Very highly recommended book to buy if you're even only half interested in geometry-meets-search-meets-AI-meets-robotics: *Robot Motion Planning* by Jean Claude Latombe (Kluwer, 1990)

Robot Motion Planning

Andrew W. Moore
Professor
School of Computer Science
Carnegie Mellon University
www.cs.cmu.edu/~awm
awm@cs.cmu.edu
412-268-7599

Slide 1

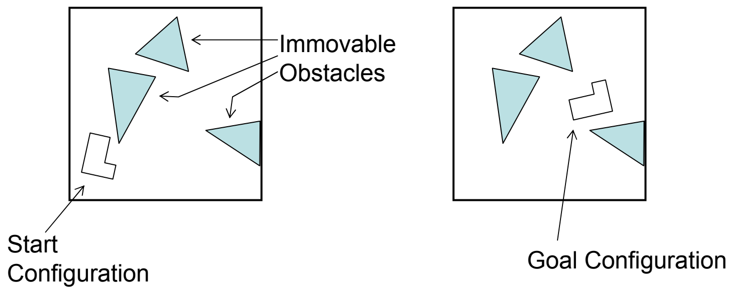
Let's Think About Automating Reasoning

- We've already seen
  - Reasoning with Constraints
  - State space search in discrete spaces
  - Reasoning with multiple agents
- Later (in this course) we'll see
  - Probabilistic Reasoning with Markov Decision Processes, Reinforcement Learning and HMMS
- But *NOW* let's think about

SPATIAL REASONING

Slide 2

Spatial Reasoning



Can’t we use our previous methods?

- Discrete Search? – Not a discrete problem
- CSP? – Not a natural CSP formulation
- Probabilistic? – Nope.

Slide 3

Robots

For our purposes, a robot is:

- A set of moving rigid objects called <u>LINKS</u> which are connected by <u>JOINTS</u>.

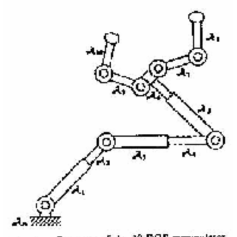


Fig. 11. Structure of the 10-DOF manipulator.

Typically, joints are REVOLUTE or PRISMATIC.

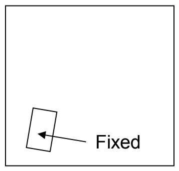
Such joints each give one DEGREE OF FREEDOM.

Given p DOFs, the configuration of the robot can be represented by p values $\boldsymbol{q} = (q_1\ q_2 \cdots q_p)$ where q_i is the angle or length of the i’th joint

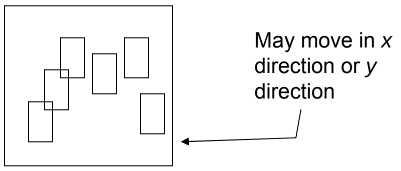
Slide 4

Free-Flying Polygons

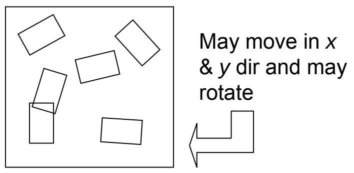
If part of the robot is fixed in the world, the joints are all the DOFs you're getting. But if the robot can be free-flying we get more DOFs.



0 DOFs



2 DOFs

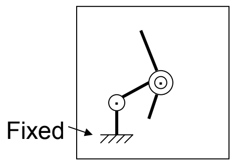


3 DOFs

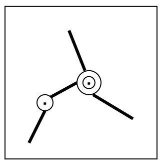
Question: How many DOFs for a polyhedron free flying in 3D space?

Slide 5

Examples

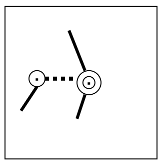


How many DOFs?



Free flying

How many DOFs?



Midline ■■■ must always be horizontal.

How many DOFs?

The configuration $\underset{\sim}{q}$ has one real valued entry per DOF.

Slide 6

Robot Motion Planning

An important, interesting, spatial reasoning problem.

- Let *A* be a robot with *p* degrees of freedom, living in a 2-D or 3-D world.
- Let *B* be a set of obstacles in this 2-D or 3-D world.
- Call a configuration LEGAL if it neither intersects any obstacles nor self-intersects.
- Given an initial configuration $\underset{\sim}{q}_{\text{start}}$ and a goal config $\underset{\sim}{q}_{\text{goal}}$, generate a continuous path of legal configurations between them, or report failure if no such path exists.

Slide 7

Configuration Space

Is the set of legal configurations of the robot. It also defines the topology of continuous motions

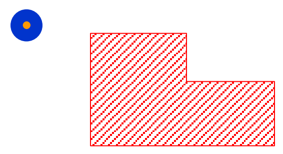
For rigid-object robots (no joints) there exists a transformation to the robot and obstacles that turns the robot into a single point. The C-Space Transform

Slide 8

Configuration Space Transform Examples

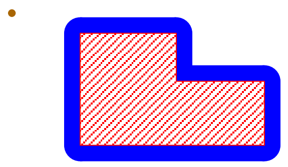
2-D World
2 DOFs

Where can I move this robot in the vicinity of this obstacle?



…is equivalent to…

Where can I move this point in the vicinity of this expanded obstacle?

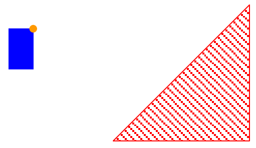


Slide 9

Configuration Space Transform Examples

2-D World
2 DOFs

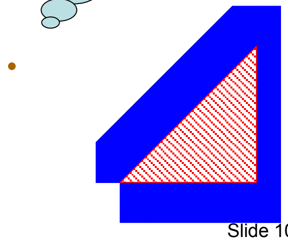
Where can I move this robot in the vicinity of this obstacle?



…is equivalent to…

Assuming you're not allowed to rotate

Where can I move this point in the vicinity of this expanded obstacle?



Slide 10

Configuration Space Transform Examples

2-D World
3 DOFs

We've turned the problem from "Twist and turn this 2-D polygon past this other 2-D polygon" into "Find a path for this point in 3-D space past this weird 3-D obstacle".

Why's this transform useful?

Because we can plan paths for points instead of polyhedra/polygons

Slide 11

Warning: Topology

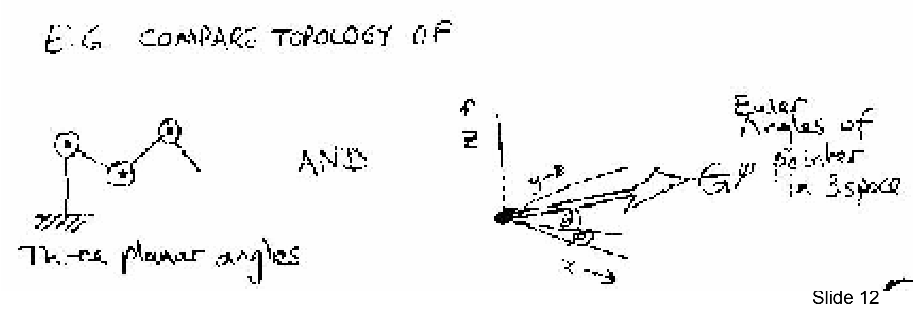
If you ever tried implementing these things you'll quickly start having to deal with spaces which aren't $\Re^N$.

E.G. The space of directions in 2-D space: SO(2).

AND The space of directions and orientations in 3-D space: SO(3).

AND Cartesian products thereof.

NOT JUST A CASE OF: "So what? Some of my real numbers happen to be angles".



Slide 12

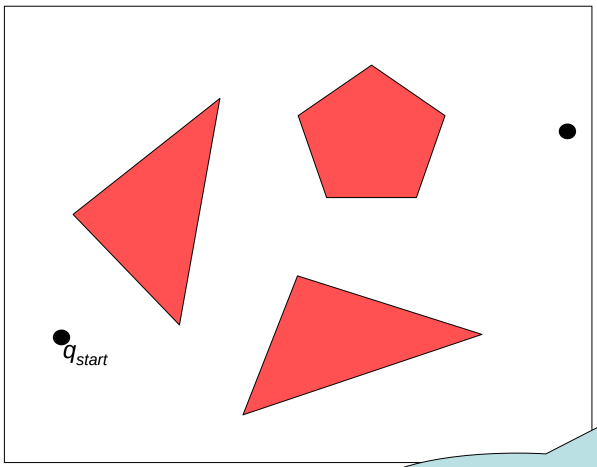
Robot Motion Planning Research

…Has produced four kinds of algorithms. The first is the Visibility Graph.

Slide 13

Visibility Graph

Suppose someone gives you a CSPACE with polygonal obstacles



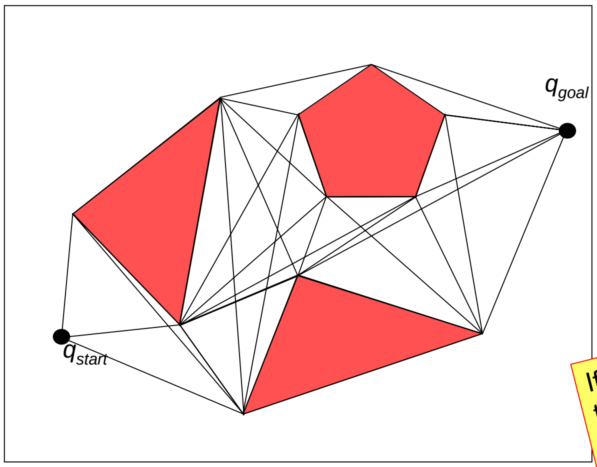
If there were no blocks, shortest path would be a straight line.

Else it must be a sequence of straight lines "shaving" corners of obstacles.

Obvious, but very awkward to prove

Slide 14

Visibility Graph Algorithm



1. Find all non-blocked lines between polygon vertices, start and goal.
2. Search the graph of these lines for the shortest path. (Guess best search algorithm?)

If there are *n* vertices, the easy algorithm is O(n^3). Slightly tougher O(n^2logn). O(n^2) in theory.

Slide 15

Visibility Graph Method

COMPLAINT

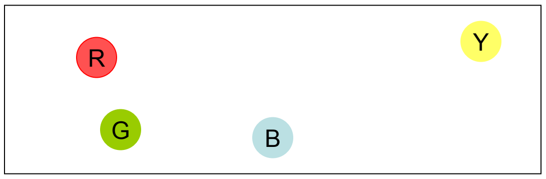
- Visibility graph method finds the shortest path.
- Bit it does so by skirting along and close to obstacles.
- Any error in control, or model of obstacle locations, and Bang! Screech!!

Who cares about optimality?

Perhaps we want to get a non-stupid path that steers as far from the obstacles as it can.

Slide 16

Voronoi Diagrams



Someone gives you some dots. Each dot is a different color.

You color in the whole of 2-D space according to this rule:

> "The color of any given point equals the color of the nearest dot."

The borders between your different regions are a VORNOI DIAGRAM.

> For *n* point in 2-D space the exact Voronoi diagram can be computed in time O(*n* log *n*).

Slide 17

Voronoi Diagram from Polygons instead of Points

Slide 18

Voronoi Diagram Methods for C-Space Motion Planning

- Compute the Voronoi Diagram of C-space.
- Compute shortest straightline path from start to any point on Voronoi Diagram.
- Compute shortest straightline path from goal to any point on Voronoi Diagram.
- Compute shortest path from start to goal along Voronoi Diagram.

Slide 19

Voronoi Diagrams

COMPLAINT

- Assumes polygons, and very complex above 2-D.

  Answer: very nifty approximate algorithms (see Howie Choset's work http://voronoi.sbp.ri.cmu.edu/~choset)
- This "use Voronoi to keep clear of obstacles" is just a heuristic. And can be made to look stupid:

Can you see how?

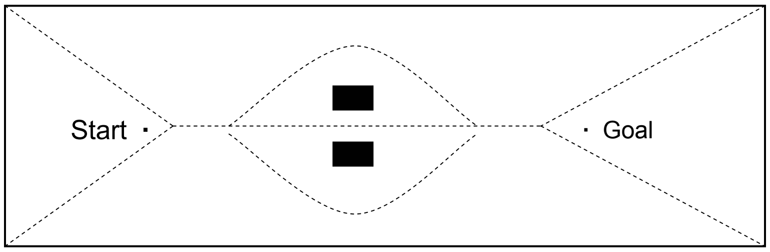
Slide 20

Voronoi Diagrams

COMPLAINT

- Assumes polygons, and very complex above 2-D.

  Answer: very nifty approximate algorithms (see Howie Choset’s work http://voronoi.sbp.ri.cmu.edu/~choset)
- This “use Voronoi to keep clear of obstacles” is just a heuristic. And can be made to look stupid:



Slide 21

Cell Decomposition Methods

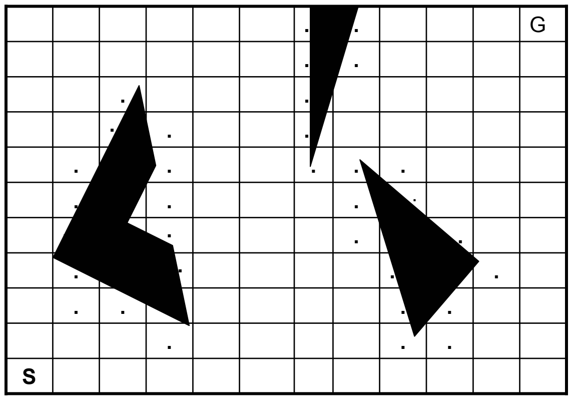
Cell Decomp Method One: Exact Decomp

- Break free space into convex exact polygons.

…But this is also impractical above 2-D or with non-polygons.

Slide 22

Approximate Cell Decomposition



- Lay down a grid
- Avoid any cell which intersects an obstacle
- Plan shortest path through other cells (e.g. with A*)

If no path exists, double the resolution and try again. Keep trying!!

Slide 23

Variable Resolution “Approximate and Decompose”

Slide 24

Variable Resolution “Approximate and Decompose”

Slide 25

Approximate Cell Decom tion

COMPLAINTS

- ❑ Not so many complaints. This is actually used in practical systems.

But

- o Not exact (no notion of “best” path)
- o Not complete: doesn’t know if problem actually unsolvable
- o Still hopeless above a small number of dimensions?

Slide 26

Potential Methods

Define a function $u\left(\underset{\sim}{q}\right)$

$u:\text{Configurations}\rightarrow\Re$

Such that

$u\rightarrow$ huge as you move towards an obstacle

$u\rightarrow$ small as you move towards the goal

Write $d_g\left(\underset{\sim}{q}\right)=$ distance from $\underset{\sim}{q}$ to $\underset{\sim}{q}$ goal

$d_i\left(\underset{\sim}{q}\right)=$ distance from $\underset{\sim}{q}$ to nearest obstacle

---

One definition of u: $u(q)=d_i(q)-d_g(q)$

Preferred definition: $u(q)=\frac{1}{2}\sum\left(d_g(q)\right)^2+\frac{1}{2}\eta\frac{1}{d_i(q)^2}$

SIMPLE MOTION PLANNER: Steepest Descent on *u*

Slide 27

Potential Field Example

Slide 28

Spot the Obvious Problem!

Solution I:

Use special local-minimum-free potential fields (Laplace equations can do this) – But very expensive to compute

Solution II:

When at a local minimum start doing some searching - example soon

Slide 29

Comparison

	Potential Fields	Approx Cell Decomp	Voronoi	Visibility
Practical above 2 or 3 D?				
Practical above 8 D?				
Fast to Compute?				
Usable Online?				
Gives Optimal?				
Spots Impossibilities?				
Easy to Implement?				

Slide 30

And now…..

Let’s look at one of the state-of-the-art motion planners.

Latombe’s Numerical Potential Field Method

- Combines Cell Decomposition and Potential Fields
- Key insight: Compute an “optimal” potential field in world coordinate space (not config space)
- Define a C-space potential field in terms of world-space potential field.

Slide 31

N.P.F. Step I

Get given a problem.

You have a world ω which is a 2-D or 3-D space.

There are obstacles and a robot.

Robot may be fixed or free-flying, may be jointed or totally rigid.

You have a start configuration.

Goal spec. is more flexible than mere goal configuration:

You can specify between 1 and *P* points (*P* = number of DOFs) in world space, and state that various points on the robot should meet those world points. E.G.

“I want my tip to end up here”

or

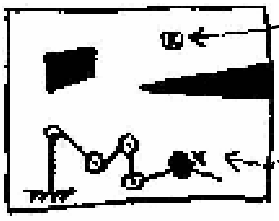
“I want my tip here, my wrist there, and my shoulder way over there.”

Slide 32

N.P.F. Step II

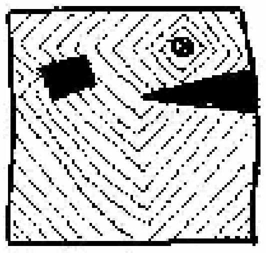
For each given world-goal-point we compute an optimal potential field in worldspace.

Example:



world-goal-point for *x* joint on arm

Optimal potential field can simply be "shortest distance".



Computed by discretizing world-space on a fine grid.

Q: Why is fine grid not too expensive now?

Slide 33

Skeletons in the World Space

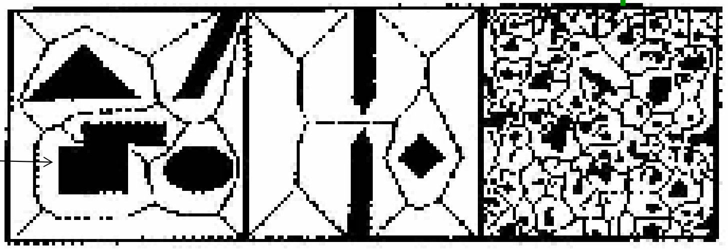


Fig. 4 Examples of worldspace skeletons.

- Easy to skeletonize with a grid

Then what?

1. Extend the skeleton to join to the goal
2. Compute least-cost-to-goal for all points on skeleton
3. Define potential field by traveling downhill to local part of skeleton.

The exact definition of this is unclear. It is NOT "shortest path to skeleton" and not "shortest path to goal" but a kind of combination. Defined inductively by first de**** all points 1 away from skeleton as 1 + least adjacent skeleton-cell cost. Then points next to them, etc., etc.

Slide 34

Potential from the Skeleton Method

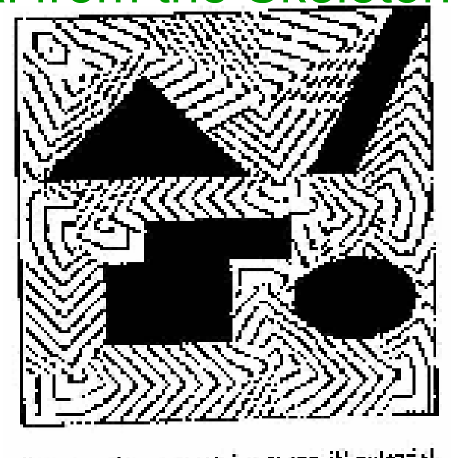


Fig. 6 Contours of improved W-potential

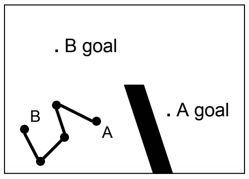
Remember, we do one of these for each world-goal-point.

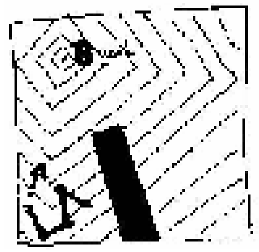
Slide 35

N.P.F. IV

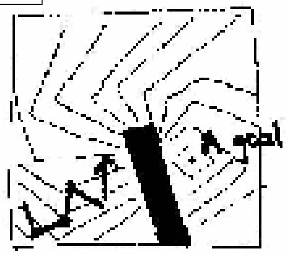
Compute a potential field for configurations.

Remember, for each goal point we have computed a world-space potential.





"B" point contours



"A" point contours

Slide 36

N.P.F. IV

Combining world space potentials into a C-space potential.

If $W_i(\text{point}_i(\underset{\sim}{q})$ is the potential field for the i'th distinguished point on the robot…

We could define

$$\mu\left(\underset{\sim}{q}\right) = \sum_{i.e.distinguished} W_i\left(\text{point}_i\left(\underset{\sim}{q}\right)\right)$$

or many other combination methods, e.g.

$$\mu\left(\underset{\sim}{q}\right) = \max_{i.e.distinguished} W_i\left(\text{point}_i\left(\underset{\sim}{q}\right)\right)$$

Empirically, this is preferred

Slide 37

NPF V

We're pretty much done!

Perform gradient descent from the start configuration with your C-space potential field…

Until…

- You get to the goal config
- You find you're in a local minimum.

Sadly, although the worldspace potentials were minimum-free, the combination is not.

What to do in a local minimum??

1. Best first search
2. Random search

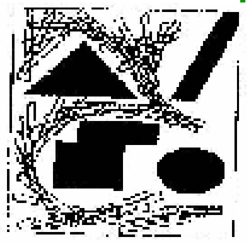
Slide 38

Best First Search with NPF

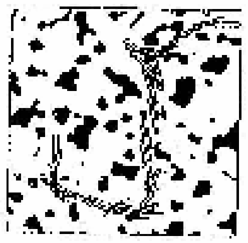
- Remember all the C-space places you've visited in the search (search on a fine C-space grid).
- When you expand a grid point consider all its neighbors on the grid.
- Always expand gridpoint with lowest C-space potential value.
  - During most of search runs quickly down gradient.
  - When in a local minimum, must wait for minimum to "fill up".

Slide 39

Empirical Results


[illegible]

1/10 second to solve on a SPARC. (In 1992 claimed to be fastest algorithm by 2 orders of magnitude, and three orders of nuns.)


[illegible]

5 seconds to solve.

3 degrees of freedom.

Slide 40

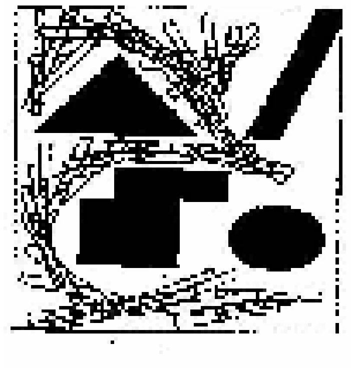
Randomized Search with NPF

- On each step choose from a random set of, say, 100 neighbors on C-space grid.
- If none reduce the potential, current state is recorded as being a local minimum.
    - If was local minimum, perform a random walk for *t* timesteps to escape (one hopes)
- Keep doing this, building up a graph of "adjacent" local minima.
- Continue until one local minimum turns out to be the goal.

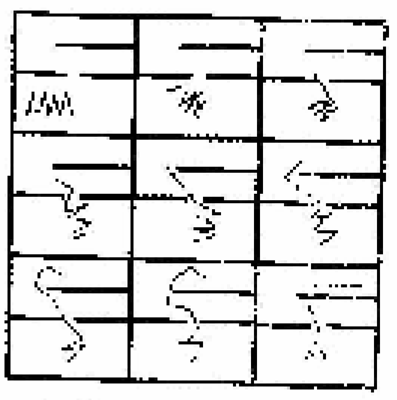
HOW? Many details, and a slightly depressingly large number of magic parameters are involved.

Slide 41

Randomized Search Results



10 secs. (Slower than best first)



2 mins. 8 DOF

Spectacular result.

Slide 42

Randomized Search Results

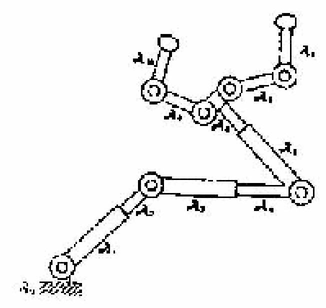


Fig. 11 Structure of the 10-DOF manipulator.

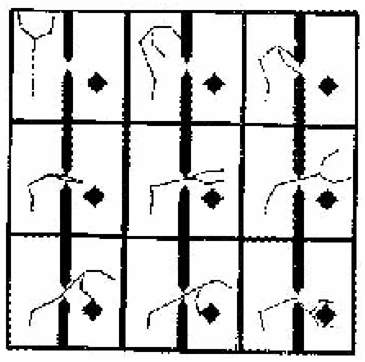


Fig. 12. Path generated for the 10 DOF manipulator.

10 DOF 3 mins.

31 DOF manipulator in 3-D workspace 15 mins.

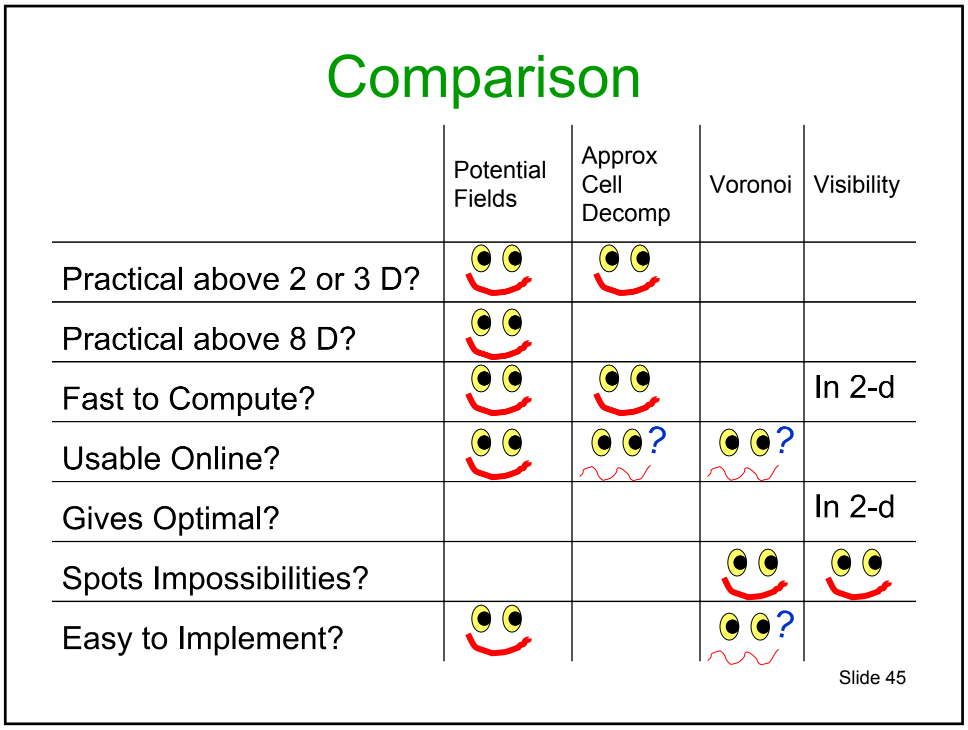
Slide 43

What Should You Know

- How to define configuration space obstacles
- The basic idea behind, and Pros and Cons of
  - VISIBILITY GRAPH METHODS
  - VORONOI METHODS
  - CELL DECOMPOSITION METHODS
  - POTENTIAL FIELD METHODS
- Latombe's trick of using optimal world-space potentials for good C-space potentials.

See also the book Robot Motion Planning by Latombe (Klumer 1990). Russell & Norvig has a rather brief chapter on the subject.

Slide 44



Comparison

	Potential Fields	Approx Cell Decomp	Voronoi	Visibility
Practical above 2 or 3 D?	😊	😊		
Practical above 8 D?	😊			
Fast to Compute?	😊	😊		In 2-d
Usable Online?	😊	😐?	😐?	
Gives Optimal?				In 2-d
Spots Impossibilities?			😊	😊
Easy to Implement?	😊		😐?	

Slide 45