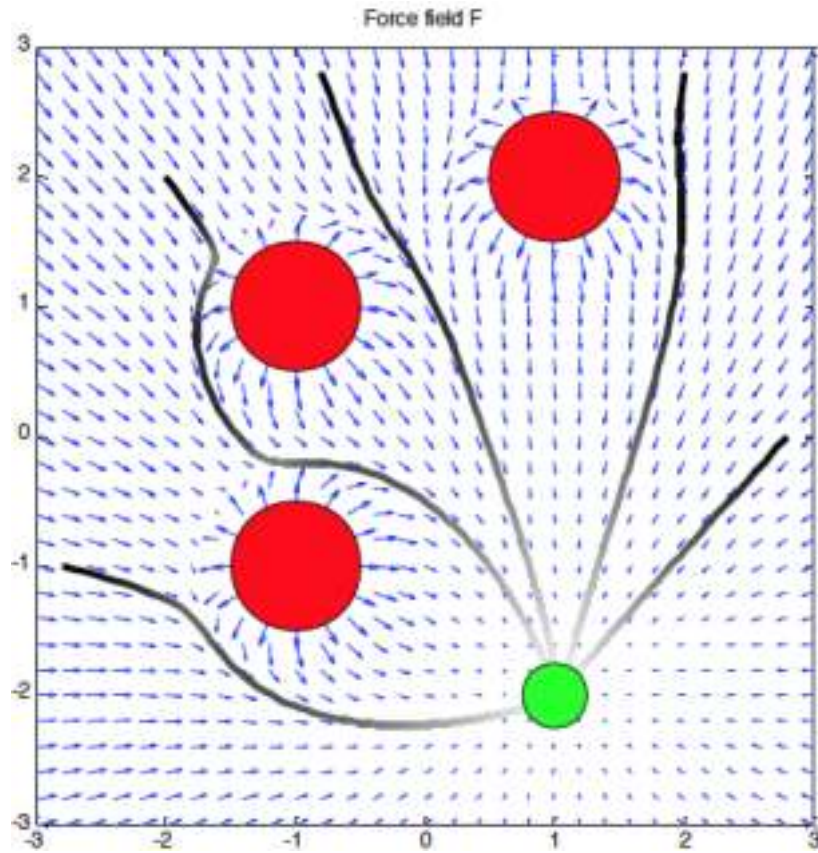
$$\mathbf{F} = -\nabla U_{attr}$$

Repulsive Potential Field



This vector field is defined by the negative gradient of the repulsive potential function:

$$\mathbf{F} = -\nabla U_{rep}$$



Force field is defined by the negative gradient of the potential function: $F = \nabla U$

Potential Fields

An ideal potential field

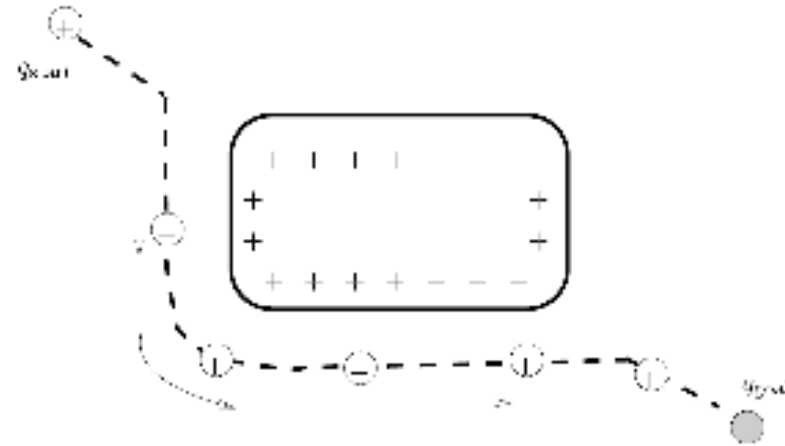
- has global minimum at the goal
- has no local minima
- grows to infinity near obstacles
- is smooth

On a real robot

- Sensor information is used to estimate the potential field gradient $\nabla U(q)$
 - Only need to compute the potential field near the robot, not over the entire map
 - Compute individual components (obstacle 1, obstacle 2, goal, etc) separately and add them together.
- Direction of motion determined by the gradient.

Useful Analogies

- A really simple idea:
 - Suppose the goal is a point $q_{goal} \in \mathbb{R}^2$
 - Suppose the robot is a point $q \in \mathbb{R}^2$
 - Think of a “spring” drawing the robot toward the goal while pushing away from obstacles.
 - Can also think of like and opposite charges



The General Idea

- Both the bowl and the spring are ways of storing potential *energy*
- The robot moves to a lower energy configuration
- A *potential function* is a function $U: \mathbb{R}^m \rightarrow \mathbb{R}$
- Energy is minimized by following the negative *gradient* of the potential energy function:

$$\nabla U(q) = \left[\frac{\partial U}{\partial q_1} \quad \cdots \quad \frac{\partial U}{\partial q_m} \right]^T$$

- We can now think of a *vector field* over the space of all q 's ...
 - at every point in time, the robot looks at the vector at its current configuration, and goes in its negative direction

Attractive/Repulsive Potential Field

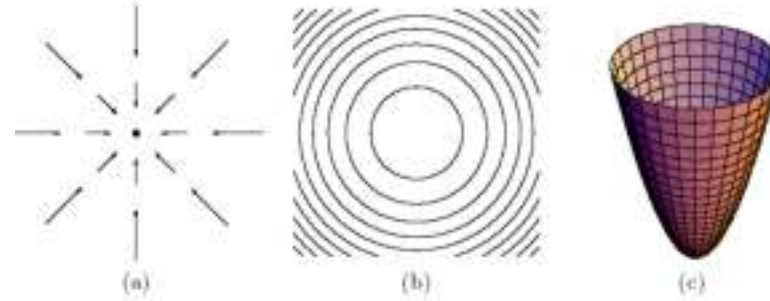
$$U(q) = U_{att}(q) + U_{rep}(q)$$

- $U_{att}(q)$ is the “attractive” potential --- move to the goal
 - Should achieve minimum value at the goal, and should increase as distance from the goal increases.
- $U_{rep}(q)$ is the “repulsive” potential --- avoid obstacles
 - Should never allow collision
 - $U_{rep}(q) \rightarrow \infty$ as $q \rightarrow QO$

Artificial Potential Field Methods: Attractive Potential

Quadratic Potential

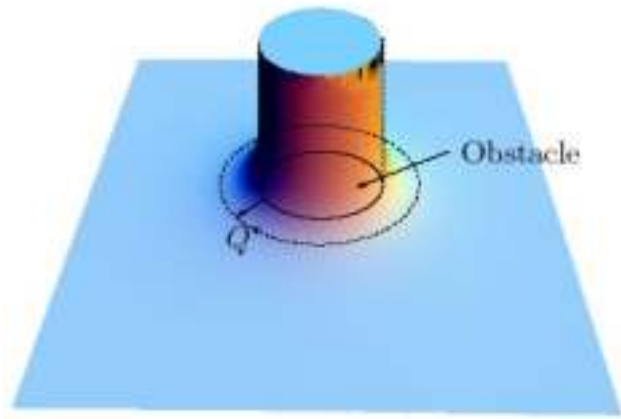
$$U_{\text{att}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{goal}}),$$



$$\begin{aligned} F_{\text{att}}(q) &= \nabla U_{\text{att}}(q) = \nabla \left(\frac{1}{2}\zeta d^2(q, q_{\text{goal}}) \right), \\ &= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}), \\ &= \zeta(q - q_{\text{goal}}), \end{aligned}$$

No need to worry about the details of how to calculate the gradients in this class!

The Repulsive Potential



Q^* is the “distance of influence” of the obstacle

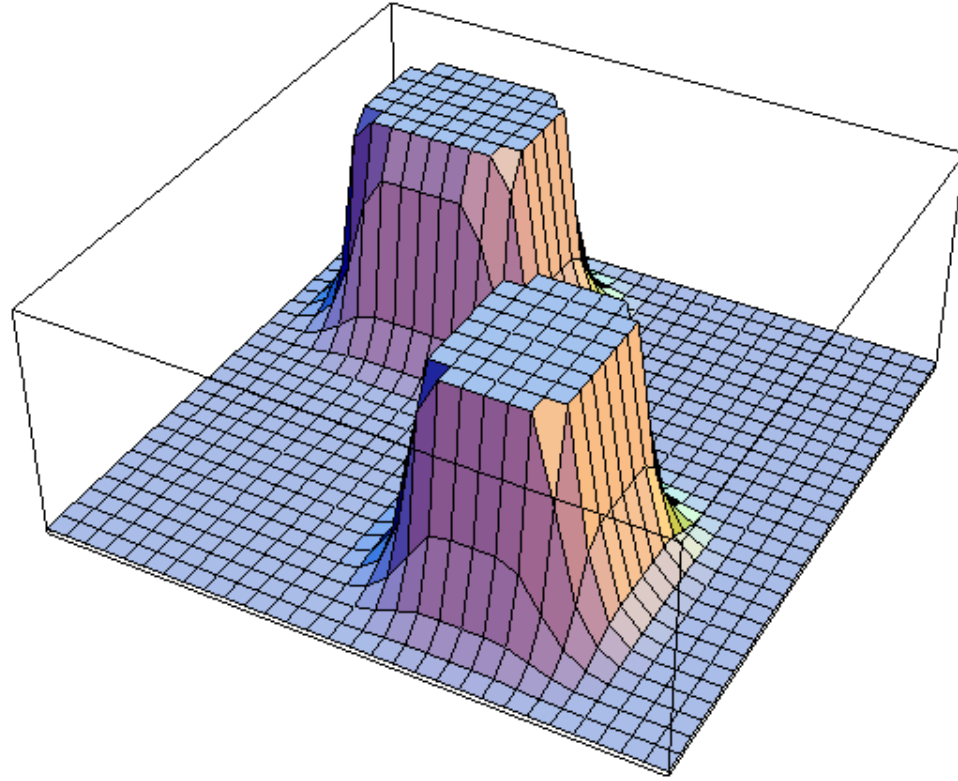
$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

whose gradient is

$$\nabla U_{\text{rep}}(q) = \begin{cases} \eta \left(\frac{1}{Q^*} - \frac{1}{D(q)} \right) \frac{1}{D^2(q)} \nabla D(q), & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

No need to worry about the details of how to calculate the gradients in this class!

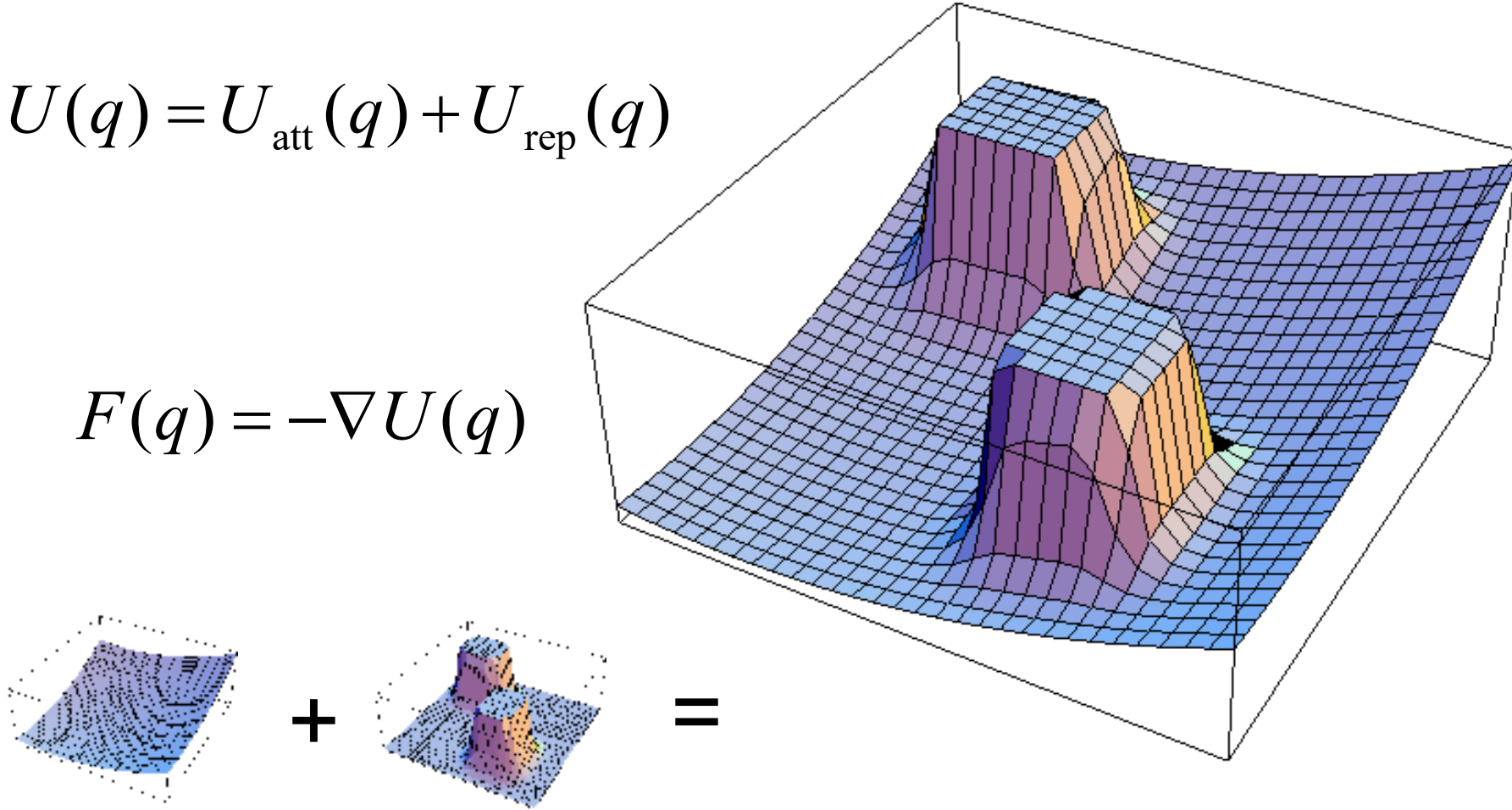
Repulsive Potential



Total Potential Function

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

$$F(q) = -\nabla U(q)$$

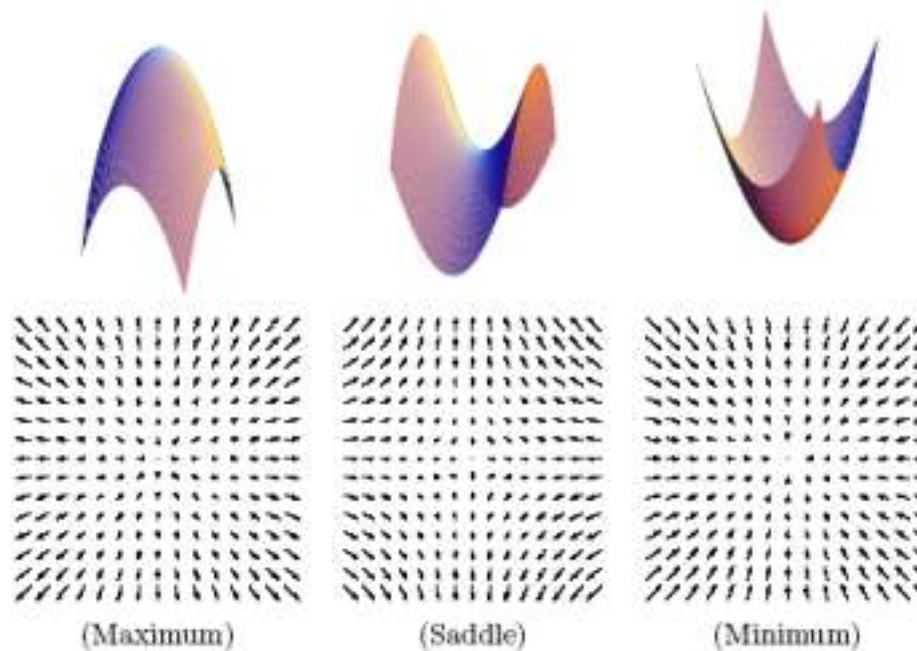


Gradient Descent

- A simple way to get to the bottom of a potential

$$\dot{q}(t) = -\nabla U(q(t))$$

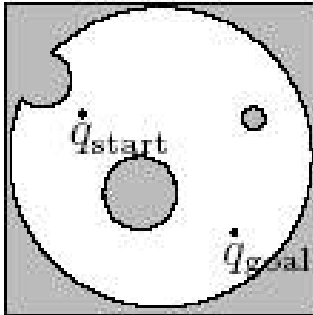
- A *critical point* is a point x s.t. $\nabla U(x) = 0$
 - Equation is stationary at a critical point
 - Max, min, saddle
 - Stability?



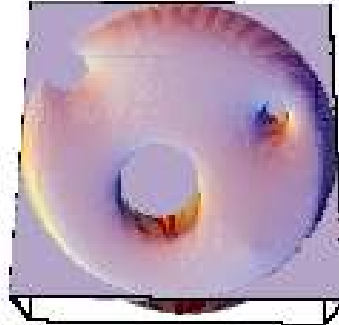
Gradient Descent

Gradient Descent:

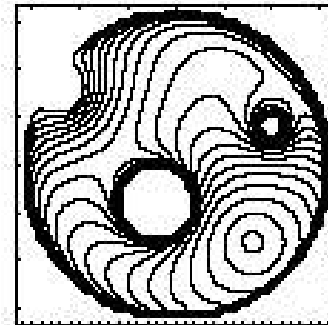
- $q(0) = q_{\text{start}}$
- $i = 0$
- while $\nabla U(q(i)) \neq 0$ do
 - $q(i+1) = q(i) - \alpha(i) \nabla U(q(i))$
 - $i = i+1$



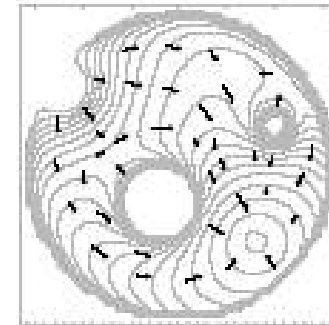
(a)



(b)



(c)



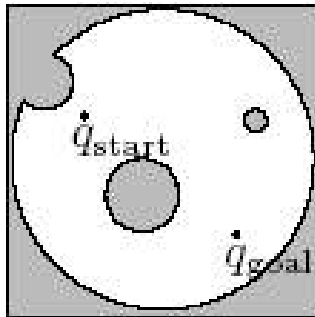
(d)

Gradient Descent

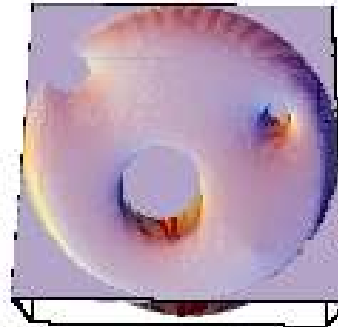
Gradient Descent:

- $q(0) = q_{\text{start}}$
- $i = 0$
- **while** $\| \nabla U(q(i)) \| > \epsilon$ **do**
 - $q(i+1) = q(i) - \alpha(i) \nabla U(q(i))$
 - $i = i+1$

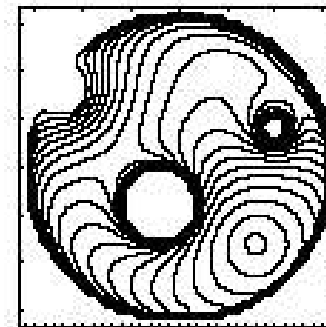
Stop when we get “close enough” to the goal



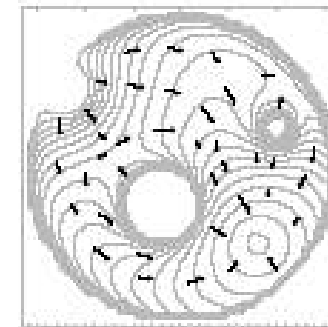
(a)



(b)



(c)



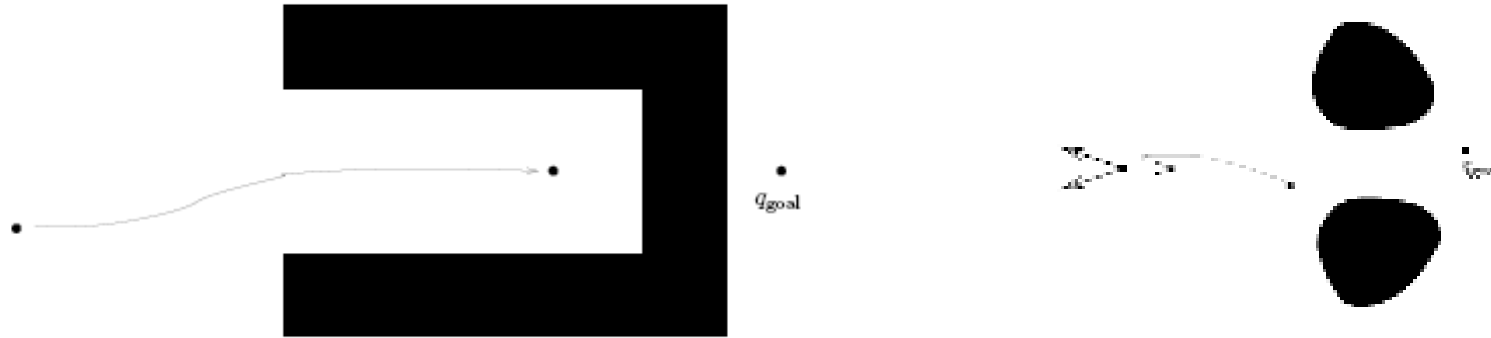
(d)

Problems with Potential Fields

- Local minima
 - Attractive and repulsive forces can balance, so robot makes no progress.
 - Common with closely spaced obstacles, and dead ends.
- Unstable oscillation
 - The dynamics of the robot/environment system can become unstable.
 - High speeds, narrow corridors, sudden changes.

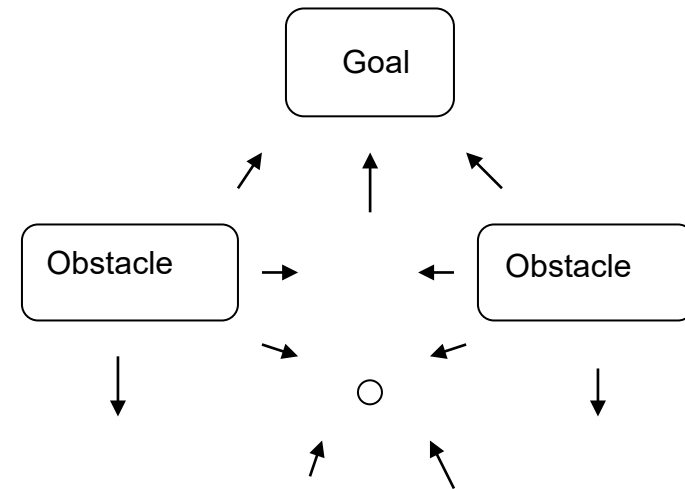
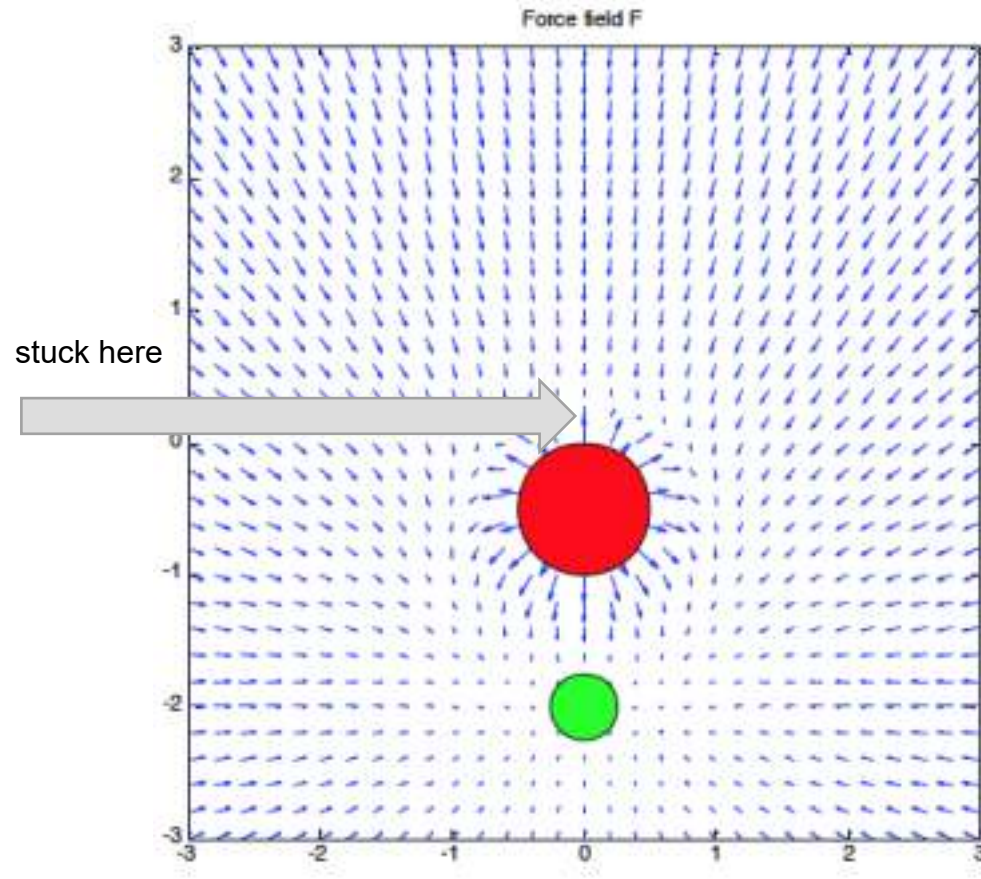
Potential Functions Question

- How do we know that we have only a single (global) minimum



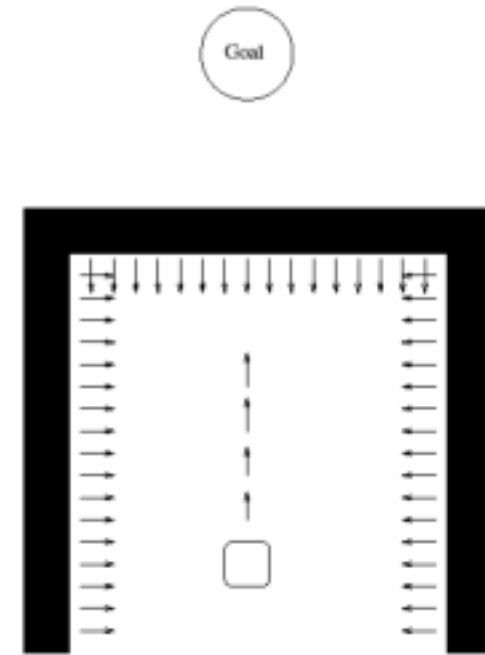
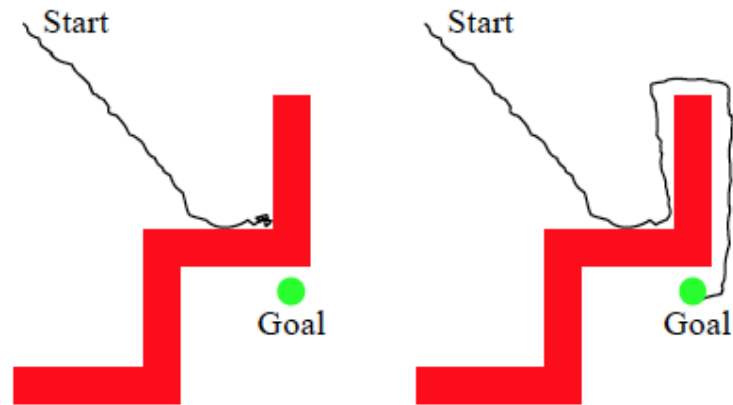
- We have two choices:
 - not guaranteed to be a global minimum: do something other than gradient descent (what?)
 - make sure only one global minimum (a navigation function, which we'll see later).

Local Minimum Problem



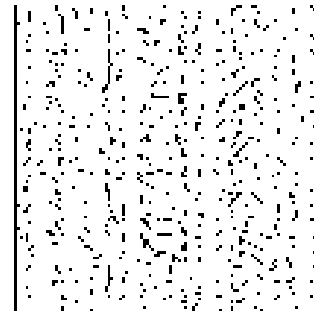
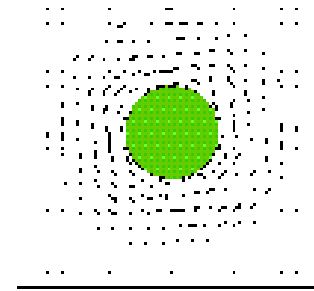
Box Canyon Problem

- Problem: local minimum
- Solution:
 - add *AvoidPast* behavior which sets up repelling potential fields for each location that the robot has visited
 - more recently visited areas have more repelling force

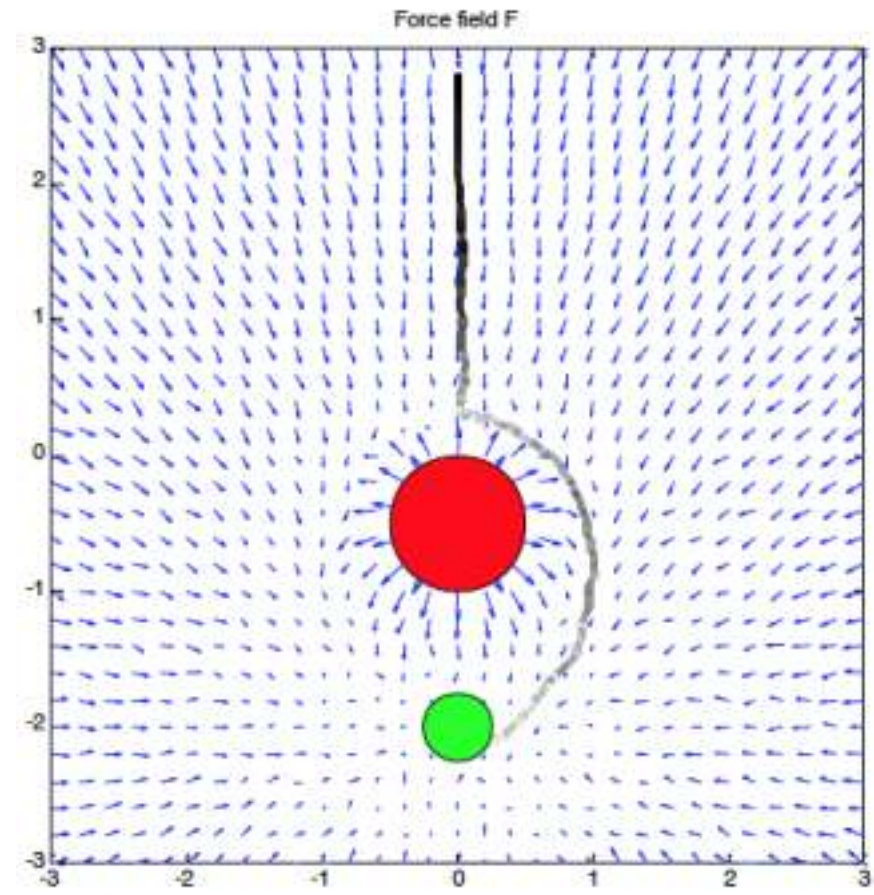


Rotational and Random Fields

- Adding a rotational field around obstacles
 - Breaks symmetry
 - Avoids some local minima
 - Guides robot around groups of obstacles
- A random field gets the robot unstuck.
 - Avoids some local minima.



Local minimum eliminated by
adding some random noise



Summary

- Potential fields represent a simple, *reactive* technique for controlling movement
- *Can* provide efficient solution to relatively simple tasks, but provide no guarantees for complex domains
- Motion planning still needed for more complex applications

Can we combine elements of long-distance planning and reactive behavior?

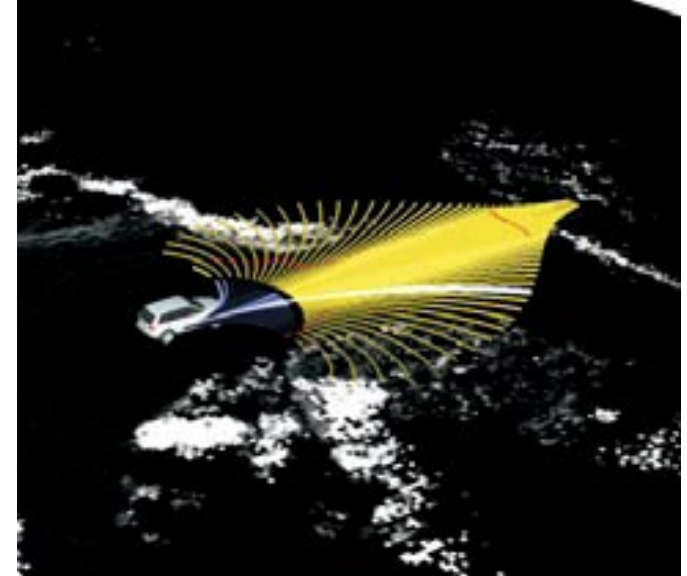
Yes! In fact, this is critical for robust behavior in mobile robots and self-driving cars.

One example... driving with “tentacles”

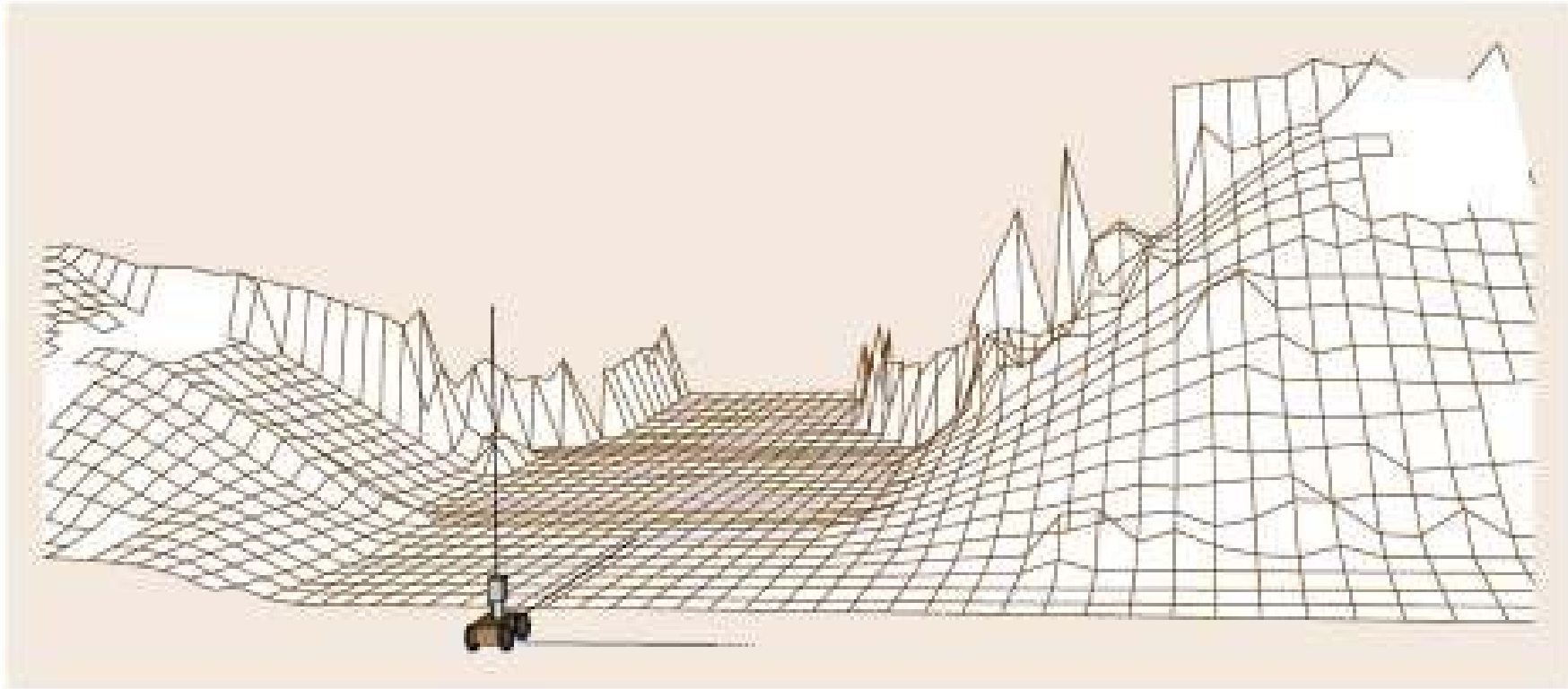


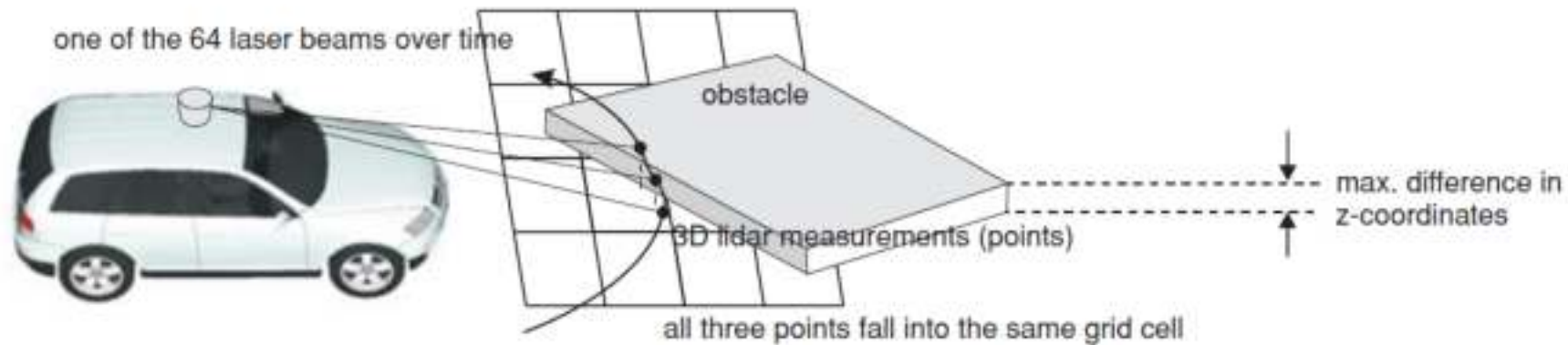
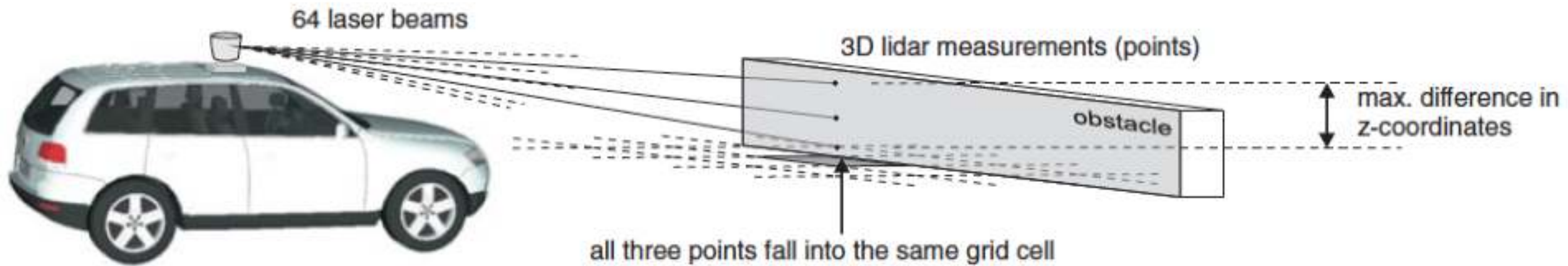
Approach demonstrated in the European Land Robot Trial and the DARPA Urban Grand Challenge.

“At the core of the method is a set of “tentacles” that represent precalculated trajectories defined in the ego-centered coordinate space of the vehicle. Similar to an insect's antennae or feelers, they fan out with different curvatures discretizing the basic driving options of the vehicle.”



$2\frac{1}{2}$ D Occupancy Grid (Elevation Map)

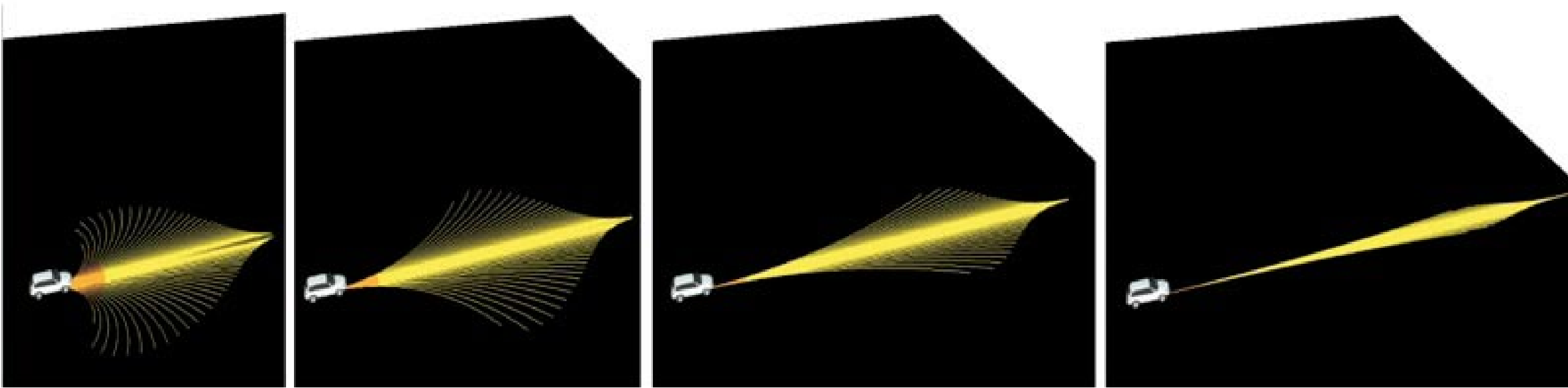




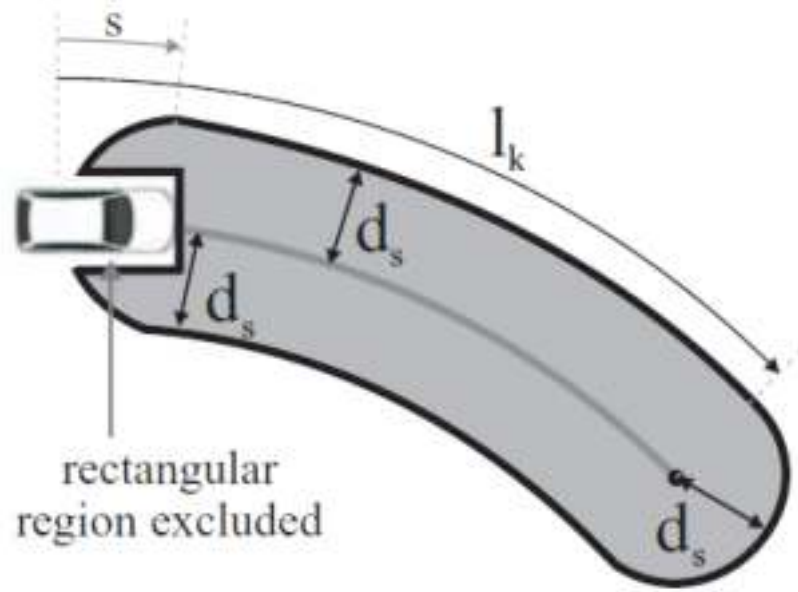
Occupancy grid value is computed to be the maximum difference of z coordinates of points in 3D space falling into that grid cell.

- Laser running at 10Hz, 100,000 3D measurements per cycle.
- No history is used in the occupancy grid data

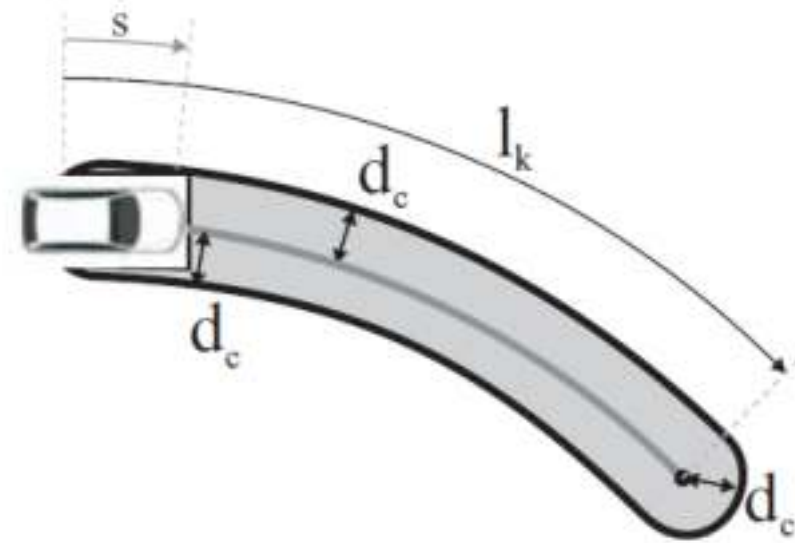
Tentacles



There are 16 predefined sets of tentacles, 4 of which are shown above. The algorithm selects the appropriate set to use based on the velocity of the vehicle, from slowest (left) to fastest (right). This enables the vehicle to avoid dangerous turns at high speed.



(a) Support area



(b) Classification area

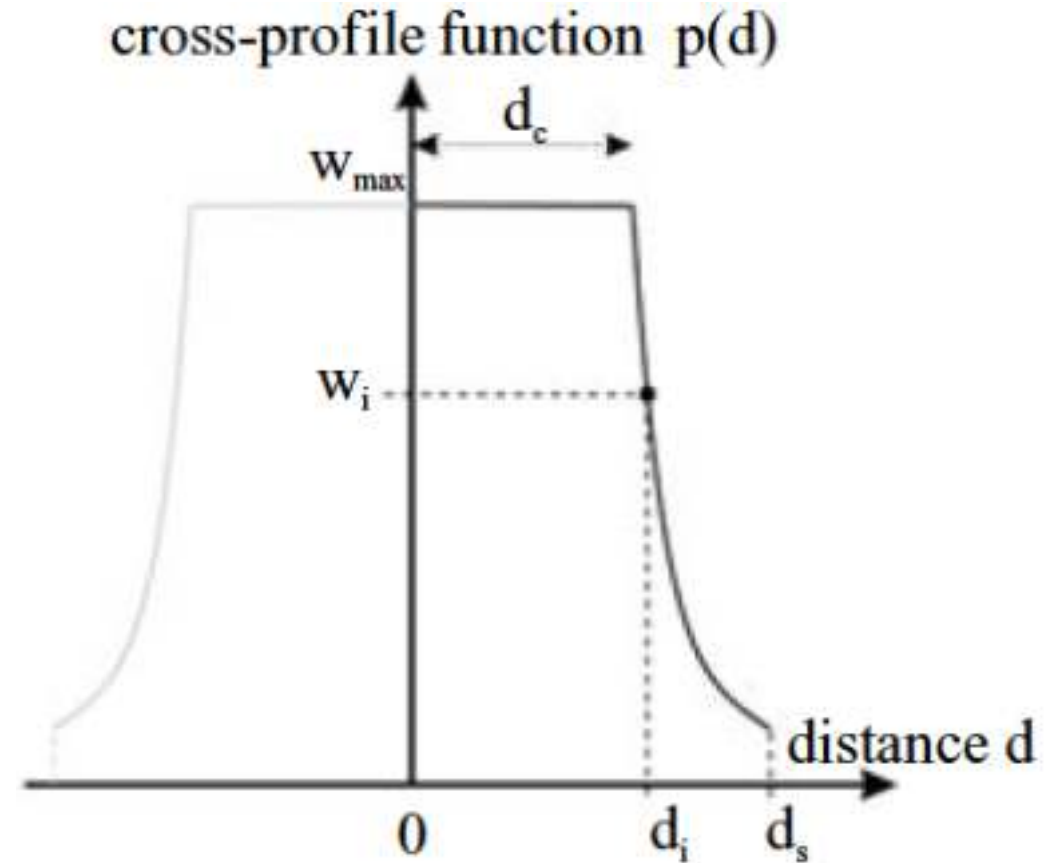
(a) The support area covers all cells within a distance d_s of the tentacle.

(b) The classification area is a subset of the support area covering all cells within a distance $d_c < d_s$ of the tentacle.

The classification area **must** be free for the tentacle to be driveable. The support area is **preferred** to be free.

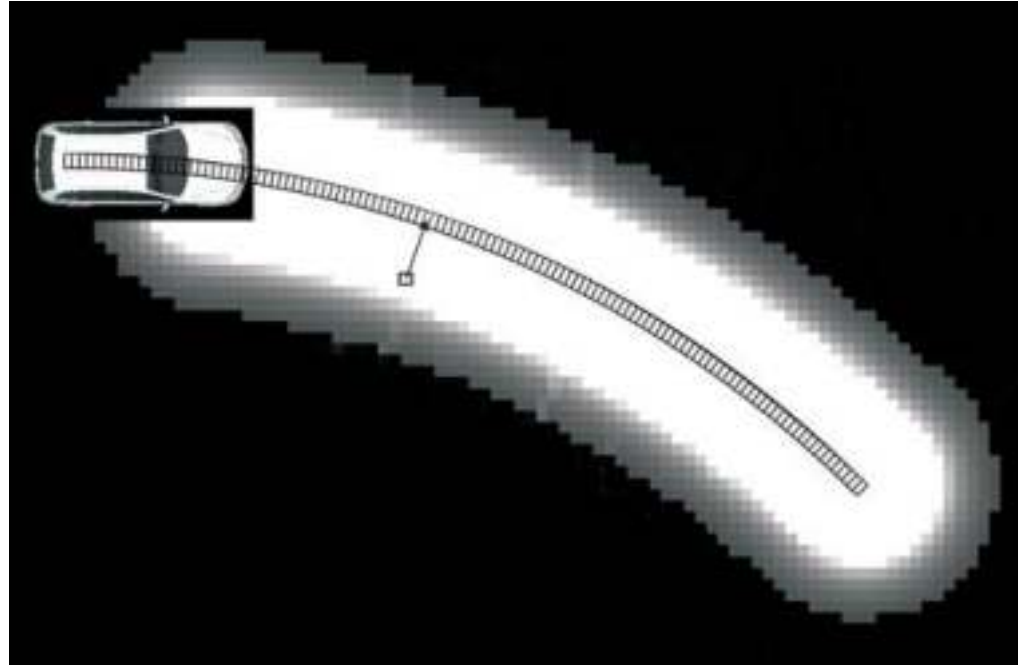
Determining clearance

$$p(d) = \begin{cases} w_{\max} & | d \leq d_c \\ \frac{w_{\max}}{\kappa + \frac{d - d_c}{\sigma}} & | d > d_c \end{cases},$$



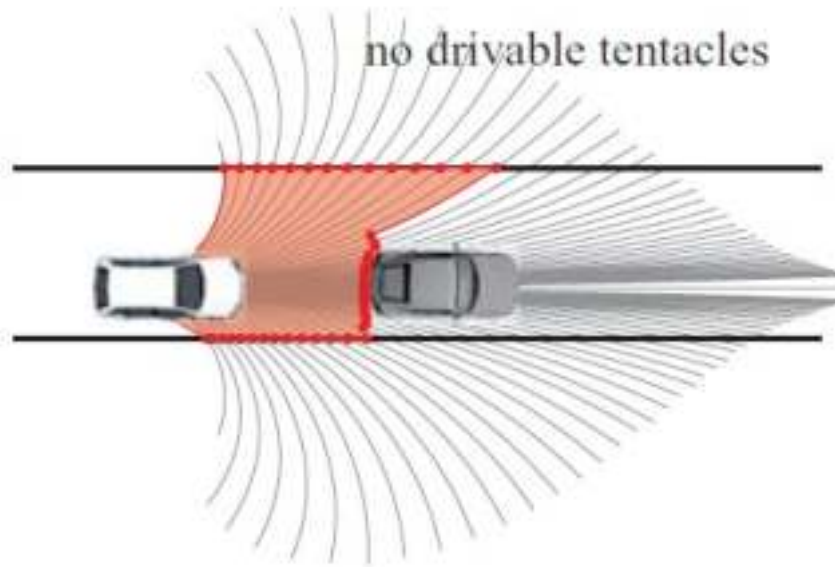
The cells of the support area are subject to different weights.

Checking for Obstacles

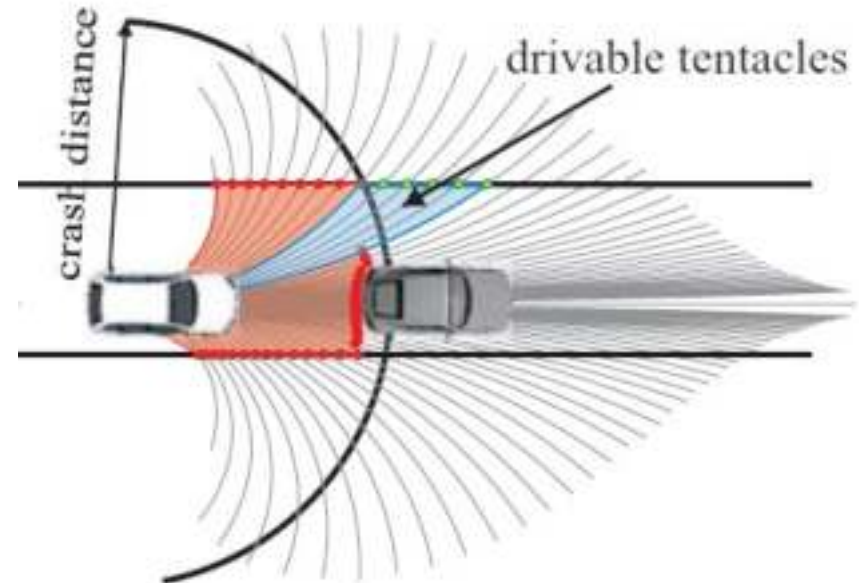


The algorithm looks at 5 consecutive cells at a time (a sliding window) and reports an obstacle if at least 2 of the cells are occupied in the occupancy grid.

The red points mark the locations along the tentacles where the vehicle would hit either the car or the road border. As can be seen, no tentacle is free of obstacles.

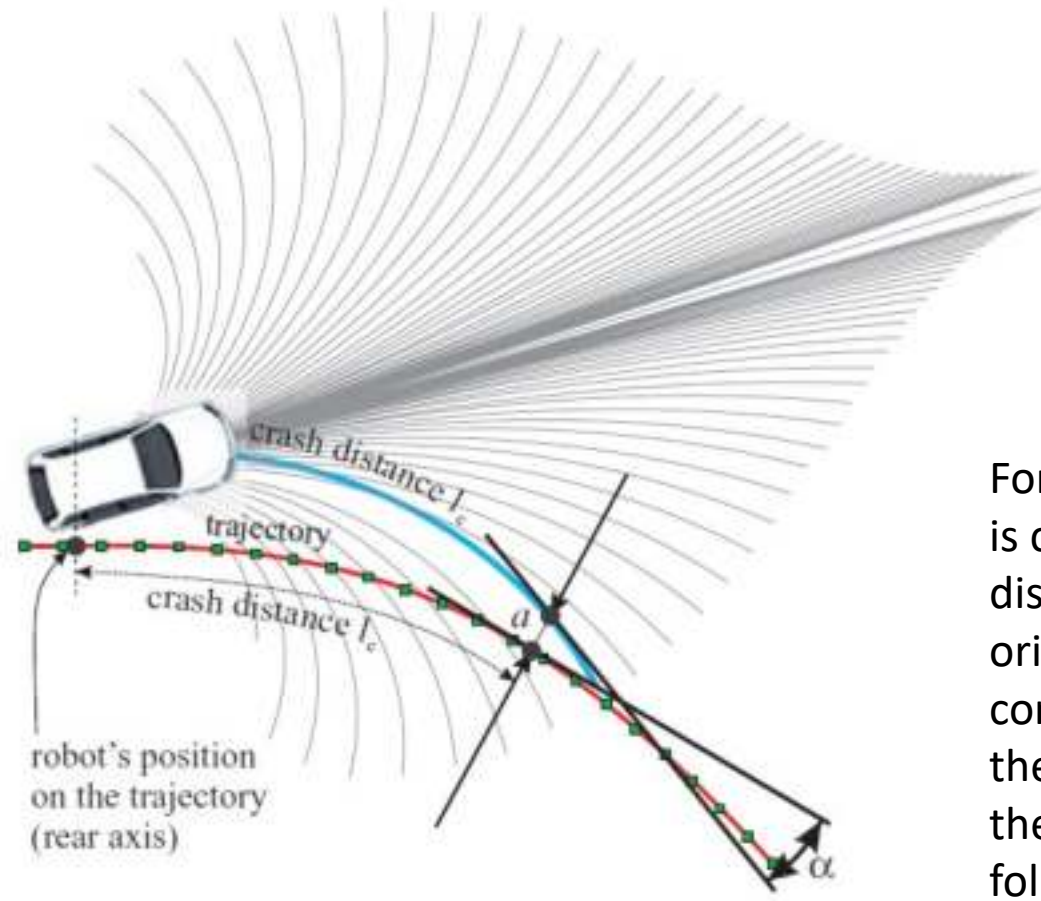


(a) By neglecting the distance to an obstacle, all tentacles would be classified undrivable.



(b) shows the concept of classifying tentacles as undrivable only in case of being occupied within a speed-dependent crash distance. In this case, some drivable tentacles remain, allowing a pass of the car.

Using tentacles to follow a path



For each tentacle, a score value is computed by considering the distance and tangent orientations of two corresponding points, one on the tentacle and the other on the (GPS) trajectory to be followed.

Which tentacle to use?

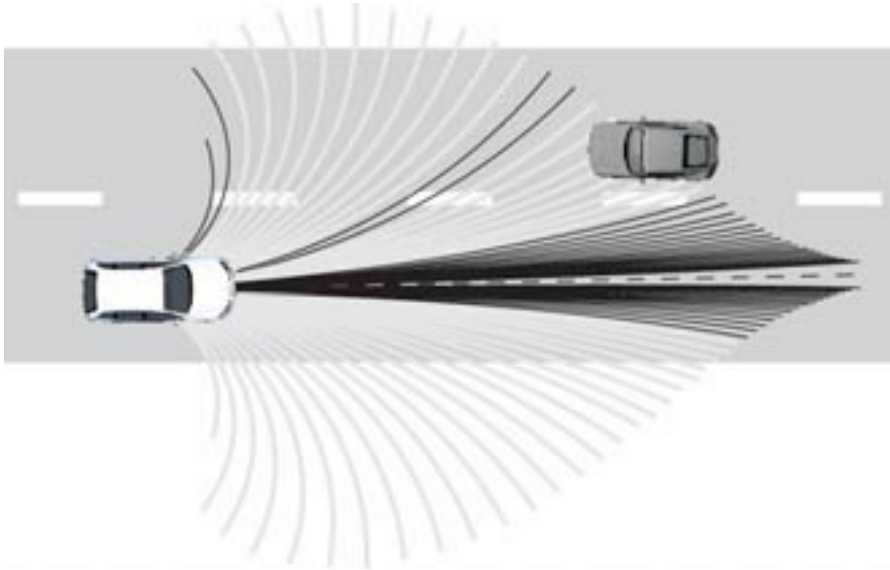
For each tentacle classified as *drivable*, the three values $v_{\text{clearance}}$, v_{flatness} , and $v_{\text{trajectory}}$ are within the range $0, \dots, 1$. They are now linearly combined into a single value:

$$v_{\text{combined}} = a_0 v_{\text{clearance}} + a_1 v_{\text{flatness}} + a_2 v_{\text{trajectory}}. \quad (16)$$

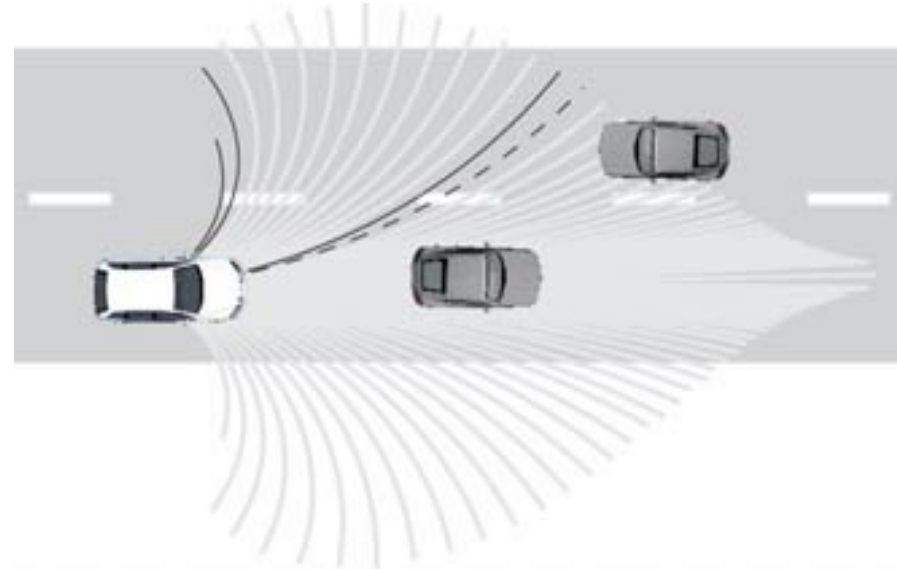
C-Elrob competition values: $a_0 = 0$ $a_1 = 1$ $a_2 = 0$

DARPA Urban Grand Challenge competition values: $a_0 = 1$ $a_1 = 0$ $a_2 = 0.5$

Traffic situations



Three possible ways to go.
Handled well by a heuristic that
prefers tentacles that are more
similar to current direction of
motion.



Relying on the tentacle
algorithm alone would take the
car out into the opposing traffic
lane. Needs to be combined
with additional safeguards.



<http://www.youtube.com/watch?v=uBZRA6wf2Qc>