Robot Motion Control and Planning

http://www.ceng.metu.edu.tr/~saranli/courses/ceng786

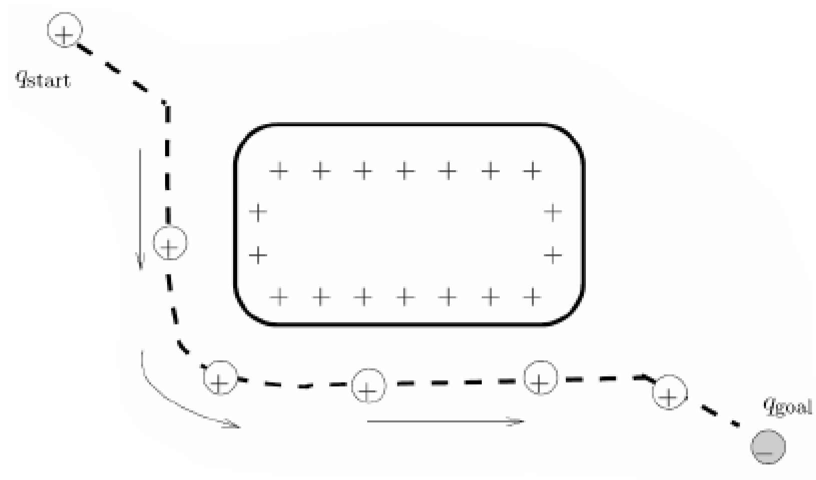
Lecture 4 – Potential Fields

Uluç Saranlı

http://www.ceng.metu.edu.tr/~saranli

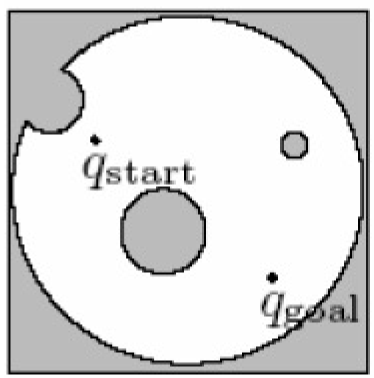
The Basic Idea

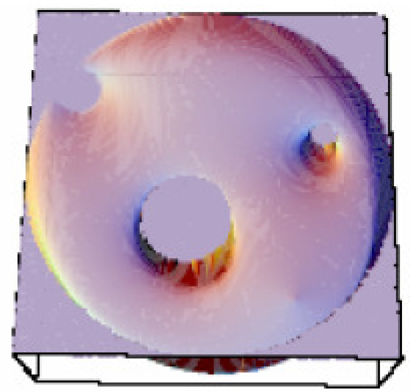
- A really simple idea:
  - Suppose the goal is a point g in $\Re^2$
  - Suppose the robot is a point r in $\Re^2$
  - Think of a "spring" drawing the robot toward the goal and away from obstacles:
  - Can also think of like and opposite charges



Another Idea

- Think of the goal as the bottom of a bowl
- The robot is at the rim of the bowl
- What will happen?





The General Idea

- Both the bowl and the spring analogies are ways of storing potential *energy*

- The robot moves to a lower energy configuration

- A *potential function* is a function U : $\mathfrak{R}^m \rightarrow \mathfrak{R}$

- Energy is minimized by following the negative *gradient* of the potential energy function:

$$\nabla U(q) = DU(q)^T = [\frac{\partial U}{\partial q_1}(q), ..., \frac{\partial U}{\partial q_m}(q)]^T$$

- 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 the point and goes in that direction

Attractive/Repulsive Potential Field

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

- U_{att} is the "attractive" potential --- move to the goal
- U_{rep} is the "repulsive" potential --- avoid obstacles

Artificial Potential Field Methods

Attractive Potentials

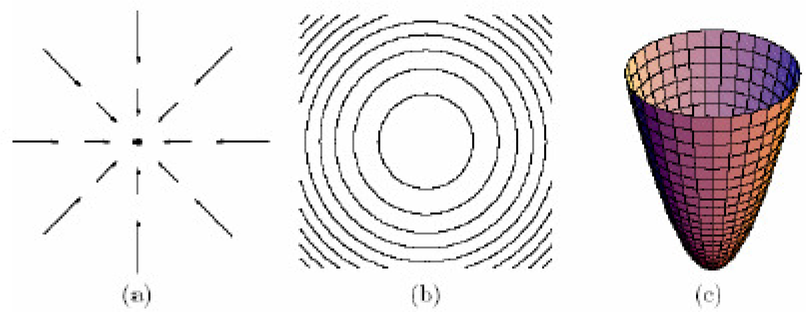
Conical Potential

$$U(q) = \zeta d(q, q_{\text{goal}}).$$

$$\nabla U(q) = \frac{\zeta}{d(q, q_{\text{goal}})}(q - q_{\text{goal}}).$$

Quadratic Potential

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



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

Artificial Potential Field Methods

Attractive Potentials

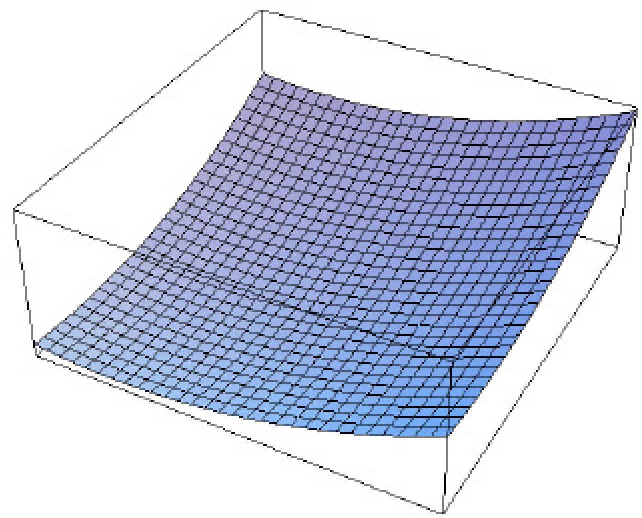
In some cases, it may be desirable to have distance functions that grow more slowly to avoid huge velocities far from the goal

one idea is to use the quadratic potential near the goal (< d*) and the conic farther away One minor issue: what?

Combined Potential

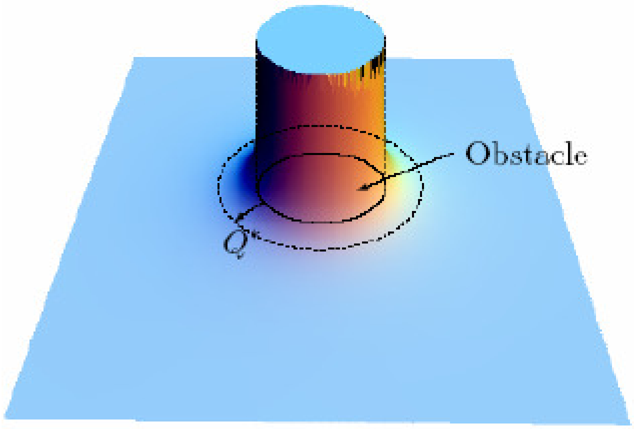
$$U_{\text{att}}(q) = \begin{cases} \frac{1}{2}\zeta d^2(q, q_{\text{goal}}), & d(q, q_{\text{goal}}) \leq d^*_{\text{goal}}, \\ d^*_{\text{goal}}\zeta d(q, q_{\text{goal}}) - \frac{1}{2}\zeta (d^*_{\text{goal}})^2, & d(q, q_{\text{goal}}) > d^*_{\text{goal}}. \end{cases}$$

$$\nabla U_{\text{att}}(q) = \begin{cases} \zeta(q - q_{\text{goal}}), & d(q, q_{\text{goal}}) \leq d^*_{\text{goal}}, \\ \frac{d^*_{\text{goal}}\zeta(q - q_{\text{goal}})}{d(q, q_{\text{goal}})}, & d(q, q_{\text{goal}}) > d^*_{\text{goal}}, \end{cases}$$



Artificial Potential Field Methods

Repulsive Potentials



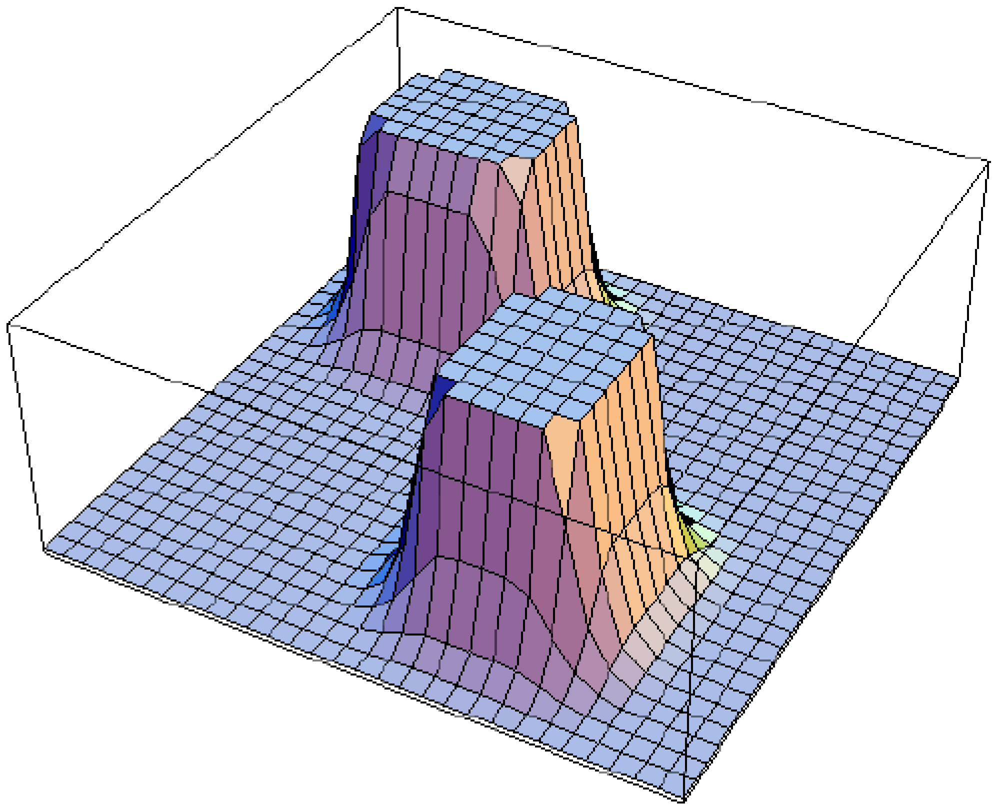
$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

whose gradient is

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

Artificial Potential Field Methods

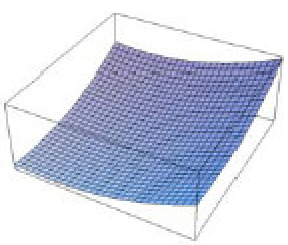
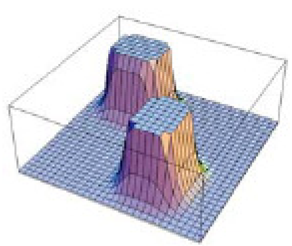
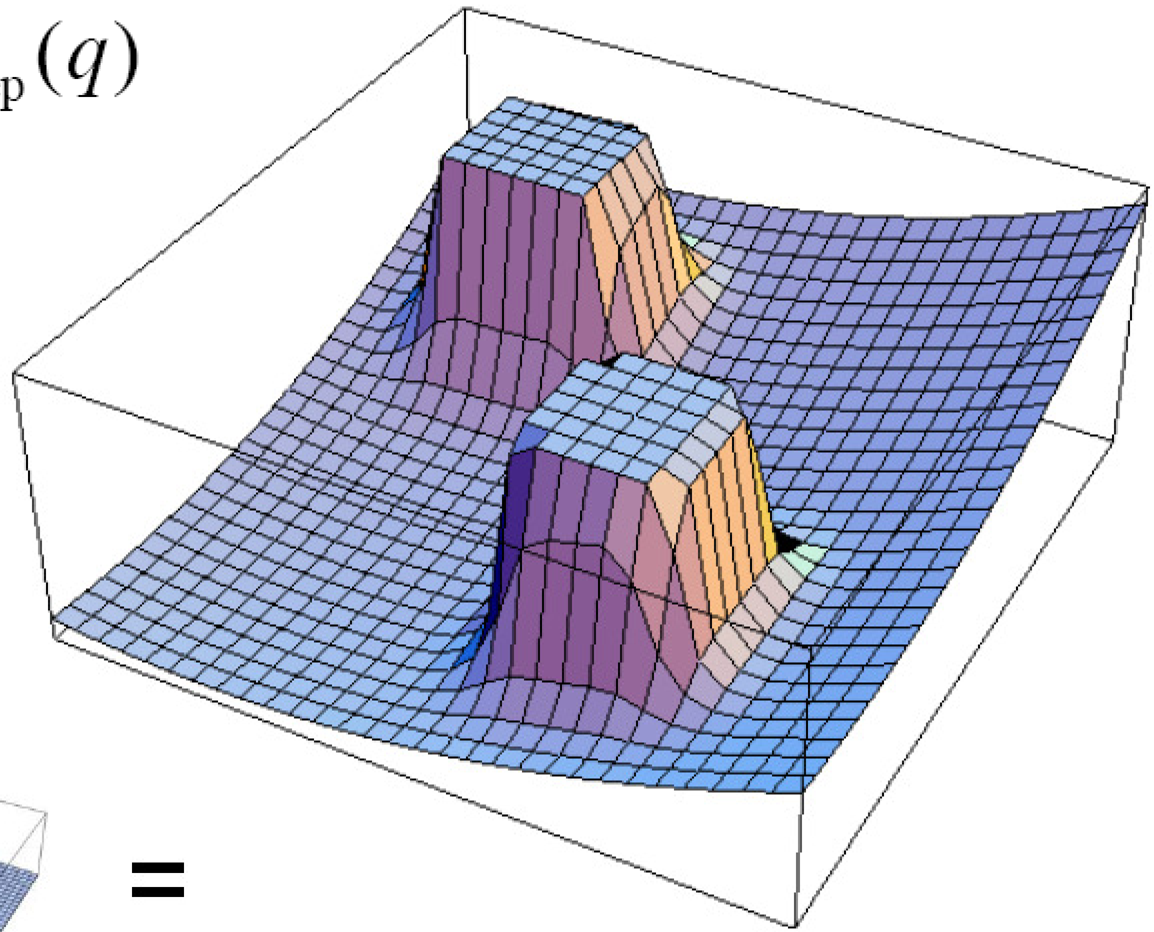
Repulsive Potentials



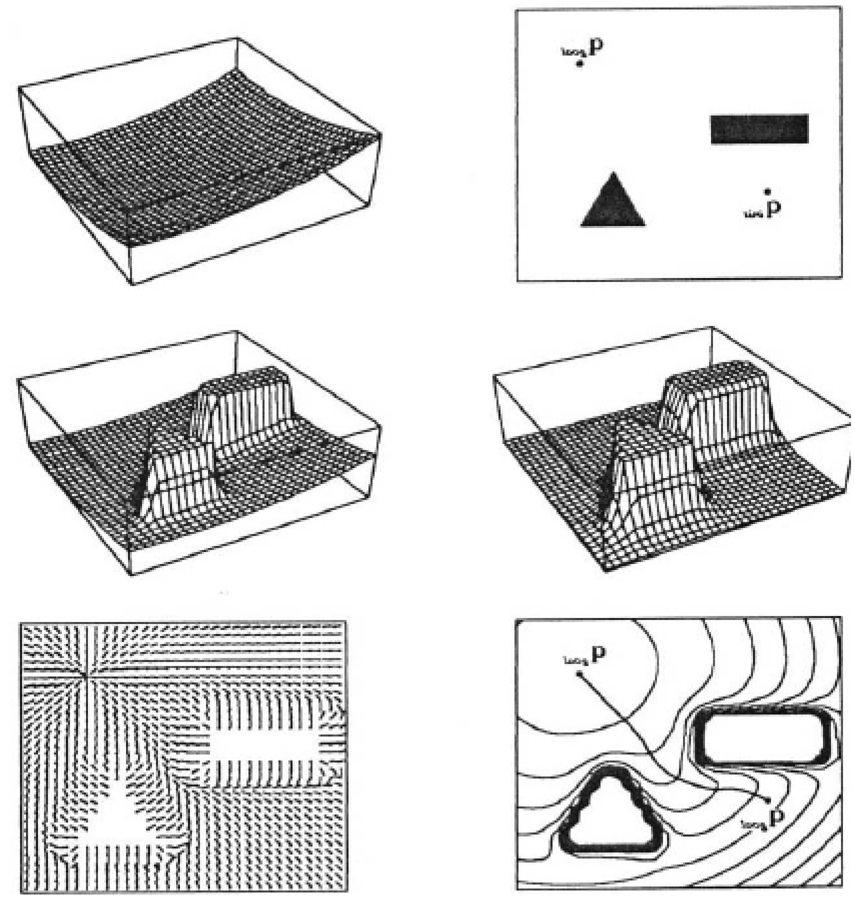
Total Potential Function

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

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

 +  = 

Potential Fields

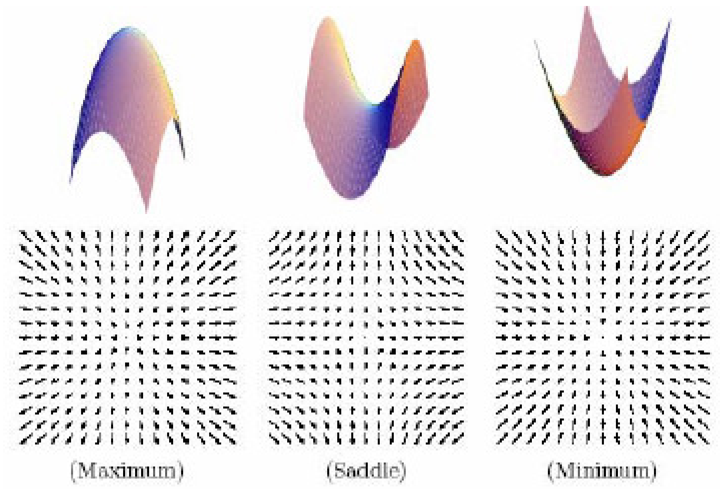


Gradient Descent

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

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

- A *critical point* is a point q* where $\nabla U(q^*) = 0$
  - Equation is stationary at a critical point
  - Max, min, saddle
  - Stability?



The Hessian

- For a 1-d function, how do we know we are at a unique minimum (or maximum)?

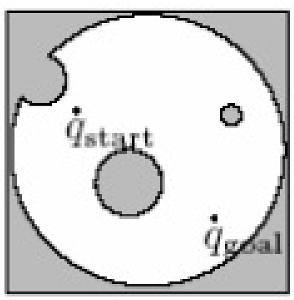
- The Hessian is the m×m matrix of second derivatives

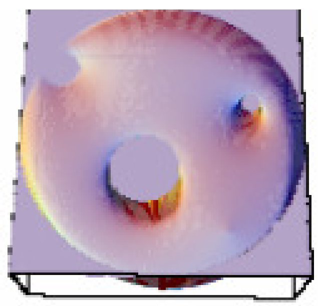
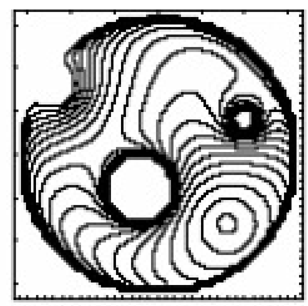
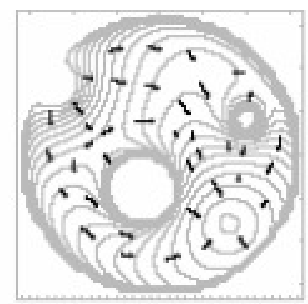
- If the Hessian is nonsingular ($Det(H) \neq 0$), the critical point is a unique point
  – if H is positive definite ($x^T H x > 0$), a minimum
  – if H is negative definite, a maximum
  – if H is indefinite, a saddle point

Gradient Descent

Gradient Descent:

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



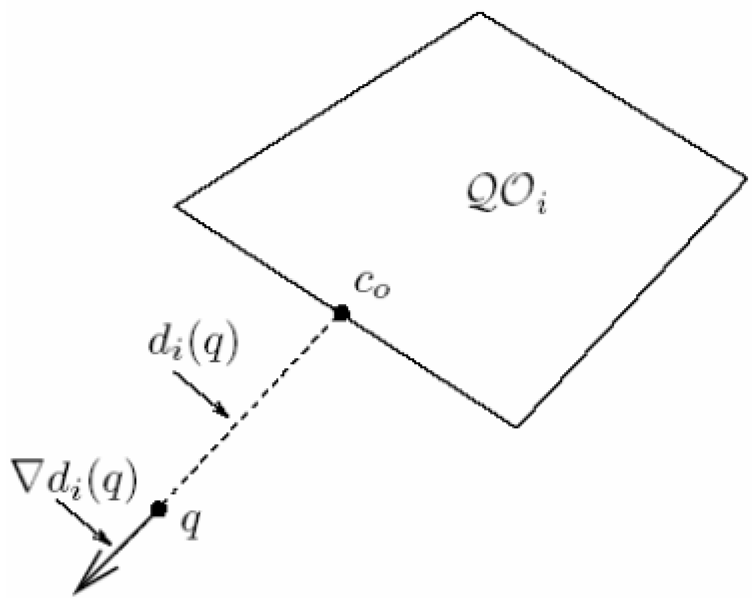




Single Object Distance

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q,c)$$

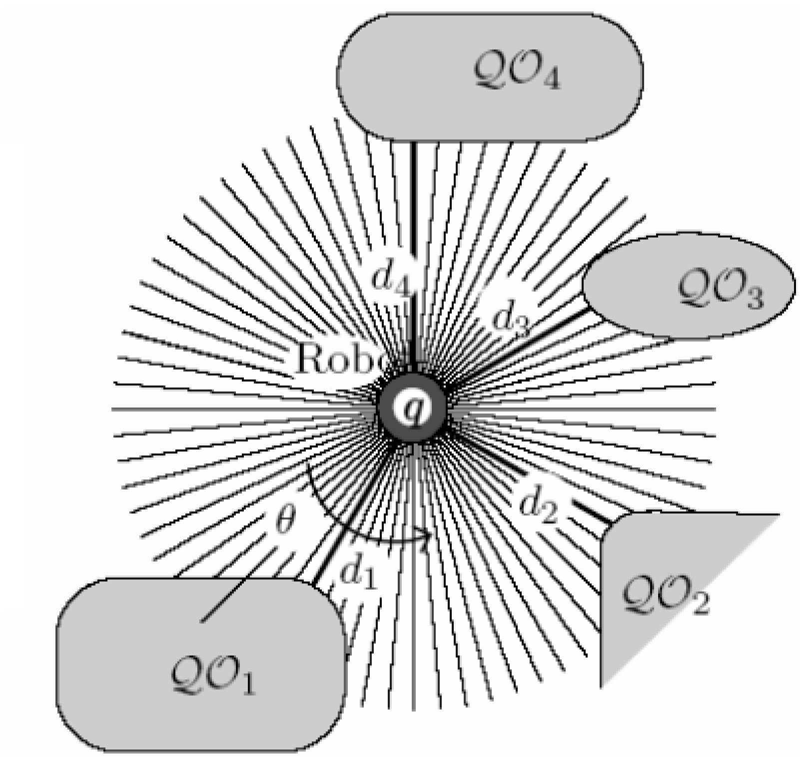
$$\nabla d_i(q) = \frac{q-c}{d(q,c)}$$

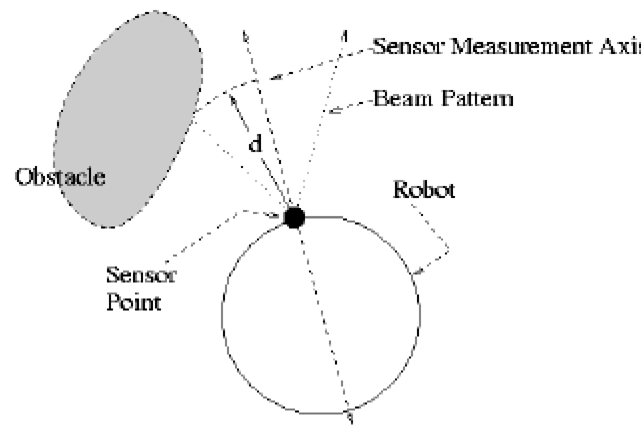
$$U_{\mathrm{rep}_i}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d_i(q)} - \frac{1}{Q_i^*}\right)^2, & \text{if } d_i(q) \le Q_i^* \\ 0, & \text{if } d_i(q) > Q_i^* \end{cases}$$



$$U_{\mathrm{rep}}(q) = \sum_{i=1}^{n} U_{\mathrm{rep}_i}(q)$$

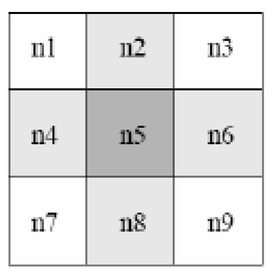
Compute Distance: Sensor Information



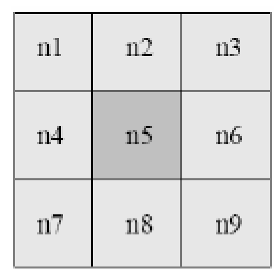


Computing Distance: Use a Grid

- use a discrete version of space and work from there
  - The Brushfire algorithm is one way to do this
    - need to define a grid on space
    - need to define connectivity (4/8)
    - obstacles start with a 1 in grid; free space is zero



4



8

Brushfire Algorithm

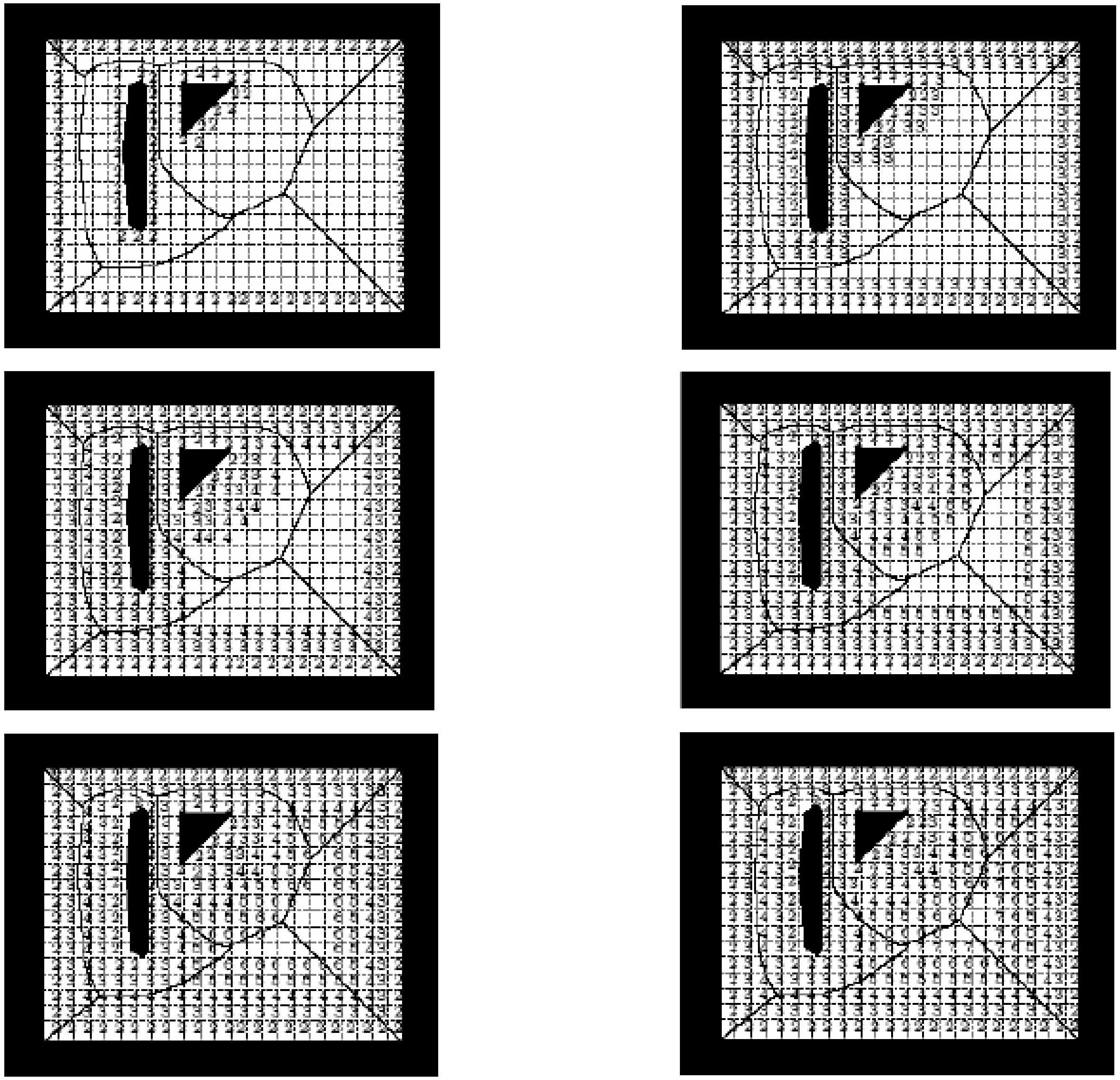
- Initially: create a queue L of pixels on the boundary of all obstacles

- While $L \neq \emptyset$
  - pop the top element t of L
  - if $d(t) = 0$,
    - set $d(t)$ to $1+\min_{t' \text{ in } N(t), d(t) \neq 0} d(t')$
    - Add all t'in N(t) with d(t)=0 to L (at the end)

- The result is a distance map d where each cell holds the minimum distance to an obstacle.

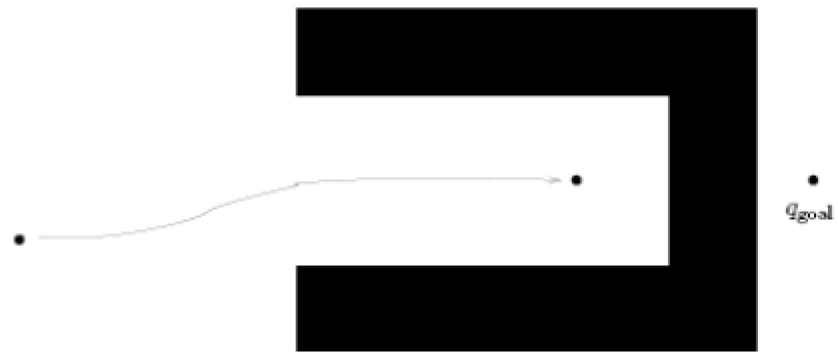
- The gradient of distance is easily found by taking differences with all neighboring cells.

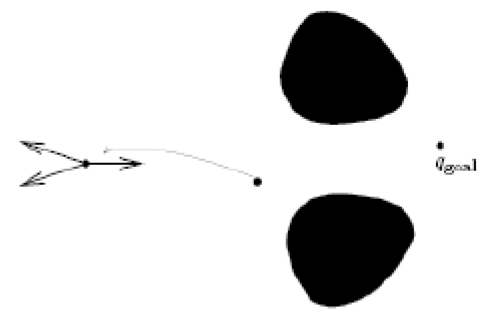
Brushfire example



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).

The Wave-front Planner

- Apply the brushfire algorithm starting from the goal
- Label the goal pixel 2 and add all zero neighbors to L
  - While $L \neq \emptyset$
    - pop the top element of L, t
    - set d(t) to $1+\min_{t' \text{ in } N(t), d(t) > 1} d(t')$
    - Add all t' in N(t) with d(t)=0 to L (at the end)
- The result is now a distance for every cell
  - gradient descent is again a matter of moving to the neighbor with the lowest distance value

The Wavefront Planner: Setup

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 1)

- Starting with the goal, set all adjacent cells with “0” to the current cell + 1
  - 4-Point Connectivity or 8-Point Connectivity?
  - Your Choice. We’ll use 8-Point Connectivity in our example

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 2)

- Now repeat with the modified cells
  - This will be repeated until no 0’s are adjacent to cells with values >= 2
    - 0’s will only remain when regions are unreachable

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 3)

- Repeat again...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 4)

- And again...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 5)

- And again until...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	7	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Done)

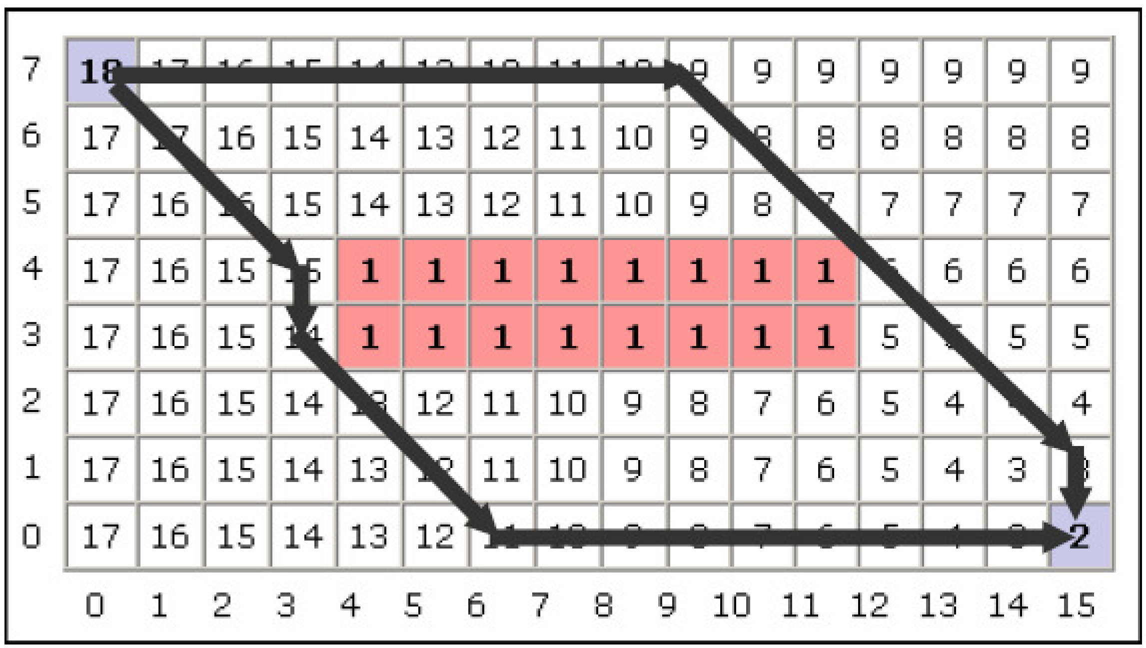
- You're done
  - Remember, 0's should only remain if unreachable regions exist

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

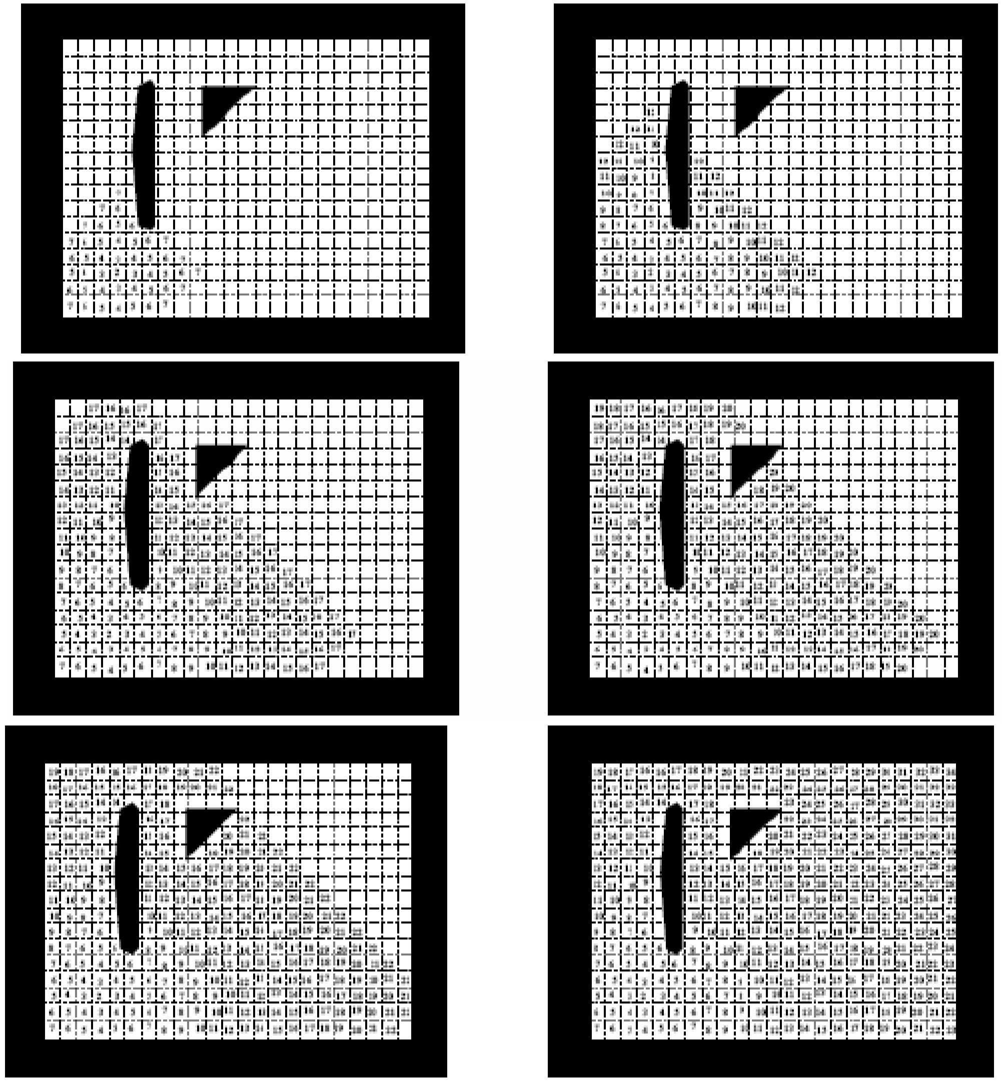
The Wavefront, Now What?

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
  - The numbers generated by the Wavefront planner are roughly proportional to their distance from the goal

Two possible shortest paths shown



Another Example



Wavefront (Overview)

- Divide the space into a grid.
- Number the squares starting at the start in either 4 or 8 point connectivity starting at the goal, increasing till you reach the start.
- Your path is defined by any uninterrupted sequence of decreasing numbers that lead to the goal.

Navigation Functions

- A function φ: Qfree $\rightarrow$ [0,1] is called a *navigation function if it*
  - is smooth (or at least C^2)
  - has a unique minimum at q_{goal}
  - is uniformly maximal on the boundary of free space
  - is Morse

- A function is Morse if every critical point (a point where the gradient is zero) is isolated.

- The question: when can we construct such a function?

Sphere World

- Suppose that the world is a sphere of radius r_0 centered at q_0 containing n obstacles of radius r_i centered at q_i, i=1 .. n
  - $\beta_0(q) = -d^2(q,q_0) + r_0^2$
  - $\beta_i(q) = d^2(q,q_i) - r_i^2$

$$QO_i = \{q | B_i(q) \leq 0\}$$

- Define $\beta(q) = \Pi\ \beta_i(q)$ (Repulsive)
  - note this is zero on any obstacle boundary, positive in free space and negative inside an obstacle

- Define $\gamma_k(q) = (d(q, q_{\text{goal}}))^{2k}$ (Attractive)
  - note this will be zero at the goal, and increasing as we move away
  - κ controls the rate of growth

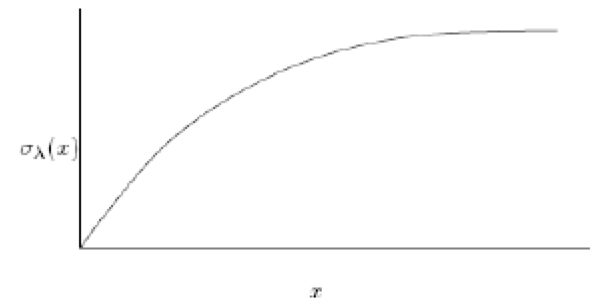
Sphere World

- Consider now $\frac{\gamma_k}{\beta}(q)$
  - $\frac{\gamma_k}{\beta}(q)$ is only zero at the goal
  - $\frac{\gamma_k}{\beta}(q)$ goes to infinity at the boundary of any obstacle
  - By increasing κ, we can make the gradient at any direction point toward the goal
  - It is possible to show that the only stationary point is the goal, with positive definite Hessian because $\frac{\partial \gamma_k}{\partial q} \text{ dominates } \frac{\partial \beta}{\partial q}$
    - therefore no local minima

- In short, following the gradient of $\frac{\gamma_k}{\beta}(q)$ is guaranteed to get to the goal (for a large enough value of κ)

An Example: Sphere World

- One problem: the value of $\frac{\gamma_k}{\beta}(q)$ may be very large
- A solution: introduce a "switch" $\sigma_\lambda(x) = \frac{x}{\lambda + x} \quad \lambda > 0$
- Now, define $s(q,\lambda) = \left(\sigma_\lambda \circ \frac{\gamma_\kappa}{\beta}\right)(q) = \left(\frac{\gamma_\kappa}{\lambda\beta + \gamma_\kappa}\right)(q)$
  - this bounds the value of the function
  - however, $s(q,\lambda)$ may turn out not to be Morse



- A solution: introduce a "sharpening function" $\xi_\kappa(x) = x^{1/\kappa}$

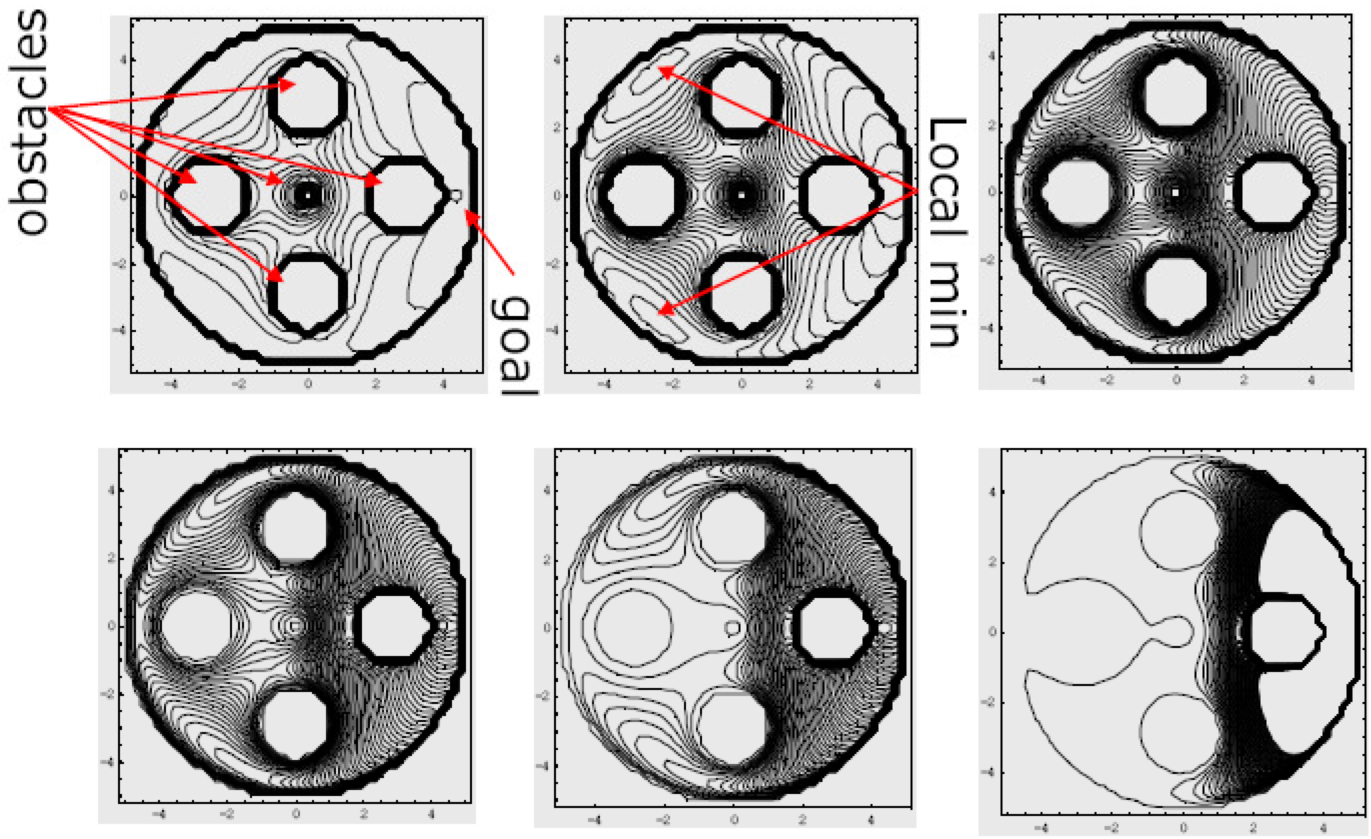
$$\psi(q) = \left(\xi_\kappa \circ \sigma_1 \circ \frac{\gamma_\kappa}{\beta}\right)(q) = \frac{d^2(q, q_{\text{goal}})}{[(d(q, q_{\text{goal}}))^{2\kappa} + \beta(q)]^{1/\kappa}}$$

- For large enough κ, this is a navigation function on the sphere world!

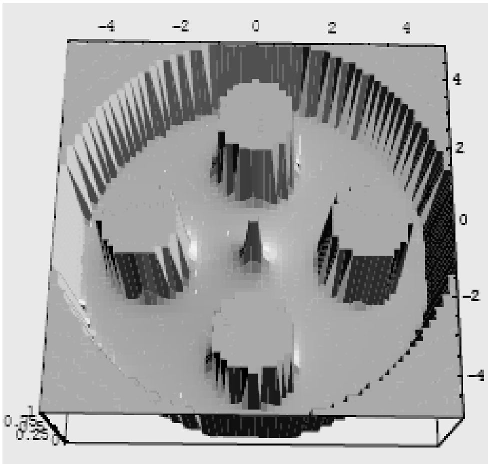
Navigation Function for Sphere World

$$\psi(q) = \left(\xi_\kappa \circ \sigma_1 \circ \frac{\gamma_\kappa}{\beta}\right)(q) = \frac{d^2(q,q_{\text{goal}})}{[(d(q,q_{\text{goal}}))^{2\kappa}+\beta(q)]^{1/\kappa}}$$

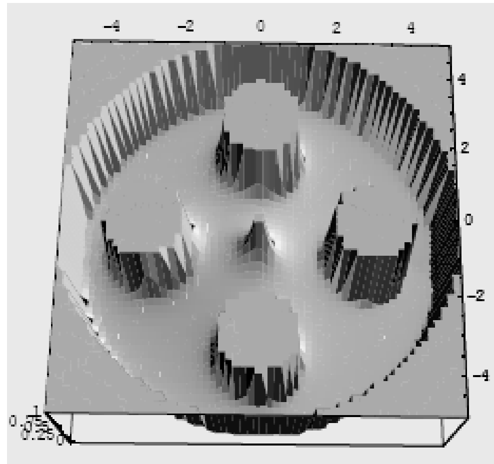
For sufficiently large k, $\psi(q)$ is a navigation function



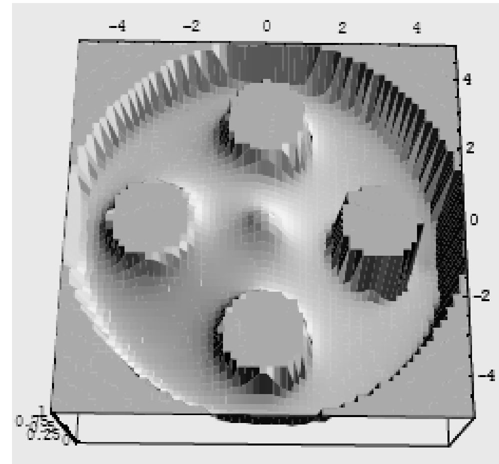
Navigation Function : $\psi(q)$, varying k



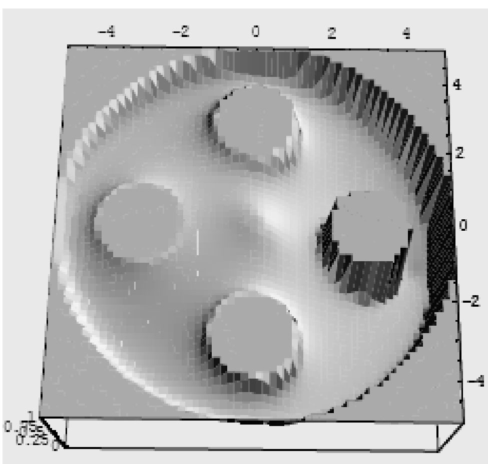
k=3



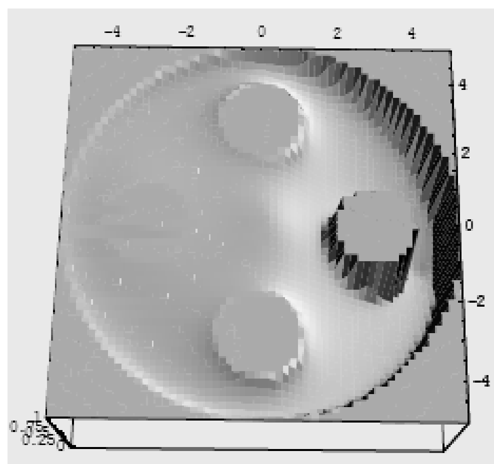
k=4



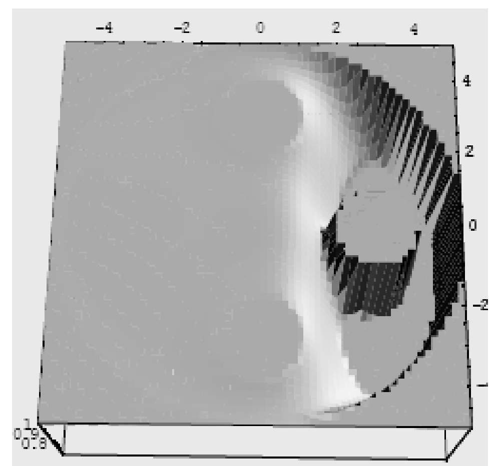
k=6



k=7



k=8



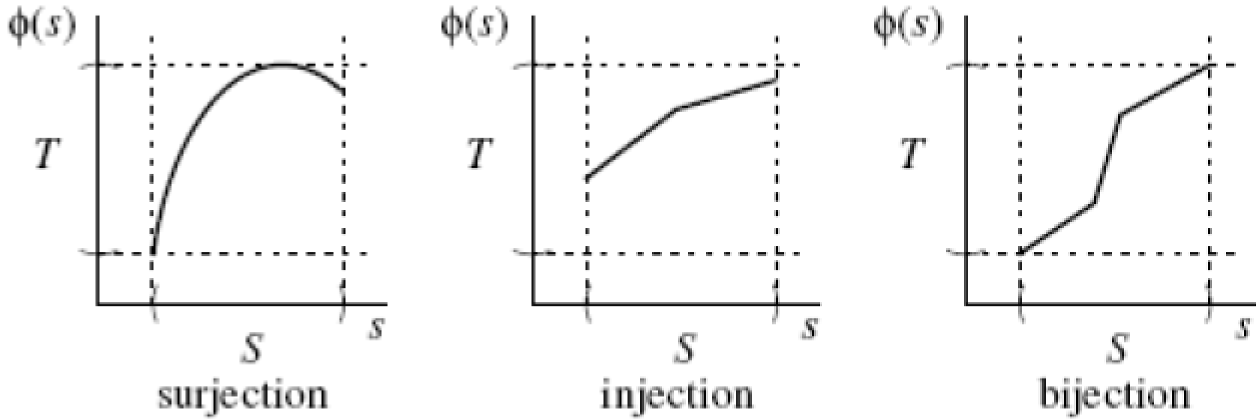
k=10

From Spheres to Stars and Beyond

- While it may not seem like it, we have solved a very general problem
- Suppose we have a **diffeomorphism** δ from some world W to a sphere world S
  - if $\psi(q)$ is a navigation function on S then
  - $\psi'(q) = \psi(\delta(q))$ is a navigation function on W!
    - note we also need to take the diffeomorphism into account for distances
    - Because δ is a diffeomorphism, the Jacobian is full rank
    - Because the Jacobian is full rank, the gradient map cannot have new zeros introduced (which could only happen if the gradient was in the null space of the Jacobian)
- A star world is one example where a diffeomorphism is known to exist
  - a star-shaped set is one in which all boundary points can be “seen” from some single point in the space.

$$\exists x \text{ such that } \forall y \in S, tx + (1-t)y \in S \quad \forall t \in [0,1]$$

_____ jections

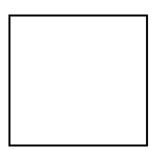


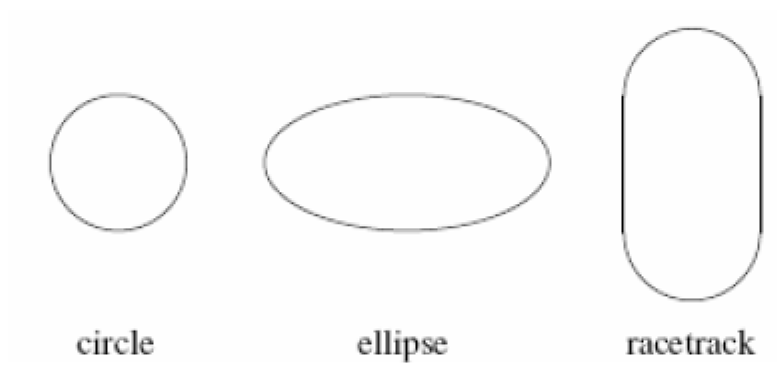
Diffeomorphism vs. Homeomorphism

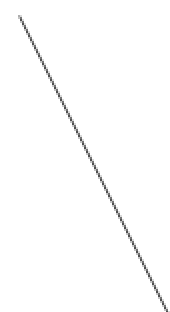
HOMEOMORPHISM *If* $\phi: S \to T$ *is a bijection, and both* ϕ *and* ϕ^{-1} *are continuous, then* ϕ *is a* homeomorphism. *When such a* ϕ *exists,* S *and* T *are said to be* homeomorphic.

A mapping $\phi: U \to V$ is said to be *smooth* if all partial derivatives of ϕ, of all orders, are well defined (i.e., ϕ is of class C^∞). With the notion of smoothness, we define a second type of bijection.

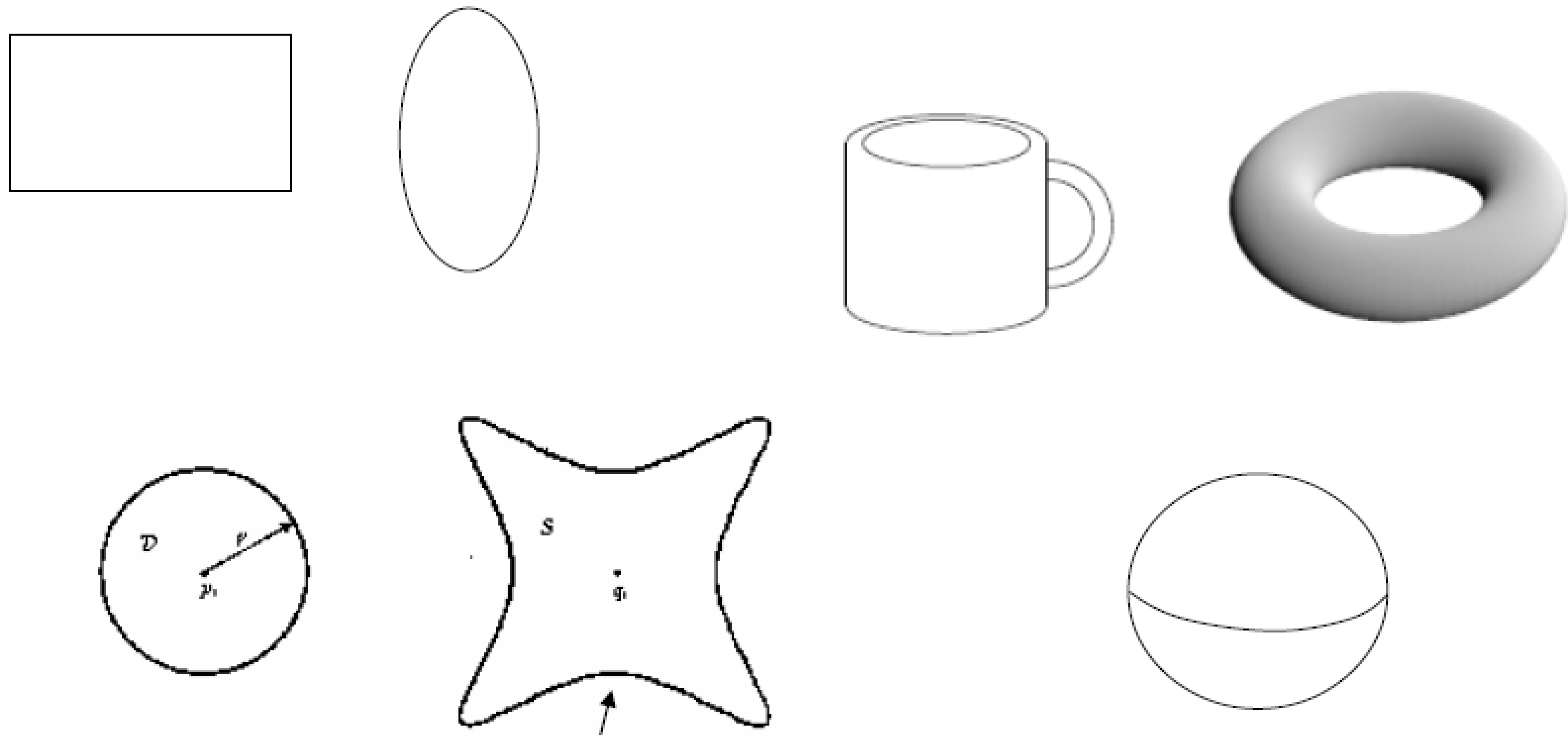
DIFFEOMORPHISM *A smooth map* $\phi: U \to V$ *is a* diffeomorphism *if* ϕ *is bijective and* ϕ^{-1} *is smooth. When such a* ϕ *exists,* U *and* V *are said to be* diffeomorphic.







Which of the following are the same?



$\exists x$ such that $\forall y \in S, \quad tx + (1-t)y \in S \quad \forall t \in [0,1]$

Construct the Mapping

Radius of Circle shaped set

$$T_i(q) = \nu_i(q)(q - q_i) + p_i \qquad \nu_i(q) = (1 + \beta_i(q))^{1/2} \frac{r_i}{\|q - q_i\|}$$

Center of Star shaped set

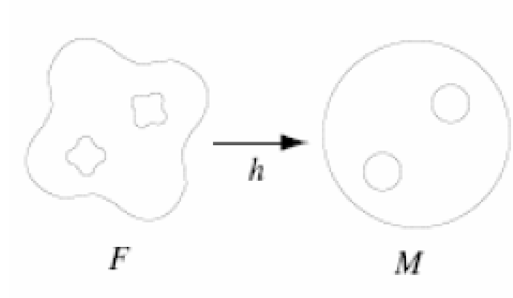
Center of Circle shaped set

Star shaped set

Maps stars to spheres

For points on boundary of star shaped set $(1 + \beta_i(q)) = 1$

$$T_i(q) = r_i \frac{q - q_i}{d(q, q_i)} + p_i.$$



For the star-shaped obstacle $\mathcal{QO}_i$,

$$s_i(q,\lambda) = \left(\sigma_\lambda \circ \frac{\gamma_\kappa \bar{\beta}_i}{\beta_i}\right)(q) = \left(\frac{\gamma_\kappa \bar{\beta}_i}{\gamma_\kappa \bar{\beta}_i + \lambda \beta_i}\right)(q).$$

$$\bar{\beta}_i = \prod_{j=0, j\neq i}^{n} \beta_j$$

Zero on boundary of obstacles except the "current" one

$$s_{q_{\text{goal}}}(q,\lambda) = 1 - \sum_{i=0}^{M} s_i$$

$s_i(q,\lambda)$ One on the boundary of $\mathcal{QO}_i$ and Zero on the goal and other obstacle boundaries

$h_\lambda(q)$ is exactly $T_i(q)$ on the boundary of the $\mathcal{QO}_i$

$$h_\lambda(q) = s_{q_{\text{goal}}}(q,\lambda) T_{q_{\text{goal}}}(q) + \sum^{M} s_i(q,\lambda) T_i(q). \qquad T_{q_{\text{goal}}}(q) = q$$

for a suitable λ, $h_\lambda(q)$ is smooth, bijective, and has a smooth inverse.

Potential Fields on Non-Euclidean Spaces

- Thus far, we've dealt with points in R^n --- what about real manipulators
- Recall we can think of the gradient vectors as forces -- the basic idea is to define forces in the workspace (which is $\Re^2$ or $\Re^3$)

force f acting at a point $x = \phi(q)$

force u acting in the robot's configuration

$\dot{x} = J\dot{q}$, where $J = \partial\phi/\partial q$

$u^T\dot{q}$ Power in configuration space

$f^T\dot{x}$ Power in work space

<u>Power is conserved!</u>

$$\begin{aligned} f^T J\dot{q} &= u^T\dot{q} \\ f^T J &= u^T \\ J^T f &= u. \end{aligned}$$

Force on an Object

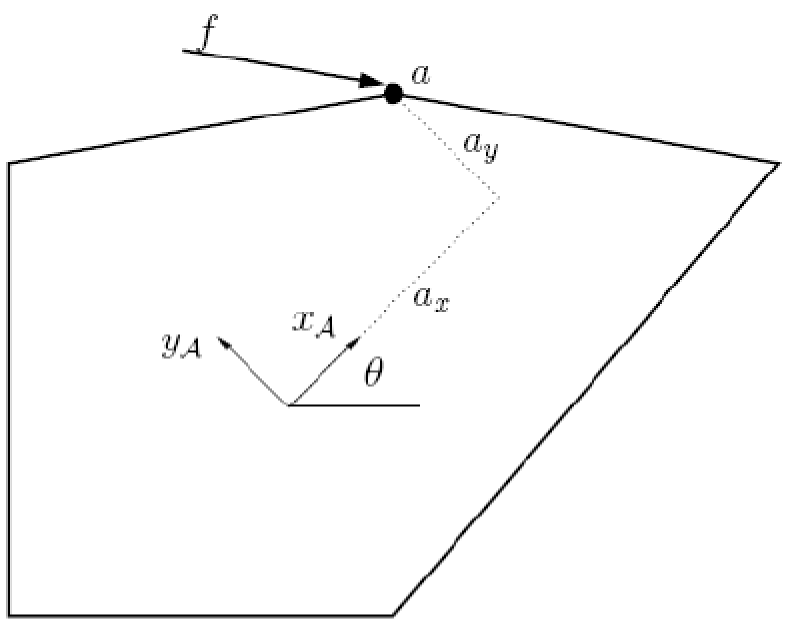
$$q = [x, y, \theta]^T$$

$$[a_x, a_y]^T$$

robot's local coordinate frame.

$$\phi(q) = \begin{bmatrix} x + a_x \cos\theta - a_y \sin\theta \\ y + a_x \sin\theta + a_y \cos\theta \end{bmatrix}$$

$$J(q) = \frac{\partial \phi}{\partial q}(q) = \begin{bmatrix} 1 & 0 & -a_x \sin\theta - a_y \cos\theta \\ 0 & 1 & a_x \cos\theta - a_y \sin\theta \end{bmatrix}$$

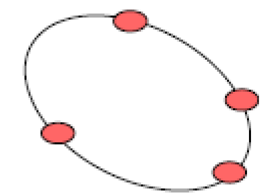


$$\begin{aligned} \begin{bmatrix} u_x \\ u_y \\ u_\theta \end{bmatrix} &= J^T(q) \begin{bmatrix} f_x \\ f_y \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -a_x \sin\theta - a_y \cos\theta & a_x \cos\theta - a_y \sin\theta \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} \\ &= \begin{bmatrix} f_x \\ f_y \\ -f_x(a_x \sin\theta + a_y \cos\theta) + f_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix} \end{aligned}$$

torque

Potential Function on Rigid Body

pick control points $\{r_i\}$ on the robot



Pick enough points to “pin down” robot (2 in plane)

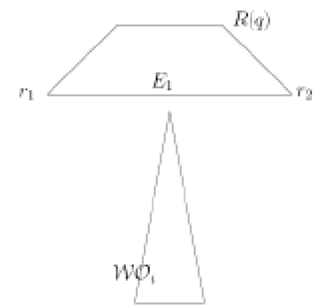
$$U_{\text{att},j}(q) = \begin{cases} \frac{1}{2}\zeta_i d^2(r_j(q), r_j(q_{\text{goal}})), & d(r_j(q), r_j(q_{\text{goal}})) \le d_0 \\ d\zeta_i d(r_i(q), r_i(q_{\text{goal}})) - \frac{1}{2}\zeta_i d^2, & d(r_i(q), r_i(q_{\text{goal}})) > d_0. \end{cases}$$

$$\nabla U_{\text{att},j}((q) = \begin{cases} \zeta_i(r_j(q) - r_j(q_{\text{goal}})), & d(r_i(q), r_i(q_{\text{goal}})) \le d_0, \\ \frac{d\zeta_j(r_j(q) - r_j(q_{\text{goal}}))}{d(r_j(q), r_j(q_{\text{goal}}))}, & d(r_j(q), r_j(q_{\text{goal}})) > d_0. \end{cases}$$

$$U_{\text{rep}i,j}(q) = \begin{cases} \frac{1}{2}\eta_j \left(\frac{1}{d_i(r_j(q))} - \frac{1}{Q_i^*}\right)^2, & d_i(r_j(q)) \le Q_i^*. \\ 0, & d_i(r_j(q)) > Q_i^*. \end{cases}$$

$$\nabla U_{\text{rep}i,j}(q) = \begin{cases} \eta_j \left(\frac{1}{Q_i^*} - \frac{1}{d_i(r_j(q))}\right) \frac{1}{d_i^2(r_j(q))} \nabla d_i(r_j(q)), & d_i(r_j(q)) \le Q_i^*, \\ 0, & d_i(r_j(q)) > Q_i^*. \end{cases}$$

$$\begin{aligned} u(q) &= \sum_i u_{\text{att}i}(q) + \sum_j u_{\text{rep}j}(q) \\ &= \sum_i J_i^T(q) f_{\text{att}i}(q) + \sum_j J_j^T(q) f_{\text{rep}j}(q) \end{aligned}$$

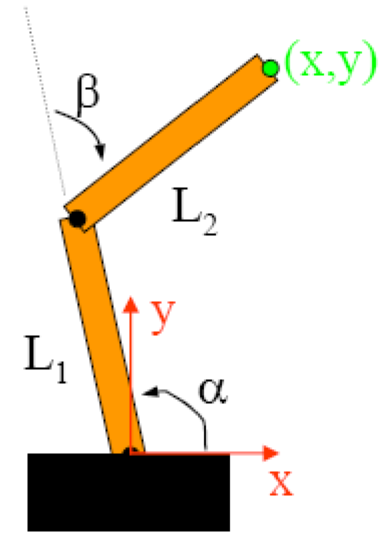


More points please

Potential Fields for Multiple Bodies

- Recall we can think of the gradient vectors as forces -- the basic idea is to define forces in the workspace (which is $\Re^2$ or $\Re^3$)
  - We have $J^T f = u$ where f is in W and u is in Q
  - Thus, we can define forces in W and then map them to Q
  - Example: our two-link manipulator

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 c_\alpha \\ L_1 s_\alpha \end{pmatrix} + \begin{pmatrix} L_2 c_+ \\ L_2 s_+ \end{pmatrix} \quad \text{Position}$$



Potential Fields on Non-Euclidean Spaces

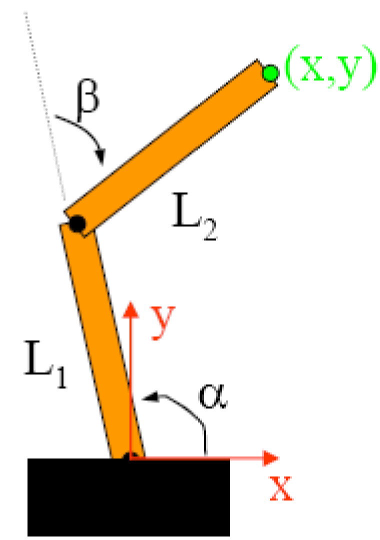
Example: our two-link manipulator

- $J = \begin{matrix} -L_1 s_\alpha - L_2\, s_{\alpha+\beta} & -L_2\, s_{\alpha+\beta} \\ L_1\, c_\alpha + L_2\, c_{\alpha+\beta} & L_2\, c_{\alpha+\beta} \end{matrix}$

Suppose $q_{goal} = (0,0)^t$, then $f_W = (x,y)$

- $f_q = \begin{matrix} x\,(-L_1\, s_\alpha - L_2\, s_{\alpha+\beta}) + y\,(L_1\, c_\alpha + L_2\, c_{\alpha+\beta}) \\ x\,(-L_2\, s_{\alpha+\beta}) + y\, L_2\, c_{\alpha+\beta} \end{matrix}$

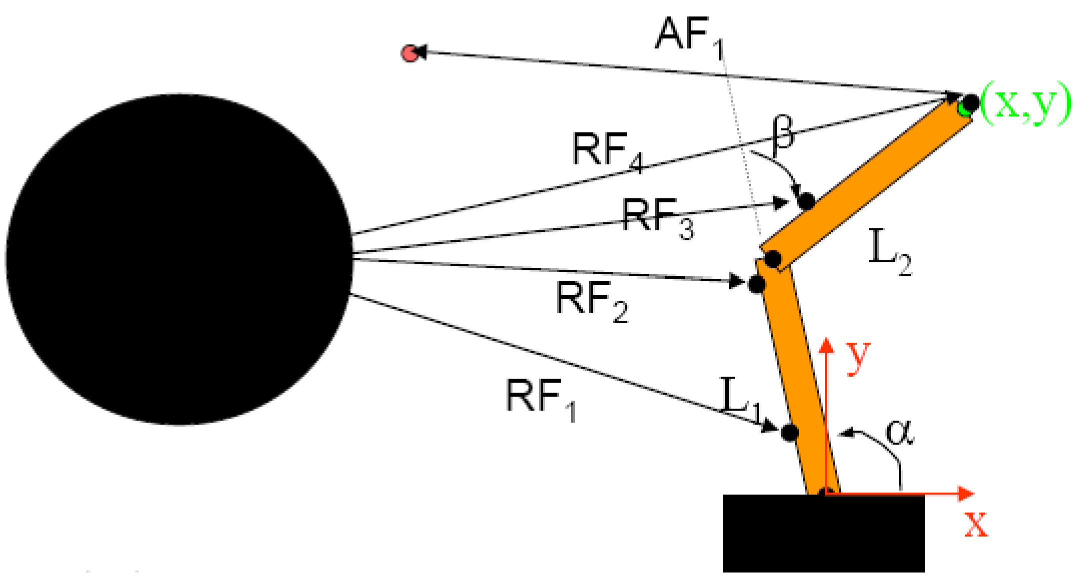
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 c_\alpha \\ L_1 s_\alpha \end{pmatrix} + \begin{pmatrix} L_2 c_+ \\ L_2 s_+ \end{pmatrix} \quad \text{Position}$$



In General

- Pick several points on the manipulator
- Compute attractive and repulsive potentials for each
- Transform these into the configuration space and add
- Use the resulting force to move the robot (in its configuration space)

Be careful to use the correct Jacobian!



Summary

- Basic potential fields
  - attractive/repulsive forces
- Gradient following and Hessian
- Navigation functions
- Extensions to more complex manipulators