Potential Field Methods

- Idea robot is a particle
- Environment is represented as a potential field (locally)
- Advantage – capability to generate on-line collision avoidance

Compute force acting on a robot – incremental path planning

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

Example: Robot can translate freely , we can control independently
Environment represented by a potential function

$$U(x, y)$$

Force is proportional to the gradient of the potential function

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\nabla U(x, y)$$

Some slide thanks to http://cs.cmu.edu/~motionplanning

Attractive potential field

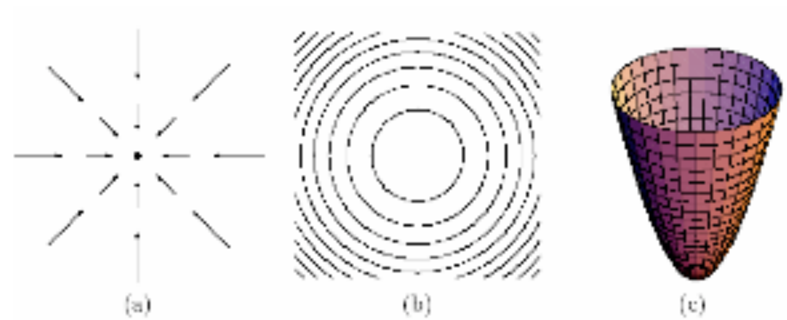
- Linear function of distance

$$U_a(q) = \xi ||q - q_{goal}|| \qquad F_a(q) = -\nabla U_a(q) = -\xi \frac{(q-q_{goal})}{||q-q_{goal}||}$$

- Quadratic function of distance

$$U_a(q) = \xi \frac{1}{2} ||q - q_{goal}||^2 \quad F_a(q) = -\nabla U_a(q) = -\xi (q - q_{goal})$$

Combination of two – far away use li
closer by use parabolic well



Repulsive potential field

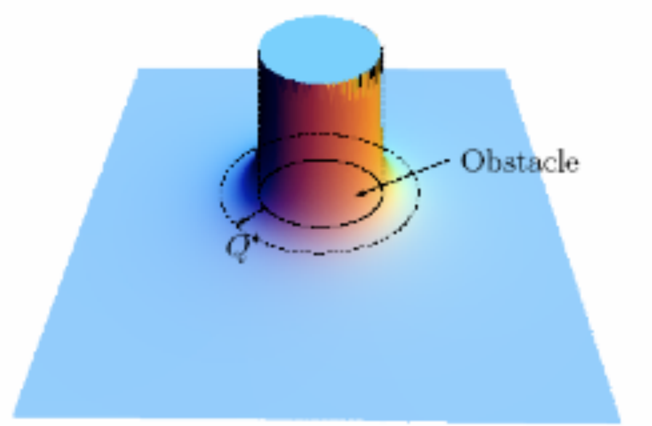
$$U_{rep} = \frac{1}{2}\nu\left(\frac{1}{\rho(q, q_{obst})} - \frac{1}{\rho_0}\right)^2 \quad \text{if} \quad \rho(q, q_{obst})\| \leq \rho_0$$

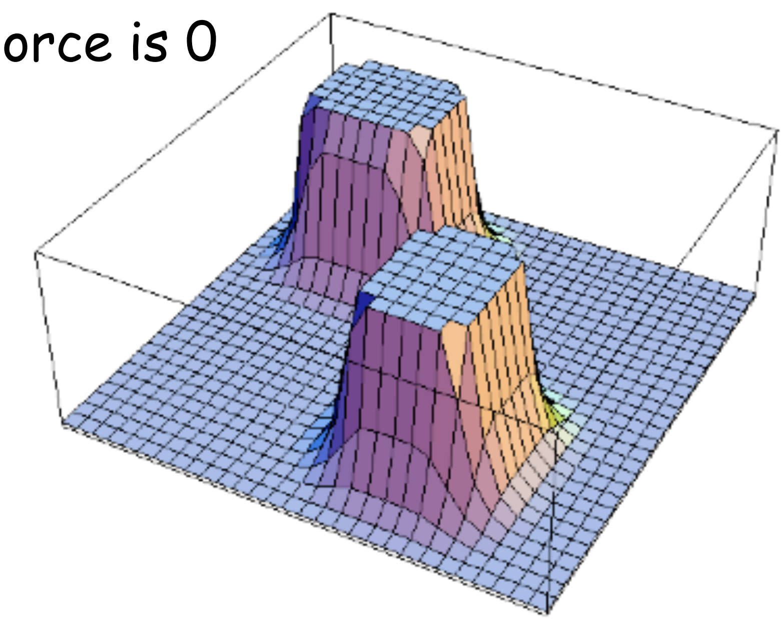
$$\text{else} \quad U_r(q) = 0$$

Minimal distance between the robot and the obstacle

$$F_{rep} = -\nabla U_{rep} = \nu\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho(q)^2}\frac{q - q_{obs}}{\rho(q)}$$

Outside of sensitivity zone repulsive force is 0





Jana Kosecka,

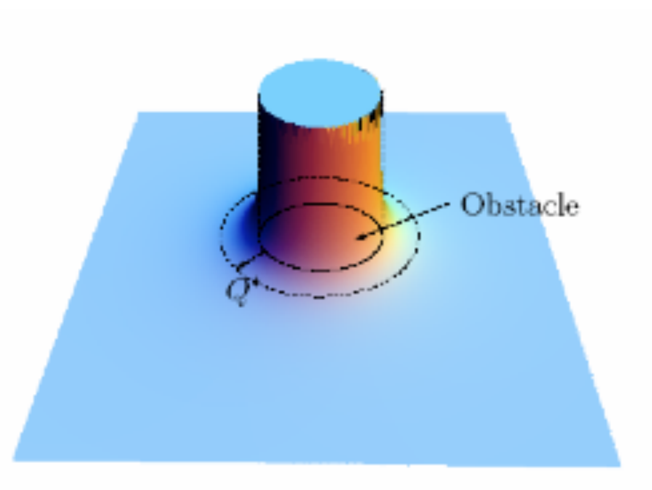
Another example of repulsive potential field

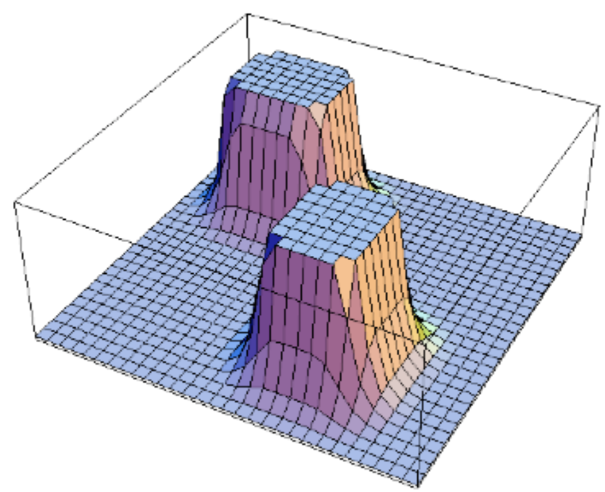
$$U_r(q) = \frac{1}{2}\nu\left(\frac{1}{\rho(q,q_{obst})} - \frac{1}{\rho_o}\right) \quad \text{if} \quad \rho(q, q_{obst})\| \le \rho_0$$

$$\text{else} \quad U_r(q) = 0$$

Minimal distance between the robot and the obstacle

Previously – repulsive potential related to the square of the Inverse distance – here just proportional to inverse distance
Note: need to compute gradient to get the force

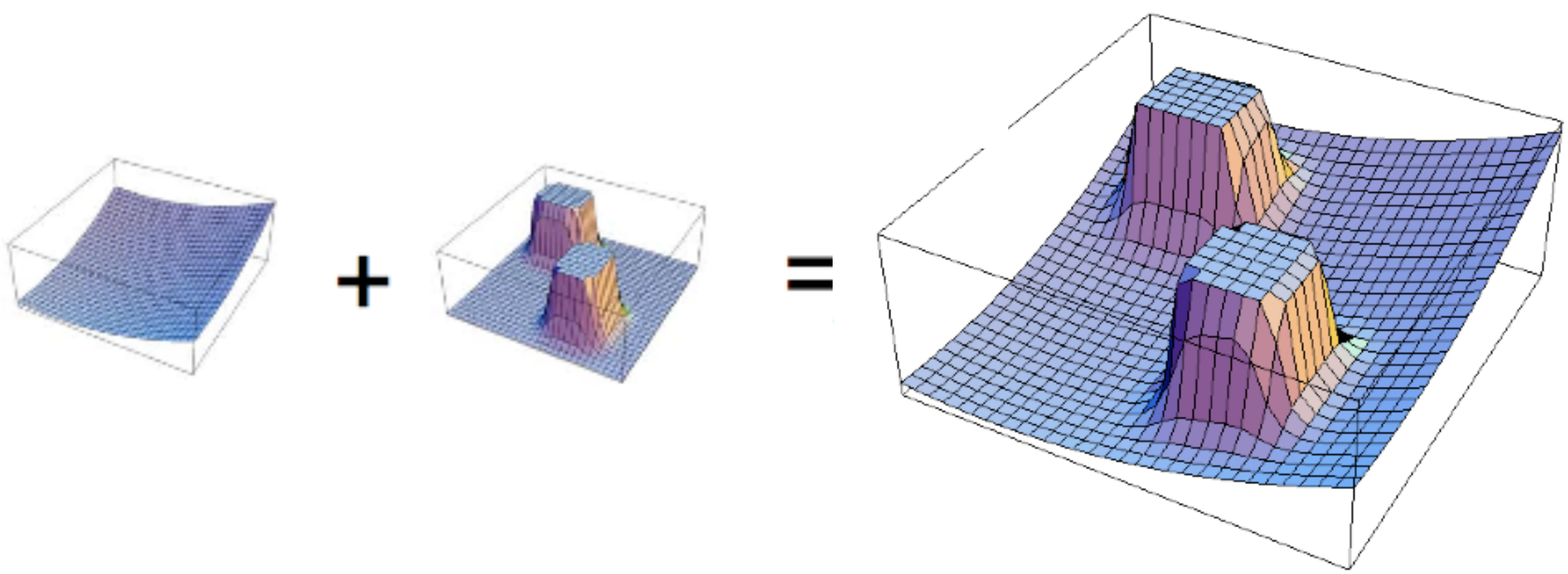




Potential Function

Resulting force $$F(q) = -\nabla(U_a(q) + U_r(q))$$

Iterative gradient descent planning $$q_{i+1} = q_i + \delta_i \frac{F(q)}{\|F(q)\|}$$

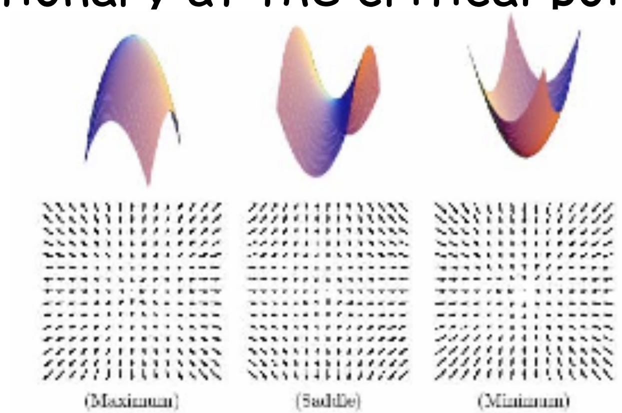


Potential Fields

- Simple way to get to the bottom, follow the gradient

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\nabla U(x, y) \qquad \dot{q} = -\nabla U(q)$$

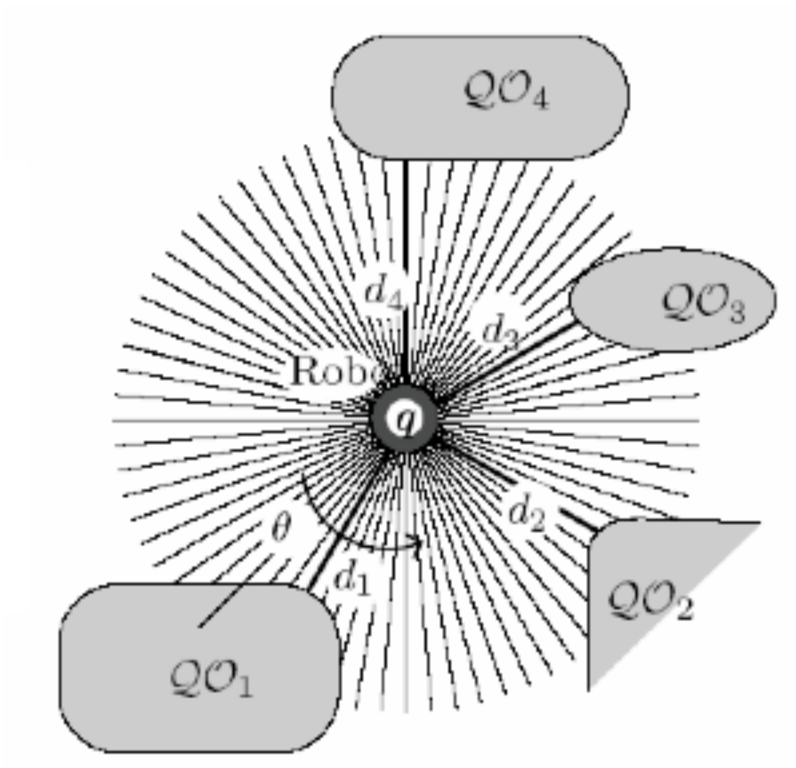
- i.e. Gradient descent strategy $q_{i+1} = q_i + \delta_i \frac{F(q)}{\|F(q)\|}$
- A critical, stationary point is such that $\nabla U(q) = 0$
- Equation is stationary at the critical point



- To check whether critical point is a minimum – look at the second order derivatives (Hessian for m -> n function)

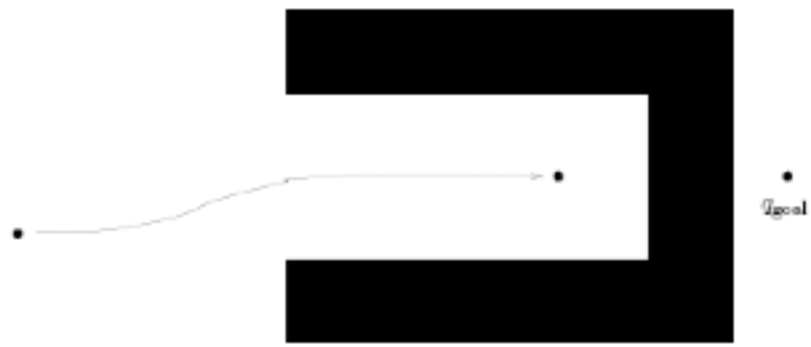
Computing Distances

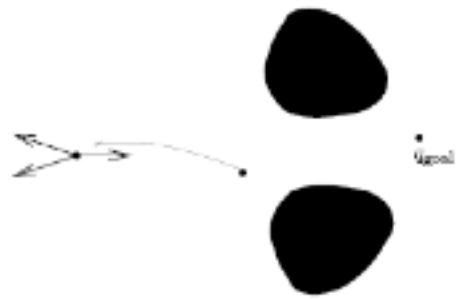
- Using sensor measurements



Potential Functions

- How do we know we have single global minimum ?





- If global minimum is not guaranteed, need to do something else then gradient descent
- Design functions in such a way that global minimum can be guaranteed

Potential function

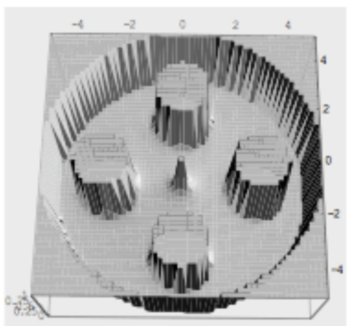
- Heuristics for escaping the local minima
- Can be used in local and global context
- Numerical techniques, Random walk methods

- Navigation functions (Rimon & Kodistchek, 92)
- Navigations in sphere worlds and worlds diffeomorphic to them

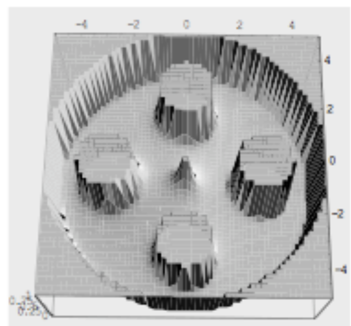
Navigation functions

$$\phi(q) = -\frac{d^2(q, q_{goal})}{[d(q, q_{goal})^{2k} + \beta(q)]^{1/k}} \qquad \beta(q) \quad \text{Obstacle term}$$

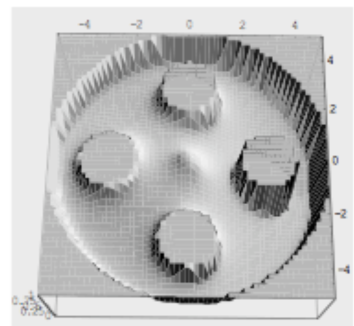
- For sufficiently large k – this is a navigation function [Rimon-Koditschek, 92]



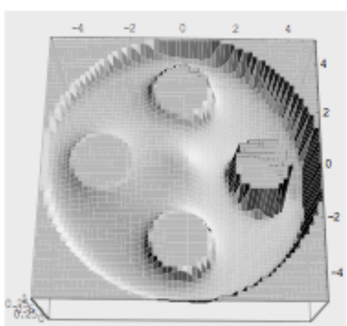
k=3

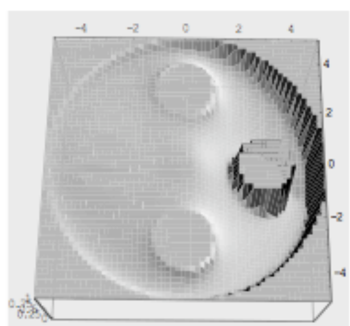


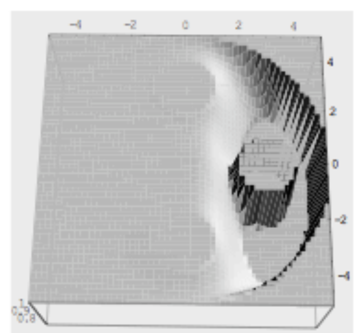
k=4



k=6

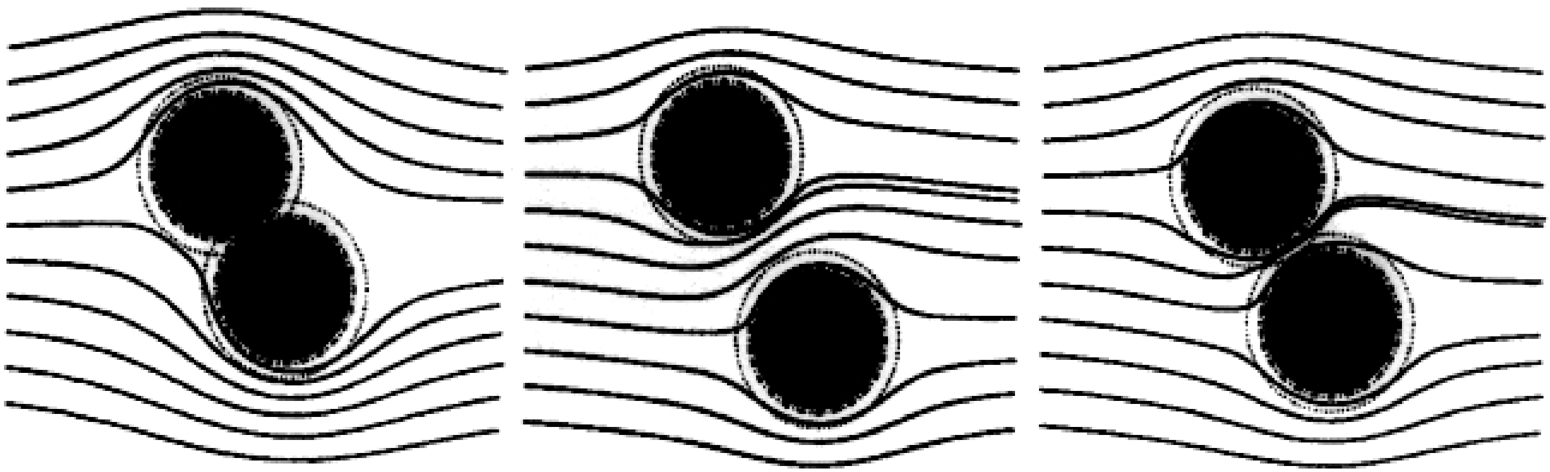






Potential Field Path Planning: Using Harmonic Potentials

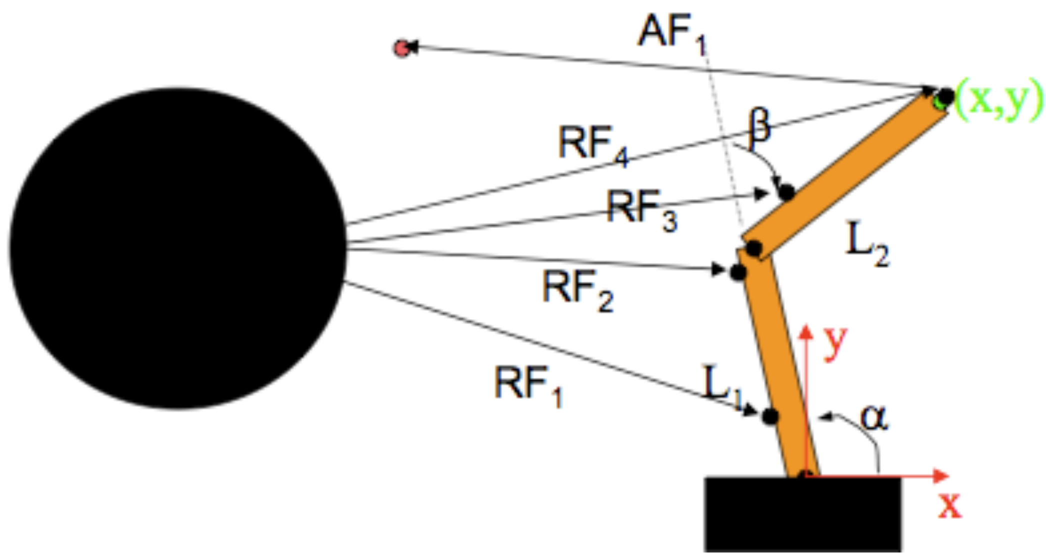
- Hydrodynamics analogy
  - robot is moving similar to a fluid particle following its



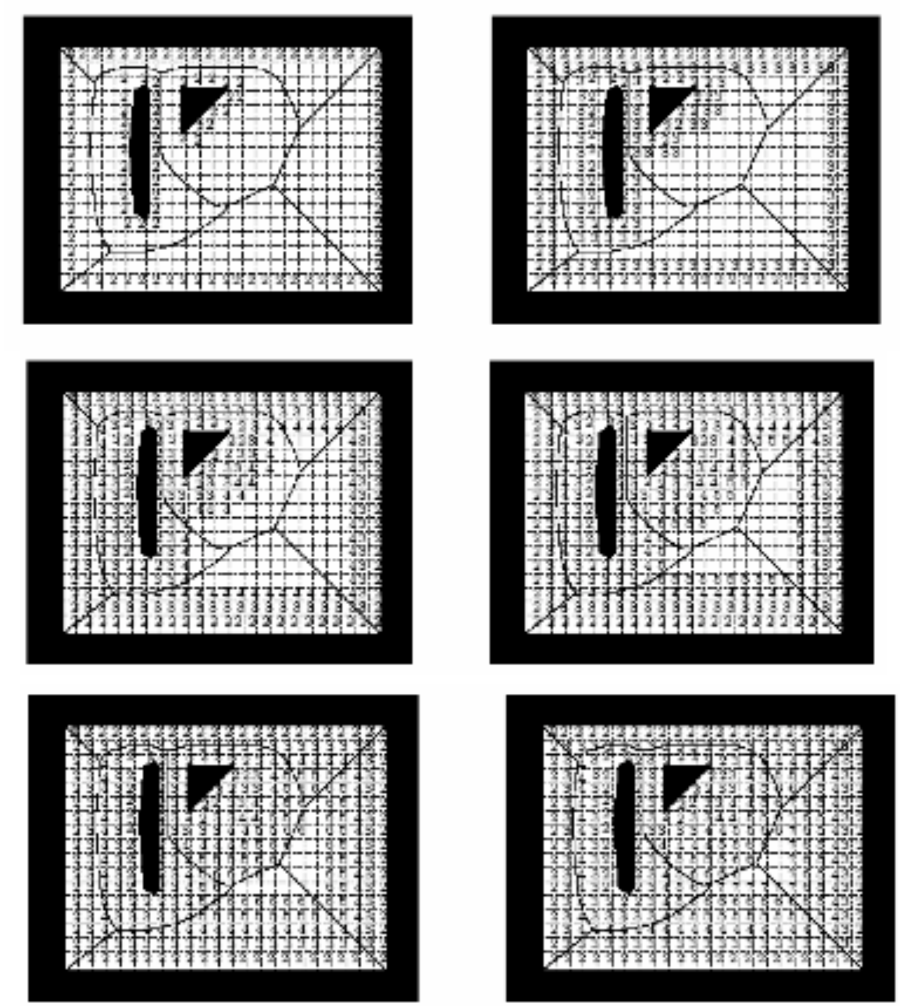
- Note:
  - Complicated, only simulation shown

Potential fields for Rigid Bodies

- So far robot was considered a point- gradient of the potential function – force acting on a point
- how to generalize to manipulators of objects ?
- Idea – forces acting on objects – forces acting on multiple points of the object (black board)
- http://www.cs.cmu.edu/~motionplanning/
- For robots, pick enough control points to pin down the robot – define forces in workspace – map them to configuration space



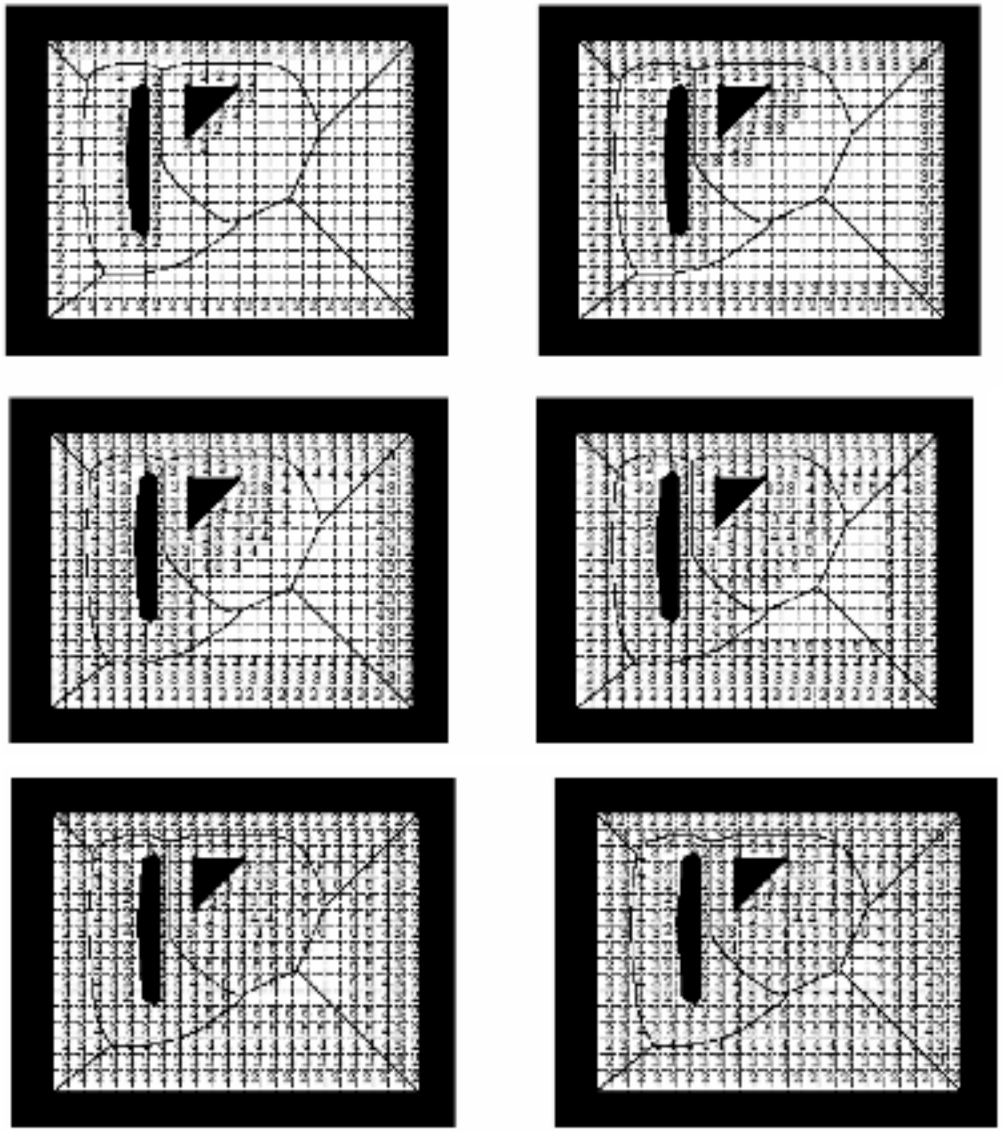
Brush fire algorithm



Create a queue Q of all pixels at the boundary of obstacles
For each, set the boundary to 1
And the free space to 0.

- For each element in the Q
- If d(q) = 0 set d(q) = 1+min d(q') of the neighbours which differ from 0
- Add all neighbours to the Q with d(q) = 0

Resulting map – distance to the nearest obstacle – gradient of the distance



Wave-front planner

- Apply brushfire algorithm starting from the goal
- Label goal 2, add all zeros labels to Q
- While Q non-empty
- Set d(q) = 1+ min d(q) of the neighbours which are different from 1 and 0
- Add all neighbours of the selected point to the Q

- The results is a distance for each point to the goal
- Follow gradient descent to the goal

Navigation functions - Discrete version

- Useful potential functions – for any starting point you will reach a goal: Navigation function (see previously how to define these in continuous spaces)
- In discrete state space – navigation function has to have some value for each state, consider grid
- Goal: will have zero potential
- Obstacles have infinite potential
- Example of useful navigation function: optimal cost to go to goal G* (example) , assuming that for each state

$$l(x,u) = 1$$

Wavefront planner

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

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

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

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

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

- Zeros should exist only in unreachable regions exist
- Ones for regions which cannot be reached
- Other book keeping methods can be used

Finding the path

- From any initial grid cell, move toward the cell
- with the lowest number – follow the steepest
- descent