

Course Logistics

Dmitry Berenson

Assistant Professor

Robotics Institute

Electrical Engineering and Computer Science

Practical Issues

- Office hours are available by appointment
- See course website for most current information:
<http://web.eecs.umich.edu/~dmitryb/courses/winter2019motionplanning/index.html>
(Link to this site is on my homepage)
- For those not yet enrolled, keep coming to class to see if spots open

Prerequisites

- Undergraduate Linear Algebra (e.g. MATH 214)
- Significant programming experience (e.g. EECS 281)
 - We will use python and C++ in this course
 - You don't need to be an expert but I expect you to pick it up

SENSORS, SOFTWARE AND SYSTEMS FOR SMARTER CARS

Cars That Think


IEEE SPECTRUM

24 October 2018

C/C++ and Python Top List of Hot Skills for Autonomous Vehicle Engineers

It's a good time to be an engineer with a background in self-driving cars. According to job site Indeed.com, the numbers of people in the United States looking to get into the field or change jobs within it has increased by more than 600 percent in the past three years. And demand hasn't slowed. Read on to find out who's hiring and where.

→ [Read more](#)



Python Example

```
#get the second TSR's offset from the handle to the hand
Tw1_e = linalg.inv(T0_w0*Tw0_e)*T0_RH1

#define bounds to allow rotation of the hand about z axis of the handle from -pi
Bw1 = mat([0, 0, 0, 0, 0, 0, 0, 0, -pi/2, 0])

#serialize the TSRs
TSRstring1 = SerializeTSR(0, 'NULL', T0_w0, Tw0_e, Bw0)
TSRstring2 = SerializeTSR(0, 'NULL', mat(eye(4)), Tw1_e, Bw1) #note the T0_w value k
TSRChainString = SerializeTSRChain(0, 0, 1, 2, TSRstring1 + ' ' + TSRstring2, 'kitchen')

#set the desired amount to open the door
door_rot = pi/1.5
#set the desired amount to rotate the hand about the handle at the goal
handle_rot = -pi/2.1

#keep track of how many DOF of the mimic body are being mimicked
numTSRChainMimicDOF = 1;

#get goal transform
Tdoor_rot = MakeTransform(rodrigues([0, 0, door_rot]), mat([0, 0, 0]).T)
Thandle_rot = MakeTransform(rodrigues([0, 0, handle_rot]), mat([0, 0, 0]).T)
T0_doornew = T0_w0*Tdoor_rot;
T0_RH2 = T0_doornew*Tw0_e*Thandle_rot*linalg.inv(T0_doornew*Tw0_e)* T0_doornew*linalg.inv(T0_w0)

goalik = str2num(probs_cbirrt.SendCommand('DoGeneralIK exec nummanips 1 maniptm 0 %s' % robot.SetActiveDOFValues(goalik)))

#set robot to starting configuration
```

C++ example

```
bool bGotSolution = false;
for(int i = 0; i < qGuesses.size(); i++)
{
    _planner->ClearTSRChainValues();
    // if a guess is valid, this will confirm it and return quickly, if not, it will
    if(ConstrainedNewConfig(qGuesses[i], q_near, _planner->vTSRChainValues_temp, false)
    {

        RrtNode * pikNode = new RrtNode(_planner->GetNumDOF(), ptree->GetFromGoal());
        pikNode->SetConfig(&qGuesses[i][0]);
        pikNode->SetTSRChainValues(_planner->vTSRChainValues_temp);

        ptree->AddNode(*pikNode); //this will automatically increment the size
        pikNode->Print();
        delete pikNode;
        //return true;
        bGotSolution = true;
    }
}
if(bGotSolution)
    RAVELOG_INFO("Constrained IK solution(s) found!\n");
else
    RAVELOG_INFO("No Constrained IK solution found!\n");

return bGotSolution;
}

bool CBirrtPlanner::_CheckCollision(std::vector<dReal>& pConfig)
{
    collisioncheckcalls++;
#ifdef RECORD_TIMES
    double starttime = timeGetThreadTime();
#endif
```

Math expectations

- Linear algebra proficiency is a MUST
 - Matrix operations
 - Dot products, cross products, etc.
- Be able to read math notation
 - Some examples from the book:

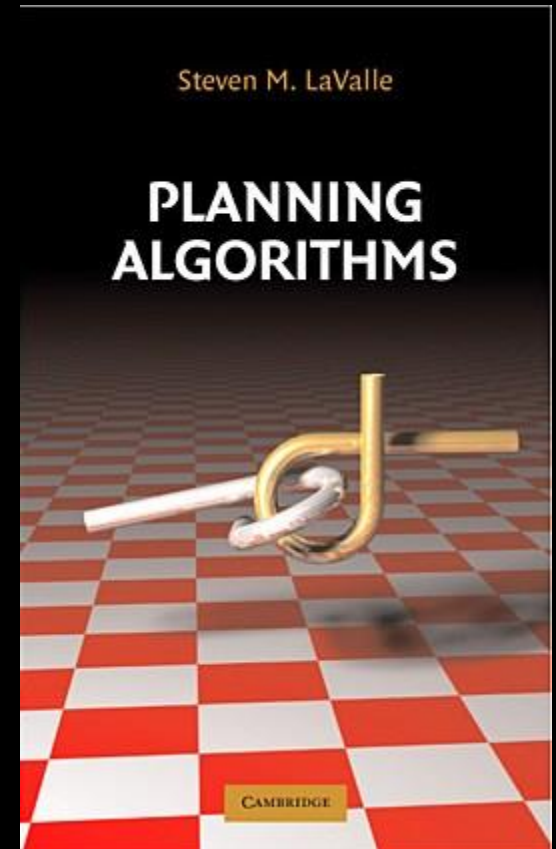
$$X_{obs} = \left(\bigcup_{t=1}^m X_{obs}^t \right) \cup \left(\bigcup_{tj, t \neq j} X_{obs}^{tj} \right)$$

$$X_{obs}^{tj} = \{(s_1, \dots, s_m) \in X \mid \mathcal{A}^t(\tau_t(s_t)) \cap \mathcal{A}^j(\tau_j(s_j)) \neq \emptyset\}.$$

$$\mathcal{C}_{obs}^p = \{(q^a, q^p) \in \mathcal{C} \mid \text{int}(\mathcal{P}(q^p)) \cap \mathcal{O} \neq \emptyset\}$$

Readings

- Book for this course is available **free** online (link is on course webpage)
- Early on, we will read mostly from the book
- Later on, we will read mostly research papers
- Students will present and discuss research papers



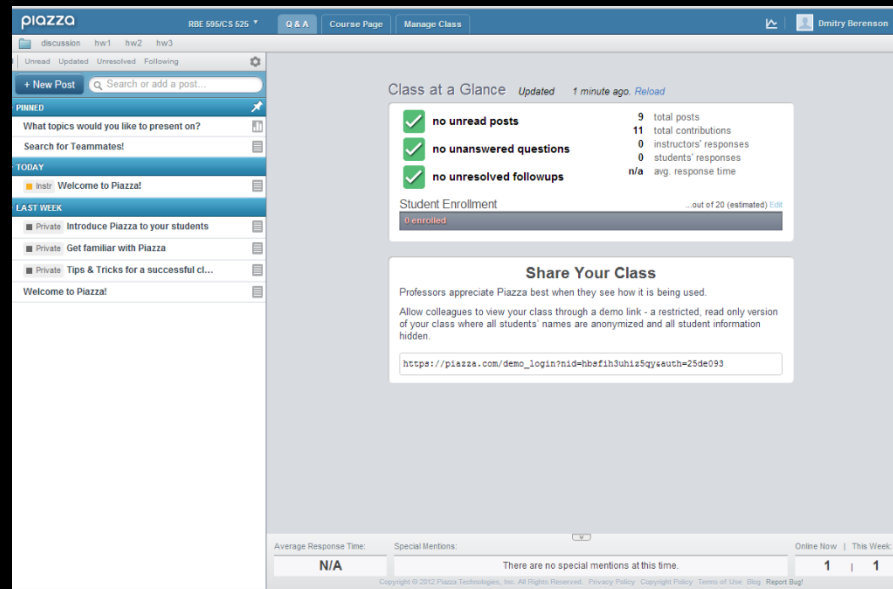
Syllabus

- No exams!
- Three homeworks
 - Individual (no groups)
 - Learn to implement algorithms
 - Use openrave open-source motion planning framework
- Final Project
 - Best option: Apply motion planning to your research
 - Individual project encouraged (tailor to your research), teams of 2 or 3 also allowed



Piazza

- <https://piazza.com/umich/winter2019/eecs598003wn2019/home>
- We will use **piazza.com** for all class communication (question/answer about homeworks, paper discussion, etc.)



- You get credit for helping your classmates!
 - But, don't give away answers

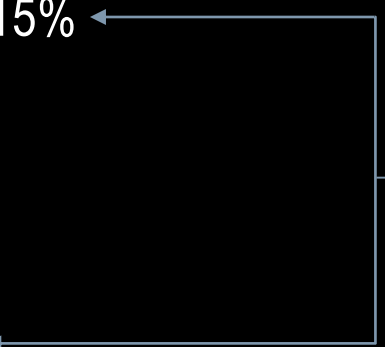
Schedule

- Latest schedule is on course website:

Date	Topic	Slides	Reading		Assignment	
1/9/2019	L: Introduction to Motion Planning		Chapter 1	Graph Review	Sign up for the class on Piazza	
1/14/2019	L: Planning for point robots		Chapter 4.0 - 4.3			
1/16/2019	L: Configuration Space		Chapter 2.0 - 2.3		Homework 1 Out	
1/21/2019	MLK DAY: NO CLASS					
1/23/2019	L: Discrete Motion Planning		Chapter 3.2 - 3.3			
1/28/2019	L: Transformations				Homework 1 Due	Enter presentation topic preferences into Piazza
1/30/2019	L: Collision Checking		Probability Review	Chapter 5		
2/4/2019	L: Sampling-based Planning 1		How to read a paper			
2/6/2019	L: Sampling-based Planning 2		Chapter 14-14.5			
2/11/2019	L: Non-holonomic motion planning		RRT* paper			
2/13/2019	L: Asymptotically Optimal Planners					
2/18/2019	L: How to do a research presentation					
2/20/2019						
2/25/2019						
2/27/2019						
3/4/2019	BREAK					
3/6/2019	BREAK					
3/11/2019	L: Planning for articulated robots 1					
3/13/2019	L: Planning for articulated robots 2					
3/18/2019						
3/20/2019						
3/25/2019						
3/27/2019						
4/1/2019						
4/3/2019						
4/8/2019						
4/10/2019						
4/15/2019						
4/17/2019						
4/22/2019						
	L = instructor lecture					
	P = student presentation					
	TBD					

Grading

- In-Class Participation and Preparation 10%
 - Asking questions during lecture
 - Participating in paper discussions
 - Participating in discussions on Piazza
 - ATTENDANCE ALONE WILL ONLY GIVE YOU 50% PARTICIPATION CREDIT
- Research Paper Reviews 5%
- Research Paper Presentations 15%
- Homeworks 20%
- Final Project Proposal 10%
- Final Project Writeup 25%
- Final Project Presentation 15%



30% of your grade comes from presentations!

Presentations

- Each student will do an individual research paper presentation
- In-depth understanding of a paper
- Tentatively 15 minutes long + 2 minutes of questions
 - Like a standard conference talk
- Evaluated on
 - Depth of understanding
 - Clarity of presentation
 - Presentation skill (don't run out of time!)

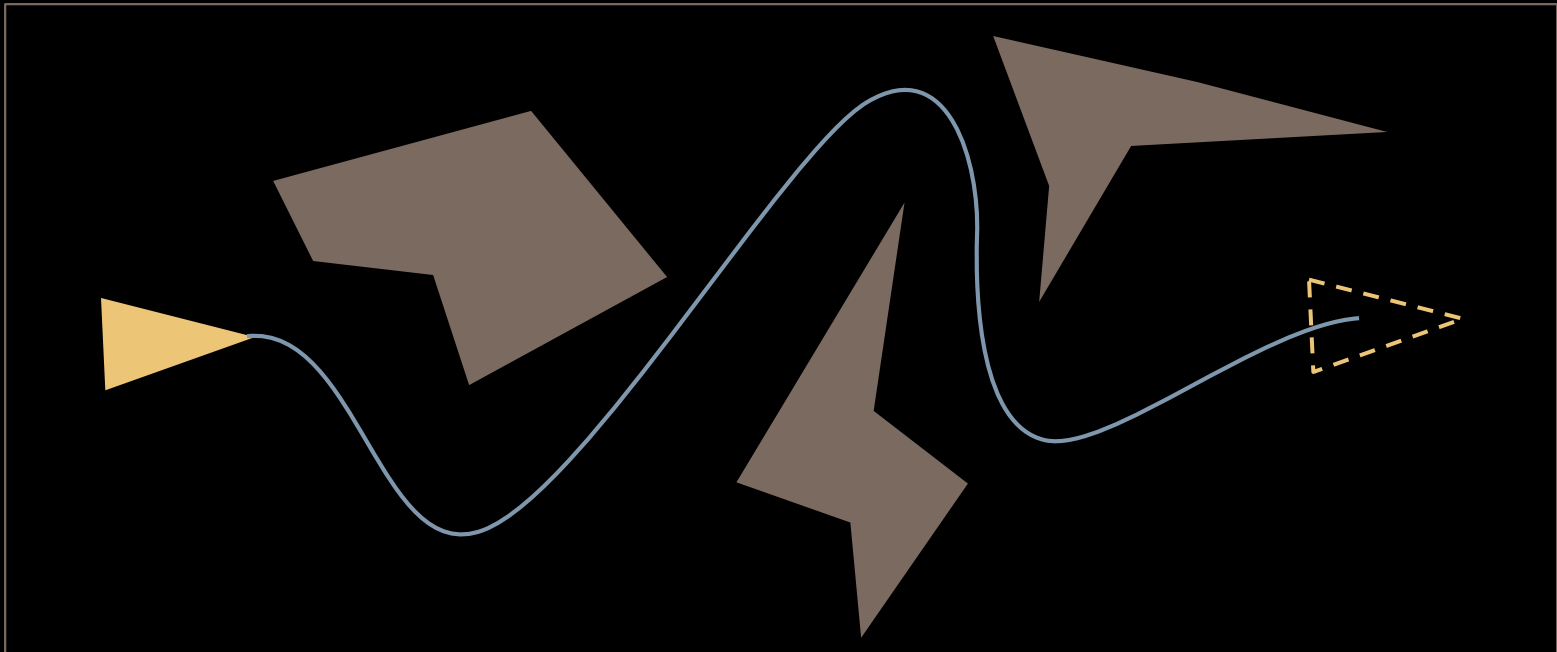
What you can expect to get from this course

- An understanding of modern motion planning methods
- Overview of recent research in motion planning
- Programming experience
- Hands-on experience working with motion planning
- ***Paper-reading, presentation, and research skills***

Introduction to Motion Planning

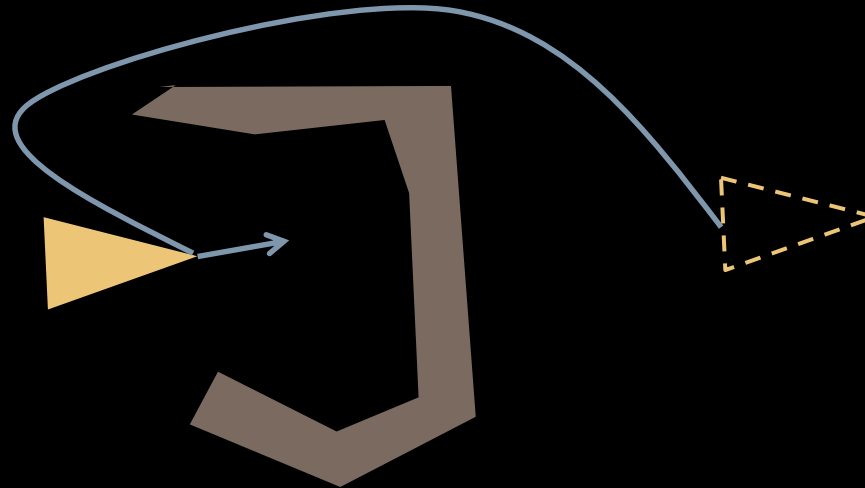
What is motion planning?

- The automatic generation of motion

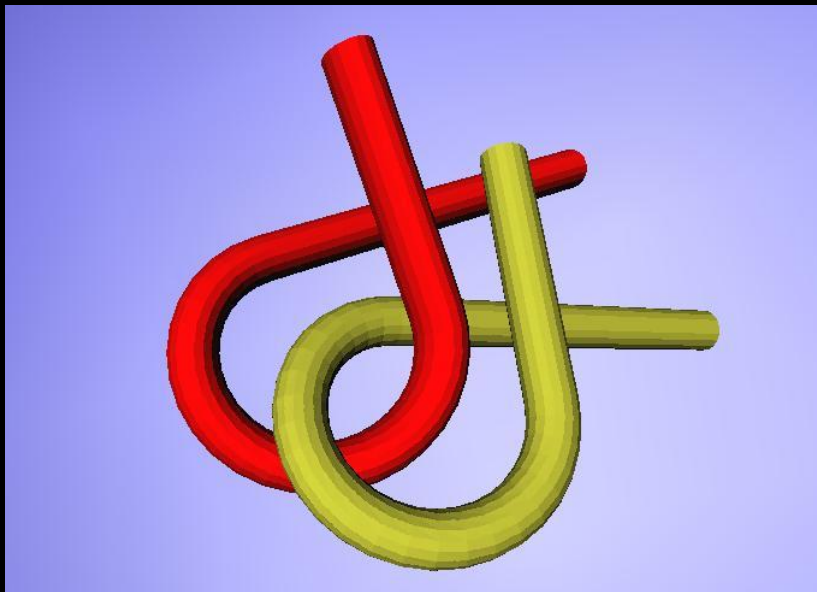


Why Motion Planning Instead of Obstacle Avoidance?

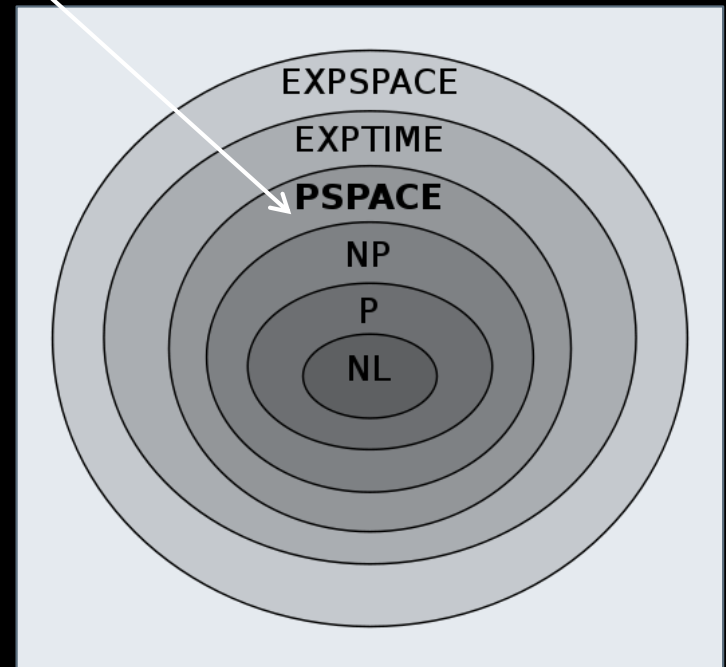
- Path planning
 - low-frequency, time-intensive search method for global finding of a (optimal) path to a goal
- Obstacle avoidance (aka “local navigation”)
 - fast, reactive method with local time and space horizon
- Distinction: Global vs. local reasoning



Is motion planning hard?

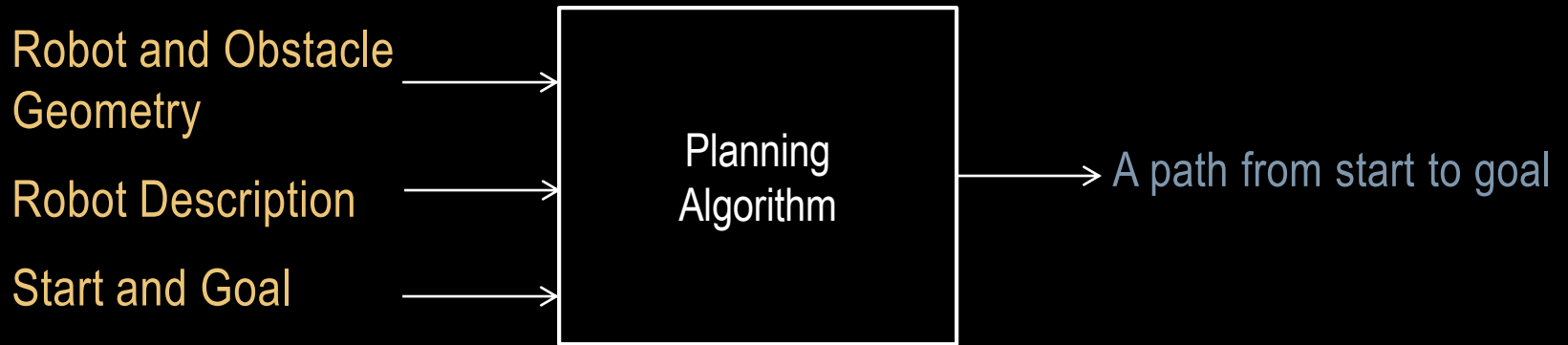


Basic Motion
Planning Problems



Basic Problem Statement

- *Automatically compute a path for an object/robot that does not collide with obstacles.*

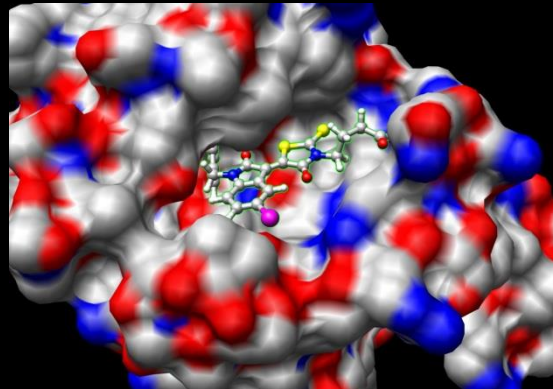
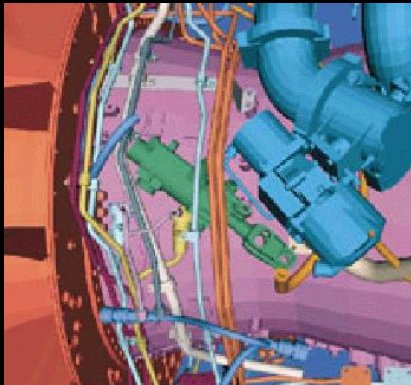


What can motion planning do?

- Automatically generate motion



- Automatically validate



Applications: Mobile Robots



Roomba iCreate



Mars Rovers



DARPA Urban Challenge



Google Self-Driving Car

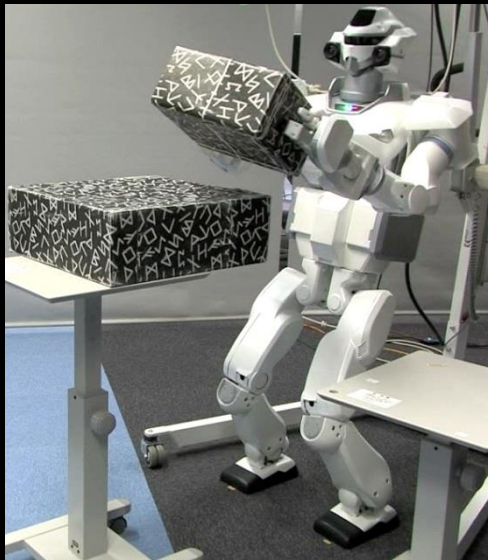
Applications: Robotic Manipulation



Factory Automation



Personal Robots



Humanoid Robots



Personal Robots

Applications: Computer Games/Graphics



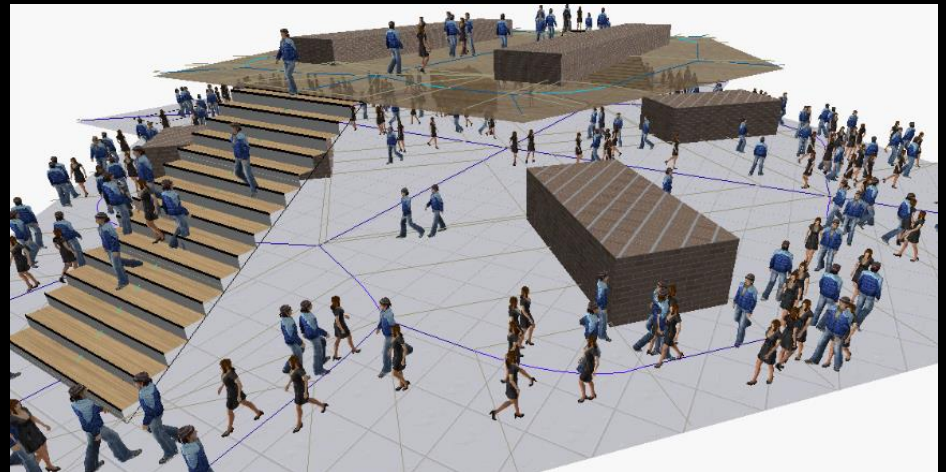
Path Finding in Games



Character Animation

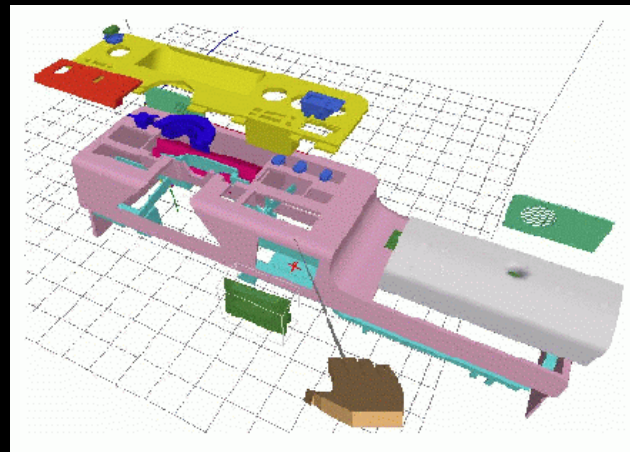
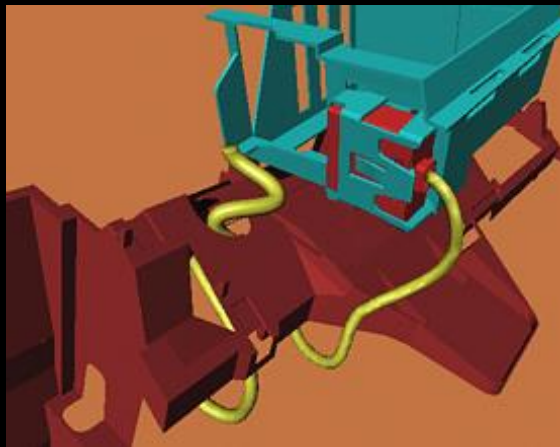
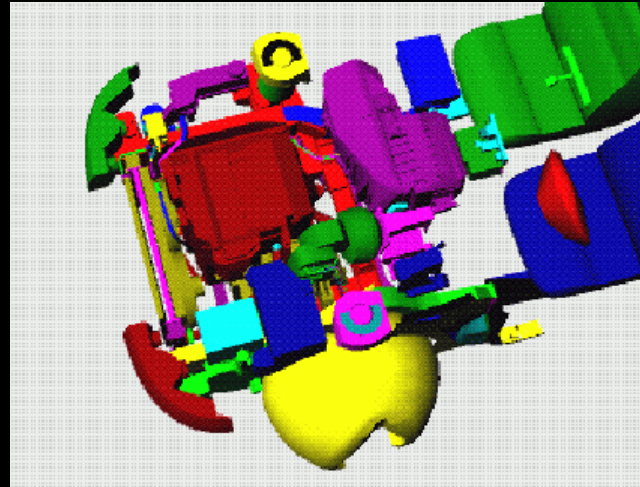
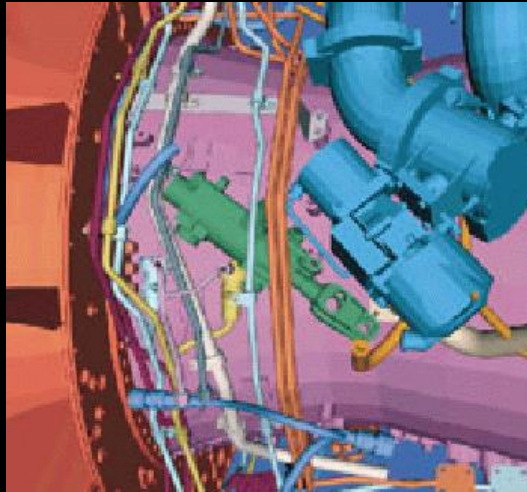


Retargeting Motion Capture

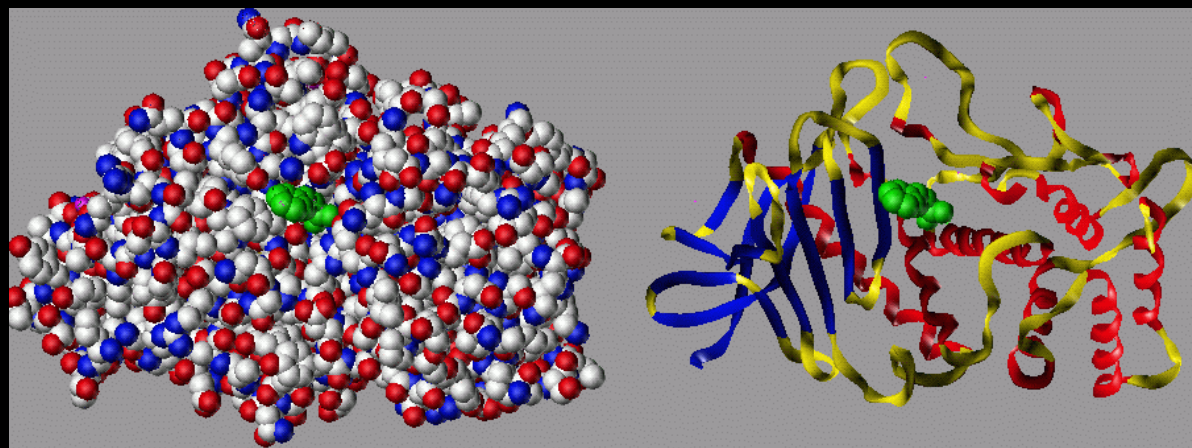
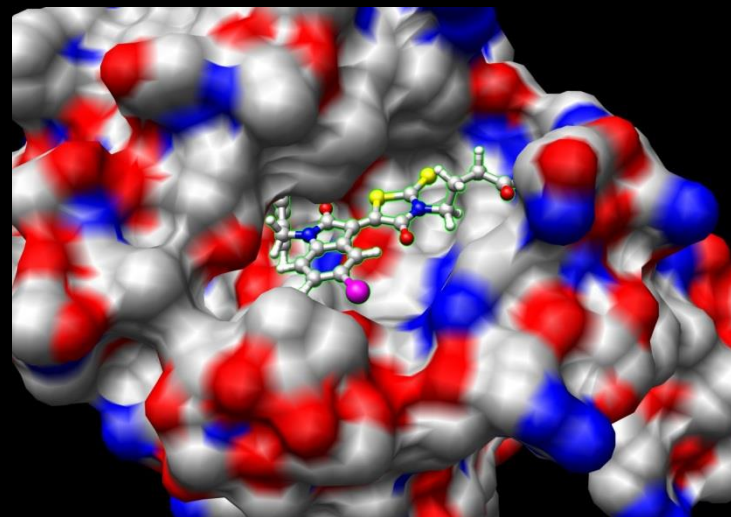
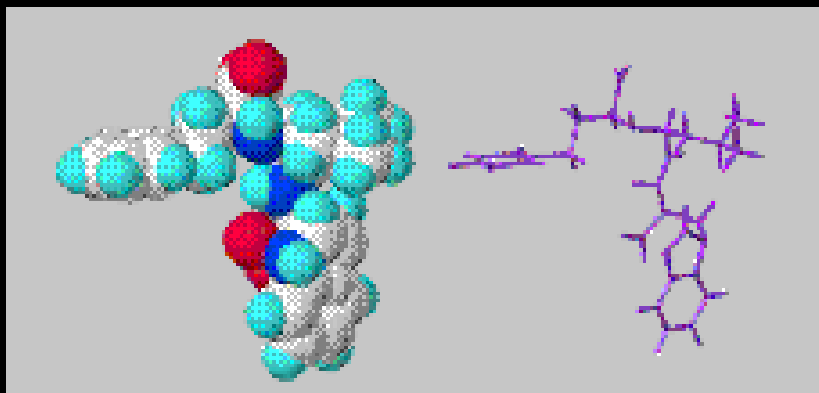


Animation of Crowds

Applications: Assembly Planning

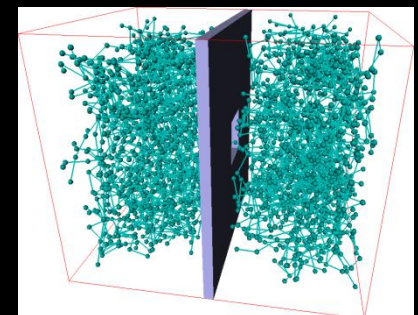
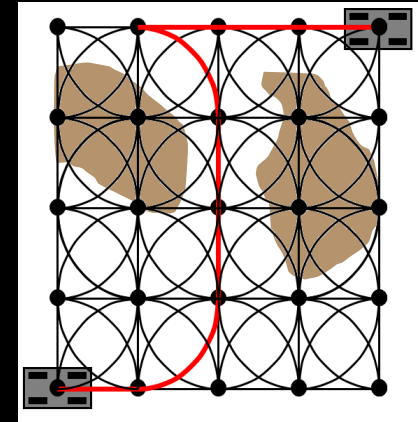
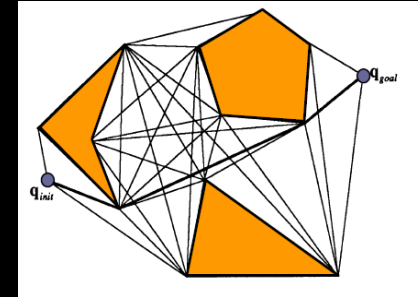


Applications: Computational Biology



Approaches

- Exact algorithms
 - Either find a solution or prove none exists
 - Very computationally expensive
 - Unsuitable for high-dimensional spaces
- Discrete Search
 - Divide space into a grid, use A* to search
 - Good for vehicle planning
 - Unsuitable for high-dimensional spaces
- Sampling-based Planning
 - Sample the C-space, construct path from samples
 - Good for high-dimensional spaces
 - Weak completeness and optimality guarantees



What matters?

- Motion planning algorithms are judged on
 - Completeness
 - Optimality
 - Speed (AKA efficiency)
 - Generality
- These vary in importance depending on the application

What matters: Completeness

- Will the algorithm solve all solvable problems?
 - Will the algorithm return no solution for unsolvable problems?
 - What if the algorithm is probabilistic?
-
- For what application(s) is completeness very important?
 - For what application(s) is completeness not important?

What matters: Optimality

- Will the algorithm generate the shortest path?
 - Will the algorithm generate the least-cost path (for an arbitrary cost function)?
 - Do we need optimality or is feasibility enough?
-
- For what application(s) is optimality very important?
 - For what application(s) is optimality not important?

What matters: Speed (AKA Efficiency)

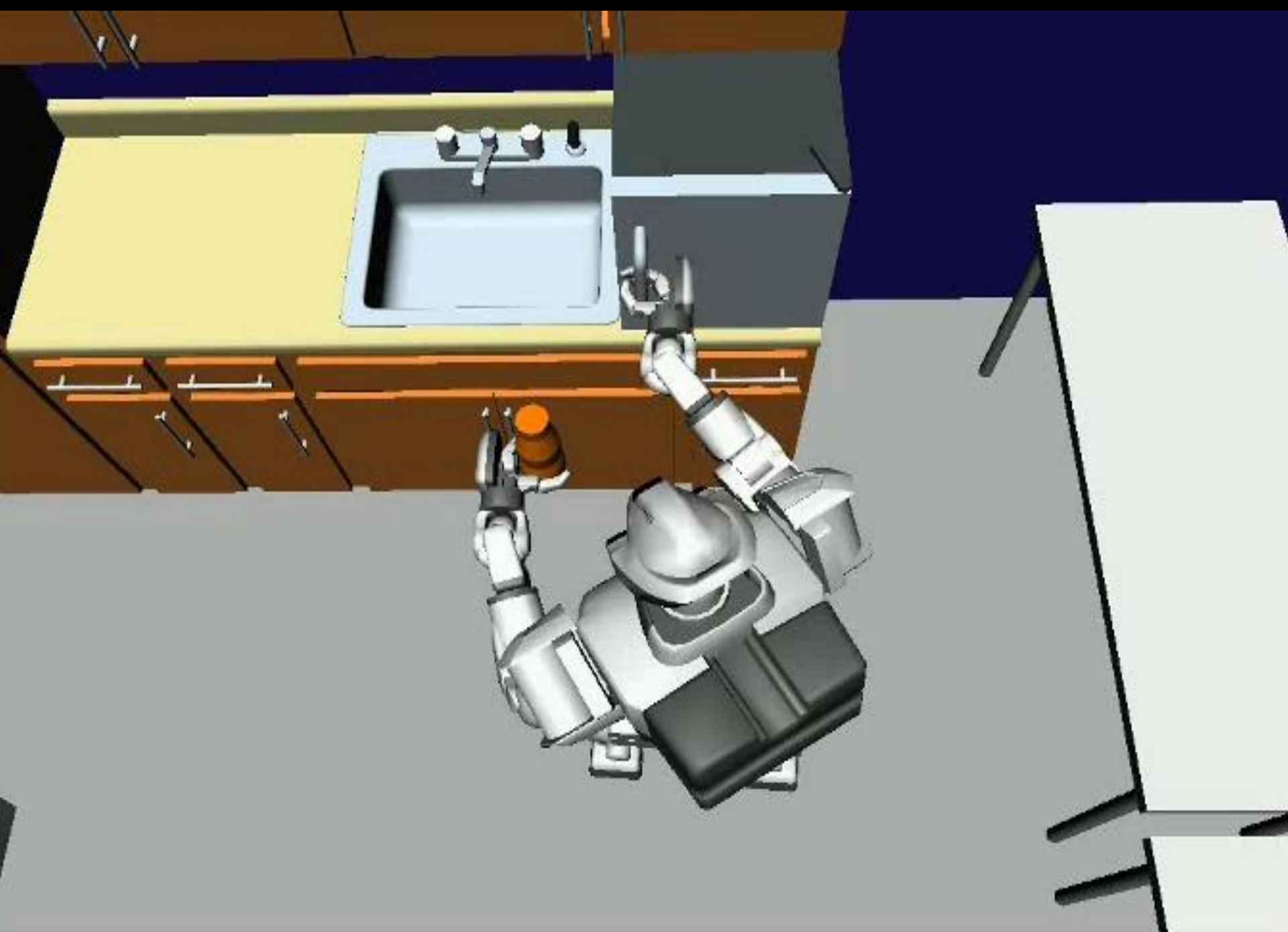
- How long does it take to generate a path for **real-world** problems?
- How does the run-time scale with dimensionality of the problem and complexity of models?
- Is there a quality vs. computation time tradeoff?
- For what application(s) is speed very important?
- For what application(s) is speed not important?

What matters: Generality

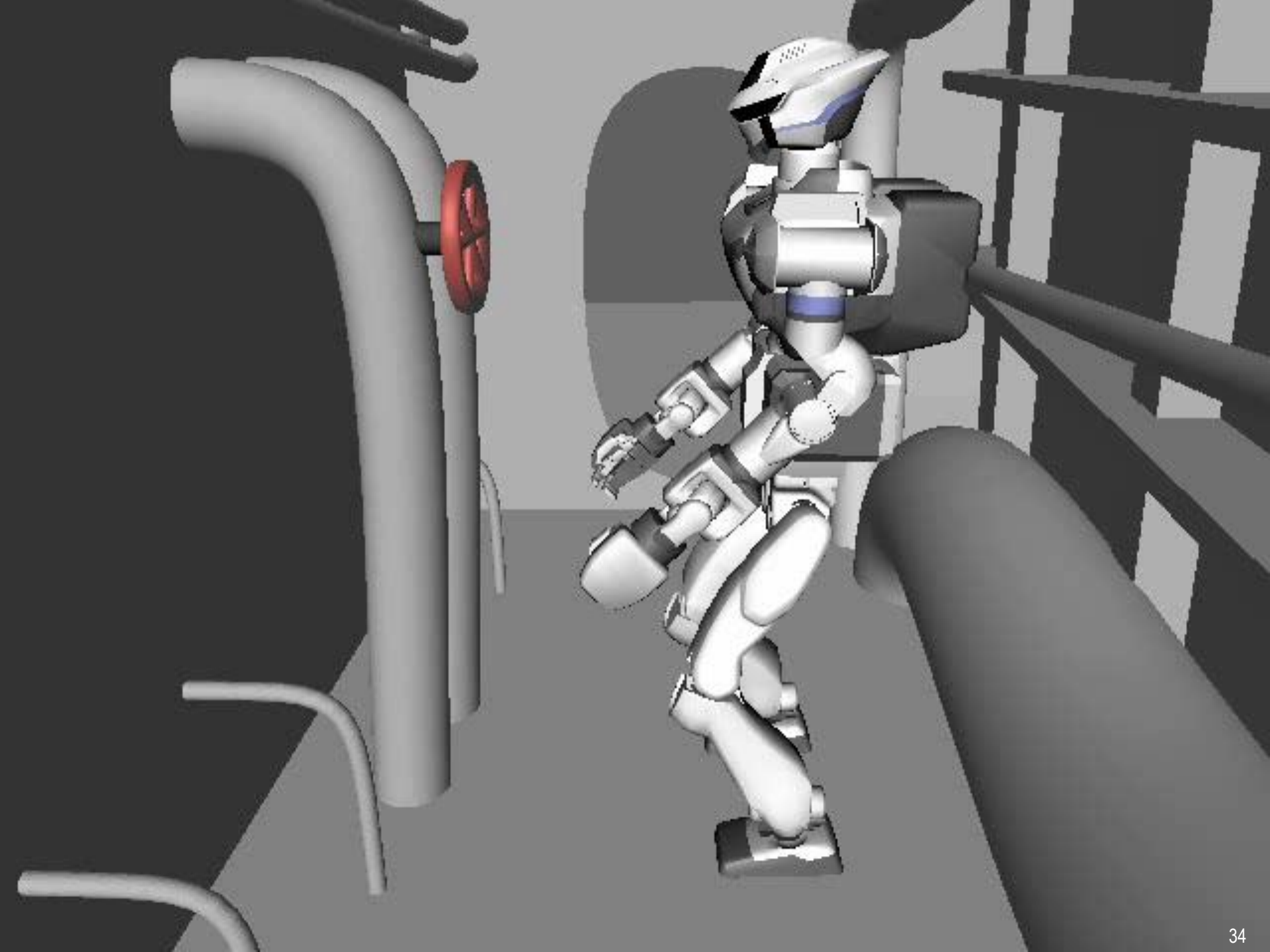
- What types of problems can it solve?
 - What types of problems can't it solve?
-
- For what application(s) is generality very important?
 - For what application(s) is generality not important?

$w = 4.98\text{kg}$



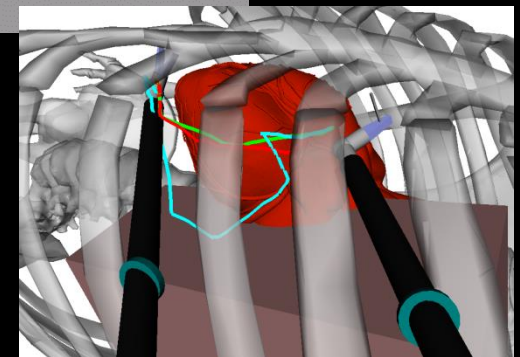
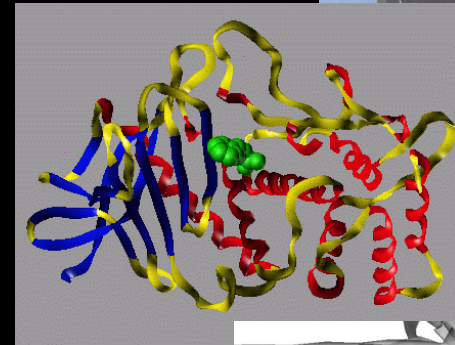
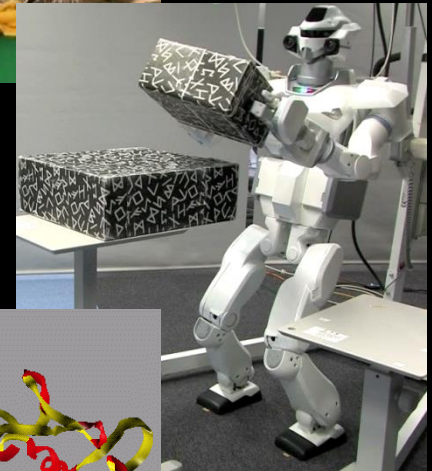






Summary

- Robot motion planning is used for...
 - Vehicles
 - Robots
 - Digital Characters
 - Molecules
 - Design verification
- Types of planning methods
 - Exact algorithms
 - Discrete Search
 - Sampling-based planners
- *Many frontiers to explore*



Activity

From "Mechanics of Manipulation" by Matthew T. Mason, MIT Press

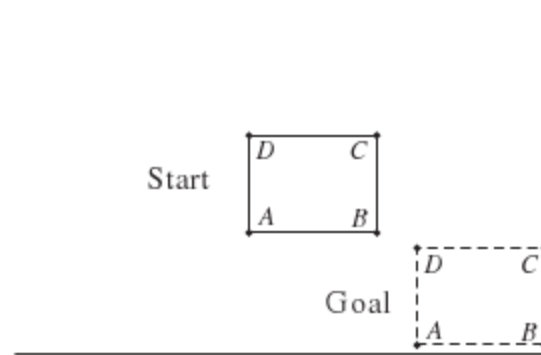


Figure 2.24
Construction for exercise 2.4.

Exercise 2.4: Your new refrigerator has been delivered, and has to be moved from the center of your kitchen into the corner. (See figure 2.24.) You can “walk” it by shifting the weight towards one leg, then pushing so the refrigerator rotates about that leg. Find a short sequence of rotations to walk the refrigerator into the corner, Don't go through any walls. (Hint: assembly problems are often easier to solve backwards, so find a path from the goal to the start.)

Activity

From "Mechanics of Manipulation" by Matthew T. Mason, MIT Press

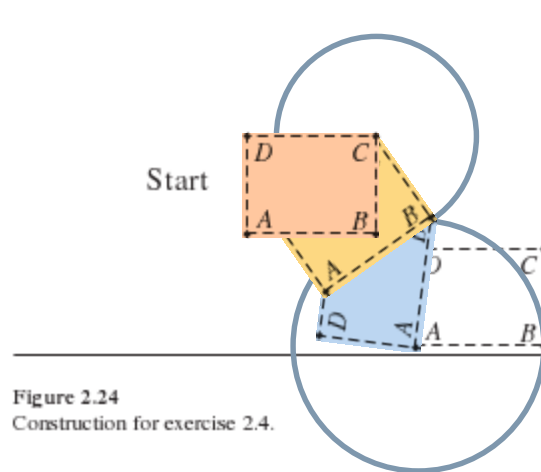


Figure 2.24
Construction for exercise 2.4.

Exercise 2.4: Your new refrigerator has been delivered, and has to be moved from the center of your kitchen into the corner. (See figure 2.24.) You can “walk” it by shifting the weight towards one leg, then pushing so the refrigerator rotates about that leg. Find a short sequence of rotations to walk the refrigerator into the corner, Don’t go through any walls. (Hint: assembly problems are often easier to solve backwards, so find a path from the goal to the start.)

Homework

- Read Graph Review
- Read LaValle chapter 1
- Sign up for course on Piazza