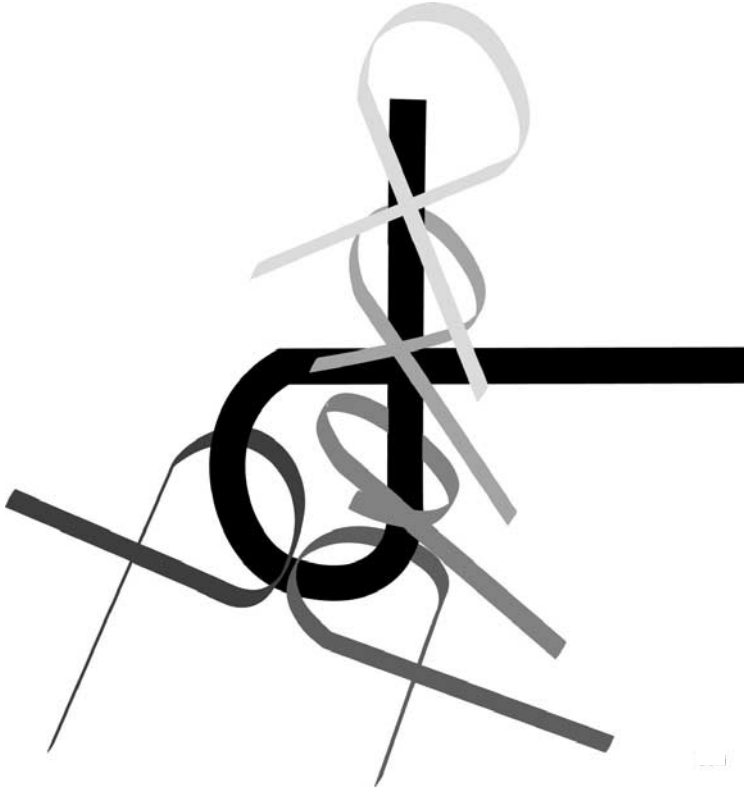
PART I

Introductory Material



1

Introduction

1.1 Planning to plan

Planning is a term that means different things to different groups of people. *Robotics* addresses the automation of mechanical systems that have sensing, actuation, and computation capabilities (similar terms, such as *autonomous systems* are also used). A fundamental need in robotics is to have algorithms that convert high-level specifications of tasks from humans into low-level descriptions of how to move. The terms *motion planning* and *trajectory planning* are often used for these kinds of problems. A classical version of motion planning is sometimes referred to as the *Piano Mover's Problem*. Imagine giving a precise computer-aided design (CAD) model of a house and a piano as input to an algorithm. The algorithm must determine how to move the piano from one room to another in the house without hitting anything. Most of us have encountered similar problems when moving a sofa or mattress up a set of stairs. Robot motion planning usually ignores dynamics and other differential constraints and focuses primarily on the translations and rotations required to move the piano. Recent work, however, does consider other aspects, such as uncertainties, differential constraints, modeling uncertainties, and optimality. Trajectory planning usually refers to the problem of taking the solution from a robot motion planning algorithm and determining how to move along the solution in a way that respects the mechanical limitations of the robot.

Control theory has historically been concerned with designing inputs to physical systems described by differential equations. These could include mechanical systems such as cars or aircraft, electrical systems such as noise filters, or even systems arising in areas as diverse as chemistry, economics, and sociology. Classically, control theory has developed *feedback policies*, which enable an adaptive response during execution, and has focused on *stability*, which ensures that the dynamics do not cause the system to become wildly out of control. A large emphasis is also placed on optimizing criteria to minimize resource consumption, such as energy or time. In recent control theory literature, *motion planning* sometimes refers to the construction of inputs to a nonlinear dynamical system that drives it from an initial state to a specified goal state. For example, imagine trying to operate a remote-controlled hovercraft that glides over the surface of a frozen pond. Suppose we would like the hovercraft to leave its current resting location and come to rest at another specified location. Can an algorithm be designed that computes the desired inputs, even in an ideal simulator that neglects uncertainties that arise from model inaccuracies? It is possible to add other considerations, such as uncertainties, feedback, and optimality; however, the problem is already challenging enough without these.

In *artificial intelligence*, the terms *planning* and *AI planning* take on a more discrete flavor. Instead of moving a piano through a continuous space, as in the robot motion planning problem, the task might be to solve a puzzle, such as the Rubik's cube or a sliding-tile puzzle, or to achieve a task that is modeled discretely, such as building a

stack of blocks. Although such problems could be modeled with continuous spaces, it seems natural to define a finite set of actions that can be applied to a discrete set of states and to construct a solution by giving the appropriate sequence of actions. Historically, planning has been considered different from *problem solving*; however, the distinction seems to have faded away in recent years. In this book, we do not attempt to make a distinction between the two. Also, substantial effort has been devoted to representation language issues in planning. Although some of this will be covered, it is mainly outside of our focus. Many decision-theoretic ideas have recently been incorporated into the AI planning problem, to model uncertainties, adversarial scenarios, and optimization. These issues are important and are considered in detail in Part III.

Given the broad range of problems to which the term planning has been applied in the artificial intelligence, control theory, and robotics communities, you might wonder whether it has a specific meaning. Otherwise, just about anything could be considered as an instance of planning. Some common elements for planning problems will be discussed shortly, but first we consider planning as a branch of algorithms. Hence, this book is entitled *Planning Algorithms*. The primary focus is on algorithmic and computational issues of planning problems that have arisen in several disciplines. On the other hand, this does not mean that planning algorithms refers to an existing community of researchers within the general algorithms community. This book it not limited to combinatorics and asymptotic complexity analysis, which is the main focus in pure algorithms. The focus here includes numerous concepts that are not necessarily algorithmic but aid in modeling, solving, and analyzing planning problems.

Natural questions at this point are, What is a plan? How is a plan represented? How is it computed? What is it supposed to achieve? How is its quality evaluated? Who or what is going to use it? This chapter provides general answers to these questions. Regarding the user of the plan, it clearly depends on the application. In most applications, an algorithm executes the plan; however, the user could even be a human. Imagine, for example, that the planning algorithm provides you with an investment strategy.

In this book, the user of the plan will frequently be referred to as a *robot* or a *decision maker*. In artificial intelligence and related areas, it has become popular in recent years to use the term *agent*, possibly with adjectives to yield an *intelligent agent* or *software agent*. Control theory usually refers to the decision maker as a *controller*. The plan in this context is sometimes referred to as a *policy* or *control law*. In a game-theoretic context, it might make sense to refer to decision makers as *players*. Regardless of the terminology used in a particular discipline, this book is concerned with planning algorithms that find a strategy for one or more decision makers. Therefore, remember that terms such as *robot*, *agent*, and *controller* are interchangeable.

1.2 Motivational examples and applications

Planning problems abound. This section surveys several examples and applications to inspire you to read further.

Why study planning algorithms? There are at least two good reasons. First, it is fun to try to get machines to solve problems for which even humans have great difficulty. This involves exciting challenges in modeling planning problems, designing efficient algorithms, and developing robust implementations. Second, planning algorithms have achieved widespread successes in several industries and academic disciplines, including robotics, manufacturing, drug design, and aerospace applications. The rapid growth in

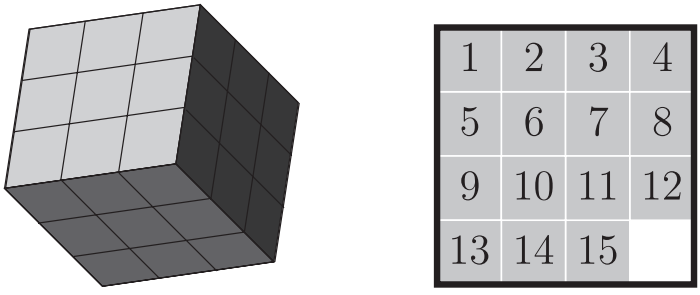


Figure 1.1: The Rubik's cube (a), sliding-tile puzzle (b), and other related puzzles are examples of discrete planning problems.

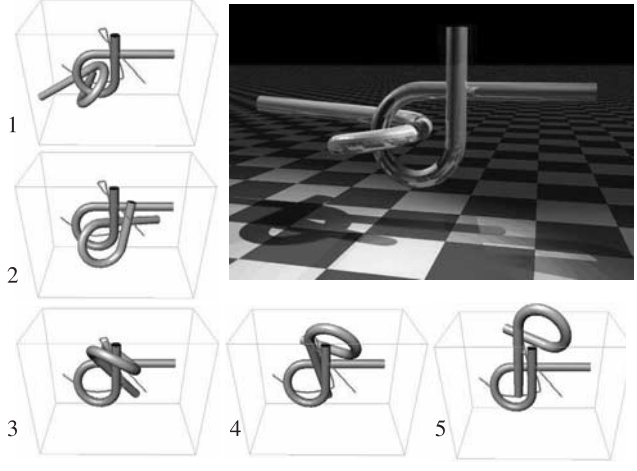


Figure 1.2: Remember puzzles like this? Imagine trying to solve one with an algorithm. The goal is to pull the two bars apart. This example is called the Alpha 1.0 Puzzle. It was created by Boris Yamrom and posted as a research benchmark by Nancy Amato at Texas A&M University. This solution and animation were made by James Kuffner (see [561] for the full movie).

recent years indicates that many more fascinating applications may be on the horizon. These are exciting times to study planning algorithms and contribute to their development and use.

Discrete puzzles, operations, and scheduling

Chapter 2 covers discrete planning, which can be applied to solve familiar puzzles, such as those shown in Figure 1.1. They are also good at games such as chess or bridge [899]. Discrete planning techniques have been used in space applications, including a rover that traveled on Mars and the Earth Observing One satellite [209, 384, 897]. When combined with methods for planning in continuous spaces, they can solve complicated tasks such as determining how to bend sheet metal into complicated objects [422]; see Section 7.5 for the related problem of folding cartons.

A motion planning puzzle

The puzzles in Figure 1.1 can be easily discretized because of the regularity and symmetries involved in moving the parts. Figure 1.2 shows a problem that lacks these properties

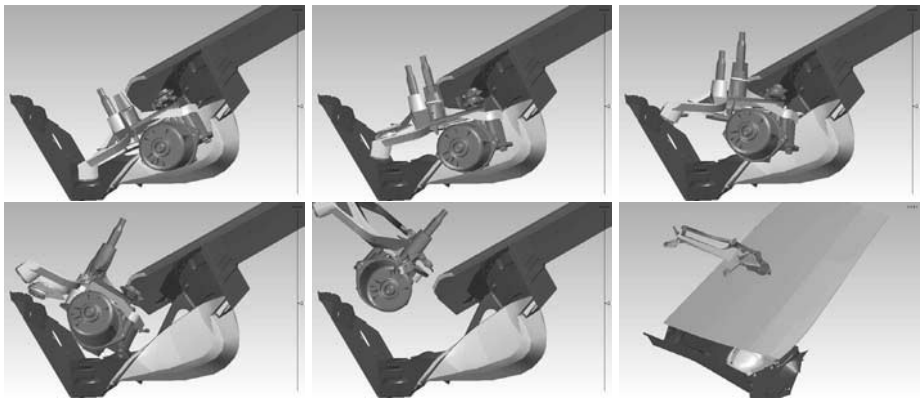


Figure 1.3: An automotive assembly task that involves inserting or removing a windshield wiper motor from a car body cavity. This problem was solved for clients using the motion planning software of Kineo CAM (courtesy of Kineo CAM).

and requires planning in a continuous space. Such problems are solved by using the motion planning techniques of Part II. This puzzle was designed to frustrate both humans and motion planning algorithms. It can be solved in a few minutes on a standard personal computer (PC) using the techniques in Section 5.5. Many other puzzles have been developed as benchmarks for evaluating planning algorithms.

An automotive assembly puzzle

Although the problem in Figure 1.2 may appear to be pure fun and games, similar problems arise in important applications. For example, Figure 1.3 shows an automotive assembly problem for which software is needed to determine whether a wiper motor can be inserted (and removed) from the car body cavity. Traditionally, such a problem is solved by constructing physical models. This costly and time-consuming part of the design process can be virtually eliminated in software by directly manipulating the CAD models.

The wiper example is just one of many. The most widespread impact on industry comes from motion planning software developed at Kineo CAM. It has been integrated into Robcad (eM-Workplace) from Tecnomatix, which is a leading tool for designing robotic workcells in numerous factories around the world. Their software has also been applied to assembly problems by Renault, Ford, Airbus, Optivus, and many other major corporations. Other companies and institutions are also heavily involved in developing and delivering motion planning tools for industry (many are secret projects, which unfortunately cannot be described here). One of the first instances of motion planning applied to real assembly problems is documented in [188].

Sealing cracks in automotive assembly

Figure 1.4 shows a simulation of robots performing sealing at the Volvo Cars assembly plant in Torslanda, Sweden. Sealing is the process of using robots to spray a sticky substance along the seams of a car body to prevent dirt and water from entering and causing corrosion. The entire robot workcell is designed using CAD tools, which automatically provide the necessary geometric models for motion planning software. The solution shown in Figure 1.4 is one of many problems solved for Volvo Cars and others using motion planning software developed by the Fraunhofer Chalmers Centre (FCC). Using motion planning software, engineers need only specify the high-level task of performing the

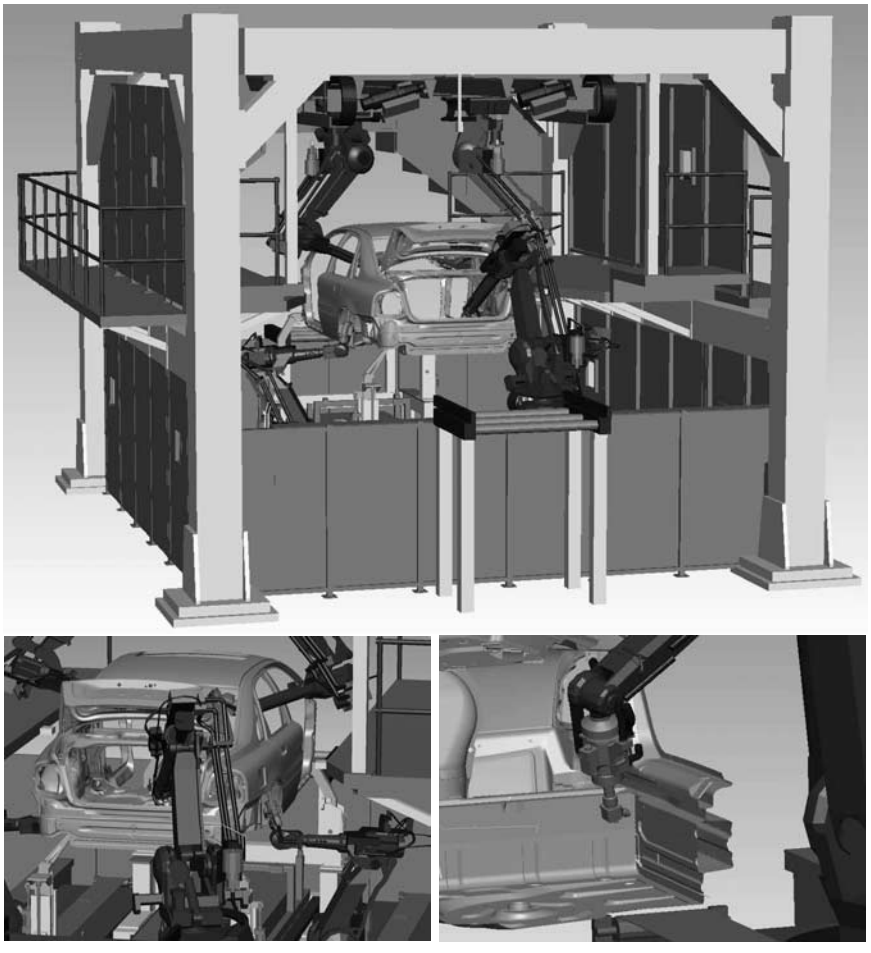


Figure 1.4: An application of motion planning to the sealing process in automotive manufacturing. Planning software developed by the Fraunhofer Chalmers Centre (FCC) is used at the Volvo Cars plant in Sweden (courtesy of Volvo Cars and FCC).

sealing, and the robot motions are computed automatically. This saves enormous time and expense in the manufacturing process.

Moving furniture

Returning to pure entertainment, the problem shown in Figure 1.5 involves moving a grand piano across a room using three mobile robots with manipulation arms mounted on them. The problem is humorously inspired by the phrase *Piano Mover's Problem*. Collisions between robots and with other pieces of furniture must be avoided. The problem is further complicated because the robots, piano, and floor form closed kinematic chains, which are covered in Sections 4.4 and 7.4.

Navigating mobile robots

A more common task for mobile robots is to request them to navigate in an indoor environment, as shown in Figure 1.6a. A robot might be asked to perform tasks such as building a map of the environment, determining its precise location within a map, or arriving at a particular place. Acquiring and manipulating information from sensors is quite challenging and is covered in Chapters 11 and 12. Most robots operate in spite of

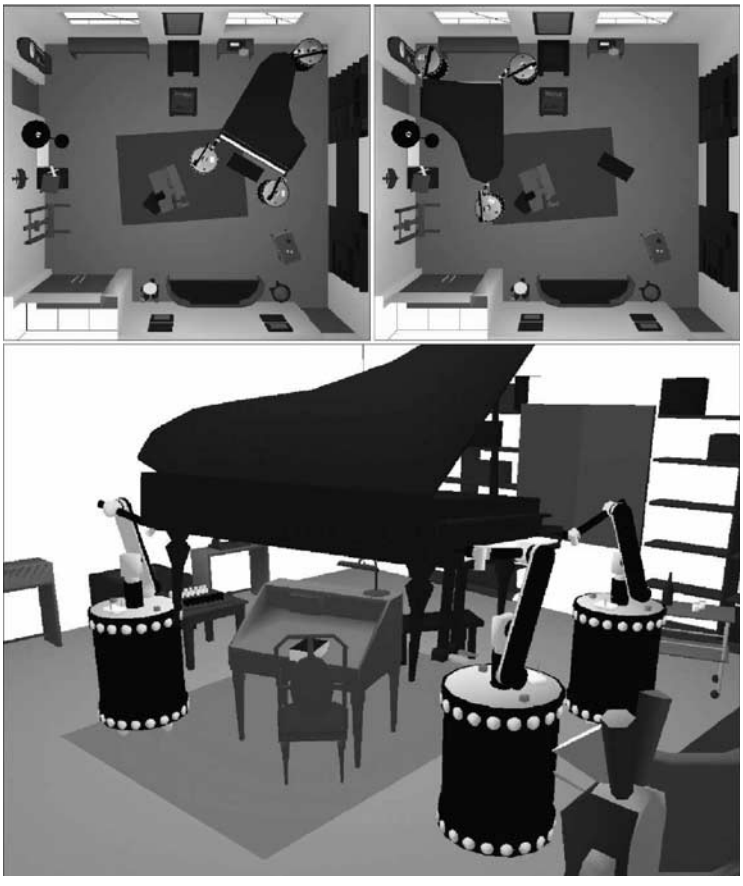
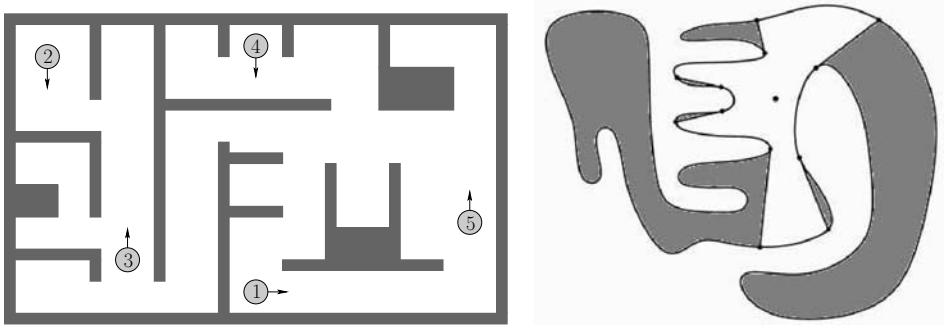


Figure 1.5: Using mobile robots to move a piano [246].



(a) (b)

Figure 1.6: (a) Several mobile robots attempt to successfully navigate in an indoor environment while avoiding collisions with the walls and each other. (b) Imagine using a lantern to search a cave for missing people.

large uncertainties. At one extreme, it may appear that having many sensors is beneficial because it could allow precise estimation of the environment and the robot position and orientation. This is the premise of many existing systems, as shown for the robot system in Figure 1.7, which constructs a map of its environment. It may alternatively be preferable to develop low-cost and reliable robots that achieve specific tasks with little

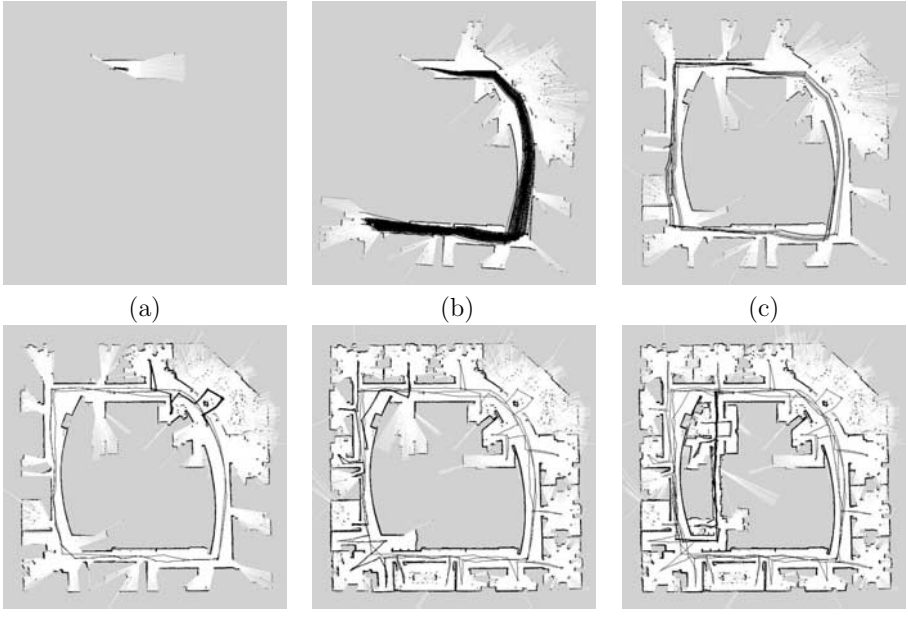


Figure 1.7: A mobile robot can reliably construct a good map of its environment (here, the Intel Research Lab) while simultaneously localizing itself. This is accomplished using laser scanning sensors and performing efficient Bayesian computations on the information space [353].

or no sensing. These trade-offs are carefully considered in Chapters 11 and 12. Planning under uncertainty is the focus of Part III.

If there are multiple robots, then many additional issues arise. How can the robots communicate? How can their information be integrated? Should their coordination be centralized or distributed? How can collisions between them be avoided? Do they each achieve independent tasks, or are they required to collaborate in some way? If they are competing in some way, then concepts from game theory may apply. Therefore, some game theory appears in Sections 9.3, 9.4, 10.5, 11.7, and 13.5.

Playing hide and seek

One important task for a mobile robot is playing the game of hide and seek. Imagine entering a cave in complete darkness. You are given a lantern and asked to search for any people who might be moving about, as shown in Figure 1.6b. Several questions might come to mind. Does a strategy even exist that guarantees I will find everyone? If not, then how many other searchers are needed before this task can be completed? Where should I move next? Can I keep from exploring the same places multiple times? This scenario arises in many robotics applications. The robots can be embedded in surveillance systems that use mobile robots with various types of sensors (motion, thermal, cameras, etc.). In scenarios that involve multiple robots with little or no communication, the strategy could help one robot locate others. One robot could even try to locate another that is malfunctioning. Outside of robotics, software tools can be developed that assist people in systematically searching or covering complicated environments, for applications such as law enforcement, search and rescue, toxic cleanup, and in the architectural design of secure buildings. The problem is extremely difficult because the status of the pursuit must be carefully computed to avoid unnecessarily allowing the evader to sneak

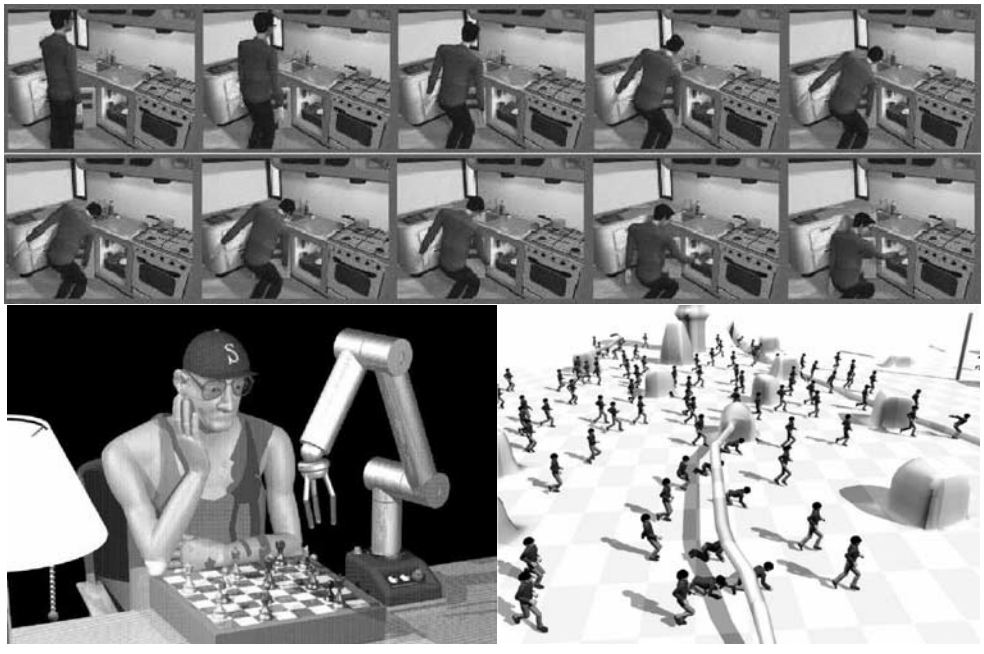


Figure 1.8: Across the top, a motion computed by a planning algorithm, for a digital actor to reach into a refrigerator [502]. In the lower left, a digital actor plays chess with a virtual robot [547]. In the lower right, a planning algorithm computes the motions of 100 digital actors moving across terrain with obstacles [594].

back to places already searched. The information-space concepts of Chapter 11 become critical in solving the problem. For an algorithmic solution to the hide-and-seek game, see Section 12.4.

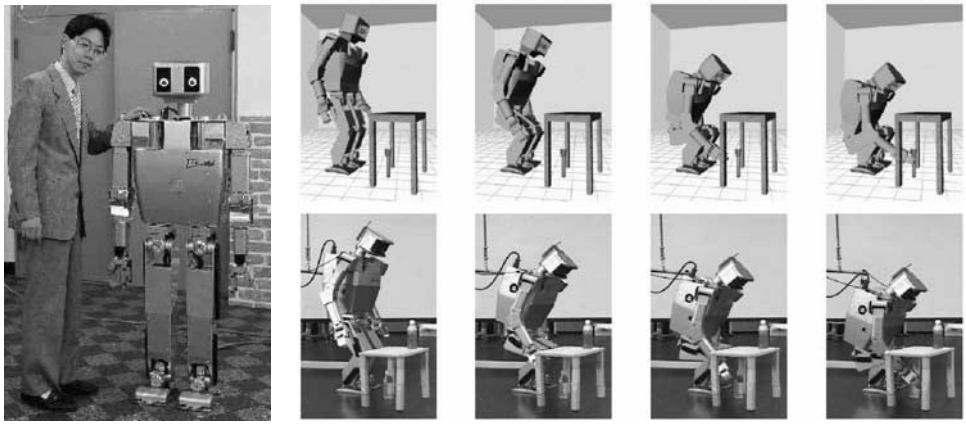
Making smart video game characters

The problem in Figure 1.6b might remind you of a video game. In the arcade classic *Pacman*, the ghosts are programmed to seek the player. Modern video games involve human-like characters that exhibit much more sophisticated behavior. Planning algorithms can enable game developers to program character behaviors at a higher level, with the expectation that the character can determine on its own how to move in an intelligent way.

At present there is a large separation between the planning-algorithm and video-game communities. Some developers of planning algorithms are recently considering more of the particular concerns that are important in video games. Video-game developers have to invest too much energy at present to adapt existing techniques to their problems. For recent books that are geared for game developers, see [154, 373].

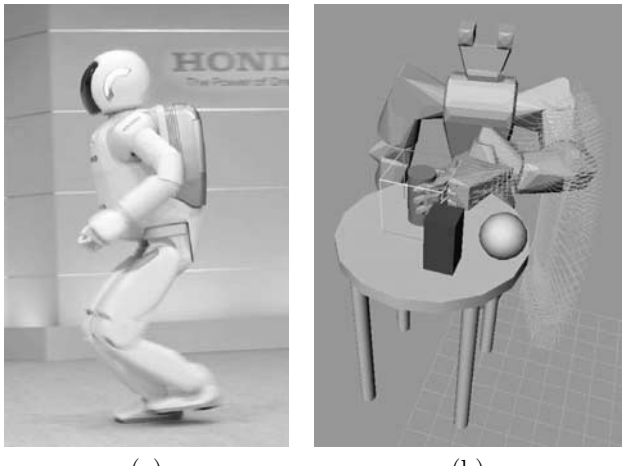
Virtual humans and humanoid robots

Beyond video games, there is broader interest in developing virtual humans. See Figure 1.8. In the field of computer graphics, computer-generated animations are a primary focus. Animators would like to develop digital actors that maintain many elusive style characteristics of human actors while at the same time being able to design motions for them from high-level descriptions. It is extremely tedious and time consuming to specify all motions frame-by-frame. The development of planning algorithms in this context is rapidly expanding.



(a) (b)

Figure 1.9: (a) This is a picture of the H7 humanoid robot and one of its developers, S. Kagami. It was developed in the JSK Laboratory at the University of Tokyo. (b) Bringing virtual reality and physical reality together. A planning algorithm computes stable motions for a humanoid to grab an obstructed object on the floor [564].



(a) (b)

Figure 1.10: Humanoid robots from the Japanese automotive industry: (a) The latest Asimo robot from Honda can run at 3 km/hr (courtesy of Honda); (b) planning is incorporated with vision in the Toyota humanoid so that it plans to grasp objects [451].

Why stop at *virtual* humans? The Japanese robotics community has inspired the world with its development of advanced humanoid robots. In 1997, Honda shocked the world by unveiling an impressive humanoid that could walk up stairs and recover from lost balance. Since that time, numerous corporations and institutions have improved humanoid designs. Although most of the mechanical issues have been worked out, two principle difficulties that remain are sensing and planning. What good is a humanoid robot if it cannot be programmed to accept high-level commands and execute them autonomously? Figure 1.9 shows work from the University of Tokyo for which a plan computed in simulation for a humanoid robot is actually applied on a real humanoid. Figure 1.10 shows humanoid projects from the Japanese automotive industry.