Motion Planning

(It's all in the discretization)

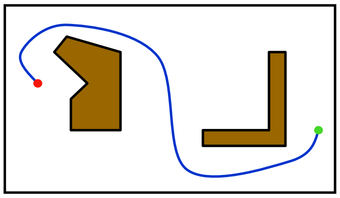
R&N: Chap. 25 gives some background

Motion planning is the ability for an agent to compute its own motions in order to achieve certain goals. All autonomous robots and digital actors should eventually have this ability

Digital Actors

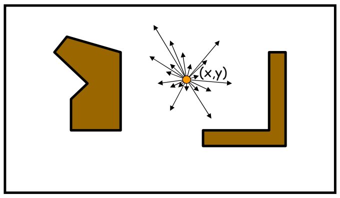
- video 1
- video 2

Basic problem



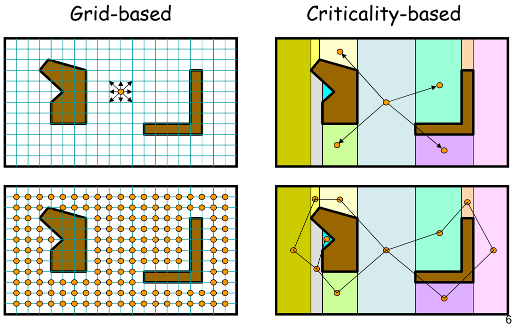
- Point robot in a 2-dimensional workspace with obstacles of known shape and position
- Find a collision-free path between a start and a goal position of the robot

Basic problem

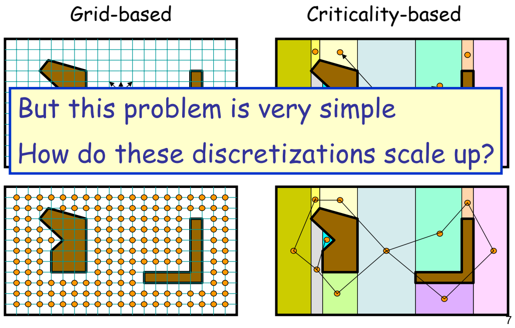


- Each robot position (x,y) can be seen as a state
- → Continuous state space
- Then each state has an infinity of successors
- We need to discretize the state space

Two Possible Discretizations



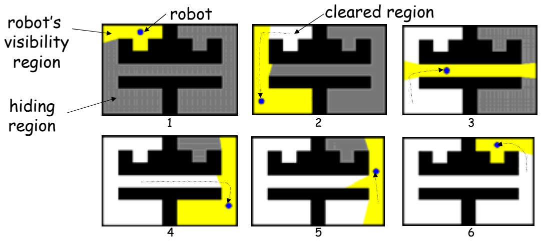
Two Possible Discretizations



But this problem is very simple

How do these discretizations scale up?

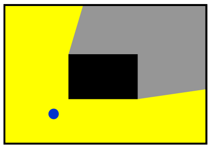
Intruder Finding Problem



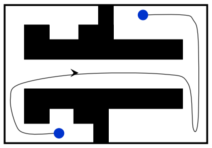
- A moving intruder is hiding in a 2-D workspace
- The robot must "sweep" the workspace to find the intruder
- Both the robot and the intruder are points

Does a solution always exist?

No !

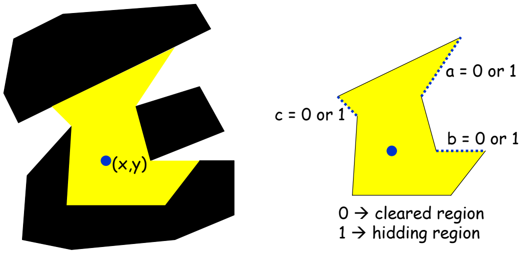


Easy to test:
"Hole" in the workspace



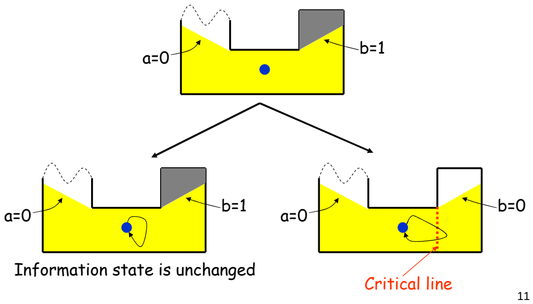
Hard to test:
No "hole" in the workspace

Information State



- Example of an information state = (x,y,a=1,b=1,c=0)
- An initial state is of the form (x,y,1, 1, ..., 1)
- A goal state is any state of the form (x,y,0,0, ..., 0)

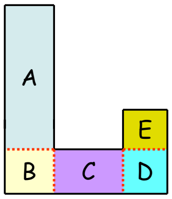
Critical Line



Information state is unchanged

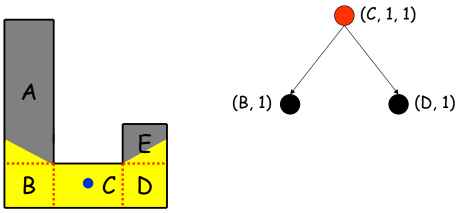
Critical line

Criticality-Based Discretization

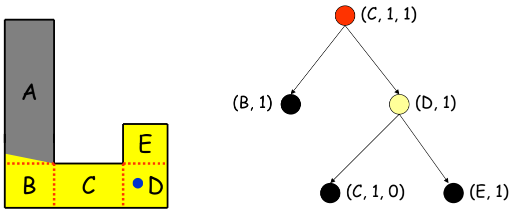


Each of the regions A, B, C, D, and E consists of "equivalent" positions of the robot, so it's sufficient to consider a single position per region

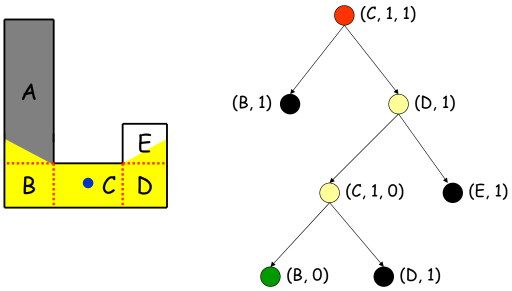
Criticality-Based Discretization



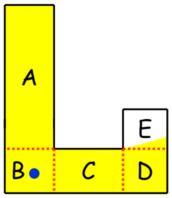
Criticality-Based Discretization

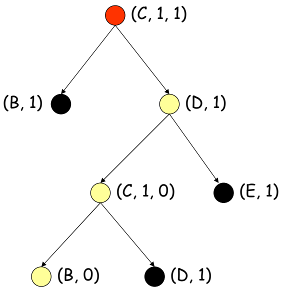


Criticality-Based Discretization



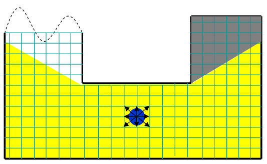
Criticality-Based Discretization





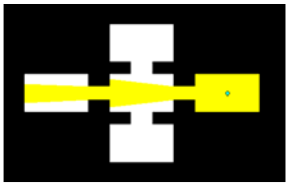
Much smaller search tree than with grid-based discretization !

Grid-Based Discretization



- Ignores critical lines → Visits many "equivalent" states
- Many information states per grid point
- Potentially very inefficient

Example of Solution

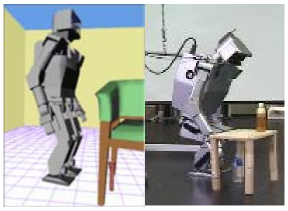


But ...

Criticality-based discretization does not scale well in practice when the dimensionality of the continuous space increases

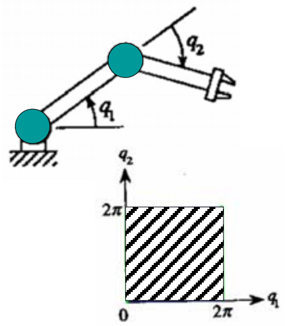
(It becomes prohibitively complex to define and compute)

Motion Planning for an Articulated Robot



Find a path to a goal configuration that satisfies various constraints: collision avoidance, equilibrium, etc...

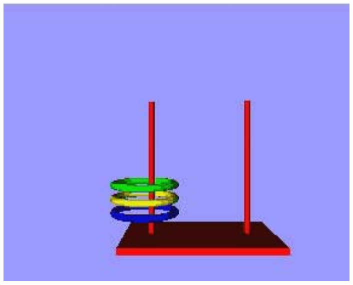
Configuration Space of an Articulated Robot



- A configuration of a robot is a list of non-redundant parameters that fully specify the position and orientation of each of its bodies
- In this robot, one possible choice is: (q_1, q_2)

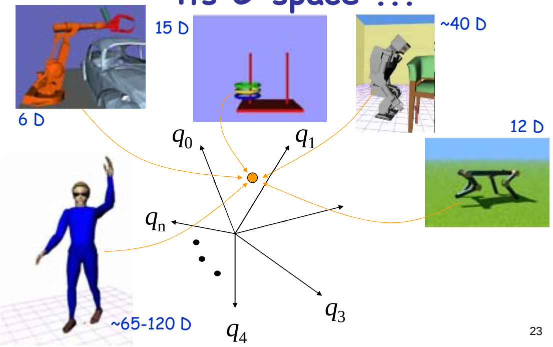
The configuration space (C-space) has 2 dimensions

How many dimensions has the C-space of these 3 rings?

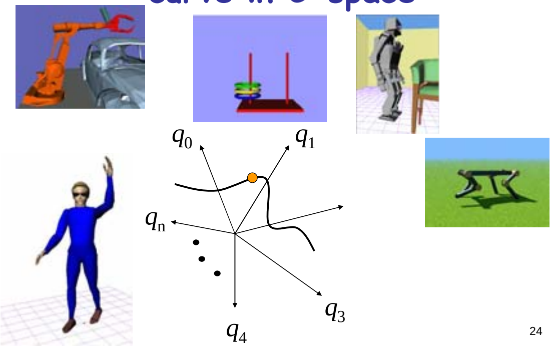


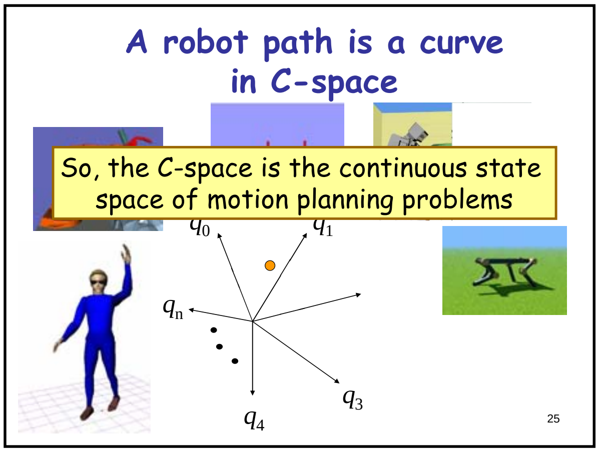
Answer: 3×5 = 15

Every robot maps to a point in its C-space ...



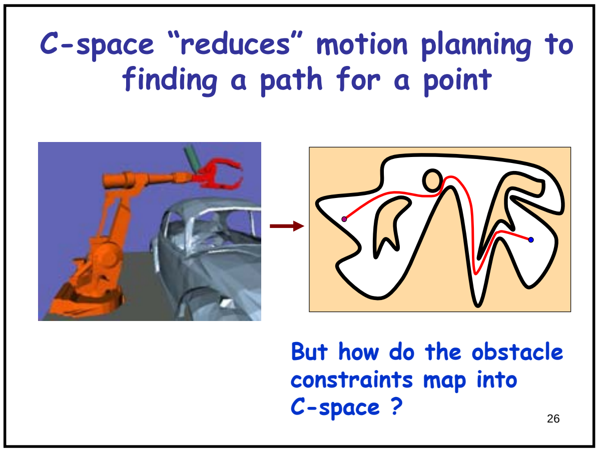
... and every robot path is a curve in C-space





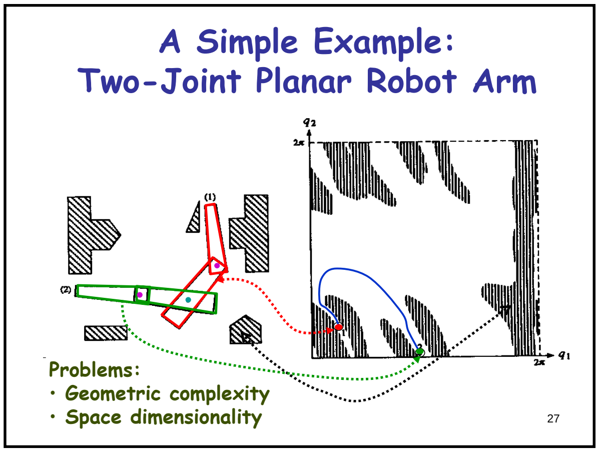
A robot path is a curve in C-space

So, the C-space is the continuous state space of motion planning problems



C-space "reduces" motion planning to finding a path for a point

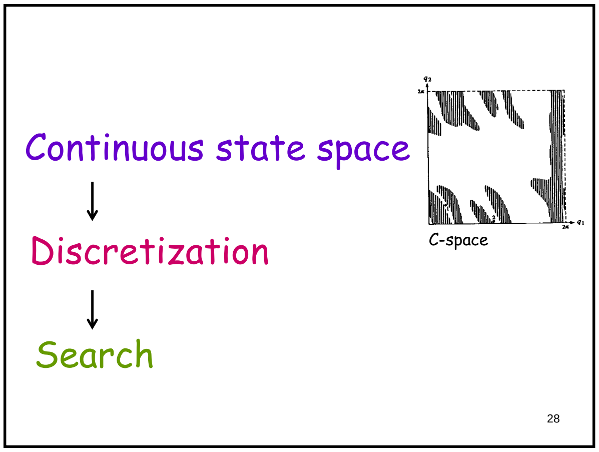
But how do the obstacle constraints map into C-space ?



A Simple Example: Two-Joint Planar Robot Arm

Problems:

- Geometric complexity
- Space dimensionality



Continuous state space

↓

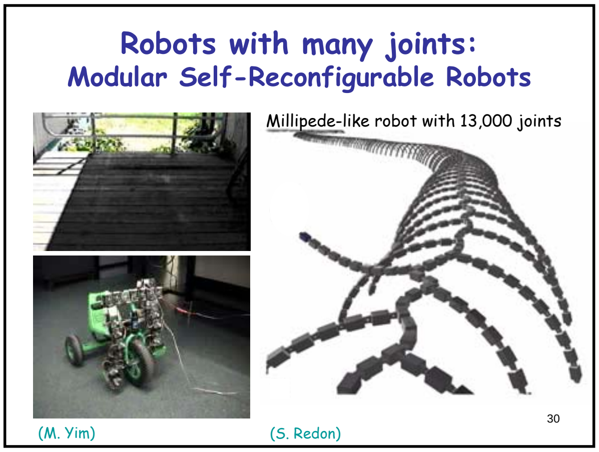
Discretization

↓

Search

About Discretization

- Dimensionality + geometric complexity
  → Criticality-based discretization turns out to be prohibitively complex
- Dimensionality
  → Grid-based discretization leads to impractically large state spaces for dim(C-space) > 6
  Each grid node has 3^n-1 neighbors, where n = dim(C-space)

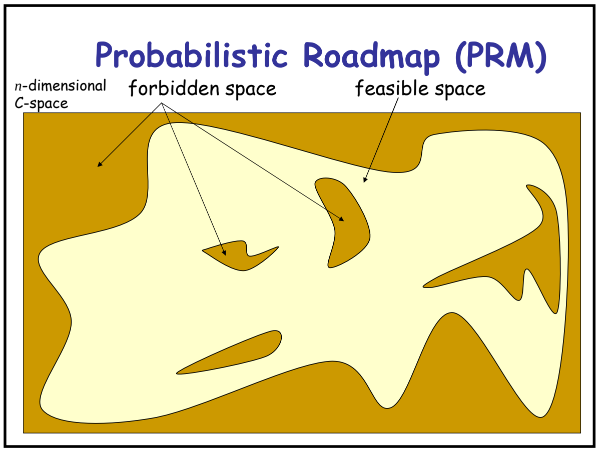


Robots with many joints: Modular Self-Reconfigurable Robots

Millipede-like robot with 13,000 joints

(M. Yim)

(S. Redon)

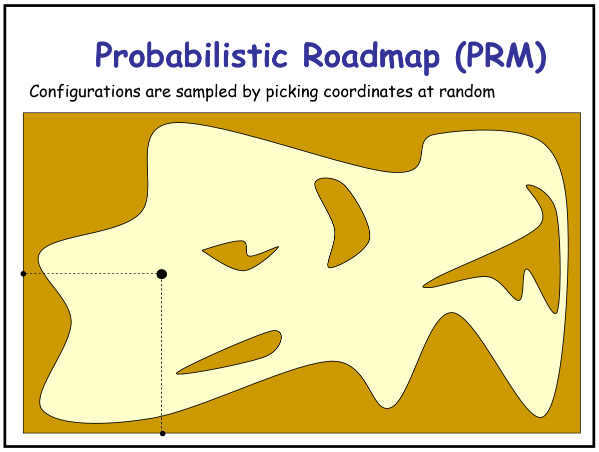


Probabilistic Roadmap (PRM)

n-dimensional *C*-space

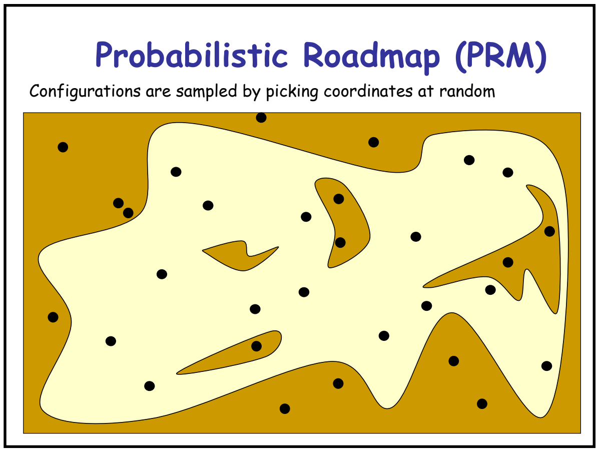
forbidden space

feasible space



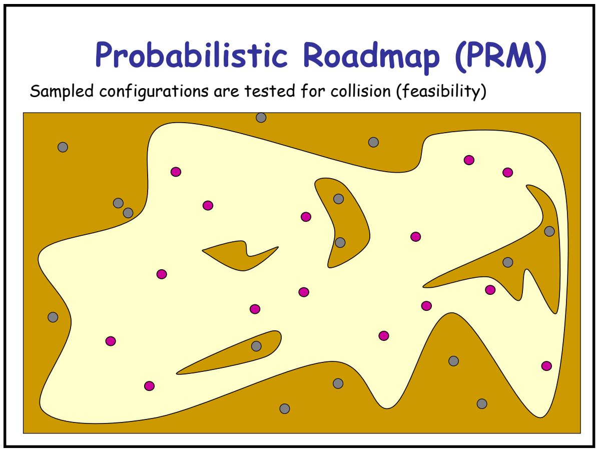
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



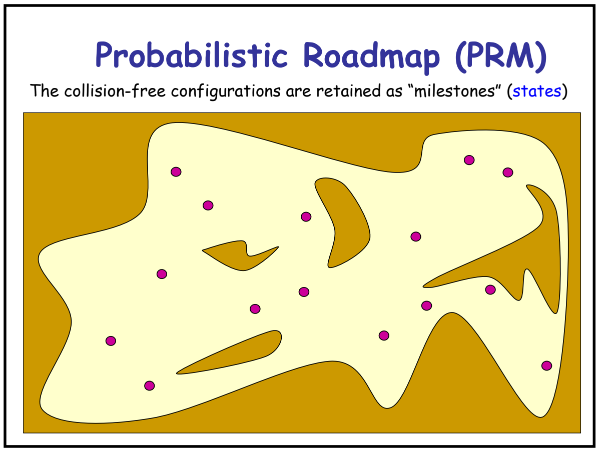
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



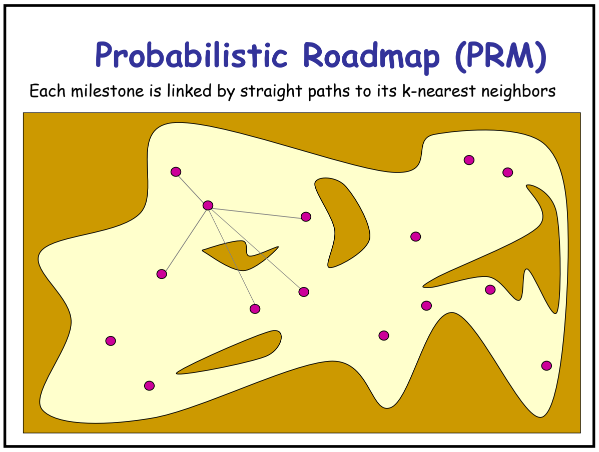
Probabilistic Roadmap (PRM)

Sampled configurations are tested for collision (feasibility)



Probabilistic Roadmap (PRM)

The collision-free configurations are retained as “milestones” (states)

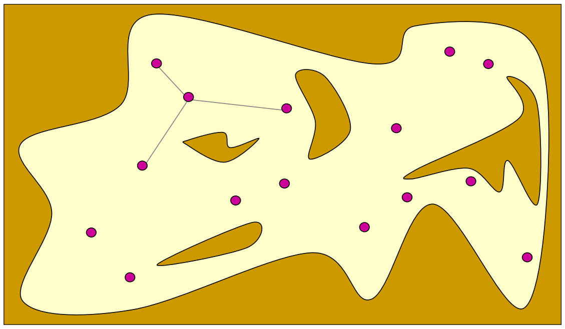


Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors

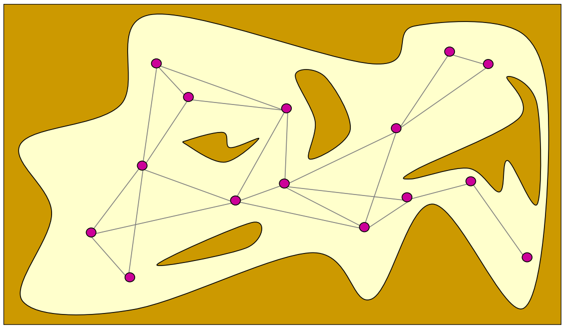
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



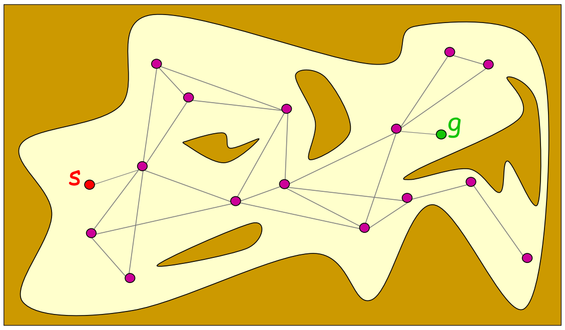
Probabilistic Roadmap (PRM)

The collision-free links are retained to form the PRM (state graph)



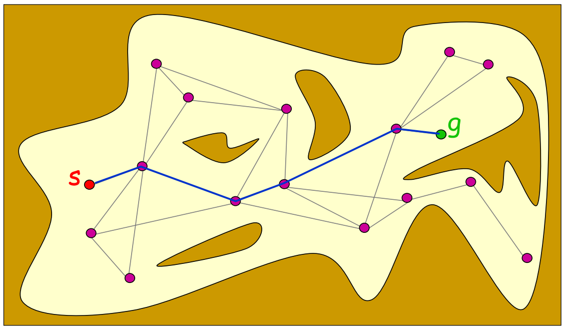
Probabilistic Roadmap (PRM)

The start and goal configurations are connected to nodes of the PRM



Probabilistic Roadmap (PRM)

The PRM is searched for a path from s to g



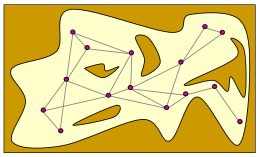
Continuous state space

↓

Discretization

↓

Search A*

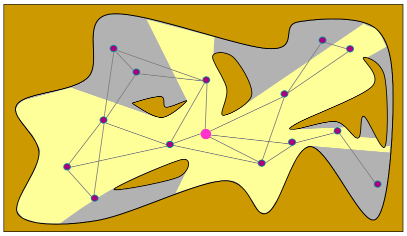


Why Does PRM Work?

Because most feasible spaces verifies some good geometric (visibility) properties

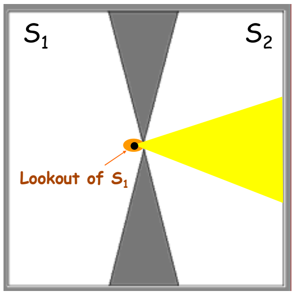
Why Does PRM Work?

In most feasible spaces, every configuration "sees" a significant fraction of the feasible space



→ A relatively small number of milestones and connections between them are sufficient to cover most feasible spaces with high probability

Narrow-Passage Issue



The lookout of a subset S of the feasible space is the set of all configurations in S from which it is possible to "see" a significant fraction of the feasible space outside S

The feasible space is **expansive** if all of its subsets have a large lookout

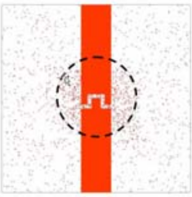
Probabilistic Completeness of a PRM Motion Planner

In an expansive feasible space, the probability that a PRM planner with uniform sampling strategy finds a solution path, if one exists, goes to 1 exponentially with the number of milestones (~ running time)

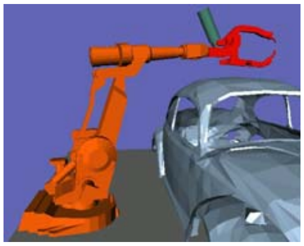
A PRM planner can't detect that no path exists. Like A*, it must be allocated a time limit beyond which it returns that no path exists. But this answer may be **incorrect**. Perhaps the planner needed more time to find one !

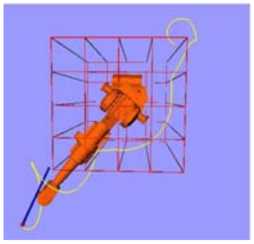
Sampling Strategies

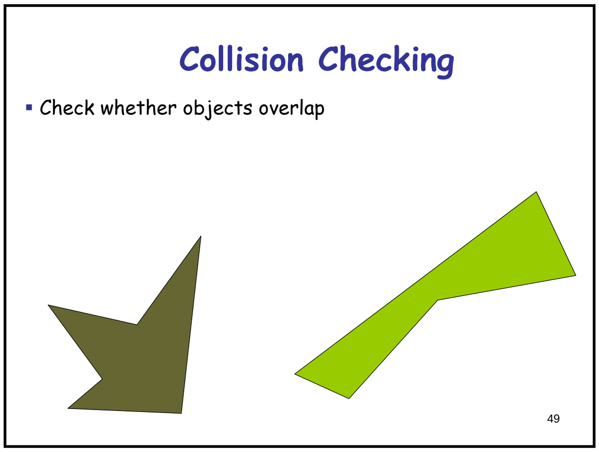
- Issue: Where to sample configurations? That is, which probabilistic distribution to use?
- Example: Two-stage sampling strategy:
  1. Construct initial PRM with uniform sampling
  2. Identify milestones that have few connections to their close neighbors
  3. Sample more configurations around them



→ Greater density of milestones in "difficult" regions of the feasible space

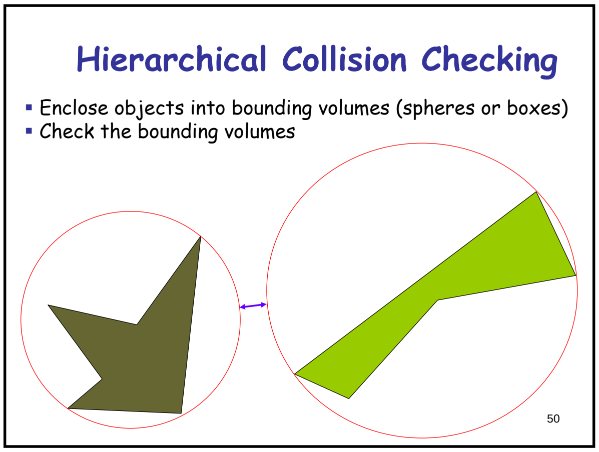






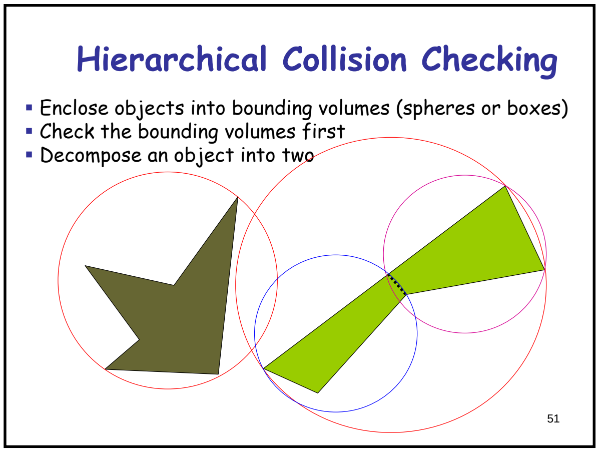
Collision Checking

- Check whether objects overlap



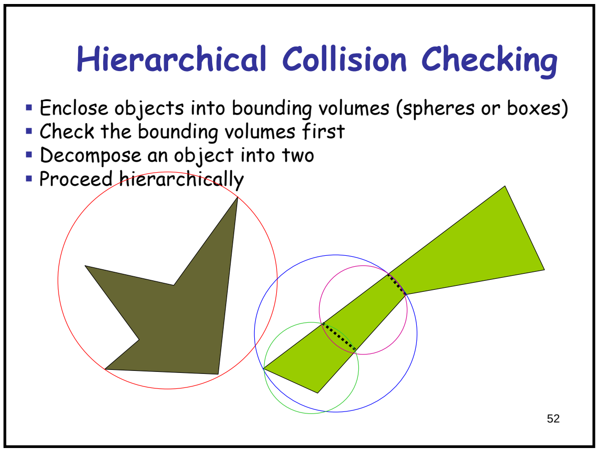
Hierarchical Collision Checking

- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes



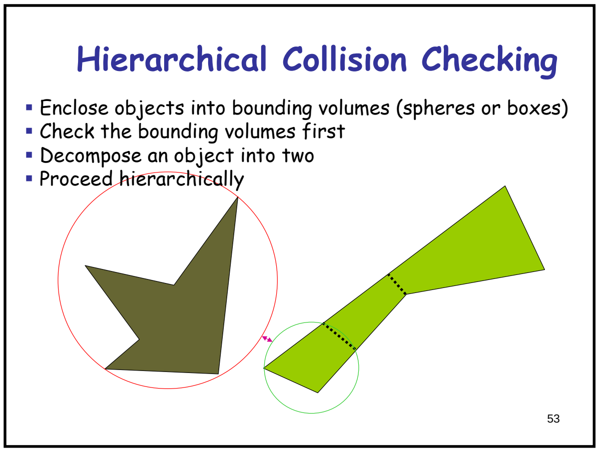
Hierarchical Collision Checking

- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two



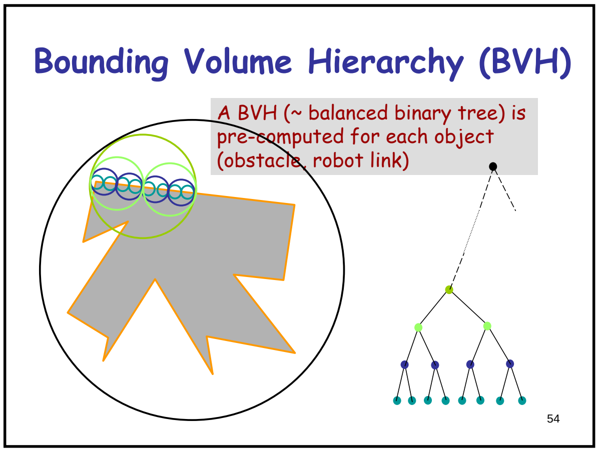
Hierarchical Collision Checking

- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two
- Proceed hierarchically



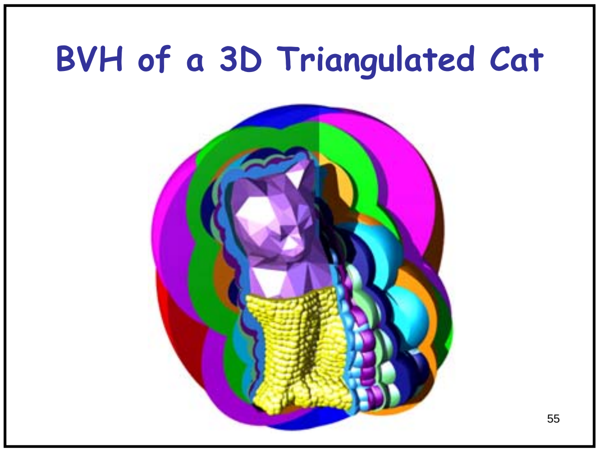
Hierarchical Collision Checking

- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two
- Proceed hierarchically



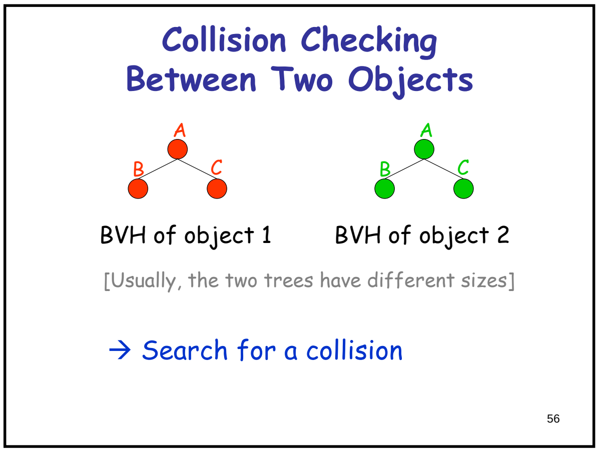
Bounding Volume Hierarchy (BVH)

A BVH (~ balanced binary tree) is pre-computed for each object (obstacle, robot link)



BVH of a 3D Triangulated Cat

55



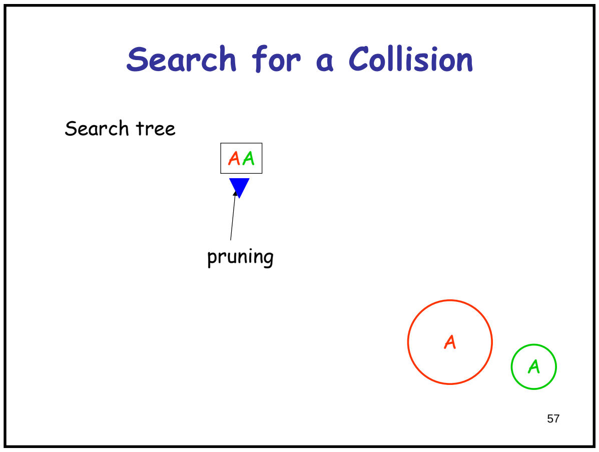
Collision Checking Between Two Objects

BVH of object 1 BVH of object 2

[Usually, the two trees have different sizes]

→ Search for a collision

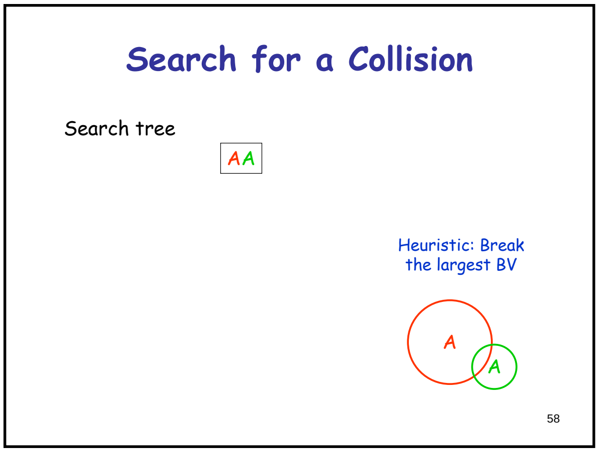
56



Search for a Collision

Search tree

57

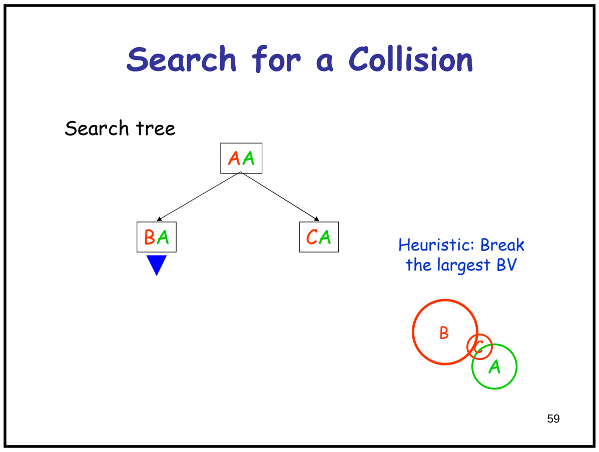


Search for a Collision

Search tree

Heuristic: Break the largest BV

58

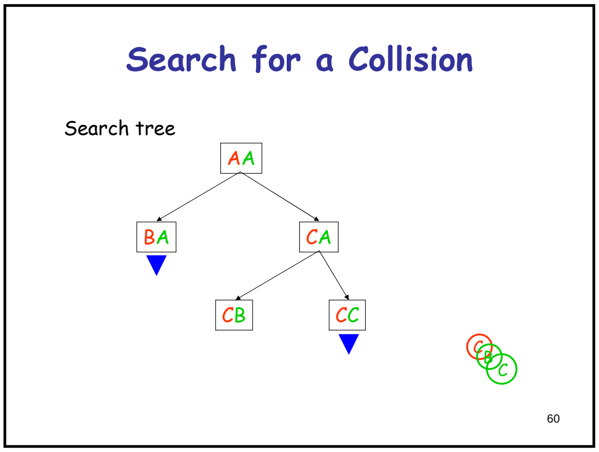


Search for a Collision

Search tree

Heuristic: Break the largest BV

59

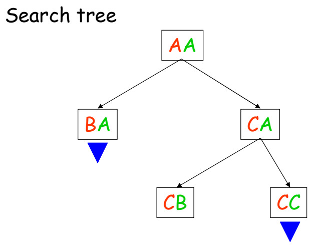


Search for a Collision

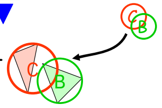
Search tree

60

Search for a Collision



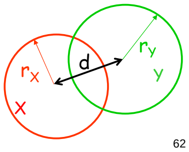
If two leaves of the BVH's overlap (here, C and B) check their content for collision



Search Strategy

- If there is no collision, all paths must eventually be followed down to pruning or a leaf node
- But if there is collision, one may try to detect it as quickly as possible
- → Greedy best-first search strategy with
  $f(N) = h(N) = d/(r_X+r_Y)$

[Expand the node XY with largest relative overlap (most likely to contain a collision)]



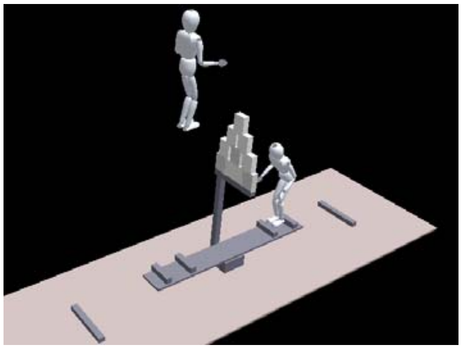
So, to discretize the state space of a motion planning problem, a PRM planner performs **thousands of auxiliary searches** (sometimes even more) to detect collisions !

But from an outsider's point of view the search of the PRM looks like the main search

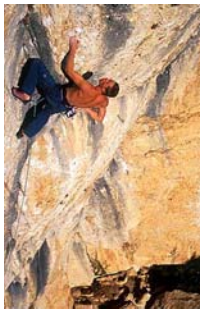
Fortunately, hierarchical collision checkers are quite fast

On average, over 10,000 collision checks per second for two 3-D objects each described by 500,000 triangles, on a contemporary PC

Checks are much faster when the objects are either neatly separated (→ early pruning) or neatly overlapping (→ quick detection of collision)



Free-Climbing Robot



LEMUR IIb robot (created by NASA/JPL)

Only friction and internal degrees of freedom are used to achieve equilibrium

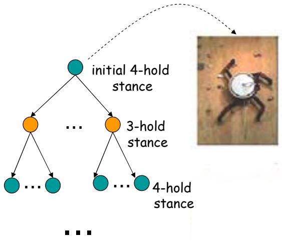
[Bretl, 2003]

Two Levels of Planning

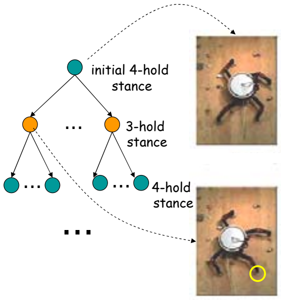
1) One-step planning:
Plan a path for moving a foot/hand from one hold to another
Can be solved using a PRM planner

2) Multi-step planning:
Plan a sequence of one-step paths
Can be solved by searching a stance space

Multi-Step Planning

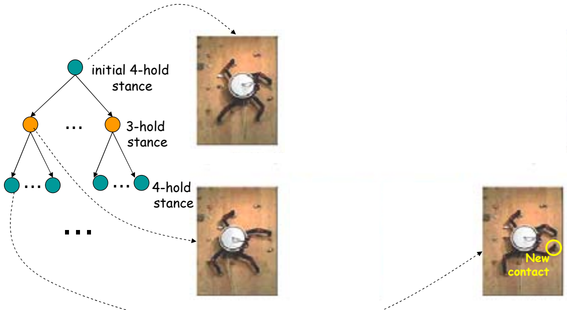


Multi-Step Planning



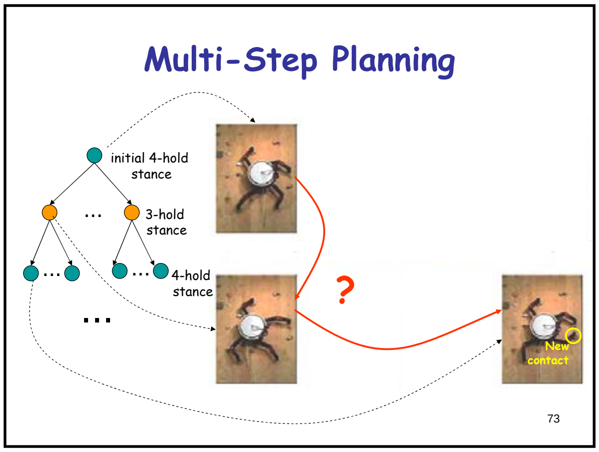
4 possible 3-hold stances

Multi-Step Planning

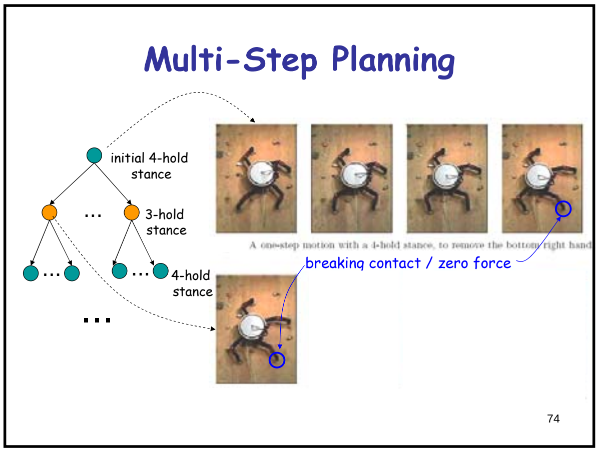


several candidate 4-hold stances

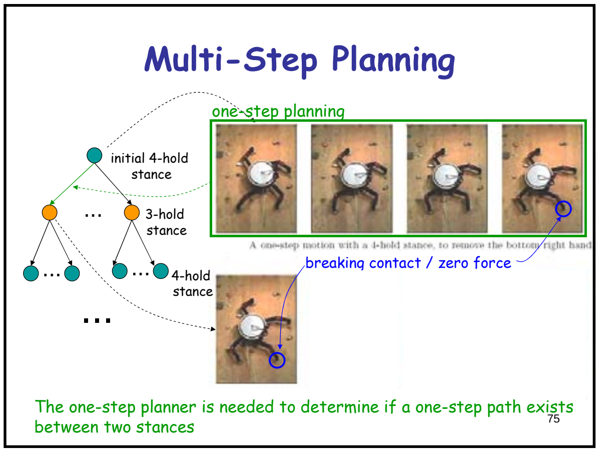
Multi-Step Planning



Multi-Step Planning

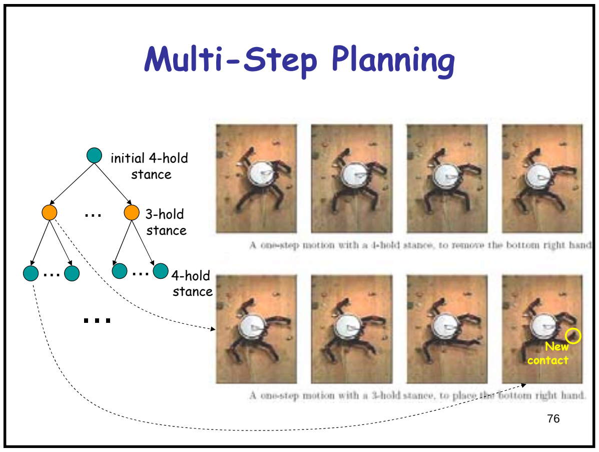

initial 4-hold stance
3-hold stance
4-hold stance
…
A one-step motion with a 4-hold stance, to remove the bottom right hand
breaking contact / zero force

Multi-Step Planning

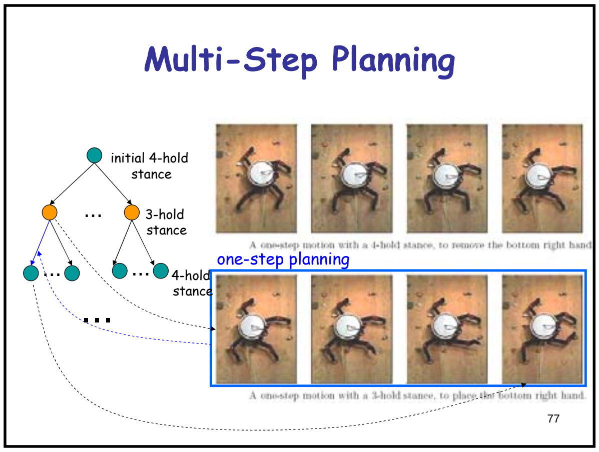

one-step planning
initial 4-hold stance
3-hold stance
4-hold stance
…
A one-step motion with a 4-hold stance, to remove the bottom right hand
breaking contact / zero force

The one-step planner is needed to determine if a one-step path exists between two stances

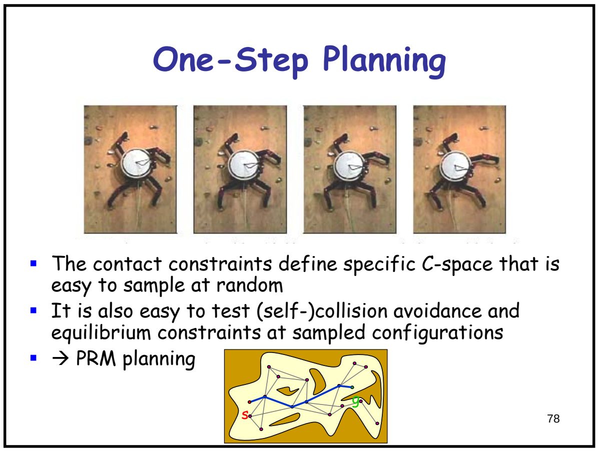
Multi-Step Planning


initial 4-hold stance
3-hold stance
4-hold stance
…
A one-step motion with a 4-hold stance, to remove the bottom right hand
New contact
A one-step motion with a 3-hold stance, to place the bottom right hand.

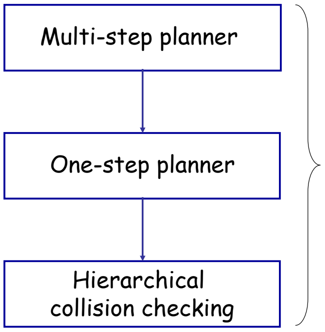
Multi-Step Planning


initial 4-hold stance
3-hold stance
4-hold stance
…
A one-step motion with a 4-hold stance, to remove the bottom right hand
one-step planning
A one-step motion with a 3-hold stance, to place the bottom right hand.

One-Step Planning



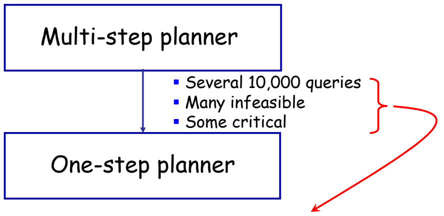
- The contact constraints define specific C-space that is easy to sample at random
- It is also easy to test (self-)collision avoidance and equilibrium constraints at sampled configurations
- → PRM planning



Many searches !!

(+ searches for planning exploratory moves, for interpreting 3D sensor data, ...)

Dilemma

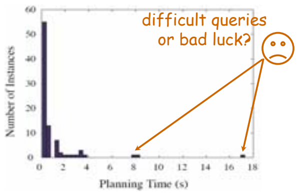


- Several 10,000 queries
- Many infeasible
- Some critical

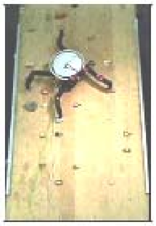
- A PRM planner can't detect that a query is infeasible
- How much time should it spend on each query?
- Too little, and it will often fail incorrectly
- Too much, and it will waste time on infeasible queries

Dilemma

- More than 1,000,000 stances, but only 20,000 feasible ones
- Several 1000 one-step path queries
- A large fraction of them have no solution
- The running times for (feasible) queries making up an 88-step path are highly variable



difficult queries or bad luck? ☹

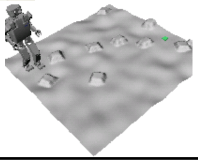


Possible Solution

- Use learning method to train a "feasibility" classifier
- Use this classifier to avoid infeasible one-step queries in the multi-step search tree
- More on this later in a lecture on Learning (if there is enough time)

Another Multi-Step Planning Problem:

Navigation of legged robots on rough terrain (stances → foot placements)



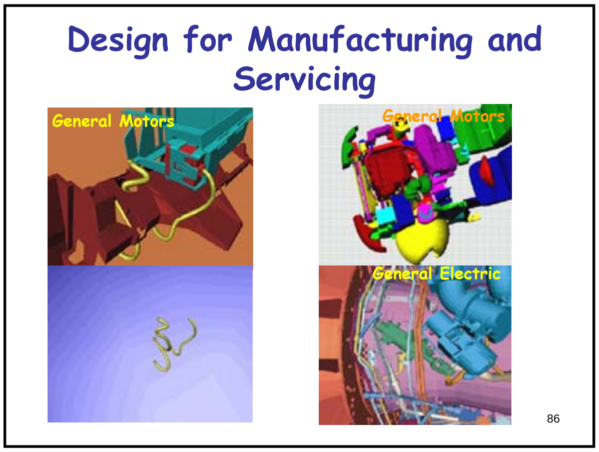
Another Multi-Step Planning Problem

Object manipulation (stances → grasps)

[Yamane, Kuffner and Hodgins]

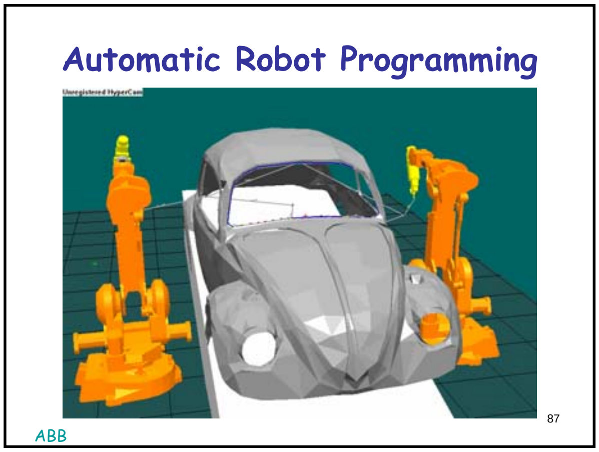
Some Applications of Motion Planning

85



Design for Manufacturing and Servicing

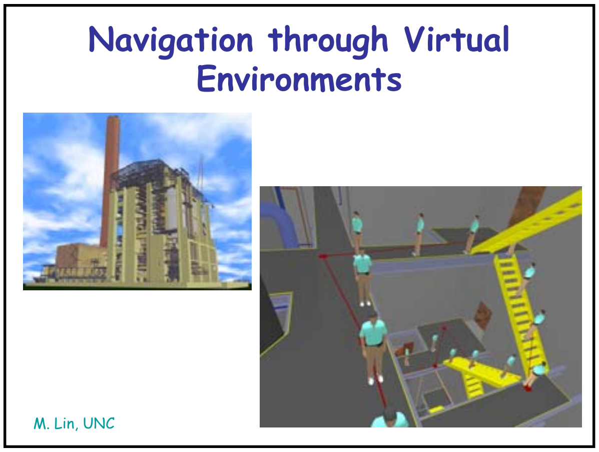
86



Automatic Robot Programming

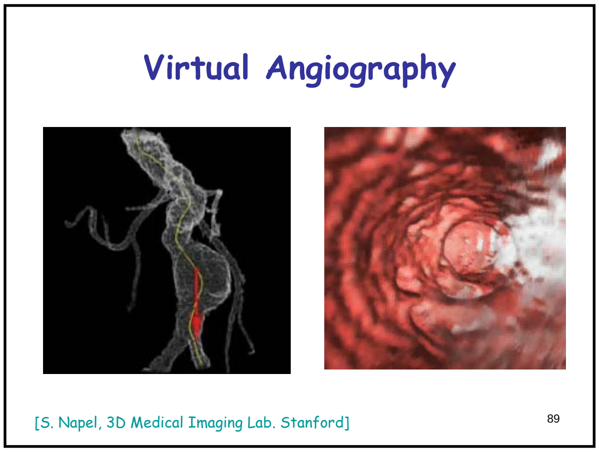
87

ABB



Navigation through Virtual Environments

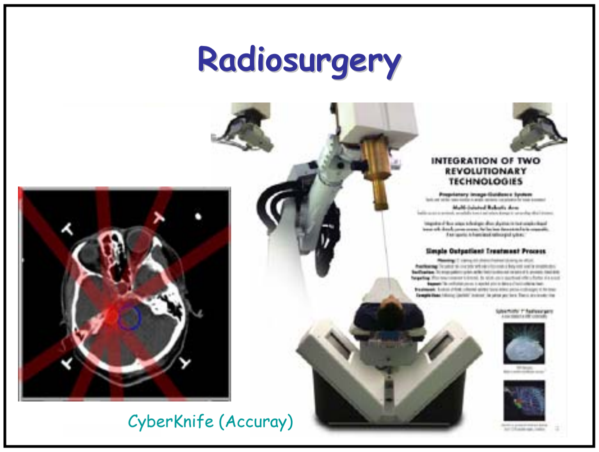
M. Lin, UNC



Virtual Angiography

[S. Napel, 3D Medical Imaging Lab. Stanford]

89



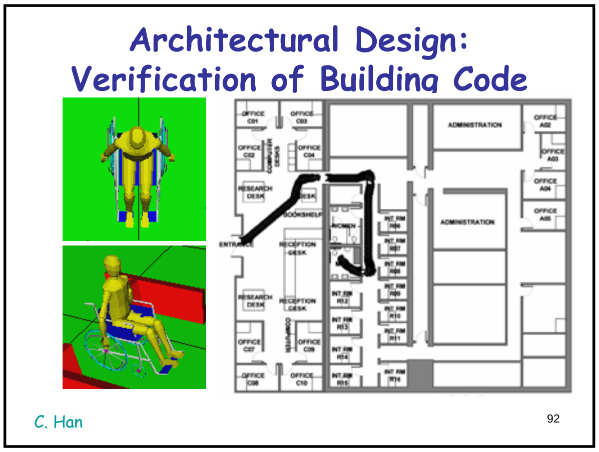
Radiosurgery

INTEGRATION OF TWO REVOLUTIONARY TECHNOLOGIES
Simple Outpatient Treatment Process

CyberKnife (Accuray)

Transportation of A380 Fuselage through Small Villages

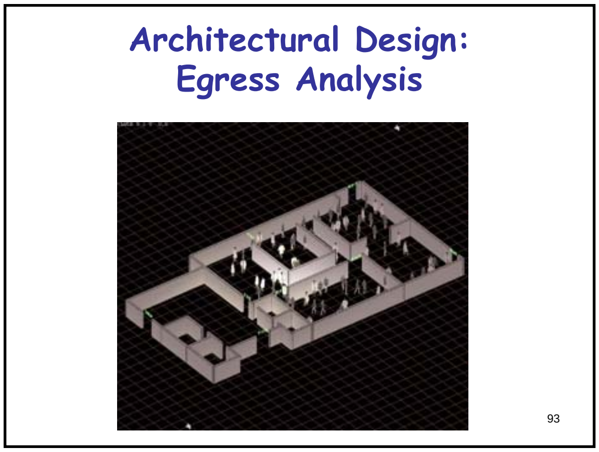
Kineo



Architectural Design: Verification of Building Code

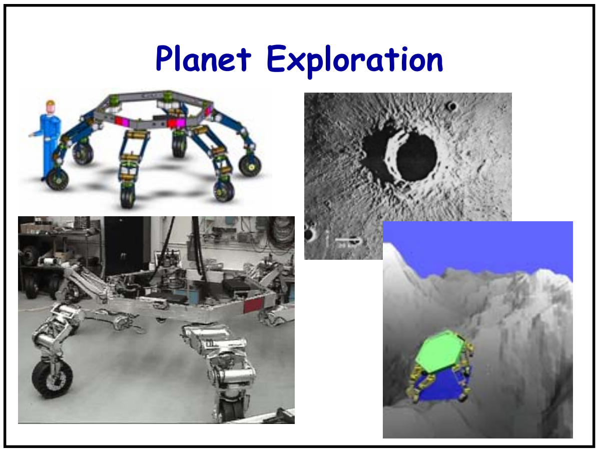
C. Han

92

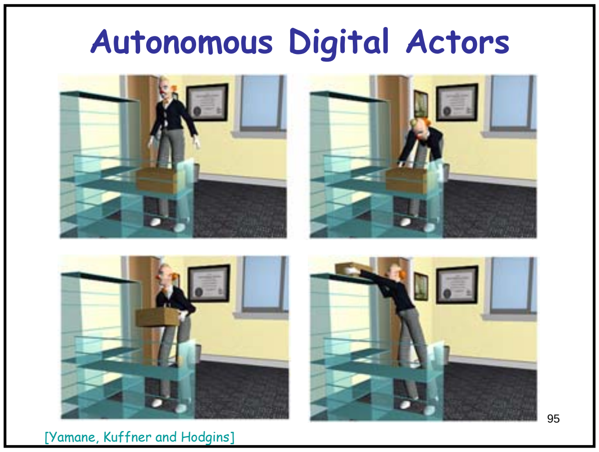


Architectural Design: Egress Analysis

93



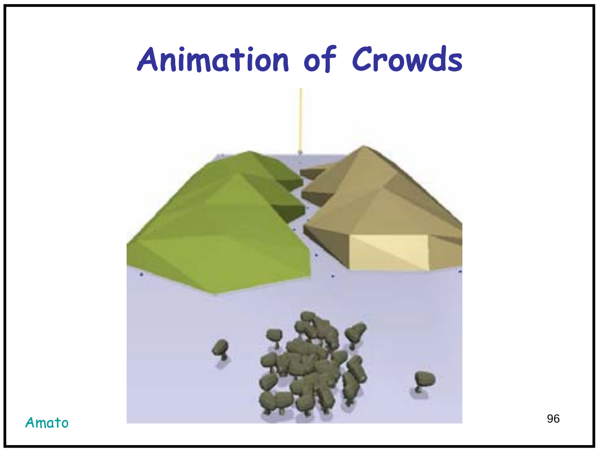
Planet Exploration



Autonomous Digital Actors

95

[Yamane, Kuffner and Hodgins]



Animation of Crowds

Amato

96