A GAME-THEORETIC FRAMEWORK FOR ROBOT MOTION PLANNING

BY

STEVEN MICHAEL LAVALLE

B.S., University of Illinois at Urbana-Champaign, 1990
M.S., University of Illinois at Urbana-Champaign, 1993

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1995

Urbana, Illinois

A GAME-THEORETIC FRAMEWORK FOR ROBOT MOTION PLANNING

Steven Michael LaValle, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1995
Seth A. Hutchinson, Advisor

The primary contribution of this dissertation is the presentation of a dynamic game-theoretic framework that is used as an analytical tool and unifying perspective for a wide class of problems in robot motion planning. The framework provides a precise mathematical characterization that can incorporate any of the essential features of decision theory, stochastic optimal control, and traditional multiplayer games. The determination of strategies that optimize some precise performance functionals is central to these subjects, and is of fundamental value for many types of motion planning problems.

The basic motion planning problem is to compute a collision-free trajectory for the robot, given perfect sensing, an exact representation of the environment, and completely predictable execution. The best-known algorithms have exponential complexity, and most extensions to the basic problem are provably intractable. The techniques in this dissertation characterize several extensions to the basic motion planning problem, and lead to computational techniques that provide practical, approximate solutions. A general perspective on motion planning is also provided by relating the similarities between various extensions to the basic problem within a common mathematical framework.

Modeling, analysis, algorithms, and computed examples are presented for each of three problems: (1) motion planning under uncertainty in sensing and control; (2) motion planning under environment uncertainties; and (3) multiple-robot motion planning. Traditional approaches to the first problem are often based on a methodology known as *preimage planning*, which involves worst-case analysis. In this context, a general method

for determining feedback strategies is developed by blending ideas from stochastic optimal control and dynamic game theory with traditional preimage planning concepts. This generalizes classical preimages to *performance preimages* and preimage plans to *motion strategies with information feedback*. For the second problem, robot strategies are analyzed and determined for situations in which the environment of the robot is changing, but not completely predictable. Several new applications are identified for this context. The changing environment is treated in a flexible manner by combining traditional configuration space concepts with stochastic optimal control concepts. For the third problem, dynamic game-theoretic and multiobjective optimization concepts are applied to motion planning for multiple robots. This allows the synthesis of motion plans that simultaneously optimize an independent performance criterion for each robot. Several versions of the formulation are considered: fixed-path coordination, coordination on independent configuration-space roadmaps, and centralized planning.

A GAME-THEORETIC FRAMEWORK FOR ROBOT MOTION PLANNING

Steven Michael LaValle, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1995
Seth A. Hutchinson, Advisor

The primary contribution of this dissertation is the presentation of a dynamic game-theoretic framework that is used as an analytical tool and unifying perspective for a wide class of problems in robot motion planning. The framework provides a precise mathematical characterization that can incorporate any of the essential features of decision theory, stochastic optimal control, and traditional multiplayer games. The determination of strategies that optimize some precise performance functionals is central to these subjects, and is of fundamental value for many types of motion planning problems.

The basic motion planning problem is to compute a collision-free trajectory for the robot, given perfect sensing, an exact representation of the environment, and completely predictable execution. The best-known algorithms have exponential complexity, and most extensions to the basic problem are provably intractable. The techniques in this dissertation characterize several extensions to the basic motion planning problem, and lead to computational techniques that provide practical, approximate solutions. A general perspective on motion planning is also provided by relating the similarities between various extensions to the basic problem within a common mathematical framework.

Modeling, analysis, algorithms, and computed examples are presented for each of three problems: (1) motion planning under uncertainty in sensing and control; (2) motion planning under environment uncertainties; and (3) multiple-robot motion planning. Traditional approaches to the first problem are often based on a methodology known as *preimage planning*, which involves worst-case analysis. In this context, a general method

for determining feedback strategies is developed by blending ideas from stochastic optimal control and dynamic game theory with traditional preimage planning concepts. This generalizes classical preimages to *performance preimages* and preimage plans to *motion strategies with information feedback*. For the second problem, robot strategies are analyzed and determined for situations in which the environment of the robot is changing, but not completely predictable. Several new applications are identified for this context. The changing environment is treated in a flexible manner by combining traditional configuration space concepts with stochastic optimal control concepts. For the third problem, dynamic game-theoretic and multiobjective optimization concepts are applied to motion planning for multiple robots. This allows the synthesis of motion plans that simultaneously optimize an independent performance criterion for each robot. Several versions of the formulation are considered: fixed-path coordination, coordination on independent configuration-space roadmaps, and centralized planning.

To my dad, Richard S. LaValle

ACKNOWLEDGMENTS

I would first like to thank my advisor, Professor Seth Hutchinson, for his encouragement and advice over the years and for providing me with the opportunities to learn and expand my interests. The quality of this work was greatly improved through his many helpful comments and suggestions.

I would also like to thank Professor Rajeev Sharma for his collaborative contributions to the work in Chapter 3, for being a co-supervisor of my Beckman Institute research assistantship, and for being a good friend. I am very grateful for the helpful insights and comments made by the remaining dissertation committee members: Professors Narendra Ahuja, Tamer Başar, Jean Ponce and Mark Spong. I also thank Dr. Ricardo Uribe for being a good friend and always challenging me to think in new directions.

I have been fortunate to have had the opportunity to meet with many researchers, which has led to many interesting research discussions. I would especially like to thank Randy Brost, Garry Didinsky, Mike Erdmann, Pierre Ferbach, Piotr Gmytrasiewicz, and Dan Koditschek, for their helpful comments.

Peggy Currid made many helpful comments and suggestions regarding the manuscript.

I am grateful for the many friends at the University of Illinois who have made my graduate-school days quite enjoyable. I especially thank Brian Calvert, Becky Castaño, Andrés Castaño, Scott Finkle, Keith Foszcz, Armando Fox, Jim Freeman, Fred Geiger, Gary Kalmanovich, Kevin Lee, Ken Moroney, Kevin Nickels, Sandeep Pandya, Lorraine

Sandford, Ilan Shimshoni, Narayan Srinivasa, and Steve Sullivan. I also thank Sharon Collins for her constant help and enjoyable sense of humor over the years.

I am grateful for having several excellent teachers at John F. Kennedy H.S. in Manchester, Missouri, who provided inspiration at a pivotal point in my life. I especially thank Nancy Wesselman for providing needed encouragement and convincing me of the value of education, and I thank Patricia Short for giving me a chance when almost no one else would. I also thank Mary John, Tom Smith, and Joseph Walsh.

I thank my family for being patient and understanding during my time in graduate school, especially my dad, Richard, my sister, Renee Martise, and my grandma, Anna Pasternak. I also thank my family in Poland and Ukraine for their boundless hospitality during my visits. I thank Bill Hurst for being my best friend for many years. I am very grateful for all of the love and support offered by my mom, Rose, who passed away while I was in graduate school.

Finally, I thank Jean-Claude Latombe for inspiring me to pursue motion planning when he visited Illinois and for offering me the chance to work with him at Stanford.

Portions of this work were sponsored by the National Science Foundation under grant #IRI-9110270, a Beckman Institute research assistantship, and a Mavis Fellowship.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Role of this Dissertation

We are confronted with a number of difficult issues as we strive to design robots that operate with a great deal of autonomy in complex or cluttered environments. Robots are requested to accomplish tasks such as assembling parts, exploring terrain, handling hazardous materials, delivering objects, and performing routine maintenance or surveillance. Advancement in such areas as robot control, sensor modeling and integration, grasping, manipulation, and motion planning continues to increase our understanding and ability to develop sophisticated robot systems that accomplish more difficult tasks.

The primary focus of this dissertation is on motion planning, which can be considered a topic of fundamental robotics research. Motion planning represents a task that is required to accomplish many larger tasks, in a variety of robotic systems. Such systems include mobile robots that navigate in potentially hostile environments, or multiple-link manipulators that must avoid collisions in a workcell. At a general level, motion planning can be considered as a way to simplify the interface between a human operator and a robotic system, by allowing the robot to automatically determine detailed motion commands that accomplish a particular task. This relieves the user from the burden of

specifying tediously detailed instructions for the robot. In its basic formulation, the goal of motion planning is to compute a plan that, when executed, will bring a robot from an initial configuration (or location) to a goal configuration, while avoiding obstacles in the robot's environment. Extensions of the basic problem incorporate such additional concerns as kinematic or dynamic constraints, compliant motions, imperfect sensing and execution, uncertain information about the environment, or the coordination of multiple robots.

The central theme in this dissertation is that dynamic game theory represents a general mathematical structure and unifying perspective on motion planning problems. Although game theory is sometimes viewed informally as a formulation of classical board games, *game theory* is used in this dissertation to refer to a more general mathematical formulation that can incorporate any of the essential features of statistical decision theory, Markov processes, optimal control theory, and interactions between multiple decision makers. In general, game theory should not be viewed as a direct solution method nor algorithm, but rather as a precise mathematical formulation of optimal decision-making problems. In the same sense that configuration space concepts have been useful for the presentation and comparison of different motion planning problems (see [109]), game theory represents a powerful set of concepts and tools that can be exploited in motion planning research.

Additionally, this mathematical formulation is useful for the development of computational methods that yield useful motion plans. Approaches to several extensions of the basic motion planning problem will be presented. These extensions might traditionally require quite different methodologies, but are attacked in this dissertation with a game-theoretic approach that yields novel results on the individual problems and a unifying view that relates the similarities between the problems.

The remainder of this chapter provides the general background and motivation for this dissertation. Section 1.2 presents a brief overview of motion planning research and related issues. Before describing a game-theoretic view of motion planning, Section 1.3 describes how *basic* motion planning can be characterized as an optimal control problem. By generalizing this optimal control problem to a game formulation, many extensions of the basic motion planning problem can be characterized. Such extensions include motion planning under various forms of uncertainty and planning for multiple robots. These problems are addressed in this dissertation from a game-theoretic perspective; the general motivation for formulating extended motion planning problems in this manner is the subject of Section 1.4. Section 1.5 indicates the topical organization of the remainder of this dissertation.

1.2 An Overview of Robot Motion Planning

Extensive surveys of motion planning research can be found in [91], [109]; in this section, we present only a brief overview of motion planning. Section 1.2.1 describes the basic motion planning problem. Section 1.2.2 provides an overview of motion planning under various forms of uncertainty. Section 1.2.3 discusses additional extensions of the basic motion planning problem.

1.2.1 The basic problem

Our description of the *basic motion planning problem* is a very brief summary of the presentation in [109]. This problem is also referred to as as *gross-motion planning* [91] or *collision-free planning*. A robot, $\mathcal{A}$, is considered as a rigid object that is capable of moving in a physical workspace, $\mathcal{W}$, which is usually defined as $\Re^n$ with $n = 2$ or $n = 3$. Mathematically, $\mathcal{A}$ is treated as a closed set that is allowed to undergo rigid body

transformations (positive isometries) in the workspace. Coordinate frames are attached to both the robot and the workspace, which are related through the specification of a vector quantity $\mathbf{q}$ that in general indicates the position and orientation of the robot frame with respect to the workspace frame. The quantity $\mathbf{q}$ is termed a *configuration* of $\mathcal{A}$. The set of possible configurations is termed the *configuration space*, and is denoted by $\mathcal{C}$, which can be represented as the subset $\Re^n \times SO(n)$ of order n, in which $SO(n)$ represents the Special Orthogonal Group. The configuration space in general is a manifold that admits a local parameterization of dimension six when $n = 3$, and dimension three when $n = 2$.

Alternatively, a nonrigid robot could be defined in the workspace. For instance, this is useful for modeling a multiple-link manipulator that has revolute or prismatic joints. The position of each joint can be defined in its own local coordinate frame, and the simultaneous specification of each joint angle will determine the configuration of the robot. Once the configuration space is defined, the motion planning problem can be analyzed in the same manner as for a rigid robot.

Let a collection of closed sets $\{\mathcal{B}_1, \ldots, \mathcal{B}_q\}$ be called a set of *obstacles*, for which each $\mathcal{B}_i \subset \mathcal{W}$. In the workspace, the robot is not allowed to "collide" with obstacles. Let the closed subset $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_q$ of the workspace be called the *obstacle region*. The open subset of $\mathcal{C}$ that corresponds to configurations in which the transformed robot does not intersect the obstacle region is referred to as the *free configuration space*, and is denoted by $\mathcal{C}_{free}$. This can be stated formally as

$$\mathcal{C}_{free} = \{\mathbf{q} \in \mathcal{C} | \mathcal{A}(\mathbf{q}) \cap \mathcal{B} = \emptyset\}, \tag{1.1}$$

in which $\mathcal{A}(\mathbf{q})$ represents the transformed robot, and $\mathcal{B}$ is the obstacle region $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_q$. The set $\mathcal{C} \setminus \mathcal{C}_{free}$ can be considered as the *obstacle region in configuration space*.

Let $\mathbf{q}_{init} \in \mathcal{C}$ denote an *initial configuration*, and $\mathbf{q}_{goal}$ denote a *goal configuration*. The basic motion planning problem can be formulated as follows:

Find a continuous path $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, *such that* $\tau(0) = \mathbf{q}_{init}$ *and* $\tau(1) = \mathbf{q}_{goal}$.

If the robot is permitted to "touch" the obstacle region, $\mathcal{C}_{free}$ is replaced by the *valid configuration space*, $\mathcal{C}_{valid}$, which is the closure of $\mathcal{C}_{free}$ (i.e., it includes the boundary of $\mathcal{C}_{free}$). This enables the possibility of guarded motion and compliance, which, for example, allows the robot to execute a motion along the tangent of an obstacle boundary [130], [195].

Figure 1.1 illustrates a simple motion planning example. Figure 1.1(a) shows a planar, rectangular robot that is only capable of translating in the plane. Suppose that the task is to bring the robot from its configuration shown in Figure 1.1(a) to the upper right portion of the workspace. Notice that there are two "openings" that lead into the upper right portion of the workspace. Figure 1.1(b) shows the configuration space in which the obstacle region is shaded. The point in the robot represents the origin of the robot's local coordinate system; this system is translated to produce $\mathcal{A}(\mathbf{q})$, in which $\mathbf{q}$ in this case represents a two-dimensional translation vector. Because the robot is not capable of rotation, there is only one "opening" through which the robot is allowed to travel while avoiding collision. Figure 1.1(c) shows a path in the configuration space that allows the robot to reach its goal configuration.

There are roughly three general approaches to the basic motion planning problem [109]: cell decomposition methods, roadmap methods, and artificial potential field methods. Cell decomposition methods partition the configuration space into regions within which a collision-free path is easy to determine, and this path can be easily connected to collision-free paths of adjacent regions. A solution to the motion planning problem is determined by searching for a sequence of collision-free cells that initially includes $\mathbf{q}_{init}$, and finally includes $\mathbf{q}_{goal}$. Approaches to configuration space partitioning can be categorized as *exact cell decomposition*, in which cells are typically constructed by determining

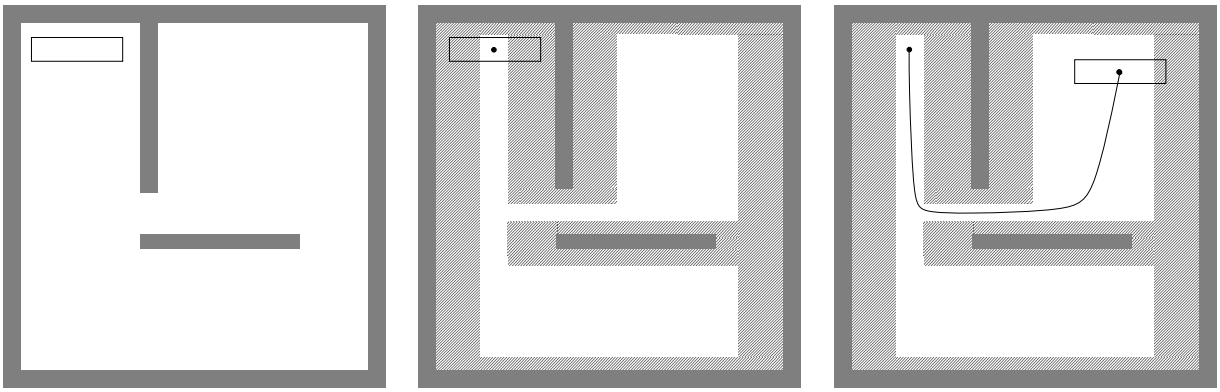


Figure 1.1 A basic, two-dimensional motion planning example is shown: (a) a planar robot translates in the plane and must avoid obstacles; (b) the free configuration space, $\mathcal{C}_{free}$ is indicated in white; (c) a path in $\mathcal{C}_{free}$ that can bring the robot to a goal configuration.

critical sets in an algebraic description of the configuration space (e.g., [4], [39], [166]), and *approximate cell decomposition*, in which each cell is assumed to be of the same basic shape, and the configuration space is usually decomposed in a hierarchical manner (e.g., [8], [109], [122]).

In a roadmap approach, a one-dimensional network of paths is constructed that preserves the connectivity of the free configuration space. A solution path is constructed by connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ to the roadmap and performing a graph search on the extended roadmap. The *visibility graph* approach generates a roadmap by connecting certain vertices of the boundary of the free configuration space, $\mathcal{C}_{free}$, and is primarily suitable for two-dimensional, polygonal configuration-space planning problems [54], [71], [125], [161]. The topological *retraction* operation has been used in a roadmap generation approach that continuously retracts $\mathcal{C}_{free}$ onto its generalized Voronoi diagram [138]. Other roadmap methods are described in [23], [31], [36].

In the prior two approaches, a path is constructed by analyzing the global structure of $\mathcal{C}_{free}$; however, in most artificial potential field methods, the robot moves locally according to some forces that are defined as the negative gradient of an artificial potential function. This approach was taken in [98] for real-time collision avoidance. The potential function is typically defined on $\mathcal{C}_{free}$ (although sometimes on the workspace [12]) as the sum of an attractive potential that pulls the robot toward the goal, and a repulsive potential that pushes the robot away from obstacles. One of the primary concerns in this approach is the existence of local minima in the potential function. In [92] heuristic search is performed over nominal paths at a global level, and in [11], [12] Brownian motion is combined with motion planning to escape local extrema. In [158], [159], [160] it has been shown that for some problems, a *global navigation function* can be constructed that ensures the existence of a single minimum that lies at the goal configuration.

Approaches to the basic motion planning problem are often evaluated with regard to completeness and computational complexity. An algorithm is considered to be *complete* if it is guaranteed to find a solution path whenever one exists. The exact cell decomposition method is complete, but extremely expensive with doubly-exponential time complexity in the dimension of the configuration space. The best-known complete method constructs a roadmap through geometric stratifications and has time complexity that is exponential in dimension [31]. Approximate cell decomposition methods usually achieve *resolution completeness*, which means that the algorithm can dynamically adjust the resolution until a solution is found; however, due to the hierarchical partitioning of n-dimensional cells, the time and space complexity grows quickly with the dimension of the configuration space [109]. Potential field methods are typically far more computationally efficient than the other approaches; however, completeness is usually sacrificed. A notable exception is the work of [160], which is a complete and efficient method, for a certain class of planning problems in which the free configuration space is diffeomorphic to a "star world."

1.2.2 Planning under uncertainties

The success of a motion planning approach in an implemented robotic system depends to a large extent on the manner in which various forms of uncertainty are modeled and treated. Motion planning under uncertainty therefore represents a very important extension of the basic problem and has received much attention from the motion planning community.

Two common representations of uncertainty have been applied to motion planning problems. One representation restricts parameter uncertainties to lie within a specified set. A motion plan is then generated that is based on *worst-case analysis* (e.g., [32], [62], [110], [124]). We refer to this representation as *nondeterministic uncertainty*. The other popular representation expresses uncertainty in the form of a probability density function. This often leads to the construction of motion plans through *average-case or expected-case analysis* (e.g., [26], [74], [170]). We refer to this case as *probabilistic uncertainty*.

Uncertainty can be introduced into a motion planning problem in a number of ways. We organize this uncertainty into four basic sources for discussion (as categorized in [117]). See Figure 1.2. We will describe each of the sources of uncertainty in isolation, although in general any combination of these types can be considered simultaneously in a motion planning formulation. The first two forms of uncertainty will be the topic of Chapter 2, and the remaining two forms will be the topic of Chapter 3.

Configuration-sensing uncertainty Suppose that $\mathcal{C}_{free}$ (or $\mathcal{C}_{valid}$) is given. Under uncertainty in configuration sensing, incomplete or imperfect information is utilized by the robot to make an inference about its configuration. This information could come from sensor measurements or motion history. With a nondeterministic uncertainty model, the robot might have sufficient information to infer that $\mathbf{q}$ lies in some subset $Q \subset \mathcal{C}_{free}$. For

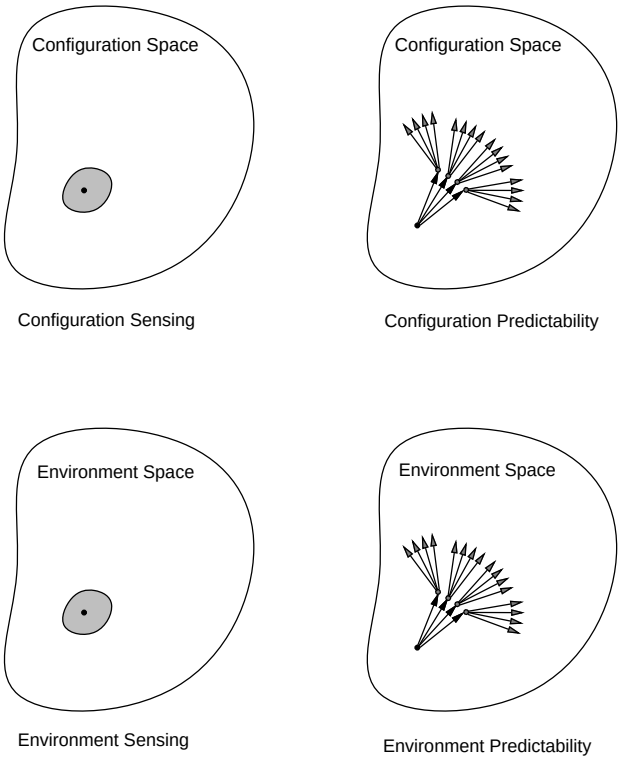


Figure 1.2 Four sources of uncertainty in the motion planning problem.

example, in [32], [62], [110], [124] this representation of uncertainty is used to guarantee that the robot recognizably terminates in a goal region. With a probabilistic model, the robot might infer a posterior probability density over configurations, $p(\mathbf{q})$, that is conditioned on sensor observations, initial conditions, or additional knowledge. Examples that handle configuration-sensing uncertainty with probabilistic representations include [26], [189].

Configuration-predictability uncertainty Suppose that both $\mathcal{C}_{free}$ and the current configuration, $\mathbf{q} \in \mathcal{C}_{free}$ are given. Motion commands can be given to the robot, but with control uncertainty the future configurations cannot, in general, be completely predicted. With nondeterministic uncertainty, the robot may infer that some future configuration will belong to a subset $Q \subset \mathcal{C}_{free}$. The method of *preimage backchaining* constitutes of

large body of work in which bounded uncertainties are propagated and combined with configuration-sensing uncertainty, to guarantee that the robot will achieve a goal (e.g., [32], [47], [62], [66], [110], [124]). With a probabilistic model, future configurations can be described by a posterior density over configurations, $p(\mathbf{q})$, that is conditioned on the initial configuration and the executed motion command. Examples of work that include a probabilistic representation of configuration-predictability uncertainty include [15], [26], [74].

Environment-sensing uncertainty Analogous to configuration-sensing uncertainty, suppose that a space of possible environments, E, is known to the robot. Although a space of configurations is a well-defined concept in robotics literature, we must define what is meant by a "space of environments." For the purpose of discussion, let E contain different possibilities for $\mathcal{C}_{free}$.

Under environment-sensing uncertainty, incomplete or imperfect information is utilized by the robot to make an inference about its environment. With a nondeterministic uncertainty model, the robot might have sufficient information to infer that the environment e belongs to some subset $F \subset E$. For example, in [140], [155], the environment is restricted to a plane populated with unknown polygonal obstacles, which are then discovered using visual "scans" to build a visibility graph for motion planning. In [127], unknown obstacles are allowed to be of arbitrary shape, and the sensor data consists of "tactile" information for a point robot. With a probabilistic model, the robot might infer a posterior probability density, $p(e)$, over environments, which is conditioned on sensor observations, initial conditions, or additional knowledge (e.g., [53], [56], [88], [185]).

Environment-predictability uncertainty Suppose again that the space of environments E is known by the robot; however, in addition, the robot knows its current en-

vironment $e \in E$. Predictable motion commands might be given to the robot, but with environment-predictability uncertainty, future environments cannot be completely predicted. With nondeterministic uncertainty, the robot may infer that some future environment will belong to a subset $F \subset E$. With a probabilistic model, future environments can be described by a posterior density over environments $p(e)$ that can be conditioned on the initial environment, the robot configuration, or an executed motion command. This form of uncertainty has received less attention, and will be elaborated on in Section 3.2.

1.2.3 Additional extensions

Several other extensions have received considerable attention in motion planning literature.

Multiple robots Suppose there are N robots $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^N$ that are capable of moving in a common workspace. One formulation of the multiple-robot motion planning task is to bring each robot from an initial configuration $\mathbf{q}^i_{init}$ to a goal configuration $\mathbf{q}^i_{goal}$, while avoiding collisions with obstacles and other robots.

Numerous approaches to multiple-robot motion planning have been considered in previous literature; a more detailed description of the multiple-robot motion planning problem and related issues will be presented in Chapter 4. Approaches are often categorized as *centralized* (as in, e.g., [11], [167]) or *decoupled* (as in, e.g., [61], [137]). A centralized approach typically constructs a path in a composite configuration space, which is formed by combining the configuration spaces of the individual robots. A typical decoupled approach independently generates a path for each robot, and then considers the interactions between the robots. The suitability of one approach over the other is usually determined by the tradeoff between the computational complexity associated with a given problem, and the amount of completeness that is lost.

Constrained motions In addition to geometric constraints imposed by obstacles, the motions of the robot might be further constrained. For instance, consider a four-wheeled mobile robot that has a steering mechanism with a bounded turning radius. This type of example is further described in Section 3.5.3.

Constraints can be categorized as *holonomic* or *nonholonomic*. A holonomic constraint is an algebraic restriction of the configuration space. A nonholonomic constraint is expressed by a nonintegrable, differential equation on the configuration space. This implies that velocities in the configuration space are restricted and must be taken into account in addition to geometric configuration space constraints.

Nonholonomic motion planning is often considered one of the most mathematically challenging extensions of the basic motion planning problem. An overview of the central difficulties in nonholonomic motion planning appears in [111]. Some recent references for nonholonomic motion planning include [13], [67], [118], [121].

Manipulation If the robot is allowed to interact with objects in its environment, many additional issues must be addressed. One of the most important manipulation planning concerns is the motion of the robot when it is in contact with obstacles. *Force compliant control* is usually permitted, which enables the robot to move along obstacle boundaries [130]. A variety of motion models have been considered for compliant control (e.g., [35], [148], [154], [169], [195], [196]). More complex interactions have also been analyzed, such as the motion of objects as they are pushed by a manipulator [25], [59], [128], [132], [191]. Object grasping also becomes an important concern when incorporated into a motion plan [24], [41], [76], [150]. In [1] a method for analyzing the transfer of objects in the workspace to accomplish a motion planning task is presented. Coordinated, multirobot manipulation has been considered in [90], [133], [203].

Moving obstacles Suppose that the workspace contains obstacles $\mathcal{B}_1(t), \ldots, \mathcal{B}_q(t)$ that may possibly be in motion. A motion planning problem in this environment can be analyzed by constructing a time-varying free configuration space, $\mathcal{C}_{free}(t)$ [109]. The *configuration-time space* can be constructed as an extension of the configuration space, and standard motion planning methods can be applied. One difficulty with this form of planning is the computational complexity. In [157], for example, Reif and Sharir show that motion planning in a three-dimensional environment is PSPACE-hard when the robot's velocity is bounded, and NP-hard when there is no such bound. In [30] Canny and Reif prove the NP-hardness of a more restricted problem—that of motion planning for a point robot in the plane with bounded velocity, when the moving obstacles are convex polygons moving at constant linear velocities without rotation. These intractability results have encouraged heuristic approaches, especially as applied to more limited geometric models—for example, in [69], [96], [175].

1.3 Basic Motion Planning as an Optimal Control Problem

In this section we describe how basic motion planning can be characterized as an optimal control problem. This type of characterization was introduced for motion planning in [72]. In this dissertation, extensions of the basic motion planning problem will be viewed as extensions of the optimal control problem. It is, therefore, useful to characterize basic motion planning as an optimal control problem to make these extensions more clear. It is important to note that we are not using control theory in this section to study such issues as robot dynamics, controllability, or stability, but simply to recast the basic motion planning problem.

Optimal control concepts Optimal control theory is a vast subject, and only some key definitions are provided here. A more thorough introduction to optimal control theory can be found in [2], [28], [106]. Let $X \subseteq \Re^n$ represent a *state space* in which $x_0 \in X$ represents the initial state of a system. Let $u : [0, t_f] \rightarrow \Re^n$ represent a *control function* in which $[0, t_f]$ represents an interval of time. The control at time t is given by $u(t)$, and the system state at time t is given by $x(t)$. The *system equation* can be represented as $\dot{x} = f(x(t), u(t))$, which defines how the state will evolve over time.

A *loss functional* is defined that evaluates any state trajectory and control function:

$$L(x(\cdot), u(\cdot)) = \int_0^{t_f} l(x(t), u(t))dt + Q(x(t_f)). \tag{1.2}$$

The integrand $l(x(t), u(t))$ represents an instantaneous cost, which when integrated can be imagined as the total amount of energy that is expended. The term $Q(x(t_f))$ is a final cost that can be used to induce a preference over trajectories that terminate in a particular portion of the state space by penalizing the final state of the system. One can also take $t_f = \infty$ and describe an asymptotic final state $x(\infty)$.

Suppose that the initial state, x_0, is given. The optimal control design task is to select a control function $u(\cdot)$ that causes Equation (1.2) to be minimized.

Encoding the basic problem Recall that the goal of the basic motion planning problem is to compute a path $\tau[0, 1] \rightarrow \mathcal{C}_{free}$ such that $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$, when such a path exists. The natural choice for the state space is $X = \mathcal{C}_{free}$. If robot dynamics were also included in the problem specification, then X might be expanded to include time derivatives on the configuration space.

We next define a simple system equation, $f(x(t), u(t)) = u(t)$ for all t. This is not intended to be the most reasonable model of a particular robotic system, but rather it

is used to encode the basic motion planning problem. We can assume for all t that the control input is either normalized, $\|u(t)\| = 1$, or $u(t) = 0$.

The initial state of the system is fixed, $x_0 = \mathbf{q}_{init}$. The loss functional can be simplified to $L(x(\cdot), u(\cdot)) = Q(x(t_f))$. We take $Q(x(t_f)) = 0$ if $x(t_f) = \mathbf{q}_{goal}$, and $Q(x(t_f)) = 1$ otherwise. This partitions the space of admissible controls into two classes: control functions that cause the basic motion planning problem to be solved receive zero loss; otherwise, unit loss is received.

The motion planning problem requires a collision-free path. This can be obtained by mapping the space of control functions into the space of state trajectories. For a given $u(t), t > 0$ and x_0, a state trajectory $x_u(t), t > 0$ can be completely predicted. If $L(x_u(\cdot), u(\cdot)) = 0$, then the determined state trajectory is a solution to the basic motion planning problem, which can be expressed as $\tau(s) = x_u(s\, t_f)$.

Incorporating optimality The previous formulation considered all control inputs that achieve the goal to be equivalent. By changing the loss functional, the optimal-path-length motion planning problem can be formulated:

$$L(x(\cdot), u(\cdot)) = \begin{cases} \int_0^{t_f} \|\dot{x}\| dt & \text{if } x(t_f) = \mathbf{q}_{goal} \\ \infty & \text{otherwise} \end{cases} . \tag{1.3}$$

The term $\int_0^{t_f} \|\dot{x}\| dt$ measures path length, and recall that $\dot{x}(t) = u(t)$ for all t.

It is well known that the optimal path generally maps into the closure of $\mathcal{C}_{free}$ (for example, see [109], which discusses the search for semi-free paths). Because $\mathcal{C}_{free}$ is open, we define the state space for this case to be $X = \mathcal{C}_{valid}$. A variety of other possibilities exist for defining the loss functional. For example, Gilbert and Johnson defined a loss functional for motion planning that takes into account the distance between the robot and obstacles [72].

A discrete-time representation The motion planning problem can alternatively be characterized in discrete time. For the systems that we will consider, discrete-time representations can provide arbitrarily close approximations to the continuous case, and will be used throughout this dissertation. It will be seen in coming chapters that discrete-time representations can simplify the development of computational methods.

With the discretization of time, $[0, t_f]$ is partitioned into *stages*, denoted by $k \in \{1, \ldots, K+1\}$. Stage k refers to time $(k-1)\Delta t$. If t_f is finite, the final stage is given by $K = \lfloor t_f / \Delta t \rfloor$. Let x_k represent the state at stage k. At each stage k, an *action* u_k can be chosen. Because

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t}, \tag{1.4}$$

the state transition equation can be approximated as

$$x_{k+1} = x_k + u_k. \tag{1.5}$$

As an example of how this representation approximates the basic motion planning problem, consider the following example. Suppose $\mathcal{C}_{free} \subseteq \Re^2$. It is assumed that $\|u_k\| = 1$ and, hence, the space of possible actions can be sufficiently characterized by the parameter $\phi_k \in [0, 2\pi)$. The discrete-time state transition equation becomes

$$x_{k+1} = x_k + \Delta t \begin{bmatrix} cos(\phi_k) \\ sin(\phi_k) \end{bmatrix}. \tag{1.6}$$

At each stage, the direction of motion is controlled by selecting ϕ_k. Any K-segment polygonal curve of length $K\Delta t$ can be obtained as a possible state trajectory of the system. If an action is included that causes no motion, shorter polygonal curves can also be obtained.

A discrete-time representation of the loss functional must also be defined:

$$L(x_1, \ldots, x_{K+1}, u_1, \ldots, u_K) = \sum_{k=1}^{K} l_k(x_k, u_k) + l_{K+1}(x_{K+1}), \tag{1.7}$$

in which l_k and l_{K+1} serve the same purpose as l and Q in the continuous-time loss functional.

The basic motion planning problem can be represented in discrete time by letting $l_k = 0$ for all $k \in \{1, \ldots, K\}$, and defining the final term as $l_{K+1}(x_{K+1}) = 0$ if $x_k = \mathbf{q}_{goal}$, and $l_{K+1}(x_{K+1}) = 1$ otherwise. This gives equal preference to all trajectories that reach the goal. To approximate the problem of planning an optimal-length path, $l_k = 1$ for all $k \in \{1, \ldots, K\}$. The final term is then defined as $l_{K+1}(x_{K+1}) = 0$ if $x_k \in \mathbf{q}_{goal}$, and $l_{K+1}(x_{K+1}) = \infty$ otherwise.

Implications The previous formulations have shown an equivalence between the basic motion planning problem and a specific version of the optimal control problem. Therefore, an algorithm that solves a basic motion planning problem equivalently solves a specific optimal control problem. For example, the visibility graph approach can be considered as a method for determining an optimal controller for a particular optimal control problem in $\Re^2$ with polygonal constraints on the state space and the loss functional (Equation (1.3)).

One key difference between this optimal control problem and those typically considered in control theory literature is the set of geometric constraints on the state space that appear because of obstacles in the workspace. These constraints represent an unusual twist to the control problem and must be confronted in a robotics context. Another difference is that a goal (or reference) trajectory that serves as a reference for comparing solutions is often specified for standard control problems; however, in basic motion planning the primary task is to select the trajectory (the collision-free path).

The optimal control formulations that are considered in this section are limited, however, in a number of ways: (1) only open-loop control functions are considered; (2) no form of uncertainty is assumed; and (3) only a single robot is being controlled. One

purpose of this dissertation is to generalize this specific optimal control formulation to a useful mathematical structure that characterizes many extended motion planning problems.

1.4 Extending the Basic Motion Planning Problem with Game Theory

This section provides an overview of how game-theoretic concepts can be used to model and analyze extensions of the basic motion planning problem. The discussion is presented at a high level because specific modeling details are covered in the individual chapters that handle particular motion planning problems. In Chapter 5 a detailed mathematical structure is presented that unifies a broad class of motion planning problems, including those particularly addressed in this dissertation. In a sense, Chapter 5 can be considered as a continuation of the current section in which more detailed arguments are made through the support of its preceding chapters. Section 1.4.1 provides an overview of game theory as it is used in this dissertation. Section 1.4.2 describes how motion planning concepts can be combined with game theory. Section 1.4.3 discusses the general benefits of this framework.

1.4.1 An overview of game theory

The subject of game theory has been pursued for over sixty years, leading to a wide variety of literature and viewpoints on the subject. In this dissertation *game theory* is used to describe a dynamic (or sequential) decision-making problem that involves multiple decision makers and one or more loss functionals (or performance criteria). In this case, its use is much more general than a "game" in the intuitive sense. The formulations

presented in this dissertation share similarities with concepts from statistical decision theory (e.g., [20], [46], [45]), optimal control theory (e.g., [28], [105], [202]), dynamic noncooperative game theory [6], and games considered in artificial intelligence research (e.g., [149], [198]).

The amount of cooperation that occurs between decision makers in a game is one of the key differences between different branches of game theory literature. If the decision makers act in unison but each has different loss functionals, the *multiobjective optimization problem* is obtained [85], [164], [205]. In a situation in which there is a common loss functional and all decision makers wish to act cooperatively, *team theory* is obtained [37], [86], [99]. If some subsets of the decision makers can choose their actions in unison, so that a mutually beneficial outcome can be obtained, we have a *cooperative game* [18], [141]. Games of this type can sometimes be cast as optimization problems in which some function of the individual decision maker performance criteria is optimized. Without cooperation the decision makers choose actions that take into account interests that conflict with the other decision makers. This is referred to as a *noncooperative game*. The most extreme case of conflicting interest is a *zero-sum* game, in which two decision makers are diametrically opposed. A "solution" to a game of the noncooperative type is referred to as an *equilibrium* because it represents a game strategy that provides a balance between the independent interests of the decision makers. One well-studied branch of noncooperative game theory involves problems of pursuit and evasion [81], [94], [199], [200], [201].

Game theory can also model a situation in which some decision makers represent disturbances that must be overcome by the other decision makers. For instance, uncertainty can be introduced into the state transition equation of an optimal control problem. As will be discussed shortly, such problems can be viewed as a *game against nature* [144]. If this uncertainty is represented probabilistically, the game against nature becomes a

problem of *stochastic optimal control theory* [17], [105], which represents a well-studied branch of control theory. If the uncertainty is represented nondeterministically, worst case analysis is performed, and the game against nature can be considered as a form of robust controller design [5].

The participants in a game are referred to as *decision makers* (or players). The optimal control problem that was formulated in Section 1.3 can be considered as a degenerate game that involves only a single decision maker. Each decision maker has available to it a set of *actions* (or control functions). With a discrete-time formulation, the decision makers repeatedly choose actions; with a continuous-time formulation, a control function is specified [6], [28], [151]. The decision makers are capable of manipulating the game *state*. A *state transition equation* defines how the state space changes as a function of the current state and the actions of the players.

State measurements are specified as mappings from the state space to an *observation* space. This can model, for instance, the projection from the workspace or configuration space to a sensor space, which has been investigated in the robotics literature [50], [100]. An *information space* can be defined that is generated by some amount of history or memory that a decision maker maintains. In general there are two types of information that are considered: (1) actions taken in the past by one or more decision makers, and (2) state measurements. Each point in the information space represents a particular history that has resulted, and is referred to as an *information state*.

A *strategy* specifies the behavior of a decision maker by conditioning its actions on the information state. A strategy could be specified deterministically, yielding a *pure strategy* or *deterministic strategy*, or it could be specified with probability distributions over the sets of actions, yielding a *mixed strategy* or *randomized strategy*.

The effects of noise or disturbances can be modeled by the addition of a special decision maker called *nature* [20]. One of the primary interests in this dissertation is

the modeling and incorporation of uncertainty into a robot strategy. Nature might be assumed to have a known, randomized strategy that interferes with the efforts of the decision makers. Nature could alternatively be assumed to implement a nondeterministic strategy. A game that involves only one "true" decision maker and nature is often referred to as a *game against nature* [20], [190].

The selection of a strategy for a decision maker is governed by its real-valued *loss functional*. The loss functional in Section 1.3 was used to select the best control function as a solution to the basic motion planning problem. Typically, the loss functional will not directly evaluate the strategy, but instead will evaluate the full history of actions and states that occurred. Given one or more loss functionals, an appropriate concept of optimality must be chosen.

1.4.2 A synthesis of motion planning and game theory concepts

This section relates game-theoretic concepts to robot motion planning at a general level. In robotics, some connections have previously been drawn between certain motion planning problems and specific forms of game theory. For instance, in [190] it was argued at an abstract level that robot manipulation planning in the presence of probabilistic uncertainty can be considered as a game against nature (which is equivalent to the stochastic control problem [105]). In [52], [79] team theory was proposed for planning the locations of sensors and integrating sensor information. A two-person zero-sum game was defined in [80] that models robustness with respect to uncertainty in a sensor-based robot plan. In [3], a two-player differential game was defined, which leads to the coordination of two independent robot manipulators.

First, consider the "decision makers" involved in a robot motion planning game. Each robot that has the potential to make decisions can be considered as a decision maker in

a game. Different aspects of uncertainty can be modeled as decisions that are made by nature.

In robot motion planning research, the notion of a configuration space has been extremely useful for the comparison and analysis of problems (e.g., [109]). In game theory, the state space serves the same basic purpose, which is to precisely represent each situation that can occur by a point in a space. For a basic motion planning problem that involves a single robot, a natural choice for the state space would be $\mathcal{C}_{free}$ or $\mathcal{C}_{valid}$. With the multiple robot problem considered in Chapter 4, we will define the state space as a subset of the Cartesian product of the configuration spaces for each of the robots. For the application in Chapter 3, the state space represents one copy of $\mathcal{C}_{free}$ for each possible mode of an environment. If dynamic constraints are also relevant to the planning problem, then the state space can be expanded to incorporate these additional parameters.

Once a state space is selected for a given problem, the state transition equation provides local laws of motion for the robot(s). This view perhaps shares the most similarities with the artificial potential field approach to motion planning, in which the robot often considers a sequence of local motion decisions that lead to the achievement of the goal. For most of the problems considered in this dissertation, the motions of the robots are not very constrained by the state transition equation: they are usually allowed to move at a fixed velocity in any direction in the state space, except when prohibited by obstacles. One exception is in Section 3.5.3, in which results for a nonholonomic, car-like robot are presented. In general, the state transition equation can be used to model very complex motions.

Concepts similar to the state space and state transition equation have appeared in motion planning literature; however, two key differences in our context are the additional concepts of strategy and loss. These produce a different line of reasoning when determining a solution to the problem. As stated previously, the solution representation for

the basic motion planning problem is a path. In this dissertation, the solution representation is a strategy, which can be considered as a controller that is based on information feedback. This distinction is particularly important when uncertainty is involved, since it allows a robot to dynamically choose its actions based on current information. This type of behavior is similarly achieved with conditional multistep plans in preimage planning [110].

Loss functionals are used to explicitly guide the selection of a strategy. Once a concept of optimality is defined, a statement of the ideal strategy can be expressed in terms of the loss functional(s). As described in Section 1.3, the loss functional is useful for evaluating a control function, and in the game-theoretic sense, loss functional(s) are useful for evaluating a strategy.

1.4.3 Benefits of the proposed framework

The primary claim of this dissertation is that game theory represents a unifying mathematical structure for a wide class of problems in motion planning. Section 1.4.2 indicated how game theory can be used in general to describe motion planning problems. This section states some of the benefits of this characterization at a general level. More specific benefits are described in Chapters 2 through 5, after the background and motivations are given for particular problems.

One concern might be that the generality offered by a game-theoretic formulation comes at the cost of sacrificing important parts of the problem description. Generalizations that relate fairly distinct problems can sometimes be obtained by abstracting away details. In almost any case, the construction of a mathematical model for a physical process involves the abstraction of certain details for the purpose of analysis and obtaining solutions. It is important, however, to understand that we are using game

theory to describe problems that are already models of physical robotics problems. The game-theoretic formulation does not necessarily omit any of the details that were part of an original motion planning problem formulation. In a basic motion planning problem, $\mathcal{C}_{free}$ (along with $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$) completely encodes a problem and, therefore, directly defines the state space.

The game structure allows the basic model to be expanded in a useful way that does not necessarily eliminate details from the original formulation. Once a problem has been mathematically characterized, there are a number of clear directions along which it can be generalized. A problem generalization might combine the fundamental aspects of two distinct problems that were already analyzed. For instance, if solutions are known for a two-robot planning problem, and for a problem with one robot that faces environment-predictability uncertainty, then a game structure can be defined that models two robots in a changing uncertain environment.

Alternatively, once a specific motion planning application has been encoded in game-theoretic terms, individual aspects of that application can then be generalized in a straightforward manner. For example, the preimage concept has been useful in specific motion planning problems that involve configuration-predictability uncertainty. By using the game-theoretic formulation, this concept is generalized to a broad class of problems in Chapter 2, and further generalized in Section 5.3. The notion of a performance preimage is obtained that can be utilized for problems with probabilistic representations, information-feedback strategies, multiple robots, etc.

The perspective provided by a general game structure can also serve as a basis for comparing different planning problems. It was shown in [109] that many motion planning problems that appear distinct can be shown to be nearly equivalent by mapping the problems into the configuration space. The same type of situation can occur with a

general game formulation because each problem can be represented with components that are embedded in a common mathematical structure.

The game formulation still might not appear useful if the solutions to any of the formulated problems were impossible to obtain. However, one advantage is that there are very powerful optimization tools from control theory that can be exploited. One of the most useful concepts is the principle of optimality [16], which states that an optimal solution can be recursively decomposed into optimal parts. This results in a dynamic programming methodology, which forms the basis of the computational methods for all of the problems investigated in this dissertation. In general, this optimization concept has been useful in a variety of contexts, both for producing analytical solutions and for numerical computation procedures.

The class of problems that can be solved by directly using the principle of optimality is fairly restrictive. In both control theory and game theory, the classic set of problems that can be solved are those with a linear state transition equation and quadratic loss functional [2], [6], [28]. As an example, the principle of optimality forms the basis of the analytic solution to the linear-quadratic Gaussian (LQG) optimal control problem [105]. Because few problems can be solved analytically, there has been a large focus on numerical dynamic programming procedures [17], [107], [108], even in robotics applications [9], [89], [135], [186].

In many situations, the game-theoretic formulation cannot be expected to directly yield a numerical solution, particularly if the dimension of the state space or information space is high. Nevertheless, we believe that the framework can serve as a reference for comparing partial or approximate solutions to a problem. This type of comparison occurs with Bayesian decision theory (of which game theory is a generalization). In practice, the optimal Bayes decision rule often cannot be evaluated; however, methods such as the nearest-neighbor decision rule are developed in which the performance can

be related to that of the optimal rule [46], [51]. Hence, the tradeoffs can sometimes be determined between an optimal solution that is computationally prohibitive, and a proposed approximate or heuristic solution.

1.5 Dissertation Organization

This chapter has provided a general introduction and background for this dissertation. Chapters 2 through 4 present detailed analysis and computation methods for specific classes of motion planning problems.

Chapter 2 handles motion planning under uncertainty in configuration predictability and configuration sensing (commonly referred to as planning under uncertainty in sensing and control [62], [124], [110]). Traditional approaches are often based on a methodology known as *preimage planning*, which involves worst-case analysis. We have developed a general method for determining feedback strategies by blending ideas from stochastic optimal control and dynamic game theory with traditional preimage planning concepts. This generalizes classical preimages to *performance preimages* and preimage plans to *motion strategies with information feedback*. For a given problem, one can define a performance criterion that evaluates any executed trajectory of the robot. We present methods for selecting a motion strategy that optimizes this criterion under either nondeterministic uncertainty (resulting in worst-case analysis) or probabilistic uncertainty (resulting in expected-case analysis). We present dynamic programming-based algorithms that numerically compute forward projections (a concept used in previous motion planning literature), performance preimages, and optimal strategies; several computed examples of forward projections, performance preimages, and optimal strategies are presented.

Chapter 3 handles motion planning under uncertainty in environment predictability and environment sensing. This results in a general method for analyzing and computing

motion plans for a robot that operates in an environment that changes over time in an uncertain manner. The changing environment is treated in a flexible manner by combining traditional configuration space concepts with a Markov process that models the environment. A motion strategy provides a motion command for the robot for each contingency with which it could be confronted. We allow the specification of a desired performance criterion, such as time or distance, and determine a motion strategy that is optimal with respect to that criterion. Computed examples are presented for problems that involve changing configuration spaces, hazardous regions and shelters, processing of service requests, and delivering objects that appear in different locations. Several extensions of the basic framework that incorporate additional concerns, such as sensing issues or changes in the geometry of the robot, are also presented.

Chapter 4 describes two contributions to geometric motion planning for multiple robots: (1) motion plans can be determined that simultaneously optimize an independent performance criterion for each robot; and (2) a general spectrum is defined between decoupled and centralized planning, in which we introduce the concept of coordination along independent roadmaps. By considering independent performance criteria, we introduce a form of optimality that is consistent with concepts from multiobjective optimization and game theory research. Previous multiple-robot motion planning approaches that consider optimality combine individual criteria into a single criterion. As a result, these methods can fail to find many potentially useful motion plans. Roadmap coordination is useful because greater flexibility is obtained over fixed-path coordination without the computational burden of optimizing strategies in a higher-dimensional space. We present implemented, multiple-robot motion planning algorithms that are derived from applying the principle of optimality to a partially-ordered space, for three problem classes along the spectrum between centralized and decoupled planning: (1) coordination along fixed, independent paths; (2) coordination along independent roadmaps; and (3) general,

unconstrained motion planning. Several computed examples are presented for all three problem classes that illustrate the concepts and algorithms.

Chapter 5 presents a general mathematical structure that unifies the concepts presented in Chapters 2 through 4, and provides a discussion about future research based on this perspective. Chapter 6 summarizes the contributions made in this dissertation.

CHAPTER 2

MOTION PLANNING UNDER UNCERTAINTY IN SENSING AND CONTROL

2.1 Introduction

It has been widely acknowledged in the literature that modeling and overcoming uncertainty is crucial to successful motion planning in practical environments. This chapter focuses on planning under uncertainty in configuration sensing and configuration predictability, which were discussed in Section 1.2.2. In motion planning literature, this problem has also been described as motion planning under uncertainty in sensing and control.

Traditional approaches are often based on a methodology known as *preimage planning*, which involves worst-case analysis. In this chapter, a general approach is presented for determining feedback strategies by blending traditional preimage planning ideas with concepts from stochastic optimal control and dynamic game theory. This generalizes classical preimages to *performance preimages* and preimage plans to *motion strategies with information feedback*. Both nondeterministic and probabilistic representations of uncertainty are considered. Constructing a traditional *termination criterion* becomes equivalent to formulating an *optimal stopping problem* [17] within our approach. Motion

strategies are selected that optimize a loss functional, and are computed by a dynamic programming-based algorithm.

Instead of avoiding collisions, we exploit interactions between the robot and objects, e.g., such operations as compliant motions, pushing, and grasping. Many researchers have considered the problem of achieving a goal configuration under uncertainty while permitting compliant motions; this has been referred to as the *fine-motion* planning problem in, e.g., [32], [62], [110], [124], and the *manipulation planning* problem in, e.g., [26]. In other research, the phrase *manipulation planning* has been used to refer to problems that involve the transfer of objects in the workspace (e.g., [1], [109]), but we do not consider such problems in this chapter.

We use the term *manipulation planning* in this chapter to refer specifically to the problem of achieving a goal configuration under uncertainty in sensing and control while permitting compliant motions. The methods presented in this chapter can also be adapted to collision-free planning (we have computed results for many cases); however, in this chapter we focus only on manipulation planning because the basic collision-free planning issues under sensing and control uncertainties are included in the manipulation planning model. Section 2.2 provides an overview of manipulation planning research and discusses the significance of our approach.

Section 2.3 provides the general definitions and concepts that form the basis of our approach. Some of the notation, although not previously used to characterize manipulation planning problems, is borrowed from control theory (as used in [6], [105] and Section 1.3). After the general concepts and definitions have been presented, they are utilized in Sections 2.4, 2.5, and 2.6 to present forward projections, performance preimages, and the determination of optimal motion strategies, respectively. Computed examples are presented at the conclusion of each of these three sections.

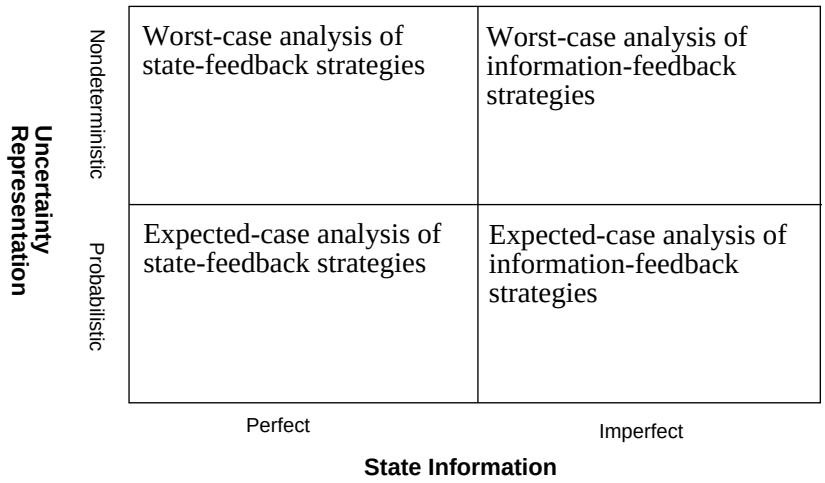


Figure 2.1 Four types of uncertainty that are considered in this chapter.

The concepts in each of Sections 2.4 through 2.6 can be logically divided into four components that represent different types of uncertainty (see Figure 2.1). The precise meaning of these types of uncertainty should become clearer after reading Sections 2.2 and 2.3. These types are based on two choices: (1) using probabilistic representations versus nondeterministic (or bounded-set) representations; and (2) modeling control errors and assuming perfect sensing versus modeling both control and sensing errors. Probabilistic and nondeterministic representations each offer distinct advantages; hence, both are included in our approach. Probabilistic representations lead to expected-case analysis, and nondeterministic representations lead to worst-case analysis; relevant issues are discussed in Section 2.2. By assuming perfect sensing, we can observe many effects of control uncertainty on manipulation planning. These results are applicable when reasonable state estimation (i.e., configuration estimation) can be performed. The addition of sensing uncertainty can be considered as an extension in which the configuration space is replaced by an *information space* derived from the sensing and action history. The motion planning analysis for this case is performed on this information space, as opposed to the original configuration space.

Section 2.7 discusses several issues regarding our current approach and how it can be extended. Section 2.8 provides some brief conclusions.

2.2 Background and Motivation

Section 2.2.1 provides an overview of previous manipulation planning research. Section 2.2.2 discusses the key contributions of this chapter in the context of previous research.

2.2.1 Prior research

Preimage planning constitutes a large body of research that assumes that sensing and control errors lie within bounded sets. The approach was first conceptualized by Lozano-Pérez et al. [124]. Geometric reasoning techniques are used to construct a robot plan that guarantees the robot will terminate in a specified subset of configuration space, regardless of where the errors might lie within the bounded sets. This plan is generally constructed using recursive subgoals, as a form of backchaining. For each subgoal, a *preimage* is formed that allows the robot to achieve the subgoal for a fixed command, starting from the subset of the configuration space attained from the previous subgoal. Figure 2.2 shows a simple example of a preimage. In general, a preimage is defined as the set of all configurations from which a robot is guaranteed to halt in the goal region under the execution of a motion command.

We will next discuss several important aspects of manipulation planning that have been elucidated in preimage planning research, and conclude Section 2.2.1 by summarizing some additional related motion planning research.

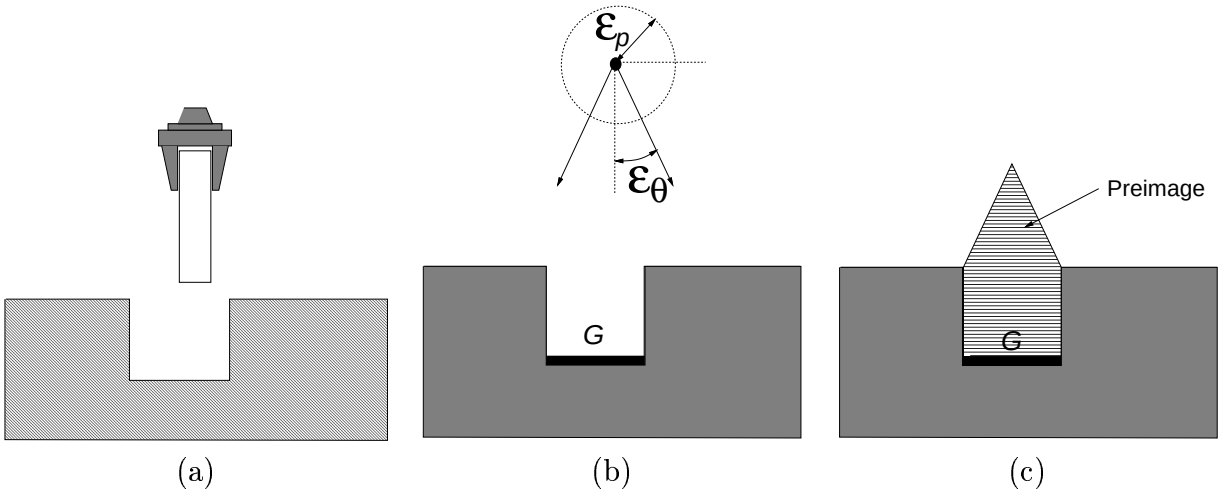


Figure 2.2 (a) A classic 2D "peg-in-hole" insertion task without rotation; (b) such a task can be represented in configuration space with bounded uncertainty in commanded velocity and sensed configuration (ϵ_θ denotes the angular uncertainty bound; ϵ_p denotes the positional uncertainty bound, and G denotes the goal region); (c) the classical preimage.

Motion models One of the most important manipulation planning concerns is the motion of the robot when it is in contact with obstacles. *Force compliant control* is usually permitted, which enables the robot to move along obstacle boundaries [35], [78], [130], [148], [154], [169], [195], [196]. As stated in Section 1.2.3, a variety of motion models have been considered for compliant control, and the *generalized damping model* has been most often considered in the context of manipulation planning [32], [62], [109], [110], [124], [195]. More complex interactions have also been analyzed, such as the motion of objects as they are pushed by a manipulator [25], [59], [128], [132], [191]. Figure 2.3 depicts the type of motion that occurs when the generalized damping model is combined with friction. If the commanded velocity does not point into the friction cone, but points into the obstacle boundary, then the robot moves along the boundary. If the commanded velocity points into the friction cone, then the robot remains motionless. When the robot is not in contact, the set of available motions could be constrained, as for nonholonomic

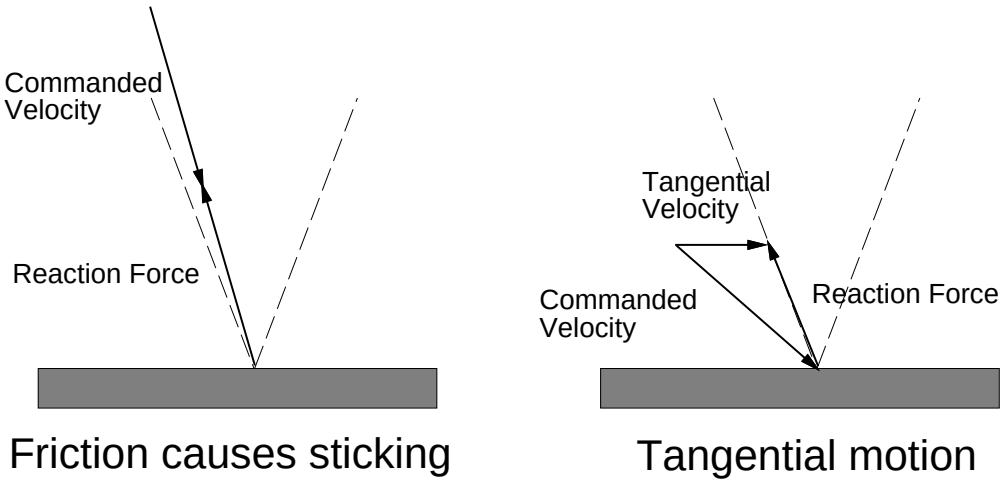


Figure 2.3 Motions under the generalized damping model with friction. The dashed lines indicate the friction cone.

robots as discussed in [13], or the robot might be free to move along any continuous path. For manipulation planning problems, it is thus important to allow different motions to be defined on different subsets of the configuration space.

Modeling uncertainty Two basic representations of sensing and control uncertainty have been proposed in the manipulation planning literature. We refer to these as *nondeterministic uncertainty* and *probabilistic uncertainty*, as done in [57]. Under nondeterministic uncertainty, it is assumed that parameter uncertainties lie in a bounded set. Worst-case analysis is performed to yield a motion plan that is guaranteed to be successful, regardless of the true value of uncertain parameters within the bounded set. This uncertainty representation is the most common in previous manipulation planning research. Under probabilistic uncertainty, probability densities are used to represent uncertainty associated with parameters. Each uncertainty representation offers advantages. Nondeterministic models do not require a statistical representation of the errors and, hence, are often easier to specify. If the uncertainty model is correct, the guarantee that the goal is achieved is useful, particularly when the penalty is severe for not achieving it.

As noted in [26], [57], a guaranteed motion plan does not exist for many tasks; however, a plan can alternatively be computed that achieves the goal with some probability.

History Operation under sensing uncertainty can be improved by incorporating sensing and/or control history into the robot's decision making. This is mathematically equivalent to the use of information feedback in control theory and dynamic game theory. Several approaches to manipulation planning have incorporated sensor information. Erdmann presented an approach that yields motion strategies that are conditioned on knowledge states [58]. These knowledge states are inferred from the sensing history and, at a high level, correspond closely to using information feedback. Latombe et al. used sensing and control history to infer a subset of configuration space defined as a *goal kernel*, in which the robot can successfully switch between commands in a multiple step plan, or terminate in the goal region [110]. Goldberg assumed that no sensor information was available, and conditioned squeezing operations on the history of previous operations (which is equivalent to the control history) [74]. Donald and Jennings have defined *perceptual equivalence classes* that determine distinguishable scenarios for a mobile robot based on its sensing history and the projection from the configuration space onto the sensor space [50].

Termination condition The decision to halt the robot has been given careful attention in manipulation planning research, particularly in cases that involve configuration-sensing uncertainty. A motion plan might bring the robot into a goal region (*reachability*), but the robot may not halt if does not realize that it is in the goal region (*recognizability*) [62]. Under nondeterministic uncertainty, it is said that a plan *achieves* a goal if the robot is guaranteed to halt in the goal region.

A logical predicate called the *termination condition* is typically defined [110],

$$TC(t, \mathbf{q}^*[0,t], \mathbf{f}^*[0,t]) \in \{true, false\}, \tag{2.1}$$

in which $\mathbf{q}^*[0,t]$ and $\mathbf{f}^*[0,t]$ represent the full history of position and force sensing up to time t. A specialized version of (2.1) is usually implemented for practical considerations. Several specialized definitions of (2.1) are given under different assumptions about history in [62], [109].

Computing manipulation plans Mason has shown that preimage backchaining is bounded complete (i.e., if a solution with a bounded number of motions exists, the preimage backchaining method will find it), and that it suffices to consider directional preimages (i.e, preimages computed with respect to a single, fixed motion direction) as subgoals in the recursive backchaining process [131]. These results, however, do not imply that preimages are computable. In fact, Erdmann has proven by a reduction from the halting problem that, in arbitrary environments, preimages and backprojections are uncomputable [62]. That the recursively defined constraints in the proof do not generally occur in practice led Erdmann to conjecture without proof that in an environment with a known finite number of constraints, preimages should be computable. Canny has shown that this is indeed the case when the set of possible robot trajectories has a finite parameterization and when the set of feasible trajectories is a semi-algebraic subset of the parameter space [32]. Canny's approach is to cast the manipulation planning problem as a decision problem in the theory of the real numbers, and then to use quantifier elimination algorithms (see, e.g., [39]) to derive parametric semi-algebraic sets that are preimages.

Erdmann realized that conditions for recognizability and reachability could be decoupled [62]. This led to the development of the *backprojection*, which can be considered

as an approximation to the preimage that does not directly incorporate the termination condition. Backprojections from recognizable goal regions constitute valid preimages (although not necessarily maximal), and are considered in [47], [62], [110]. A directional backprojection is a backprojection computed for a single, fixed motion direction. The *nondirectional backprojection* [22], [48] is a subset of $\mathcal{C} \times S^1$ in which there is one directional backprojection for each orientation in S^1 (we use $\mathcal{C}$ to denote the configuration space). Donald has shown that the topology of the directional backprojection changes only at a finite set of critical velocity orientations in S^1 [47]. Therefore, the nondirectional backprojection can be represented by a finite set of directional backprojections; one for each critical orientation, and one for each noncritical interval.

In general, the manipulation planning problem under sensing and control uncertainty has been shown to have high complexity. Consider the problem of finding a sequence of motions that is guaranteed to move every point in a polyhedral initial region to a polyhedral goal region, amidst polyhedral obstacles, while permitting compliance. This problem, considered in traditional preimage planning, has been shown by Natarajan to be PSPACE-hard in $\Re^2$ [136]. Canny showed that this problem is nondeterministic exponential-time hard in $\Re^3$ [31].

Other related approaches Several other planning problems and approaches are related to the context developed in this chapter. In a series of papers by Donald [47], [48], [49], it was shown how a planner that is capable of recognizing failure (in addition to success) can be used to implement *error detection and recovery strategies*. Under this model, the robot is allowed to try a new plan after realizing that a failure has occurred, as opposed to continuing the failed plan. This represents an important use of sensor information, and expands the previous notion of reachability to include failure. Goldberg applied preimage planning ideas to construct manipulation plans that orient an

object using a parallel gripper without sensors [74], [75]. Fox and Hutchinson developed methods for computing backprojections that include visual constraint rays that result from the correspondence between edges in the workspace and the image plane [66]. Algorithms for computing motion plans in the presence of probabilistic uncertainty for mobile robots have been developed in [44], [89], [100]. In [188] a *sensory uncertainty field* was introduced that indicates positional sensing accuracy for a mobile robot as a function of configuration. The field is then used to determine a continuous path from the initial configuration to the goal configuration that minimizes the amount of expected sensing error or some combination of sensing error and path length. A sensor uncertainty field that includes differential sensor observations was presented in [142]. Object grasping has also been carefully studied in the context of manipulation planning under uncertainty (e.g., [24]).

2.2.2 Motivation

Consider two basic questions about the performance of a particular robot strategy: (1) How likely is the goal to be achieved? and (2) If the goal is achieved, how efficient is the solution? In manipulation planning work, usually only the first question is precisely addressed, although there is often some weak preference for more efficient plans (e.g., fewer backchaining steps). In traditional preimage backchaining work, people have been interested in generating strategies that answer the first question by guaranteeing that the goal will be achieved.[1]

When a probabilistic or stochastic formulation is considered, both of these questions should be carefully considered. Many stochastic models will lead to guaranteed goal achievement for any possible set of bounded-velocity motion commands, even though

[1] One notable exception is the consideration of sensorless backchaining in terms of geometric optimality on a Markov chain that describes grasped part orientations [74].

the probability that the goal will be achieved in some reasonable finite time interval is very small. For example, continuous Brownian motion will eventually lead the robot to any nonzero-measure goal region, which under perfect sensing indicates that Brownian motion achieves *probabilistic completeness* (a term used in [12]). This fact forms the basis for incorporating specific diffusion processes into robotic plans in [57]. For our context, however, the expected time to actually achieve the goal can be arbitrarily high. This indicates that for at least some problems, the probability that the goal will eventually be achieved is of little value for selecting a planning strategy.

In general, the same type of difficulty can be imagined under nondeterministic uncertainty. There might exist a large set robot strategies that are guaranteed to achieve the goal, but many solutions could be impractical if the execution time is too high. Therefore the efficiency of the solution (e.g., the amount of time the robot is expected to take to achieve the goal) is of great importance in evaluating a robot strategy under general models of uncertainty. This form of efficiency has also been useful in related robot control contexts (e.g., [21], [95], [176]). In this chapter, it is shown how objectives can be precisely defined that answer both of the questions above and guide the selection of a robot strategy.

The classical preimage has been a useful conceptual tool for developing manipulation planning algorithms. In its most common use, however, the preimage is defined in a limited manner. Usually a preimage requires that: (1) the robot executes a fixed motion command, and (2) elements in the preimage indicate from where the robot is *guaranteed* to achieve the goal. In our work, we replace fixed motion commands with a state-feedback or information-feedback controller for which trajectories can be evaluated with a precise objective. We also consider models that do not limit preimages to guaranteeing that the goal is achieved. A notable departure from this model was in the definition of probabilistic backprojections in [26], [27]. One motivation behind probabilistic backprojections, as

well as our approach, is that worst-case analysis tends to eliminate the consideration of many reasonable robot strategies. The absolute requirement that the goal is recognizably achieved can be too strong, particularly as the amount of uncertainty in control and sensing is increased. Furthermore, if bounded uncertainty models are replaced by smooth probability density functions, then it becomes impossible to *guarantee*[2] that the goal will be achieved in a fixed amount of time, except in restricted cases. For this reason, we consider the formulations under probabilistic uncertainty to represent important tools for analyzing manipulation planning problems.

The relationship between sensor and action history and decision making has long been considered important in planning under uncertainty (e.g. [62], [109], [124]). For our context, we want to optimize the performance of the robot, while directly taking into account the complications due to limited sensing. By using the concept of information state, as considered in stochastic control and dynamic game theory, we provide a useful characterization of this relationship. When there is perfect state information, the robot conditions its actions on its current state (or configuration); however, with imperfect state information, the robot actions are conditioned on information states. We will define a robot strategy to be a mapping (or sequence of mappings) on an information space, which *a priori* takes into account the various contingencies presented during execution. For this reason, no form of dynamic replanning is required. The information state concept is similar to the definition of knowledge states, considered in [58], and has also recently been proposed in [10].

The general structure of our formalism allows error models to be changed without altering the general computational approach. This can facilitate the iterative improvement of error models that are appropriate for a particular robotic system. For instance, if a better sensing model (in which the error is described in terms of a probability density

[2] "Guarantee" in a stochastic setting should technically be replaced by "achieve with probability one."

function) is determined for a given application, the appropriate density can be replaced, and much of the general approach remains the same. The formulation of objectives can also easily be changed.

We next discuss the type of solutions that can be obtained. Recall from Section 2.2.1 that the computational complexity of a basic manipulation planning problem is PSPACE-complete for $\Re^2$ [136] and nondeterministic exponential-time hard for $\Re^3$ [31], [32]. Solutions for various stochastic, Markov systems are also known to have difficult complexity [144], [145]. In optimal control theory, which has many commonalities with our approach, closed-form solutions can rarely be found, which allows efficient computation of the solution for a particular system. Bertsekas considers solvable stochastic control problems to typically be exceptions in applications [17]. For example, one of the most celebrated solutions in stochastic optimal control theory is for the *linear-quadratic Gaussian* (LQG) regulator problem [2], [6], [28]. For this case, however, the system (or motion model in our terms) is linear, the loss functional is quadratic, the uncertainties are characterized by Gaussian random variables, and the state space does not have geometric constraints. These represent severe restrictions on the set of problems that can be solved.

For manipulation planning problems that we define, there are several complications in comparison to a solved problem such as LQG: (1) the motion models are complex in comparison to a linear model, particularly when contact with obstacles occurs; (2) the performance depends directly on halting the robot (or system state) in some subset of the configuration space (or state space); and (3) we would like to use different densities (for probabilistic uncertainty) or bounded sets (for nondeterministic uncertainty) to model uncertainty without the burden of performing a distinct, detailed analysis for each case.

These complications offer limited hope for computation of closed form, exact solutions, which motivates us to consider numerical solutions as a practical alternative. Brost and

Christiansen have argued that numerical solutions may be the only practical approach to probabilistic backprojection computation [26]. Given the variety of problems that we wish to consider, a numerical computation method provides the greatest flexibility. We present algorithms that are based on the dynamic programming principle, and apply them to a general set of manipulation planning problems. The numerical solutions that we obtain can be improved at the expense of greater computation. The complexity of our method, however, is exponential in the dimension of the state space or information space, which we consider to be practical for a few dimensions.

2.3 General Definitions and Concepts

In this section, we define the general concepts and terminology that form the basis for our approach. We conceptualize the manipulation planning problem as a dynamic game that is played between two decision makers, the robot and nature, that influence the general state of the system. The robot has a general plan to achieve some goal, while nature makes some decisions that potentially interfere with the robot. Section 2.3.1 presents the basic concepts that characterize the motions and control of the robot. Section 2.3.2 introduces information space concepts, which are used for planning under configuration-sensing uncertainty. Section 2.3.3 describes alternative representations of the information states. Section 2.3.4 defines a robot strategy, the incorporation of a termination condition into a strategy, and the evaluation of a strategy with a loss functional. Section 2.3.5 presents a specific model that is based on previous manipulation planning research, which will be used for the examples in this chapter.

2.3.1 States, stages, and actions

State space The position of a robot, $\mathcal{A}$, in a workspace is represented by a point in an n-dimensional *configuration space*, $\mathcal{C}$, for which n is the number of degrees of freedom of $\mathcal{A}$. For manipulation planning, a subset of $\mathcal{C}$, denoted as $\mathcal{C}_{valid}$, is usually defined (see Section 1.2 and [109] for configuration-space details). This corresponds to points in $\mathcal{C}$ at which either: (1) $\mathcal{A}$ does not touch an obstacle, or (2) the boundary of $\mathcal{A}$ is in contact with the boundary of some obstacle, but the interiors do not overlap. The second condition enables the possibility of guarded motion and compliance [195], which, for example, allows the robot to execute a motion along the tangent of an obstacle boundary. For the examples in this chapter, we define the *state space* as $X = \mathcal{C}_{valid}$ (for collision-free planning, we would use $X = \mathcal{C}_{free}$, which is the interior of $\mathcal{C}_{valid}$).

Stages Consider a discretized representation of time as *stages*, with an index $k \in \{1, 2, \ldots, K\}$. Stage k refers to time $(k-1)\Delta t$. The state at stage k is denoted by x_k. We generally take Δt sufficiently small to approximate continuous paths. The final stage, K, is only defined to ease technical considerations as the system evolves toward infinite time. Because we expect the robot to achieve the goal in some finite time (if it is achievable), consideration of infinite-length trajectories is not necessary. The specification of K is not required by our algorithms due to stationarity, which will be discussed in Section 2.3.4. The formalism could also be defined in sufficient generality without discretizing time, and consequently defining controlled diffusions [104]; the definitions that would follow require the use of stochastic differential equations and measure-theoretic concepts. For our context, a discretized representation of time facilitates the development of the numerical computation approach. Furthermore, actual robot systems will be limited to discrete-time sampling for acquiring sensor information and executing motion commands. Hence, the discrete-time representation seems most appropriate.

Actions An *action* (or command), which is denoted by u_k, can be issued to $\mathcal{A}$ at each stage k. Let U denote the *action space* for $\mathcal{A}$, requiring that $u_k \in U$. Nature also chooses actions. Let θ_k denote an action for nature, which is chosen from a set Θ. Consider θ_k to be a vector quantity that is divided into two subvectors, θ_k^a and θ_k^s (i.e., $\theta_k = [\theta_k^a \;\; \theta_k^s]$). As will be seen shortly, θ_k^a affects the outcome of $\mathcal{A}$'s actions, and θ_k^s affects the sensor observations of $\mathcal{A}$.

State transition equation To describe the effect of a robot action with respect to state, we define a *state transition equation* as

$$x_{k+1} = f(x_k, u_k, \theta_k^a). \tag{2.2}$$

Hence, given a robot action, nature's action, and the current state, the next state is deterministically specified. During execution, however, $\mathcal{A}$ will not know the action of nature. A specific example of a state transition equation is given in (2.15) of Section 2.3.5.

When considering nondeterministic uncertainty, we use the state transition equation to obtain the following subset of X:

$$F_{k+1}(x_k, u_k) = \{f(x_k, u_k, \theta_k^a) \in X | \theta_k^a \in \Theta^a\}. \tag{2.3}$$

This set represents the possible next states that can result from a single application of the state transition equation.

Under probabilistic uncertainty, we assume that the probability density function (pdf), $p(\theta_k^a)$, is known. For this probability density and the remaining probability densities in the chapter, we implicitly assume there is some underlying probability space, and random variables with densities are constructed using appropriate measurability conditions (see [197] for a treatment of these technical concerns). By using the state transition equation, we can obtain a pdf for x_{k+1}, which is represented by $p(x_{k+1}|x_k, u_k)$.

In general, we will use the notation, F, to refer to minimal subsets of X that can be inferred from the arguments. The role of F in our expressions for nondeterministic uncertainty can be considered analogous to the role of p in probabilistic expressions. Thus, F is a generic representation for a subset of X, while p is a generic representation for a density on X.

2.3.2 Imperfect sensing and information spaces

In this section, we consider uncertainty in sensing, which implies that the current state is not known by the robot, and actions must be chosen on the basis of imperfect information. Therefore, actions taken by the robot will be conditioned on an information space, as opposed to the state space. This information space concept has been adapted from stochastic control [105] and dynamic game theory [6] to fit our particular context.

Sensor observations We begin by defining a general model of robot sensing. A sensor can be viewed as a mapping from states onto sensor values with potential interference that is caused by nature. At every stage k, the robot makes an *observation* that is governed by the equation,

$$y_k = h_k(x_k, \theta_k^s), \tag{2.4}$$

which we term the *observation equation*. A specific example of an observation equation is given in Equations (2.16) and (2.17) of Section 2.3.5.

The values, y_k, belong to a *sensor space*, denoted by Y. This model indicates that the robot receives information at every possible stage; however, this assumption can be relaxed. For example, in visual servo control applications, the servo rate for the robot joint controllers is typically much faster than the sampling rate of the vision system (see, e.g., [129]). It might also be the case that a sensor only provides information at randomly

chosen stages (as is the case for the visual servo system reported in [65], in which the vision system's sampling rate varies according to the amount of processing required to track moving objects in the scene). Such variations are straightforward to consider.

Suppose that we are considering nondeterministic uncertainty. The set of possible values for x_k after only observing y_k can be determined from the observation equation as:

$$F_k(y_k) = \{x_k \in X | y_k = h(x_k, \theta_k^s), \theta_k^s \in \Theta^s\}. \tag{2.5}$$

Under probabilistic uncertainty, we assume that the pdf, $p(\theta_k^s)$, is known. By using the observation equation, we can obtain a pdf for x_k, which is represented by $p(x_k|y_k)$. As a simple example, h could represent a position sensor that measures x_k with Gaussian noise. If $h(x_k, \theta_k^s) = x_k + \theta_k^s$, and $p(\theta_k^s)$ is a Gaussian density, then $p(x_k|y_k)$ is Gaussian. If $Y = X$ and h_k is reduced to the identity map from X to Y, then the sensing model reduces to perfect state information.

Equation (2.4) represents the output equation used in control theory, and is also similar to the projection of world states onto sensor values, as used in previous robotics contexts (e.g., [50]). Also, such transformations have been studied extensively in statistical image modeling [70], [115], [178] and in sensor error modeling [93].

History and information As discussed in Section 2.2.2, the relationship between sensing and action has long been considered important in motion planning under uncertainty. The following definitions precisely describe the sensing and action history that $\mathcal{A}$ has available. For a given stage k, let η_k denote some subset:

$$\eta_k \subseteq \{u_1, u_2, \ldots, u_{k-1}, y_1, y_2, \ldots, y_k\}. \tag{2.6}$$

The value η_k is a set of past actions and observations that are known to $\mathcal{A}$ at stage k, and is termed the *information state* of $\mathcal{A}$. For instance, we could consider a memoryless robot,

in which $\eta_k = y_k$. As another example, we could have a sensorless robot as considered in [74], in which $\eta_k = \{u_1, \ldots, u_{k-1}\}$. We could also consider the stage index k as part of the information space for the purpose of developing robot strategies that involve timing; however, we will not explicitly consider k as part of η_k in this chapter.

The set of values that η_k can assume is denoted by N_k, and is termed the *information space*. We define an *information structure* as the set of N_k for all $1 \leq k \leq K$. As it is presently defined, the dimension of the information space grows linearly with the number of stages, which appears impractical. It turns out that alternative representations of the information space can be determined; this is the subject of Section 2.3.3.

2.3.3 Representations of the information state

In this section, we present alternative ways to interpret the information space. Under nondeterministic uncertainty, this interpretation will be a subset of the state space. Under probabilistic uncertainty, this interpretation will be a pdf on the state space.

Consider the case in which the robot has perfect memory. Each η_k then corresponds to complete history of previous robot actions and observations. If U is n_1-dimensional and Y is n_2-dimensional, then, in general, the dimension of N_k will be $[k(n_1 + n_2) + n_2]$-dimensional. A space that grows significantly with each stage (and becomes infinite-dimensional when $K = \infty$) is very unappealing for designing strategies. The information space representations presented in this section do not grow with the number of stages and provide more intuitively satisfying characterizations of the information state.

The case of nondeterministic uncertainty An alternative to maintaining a growing history is to consider subsets of X that represent the possible current states, x_k, for a given information space value, η_k. We will represent the minimal subset of X that can be inferred from η_k as $F_k(\eta_k)$. In other words, $F_k(\eta_k)$ represents the set of all states, x_k, that

could possibly be the true system state, given the history η_k. By using this approach, we will show that the information state subset $F_{k+1}(\eta_{k+1})$ can be determined from $F_k(\eta_k)$, when u_k and y_{k+1} are given.

For practical purposes, $F_k(\eta_k)$ can be considered as an alternative representation of the information state. This representation is intuitively satisfying because we can think of the uncertainty as a set that represents possible locations in the state space X. In traditional preimage planning research, uncertainty due to imperfect sensing has often been viewed in this way.

We briefly indicate how an information state subset is obtained at a given stage. Initially, we have $F_1(\eta_1)$. An expression for $F_{k+1}(\eta_{k+1})$ can be derived in terms of $F_k(\eta_k)$, u_k, and y_{k+1}. Suppose inductively that we have $F_k(\eta_k)$. First consider the effect on the state space of using the action, u_k. Recall from Equation (2.3) that $F_{k+1}(x_k, u_k)$ represents the possible values x_{k+1} that could be obtained through a single application of the state transition equation. We can define

$$F_{k+1}(\eta_k, u_k) = \bigcup_{x_k \in F_k(\eta_k)} F_{k+1}(x_k, u_k). \tag{2.7}$$

Note that η_{k+1} can be specified with η_k, u_k, and y_{k+1}. Recall from Equations (2.5) that $F_k(y_k)$ denotes the set of possible values for x_k after only observing y_k. By maintaining consistency with the observation of y_{k+1}, the following can be obtained:

$$F_{k+1}(\eta_{k+1}) = F_{k+1}(y_{k+1}) \bigcap F_{k+1}(\eta_k, u_k), \tag{2.8}$$

which depends on (2.3) and (2.5).

If $\mathcal{A}$ does not have perfect memory, then the condition $\{\eta_k, u_k\}$ is replaced in (2.8) by the appropriate subset of history.

The case of probabilistic uncertainty Under probabilistic uncertainty, the information state can be considered as a conditional density on the state space, denoted

as $p(x_k|\eta_k)$. This approach can be used to determine the information state density $p(x_{k+1}|\eta_{k+1})$ from $p(x_k|\eta_k)$, when u_k and y_{k+1} are given. This observation allows the development of several well-known stochastic control results, such as the Kalman filter, when all densities in the information space take some parametric form of fixed, low dimension [105]. This representation is intuitively satisfying, because we can think of $\mathcal{A}$'s uncertainty model as a density representing possible locations in the state space X.

We briefly indicate how the information state density is obtained. These equations can be considered as probabilistic versions of the nondeterministic results. Initially, we have $p(x_1|\eta_1)$. We can derive an expression for $p(x_{k+1}|\eta_{k+1})$ in terms of $p(x_k|\eta_k)$, u_k, and y_{k+1}. Suppose inductively that we have $p(x_k|\eta_k)$. First consider the effect on the state space of using the action, u_k. Using the density representation of the state transition equation (from Section 2.3.1) we obtain, through marginalization with respect to X_k,

$$p(x_{k+1}|\eta_k, u_k) = \int p(x_{k+1}|x_k, u_k)p(x_k|\eta_k)dx_k. \tag{2.9}$$

Recall from Section 2.3.1 that $p(x_{k+1}|x_k, u_k)$ is inferred from the state transition equation. Note that η_{k+1} can be specified with η_k, u_k, and y_{k+1}. By using Bayes' rule on X_{k+1} and Y_{k+1}, the following can be obtained:

$$p(x_{k+1}|\eta_{k+1}) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k, u_k)}{\int p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k, u_k)dx_{k+1}}, \tag{2.10}$$

which is a function of $p(y_{k+1}|x_{k+1})$ as defined in Section 2.3.2. A more detailed discussion of (2.10) can be found in [105]. If $\mathcal{A}$ does not have perfect memory, then the condition $\{\eta_k, u_k\}$ is replaced in (2.10) by the appropriate subset of history.

2.3.4 Strategy concepts

Motion strategies At first it might seem appropriate to define some action u_k for each stage. In general, due to the control uncertainty, it is not possible to predict the trajectory of the robot for given motion commands. It is, therefore, advantageous to allow the robot to respond to information that becomes available during execution.

We consider robot strategies for two cases: perfect information and imperfect information. Figure 2.4 shows block-diagram representations of the concepts that will be described in this section. Suppose that the robot has perfect state information. We can implement a *state-feedback strategy at stage k* as a function $g_k : X \rightarrow U$. For each state, x_k, a strategy yields an action $u_k = g_k(x_k)$. The set of mappings $\{g_1, g_2, \ldots, g_K\}$ is denoted by g and termed a (robot) *strategy* of $\mathcal{A}$.

If the robot does not have direct access to state information, its actions are instead conditioned on the information state. In this case, we define a *strategy at stage k* of $\mathcal{A}$ as a function $g_k : N_k \rightarrow U$. For each information state, η_k, a strategy yields an action $u_k = g_k(\eta_k)$. In a sense, the "planning" actually occurs in this information space. These strategy concepts are equivalent to a feedback control law [17], [105], and are similar to a conditional multi-step plan in manipulation planning [110].

We also define a strategy, γ^θ, for nature. Nature is considered as a decision maker that can interfere with the robot; therefore, we allow nature's actions to depend in general on the state, x_k, and the action of the robot, u_k. We can define a *pure* or *deterministic* strategy for nature as a mapping at each stage as $\gamma_k^\theta : X \times U \rightarrow \Theta$. Under nondeterministic uncertainty, we will assume that nature implements a deterministic strategy that is unknown to the robot. Let Γ^Θ refer to the space of strategies that are available to nature under nondeterministic uncertainty.

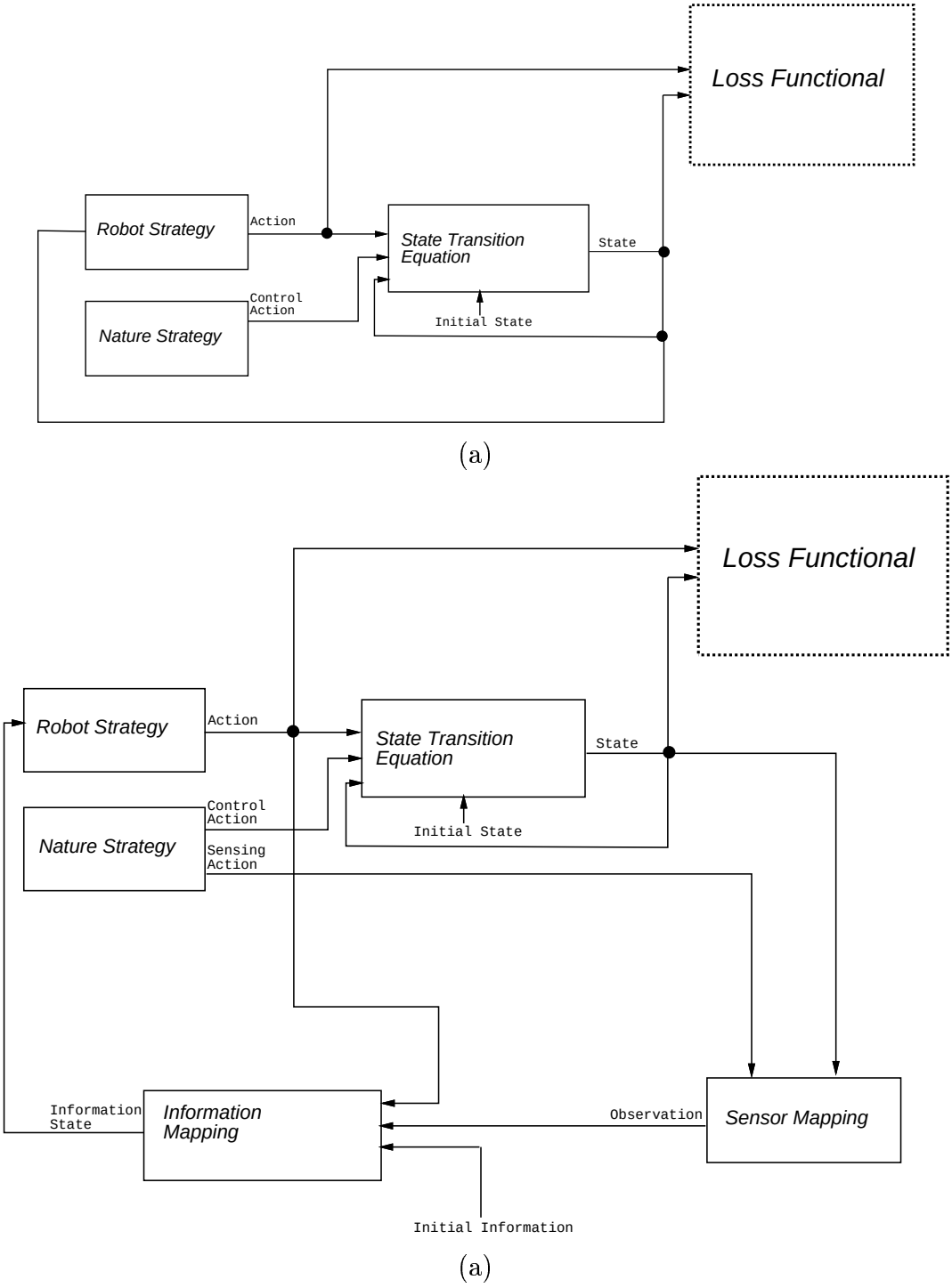


Figure 2.4 A dynamic game against nature with: (a) perfect information and (b) imperfect information.

Under probabilistic uncertainty, we consider a *randomized* or *mixed* strategy for nature, in which the action of nature is represented by a pdf, $p(\theta_k)$ (or we can more generally consider $p(\theta_k|x_k, u_k)$). The specific action of nature at stage k is denoted by θ_k, sampled from the random variable Θ_k. Therefore, the robot is given a pdf, $p(\theta_k)$, that characterizes the action taken by nature at stage k. Although the randomized *strategy* is known by the robot, the *actions* that will be chosen are sampled from a random variable at each stage, and are thus unknown to the robot.

Termination conditions The notion of a termination condition has been quite useful for formulating robot plans that tell the robot when to halt, based on its current, partial information [62], [110], [124]. The same concept is needed in our context; hence, we define a *termination condition* TC_k at each stage by a binary-valued mapping,

$$TC_k : N_k \rightarrow \{true, false\}. \tag{2.11}$$

The termination condition can be considered as a special type of action that can cause the robot to halt. With perfect state information, N_k is simply replaced by X in Equation (2.11). We require that if $TC_k = true$, then $TC_{k+1} = true$. This implies that once the termination condition has been applied, it cannot be retracted (i.e., the robot terminates its motions).

Let TC denote the complete specification of TC_k for all k. The termination condition is implemented so that the robot terminates at some stage $k \leq K+1$, making the specific choice of K not important, except that it is sufficiently large. We will use the notation γ to denote the pair (g, TC), which can be considered as a *strategy with termination condition*. This pairing is similar to the concept of a *motion command* as defined in [110]. We will use the notation Γ to denote the set of all γ that are available to the robot. It can also been seen that the use of this termination condition in the determination of an

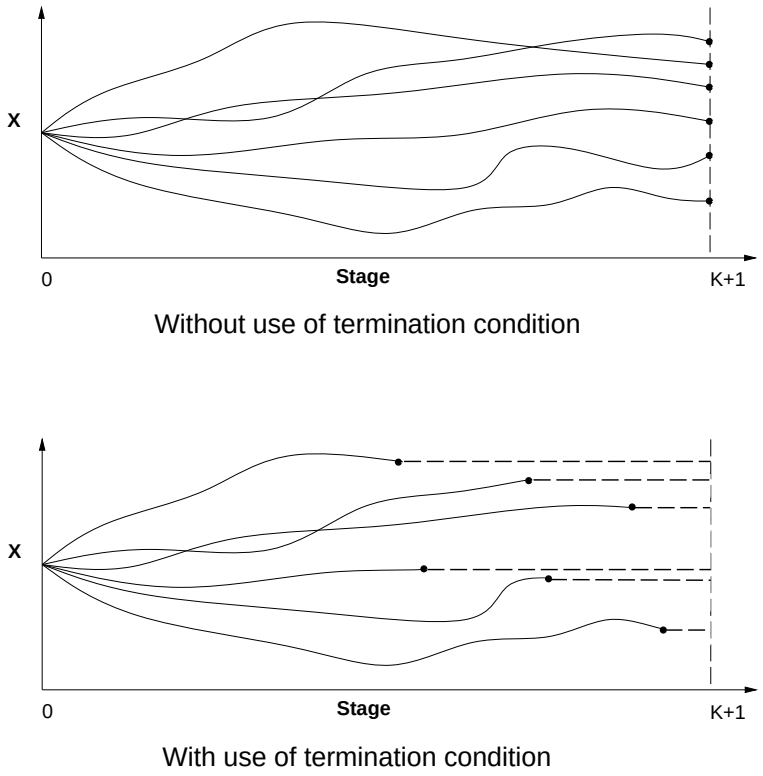


Figure 2.5 The termination condition forces the robot to halt at a given state. The halting stage is *a priori* uncertain for a given initial state and strategy.

optimal strategy is equivalent to defining an *optimal stopping rule*, as done in optimal control theory [105].

When perfect information is available, the most appropriate choice for TC_k is $TC_k =$ *true* if and only if x_k lies in the goal region. The termination condition becomes more interesting under imperfect information. Because the state x_k cannot necessarily be predicted for a given initial state, x_1 and strategy, γ, the particular stage at which the termination condition will be applied is uncertain. However, because the robot remains motionless after the termination condition becomes true, we can consider the resulting states at stage $K + 1$. Figure 2.5 indicates the effect of the termination condition.

Loss functionals We encode the objectives that are to be achieved by a nonnegative real-valued functional $L(x_1, ..., x_{K+1}, u_1, ..., u_K, TC)$, called the *loss functional*. Note that

the loss functional is not written as a function of γ, but in terms of the *actual* executed trajectory and action histories. The ultimate goal of the planner is to determine a strategy g and termination condition TC that causes L to be optimized in an appropriate sense. As will be discussed in Section 2.6, with nondeterministic uncertainty a strategy is selected that optimizes worst-case performance, and with probabilistic uncertainty a strategy is selected that optimizes expected-case performance.

We assume that a loss functional is of the following additive form, which, except for the termination condition, is often used in optimal control theory [105] and is similar to (1.7):

$$L(x_1, ..., x_{K+1}, u_1, ..., u_K, TC) = \sum_{k=1}^{K} l_k(x_k, u_k, TC_k) + l_{K+1}(x_{K+1}). \tag{2.12}$$

The first K terms correspond to costs that are received at each step during execution of the strategy. The final term, l_{K+1} is a cost that can be used to indicate the importance of terminating in the goal. This form is quite general and facilitates the application of the dynamic programming principle, as discussed in Section 2.6.

We next present two useful loss functionals that we have considered. Let $G \subset X$ represent a goal region in the state space. The following loss functional distinguishes only between success and failure to achieve the goal:

$$L(x_1, ...x_{K+1}, u_1, ..., u_K, TC) = \begin{cases} 0 & \text{if } x_{K+1} \in G \\ 1 & \text{otherwise} \end{cases} . \tag{2.13}$$

Under probabilistic uncertainty, this loss functional will yield the probability of success for a given strategy (in the same manner that a 0-1 loss results in the probability of an incorrect decision in Bayesian decision theory [46]).

Often we will want to consider the cumulative cost of executing motions. Under the bounded velocity assumption, the following loss functional can measure the length of the

executed trajectory:

$$L(x_1, ...x_{K+1}, u_1, ..., u_K, TC) = \begin{cases} \sum_{k=1}^{K+1} l(u_k, TC_k) & \text{if } x_{K+1} \in G \\ \\ C_f & \text{otherwise} \end{cases} . \quad (2.14)$$

Above, $l(u_k, TC_k)$ denotes the cost associated with taking action u_k, and we require that $l(u_k, TC_k) = 0$ if $TC_k = \mathit{true}$. Hence, loss does not accumulate after the robot has terminated. We use C_f to express how important it is to achieve the goal. As C_f becomes less than a typical aggregate action cost that achieves the goal, then strategies will be preferred that do not necessarily expect to achieve the goal.

Stationarity For the problems that we consider, the optimal action u_k, and termination condition, TC_k, do not depend on the stage index, k. This is because the optimal strategies for the problems in this context are *stationary*, a concept that is defined and discussed in [6], [105]. Stationarity implies that such components as the state transition equation, free configuration space, or loss functional are not stage-dependent. For example, it might be the case that the workspace contains a known, moving obstacle. Many quantities would then depend on the stage index, resulting in a robot strategy that was time-dependent. This is analogous to the configuration-time space, $\mathcal{CT}$, as considered in [109] for planning amidst moving obstacles. In general, our approach supports the analysis of time-dependent problems; however, we preclude them from consideration in this chapter.

2.3.5 Specific model details

In this section, we present specific definitions of a state transition equation and an observation equation. These models are inspired by those used in previous manipulation

planning research and are used in examples throughout this chapter. In general, a variety of other types of models could be defined.

The control model Suppose the robot $\mathcal{A}$ is a polygon translating in the plane amidst polygonal obstacles. The action set of $\mathcal{A}$ is a set of commanded velocity directions, which can be specified by an orientation, yielding $U = [0, 2\pi)$. The robot will attempt to move a fixed distance $\|v\|\Delta t$ (expressed in terms of a constant velocity modulus, $\|v\|$) in the direction specified by u_k. The action space of nature is a set of angular displacements θ_k^a, such that $-\epsilon_\theta \leq \theta_k^a \leq \epsilon_\theta$, for some maximum angle ϵ_θ. Under nondeterministic uncertainty, any action $\theta_k^a \in [-\epsilon_\theta, \epsilon_\theta]$ can be chosen by nature. When using probabilistic uncertainty, $p(\theta_k^a)$ could be a continuous pdf, which is zero outside of $[-\epsilon_\theta, \epsilon_\theta]$.

There are several cases to consider in defining the state transition equation, f. First consider the state transition equation when $x_k \in \mathcal{C}_{free}$, at a distance of at least $\|v\|\Delta t$ away from the obstacles. If $\mathcal{A}$ chooses action u_k from state x_k, and nature chooses θ_k^a, then x_{k+1} is given by

$$f(x_k, u_k, \theta_k^a) = x_k + \|v\|\Delta t \begin{bmatrix} cos(u_k + \theta_k^a) \\ sin(u_k + \theta_k^a) \end{bmatrix}. \tag{2.15}$$

Let $\mathcal{C}_{contact}$ represent the boundary of $\mathcal{C}_{free}$ (hence, $\mathcal{C}_{contact} = \mathcal{C}_{valid} \backslash \mathcal{C}_{free}$). If $x_k \in \mathcal{C}_{contact}$, with a distance of at least $\|v\|\Delta t$ from the edge endpoints, then a compliant motion is generated by using the generalized damper model (see e.g., [195]) for certain choices of u_k. If u_k points into the obstacle edge with a sufficient angle to overcome friction, then the robot moves a fixed distance parallel to the edge. Otherwise, the robot either remains fixed, or moves away into $\mathcal{C}_{free}$. The remaining cases describe when the robot moves from $\mathcal{C}_{free}$ to $\mathcal{C}_{contact}$, from $\mathcal{C}_{contact}$ to $\mathcal{C}_{free}$, or from one edge in $\mathcal{C}_{valid}$ to another. These cases are straightforward to define with the generalized damper model, as discussed in Section 2.2.1.

This model of uncertainty does not correspond completely to that used in [26], [110], [124]. In our model, nature repeatedly acts at each time Δt. To correspond more closely with the traditional model, no additional uncertainty would be introduced if $u_k = u_{k+1}$. Under the control model that we have defined, this would mean $\theta_{k+1} = \theta_k$ if $u_k = u_{k+1}$. This model can be implemented by including the previous u_k as a component of the state space.

The sensing model We now present a sensing model that is similar to that used in [26], [62], [110]. This sensing model will be used in Section 2.6.6. The robot $\mathcal{A}$ is equipped with a position sensor and a force sensor. Assume that the position sensor is calibrated in the configuration space, yielding values in $\Re^2$. The force sensor provides values in $[0, 2\pi) \cup \{\emptyset\}$, indicating either the direction of force or no force (represented by $\emptyset$).

We consider independent portions of the observation equation: h^p for the position sensor and h^f for the force sensor (which together form a three-dimensional vector-valued function). We partition the sensing action of nature, θ_k^s into subvectors $\theta_k^{s,p}$ and $\theta_k^{s,f}$, which act on the position sensor and force sensor, respectively. The observation for the position sensor is $y_k^p = h^p(x_k, \theta_k^{s,p}) = x_k + \theta_k^{s,p}$. Under nondeterministic uncertainty, $\theta_k^{s,p}$ could be any value in $\Theta_k^{s,p}$. If probabilistic uncertainty is used, a density is presented, such as

$$p(\theta_k^{s,p}) = \begin{cases} \frac{2}{\pi \epsilon_p^2} & \text{for } \|\theta_k^{s,p}\| < \epsilon_p \\ 0 & \text{otherwise} \end{cases}, \tag{2.16}$$

which is used for some examples in Section 2.6.6. In (2.16) a radius ϵ_p is specified, and $\theta_k^{s,p}$ is two-dimensional.

For the force sensor we obtain either: (1) a value in $[0, 2\pi)$, governed by $y_k^f = h^f(x_k, \theta_k^{s,f}) = \alpha(x_k) + \theta_k^{s,f}$, in which $x_k \in \mathcal{C}_{contact}$, and the true normal is given by $\alpha(x_k)$, or (2) an empty value, $\emptyset$, when the robot is in $\mathcal{C}_{free}$. When the robot configuration lies

in $\mathcal{C}_{contact}$ and probabilistic uncertainty is in use, then the density can be represented as

$$p(\theta_k^{s,f}) = \begin{cases} \frac{1}{2\epsilon_f} & \text{for } |\theta_k^{s,f}| < \epsilon_f \\ 0 & \text{otherwise} \end{cases}, \tag{2.17}$$

for some positive prespecified constant $\epsilon_f < \frac{1}{2}\pi$. We consider the random variables of $\theta_k^{s,p}$ and $\theta_k^{s,f}$ to be independent and identically distributed over all stages.

2.4 Forward Projections

In this section, we present forward projections for each of the four uncertainty cases that are considered in this chapter. A forward projection is used to characterize the possible future states, under the implementation of a strategy, from an initial state. The forward projection concepts presented here are based on forward projection concepts that have appeared in manipulation planning research (e.g., [26], [62]). In our work, the forward projections result from the implementation of a strategy, γ. We conclude this section by presenting some computed examples of forward projections.

2.4.1 Nondeterministic forward projections

Recall from Section 2.3.4 that under nondeterministic uncertainty, the strategy of nature γ^θ is deterministic, but unknown to the robot. Under perfect sensing, γ^θ defines a specific action $\theta_k \in \Theta$ that will be taken by nature at every stage, k. The resulting nondeterministic forward projection will include all of the system states that could result from the various actions of nature. In this way, it yields a set of possible futures under the implementation of a strategy.

2.4.1.1 The perfect information case

We use the notation $F_j(x_i, g)$ to denote the minimal subset of X that is guaranteed to contain x_j, if the system begins in state x_i at stage i and strategy g is implemented up to stage j.

Assume that some g is given, and that at stage k, the state x_k is known. The action taken by the robot at stage k is known to be $u_k = g_k(x_k)$. Therefore, we can write

$$F_{k+1}(x_k, g) = F_{k+1}(x_k, g_k(x_k)) = F_{k+1}(x_k, u_k), \tag{2.18}$$

in which $F_{k+1}(x_k, u_k)$ is given by Equation (2.3). Although the action is known, the resulting next state x_{k+1} is nondeterministic because of nature, $\theta_k^a \in \Theta^a$.

Suppose that we wish to determine the outcome at stage x_{k+2}, if we know x_k. From (2.3), we already know that $x_{k+1} \in F_{k+1}(x_k, u_k)$. The nondeterministic action of nature at stage $k+1$ must next be taken into account to yield

$$F_{k+2}(x_k, g) = \{ f(x_{k+1}, u_{k+1}, \theta_{k+1}^a) \in X \,|\, x_{k+1} \in F_{k+1}(x_k, g), \;\; \theta_{k+1}^a \in \Theta^a \}. \tag{2.19}$$

This forward projection can also be expressed with a set union as

$$F_{k+2}(x_k, g) = \bigcup_{x_{k+1} \in F_{k+1}(x_k, g)} F_{k+2}(x_{k+1}, g). \tag{2.20}$$

One interpretation for this representation is that from each possible state in the single-stage forward projection from stage k to stage $k+1$, the single-stage forward projection from stage $k+1$ to stage $k+2$ is possible. The resulting subset of X represents the union of all of these single-stage forward projections (see Figure 2.6).

The forward projection for a finite number of stages from stage 1 can be considered as an iterated union,

$$F_k(x_1, g) = \bigcup_{x_2 \in F_2(x_1, g)} \; \bigcup_{x_3 \in F_3(x_2, g)} \cdots \bigcup_{x_{k-1} \in F_{k-1}(x_{k-2}, g)} F_k(x_{k-1}, g), \tag{2.21}$$

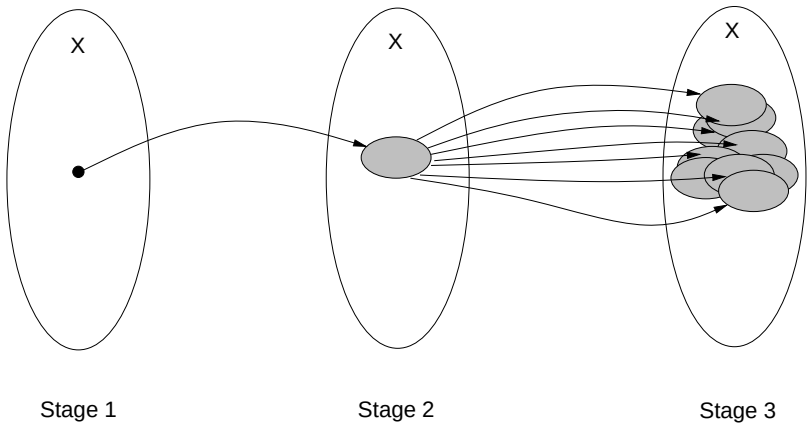


Figure 2.6 A depiction of a two-stage forward projection under nondeterministic control uncertainty.

which is an extension of (2.20). The projection from any stage k to stage $k+N$ can be similarly defined.

The next step is to include the termination condition to determine a forward projection for γ, as opposed to g. Recall that the robot remains motionless after TC becomes *true*. Hence, the effect of the termination condition is equivalent to considering the resulting location of the robot at stage $K+1$ (assuming that under all possible trajectories, the termination condition was met before $K+1$). This results in $F_{K+1}(x_1, \gamma)$, which can be constructed by replacing g with γ in Equations (2.18) to (2.21).

The classical reachability and recognizability concepts [62] can be defined using our formalism. We say that the goal is *reachable at stage* k under γ if $F_k(x_1, g) \subseteq G$. In other words, if the strategy is guaranteed to bring the robot into the goal region for some k, then reachability at stage k holds. We can also define a reachability that does not depend on k. We can say that the goal is *reachable* if, for every possible state trajectory $\{x_1, .., x_{K+1}\}$ (under the implementation of a given g), there exists a k such that $x_k \in G$.

A stronger condition is that the goal is *recognizably achieved* under γ, which means that $F_{K+1}(x_1, \gamma) \subseteq G$. This condition implies that the robot is guaranteeed to terminate in the goal region.

2.4.1.2 The imperfect information case

We consider, as in the perfect information case, a deterministic strategy for nature, γ^θ, which is unknown to the robot. We will define the forward projection in a manner similar to the perfect information case.

The previous forward projection, Equation (2.21), provided a subset of X in which the system state will lie after the execution of a strategy. With imperfect sensing, we can consider the motions to occur in the information space. In fact, we can consider the information space as a new "state space" in which there is perfect "state" information. For this reason, a forward projection can also be defined directly on the information space.

It is assumed for the forward projection that the history has not yet been given. Suppose that an information state, $\eta_k \in N_k$, is given. Under the implementation of g, the action $u_k = g_k(\eta_k)$ is known.

We now define the *information forward projection* for a single stage. This will yield a future subset of the information space, which is an intermediate concept that is used to define the forward projection as a subset of the state space. We have previously used F to represent a subset of X, and we will use $\tilde{F}$ to refer to a subset of the information space. We obtain $\tilde{F}_{k+1}(\eta_k, g(\eta_k)) = \tilde{F}_{k+1}(\eta_k, u_k)$. The forward projection, $\tilde{F}_{k+1}(\eta_k, u_k)$, is defined as the set of all $\eta_{k+1} \in N_{k+1}$ such that if $y_{k+1} \in \eta_{k+1}$, then $y_{k+1} = h_{k+1}(x_{k+1}, \theta^s_{k+1})$ for some $x_{k+1} \in F_k(x_k, u_k)$, $\theta^s_{k+1} \in \Theta^s$, and $x_k \in F_k(\eta_k)$. This forward projection depends on Equation (2.3), and $F_k(\eta_k) \subseteq X$ is the subset representation of the information state from Section 2.3.2. To obtain the information forward projection from stage 1 to some stage k, we can iteratively apply the same steps.

The information forward projection can be mapped to subsets of the state space. For a given $\tilde{F}_k(\eta_1, g)$, the subset of X in which the system state will lie is

$$\bigcup_{\eta_k \in \tilde{F}_k(\eta_1, g)} F_k(\eta_k). \tag{2.22}$$

The goal is *reachable at stage* k if the set defined in (2.22) is a subset of G. As in Section 2.4.1.1, we can replace g with γ in the above expressions to yield the forward projection with termination condition, $F_{K+1}(\eta_1, \gamma)$. Hence, recognizability can also be defined.

2.4.2 Probabilistic forward projections

Under nondeterministic uncertainty, the forward projections yield subsets of the state space; however, for probabilistic uncertainty, the forward projections will be specified by pdf's on the state space. We use the notation $p(x_{k'}|x_k, g)$ in this section to represent the density obtained at stage k' if the system begins at state x_k at stage k and strategy g is implemented. This density follows directly from the state transition equation, and the densities for nature of the form, $p(\theta_k^a)$.

2.4.2.1 The perfect information case

The following development parallels the development of the forward projection in Section 2.4.1.1. Assume that some g is given, and that at stage k, the state x_k is known. The action taken by the robot at stage k is known to be $u_k = g_k(x_k)$. Therefore, we can write

$$p(x_{k+1}|x_k, g) = p(x_{k+1}|x_k, g_k(x_k)) = p(x_{k+1}|x_k, u_k). \tag{2.23}$$

Recall from Section 2.3.1 that $p(x_{k+1}|x_k, u_k)$ can be determined from the state transition equation.

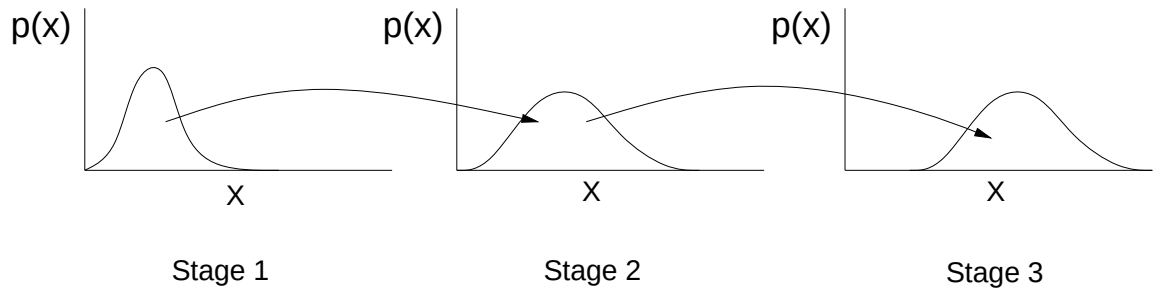


Figure 2.7 A depiction of a two-stage forward projection under probabilistic control uncertainty.

Next consider predicting the outcome at stage $k+2$, if we begin at stage k and apply g:

$$p(x_{k+2}|x_k, g) = \int p(x_{k+2}|x_{k+1}, g_{k+1}(x_{k+1}))p(x_{k+1}|x_k, g_k(x_k))dx_{k+1}. \tag{2.24}$$

The result after applying two actions is a posterior density on X. Figure 2.7 depicts the forward projection; this can be contrasted to Figure 2.6, which shows the forward projection under nondeterministic uncertainty.

The forward projection for a finite number of stages from stage 1 results in the posterior:

$p(x_k|x_1, g) =$

$$\int p(x_k|x_{k-1}, g_{k-1}(x_{k-1}))p(x_{k-1}|x_{k-2}, g_{k-2}(x_{k-2})) \cdots p(x_2|x_1, g_1(x_1))dx_2 dx_3 \cdots dx_{k-1}. \tag{2.25}$$

The projection from any stage k to stage $k+N$ can be similarly defined.

The next step is to include the termination condition to determine a forward projection for γ, as opposed to g. We can replace g with γ in the conditions above, and define $p(x_{K+1}|x_1, \gamma)$ by using the assumption that the robot remains motionless after the termination condition becomes *true*.

We can now define probabilistic notions of reachability and recognizability. The probability that the goal is reached at stage k is given by

$$\int_G p(x_k|x_1, g)dx_k, \tag{2.26}$$

in which the region of integration is the goal region, $G \subseteq X$.

The probability that the goal is recognizably achieved is

$$\int_G p(x_{K+1}|x_1, \gamma)dx_{K+1}. \tag{2.27}$$

2.4.2.2 The imperfect information case

In this section, we develop the forward projections for the case in which there is probabilistic uncertainty in both sensing and control. The forward projection for this case will be considered as a density on X, which is conditioned on a particular strategy and initial state (either x_1 or η_1). This density indicates where the robot will be likely to end up when a fixed γ is implemented, either after TC is satisfied, or at some specified stage. We could also derive $p(\eta_k|\eta_1, \gamma)$, resulting in a pdf on the information space.

At stage k, the density on X after starting at η_1 is given by

$p(x_k|\eta_1, g) =$

$$\int p(x_k|\eta_{k-1}, g_{k-1}(\eta_{k-1}))p(\eta_{k-1}|\eta_{k-2}, g_{k-2}(\eta_{k-2})) \cdots p(\eta_2|\eta_1, g_1(\eta_1))d\eta_{k-1} \cdots d\eta_2. \tag{2.28}$$

The first term in the integrand can be determined using Equation (2.9). Each of the remaining terms can be reduced to

$$p(\eta_{k+1}|\eta_k, g_k(\eta_k)) = p(y_1, \ldots, y_{k+1}, u_1, \ldots, u_k|y_1, \ldots, y_k, u_1, \ldots, u_k) = p(y_{k+1}|\eta_k, u_k). \tag{2.29}$$

This reduction occurs because most of the sensing and action history appears on both sides of the density expression. The right side of (2.29) can be further reduced to

$$p(y_{k+1}|\eta_k, u_k) =$$

$$\int p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k, u_k)dx_{k+1} = \int\int p(y_{k+1}|x_{k+1})p(x_{k+1}|x_k, u_k)p(x_k|\eta_k)dx_k dx_{k+1}, \tag{2.30}$$

in which all three terms in the final integrand are known. The density $p(y_{k+1}|x_{k+1})$ is inferred from the sensing model; $p(x_{k+1}|x_k, u_k)$ is inferred from the state transition equation; and $p(x_k|\eta_k)$ is the density representation of the current information state.

To include the termination condition we replace g by γ above to obtain $p(x_{K+1}|\eta_1, \gamma)$. Reachability and recognizability can be defined in a manner similar to that in Section 2.4.2.1.

2.4.3 Computed examples

In this section we present computed examples that illustrate the forward projection concepts. These forward projections are provided under the assumption that constant motion commands are given to the robot. In other words, some $u \in U$ is chosen, and a strategy is defined as $\gamma_k \equiv u$ for all $k \in \{1, \ldots, K\}$. This will make the comparison of our forward projections to previous research more clear. In Section 2.6.6, we will present forward projections obtained under the implementation of the optimal strategies, as computed by our algorithms. These will also be compared to the forward projections shown in this section.

We have computed forward projection examples in a straightforward way, by using a discretized, array representation for the state space. Under nondeterministic uncertainty, this can be considered as a bitmap representation of the forward projection. Under probabilistic uncertainty, the representation approximates a pdf on X by using a fine grid. In the first step of the computation, the array is initialized to reflect the uncertainty associated with the initial state. At each additional step, the forward projection for the

next stage is represented in a new array, which is determined by applying the given strategy to the elements in the previous array (in the implementation, only two copies of the array are needed at any given time). We have found that this computational technique produces reasonable representations of forward projections.

For the examples considered in this chapter, we assume a two-dimensional, bounded state space (i.e., $\mathcal{C}_{valid} \subseteq \Re^2$), in which each coordinate is constrained to lie in the interval $[0, 100]$. This could, for example, represent the configuration space of a planar robot that is capable of translating in the plane. The obstacles in the workspace will be indicated in figures by gray regions, and a black region will represent the goal.

The first example is depicted in Figure 2.8(a), and can be considered as a configuration-space representation of the classical *peg-in-hole* problem (see for example [26], [62], [110], [124]). The second example is depicted in Figure 2.8(b), and is designed to spread the possible locations of the robot over a large portion of the state space. The initial configuration, x_1, for these two examples is $(50, 96)$, and the top, central part of the state space. We use the control model discussed in Section 2.3.5 and assume that $\|v\|\Delta t = 3$, which implies that the robot is capable of moving three units at each stage. We assume that the maximum angular displacement that can be caused by nature is $\epsilon_\theta = 48.8°$. The given strategy is $\gamma_k \equiv \frac{3}{2}\pi$ for all $k \in \{1, \ldots, K\}$ (i.e., move down).

Figures 2.9 and 2.10 show the forward projections at several different stages, under nondeterministic uncertainty. Figures 2.11 and 2.12 show forward projections under probabilistic uncertainty. For these examples, we assume that $p(\theta^a)$ is uniform on the interval $[-\epsilon_\theta, \epsilon_\theta]$. Initially, the pdf is sharply peaked; however, as control uncertainty accumulates, the density becomes more diffuse. Whenever compliance is possible, the density becomes narrower in the direction perpendicular to the edge. The compliant motions have the effect of "funneling" the probability mass into smaller regions. The pdf values become larger because the density must integrate to one. This effect can be seen

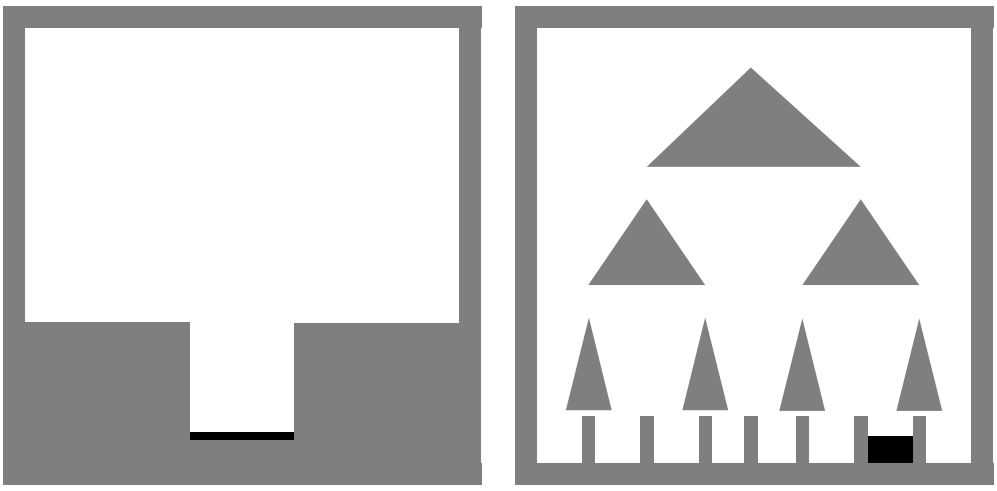


Figure 2.8 Part (a) depicts a simple peg-in-hole example, and part (b) depicts a more complicated example. The obstacles are indicated by gray regions, and the black region represents the goal.

in Figure 2.12 as a triangular obstacle causes the probability mass to divide. In the final stages, there is also a peaking effect; this corresponds to the robot sticking at some final state. Maximizing the probability that the goal will be achieved can be thought of as causing as much of the probability mass to stay in the goal as possible.

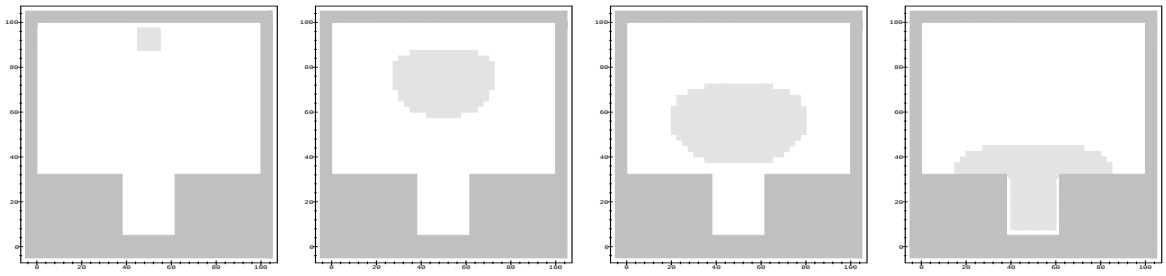


Figure 2.9 The nondeterministic forward projection is represented by the lightly-shaded regions.

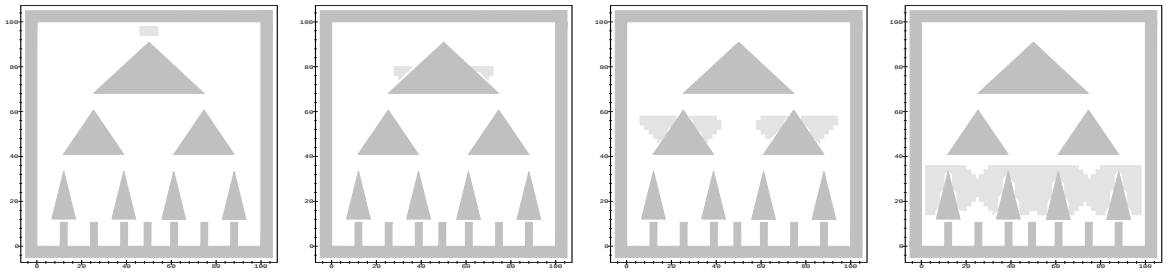


Figure 2.10 The nondeterministic forward projection is represented by the lightly-shaded regions.

A significant distinction between probabilistic and nondeterministic forward projections becomes clear after examining these results. Consider the problem from Figure 2.8(a). The nondeterministic forward projections indicate that little prediction is possible, because the set of possible states grows very quickly. The probabilistic projection is approximately distributed over the same portion of the state space as the nondeterministic projection; however, most of the probability mass appears to terminate in the goal region. This corresponds closely to the arguments about worst-case analysis eliminating many reasonable motion plans; these arguments were given in Section 2.2.2, and also in [26], [27].

It is interesting to note that the densities in Figure 2.11 appear to be Gaussian, even though the uncertainty model is specified as a uniform density. The effects of the control uncertainty combine additively; therefore, the Central Limit Theorem [197] implies that the densities will tend toward Gaussian. Even in Figure 2.12 at $k = 11$, the probability mass appears to be two disjoint Gaussians. We have observed that, as the probability mass divides because of obstacles, the individual components also tend toward being Gaussian. These results indicate that there may be little sensitivity to the particular choice of error model, as long as the mean and variance remain constant.

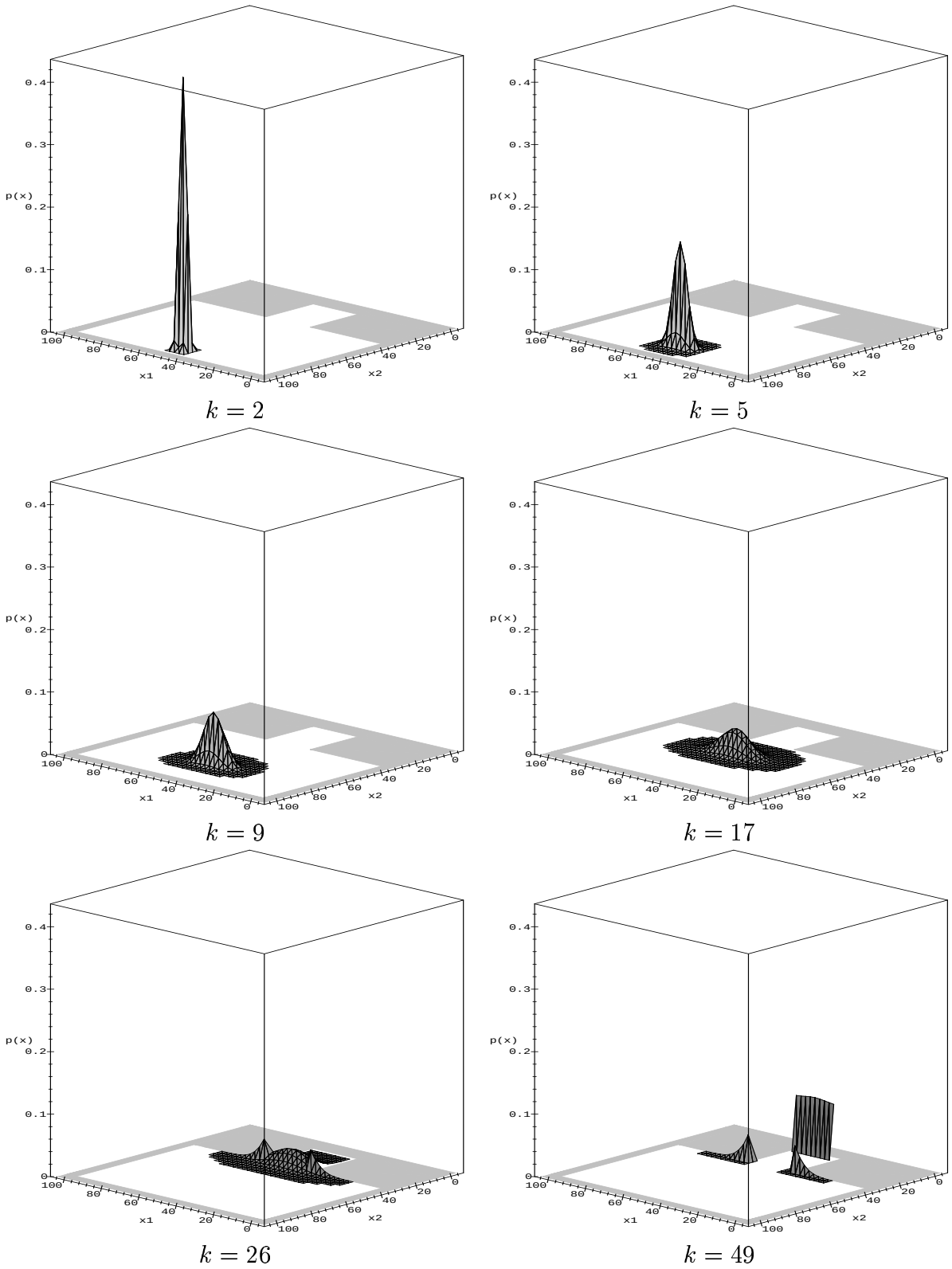


Figure 2.11 The forward projection at several stages, with probabilistic uncertainty.

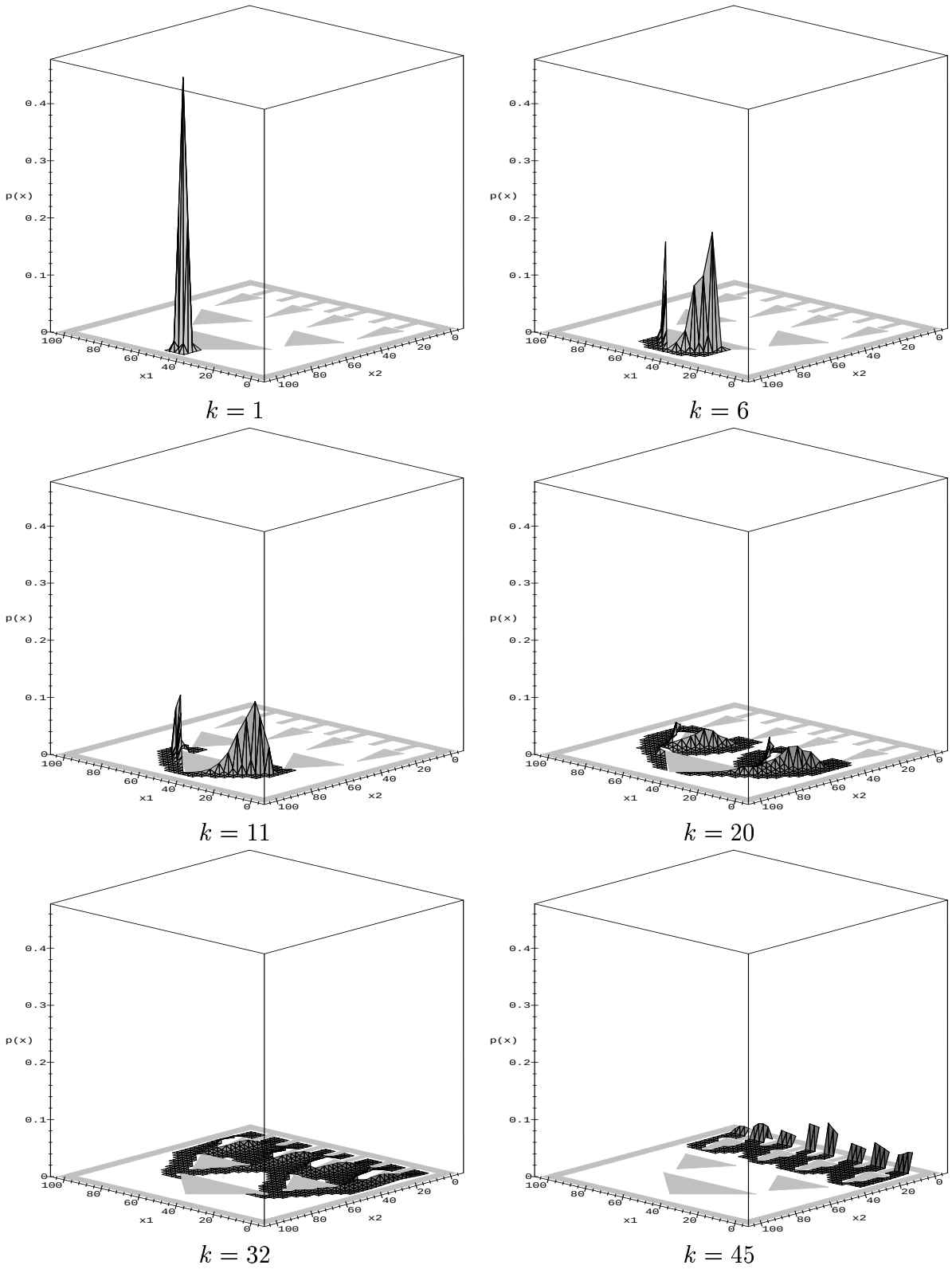
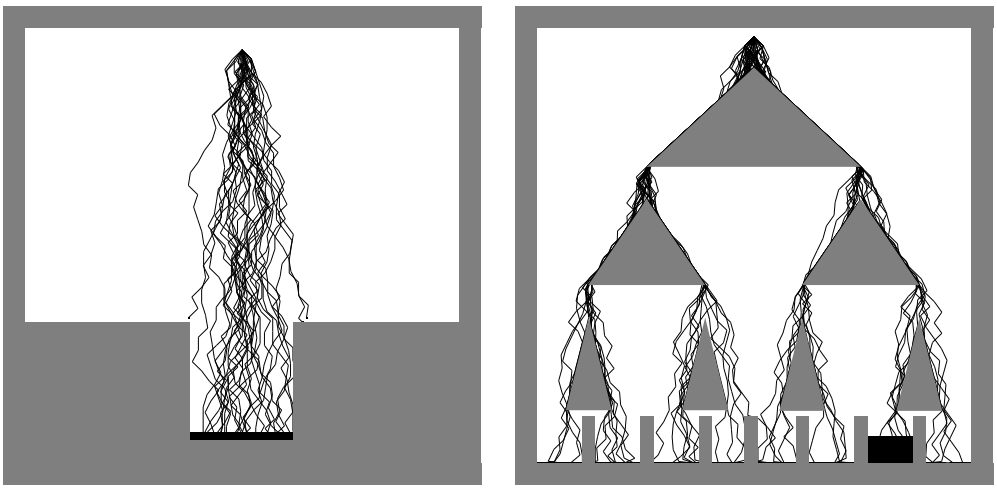


Figure 2.12 The forward projection at several stages, with probabilistic uncertainty.



(a) (b)

Figure 2.13 Sample paths of the random process that results from the given strategy (assuming probabilistic uncertainty). Part (a) corresponds to the peg-in-hole problem, and part (b) corresponds to the more complicated example.

Figure 2.13 shows some additional results that are closely related to the forward projections and that apply to the case of probabilistic uncertainty. Specifying a strategy automatically defines a random process. The transition probabilities are known, and are derived in part from the densities that represent nature's actions. The random process can be considered as a probability space in which the elements are sample paths. Each sample path corresponds to one possible trajectory that could occur under the implementation of the strategy. We can iteratively sample actions for nature and generate a path that represents the state trajectory under the implementation of a strategy. The figures show several sample paths that were superimposed.

2.5 Performance Preimages

In this section, we present performance preimages for each of the four uncertainty cases considered in this chapter. A performance preimage describes a region in the

information space or state space from which the loss in achieving the goal lies within a set of values. This concept generalizes the notion of classical preimages to arbitrary performance measures, although the preimages are defined in discretized time in our formalism. In the same way that classical preimages are useful for evaluating a motion command, the performance preimage is useful for evaluating a strategy. We conclude this section by presenting some computed examples of performance preimages and relating them to previous literature.

2.5.1 Nondeterministic performance preimages

In this section, we assume that nature implements a deterministic, unknown strategy γ^θ, as defined in Section 2.3.4.

2.5.1.1 The perfect information case

We combine the classical preimage concept with the loss functional to evaluate a given strategy. Suppose for a moment that the strategy for nature γ^θ was given to the robot; the loss for choosing robot strategy γ could then be expressed as $L(x_1, \gamma, \gamma^\theta)$, because the state trajectory can be deterministically predicted once x_1, γ, and γ^θ are given. This in turn implies that the action sequence $u_1, u_2, \ldots, u_K$ can also be predicted. Because the strategy of nature is not known by the robot, we define

$$\check{L}(x_1, \gamma) = \sup_{\gamma^\theta \in \Gamma^\theta} L(x_1, \gamma, \gamma^\theta), \tag{2.31}$$

which represents the maximum loss that the robot could receive under the implementation of γ from x_1. This corresponds to modeling nature as an opponent, as is done in minimax design [5].

Recall that the classical preimage is a subset of X from which the robot is guaranteed to achieve the goal for a fixed motion command. Suppose that we are evaluating the

trajectory of the robot with the loss functional in Equation (2.13) for a strategy that consists of a fixed action, repeated at every stage (which is equivalent to a fixed motion command). Elements $x_1 \in X$ such that $\check{L}(x_1, \gamma) = 0$ correspond to locations in the state space from which the robot is guaranteed to achieve the goal and, hence, lie in the classical preimage.

We will next generalize this classical preimage. Note that $\check{L}(x_1, \gamma)$ can be considered as a real-valued function of x_1 for a fixed γ. Consider some subset of the reals, $\mathcal{R} \subseteq \Re$. We define the *performance preimage on* X as a subset of X that is given by

$$\check{\pi}_x(\gamma, \mathcal{R}) = \{x_1 \in X | \check{L}(x_1, \gamma) \in \mathcal{R}\}. \tag{2.32}$$

The set $\check{\pi}_x(\gamma, \mathcal{R}) \subseteq X$ indicates places in the state space from which if $\mathcal{A}$ begins, the loss will lie in $\mathcal{R}$.

We can consider partitioning X into *isoperformance classes* by defining an equivalence class $\check{\pi}_x(\gamma, \{r\})$ for each $r \in [0, \infty)$.

For the loss functional in Equation (2.13), $\check{\pi}_x(\gamma, \{0\})$ yields the classical preimage. Under (2.14) and $\mathcal{R} = [0, m)$ we obtain a performance preimage that indicates all $x_1 \in X$ from which the goal will be achieved with a loss that is guaranteed to be less than m.

If we replace γ with g, and replace the condition "if $x_{K+1} \in G$" in (2.13) with "if $x_k \in G$ for some k," then $\check{\pi}(g, \{0\})$ yields a *backprojection* similar to that in [62].

2.5.1.2 The imperfect information case

We next consider the case in which the robot has imperfect state information. Let $L(\eta_1, \gamma, \gamma^\theta)$ represent the loss that is obtained if the robot implements γ and nature implements γ^θ. Note that if the mapping from X to Y given by h and a fixed θ_k^s is invertible, then x_k can be recovered from y_k and θ_k^s. Hence, when γ^θ is given, the complete state trajectory can be recovered to evaluate L. If invertibility does not hold, then an

additional supremum must be taken over possible x_k, given y_k and θ_k^s. By replacing x_1 by η_1 in (2.31), we define

$$\check{L}(\eta_1, \gamma) = \sup_{\gamma^\theta \in \Gamma^\theta} L(\eta_1, \gamma, \gamma^\theta), \tag{2.33}$$

which represents the maximum amount of loss that the robot could receive under the implementation of γ, while starting from η_1. Note that here γ^θ represents both control and sensing actions.

The loss $\check{L}(\eta_1, \gamma)$ can be considered as a real-valued function of η_1 for a fixed γ. Consider some subset of the reals, $\mathcal{R} \subseteq \Re$. We define the *performance preimage on* N_1 as a subset of N_1, denoted by $\check{\pi}(\gamma, \mathcal{R})$, that is given by

$$\check{\pi}(\gamma, \mathcal{R}) = \{\eta_1 \in N_1 | \check{L}(\eta_1, \gamma) \in \mathcal{R}\}. \tag{2.34}$$

The set $\check{\pi}(\gamma, \mathcal{R}) \subseteq N_1$ indicates places in the information space from which, if $\mathcal{A}$ begins, the loss will lie in $\mathcal{R}$. Such concepts such as isoperformance classes can also be defined on the information space. We could also consider performance preimages on any N_k.

2.5.2 Probabilistic performance preimages

In this section, we assume that nature chooses actions by sampling from a known pdf, $p(\theta)$, which corresponds to a mixed strategy.

2.5.2.1 The perfect information case

Suppose that we wish to evaluate some $\gamma = (g, TC)$ with a given initial state, x_1. If θ is given along with x_1 and a strategy γ, the entire state trajectory $x_1, x_2, \ldots, x_{K+1}$ can be deterministically specified. We can therefore specify the loss for this trajectory as a function $L(x_1, \gamma, \theta)$. This is true because, by using (2.2) and (2.4), x_k can be determined for every state when the value of nature's action, θ, is given.

The *expected* loss incurred if γ is implemented can be expressed as

$$\bar{L}(x_1, \gamma) = \int L(x_1, \gamma, \theta) p(\theta) d\theta, \tag{2.35}$$

in which θ represents the actions taken by nature over all stages. The integral considers each possible action sequence for nature, θ, weighted by the probability density value $p(\theta)$. For any given θ (along with γ and x_1), the action sequence $\{u_1, \ldots, u_K\}$ and state trajectory $\{x_1, \ldots, x_{K+1}\}$ can be completely determined, allowing the evaluation of the loss functional.

We observe for a fixed γ that $\bar{L}(x_1, \gamma)$ can be considered as a real-valued function of x_1. Consider some subset of the reals, $\mathcal{R} \subseteq \Re$. We define the *performance preimage on* X as a subset of X,

$$\bar{\pi}_x(\gamma, \mathcal{R}) = \{x_1 \in X | \bar{L}(x_1, \gamma) \in \mathcal{R}\}. \tag{2.36}$$

The set $\bar{\pi}_x(\gamma, R) \subseteq X$ indicates places in the state space from which, if $\mathcal{A}$ begins, the expected loss lies within $\mathcal{R}$.

We now describe some particular choices for $\mathcal{R}$. Suppose that $\mathcal{R} = [0, r]$ for some $r \geq 0$ (recall that L is nonnegative). The performance preimage yields places in X from which the expected performance will be better than or equal to r. If $\mathcal{R} = \{r\}$ for some point $r \geq 0$, then we obtain places in X in which equal expected performance will be obtained. We can consider partitioning X into *isoperformance classes* by defining an equivalence class $\bar{\pi}_x(\gamma, \{r\})$ for each $r \in [0, \infty)$.

The loss functional in Equation (2.13) implies that we are interested only in achieving the goal, without any considering efficiency in the actual robot trajectory. The loss $\bar{L}(x_1, \gamma)$ in this case represents the probability that the goal will not be achieved using γ. We consider some $\bar{\pi}_x(\gamma, [0, \{r\}])$ for $r \in [0, 1]$ as a *probabilistic preimage on* X. The probabilistic preimage thus indicates places in X from which the goal will be achieved

with probability of at least $1 - r$. Furthermore, if we replace γ with g, and replace the condition "if $x_{K+1} \in G$" in (2.13) with "if $x_k \in G$ for some k," then $\bar{\pi}_x(g, [0, \{r\}])$ yields a *probabilistic backprojection* quite similar to that in [26]. We can also consider $\bar{\pi}_x(\gamma, \{r\})$ as an *isoprobability class*. The isoprobability class $\bar{\pi}_x(\gamma, \{0\})$ corresponds to the classical preimage notion, in which the goal is guaranteed to be achieved, although in our case, it is more appropriate to claim that the goal will be achieved with probability one.

2.5.2.2 The imperfect information case

In general, the robot $\mathcal{A}$ will begin in some uncertain initial state. Therefore, we also consider performance preimages on the robot's initial information space, N_1. The preimages represent places in the initial information space where, if $\mathcal{A}$ begins, the expected performance will lie within some $\mathcal{R} \subseteq \Re$. This result specializes to Equation (2.36) in the case of a given initial state, $\eta_1 = y_1 = x_1$. As a minor extension, we could also consider performance preimages on any information space N_k.

The *expected* loss incurred if γ is implemented can be expressed as

$$\bar{L}(\eta_1, \gamma) = \int L(\eta_1, \gamma, \theta) p(\theta) d\theta. \tag{2.37}$$

For a subset $\mathcal{R} \subseteq \Re$, we define the *performance preimage on* N_1 as a subset of N_1, denoted by $\bar{\pi}(\gamma, \mathcal{R})$, that is given by

$$\bar{\pi}(\gamma, \mathcal{R}) = \{\eta_1 \in N_1 | \bar{L}(\eta_1, \gamma) \in \mathcal{R}\}. \tag{2.38}$$

Implications about the invertibility of h given y_k and θ_k^s are relevant here as in Section 2.5.1.2.

2.5.3 Computed examples

In this section we present several computed preimages. As in Section 2.4.3, these results are provided under the assumption that constant motion commands are given

to the robot. This will make the comparison of our preimages to previous research clearer. In Section 2.6.6, we will show preimages obtained under the implementation of the optimal strategies, as computed by our algorithms.

These examples were computed using the techniques that will be presented in Section 2.6. It will be seen that representation of the performance preimages is a by-product of determining the optimal strategy. The evaluation of a given strategy, which is done in this section, can be considered as the trivial case of computing the optimal strategy for which there is only one choice in the space of possible strategies.

We begin by returning to the peg-in-hole problem, which was discussed in Section 2.4.3. Suppose that the fixed action is $\frac{3}{2}\pi$, and that the maximum angular displacement, $\epsilon_\theta = 14.3°$. Recall that ϵ_θ was given in Section 2.3.5. We will use Equation (2.13) for all of the examples in this section because we have observed that (2.14) produces very similar curves under fixed motion commands; when considering optimal strategies, however, the differences between the two loss functionals become much more important.

Figure 2.14(a) shows a performance preimage under nondeterministic uncertainty. The subset of the state space that is below the curve corresponds to places in the state space from which the goal is guaranteed to be achieved. Note that this result does not depend on $\|v\|\Delta t$; this is because with nondeterministic uncertainty, the robot configuration can lie anywhere within the cone generated from the initial state and $\pm\epsilon_\theta$. The curve shown in Figure 2.14(a) corresponds closely to the classical preimage that has been determined for this problem in previous manipulation planning research (e.g., [62], [109]).

In Figure 2.14(b), we show probabilistic backprojections that are quite similar to those that appear in [26], [27]. Figure 2.15 shows a three-dimensional plot of $\bar{L}(x_1, \gamma)$. We assume that $p(\theta^a)$ is uniform on the interval $[-\epsilon_\theta, \epsilon_\theta]$, and $\epsilon_\theta = 48.8°$. We assume that $\|v\|\Delta t = 200$, and let $K = 1$. There is only one decision-making stage, and the robot can move enough distance to accomplish the goal in a single stage. The figures shows

isoprobability curves from $\bar{\pi}_x(\gamma^*, \{0.2\})$ to $\bar{\pi}_x(\gamma^*, \{0.9\})$, at evenly spaced probability values. The innermost curve represents the $\bar{\pi}(\gamma^*, \{0.2\})$.

The remaining examples show performance preimages for cases in which there are no similar results in the literature. Suppose that instead of using a uniform density for control error, a zero-mean, truncated Gaussian density is used (this density was also used in [25]). The resulting performance contours are shown in Figure 2.14(c). Again, we show the preimages from $\bar{\pi}_x(\gamma^*, \{0.2\})$ to $\bar{\pi}_x(\gamma^*, 0.9)$, at evenly spaced probability values. In general, we can substitute any density into the model and observe the resulting preimages.

For the remaining examples in this section, we use probabilistic uncertainty and assume that $p(\theta^a)$ is uniform on the interval $[-\epsilon_\theta, \epsilon_\theta]$ and that $\epsilon_\theta = 48.8°$. We also assume that $\|v\|\Delta t = 3$, which implies that the robot moves only a small amount before additional control uncertainty is added. Figure 2.14(d) shows the resulting contours (with the same preimage values as used previously).

We conclude this section with two additional examples, which are depicted in Figures 2.16(a) and 2.16(b). Probabilistic performance preimages are shown for these problems in Figures 2.16(c) and 2.16(d), respectively. These examples differ from the previous example, because the configuration-space obstacles are more complex. We assume that the initial state and fixed motion command are the same as used in the previous example. In these examples, the curves appear separated around obstacle boundaries due to the effects of compliant motion. For example, in Figure 2.16(c), the curves are separated because of compliant motions on the top part of the triangular obstacle. Although there is significant uncertainty in control, the edge of the triangle guides the robot into the goal region, significantly reducing the expected loss.

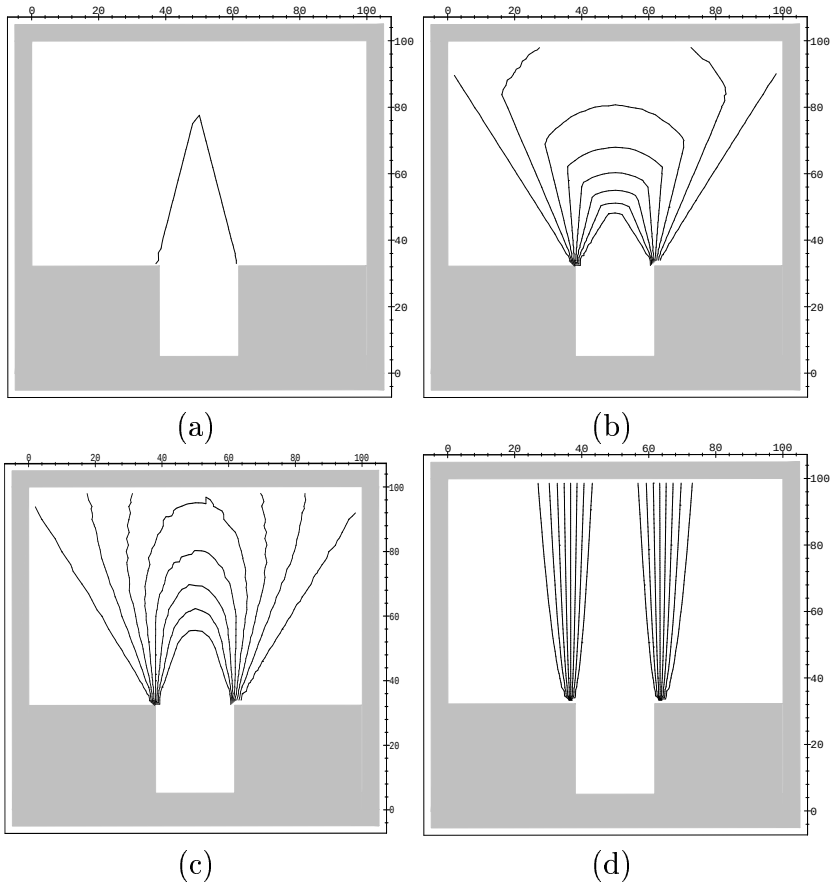


Figure 2.14 Several computed performance preimages for the classic peg-in-hole problem: (a) a classical preimage; (b) a single-stage probabilistic preimage for a uniform state transition pdf; (c) a single-stage probabilistic preimage for a truncated Gaussian state transition pdf; (d) a multi-stage probabilistic preimage for a uniform state transition pdf.

2.6 Designing Optimal Strategies

Sections 2.4 and 2.5 have presented methods that evaluate a *given* strategy. Section 2.6 defines concepts of optimality and presents a computational approach that *selects* a strategy. Section 2.6.1 defines optimality conditions for each of the four uncertainty types that have been considered throughout this chapter. Section 2.6.2 presents the principle of optimality, which represents a recursive constraint that significantly reduces the amount of computation necessary to determine a solution. Section 2.6.3 discusses

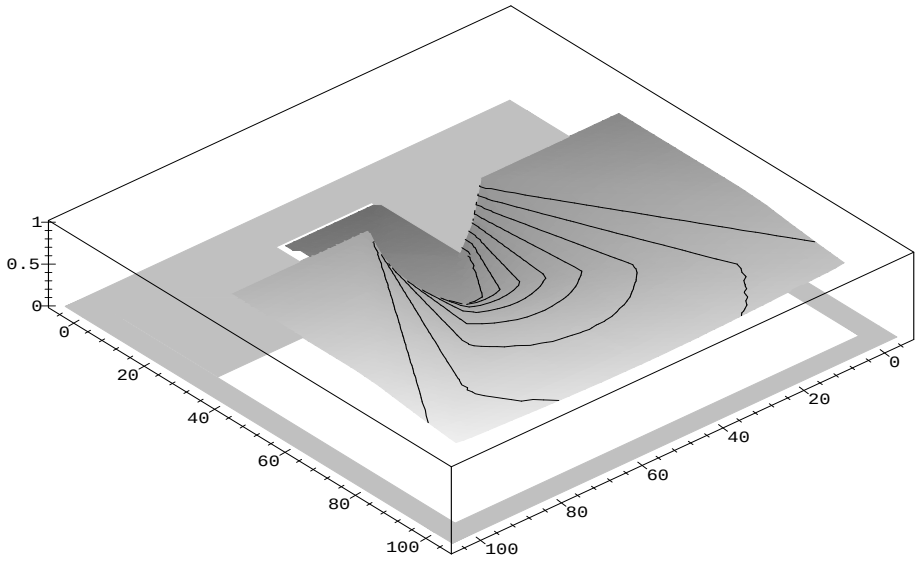


Figure 2.15 A plot of $\bar{L}$ under probabilistic uncertainty.

computational issues caused by uncertainty in control. Section 2.6.4 discusses the additional issues involved when planning in the information space. Section 2.6.5 discusses the computational performance. Finally, Section 2.6.6 presents computed optimal strategies, including forward projections and preimages.

2.6.1 Defining optimality

A strategy, γ, has been used in this work to precisely describe the behavior of the robot with respect to state or information. The design problem is to select the most desirable strategy, γ^*, from the space of allowable strategies Γ. We have already defined a loss functional that evaluates the state trajectory. Without the effects of nature's actions, the design problem would simply be formulated as selecting a strategy to produce a trajectory that minimizes the loss. Under nondeterministic uncertainty, nature is considered as an opponent with diametrically opposed interests; therefore, a strategy is selected to minimize the maximum amount of loss that could result from the strategy of

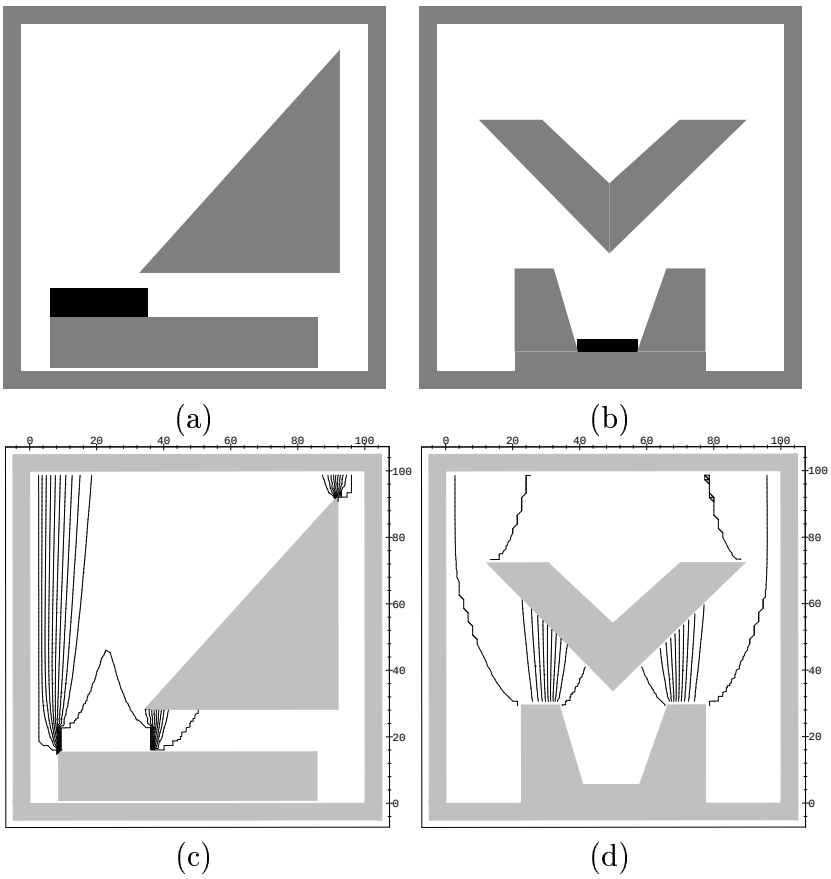


Figure 2.16 In these examples, the obstacles displace the curves. Parts (a) and (b) show example problems, and parts (c) and (d) show corresponding computed performance preimages. In these examples, the obstacles displace the curves.

nature. Under probabilistic uncertainty, the actions of nature can be characterized with probability densities; hence, a strategy is selected to minimize the expected loss.

2.6.1.1 Optimality under nondeterministic uncertainty

Perfect information Recall that Γ^θ denotes the space of deterministic strategies for nature when nondeterministic uncertainty is considered. Under perfect information and

nondeterministic control uncertainty, the ideal choice for a strategy, $\gamma^* \in \Gamma$, satisfies

$$\check{L}(x_1, \gamma^*) = \inf_{\gamma \in \Gamma} \check{L}(x_1, \gamma) = \inf_{\gamma \in \Gamma} \sup_{\gamma^\theta \in \Gamma^\theta} L(x_1, \gamma, \gamma^\theta) \tag{2.39}$$

for all $x_1 \in X$. This indicates that from any initial state, the strategy will guarantee the least possible loss given the worst-case actions of nature. This concept has been used previously to design controllers based on worst-case analysis [5].

If the loss functional in Equation (2.13) is used, then the space of strategies Γ can be partitioned into two equivalence classes: those that achieve the goal (resulting in a worst-case loss of one) and those that may fail to achieve the goal (resulting in a worst-case loss of zero). Any strategy in the first equivalence class satisfies Equation (2.39), and directly corresponds to the common approach in the previous manipulation planning research of selecting a strategy that is guaranteed to achieve the goal. By using another loss functional, such as Equation (2.14), Equation (2.39) can be used to partition Γ into many more classes; this induces preferences on the set of strategies that achieve the goal.

Imperfect information To obtain the concept of optimality for the imperfect information case, the objective is to select $\gamma^* \in \Gamma$ such that

$$\check{L}(\eta_1, \gamma^*) = \inf_{\gamma \in \Gamma} \check{L}(\eta_1, \gamma) = \inf_{\gamma \in \Gamma} \sup_{\gamma^\theta \in \Gamma^\theta} L(\eta_1, \gamma, \gamma^\theta) \tag{2.40}$$

for all $\eta_1 \in N_1$. Equation (2.40) is almost identical to (2.39), except that the state space is replaced by the information space.

2.6.1.2 Optimality under probabilistic uncertainty

Perfect information Recall that the actions of nature can be partially predicted through the specification of a pdf, $p(\theta)$, in which θ represents the action, θ_k, of nature at every stage.

Under perfect information and probabilistic control uncertainty, the design task is to select a strategy $\gamma^* \in \Gamma$ such that

$$\bar{L}(x_1, \gamma^*) = \inf_{\gamma \in \Gamma} \bar{L}(x_1, \gamma) = \inf_{\gamma \in \Gamma} \int L(x_1, \gamma, \theta) p(\theta) d\theta \tag{2.41}$$

for all $x_1 \in X$. This corresponds to selecting a strategy that minimizes the loss in the expected sense, as considered in stochastic optimal control theory [105].

Imperfect information Under imperfect information, the state space in (2.41) is replaced by the information space to obtain

$$\bar{L}(\eta_1, \gamma^*) = \inf_{\gamma \in \Gamma} \bar{L}(\eta_1, \gamma) = \inf_{\gamma \in \Gamma} \int L(\eta_1, \gamma, \theta) p(\theta) d\theta \tag{2.42}$$

for all $\eta_1 \in N_1$.

2.6.2 The principle of optimality

Dynamic programming is a powerful tool that underlies many of the solution techniques for dynamic decision-making problems. The key is the principle of optimality [16], which states that an optimal solution can be recursively decomposed into optimal parts. In general, this optimization concept has been useful in a variety of contexts, both for producing analytical solutions and for numerical computation procedures. In this section we formulate the principle of optimality for each of the four types of uncertainty that are considered in this chapter. For each type, this principle can be viewed as a useful constraint that significantly simplifies the amount of computation that is required to determine an optimal solution.

2.6.2.1 The nondeterministic case

Perfect information Suppose that for some k, the optimal strategy is known for each stage $i \in \{k, \ldots, K\}$. The optimal worst-case loss obtained by starting from stage k and implementing the portion of the optimal strategy $\{\gamma_k^*, \ldots, \gamma_K^*\}$ can be represented as (recall Equation (2.12)):

$$\check{L}_k^*(x_k) = \sup_{\gamma^\theta \in \Gamma^\theta} \left\{ \sum_{i=k}^{K} l_i(x_i, \gamma_i^*(x_i)) + l_{K+1}(x_{K+1}) \right\}, \tag{2.43}$$

We use $\gamma_i^*(x_i)$ to represent the simultaneous choice of u_i^* and TC_i^*. Under the implementation of a strategy, the state trajectory depends on the actions chosen by nature; therefore, the expression in the sup depends on nature. The function $\check{L}_k^*(x_k)$ is sometimes referred to as the *cost-to-go* function in dynamic optimization literature [17].

The principle of optimality [6] states that $\check{L}_k^*(x_k)$ can be obtained from $\check{L}_{k+1}^*(x_{k+1})$ by the following recurrence:

$$\check{L}_k^*(x_k) = \inf_{\gamma_k \in \Gamma_k} \sup_{\theta_k^a \in \Theta_k^a} \left\{ l_k(x_k, \gamma_k(x_k)) + \check{L}_{k+1}^*(f(x_k, u_k, \theta_k^a)) \right\}. \tag{2.44}$$

Recall that $f(x_k, u_k, \theta_k^a)$ represents x_{k+1}; hence, this defines a recursion. We assume that termination is implicitly included as a possible choice.

The goal is to determine the optimal action, u_k, and termination condition, TC_k, for every value of x_k, and every stage $k \in \{1, \ldots, K\}$. We can begin with stage $K+1$, and repeatedly apply (2.44) to obtain the optimal actions. At stage $K+1$, we can use the last term of (2.13) to obtain $\check{L}_{K+1}^*(x_{K+1}) = l_{K+1}(x_{K+1})$. The cost-to-go, $\check{L}_K^*$, can be determined from $\check{L}_{K+1}^*$ through (2.44). Using the $u_K \in U$ and TC_k that minimize (2.44) at x_K, we define $\gamma_K^*(x_K) = \{u_K, TC_K\}$ for each x_k. We then apply (2.44) again, using $\check{L}_K^*$ to obtain $\check{L}_{K-1}^*$ and γ_{K-1}^*. These iterations continue until $k = 1$. Finally, we take $\gamma^* = \{\gamma_1^*, \ldots, \gamma_K^*\}$.

The loss function, $\check{L}_1^*$, is similar to the concept of a global navigation function in motion planning [109], [160], as each represents a function on the configuration space that provides local control of the robot and contains a single minimum, at the goal. The *wavefront expansion method* that is described in [109] can be viewed as constructing a cost-to-go function.

Recall that because of stationarity, the strategy γ_k does not depend on the stage index k for the problems that we consider.

Imperfect information We now describe how the dynamic programming equation is applied under nondeterministic sensing and control uncertainties. From a given information state, we wish to evaluate a partial strategy from stage k to stage K. Previously, we used the notation $\check{L}_k^*(x_k)$ to evaluate a part of an optimal strategy from a given state. Using the information state representation $F_k(\eta_k)$, which was defined in Section 2.3.3, we have

$$\check{L}_k^*(\eta_k) = \sup_{x_k \in F_k(\eta_k)} \check{L}_k^*(x_k). \tag{2.45}$$

We want to consider the effect of selecting $\gamma_k(\eta_k)$ in the information space, η_k. This results in $F_{k+1}(\eta_{k+1})$, as defined in (2.18). We additionally assume that the per-stage loss does not depend on state, $l(x_k, u_k, TC_k) = l(u_k, TC_k)$, which encompasses the loss functionals considered thus far (this assumption is not required in general).

The dynamic programming principle states that $\check{L}_k^*(\eta_k)$ can be obtained from $\check{L}_{k+1}^*(\eta_{k+1})$ by the following recurrence:

$$\check{L}_k^*(\eta_k) = \inf_{\gamma_k \in \Gamma_k} \sup_{\eta_{k+1} \in \tilde{F}_{k+1}(\eta_k, u_k)} \left\{ l(\gamma_k(\eta_k)) + \check{L}_{k+1}^*(\eta_{k+1}) \right\}, \tag{2.46}$$

in which $\check{L}_k^*(\eta_k)$ represents the optimal worst-case loss, obtained by implementing the optimal strategy γ^* from stage k to stage $K+1$.

At stage $K+1$, we can use the last term of Equation (2.12) to obtain

$$\bar{L}^*_{K+1}(\eta_{K+1}) = \sup_{x_{K+1} \in F_{K+1}(\eta_{K+1})} l_{K+1}(x_{K+1}). \tag{2.47}$$

2.6.2.2 The probabilistic case

Perfect information We next present the principle of optimality under probabilistic control uncertainty. The resulting equation can be applied in the same iterative manner to obtain an optimal solution.

The expected loss obtained by starting from stage k and implementing the portion of the optimal strategy $\{\gamma^*_k, \ldots, \gamma^*_K\}$ can be represented as

$$\bar{L}^*_k(x_k) = E\left\{\sum_{i=k}^{K} l_i(x_i, \gamma^*_i(x_i)) + l_{K+1}(x_{K+1})\right\}, \tag{2.48}$$

in which $E\{\}$ denotes expectation taken over the actions of nature.

The principle of optimality [105] states that $\bar{L}^*_k(x_k)$ can be obtained from $\bar{L}^*_{k+1}(x_{k+1})$ by the following recurrence:

$$\bar{L}^*_k(x_k) = \min_{\gamma_k \in \Gamma_k} \left\{ l_k(x_k, u_k) + \int \bar{L}^*_{k+1}(x_{k+1}) p(x_{k+1}|x_k, u_k) dx_{k+1} \right\}. \tag{2.49}$$

Note that the integral is taken over states that can be reached using the state transition equation.

Imperfect information We now describe how the dynamic programming equation is applied under probabilistic sensing and control uncertainties. From a given information state, we wish to evaluate a partial strategy from stage k to stage K. Previously, we used the notation $\bar{L}^*_k(x_k)$ to represent the expected loss of executing a partial, optimal strategy from a given state. Using the information state density $p(x_k|\eta_k)$ on X,

$$\bar{L}^*_k(\eta_k) = \int \bar{L}^*_k(x_k) p(x_k|\eta_k) dx_k. \tag{2.50}$$

We also consider the one-stage expected loss associated with taking an action, from a given information state, η_k:

$$\bar{l}(\eta_k, \gamma_k(\eta_k)) = \int l(x_k, \gamma_k(\eta_k)) p(x_k|\eta_k) dx_k. \tag{2.51}$$

This is the expected loss that will be incurred if an action u_k and TC_k are taken from state η_k, resulting in some η_{k+1}. The integrand of (2.51) is determined from (2.12) and (2.10). Using the previous notation, the dynamic programming principle states that $\bar{L}^*_k(\eta_k)$ can be obtained from $\bar{L}^*_{k+1}(\eta_{k+1})$ by the following recurrence:

$$\bar{L}^*_k(\eta_k) = \inf_{\gamma_k(\eta_k)} \left\{ \bar{l}(\eta_k, \gamma_k(\eta_k)) + \int \bar{L}^*_{k+1}(\eta_{k+1}) p(\eta_{k+1}|\eta_k, \gamma_k(\eta_k)) d\eta_{k+1} \right\}. \tag{2.52}$$

Above, $p(\eta_{k+1}|\eta_k, \gamma_k(\eta_k))$ is determined by replacing g with γ in (2.29).

At stage $K+1$, we can use the last term of (2.12) to obtain

$$\bar{L}^*_{K+1}(\eta_{K+1}) = \int l_{K+1}(x_{K+1}) p(x_{K+1}|\eta_{K+1}) d\eta_{K+1}. \tag{2.53}$$

2.6.3 Approximating the state space

We determine optimal strategies numerically by successively building approximate representations of $\bar{L}^*_k$ (or $\check{L}^*_k$). This offers flexibility in spite of the fact that analytical solutions are very difficult to obtain. Each dynamic programming iteration can be considered as the construction of an approximate representation of $\bar{L}^*_k$. We decompose the state space into cells of uniform size; however, it is important to note the differences between the use of this decomposition in our context and the hierarchical decompositions often used in geometric motion planning (see, for example, [109]). Our primary interest in using the decomposition is to construct a good approximation of the continuous function $\bar{L}^*_k$ (or $\check{L}^*_k$) over the entire state space (or information space under sensing uncertainty). This makes the application of a hierarchical decomposition more difficult.

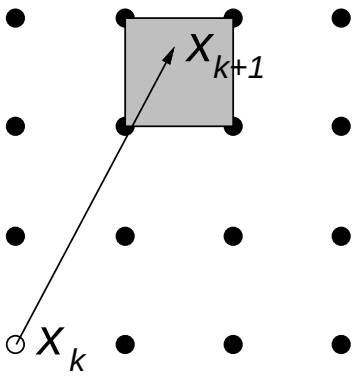


Figure 2.17 Interpolation is performed on the shaded region to obtain a more accurate value for $\bar{L}^*_{k+1}$.

We obtain the value for $\bar{L}^*_k(x_k)$ by computing the right side of (2.44) (or the appropriate dynamic programming equation) for various values of u_k and TC_k and using linear interpolation (see Figure 2.17). Other schemes, such as quadratic interpolation, can be used to improve numerical accuracy [108].

Note that $\bar{L}^*_K$ represents the cost of the optimal one-stage strategy from each state x_K. More generally, $\bar{L}^*_{K-i}$ represents the cost of the optimal $(i+1)$-stage strategy from each state x_{K-i}. For a motion planning problem, we are concerned only with strategies that require a finite number of stages before terminating in the goal region, and we assume that stationarity holds, as discussed in Section 2.3.4. We select a positive $\delta \approx 0$ and terminate the dynamic programming iterations when $|\bar{L}^*_k(x_k) - \bar{L}^*_{k+1}(x_{k+1})| < \delta$ for all values in the state space. The resulting strategy is formed from the optimal actions and termination conditions in the final iteration. Note that no choice of K is necessary. Also, at each iteration of the dynamic programming algorithm, we retain only the representation of $\bar{L}^*_{k+1}$ while constructing $\bar{L}^*_k$; earlier representations can be discarded.

To execute a strategy, the robot uses the final cost-to-go representation (which we call $\bar{L}^*_1$) in a way similar to the use of a potential field [12], [92], [98], [160]. The optimal action can be obtained from any real-valued location $x \in X$ though the use of (2.44) (or

the appropriate dynamic programming equation), interpolation, and the approximate representation of $\bar{L}_1^*$. A real-valued initial state is given. Thus, the robot is not confined to move along the quantization grid that is used for determining the cost-to-go functions. The application of the optimal action will yield a new real-valued configuration for the robot. This form of iteration continues until $TC_k = true$.

2.6.4 Approximating the information space

With sensing uncertainty, planning occurs in the information space. This space, however, is considerably more difficult to utilize than the state space. In one representation, the dimension of the information space is proportional to the number of stages. In another representation, the state space is considered as a function space of pdf's or a space of subsets. In this section, we discuss methods that can be used to simplify the information space. These methods involve tradeoffs between the computational expense and the quality of the information space representation. We are presently experimenting with these alternatives, and many issues exist that may cause one method to be preferable to another. In the current implemented examples, which are presented in Section 2.6.6, we limit the history to the past sensor observation. This results in *sensor feedback*, which is similar to the approach used in [58]. In [10], an approach was recently proposed that uses a state aggregation technique to approximate the information space in a mobile-robot planning problem.

These techniques can be used in combination with interpolation, which was discussed in Section 2.6.3. Strategies are determined, however, by successively building cost-to-go functions in the information space, rather than in the state space.

Limiting history As defined in Section 2.3.2, η_k is defined as a subset of the sensing and action history. One straightforward way to keep the information space dimension

fixed is to limit the amount of history that is retained. For instance, we can maintain i stages of history, to obtain:

$$\eta_k = \{u_{k-i+1}, u_{k-i+2}, \ldots, u_{k-1}, y_{k-i}, y_{k-i+1}, \ldots, y_k\}. \tag{2.54}$$

If $i = 0$, then only the last sensor observation can be retained for decision making, which results in a sensor-feedback strategy. If position sensing is used along with a directional force sensing, then the information space is reduced to having one more dimension than is the state space.

Introducing statistics A more general way to reduce the information space complexity is to transform the history into a lower-dimensional space. This technique encompasses the history limiting approach. An ideal situation exists when an information space can be transformed using a low-dimensional *sufficient statistic* [51], [105]. A sufficient statistic implies that the transformation does not cause any loss of "information." In other words, any decision that is based on the complete history can equivalently be made by considering only the statistic. The identity function is trivially a sufficient statistic. A statistic can be selected that is not sufficient; however, the effects that the projection has on the information space and decisions must be carefully considered.

In general, a transformation of the form

$$\eta_k = z_k(u_1, \ldots, u_{k-1}, y_1 \ldots, y_k), \tag{2.55}$$

is applied to the history. The information space N_k, as defined in previous sections, can be replaced by the statistic space Z_k, for which $z_k \in Z_k$. A strategy is then defined as a mapping on Z_k, and dynamic programming can again be applied to yield solutions.

Functional approximation with moments One special type of statistic that can be used to approximate the information spaces is the set of *moments*. Because that the

information space can be considered as a function space. For example, with probabilistic uncertainty in sensing, the information space can be represented by the space of possible density functions. We will discuss the moment-based approach with respect to probabilistic uncertainty; however, a similar principle can be applied when using nondeterministic uncertainty. Because moments summarize information in random variables, we can, in general, use moment computations to approximate this function space. Recall that continuous functions can be approximated by coefficients of a Taylor series expansion. As the dimension of the expansion is increased, the approximations are more accurate. We can obtain a similar situation when considering the information space. The implication of using fixed-order moments is that all decisions must be made without being able to distinguish information states that differ in higher-order moments.

Consider, as an example, a second-order approximation. Recall from Section 2.4.2.2 that from any sensing and action history (i.e., $\{u_1, u_2, \ldots, u_{k-1}, y_1, y_2, \ldots, y_k\}$), the pdf on the state space can be inferred, $p(x_k|\eta_k)$. Let μ_k and Σ_k represent the mean vector and covariance matrix, respectively, of $p(x_k|\eta_k)$. From any history, we can now obtain the moments μ_k and Σ_k. The computation of such moments is similar to the approach taken in [180]. For some systems, Kalman filtering could be applied to obtain the mean and covariance estimates [38].

If η_k are $\gamma_k(\eta_k)$ given, a density $p(\eta_{k+1}|\eta_k, \gamma_k(\eta_k))$ for the next information state will be obtained. This was used in (2.52) as part of the principle of optimality, which can be used to compute an optimal strategy. When using moments, we can replace $p(\eta_{k+1}|\eta_k, \gamma_k(\eta_k))$ by $p(\mu_{k+1}, \Sigma_{k+1}|\mu_k, \Sigma_k, \gamma_k(\mu_k, \Sigma_k))$ in (2.52). This can be used to determine optimal strategies from moments (optimal on the approximated information space).

An information-feedback strategy, $\gamma_k(\eta_k)$, is then replaced by a moment-feedback strategy, $\gamma_k(\mu_k, \Sigma_k)$. The hope in using moment approximation is that $\gamma_k^*(\eta_k) \approx \gamma_k^*(\mu_k, \Sigma_k)$ for all $\eta_k \in N_k$. Another way the evaluate the moment-based approach is to determine

whether

$$\bar{L}(\eta_1, \gamma^*) \approx \bar{L}(\mu_1, \Sigma_1, \gamma^*). \tag{2.56}$$

2.6.5 Computational performance

This section briefly discusses the computational performance of the dynamic programming computations. To evaluate computational performance, there are two phases to consider: determination of the optimal strategy, and execution of the optimal strategy. The iterated dynamic programming computations are performed off-line to determine the optimal strategy. Let Q denote the number of cells per dimension in the representation of $\mathcal{C}_{free}$. Let n denote the dimension of the information space (which becomes the dimension of the state space in the case of perfect information). Let $|U|$ denote the number of actions that are considered. Let $|\Theta|$ denote the number of actions that are considered by nature. The space complexity of the algorithm is $O(Q^n)$, which is proportional to the size of the state space. For each iteration of the dynamic programming, the time complexity is $O(Q^n|U|\ |\Theta|)$, and the number of iterations is proportional to the robot velocity and the complexity of the solution strategy. The computation at each cell (in the application of (2.44)) has time complexity $O(|U|\ |\Theta|)$, with n fixed.

The number of iterations required is directly proportional to the number of stages required for sample paths to reach the goal. After k iterations, sample paths that reach the goal in k or fewer stages have been considered. As k increases, the probability decreases of existing sample paths that take more than k stages to reach the goal under the implementation of the optimal strategy. This gradually leads to stabilization in the cost-to-go function. For the problems that we have considered, the number of iterations required for convergence ranges from about 50 to 200.

The computational cost of dynamic programming increases exponentially in the dimension of the state space for perfect information and in the information space for imperfect sensing; however, most algorithms that solve the basic motion planning problem without sensing and control uncertainty have exponential complexity in the dimension of the configuration space (see [91], [109] for surveys and comparisons). We consider the current approach to be reasonable for a few dimensions, which includes many interesting motion planning problems. For more difficult problems some additional computational techniques would have to be developed.

In our simulation experiments, we have considered problems in which $X \subseteq \Re^2$. We typically divide the state space into 50×50 cells, and 64 quantized actions are used to approximate translational motion. We have considered similar quantizations of the information space under sensor feedback. The computation times vary dramatically, depending on the resolutions of the representation. For the examples that we present in this chapter, the computation times vary from a few minutes to a few hours on a SPARC 10 workstation. Significant improvement of these off-line computations can be obtained through additional code optimization and parallelization; however, these implementation issues are beyond the focus of this research. It is important to note that the dynamic programming equations are highly parallelizable. For example, under probabilistic uncertainty with perfect state information, the computation of the optimal action at each location x_k, depends only on a very local portion of the representation of $\bar{L}^*_{k+1}(x_{k+1})$, and on no portion of $\bar{L}^*_k(x_k)$. A parallelized implementation of the algorithm would significantly improve performance.

The on-line execution of the optimal strategy proceeds very quickly. For each stage, a single evaluation of the dynamic programming equation is performed to yield the optimal action. This computation is on the order of a few milliseconds, and is therefore quite reasonable for practical applications.

2.6.6 Computed examples

In this section, we present computed examples of optimal strategies that were determined by the computational methods discussed in this section. For these strategies, we show forward projections and preimages, which can be compared to the results in Sections 2.4.3 and 2.5.3.

For the results in this section, we used the loss functional in Equation (2.14) with $l(u_k, TC_k) = \|v\|\Delta t$ and $C_f = 10000$. We have found the loss functional in Equation (2.13) to not be as useful for determining optimal strategies. For most problems, the cost-to-go is zero at every state from which it is possible to achieve the goal. Therefore, there are many strategies that are considered equivalent, while in reality the expected time (or worst-case time) for some of the strategies to achieve the goal may be arbitrarily longer. For fixed motion commands, however, Equation (2.13) provided useful information because a strategy was chosen that in many possible trajectories did not achieve the goal.

For the first example, we refer to the peg-in-hole problem introduced in Section 2.4.3. We assume, as considered previously, that $\|v\|\Delta t = 3$ and $\epsilon_\theta = 48.8°$. Figures 2.18(a) and 2.18(b) show computed results that were obtained under probabilistic uncertainty with perfect state information. Figure 2.18(a) depicts the optimal strategy by showing the direction of the motion command $u_k = \gamma_k^*(x_k)$ at different locations in the state space. Figure 2.18(b) shows isoperformance classes for every 6 units (i.e., there is a contour for every two expected stages of motion). Figure 2.19 shows a three-dimensional rendering of the cost-to-go function, along with the preimages. This can be compared to the preimage results from Section 2.5.3; under the implementation of the optimal strategy, the curves emanate radially from the goal region. In Figure 2.20 shows the probabilistic forward projection. Under the optimal strategy, all of the probability mass ends up in the goal

region. This can be compared to the forward projection for the fixed motion command that was shown in Figure 2.11, for which some of the probability mass did not reach the goal.

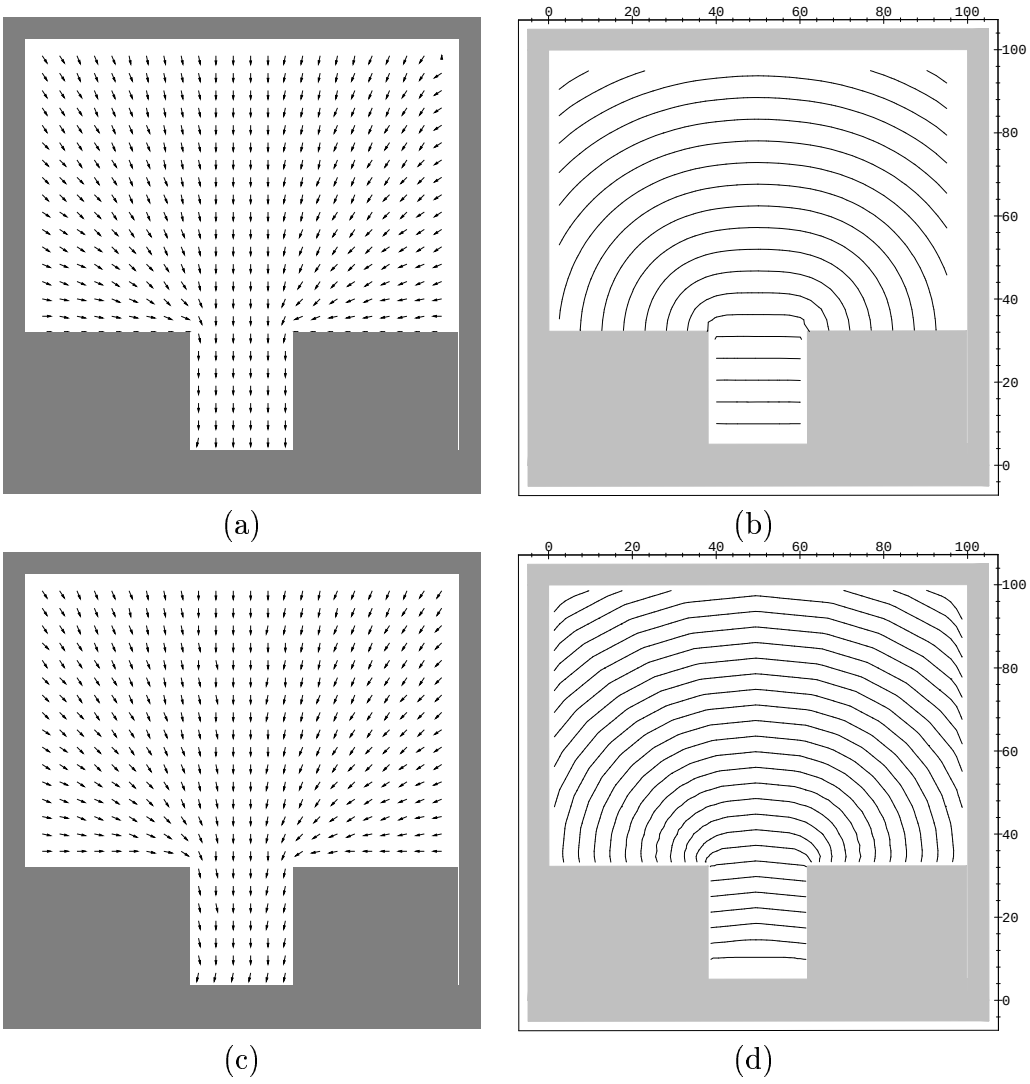


Figure 2.18 Optimal strategies and performance preimages for the peg-in-hole problem under probabilistic control uncertainty (shown in (a) and (b)) and nondeterministic control uncertainty (shown in (c) and (d)).

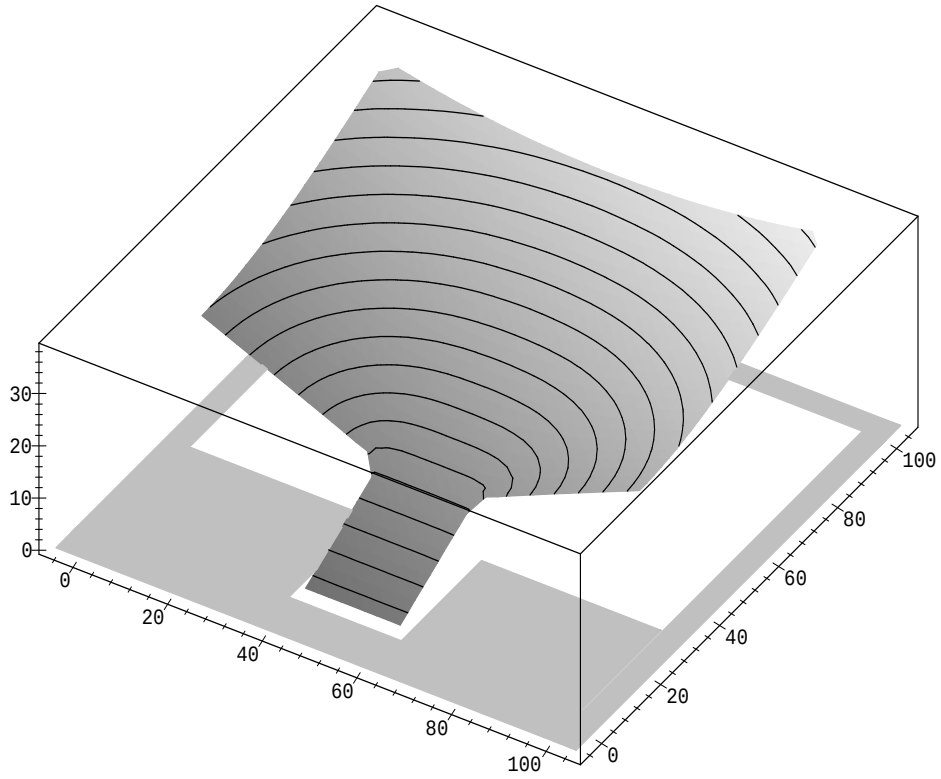


Figure 2.19 A plot of $\bar{L}^*$ with probabilistic uncertainty and perfect information.

Figures 2.18(c) and 2.18(d) show computed results obtained under nondeterministic uncertainty with perfect state information. The isoperformance curves are closer together because worst-case analysis causes the computed loss to be greater.

Figure 2.21 shows several more computed optimal strategies for probabilistic uncertainty with perfect state information. We assume for each of these examples that $\|v\|\Delta t = 3$ and $\epsilon_\theta = 48.8°$. Figure 2.22 shows a three-dimensional rendering of the cost-to-go function, along with the preimages for the example in Figures 2.21(d) through 2.21(f). Figure 2.23 shows the forward projection for the example in Figures 2.21(g) through 2.21(i).

Figure 2.24 shows computed optimal strategies for probabilistic uncertainty with imperfect state information. We assume for each of these examples that $\|v\|\Delta t = 3$ and $\epsilon_\theta = 48.8°$. We used the sensing model from Section 2.3.5, and let $\epsilon_p = 5$ and $\epsilon_f = 0$.

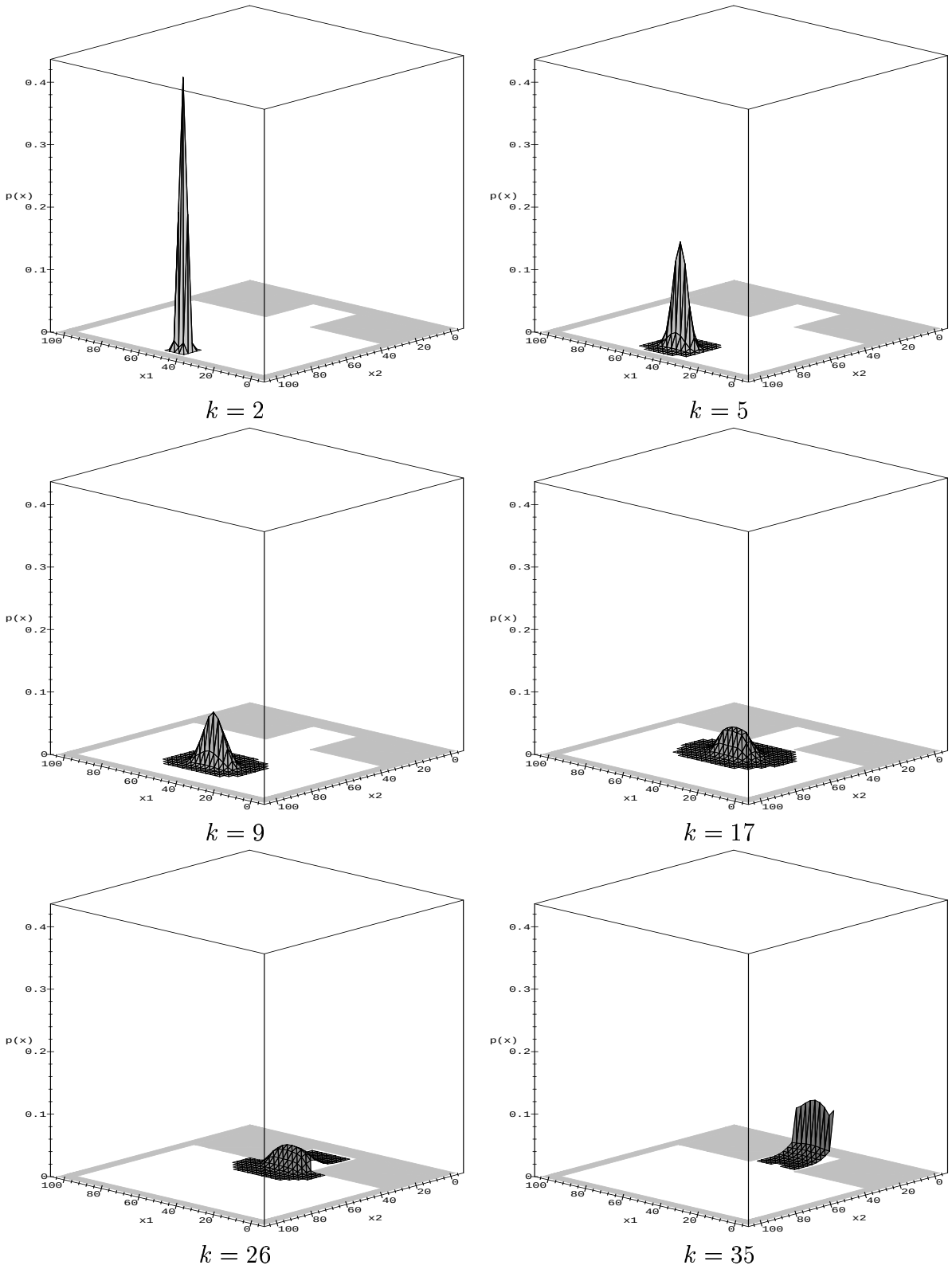


Figure 2.20 The forward projection for the optimal strategy under perfect state information.

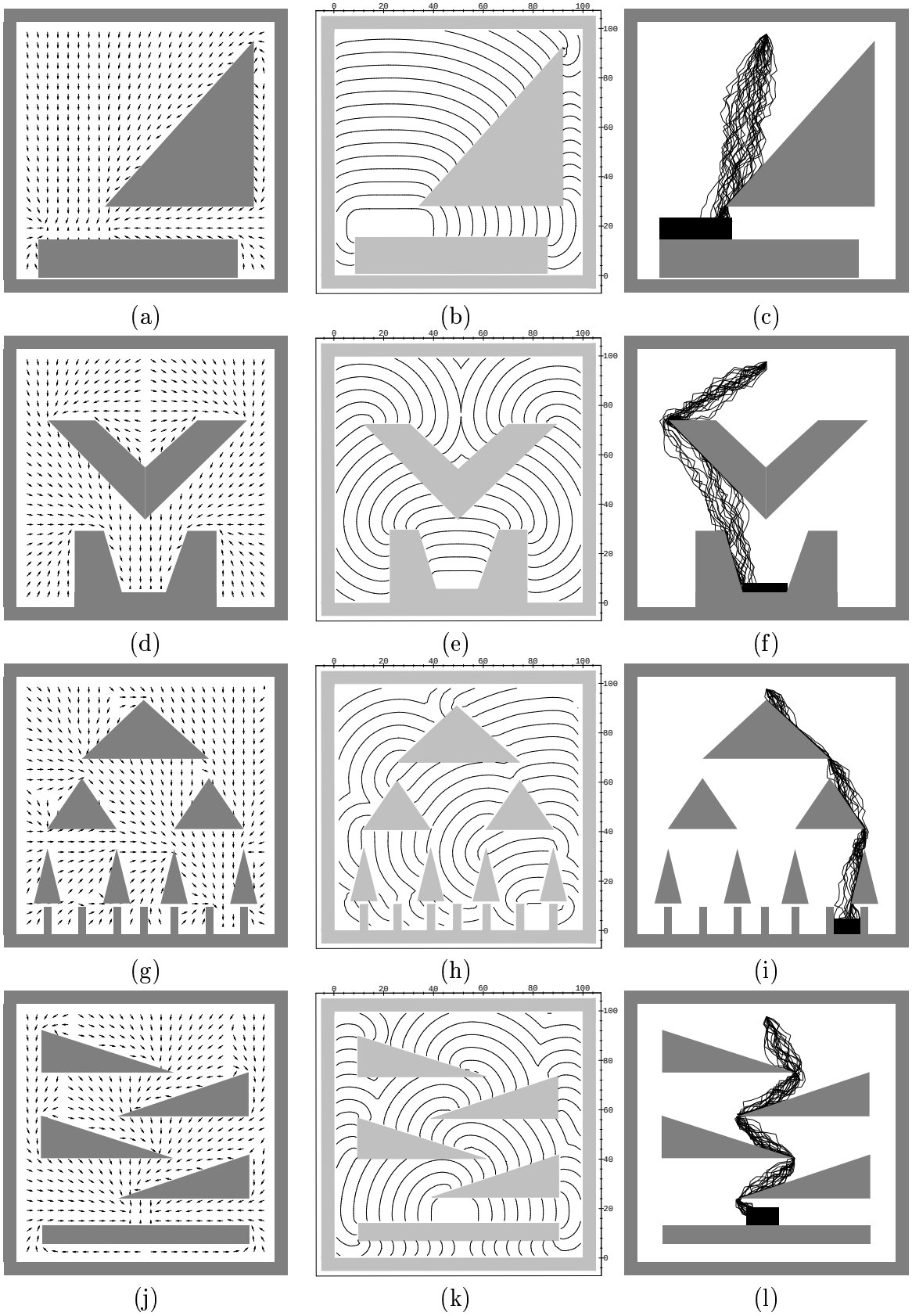


Figure 2.21 Several examples that were computed under probablistic uncertainty and perfect state information.

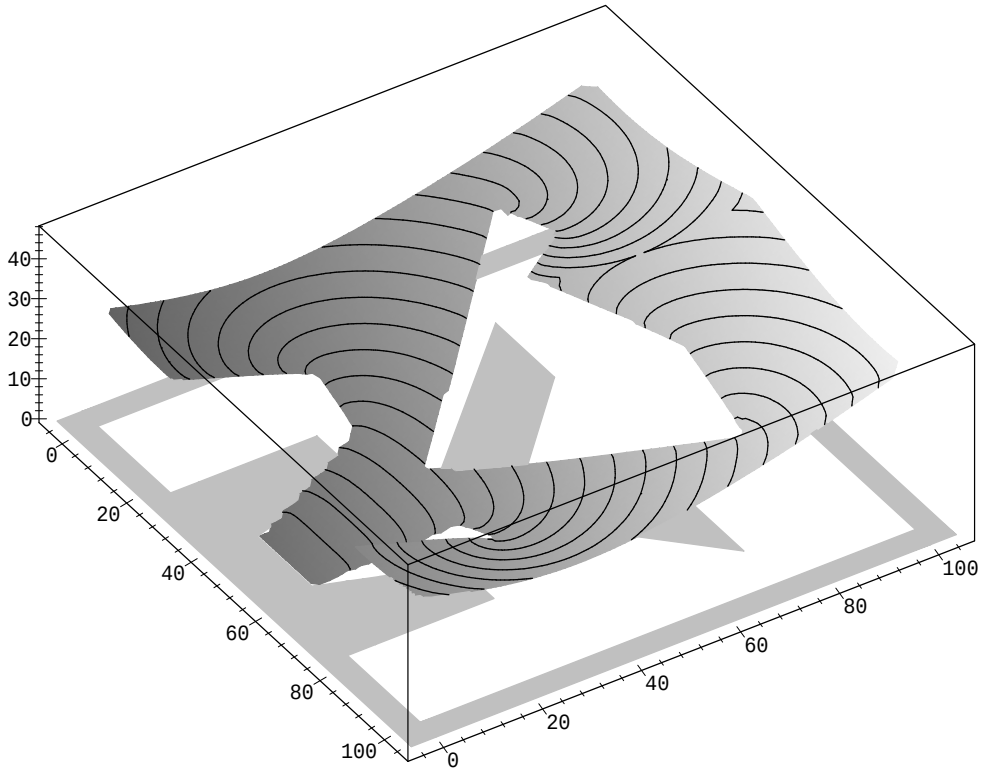


Figure 2.22 A plot of $\bar{L}^*$ under probabilistic uncertainty.

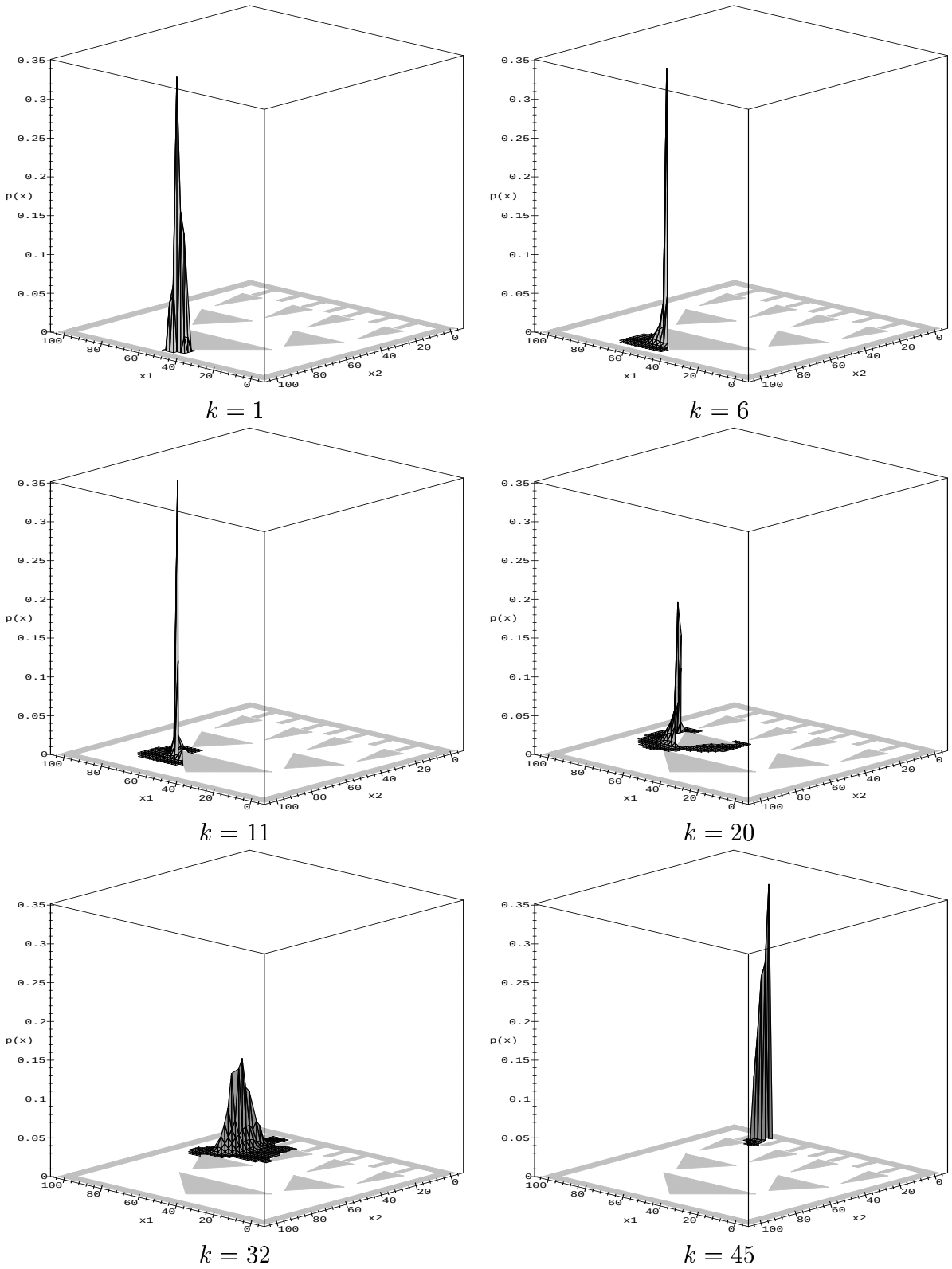


Figure 2.23 The forward projection for the optimal strategy under imperfect state information.

Without perfect sensing, the expected time to reach the goal increases, which causes the isoperformance curves to be closer together. In addition, the sample paths under the implementation of the optimal strategies involve more variations.

The strategy representation and the isoperformance curves in Figure 2.24 do not align completely with the obstacles in the workspace because the optimal actions and isoperformance curves are defined in the information space. For these examples, the information space is represented by the set of possible sensor values. Sensed force values are not shown in the figures.

Figure 2.25 shows computed optimal strategies for nondeterministic uncertainty with imperfect state information. A solution strategy could not be found using the same uncertainty models as for the probabilistic case. This occurs because worst-case analysis eliminates the consideration of many reasonable strategies, as mentioned in Section 2.2. We therefore use $\|v\|\Delta t = 10$, $\epsilon_\theta = 0.8$, $\epsilon_p = 2.5$, and $\epsilon_f = 0$. The isoperformance curves are shown for every 30 units of loss. Figure 2.26 shows $\bar{L}^*$ for the example in Figures 2.25(a) and 2.25(b).

2.7 Discussion

In this section we briefly discuss some aspects of the current approach and future directions that could be taken with this research.

Incorporation of additional uncertainties In this chapter we have considered control sensing and control uncertainty; however, in general other forms of uncertainty may exist. For instance, Donald has considered the development of strategies that are robust to errors in the model of an obstacle [48]. By parameterizing the uncertainty in the model

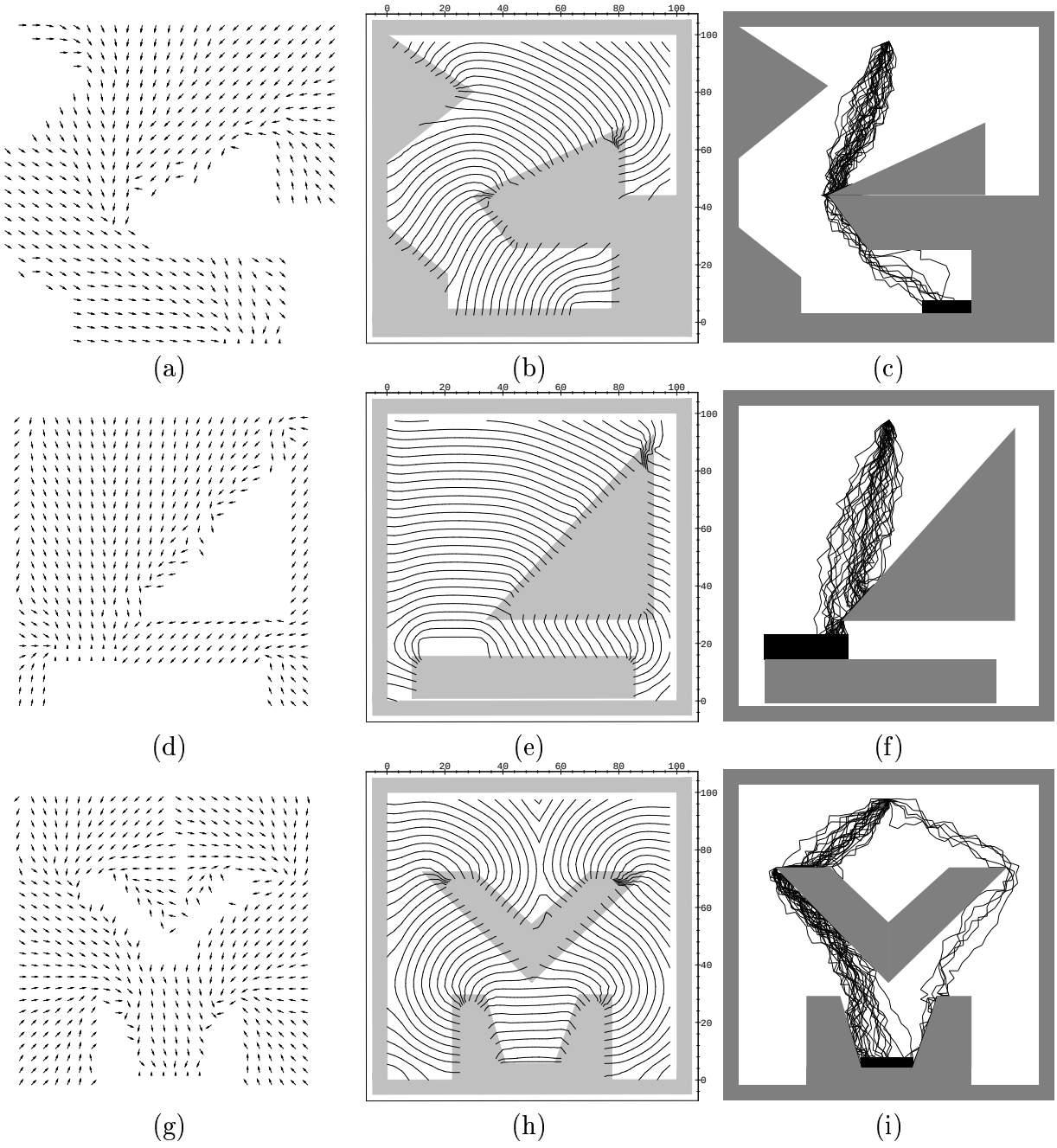


Figure 2.24 Examples that were computed under probablistic uncertainty and imperfect state information.

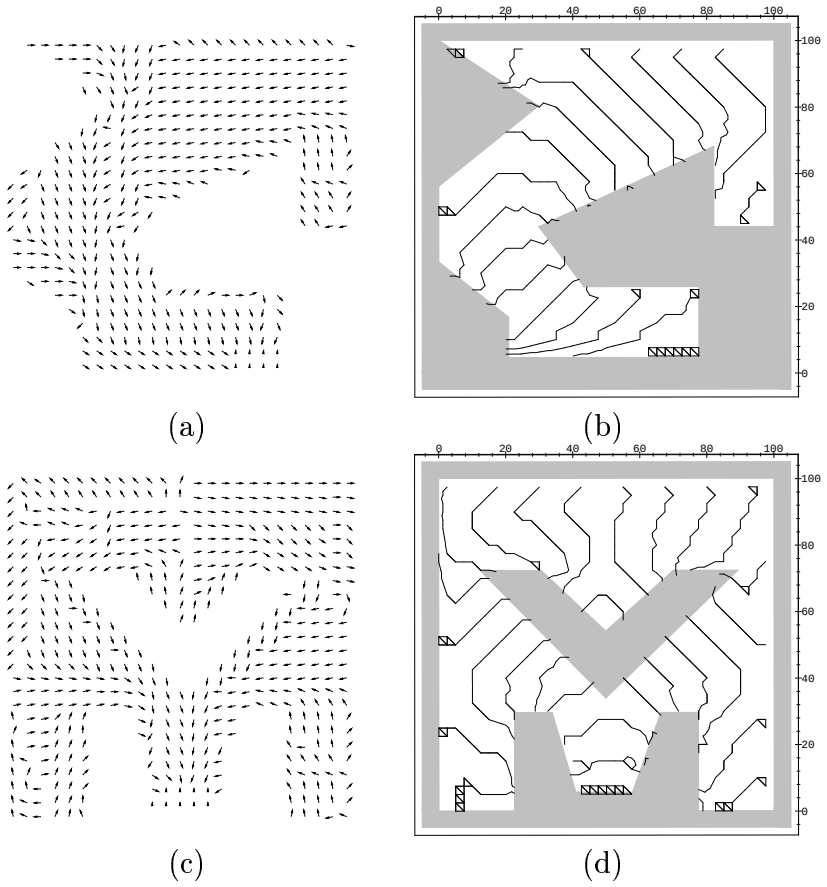


Figure 2.25 Examples that were computed under nondeterministic uncertainty and imperfect state information.

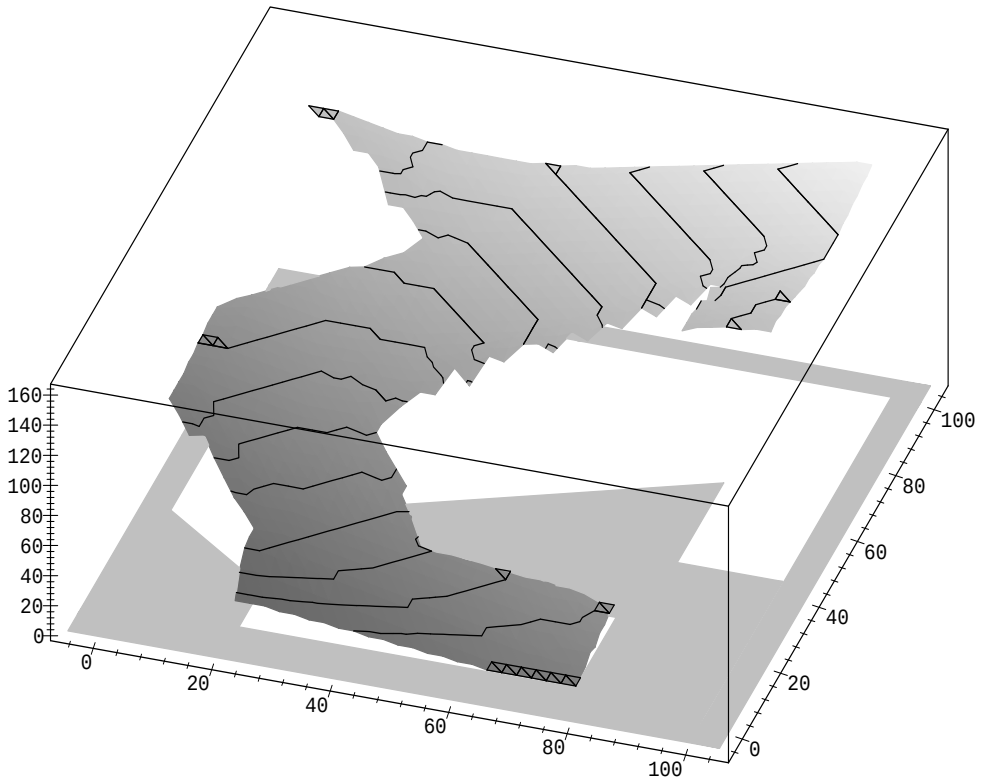


Figure 2.26 A plot of $\bar{L}^*$ under nondeterministic uncertainty and imperfect state information.

and performing worst-case analysis, a motion plan can be determined that overcomes this form of uncertainty. Recall from Section 1.2.2 that uncertainties in environment sensing and environment predictability are two other important sources of uncertainty that can affect a motion plan. These forms of uncertainty can be modeled by augmenting the state space to include information about the environment; this is the topic of Chapter 3.

Randomized actions In work by Erdmann [57], [58], [63], useful manipulation planning methods were developed around randomizing the actions of the robot. In game theory, a randomized strategy is referred to as a *mixed strategy*, as mentioned in Section 1.4.1. It is important to note that in the method presented in this chapter, the robot strategy is deterministic (or pure), even though the execution of the strategy can be considered as a random process (under probabilistic uncertainty). Erdmann has argued

that two important benefits result from using randomized strategies: (1) robustness with respect incorrect models can be obtained, and (2) multiple attempts can be made to solve a task, instead of requiring a guaranteed solution.

By conditioning our strategies on state feedback or information feedback, the robot is capable of making multiple attempts to solve a task. In a motion planning context, one can imagine a robot that attempts to execute a motion plan, reports failure, and then replans to make another attempt; this behavior is exhibited in the error detection and recovery strategies in [47], [48], [49]. An "attempt" is not as distinct in our approach, however, because the robot responds dynamically to its information. Rather than responding by determining a new strategy, the response corresponds to the optimal behavior that was determined through global analysis of the motion planning problem and its uncertainties.

Robustness with respect to incorrect modeling represents a useful feature, which has not been considered by our approach thus far. In the approach that we present, the assumption is made that the models are correct. Under nondeterministic uncertainty, the correctness of the model can become critical, because it then becomes impossible to "guarantee" a particular loss (unless the model truly represents an upper bound on the uncertainty). Under probabilistic uncertainty, the effect of modeling errors appears to be less drastic. One difficulty with introducing randomization is that it can arbitrarily increase the loss require to achieve the goal, even though robustness is strengthened. In the limiting case, pure Brownian motion can be executed. This achieves the goal while making essentially no modeling assumptions, but the loss can be extremely high. It remains to be seen whether randomized actions can be incorporated into our approach to provide a reasonable tradeoff between the distance of a strategy from optimality and the potential incorrectness of the models.

Displacement sensor measurements In some robotic systems, it may be possible to obtain sensor readings that estimate a new configuration of the robot with respect to its old configuration, which may generally be unknown. In this case, the displacement that occurs during time Δt is estimated. Sensors of this type have been considered previously for motion planning under uncertainty in [142]. We briefly describe how this type of sensor observation can be incorporated.

Assume that a displacement sensor reading, z_k, occurs after an action u_k has been executed at stage k. Suppose a *displacement observation equation*,

$$z_k = g(x_{k+1}, x_k, u_k, \theta_k^d), \tag{2.57}$$

is given in addition to the observation equation (2.5). The information space can be extended to include displacement observations, $\{z_1, z_2, \ldots, z_K\}$.

Under nondeterministic uncertainty,

$$F_k'(z_k, x_k, u_k) = \{x_{k+1} \in X | z_k = g(x_{k+1}, x_k, u_k, \theta_k^d), \theta_k^d \in \Theta^d\} \tag{2.58}$$

represents the possible next states after z_k has been observed. However, due to partial predictability in the state transition equation, it is also known that $x_{k+1} \in F_{k+1}(x_k, u_k)$, as given in (2.3). Therefore, we replace $F_{k+1}(x_k, u_k)$ by

$$F_k(z_k, x_k, u_k) = F_k'(z_k, x_k, u_k) \cap F_{k+1}(x_k, u_k) \tag{2.59}$$

in such expressions as those in Section 2.4.1.2 to incorporate displacement sensor observations.

Under probabilistic uncertainty, $p(z_k|x_{k+1}, x_k, u_k)$ represents a density that is derived from $p(\theta_k^d)$. However, due to partial predictability in the state transition equation, $p(x_{k+1}|x_k, u_k)$ was already given. We use Bayes' rule to combine the displacement observation with the partial predictability of the state transition. Therefore, we replace

$p(x_{k+1}|x_k, u_k)$ with

$$p(x_{k+1}|z_k, x_k, u_k) = \frac{p(z_k|x_{k+1}, x_k, u_k)p(x_{k+1}|x_k, u_k)}{\int p(z_k|x_{k+1}, x_k, u_k)p(x_{k+1}|x_k, u_k)dx_{k+1}} \tag{2.60}$$

to incorporate displacement sensor observations.

Hierarchical strategies In the methods developed in this chapter, the robot executes a fixed command at each Δt. In traditional preimage planning, however, a fixed action is executed until the termination condition is met. If the goal is not yet reached, another action is executed. In general, a sequence of fixed actions with termination conditions is executed until the goal is reached.

Recall that the performance preimage can be used to evaluate a particular strategy. One interesting approach would be to implement preimage backchaining and subgoals by performance preimages. We can define G_1 as a subgoal for a larger problem, and define a g and TC that achieves G_1 in a satisfactory way. The resulting posterior density $p(x_{K+1})$ would be used as the initial information state for the achievement of a second goal, G_2. We can consider abstract actions of the form $\{G_i, \gamma\}$ that attempt to achieve some original goal. Backchaining from G under explicit performance measures and a given set of choices for abstract actions is another form of dynamic programming. The relationship between standard preimage planning and dynamic programming is discussed in [58]. The reason for considering abstract actions and subgoals is the hope that a simple set of abstract actions exists that can be composed to provide quick and efficient solutions for a wide class of problems (as was the case with backprojection planning [62]).

Determining accurate uncertainty models The flexibility of our approach permits the use of a variety models for sensing and control uncertainty. In many previous approaches, the results were strongly dependent on the particular model chosen. For example, worst-case analysis in the backchaining approach has often used a bounded disk

for position uncertainty and bounded angular error for control uncertainty. With our approach, one important area of future research is to develop models that accurately reflect the uncertainty involved in a particular manipulation task. This is particularly true for the case of probabilistic uncertainty. The densities hold a large amount of expressive power; however, simple models are often chosen to obtain reasonable results. Within our approach, different uncertainty models can be substituted, and through simulations or repeated execution trials, better uncertainty models could be developed for a particular context. This direction of research was also advocated in [27], to determine valid error distributions for the computations of appropriate probabilistic backprojections.

Sampling Issues One important issue that has received little attention in manipulation planning literature is the sampling rates available for sensing and control. In the typical preimage planning formulation, the robot is allowed to issue a new command at any point in time, implying continuous-time controllability of the robot. The robot command is only changed, however, during the few occurrences of meeting the termination condition. In this chapter, we have assumed sampling rates that essentially approximate continuous-time control and sensing. By allowing the motion command to change at any discrete stage, we obtain a significant amount of control over the robot in the face of uncertainty. One useful approach might be to consider a much lower sampling rate. This models the situation in which fine motions are performed before additional sensing, or before a new control input can be applied. This seems to appropriately reflect a situation in which the planning workspace is very small, such as in a part-mating operation.

2.8 Conclusion

We presented a flexible approach for manipulation planning under uncertainty in which motion strategies are selected to optimize a loss functional. We have indicated through the discussion and simulation experiments that the efficiency of a robot motion strategy is crucial in planning under uncertainty. We have developed *performance preimages* as a useful concept for evaluating motion strategies. This work identifies *termination criteria* with *optimal stopping problems* from optimal control theory, and allows the incorporation of a termination condition into the optimal strategy. We apply information space concepts from stochastic control and dynamic game theory to incorporating history into a motion strategy with uncertainty in sensing. We additionally provide a computational approach that numerically determines optimal motion strategies under a wide class of performance functionals by applying the dynamic programming principle to approximate stationary cost-to-go functions, and illustrate the concepts through computed examples. One of the most important directions for future research will be to investigate different methods of approximately representing the information space.

CHAPTER 3

MOTION PLANNING UNDER ENVIRONMENT UNCERTAINTIES

3.1 Introduction

This chapter develops concepts for motion planning in uncertain environments. In terms of the categorization that was discussed in Section 1.2.2, this chapter primarily emphasizes uncertainty in environment predictability, with some additional consideration of uncertainty in environment sensing. Recall that environment-sensing uncertainty implies that the robot might not have perfect or complete information about its environment at a given instant in time, and that environment-predictability uncertainty implies that the robot might not be able to predict the environment at a future time. Much of the work described in this chapter has also appeared in [116], [117].

Section 3.2 provides motivation and background for the concepts in this chapter by discussing example problems and related literature. In particular, a set of problems that involve a finite number of environments is discussed. Section 3.3 presents the mathematical models used to define the robot and its interactions with the environment. A stochastic optimal control problem is defined by combining the motions of a robot that has perfect control and sensing with an environment modeled with a Markov process.

Uncertainty is thus represented probabilistically throughout most of this chapter. Section 3.4 describes the computational approach for determining optimal strategies, which is based on dynamic programming. Section 3.5 presents several computed examples. Section 3.6 presents an extension of the basic methods to a problem in which a robot must deliver parts from source regions to destination regions, when the delivery requests are only partially predictable. This could occur, for example, as part of an automoted assembly or manufacturing system [172]. Several computed examples are presented for this extension, for both rigid robots and manipulators. Section 3.7 discusses several additional extensions to the basic method, including the use of nondeterministic uncertainty, as opposed to probabilistic uncertainty. Section 3.8 presents conclusions.

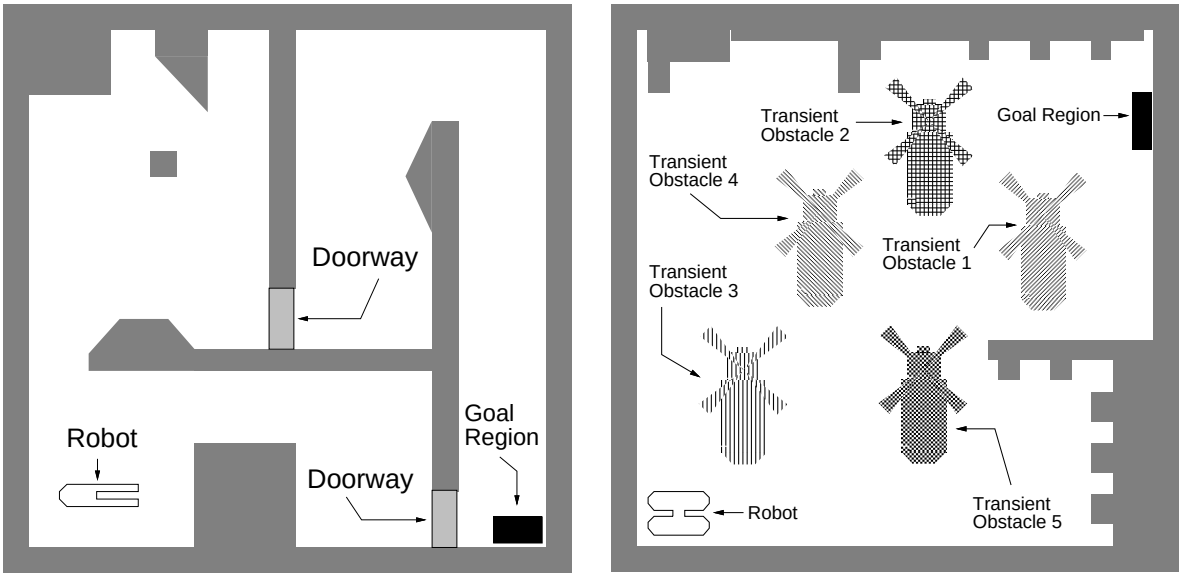
3.2 Background and Motivation

Uncertainty in environment predictability has received less attention than the other sources discussed in Section 1.2.2. Related past work is mainly concerned with local collision avoidance (e.g., [204]) or incorporating unexpected moving objects locally into the updating of the motion plan. Thus, many of the predictive aspects of the changing environment are not utilized, and the approach is similar to the incorporation of environment sensing uncertainty. For example, in an artificial potential field approach it is possible to incorporate moving obstacles with unknown trajectories using on-line sensor data without considering future changes in the environment [11], [156], [165], [181], [182]. In [42], an incremental planning scheme is presented for collision avoidance with unknown moving obstacles that are restricted to piecewise-linear trajectories.

We next discuss several examples that illustrate some important aspects of uncertainty in environment predictability. These examples will be referred to again when computed results are presented in Section 3.5. The first two examples represent problems in which

the free configuration space changes over time and is not completely predictable (see Figure 3.1). First consider the problem in Figure 3.1(a). While the robot is moving, doors over which the robot has no control could close or open. Suppose that the robot has bounded velocity and wishes to reach a goal region in a minimal amount of time. Should the robot always try to move through the lower door? Should it adjust its path depending on which of the two doors are open? What happens when the robot is moving toward a door and the door closes? Should it just wait for the door to open or should it head toward the other door if that door is open? One would like to define a formal basis for deciding on the best actions to take, given that the robot does not know exactly when certain changes will occur in the workspace. A similar type of example is shown in Figure 3.1(b), which contains partially-predictable obstacles that can instantaneously appear or disappear at future times. Completely predictable obstacles of this type were considered in [68] and termed *transient obstacles*. In this chapter, we consider problems such as these, which involve a changing, partially predictable configuration space. By introducing probabilistic models of the environment, we can determine a strategy for the robot that minimizes the expected time to reach the goal.

A problem such as that in Figure 3.1(a) can be described with a state space that is generated by a discrete set of collision-free configuration spaces. At a given time, the robot is in one of these spaces; changes in the environment can cause the robot to move to a different collision-free configuration space. These potential transitions must be taken into account in a specification of a motion plan. In our approach, a state-feedback motion strategy is designed, in which motion commands are conditioned on both the environment and the configuration. Replanning is not required when the environment changes, because the robot responds appropriately for different regions of the state space during execution. In addition, a state-feedback controller is advantageous, since it will typically be robust to small modeling errors that can develop during execution. A motion strategy is selected



(a) (b)

Figure 3.1 A changing environment in which the workspace changes over time, by: (a) the opening and closing of "doors," and (b) appearance and disappearance of "transient" obstacles.

that optimizes (in an expected sense) a loss functional that can incorporate a variety items, such as time or distance.

Because the robot cannot completely predict changes that may occur in the environment, a model of uncertainty must be expressed. One approach is to consider the changes as being generated from a stochastic process. The corresponding probabilities could in practice be estimated by observing many samples of such changes in a particular workspace. This *partial* knowledge of the changing environment can then be exploited by a motion planning strategy to improve the expected performance. For the above motion planning problem with doors, for example, one suitable means of modeling the opening of a door could be in terms of a Poisson process with arrival frequency λ_o, so that the time until a closed door opens is an exponential random variable with mean λ_o. Similarly, the closing of a door, given that it is open, could be modeled as a Poisson process with arrival

frequency λ_c. The Poisson process is a reasonable choice for many problems because it captures several realistic properties of a changing environment: (1) the probabilities that a door will open in two nonoverlapping time intervals are independent of each other; (2) the probability that a door will open in an interval is proportional to the length of the interval, which implies that events are uniformly distributed in time, and thus do not favor one epoch of time over another; and (3) the probability that a door will open in an interval becomes arbitrarily small if the interval is made sufficiently small. These Poisson processes are special cases of a Markov process, which has much greater expressive power, as discussed in Sections 3.3 and 3.5.

The problems in Figure 3.1 involve a changing configuration space; however, we can allow more general ways in which the environment can change. In some situations, it may be appropriate to *not consider* many individual moving obstacles in the environment. Instead, a cost can be assigned that corresponds to the amount of risk for traversing a region that could have moving obstacles. This leads to a set of problems that we refer to as *hazardous region and shelter problems.* The loss could, for example, directly correspond to the risk involved in traversing a hazardous region. In a similar context, this has been referred to as a *weighted region problem* [134], [163]. Assigning a cost associated with the traversal of a region provides a way of dealing with the complexity of motion planning in an environment that has several moving obstacles, particularly when their motions are unknown. Similar treatment of dynamic environments in [170], [171] led to the idea of *shelters* (regions that have low cost of traversal) and *alarms* (events that cause the cost of traversing a region to change from low to high). The treatment considered in this dissertation, however, is substantially more general.

As an example of a hazardous region and shelter problem, consider the motion planning problem (shown in Figure 3.2), for a mobile robot in a factory floor in which there might be other moving robots, vehicles, or people in the corridors. The robot may receive

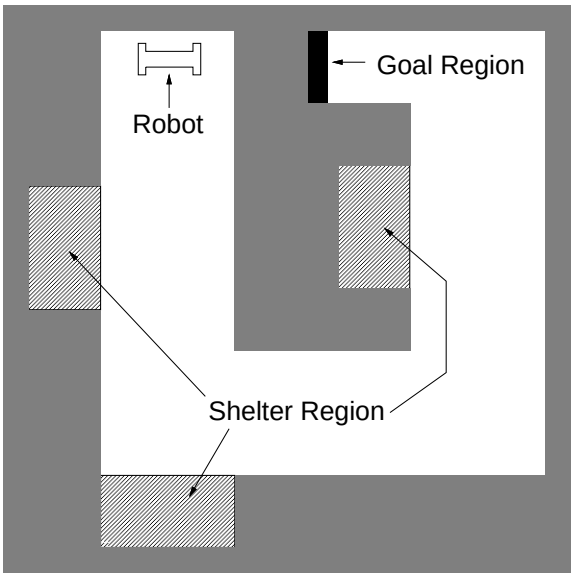


Figure 3.2 A problem that involves safe and hazardous regions in addition to obstacles.

information that other obstacles are in the corridor, and the only safe regions are the *shelters* along the sides of the corridor. By proper modeling of hazardous regions and shelters in the workspace, the need for explicitly considering multiple moving objects (as in [30], [61], [157], [187]) can be avoided, while implicitly factoring in the effect of moving obstacles in the determination of motion plans.

The previous problem provided an example that did not involve geometric changes in the configuration space, and yet represented a useful instance of planning under environment-predictability uncertainty. We will, therefore, generally consider the environment to assume different *modes*. The difference between two modes could, for example, correspond to the appearance of an obstacle or a hazardous condition. Environment modes will be formally defined in Section 3.3.1.

Both the changing configuration space problems and the hazardous region and shelter problems have one aspect in common: the robot was implicitly assumed to have no control over the changing environment. In general, however, many interesting problems can be defined in which the robot can *partially* control the environment. The robot might, for

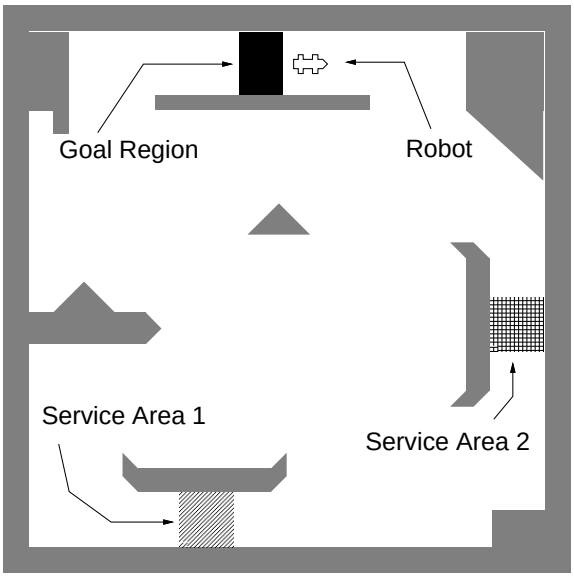


Figure 3.3 A problem of processing random service requests in the workspace.

example, be able to change the mode at a particular location in the configuration space (imagine the robot closing a door while in its proximity). Situations in which the robot can interact with the partially-predictable environment can be considered as servicing problems. Using the concept of modes, problems that involve interactions between the robot and the environment might not involve changing the free configuration space. Consider the example shown in Figure 3.3, in which there is a mobile robot capable of performing two different kinds of services, at particular locations in the workspace. Requests to perform these services might occur at any time and are not completely predictable. The robot moves at a fixed speed and must satisfy the request by appearing in the appropriate *service area*. The goal can be to minimize a combination of the time to reach a goal region and a penalty for the time that requests are unprocessed. A problem can also be defined without a goal region, and the robot simply minimizes the time that requests are outstanding. Section 3.6 presents analysis and computed examples for a problem that involves a rigid robot or manipulator that moves parts in a workspace

from partially-predictable source regions to partially-predictable destination regions; this represents another type of servicing problem.

3.3 Mathematical Modeling

In this section, we construct a stochastic optimal control formulation that models the problems discussed in Section 3.2. Section 3.3.1 defines a Markov process that models the environment and a state space that encodes both robot configurations and the environment mode. Section 3.3.2 defines how state transitions occur and the concept of a strategy in this context. Section 3.3.3 introduces the concept of dynamic regions in the robot's configuration space. These regions explicitly define the interaction that occurs between the environment and the robot. The loss functionals in this chapter rely on a precise characterization of these regions.

3.3.1 The environment process

Recall from Section 1.2.1 that for geometric motion planning problems without uncertainty, the space of possible situations that can occur is sufficiently characterized by $\mathcal{C}_{free}$ (or by $\mathcal{C}_{valid}$, as in Chapter 2). Let E denote a set of *environment modes*. To uniquely identify all of the possible situations that can occur in our problem, we define a *state space* as the Cartesian product, $X = \mathcal{C}_{free} \times E$. Thus, for each environment mode there will be a new copy of $\mathcal{C}_{free}$. This is similar to the view taken in [47], in which the space for motion planning is a Cartesian product of $\mathcal{C}_{free}$ with a single parameter that characterizes a hole width for a peg-in-hole task.

As in Chapter 2, we use discrete-time representations by defining a set of *stages*, $\{1, \ldots, K\}$. The state at stage k is denoted by x_k, which simultaneously represents both a configuration $\mathbf{q}_k$ and an environment mode, e_k. The environment modes form a

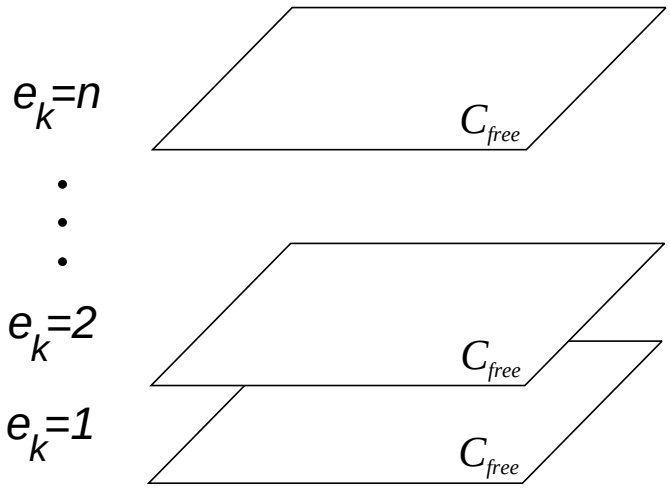


Figure 3.4 The environment modes can form a partition of X.

partition of the state space, X. Each time the environment mode changes, the robot is forced into a different layer of X (see Figure 3.4).

The changing of modes over time can be considered as the effect of nature on a state transition equation. An action, u_k, and *action space*, U, are defined as in Chapter 2. Let Θ^c represent a set of $|E|$ control actions for nature. Assume that there is no uncertainty in configuration predictability; therefore, nature affects only the environment mode. The next state can be expressed as $x_{k+1} = f_k(x_k, u_k, \theta_k^a)$. We use a probabilistic representation for the uncertainty; hence, $P(x_{k+1}|x_k, u_k)$ is a *state transition distribution* over a set of states that can be obtained. The environment mode transitions can, therefore, be considered as $P(e_{k+1}|x_k, u_k) = P(e_{k+1}|e_k, \mathbf{q}_k, u_k)$. In this case, the evolution of environment modes can be considered as a finite-state Markov process. Section 3.7.3 describes how the nondeterministic representation could alternatively be used.

We now present an example of a four-mode environment process that can model the problem from Figure 3.1(a). More general models and examples are presented in Section 3.5. We define the following four environment modes:

Mode	Interpretation
0	Door 1 open; Door 2 open
1	Door 1 closed; Door 2 open
2	Door 1 open; Door 2 closed
3	Door 1 closed; Door 2 closed

Each door is modeled with Poisson processes. Let λ denote a Poisson arrival rate. The density for the time of the first arrival is $p(t_a) = \lambda e^{-\lambda t_a}$. We denote the arrival rate of a door closing, given that it is open, as λ_c, and the arrival rate of a door opening, given that it is closed as λ_o.

Assume for this example that the two doors are governed by independent, identical Poisson processes. The probability that a closed door will open in time Δt is

$$P_{01} = \int_0^{\Delta t} \lambda_o e^{-\lambda_o t_a} dt_a = 1 - e^{-\lambda_o \Delta t}. \tag{3.1}$$

We use 0 to denote the open condition and 1 to denote the closed condition in P_{01} (indexed in the same order as a conditional probability). The probability that it will stay closed is $P_{11} = 1 - P_{01}$. For a door that is initially open, we similarly obtain $P_{10} = 1 - e^{-\lambda_c \Delta t}$, and $P_{00} = 1 - P_{10}$.

The environment transition probabilities can be generated by taking products of pairs of P_{00}, P_{01}, P_{10}, and P_{11}:

$$\begin{bmatrix} P_{00}^2 & P_{00}P_{10} & P_{10}P_{00} & P_{10}^2 \\ P_{00}P_{01} & P_{00}P_{11} & P_{10}P_{01} & P_{10}P_{11} \\ P_{01}P_{00} & P_{01}P_{10} & P_{11}P_{00} & P_{11}P_{10} \\ P_{01}^2 & P_{01}P_{11} & P_{11}P_{01} & P_{11}^2 \end{bmatrix}. \tag{3.2}$$

The $(i,j)^{th}$ element represents the probability of changing to mode i, given that the current mode is j.

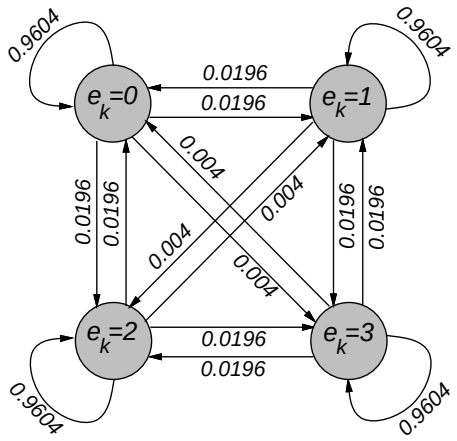


Figure 3.5 The environment process can be considered as a finite-state Markov process with state transition probabilities.

The four-mode process is depicted in Figure 3.5, in which we take $\lambda_o = \lambda_c = 0.10101354$ (approximately one expected arrival every ten seconds), and $\Delta t = 0.2$. This results in $P_{10} = P_{01} = 0.02$ and $P_{00} = P_{11} = 0.98$. Each arc in the graph represents one element of Equation (3.2).

3.3.2 Defining the robot behavior

We present a state transition distribution that applies to the case in which $\mathcal{C}_{free} \subseteq \Re^2$, and the robot is limited to translational motion. More complicated motions will be described with the examples in Section 3.5. Dynamic robot constraints could also be introduced; however, the state space would have to be expanded to include time derivatives of configuration. We define the action space as $U = [0, 2\pi) \cup \{\emptyset\}$. If $u_k \in [0, 2\pi)$, then $\mathcal{A}$ attempts to move a distance $\|v\|\Delta t$ toward a direction in $\mathcal{C}$, in which $\|v\|$ denotes some fixed speed for $\mathcal{A}$. If $u_k = \emptyset$, then the robot remains motionless.

Consider the case in which $x_k \in \mathcal{C}_{free}$ is at a distance of at least $\|v\|\Delta t$ from the obstacles. If $\mathcal{A}$ chooses action $u_k \neq \emptyset$ from state x_k, then[1]

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\|\Delta t \; cos(u_k) \\ x_k[2] + \|v\|\Delta t \; sin(u_k) \\ \\ e_{k+1} \end{bmatrix}, \tag{3.3}$$

in which the environment mode e_{k+1} is known to be sampled from $P(e_{k+1}|x_k, u_k)$. We can thus consider a finite-valued random variable X_{k+1} with corresponding distribution $P(x_{k+1}|x_k, u_k)$, which can be inferred from the given model. If $u_k = \emptyset$, then $x_k[1] = x_{k+1}[1]$ and $x_k[2] = x_{k+1}[2]$; however, e_{k+1} is not necessarily equal to e_k because the environment transition equation determines e_{k+1}. We prohibit the robot from considering actions that produce an obstacle collision in this case.

A *strategy at stage k* is a function $\gamma_k : X \rightarrow U$, and a *strategy*, γ denotes $\{\gamma_1, \ldots, \gamma_K\}$. Hence, this is a feedback controller with perfect state information. Stationarity is assumed as in Chapter 2. Section 3.7.2 presents a discussion of time-varying strategies, in which this assumption is relaxed. A *loss functional*, $L(x_1, ..., x_{K+1}, u_1, ..., u_K)$, is defined, and the goal of the planner is to determine an optimal strategy $\gamma^* = \{\gamma_1^*, \gamma_2^*, \ldots, \gamma_K^*\}$ that causes L to be minimized in an expected sense.

3.3.3 Defining loss with dynamic regions

This section discusses the key concepts that are used to model the effect of the environment on the robot. In particular, costs that appear in a loss functional directly depend on dynamic regions in the state space. If the robot enters a particular dynamic region, the amount of loss received might increase or decrease. For example, a dynamic region might correspond to the robot's collision with a closed door, which would incur a very high loss.

[1]We use the notation $x_k[i]$ to refer to the i^{th} element of the vector x_k.

We will define dynamic regions in the workspace $\mathcal{W}$, and subsequently discuss how these regions are mapped into the state space X. In addition to static obstacles, let $\mathcal{W}$ contain a set of m *dynamic regions*, denoted by $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$. Each dynamic region is a subset of $\mathcal{W}$, and pairs of dynamic regions are not necessarily disjoint.

A dynamic region $\mathcal{D}_i$ in $\mathcal{W}$ can map into the region $\mathcal{CD}_i^c \subset \mathcal{C}_{free}$, which is given by (see Figure 3.6):

$$\mathcal{CD}_i^c = \{\mathbf{q} \in \mathcal{C}_{free} | \ \mathcal{A}(\mathbf{q}) \cap \mathcal{D}_i \neq \emptyset\}. \tag{3.4}$$

We call $\mathcal{CD}_i^c$ a *contact (dynamic) $\mathcal{C}$-region.* This yields configurations in which the robot is in contact with $\mathcal{D}_i$. A contact $\mathcal{C}$-region is useful for such problems as that in Figure 3.1(a), in which contact with an obstacle must be determined.

In some situations, we will be interested in determining whether the robot is completely contained in some $\mathcal{D}_i$. For instance, in the example in Figure 3.2, the robot is only considered safe if it is completely inside the shelter region. For this situation, a dynamic region $\mathcal{D}_i$ in $\mathcal{W}$ maps into the region $\mathcal{CD}_i^e \subset \mathcal{C}_{free}$, which is given by (see Figure 3.7):

$$\mathcal{CD}_i^e = \{\mathbf{q} \in \mathcal{C}_{free} | \ \mathcal{A}(\mathbf{q}) \subseteq \mathcal{D}_i\}. \tag{3.5}$$

We now want to map the dynamic regions into the state space, because the loss functional depends on the state trajectory (see Figure 3.8). The dynamic regions have been mapped into $\mathcal{C}_{free}$; therefore, the mapping into X can be considered as lifting the contact $\mathcal{C}$-region (or enclosure $\mathcal{C}$-region) into different layers of X. We want the dynamic region to only influence the robot only at certain layers. With the example in Figure 3.1(a), we want the dynamic region to exist in X only in environment modes that correspond to the door being closed. In modes for which the door is open, the door should not interfere with the robot. For each $i \in \{1, \ldots, m\}$, we select a subset E_i of environment states $E_i \subseteq E$.

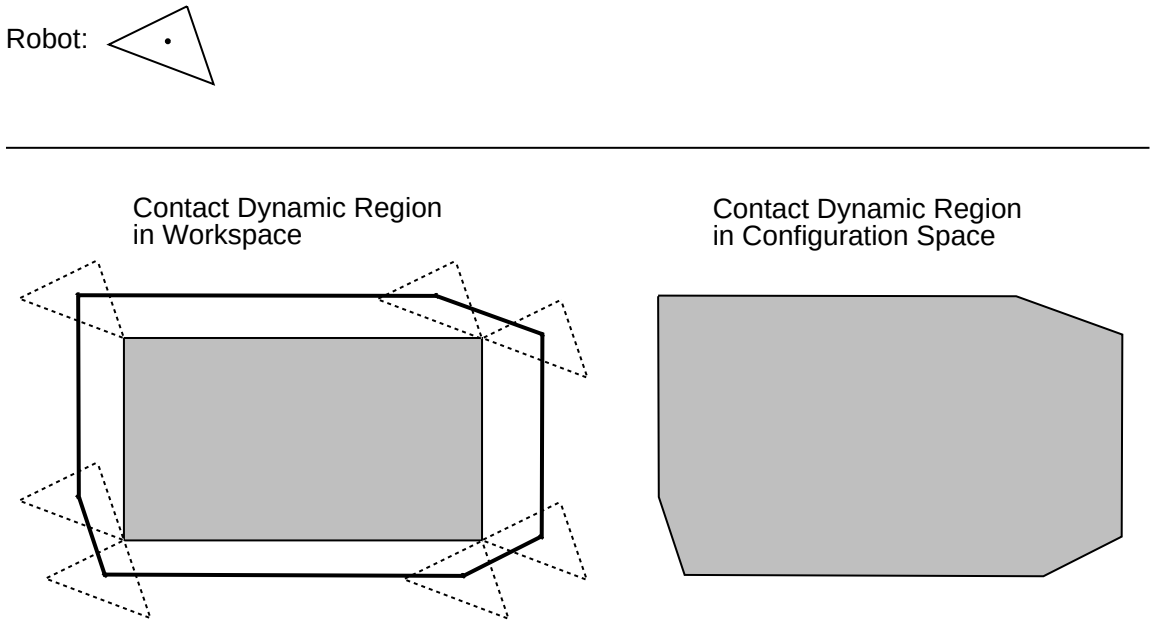


Figure 3.6 A contact dynamic region.

We can represent a state $x \in X$ by $(\mathbf{q}, e)$, in which $\mathbf{q} \in \mathcal{C}_{free}$ and $e \in E$. If $\mathcal{D}_i$ is a contact dynamic region, then we define

$$X_i = \{(\mathbf{q}, e) \in X \mid q \in \mathcal{CD}_i^c \text{ and } e \in E_i\}. \tag{3.6}$$

Alternatively, if $\mathcal{D}_i$ is an enclosure dynamic region, then we define

$$X_i = \{(\mathbf{q}, e) \in X \mid q \in \mathcal{CD}_i^e \text{ and } e \in E_i\}. \tag{3.7}$$

We call X_i the i^{th} *dynamic X-region*. Each X_i may be formed from either a contact or enclosure dynamic region.

We now define a *goal region* as a special kind of dynamic region (in addition to $\mathcal{D}_i, i \in \{1, \ldots, m\}$). We first define a subset $G \subseteq \mathcal{W}$ as a goal region in the workspace. We next consider G as a *contact goal region* (or *enclosure goal region*), and apply (3.4) (or (3.5)) with $\mathcal{D}_i = G$ to obtain the goal region in configuration space. A subset $E_g \subseteq E$ is selected, and we obtain X_G by applying (3.6) (or (3.7)). The termination condition for a given motion planning problem will be to bring the system to any state in X_G.

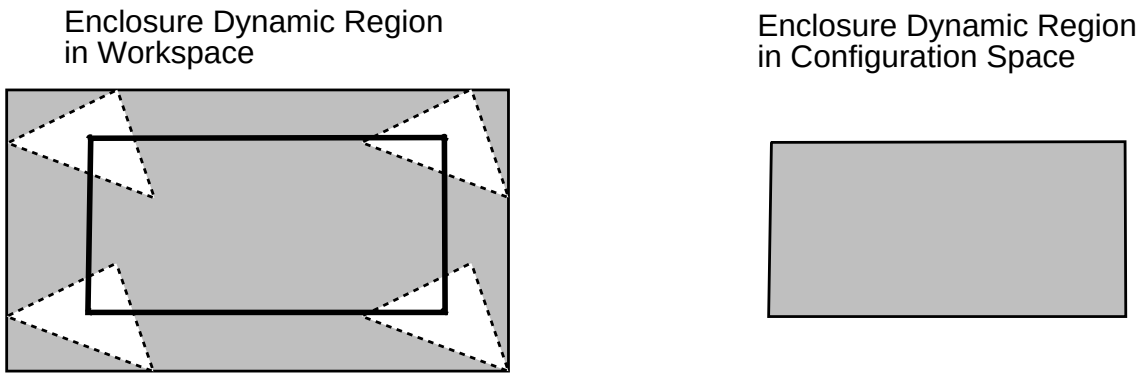


Figure 3.7 An enclosure dynamic region.

We assume that a loss functional is of the additive form, as in Equation (1.7), which was discussed in Section 2.3.4. For a given set A, let I_A denote its characteristic function: $I_A(a) = 1$ if $a \in A$, and $I_A(a) = 0$ otherwise. The term l_k is defined as

$$l_k(x_k, u_k) = \begin{cases} 0 & \text{If } x_k \in X_G \\ \\ c_u + \sum_{i=1}^{m} [c_i I_{X_i}(x_k) + c_i' I_{X_i^c}(x_k)] & \text{Otherwise} \end{cases} . \tag{3.8}$$

The constant $c_u \geq 0$ corresponds to the cost for choosing an action.

The constant $c_i \geq 0$ is the penalty added if $x_k \in X_i$. The constant $c_i' \geq 0$ is the penalty added if $x_k \notin X_i$. In (3.8), X_i^c denotes $X \setminus X_i$. For the case of a changing configuration space, for instance, these constants could become $c_i = \infty$, to indicate that a collision has occurred, and $c_i' = 0$ otherwise. The specific loss functionals for applications are presented in Section 3.5.

The term l_{K+1} is defined as

$$l_{K+1}(x_{K+1}) = c_f I_{X_G^c}(x_{K+1}), \tag{3.9}$$

in which X_G^c denotes $X \setminus X_G$. The constant c_f can be considered as the cost of failure. We typically consider $c_f = \infty$, but can also associate a finite cost with failure.

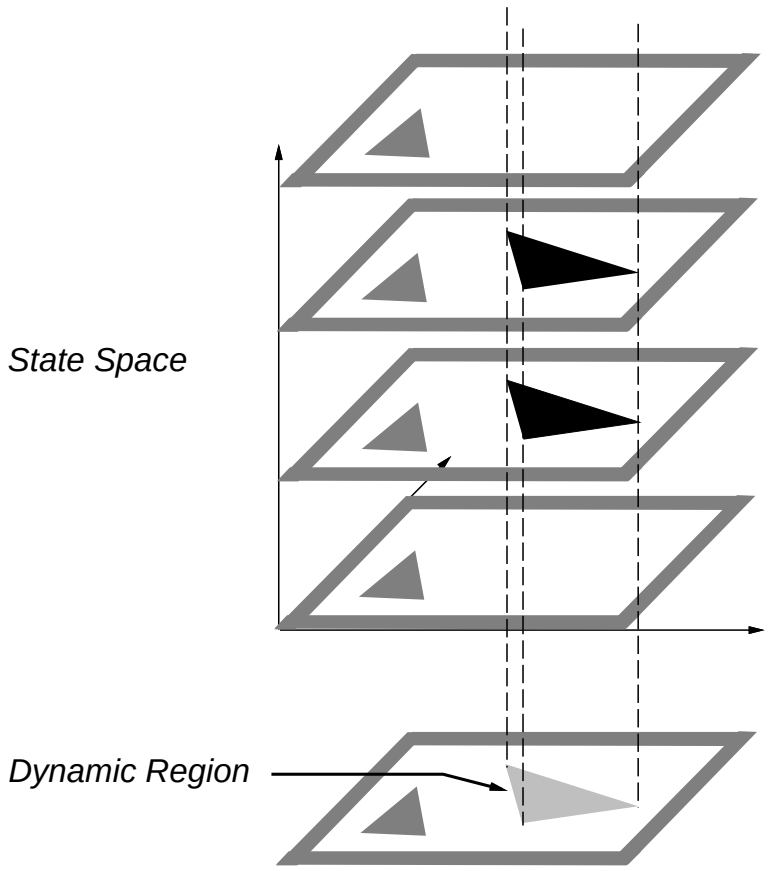


Figure 3.8 Dynamic regions are lifted into the state space.

3.4 Determining Optimal Strategies

One of the primary advantages of our approach is that a straightforward computation procedure can be used to determine optimal strategies. In Section 3.4.1, we show how the principle of optimality can be applied to our problem to obtain solutions through dynamic programming. Section 3.4.2 briefly discusses computational issues.

3.4.1 Applying the principle of optimality

Suppose that for some k, the optimal strategy is known for each stage $i \in \{k, \ldots, K\}$. The expected loss obtained by starting from stage k and implementing the portion of the

optimal strategy $\{\gamma_k^*, \ldots, \gamma_K^*\}$ can be represented as a cost-to-go function,

$$\bar{L}_k^*(x_k) = E\left\{\sum_{i=k}^{K} l_i(x_i, \gamma_i^*(x_i)) + l_{K+1}(x_{K+1})\right\}, \tag{3.10}$$

in which $E\{\}$ denotes expectation, taken over the possible environment sequences, **e**.

Applying the principle of optimality, $\bar{L}_k^*(x_k)$ can be obtained from $\bar{L}_{k+1}^*(x_{k+1})$ by the following recurrence:

$$\bar{L}_k^*(x_k) = \min_{u_k}\left\{l_k(x_k, u_k) + \sum_{x_{k+1}} \bar{L}_{k+1}^*(x_{k+1}) P(x_{k+1}|x_k, u_k)\right\}. \tag{3.11}$$

Note that the sum in (3.11) is taken over a finite number of states, which can be reached using (3.3).

3.4.2 Computational issues

The computational issues are similar to those discussed in Section 2.6.3. We are designing a feedback controller with perfect information; however, the state space additionally includes environment information. Optimal strategies are determined numerically, by successively building approximate representations of $\bar{L}_k^*$, and using linear interpolation.

For each position in the state space, one of the following occurs after some finite number of iterations: (1) the state, x_k, is in the goal region, in which case $\bar{L}_k^*(x_k) = 0$; (2) the losses $\bar{L}_k^*(x_k)$ and $\bar{L}_{k+1}^*(x_{k+1})$ become equal for $x_k = x_{k+1}$; or (3) the loss $\bar{L}_k^*(x_k) > \bar{L}_{k+1}^*(x_{k+1})$ for $x_k = x_{k+1}$. The second condition occurs when the optimal strategy from x_{k+1} has already been completely determined, and an additional stage accomplishes nothing (this additional stage can be considered as transpiring in the goal region, in which no additional loss is received). The third condition occurs when the goal cannot be reached from x_{k+1}. If we continue to perform the dynamic programming iterations until one of the three conditions is met for every $x_k \in X$, then the optimal

strategy from all initial states will be represented. The resulting strategy is formed from the optimal actions in the final iteration. The optimal strategy is stationary, because it depends only on the state, as opposed to additionally requiring the stage index. The optimal action can be obtained from any real-valued location $x \in X$ though the use of (3.11), linear interpolation, and the approximate representation of $\bar{L}_1^*$. A real-valued initial state is given (the final component represents the environment mode, and is an integer).

We briefly discuss the computational performance of the algorithm. Let $|Q|$ denote the number of cells per dimension in the representation of $\mathcal{C}_{free}$. Let n denote the dimension of $\mathcal{C}_{free}$. Let $|E|$ denote the number of environment modes. Let $|U|$ denote the number of actions that are considered. The space complexity of the algorithm is $O(|Q|^n \ |E|)$, which is proportional to the size of the state space. For each iteration of the dynamic programming, the time complexity is $O(|Q|^n \ |E|^2 \ |U|)$ and the number of iterations is proportional to the robot velocity and the complexity of the solution strategy. The number of iterations required is directly proportional to the number of stages required for the longest (in terms of stages) optimal strategy that reaches the goal. The computation at each cell (in the application of (3.11)) has time complexity $O(|E| \ |U|)$, with n fixed.

In our simulation experiments, we have considered problems in which the dimension of $\mathcal{C}_{free}$ is two or three, and we have considered up to 32 environment modes. For two-dimensional configuration space, we typically divide the space into $50 \times 50 \times |E|$ cells, and use from 16 to 64 quantized actions (excluding $\emptyset$) to approximate translational motion. For three-dimensional configuration space, we typically divide the space into $50 \times 50 \times 64 \times |E|$ cells. These levels of resolution produce very reasonable results for most motion planning problems (see the computed examples in Section 3.5).

As in Chapter 2, the computation times vary dramatically, depending on the resolution of the representation, number of environment modes, and dimension of the con-

figuration space. For the examples that we present in this chapter, the computation times vary from about one minute to a few hours, on a SPARC 10 workstation, without parallelization. Problems that involve a two-dimensional configuration space typically require a few minutes, while problems that involve a three-dimensional configuration space typically require a few hours. The on-line execution of the optimal strategy, however, proceeds very quickly (on the order of milliseconds). This implies that the robot can quickly respond to changes in the environment when executing the optimal strategy.

3.5 Computed Examples

In this section, we present computed examples that represent the sets of problems that were discussed in Section 3.2, using the computation method discussed in Section 3.4. The mathematical models from Section 3.3 are specialized to model specific problem types. Section 3.5.1 presents computed examples that involve a changing configuration space. Section 3.5.2 presents examples that involve hazardous regions and shelters. Section 3.5.3 presents examples that involve servicing.

3.5.1 Changing configuration space

Suppose there are m obstacles in the workspace that can appear or disappear and that are defined by m dynamic regions, $\mathcal{D}_1, \ldots, \mathcal{D}_m$. It is assumed here that the stochastic processes that govern these regions are independent. In general, we have 2^m environment modes, which correspond to each possible subset of obstacles that can appear. If the dynamic-region processes are dependent, several of these subsets of regions might not be possible, thereby reducing the number of environment modes. Our approach supports dependent processes by defining the appropriate environment transition probabilities;

however, we use independent processes to ease the modeling, through the use of Poisson processes.

Recall the example process given in Section 3.3.1. The complete specification of the environment process is given for $m = 2$ and identical Poisson processes that govern the doors. A straightforward extension can be made to m dynamic regions, with distinct Poisson processes. We define two Poisson arrival rates for each dynamic region, $\mathcal{D}_i$: λ_0^i and λ_1^i. Using equations similar to (3.1), probabilities of a region appearing or disappearing can be derived to yield: P_{00}^i, P_{01}^i, P_{10}^i, and P_{11}^i. Recall from (3.2) that environment transition probabilities could be constructed from products of pairs of the probabilities P_{ij}. To generalize this, each environment transition probability is given by

$$P(e_{k+1}|e_k) = \prod_{i=1}^{m} P_{kl}^i, \tag{3.12}$$

in which k represents the i^{th} bit in the binary representation of e_{k+1}, and l represents the i^{th} bit in the binary representation of e_k. The interpretation of this is that appearing or disappearing regions correspond to bits changing from 0 to 1, or from 1 to 0 in a binary integer representation of the environment mode.

We now describe how the loss functional is built, by applying definitions from Section 3.3.3. Each $\mathcal{D}_i$ is considered as a contact dynamic region, from which m dynamic X-regions are formed. We define the terms in Equation (3.8) as $c_u = \Delta t$, $c_i = \infty$, and $c_i' = 0$. By setting $c_u = \Delta t$, we obtain time-optimal solutions when the goal is reached without collision. The constant c_i provides a penalty for colliding with a dynamic region that has appeared, which precludes this alternative from the space of reasonable

strategies. This results in

$$l_k(x_k, u_k) = \begin{cases} 0 & \text{If } x_k \in X_G \\ \Delta t & \text{If } x_k \notin X_i \text{ for all } i \\ \infty & \text{Otherwise} \end{cases} . \tag{3.13}$$

We also let $c_f = \infty$ in (3.9), which results in $l_{K+1} = 0$ if $x_{K+1} \in X_G$, otherwise $l_{K+1} = \infty$.

In our current implementation, we do not allow transitions into modes that cause a collision to occur (i.e., an obstacle may not suddenly appear such that it contacts the robot). Several computed examples will now be presented. A simple example is first presented in Figure 3.9 to illustrate many of the concepts. Figure 3.9(a) shows a problem in which a point robot can translate in $\Re^2$ using (3.3). A single doorway exists in the workspace; therefore, there are two environment modes. The outer dimensions of the workspace for this and all other examples are 100×100. For this example, $\|v\|\Delta t = 2$, $P_{00} = P_{11} = 0.98$. The goal region, X_G, for this problem and other changing configuration problems exists in all layers of X (i.e., the goal does not depend on the environment mode).

Figure 3.9(b) depicts 20 sample paths from a fixed initial location to the goal region, under the implementation of the computed optimal strategy. Initially $e_1 = 0$, indicating that the door is open. Each of the 20 sample paths is obtained by sampling an environment mode sequence, $\mathbf{e}$, from the Markov process, to obtain one trajectory in X that terminates in the goal. Figure 3.9(b) illustrates different sample paths that can result during execution, even though the strategy is fixed. These differences are caused by variations in $\mathbf{e}$. Some sample paths go through the doorway; others take a longer way to the goal. In one sample path, the robot begins to go toward the other doorway because

the nearer door closed; however, the robot changes its direction and heads toward the doorway again because the door reopened.

Figures 3.9(c) and 3.9(d) depict the optimal strategy γ^*. The direction of each arrow indicates the direction of motion (specified as $u_k = \gamma^*(x_k)$) for the robot, from that particular state location. The state space was quantized into $75 \times 75 \times 2$ locations for determining the optimal strategy; however, for clarity we show actions at fewer locations in the figures. When $e = 0$, a sharp division is observed between places in the state space that lead to the doorway and places that lead to the open corridor. When $e = 1$, the robot moves through the open corridor, to the goal region.

Figures 3.9(e) and 3.9(f) show 20 level-set contours of the cost-to-go function, $L_1^*(x_1)$. This function increases as the distance from the goal increases. For translational motion, the negative gradient of the cost-to-go function represents the direction of motion for the robot. Hence, the cost-to-go function is similar to a numerical navigation function [101], [109], [160]; however, in our work, the representation of $L_1^*(x_1)$ is obtained as a by-product of determining the optimal strategy.

We next show some results for a more complex example, in which there are 18 doorways, in Figure 3.10. There is a point robot, for which $\|v\|\Delta t = 3$. There are three different classes of doors, which open and close simultaneously (see Figure 3.10(a)). This results in three disconnected dynamic regions and eight environment modes. Each class of doors is governed by the same Poisson parameters as the previous example.

Figure 3.10(b) shows 20 sample paths under the implementation of the optimal strategy, when $e_1 = 0$ (all doors are initially open). Many different sample paths are obtained under the optimal strategy, γ^*. For this example, there are places in the state space in which the optimal action is $\gamma_k^*(x_k) = u_k = \emptyset$ (i.e., the robot waits for some door(s) to open). Figure 3.11 shows 30 level-set contours of the cost-to-go function, $L_1^*(x_1)$. When some of the doors close, wells form in the cost-to-go function. Figure 3.12 shows two

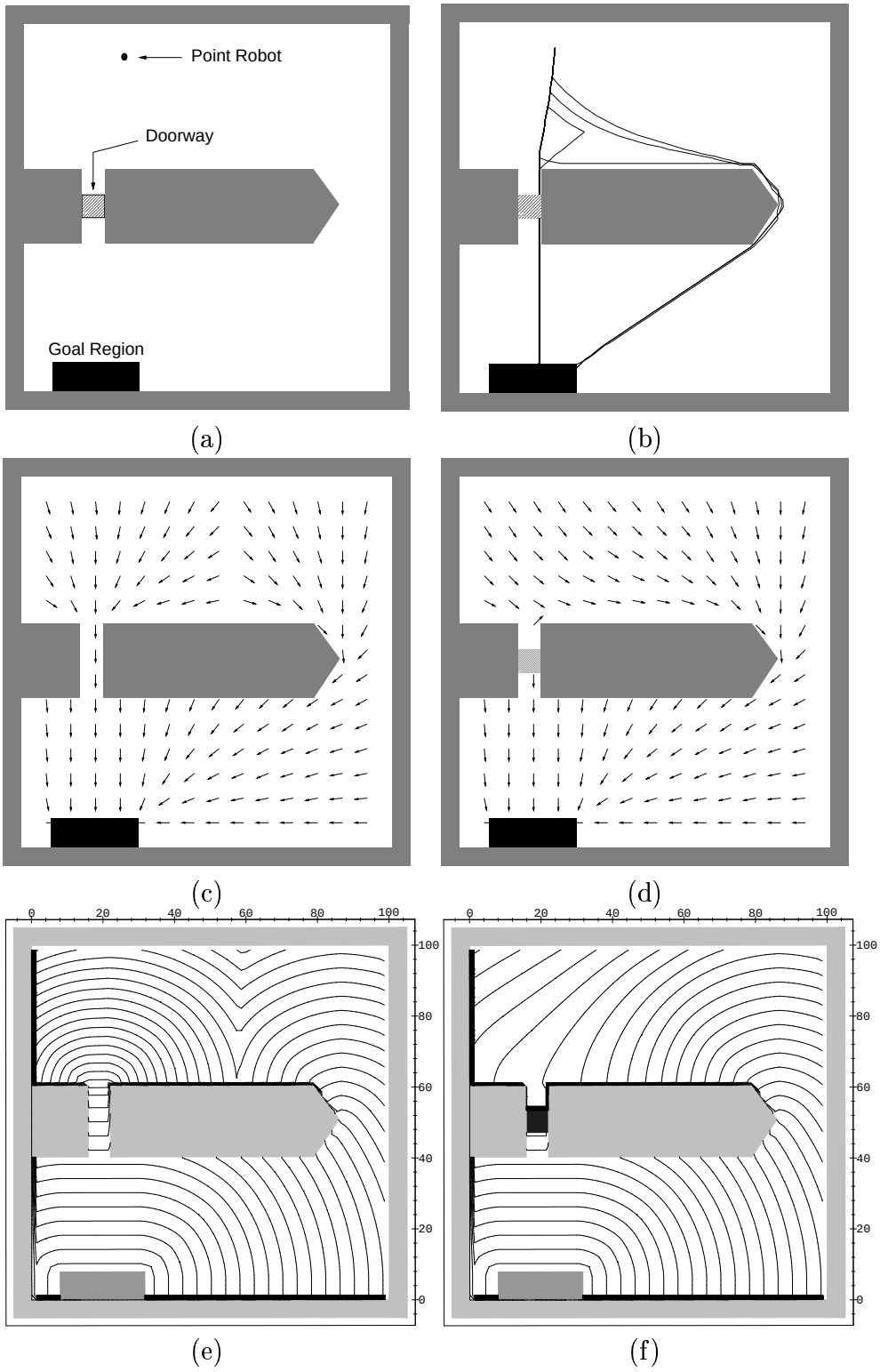
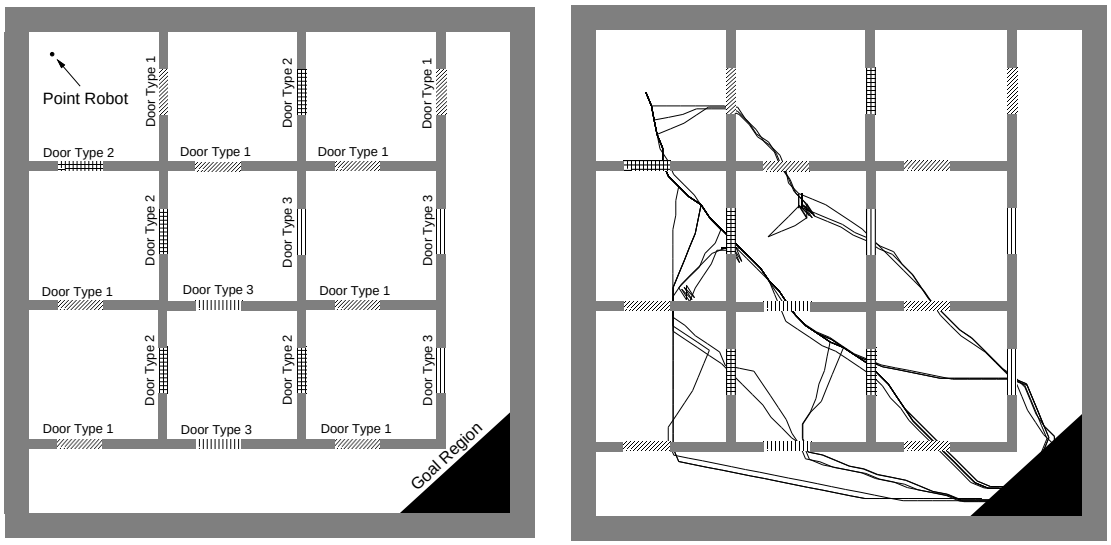


Figure 3.9 (a) A door problem; (b) 20 sample paths; (c) γ^* at $e = 0$; (d) γ^* at $e = 1$; (e) isoperformance curves at $e = 0$; (f) isoperformance curves at $e = 1$.

portions of $L_1^*(x_1)$, which correspond to $e = 0$ and $e = 7$. When $e = 7$, the robot could be trapped in any of the nine square compartments and must wait for a door to open; this causes nine wells to appear in the cost-to-go function.



(a) (b)

Figure 3.10 (a) A problem that has 18 doors; (b) 20 sample paths.

Figure 3.13 shows results from the problem discussed in Figure 3.1(a). Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 1$ (the lower door is closed, and the upper door is open). Graphs of $\mathbf{e}$ are also given. For the lower door, we have $P_{00} = P_{11} = 0.99$, and for the upper door, we have $P_{00} = P_{11} = 0.98$. There is a three-dimensional configuration space, in which two components represent position, and the final component represents orientation. The incremental motion model for the robot is constrained rotation with reverse, which consists of five actions. The first action allows the robot to remain motionless: $x_{k+1} = x_k$.

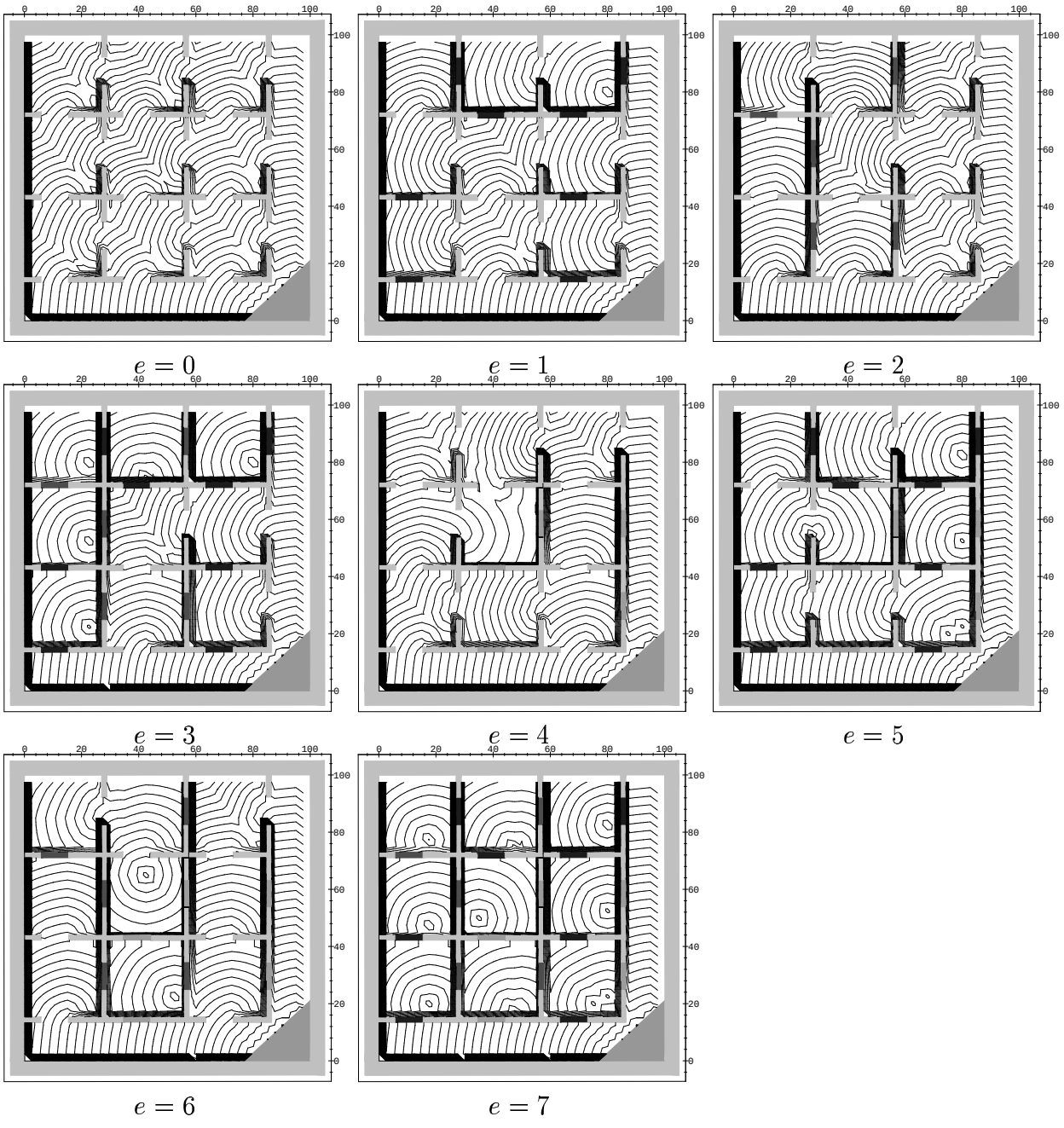
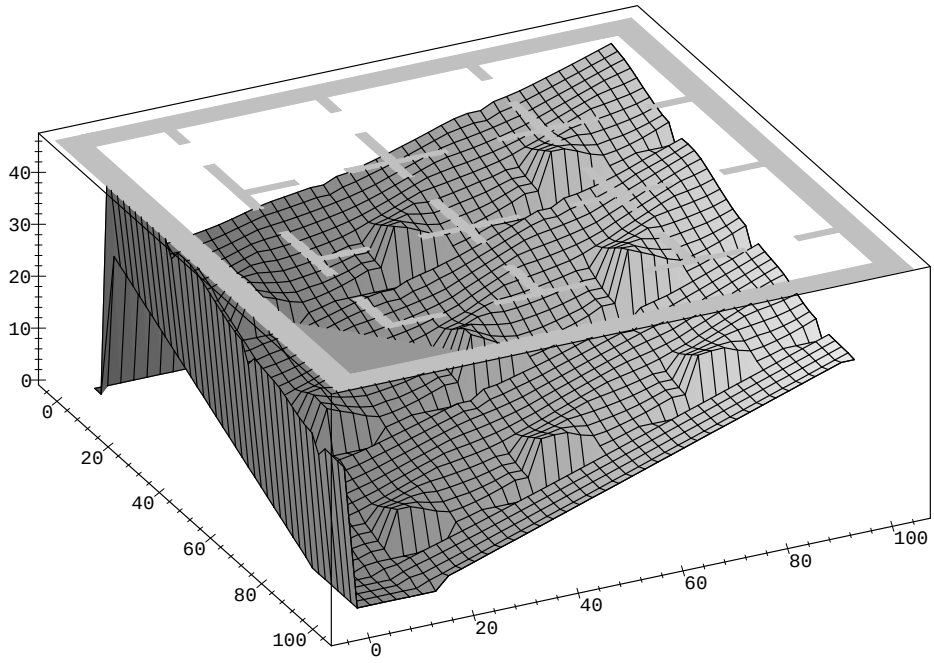
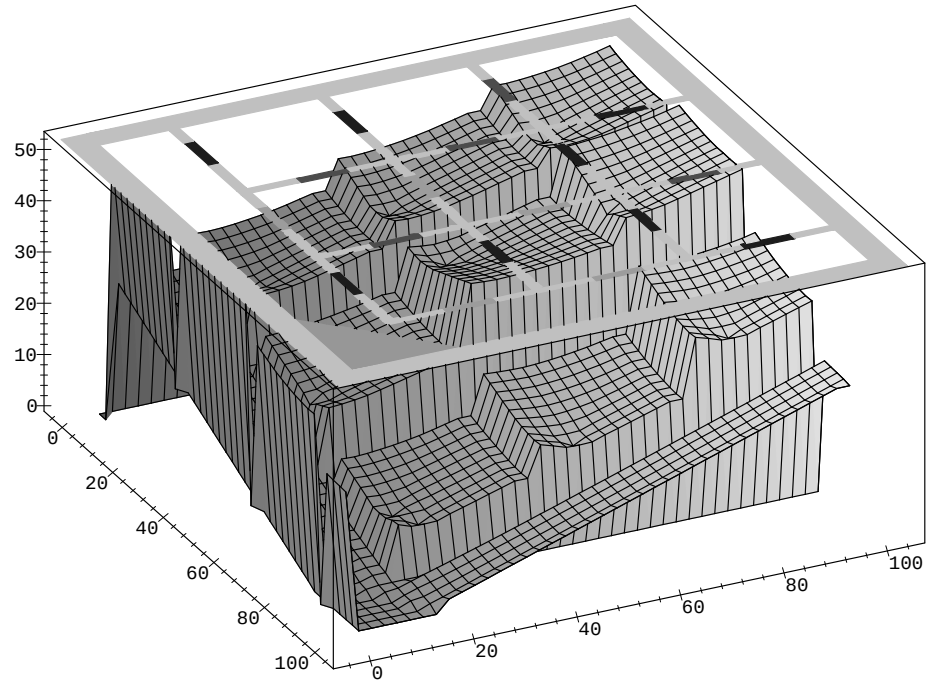


Figure 3.11 The isoperformance curves for γ^*.



(a)



(b)

Figure 3.12 Cost-to-go functions for: (a) $e = 0$ and (b) $e = 7$.

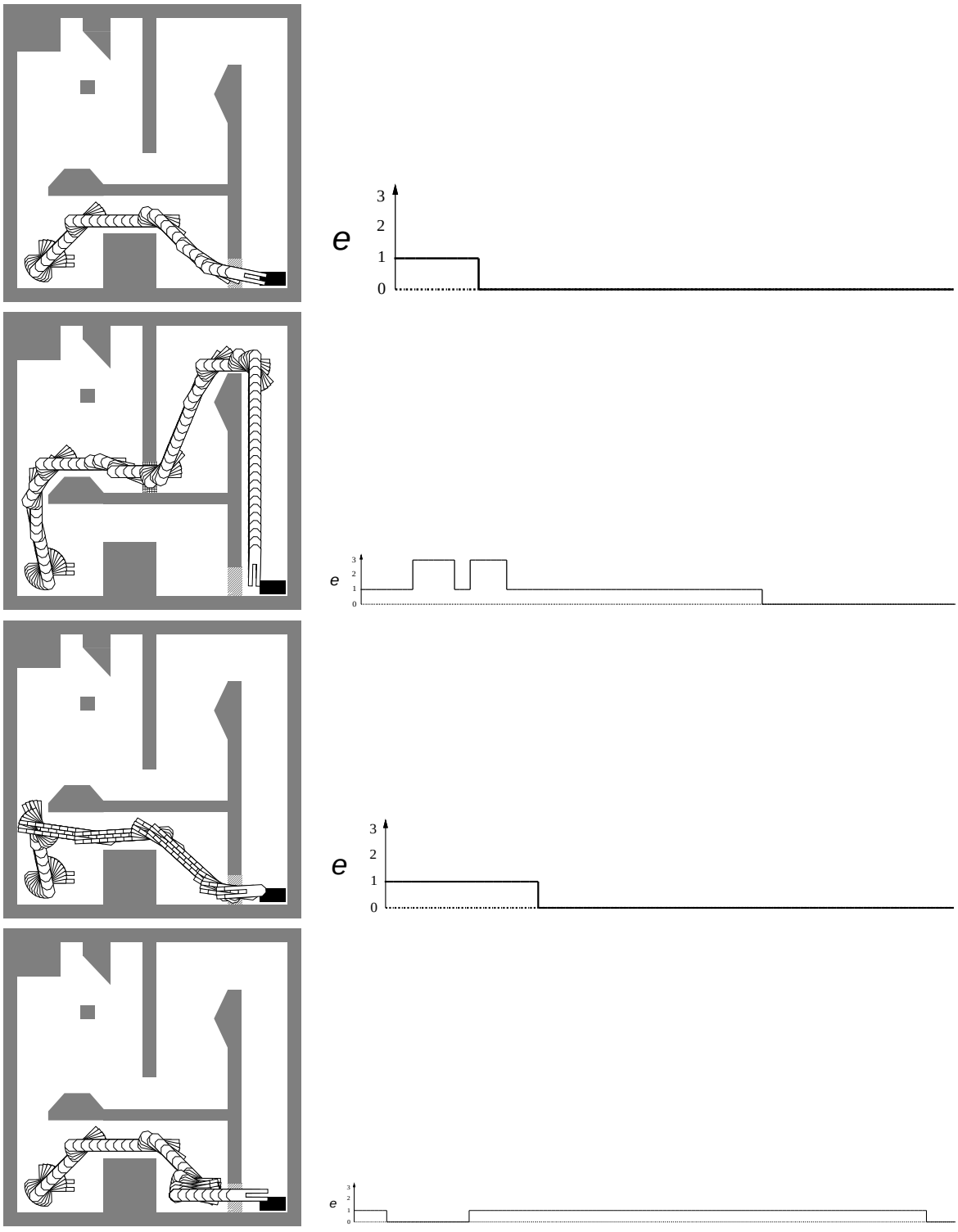


Figure 3.13 Four sample paths for a changing configuration space problem with two doors.

The robot can either turn clockwise or counterclockwise, in which

$$x_{k+1} = \begin{bmatrix} x_k[1] \\ x_k[2] \\ (x_k[3] \pm \theta_m \Delta t)_{mod 2\pi} \\ e_{k+1} \end{bmatrix}. \tag{3.14}$$

Above, θ_m represents a fixed angular velocity. The fourth and fifth actions allow the robot to translate along the orientation x_{k+1}, yielding

$$x_{k+1} = \begin{bmatrix} x_k[1] \pm \|v\| \Delta t \; cos(x_k[3]) \\ x_k[2] \pm \|v\| \Delta t \; sin(x_k[3]) \\ x_k[3] \\ e_{k+1} \end{bmatrix}. \tag{3.15}$$

The values $\|v\| \Delta t = 3$ and $\theta_m \Delta t = 0.2$ were selected.

Four very different sample paths are shown in Figure 3.13. In the first sample, the lower door opens, and the robot efficiently moves to the goal region. In the second sample, the lower door remains closed for a long period of time, and the robot chooses to move through the upper doorway, taking a much longer route. In the third sample, the robot starts to head for the upper doorway, and then changes its heading when the lower door opens. In the fourth sample, the lower door opens and then closes again. The robot waits for the door to open again, instead of taking the longer route.

Figure 3.14 shows results from the problem discussed in Figure 3.1(b). Four sample paths are shown under the implementation of the optimal strategy. There are five dynamic regions, each of which corresponds to a transient obstacle. For each transient obstacle, $P^i_{00} = P^i_{11} = 0.98$. Initially, $e = 0$, which corresponds to the existence of none of the five transient obstacles. The robot can translate in the workspace through (3.3), in which $\|v\| \Delta t = 3$. Again, we observe many different sample paths as the free configuration space changes in different ways. If a transient obstacle appears at any time during

the execution, it is shown in the figure (i.e., it may appear that the robot collides with the transient obstacle in some of the figures, but the obstacle disappears in time).

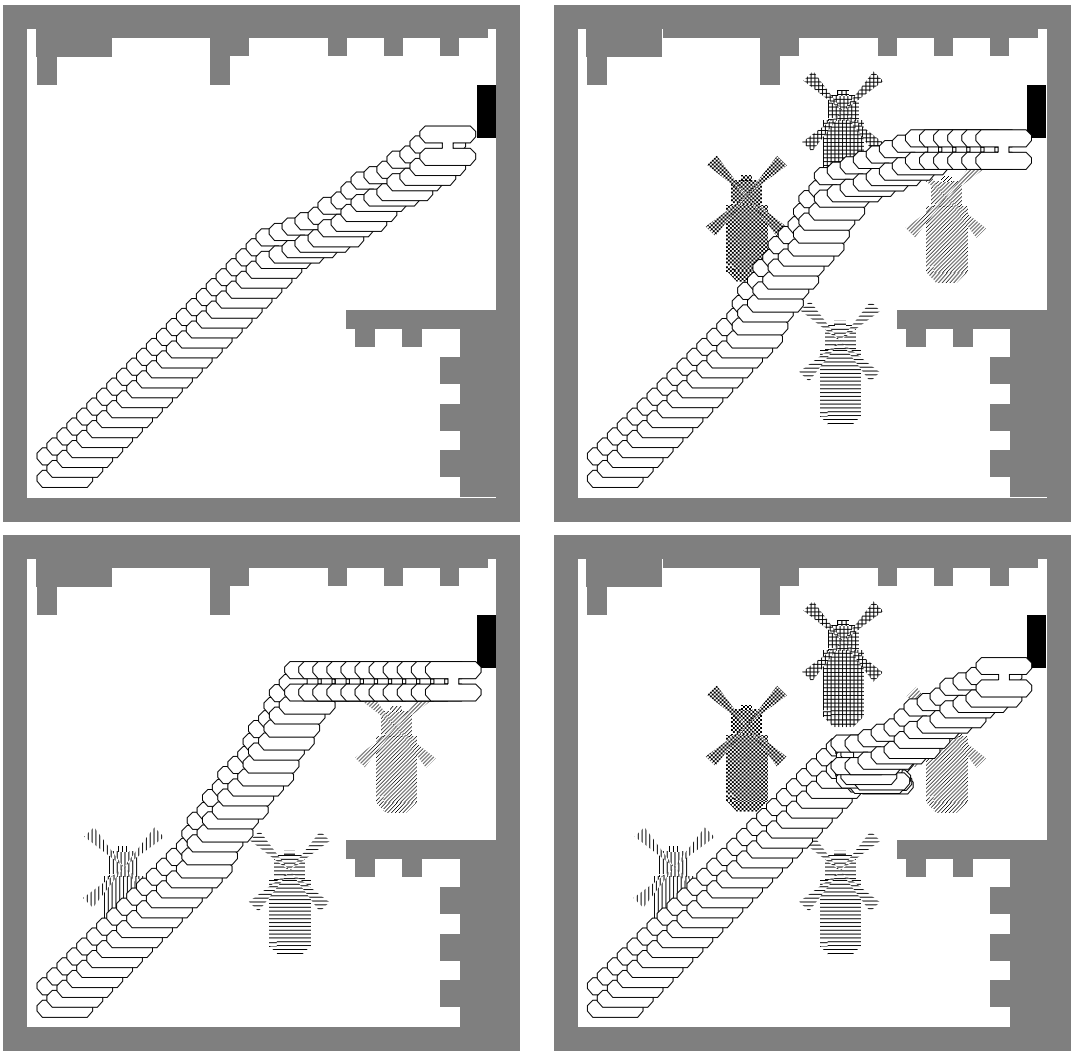


Figure 3.14 Four sample paths for a transient obstacle problem.

3.5.2 Hazardous regions and shelters

For this type of problem, we consider only two environment modes: either the environment is hazardous, or the environment is safe. Of course, generalizations of this are

possible to multiple levels of danger, or to different shelters for different types of hazards. We have a single dynamic region, $\mathcal{D}_1$. Let $c_f = \infty$, $c_u = \Delta t$, $c_i = 0$, and $c_i' > 0$. To construct the loss functional, $\mathcal{D}_1$ is considered as an enclosure dynamic region, as defined in Section 3.3.3.

Figure 3.15(a) shows a basic example that illustrates the shelter and hazardous region concepts. There is a point robot that translates in $\Re^2$ using (3.3) and four thin horizontal regions that are designated as shelters. For this example, $\|v\|\Delta t = 2$, $P_{00} = 0.75$ and $P_{11} = 0.98$. The loss function is defined with $c_0 = 0$ and $c_0' = 5$. This is a generalization of a local path optimization problem defined in [170], which involved a single horizontal shelter region; the problem was analogous to the problem of crossing a one-lane street. The current problem is analogous to the problem of crossing a multilane street that has multiple shelters. For the one lane case, an analytical solution was presented in [170], but the analysis is difficult to generalize to the multiple-lane case.

Figure 3.15(b) shows 20 sample paths under the implementation of the optimal strategy. One can see the use of the shelters during the times when the environment becomes hazardous ($e = 1$). When $e = 1$, the robot seems to head toward the next shelter and moves along the shelter until the environment mode changes back to 0. This intuitive observation about the robot's behavior is further supported by the results shown in Figures 3.15(e) and 3.15(f), which show 20 level-set contours of the cost-to-go function, $L_1^*(x_1)$. When $e = 0$, the contours indicate that the robot moves across the street and directly toward the goal. When $e = 1$, the robot moves along the shelters.

Figure 3.16 shows results from the problem discussed in Figure 3.2. Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 0$ (the environment is not hazardous). We have $P_{00} = P_{11} = 0.98$. The incremental motion model for the robot is constrained rotation with reverse (as considered for the example in Figure 3.13), in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$.

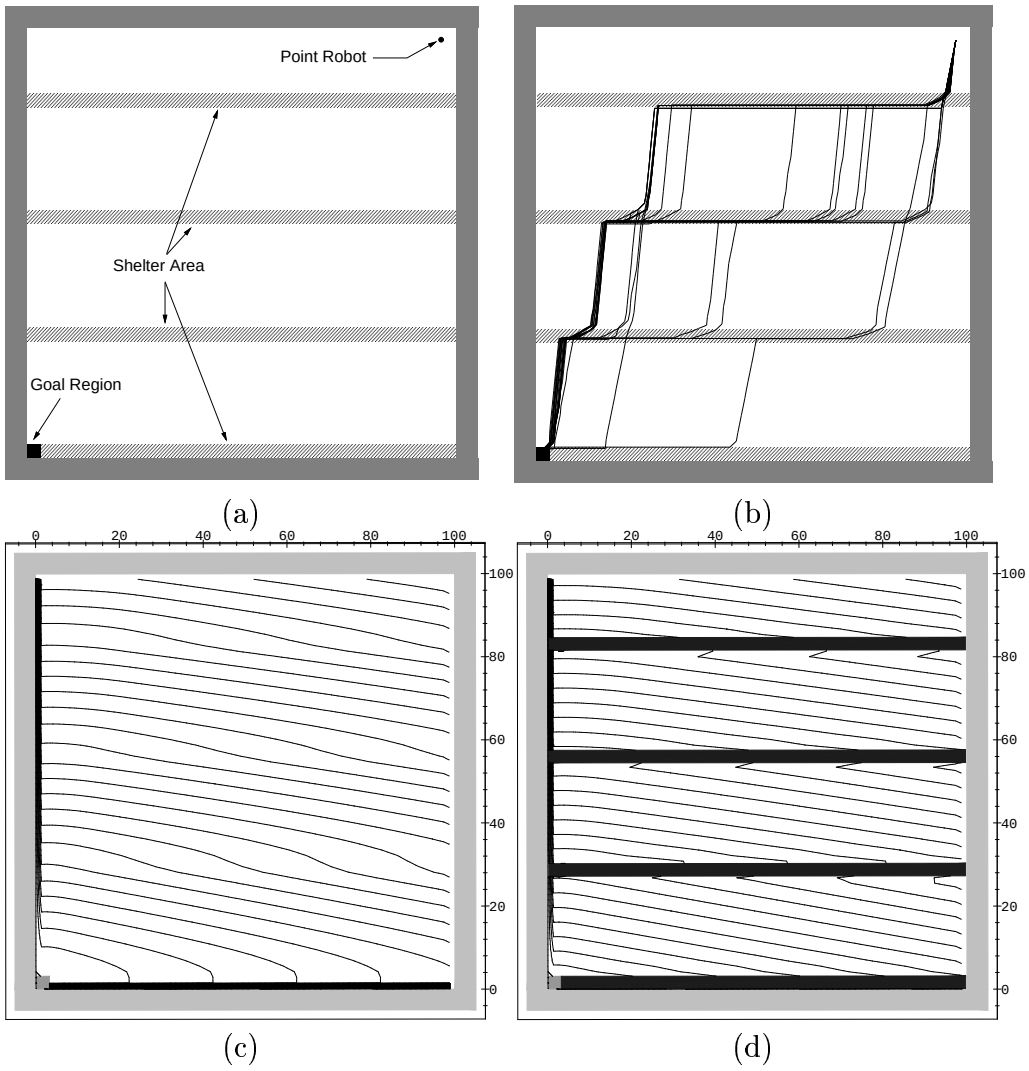


Figure 3.15 (a) A hazardous region and shelter problem; (b) 20 sample paths; (c) isoperformance curves at $e = 0$; (d) isoperformance curves at $e = 1$.

During execution, very different sample paths, which reflect the responses due to the environment becoming hazardous, are obtained. In the first sample, the environment does not become hazardous, and the robot never moves into a shelter (although it travels close to the shelters). In the remaining sample paths, the robot responds to the hazardous environment by moving into a shelter. In the final sample path, the environment became hazardous three times, causing the robot to take shelter each time. After the robot moves into a shelter, it remains there until the environment mode e switches back to 0. Further, while it is waiting inside a shelter the robot chooses an orientation that points along the remaining optimal path for $e = 0$. We have observed this behavior more clearly through animations of the robot moving along the sample paths shown in Figure 3.16.

3.5.3 Servicing problems

Suppose there are m different types of services to be performed. For simplicity we assume that a request for a particular service to be performed arrives with Poisson frequency λ_s^i. Each dynamic region, $\mathcal{D}_i$, corresponds to a place in which the robot can respond to a service request. We further assume that the robot can immediately process a request, which causes the request to be cleared. We assume that any number of services can be requested simultaneously and that the governing processes are independent. These assumptions are not, of course, necessary, but they simplify the development and presentation of examples.

We now define the environment probability distribution. If $x_k \in X_i$ then $P_{11}^i = P_{10}^i = 0$, and $P_{00}^i = P_{01}^i = 1$; otherwise, $P_{11}^i = 1$ and

$$P_{10}^i = \int_0^{\Delta t} \lambda_s e^{-\lambda_s t_a} dt_a = 1 - e^{-\lambda_s \Delta t}, \tag{3.16}$$

in which λ_s is the Poisson arrival rate for the i^{th} service request. The elements of the environment transition distribution are obtained by forming products as in (3.12).

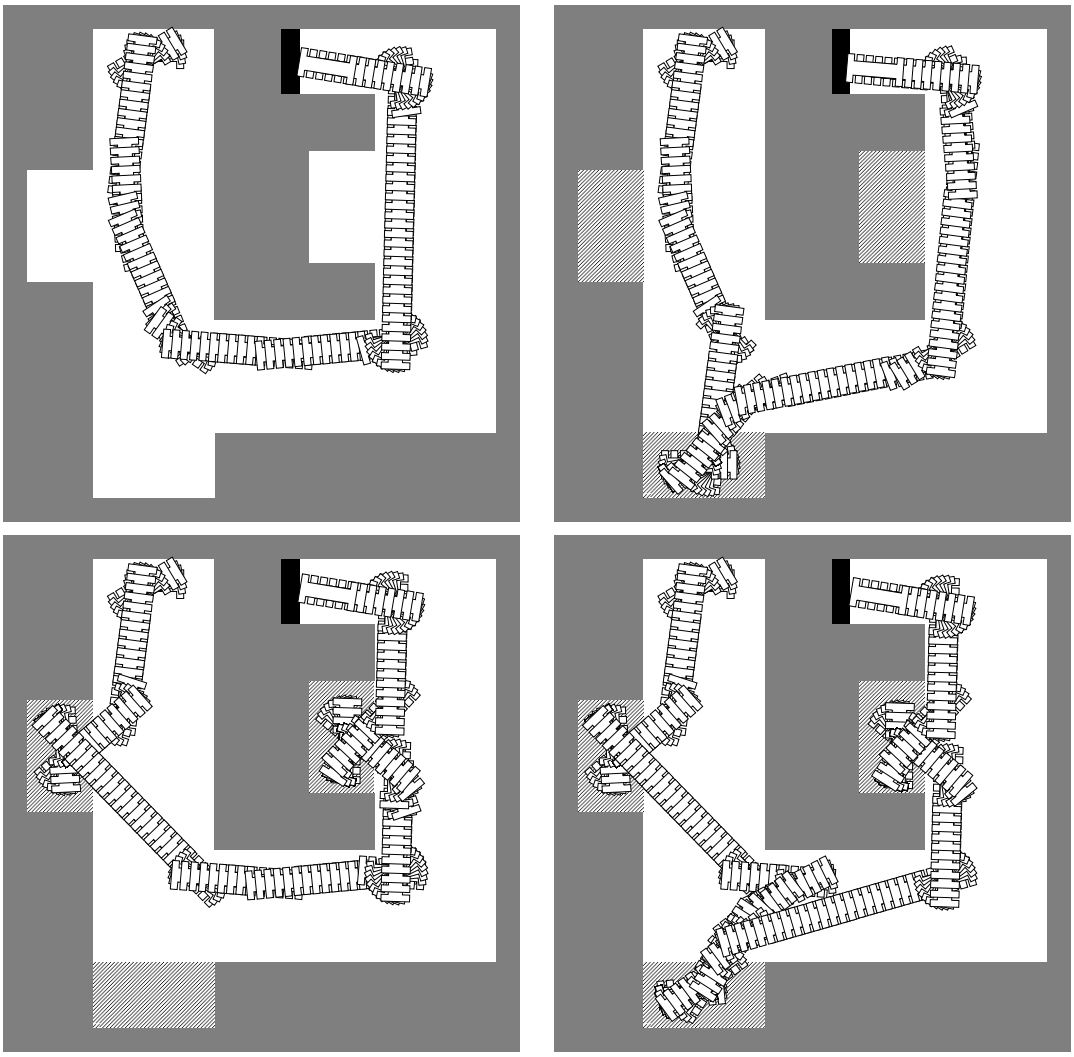


Figure 3.16 Four sample paths for a hazardous region and shelter problem.

We let $c_f = \infty$, $c_u = \Delta t$, $c_i = 0$, and $c_i' > 0$. To construct the loss functional, $\mathcal{D}_1$ is considered as an enclosure or contact dynamic region, as defined in Section 3.3.3.

Figure 3.17(a) shows a basic example in which there is a translating point robot, and five small regions in which a single type of service can be performed. There is a single dynamic region, which corresponds to the existence of a service request. The goal region, X_G, exists for both layers of X. This problem is similar to the one analyzed in [173], which has points distributed on a plane. For that model, properties of optimal paths were presented, although the actual optimal solution was not derived. Figure 3.17(b) shows 20 sample paths for our problem under the implementation of the optimal strategy. The figure shows the deviations that the robot makes to process the service requests. Figures 3.17(c) and 3.17(d) show the computed optimal strategies for the two modes 0 and 1, respectively. When there is a service request, the robot heads toward a nearby service area to service the call, except when it is near the goal region. This general behavior is exhibited in the level-set contours of the cost-to-go function, shown in Figures 3.17(e) and 3.17(f). When $e = 1$, the contours form wells that draw the robot toward a nearby region. This general behavior is supported by the theoretical analysis in [173], in which by analogy a Delaunay path [153] (a path formed by edges from the Delaunay graph on the plane) would be formed from the initial to goal position when the environment mode is 1.

Figure 3.18 shows results from the problem discussed in Figure 3.3. Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 2$ (there is a request for the second service only). For the first service type, we have $P_{00} = P_{11} = 0.99$, and for the second type, we have $P_{00} = P_{11} = 0.98$. The example uses a nonholonomic incremental motion model. For this case there are four actions. The first action causes no motion, and the second action causes straight

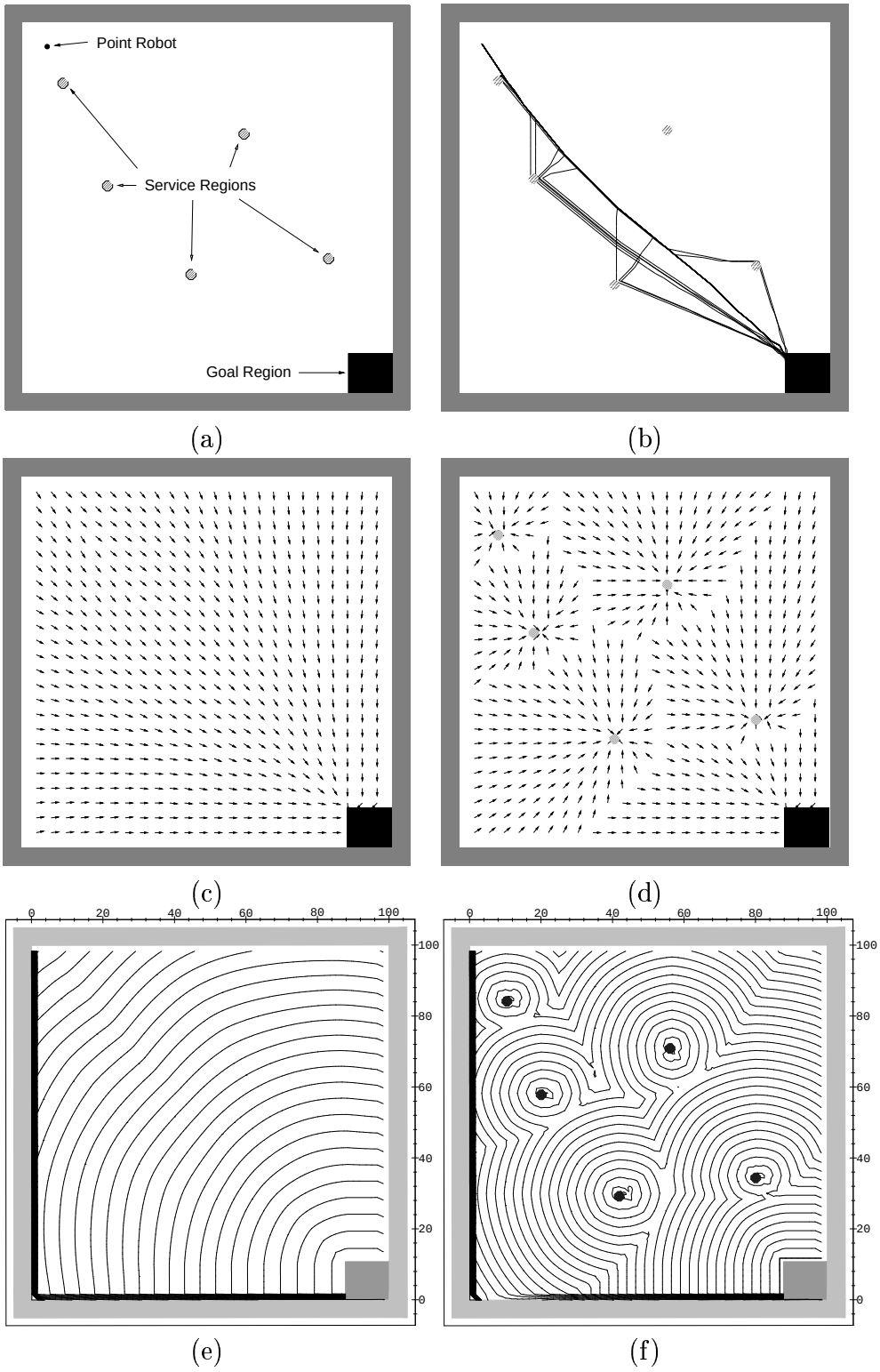


Figure 3.17 (a) A servicing problem; (b) 20 sample paths; (c) γ^* at $e = 0$; (d) γ^* at $e = 1$; (e) isoperformance curves at $e = 0$; (f) isoperformance curves at $e = 1$.

motion. The remaining two actions are of the form

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\|\Delta t \; cos(x_k[3] \pm \theta_m \Delta t) \\ x_k[2] + \|v\|\Delta t \; sin(x_k[3] \pm \theta_m \Delta t) \\ (x_k[3] + \theta_m \Delta t)_{mod 2\pi} \\ e_{k+1} \end{bmatrix}, \tag{3.17}$$

which requires the robot to translate while rotating. This incrementally implements a fixed turning radius constraint that is based on $\|v\|\Delta t$ and θ_m. For the current example, the values $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$ were used. Each service region is an enclosure dynamic region. In this example, the goal region in the state space, X_G, exists onlu for $e = 0$; this implies that the robot must reach the goal region while there are no requests for servicing. Very different sample paths are obtained because the robot must process any request that appears in order to reach X_G.

3.6 An Extension to a Part-Transferring Problem

This section presents a variation on the method presented thus far in this chapter. The problem involves the delivery of parts from source locations to destination locations in the workspace. The robot is capable of manipulating and carrying the parts, which thus involves a gross-motion planning problem. The particular part, the particular source location, and particular destination location are *a priori* unknown, but are modeled with a stochastic process. Section 3.6.1 provides the mathematical modeling that characterizes this problem as environment-predictability uncertainty by defining environment modes that correspond to delivery requests. Section 3.6.2 presents computed examples for both rigid robots and manipulators. This problem can be considered as a component in a flexible assembly or manufacturing system (e.g., [34], [77], [102], [192]); the details of this application are discussed in [172].

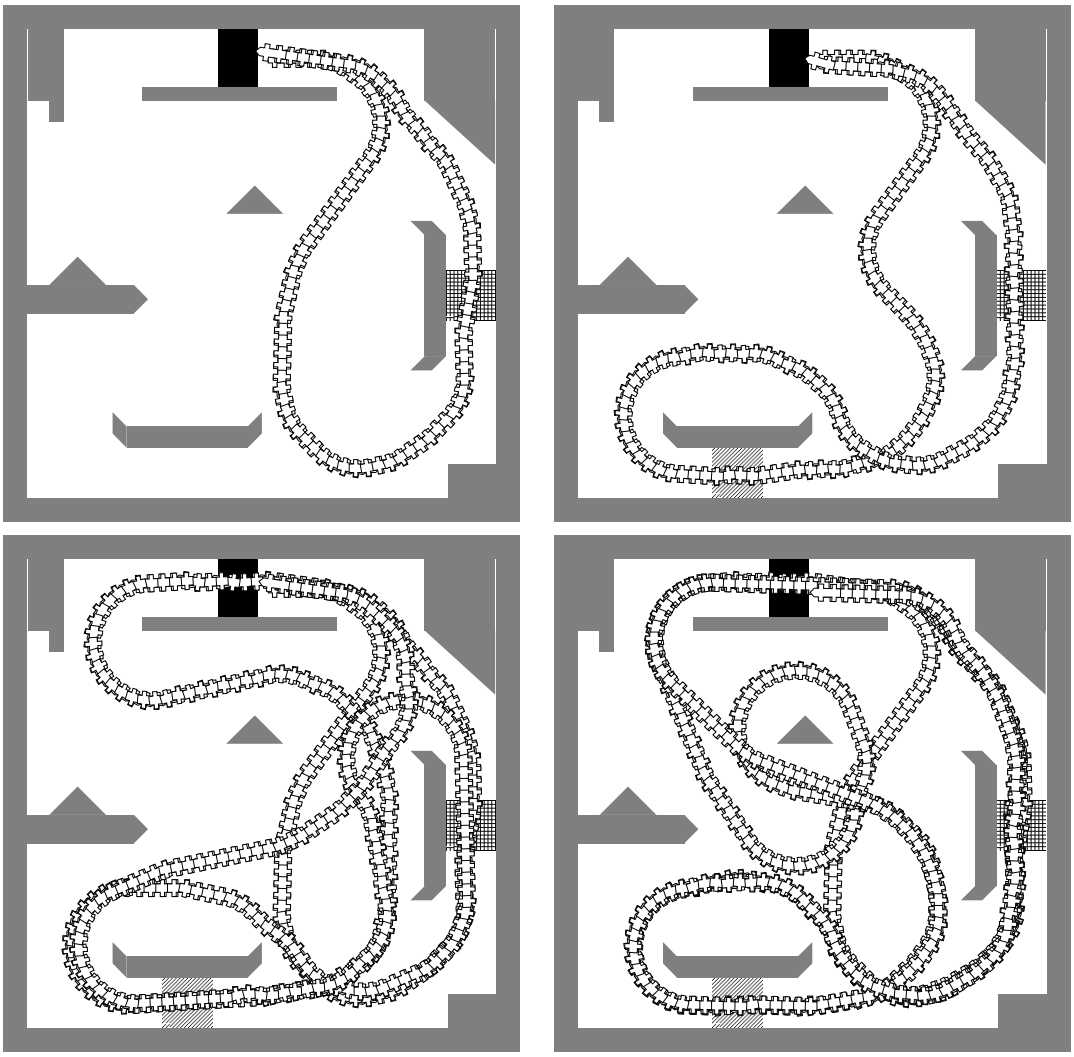


Figure 3.18 Four sample paths for a servicing problem with a nonholonomic robot.

We characterize the problem of gross-motion planning for assembly as follows. A scheduler issues requests to the robot to grasp a particular part from a specified source and to deliver the part to a specified destination. *A priori*, the only information regarding how these requests will be issued is in the form of a probability distribution on the set of possible part/source/destination requests. Because a fine-motion plan will often follow the execution of the gross motion, a source or destination is typically not specified as a single configuration, but is specified as a subset of the configuration space (which could be disconnected). The gross-motion planning problem is to derive a set of motion strategies that will produce optimal throughput of the assembly cell, in an expected sense. We idealize the object grasping problem (for relevant issues in grasping, see [41], [150]), and assume that the robot can immediately pick up and carry an object through the workspace.

3.6.1 Mathematical modeling

In addition to static obstacles, let the workspace, $\mathcal{W}$, contain a set of S *source regions*, denoted by $\{\mathcal{S}_1, \ldots, \mathcal{S}_S\}$, and D *destination regions*, denoted by $\{\mathcal{D}_1, \ldots, \mathcal{D}_D\}$. Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_P\}$ denote a collection of P rigid parts. A *request* can be issued to the robot that requires a part, $\mathcal{P}$, be picked up from a source, $\mathcal{S}$, and delivered to a destination, $\mathcal{D}$. In general there are PSD different requests that can be issued. We also allow the possibility of having no outstanding requests at a given time. It is assumed that at a given time, the robot has complete knowledge of its configuration and all parts, sources, destinations, and requests.

To characterize requests and the status of the robot with respect to requests, we define a set, E, of environment modes. An environment mode (in this section) is represented by four components, $\langle p, s, d, C/W \rangle$. The first three represent the part, source, and desti-

nation, respectively. The fourth component is W to represent a mode in which a request has been given, but the robot has not yet picked up the part, or is C to represent a mode in which the request has been given and the robot is carrying the part. In addition, we have a special mode, $NR \in E$, which represents the condition in which no requests are to be processed. Hence, we have the number of environment modes, $|M| = 2PSD + 1$.

We next define the free configuration space for different modes. We first define $E_w \subset E$ as the set of all modes such that a part is waiting to be picked up, and $E_c \subset E$ as the set of of all modes such that the robot is carrying a part. If $e = \langle p, s, d, W \rangle \in E_w$, then

$$\mathcal{C}^e_{free} = \{\mathbf{q} \in \mathcal{C} |\ \mathcal{A}(\mathbf{q}) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \mathcal{S}_{s-1} \cup \mathcal{S}_{s+1} \cdots \mathcal{S}_S) = \emptyset\}, \tag{3.18}$$

in which $\mathcal{A}(\mathbf{q})$ denotes the robot at configuration $\mathbf{q}$, and $\mathcal{B}$ denotes the static obstacle region (see [109]). In addition to avoiding collision with static obstacles, we also require that the robot avoid collision with other source regions. For our context this is preferable, because other parts presumably may arrive at other sources at any time. The collision detection with source regions could, of course, be removed in some applications.

Suppose $e = \langle p, s, d, C \rangle \in E_c$, which implies that the robot is carrying some part, $\mathcal{P}_p$. We use the notation $\mathcal{P}(\mathbf{q})$ to denote the transformed part, when grasped by the robot, which is at configuration $\mathbf{q}$. As discussed in Section 3.5.1, when a part is being carried rigidly by the robot, the effect is that of the "new" robot described as $\mathcal{A}(\mathbf{q}) \cup \mathcal{P}_p(\mathbf{q})$. Thus, the free configuration space becomes

$$\mathcal{C}^e_{free} = \{\mathbf{q} \in \mathcal{C} |\ (\mathcal{A}(\mathbf{q}) \cup \mathcal{P}_p(\mathbf{q})) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \mathcal{S}_{s-1} \cup \mathcal{S}_{s+1} \cdots \mathcal{S}_S) = \emptyset\}. \tag{3.19}$$

The only remaining environment mode in E is $e = NR$, in which we have

$$\mathcal{C}^e_{free} = \{\mathbf{q} \in \mathcal{C} |\ \mathcal{A}(\mathbf{q}) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_S) = \emptyset\}. \tag{3.20}$$

We next describe transitions that occur between environment modes as a discrete-time finite-state Markov process. A final stage, K, is defined to preclude a special treatment of infinite stages and, in practice, $K\Delta t$ can be considered as the total time that the robot is in operation. We have considered the following three types of probabilistic state transitions (although many others are possible):

1. the probability of receiving a p, s, d request while in mode NR,

2. the probability that the destination will change to a new destination while in a carrying mode,

3. the probability that the source will change while in a waiting mode.

The first transition type is the most fundamental, and can be generally expressed as $P(e_{k+1}|e_k = NR) \geq 0$ if $e_{k+1} \in E_w$, and $P(e_{k+1}|e_k = NR) = 0$ otherwise. The second transition type can be expressed as $P(e_{k+1}|e_k \in E_c)$, which is allowed to be nonzero only if e_k and e_{k+1} correspond to the same part and source. Ideally, the destination remains fixed, and $P(e_{k+1}|e_k \in E_c) = 1$ if $e_{k+1} = e_k$, and 0 otherwise. Similarly, the third type can be expressed as $P(e_{k+1}|e_k \in E_w)$, which is allowed to be nonzero for any value of $e_{k+1} \in E$. Ideally, $P(e_{k+1}|e_k \in E_w) = 1$ if $e_{k+1} = e_k$, and 0 otherwise. Any of these transition probabilities can be viewed as being derived from an underlying Poisson process, as shown in Section 3.3.1.

In addition to the above three types, there are several other key transitions that we model deterministically; these are the transitions from elements in E_w to elements in E_c, and from elements in E_c to NR. The previous transitions were independent of $\mathbf{q}_k$ and u_k; however, the following transitions depend directly on the configuration and the action. Suppose the robot has an action, $FMP \in U$, that represents fine-motion planning. To grasp or ungrasp a part, the robot can choose this action from state x_k (causing

fine-motion operation to be performed), and the robot is returned to the gross-motion planning system in some state x_{k+1}.

We assume that the fine-motion operation can be performed to pick up a part only when the robot has reached the correct source region, and to deliver a part when the robot has reached the correct destination region. When the action FMP is executed, we assume that the environment mode changes with probability one. At a source region, $e_k = \langle p, s, d, W \rangle$ changes to $e_{k+1} = \langle p, s, d, C \rangle$, and at a destination region, $e_k = \langle p, s, d, C \rangle$ changes to $e_{k+1} = NR$. We could extend the model for error-handling by defining failure modes in case a FMP is not satisfactorily executed.

Consider the motion model for the case in which $\mathcal{C} \subseteq \Re^2$, and the robot is limited to translational motion. More complicated motions will be considered in Section 3.6.2, including modeling of a redundant manipulator. The motion of the robot could also strongly depend on the environment mode; for example, the velocity bound, $\|v\|$, might depend on the part that the robot is carrying. We define the action space as $U = [0, 2\pi) \cup \{\emptyset, FMP\}$. If $u_k \in [0, 2\pi)$, then $\mathcal{A}$ attempts to move a distance $\|v\|\Delta t$, in which $\|v\|$ denotes some fixed speed for $\mathcal{A}$. If $u_k = \emptyset$, then the robot remains motionless.

For this problem, there is no goal region in which the robot must terminate. Strategies are selected that minimize the expected time that parts wait to be delivered. We describe a general definition for l_k that pertains to our motion planning problem for assembly. Let $\bar{t}_f(x_k)$ denote the *expected* time to complete a fine-motion planning task (which results in a new assembly mode) by choosing the action $u_k = FMP$ from state x_k. Figure 3.19 illustrates how the change from gross-motion planning to fine-motion planning affects the expected time for completing the fine operation. Accounting for this dependence further optimizes the expected time, especially when the time for fine-motion planning is significant. Recall that x_k simultaneously represents q_k, p, s, d, and C/W. If $m_k \in M_w$, then $\bar{t}_f(x_k)$ represents the expected time to grasp the part. If $m_k \in M_c$, then

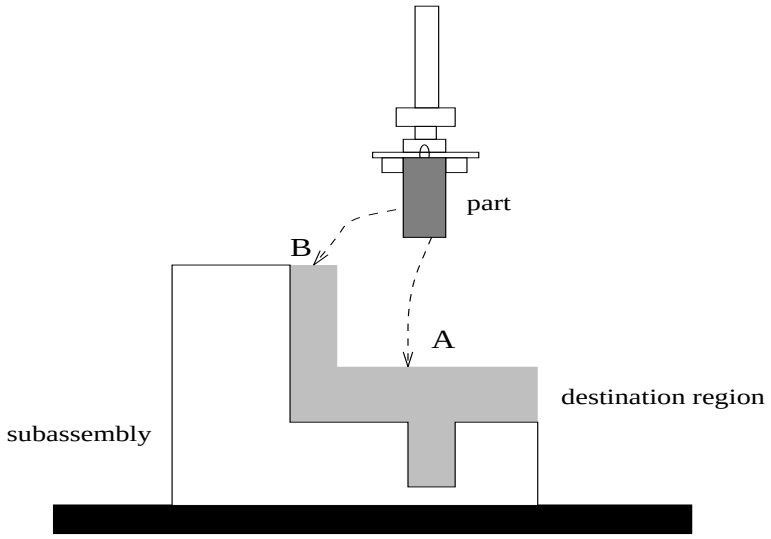


Figure 3.19 An example of the variation of the cost of the fine motion planning depending on the contact position with the destination region. Contact at A will give rise to a smaller expected time for mating compared to B.

$\bar{t}_f(x_k)$ represents the expected time for an ungrasp operation for the part (mating with a subassembly, machining, or some other fine motion). In general

$$l_k(x_k, u_k) = \begin{cases} 0 & m_k \in NR \\ \bar{t}_f(x_k) & u_k = FMP \\ \Delta t & \text{Otherwise} \end{cases} . \tag{3.21}$$

The loss thus becomes the aggregate of times that parts wait before being delivered. If there are no requests (i.e., $m_k = NR$), then no penalty is received. To reduce the loss over a long period of time, the robot will prefer actions that bring the assembly mode back to NR as quickly as possible.

3.6.2 Computed examples

In this section, we present computed solutions for six different problems that involve the transfer of parts in a workspace. The first four problems involve a rigid robot in the workspace, which could represent the end-effector of a Cartesian robot, or a mobile robot. The final two problems involve manipulators, for which optimal strategies are derived directly on the joint space. The results were computed using the algorithm that was described in Section 3.4.2.

3.6.2.1 Rigid robot simulations

The first example is designed to illustrate many of the basic concepts. It involves a rigid robot that translates in a planar workspace cluttered with obstacles (see Figure 3.20). Two different parts can be moved from either of two sources to either of two destinations. There are consequently 17 possible environment modes. The probability that a request will appear at stage $k+1$ while $e_k = NR$ is given to be 0.05. In addition, we declare that all p, s, d combinations are equally likely to occur. We assume that once a p, s, d combination is given to the robot, it will not change or be retracted until part p is delivered to destination d. The robot moves with $\|v\|\Delta t = 3.0$, and the workspace is 100 units in each axis.

Figures 3.21(a) and 3.21(b) depict the level-set contours of the cost-to-go function, $L_1^*(x_1)$ for environment modes $\langle 1, 1, 1, W \rangle$ and $\langle 1, 1, 1, C \rangle$, respectively. In Figure 3.21(a) a minimum exists at the first source region, and in Figure 3.21(b) the minimum appears at the destination region.

Figures 3.21(c) and 3.21(d) depict the optimal strategy γ^* for environment modes $\langle 1, 1, 1, W \rangle$ and $\langle 1, 1, 1, C \rangle$, respectively. The direction of each arrow indicates the direction of motion (specified as $u_k = \gamma^*(x_k)$) for the robot, from that particular state

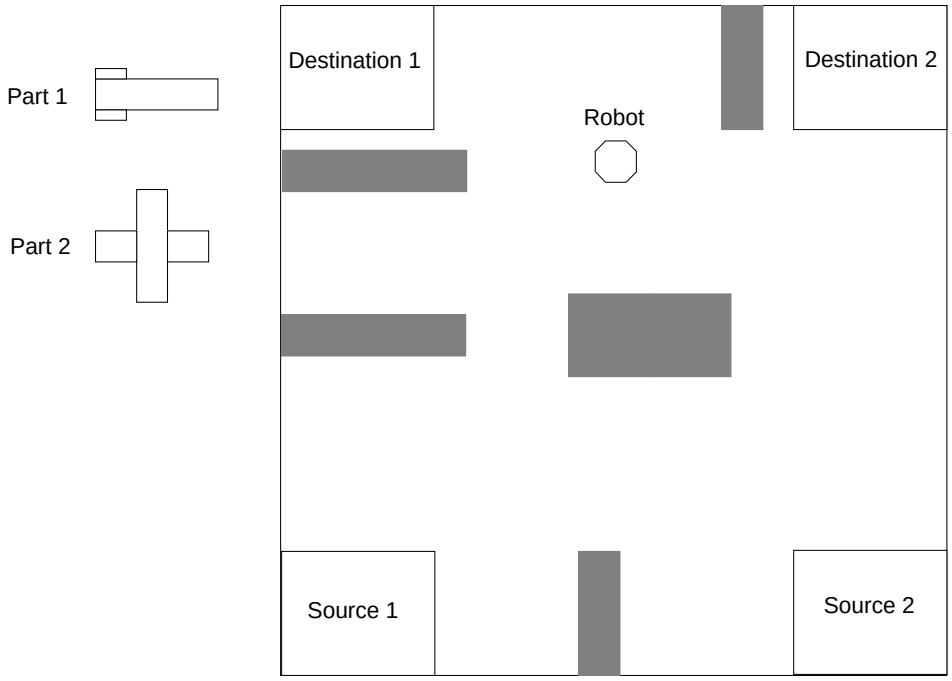


Figure 3.20 A translating robot problem in which $P = 2$, $S = 2$, and $D = 2$.

location. The motion directions are shown at fewer state locations than appear in the machine implementation to add clarity to the figure. The places in which there are no arrows correspond to configurations in which the robot (or possibly the part) is in collision with a static obstacle.

Figure 3.22 presents a simulation of the robot in the workspace over a period of time, under the implementation of γ^*. A sample run is obtained by sampling an environment mode sequence, $\mathbf{m}$, from the Markov process, to obtain a trajectory in X. The beginning of the trajectory is depicted in Figure 3.22(a), and it concludes in Figure 3.22(l). To save space in the figure, many frames are superimposed, and a new picture is shown each time the environment mode changes. This illustrates the behavior of the robot as it responds to a sample from the stochastic process. The first column of Figure 3.22 corresponds to execution during the NR mode. The second column corresponds to modes in E_w, and the final column corresponds to modes in E_c. In the last two columns, the source and

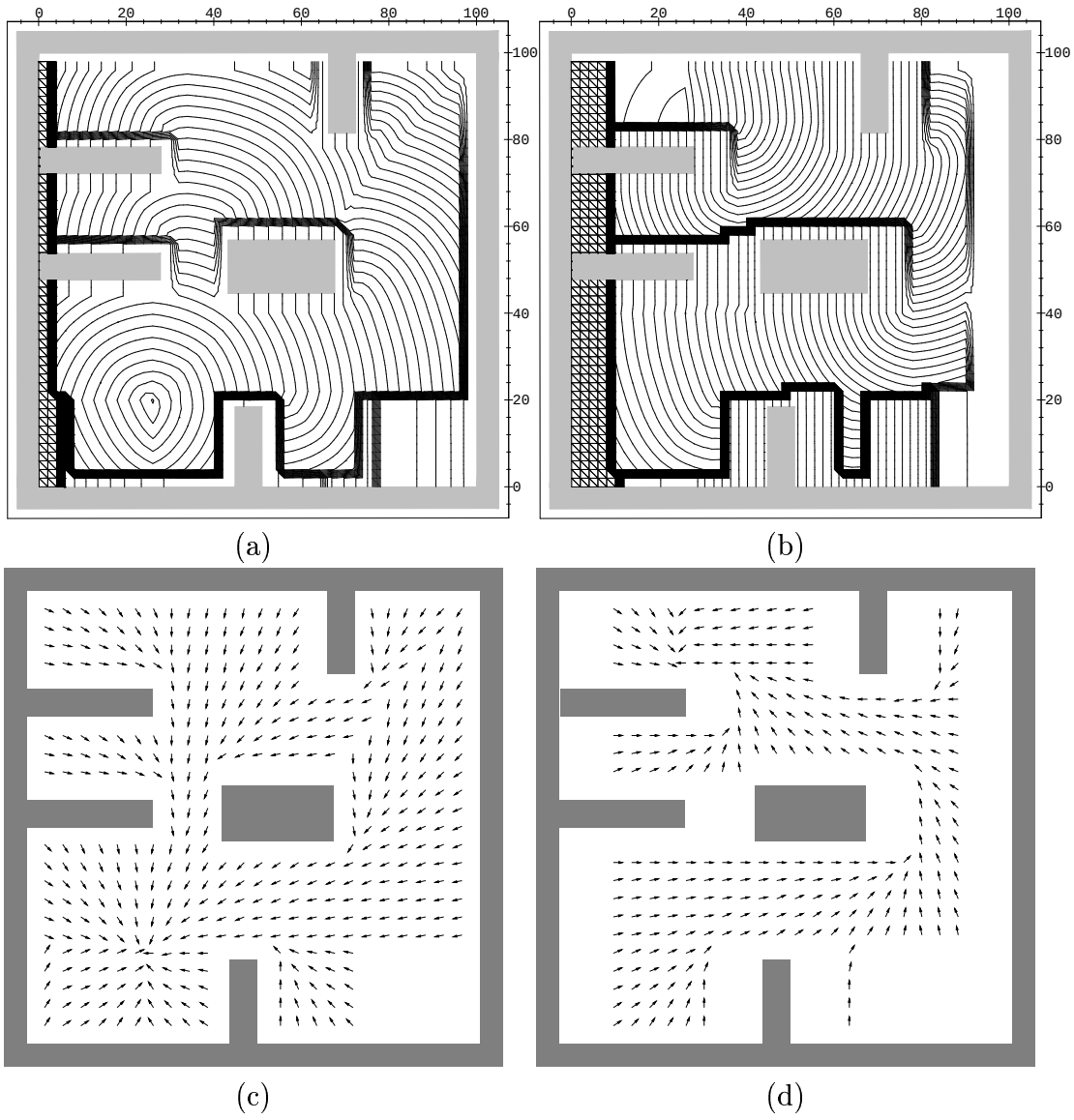


Figure 3.21 (a) Level-set contours of the cost-to-go function for $e = \langle 1, 1, 1, W \rangle$; (b) the contours for $e = \langle 1, 1, 1, C \rangle$; (c) the optimal actions as a vector field for $e = \langle 1, 1, 1, W \rangle$; (d) the optimal actions for $e = \langle 1, 1, 1, C \rangle$.

destination regions that correspond to the issued request are shaded. In the final column, the part that is carried by the robot is shaded in black.

At least two interesting behaviors are found in this solution. When the environment mode is NR, the robot moves to a location in the lower portion of the workspace. This behavior naturally occurs through the optimization of the criterion. To reduce the expected time to deliver a part that might appear, it is best for the robot to wait near sources while there are no requests. This corresponds to reducing the setup time in a scheduling system, and is hence a preferred behavior for the robot. Note also how the changing geometry affects the trajectory of the robot. In Figures 3.22(b) and 3.22(d), the robot does not carry a part and is able to move through a narrow opening. However, in Figure 3.22(c), the robot carries a part, and consequently must take a longer route to reach the destination.

For the remaining problems in this section, we will show figures similar to Figure 3.22 and which indicate the sample path under the implementation of the optimal strategy. Figure 3.23 involves a translating robot problem in which there are six parts, four sources, and three destinations. In addition, Destination 1 has two disconnected components; hence, the robot must choose the best delivery point in terms of loss. For this problem there are 72 different kinds of requests (which are equally likely to occur), resulting in 145 environment modes. Figure 3.24 shows a sample of the execution under γ^*.

Note the behavior of the robot with respect to the disconnected components of Destination 1. At the start of the time period captured in Figure 3.24(h), the robot receives a request to move Part 6 from Source 3 to Destination 1. The robot picks up the part from Source 3 and chooses to deliver it to the lower component of the Destination 1 (Figure 3.24(i)). This behavior was based on the computation of the optimal strategy for that particular position of the robot in the NR mode. In this example, the robot was allowed only to translate; this enabled us to study a more complex assembly situation involving

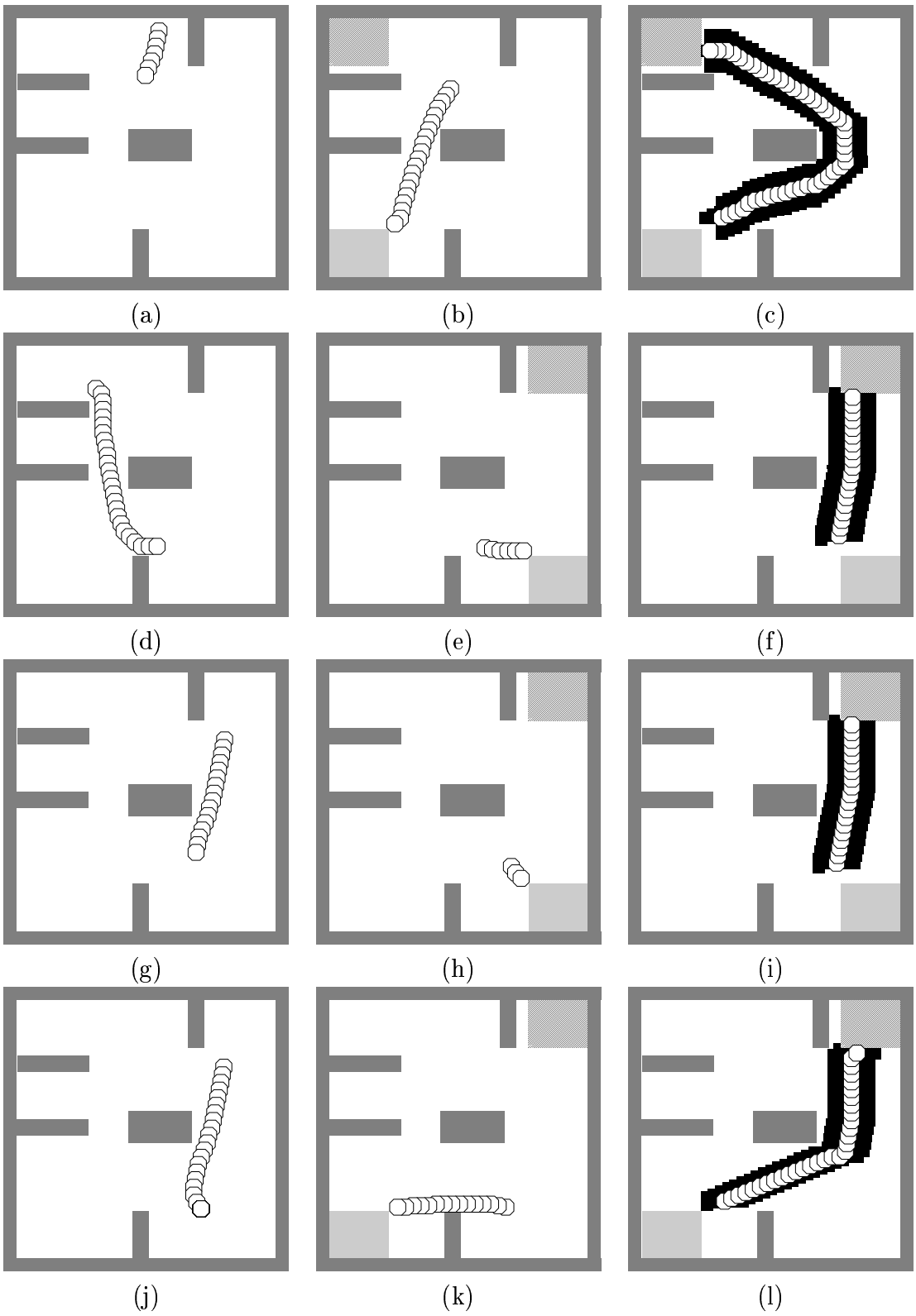


Figure 3.22 A simulation result under the implementation of γ^*.

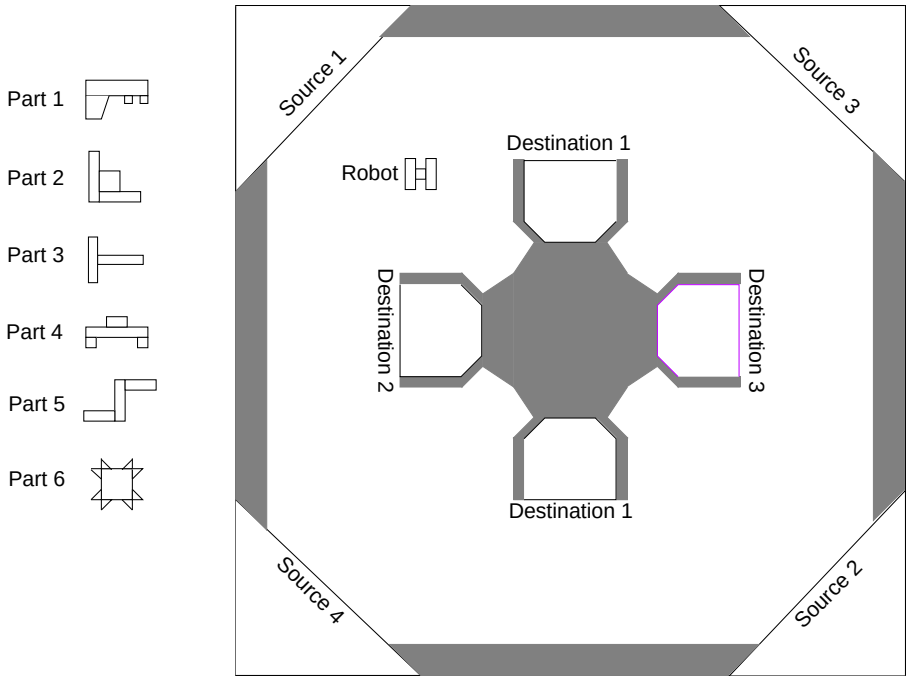


Figure 3.23 A translating robot problem in which $P = 6$, $S = 4$, and $D = 3$, with one of the destinations having two components. The first destination has two connected components.

a fairly large number of environment modes (145). In the next problem we extend the model to allow the robot to rotate, resulting in a three-dimensional configuration space.

Figure 3.25 shows the assembly situation in which there is a rotating robot, one part, two sources, and two destinations. We assume that the robot can rotate in place, or translate along its axis of orientation.

The constrained rotation motion model that was used for the example in Figure 3.13 was used for this case. Figure 3.26 shows a sample of the execution under γ^*. Because of the obstacle arrangement and because the part that the robot could carry is large relative to the opening, the optimal position of the robot in the NR mode (Figure 3.26(a)) is important since it can significantly affect the carrying time when the request arrives. Under the "tight" free configuration space that results for motion planning The importance of optimizing the expected performance that accounts for the probabilities

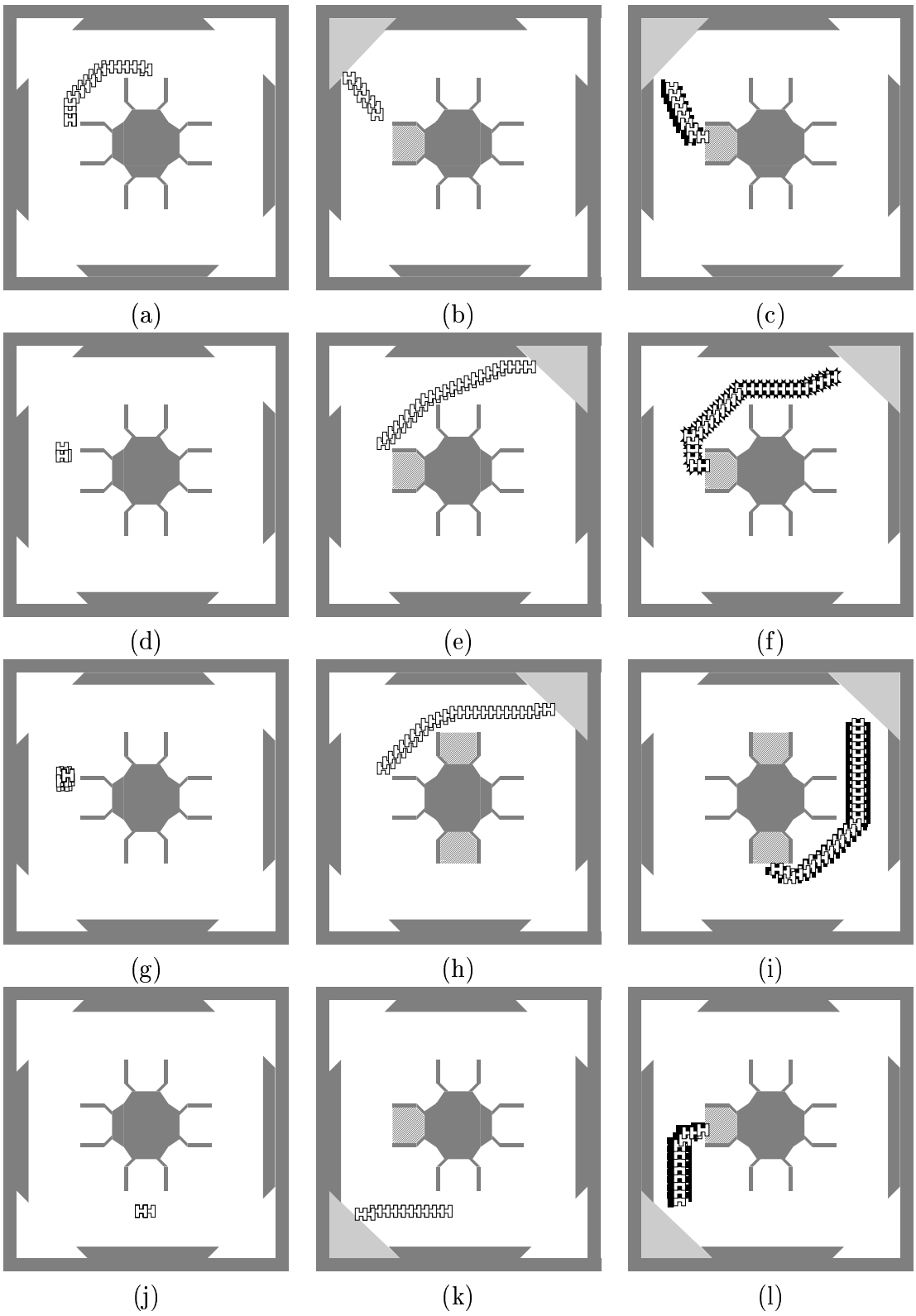


Figure 3.24 A simulation result under the implementation of γ^*.

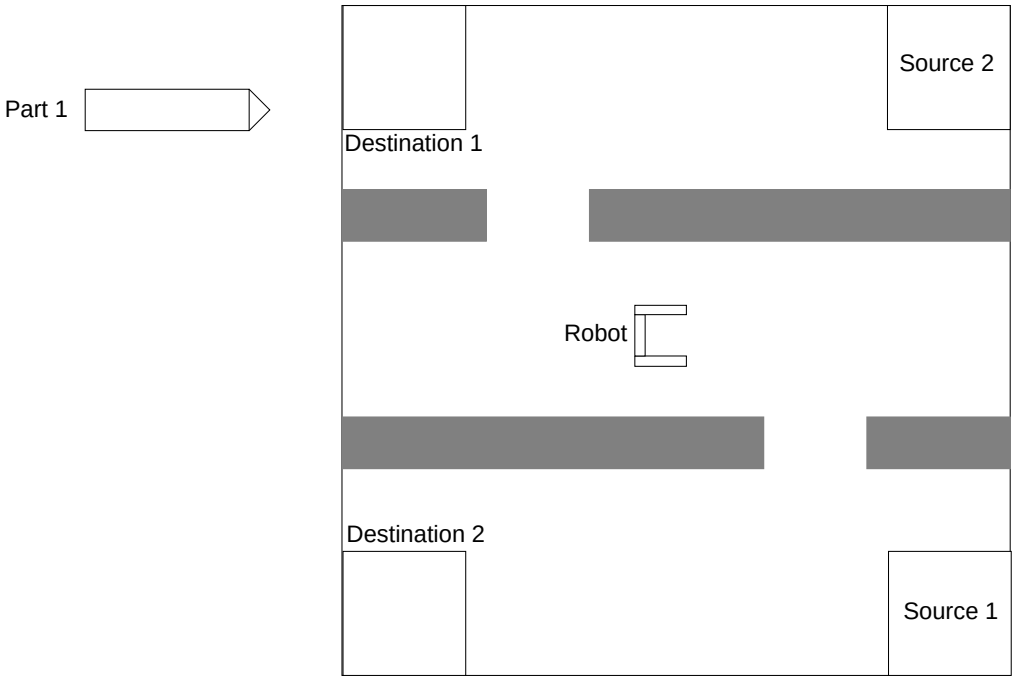


Figure 3.25 A rotating rigid robot problem in which $P = 1$, $S = 2$, and $D = 2$.

of the different requests can be appreciated. In this problem, there are four possible requests (p, s, d combinations) and nine environment modes.

In the three problems discussed so far, the stochastic model of the assembly process was defined in terms of the transition probabilities from the NR mode of the robot. In the next problem, we discuss an alternative stochastic model that defines the transition probabilities with respect to the destination regions. The specific feature that this model induces on the behavior of the robot, is the ability to change destinations *while* it is already carrying a part. This is illustrated in results of the specific problem described next.

Figure 3.27 helps illustrate the assembly situation that involves a translating robot with two parts, one source, and three destinations. For this problem, the destination is allowed to change while the robot is carrying a part. The probability that the destination will change in a given stage is 0.02. The probability for changing to each of the other

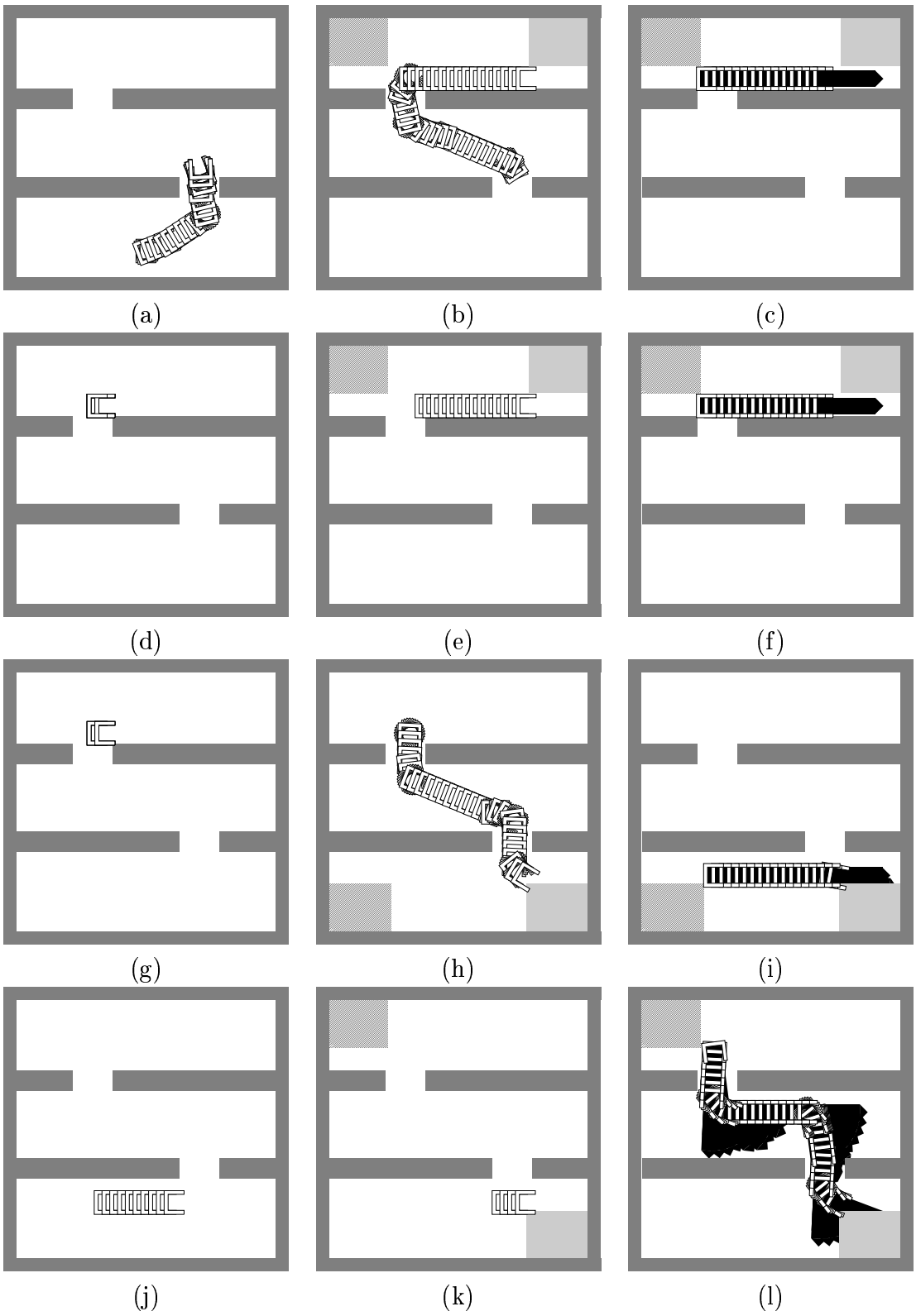


Figure 3.26 A simulation result under the implementation of γ^*.

two destinations is 0.01. The robot models the changing destination probabilistically as discussed in Section 3.6.1, and the optimal strategy causes the robot to respond to the change. Figure 3.28 shows a sample of the execution under γ^*. Figure 3.28 shows a sample under the implementation of γ. The destination changes during execution, as depicted in Figure 3.28(c) and in Figure 3.28(g). In both cases, the robot immediately responds by delivering the part to the new destination. This kind of assembly situation can arise when there is on-line monitoring of the assembly process and the robot has access to the current demand at a particular destination at any given time. For example, suppose the same part is needed by two destination regions $D1$ and $D2$ at a given time. The scheduler schedules the part to be delivered to $D1$, but an error causes a delay in the previous operation, and $D1$ is not ready for that part while the robot is in the process of carrying it. This would be detected by on-line plan monitoring, and the part would be rescheduled to arrive at $D2$. The probabilistic modeling of such an assembly situation helps in improving the gross-motion planning.

3.6.2.2 Manipulator simulations

In this section, we show how the principles in this chapter also apply to motion planning for manipulators performing assembly operations. Several additional concerns must be addressed that pertain to collision detection and the state transition distribution. Each manipulator is described by a set of links that are connected by rotating joints, and the final link contains an end-effector that can grasp or ungrasp an object. The configuration space is generated by taking the Cartesian product of the real-valued intervals that correspond to joint angles.

For this context we need to replace only $\mathcal{A}(\mathbf{q})$ in Equations (3.18), (3.19), and (3.20) with the union of the transformed end-effector and all the links. Hence, we require that the entire manipulator avoids collision with static obstacles. To define the source and

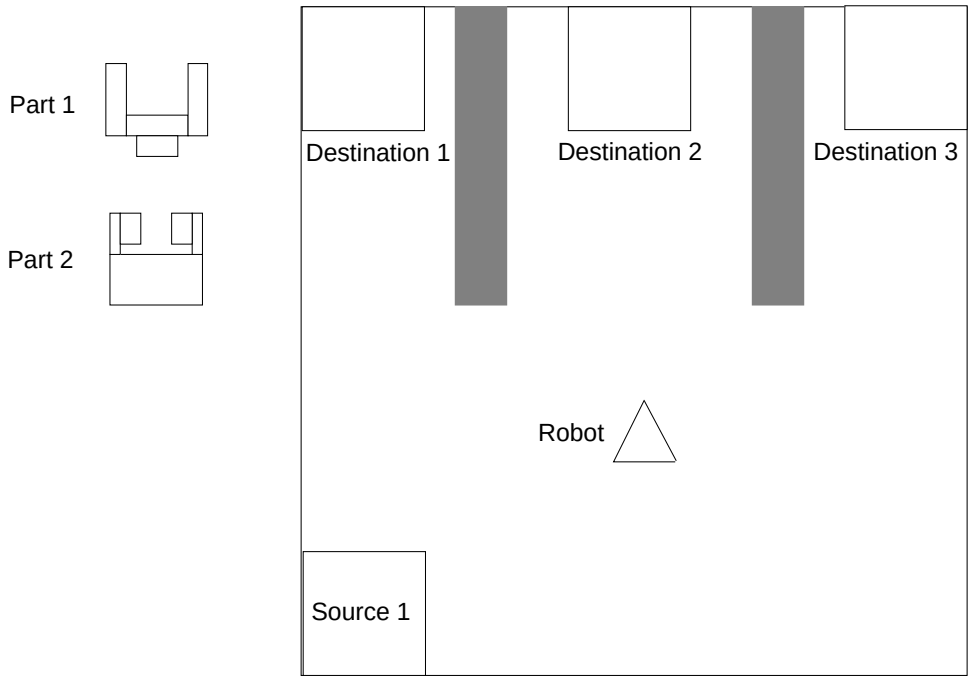


Figure 3.27 A translating robot problem in which $P = 2$, $S = 1$, and $D = 3$, and the destination of a request is allowed to change stochastically.

destination regions in the configuration space, we consider only the end-effector as the robot.

Each joint of the manipulator can be independently controlled. Thus, we define the motion strategies directly in terms of the joints, instead of considering their representation in the workspace. The collision detection for the articulated hand in our implementation is done in terms of the coordinate space of the workspace. To define the action space, at the i^{th} joint, we allow one of three possibilities: move clockwise, move counterclockwise, or remain motionless. For an n-link manipulator, there are 3^n+1 actions. A state transition distribution is straightforward to specify because each of the first $n-1$ coordinates corresponds to a unique joint angle. For the examples that we consider, the planar robot manipulator has redundant degrees of freedom. This is because that even though the robot has three degrees of freedom, the goals are *positional* in terms of the source and destination regions.

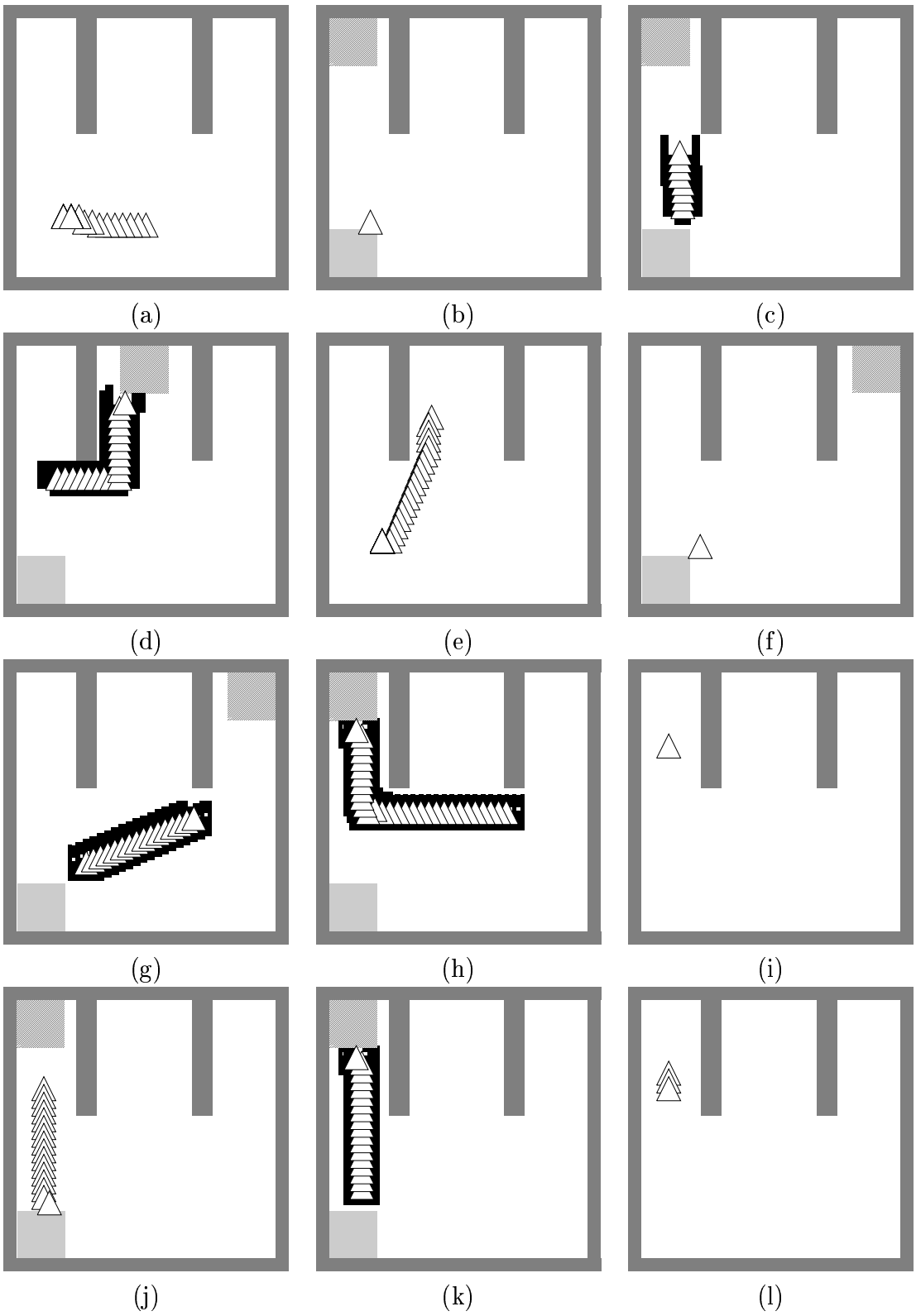


Figure 3.28 A simulation result under the implementation of γ^*.

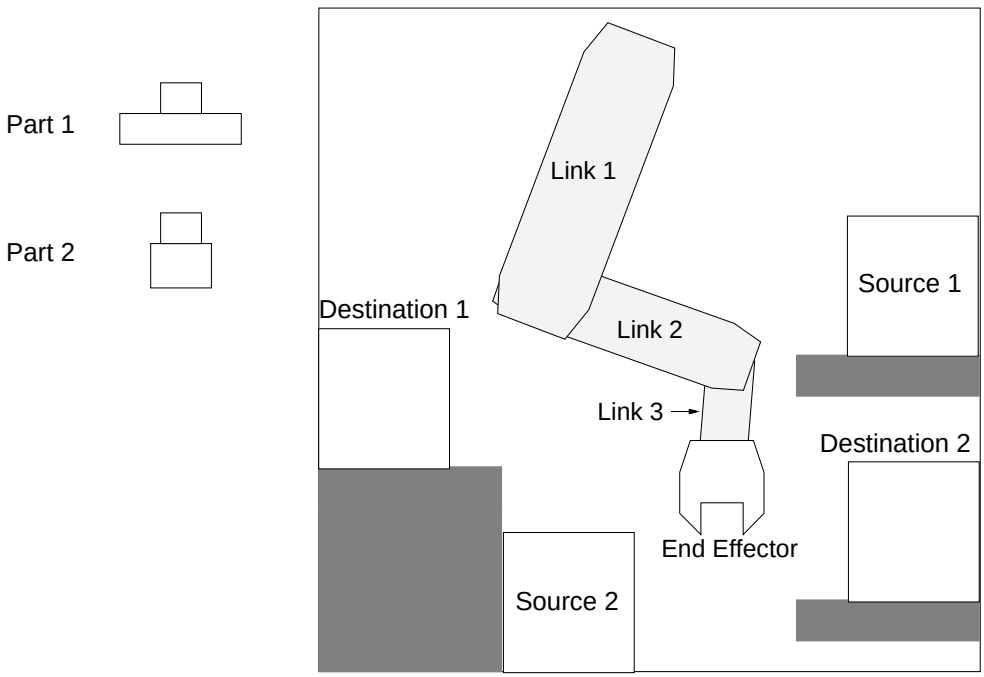


Figure 3.29 A three DOF manipulator with a constrained, rotating end-effector is in a workspace in which $P = 2$, $S = 2$, and $D = 2$.

For the first manipulator problem, there are two parts, two sources, and two destinations (see Figure 3.29). There are three links that move in the plane, and an end-effector that maintains a constant orientation. The figure can be considered as a side view of a problem in which objects are to be moved from trays that exist at different levels. Joint limits prevent joints from executing circular motions. Figure 3.30 shows a sample of the execution under different requests and under some of the 17 possible assembly modes. The third column shows the part being "carried" to the destination region with the transition to the fine-motion planning being defined in terms of contact of the end-effector with the destination region. An enclosure condition could have alternatively been defined.

For the second manipulator problem, there are one part, two sources, and four destinations (see Figure 3.31). One of the sources has two disconnected components. There are three links that move in the plane. The figure can be considered as a top view of

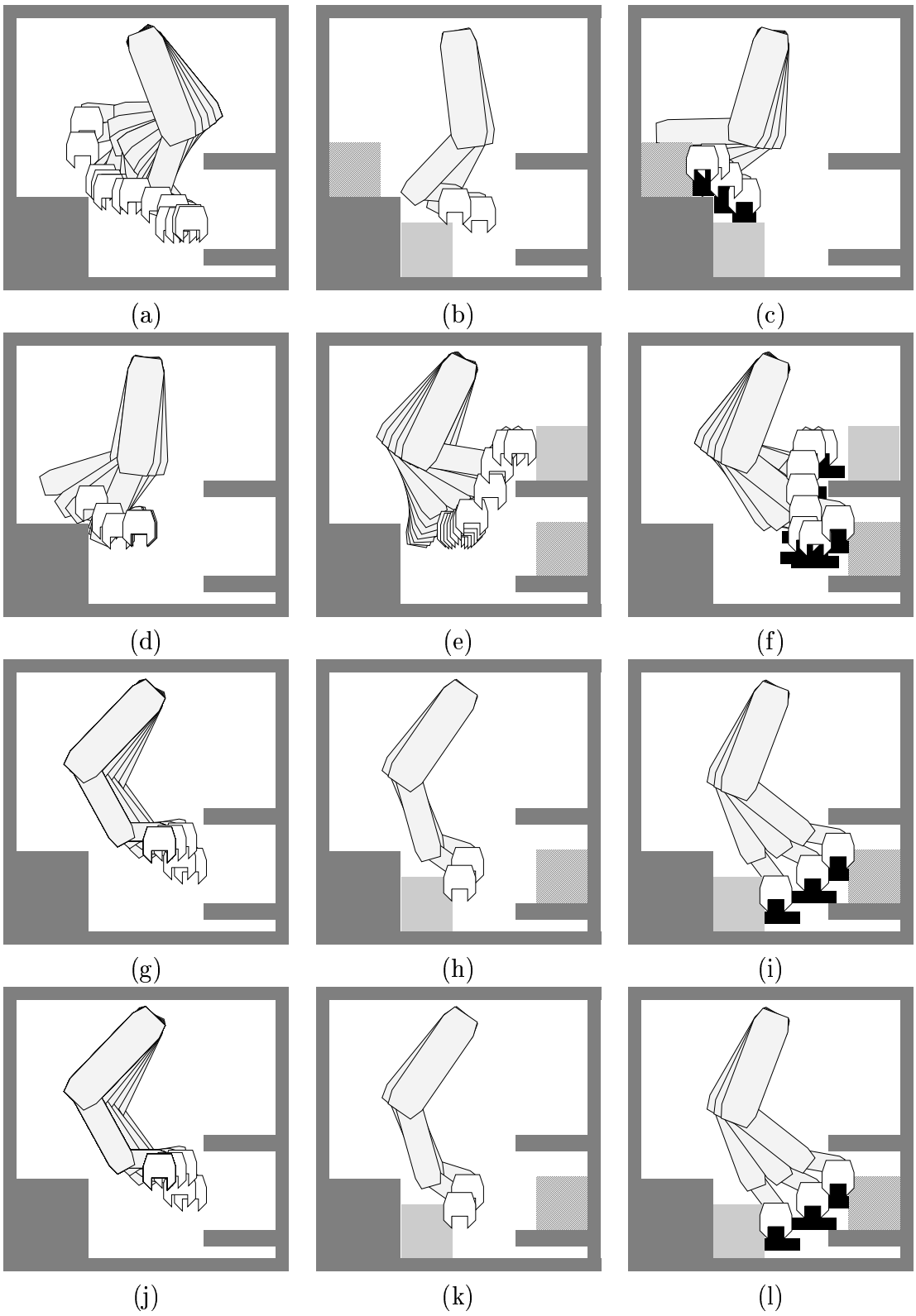


Figure 3.30 A simulation result under the implementation of γ^*.

a problem in which objects are to be moved between locations on a planar surface. We also have fixed limits for each joint. Figure 3.32 shows a sample of the execution. Note the interesting behavior of the robot induced by the fact that there are two disconnected components for Source 1. Thus, every time a request arrives involving Source 1, the strategy of the robot varies depending on its current position and the destination region that is also the part of the request. For example, in the segment of its execution captured in Figure 3.32, there were three requests involving Source 1 (this is shown in terms of the shaded regions of Figure 3.32(e), 3.32(h) and 3.32(k)). For the first such request (second row), the robot chooses to pick the part from the upper component of Source 1, although the robot was closer to the lower component of Source 1 when the request arrived. This was because the corresponding destination was closer to the upper component. In the other two cases involving Source 1 (see the third and fourth rows of Figure 3.32), the robot chose the lower component instead. Such behaviors illustrate the utility of considering motion planning in the context of assembly by an appropriate model of the process that drives its behavior over time.

3.7 Additional Models and Applications

This section presents several additional models and applications that illustrate the flexibility and extendibility of our approach. Section 3.7.1 discusses an extension in which the robot does not receive perfect information about the current environment mode. This form of uncertainty can be combined with the changing environment to yield strategies that are conditioned on sensor observations made by the robot. Section 3.7.2 discusses an extension that incorporates any time-varying, completely predictable aspect of the motion planning problem into the motion strategy, which results in a nonstationary optimal

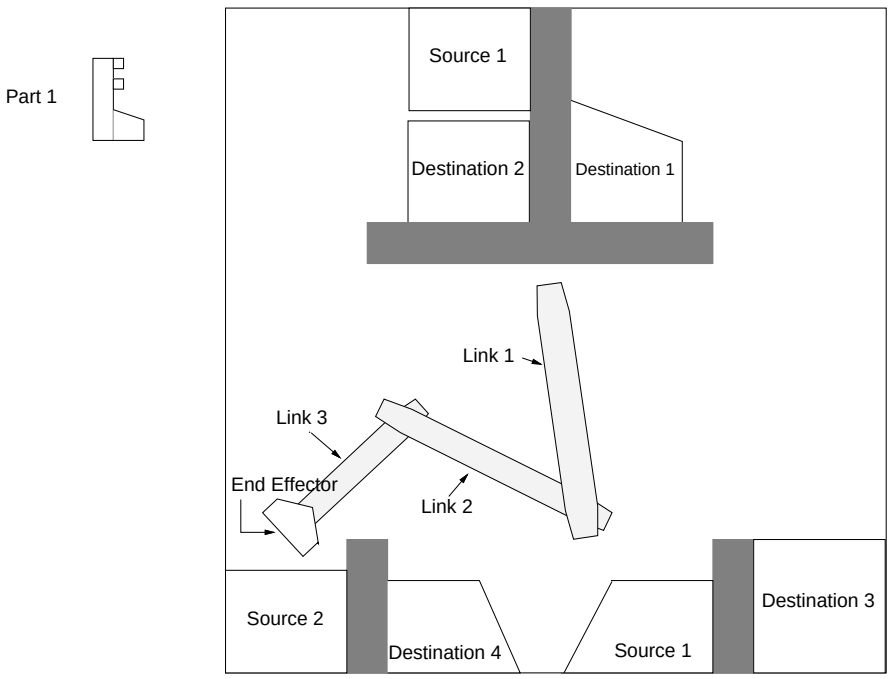


Figure 3.31 A three DOF manipulator is in a workspace in which $P = 1$, $S = 2$, and $D = 4$. The first source has two disconnected components.

strategy. Section 3.7.3 discusses how nondeterministic uncertainty representations can be used instead of probabilistic representations.

3.7.1 Imperfect environment information: incorporating uncertainty in environment sensing

It has been assumed so far that at stage k the robot knows the environment mode, e_k. In terms of the concepts from Chapter 2 this is equivalent to the case of perfect state information. In Chapter 2, the state space encoded only configuration information; however, in this chapter, the state space also includes environment information. The information space concepts from Chapter 2 can also be applied to the case of imperfect information about the environment. This leads to a treatment of environment sensing uncertainty, which is briefly discussed in this section.

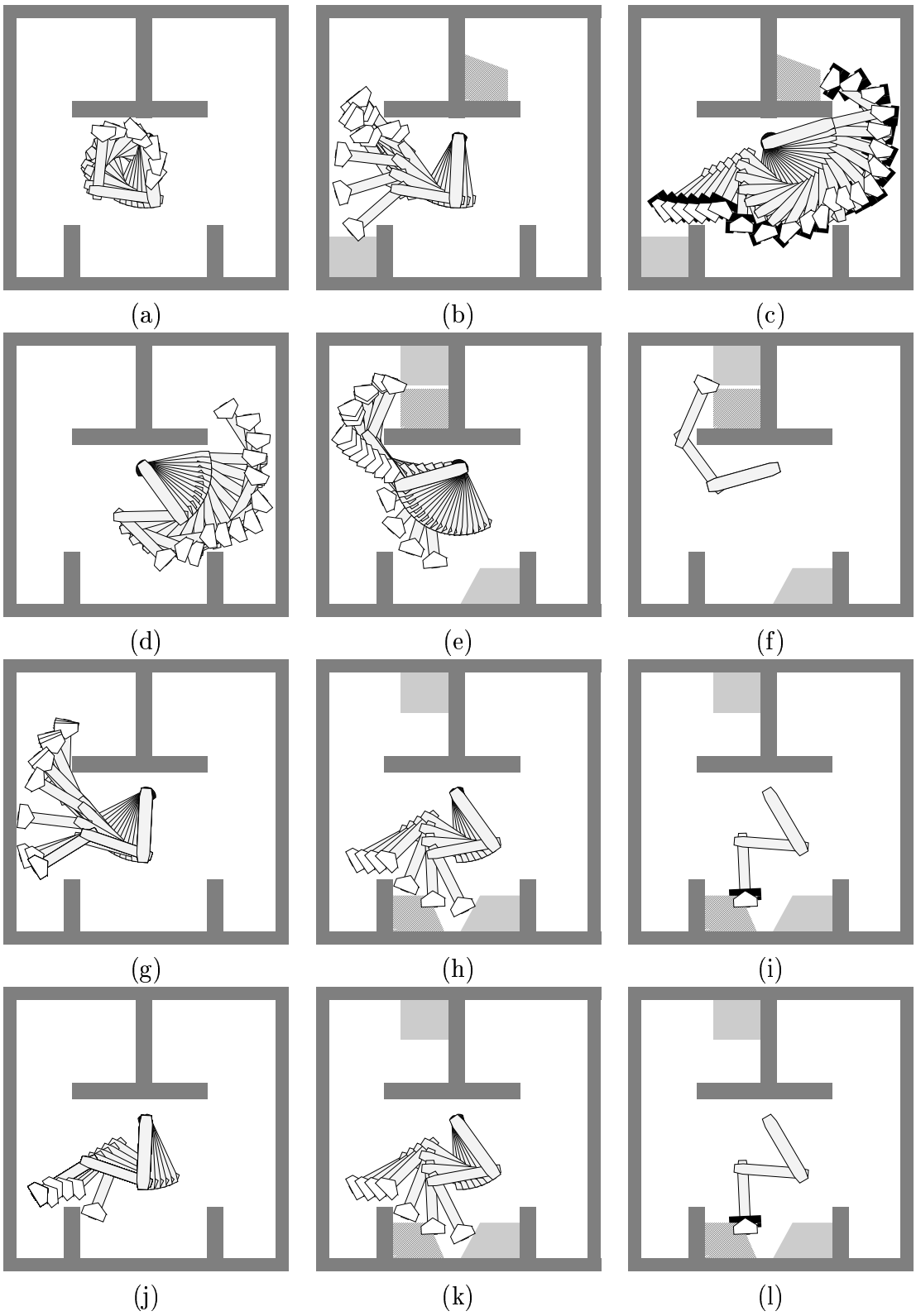


Figure 3.32 A simulation result under the implementation of γ^*.

Suppose the robot is equipped with a sensor that produces an observation o_k at each stage, $k \in \{1, \ldots, K\}$. We assume that a noise or error model for the sensor can be specified as $P(o_k|e_k)$. This characterizes the observations that are likely to be made for a given environment mode. The form $P(o_k|e_k)$ is typically used in a variety of robotics applications that involve statistical sensor error [79], [80] and, in general, for stochastic control theory [105]. Its form is similar to the observation model, $p(y_k|x_k)$, that was used in Chapter 2. We could also condition the observations on configuration to obtain $P(o_k|e_k, \mathbf{q}_k) = P(o_k|x_k)$.

We begin with a prior probability distribution over E, denoted by $P(e_1)$ (which could, for example be uniform). We next develop an incremental computation method that determines the posterior probability distribution of e_k for each k and incorporates the sensor observations. This method proceeds by induction, using $P(e_1)$ as the basis, and the transition from $P(e_k|o_k, \ldots, o_1)$ to $P(e_{k+1}|o_{k+1}, \ldots, o_1)$ as the inductive step.

If we have $P(e_k|o_k, \ldots, o_1)$, then before a new observation, the posterior distribution of e_{k+1} can be determined as

$$P(e_{k+1}|o_k, \ldots, o_1) = \sum_{e_k \in E} P(e_{k+1}|e_k) P(e_k|o_k, \ldots, o_1). \tag{3.22}$$

The new observation, o_{k+1}, can be incorporated to obtain

$$P(e_{k+1}|o_{k+1}, \ldots, o_1) = \frac{P(o_{k+1}|e_{k+1}, o_k, \ldots, o_1) P(e_{k+1}|o_k, \ldots, o_1)}{P(o_{k+1}|o_k, \ldots, o_1)} \tag{3.23}$$

in which

$$P(o_{k+1}|o_k, \ldots, o_1) = \sum_{e_{k+1} \in E} P(o_{k+1}|e_{k+1}, o_k, \ldots, o_1) P(e_{k+1}|o_k, \ldots, o_1). \tag{3.24}$$

By making appropriate substitutions above, and by reducing conditionals, we obtain

$$P(e_{k+1}|o_{k+1},\ldots,o_1) = \frac{P(o_{k+1}|e_{k+1}) \sum_{e_k \in E} P(e_{k+1}|e_k) P(e_k|o_k,\ldots,o_1)}{\sum_{e_k \in E} \sum_{e_{k+1} \in E} P(o_{k+1}|e_{k+1}) P(e_{k+1}|e_k) P(e_k|o_k,\ldots,o_1)}. \tag{3.25}$$

Equation (3.25) defines $P(e_{k+1}|o_{k+1},\ldots,o_1)$ in terms of the given probabilities: $P(e_{k+1}|e_k)$, $P(e_k|o_k,\ldots,o_1)$, and $P(o_{k+1}|e_{k+1})$. Hence, at each stage during the execution of a strategy, a new posterior distribution can be computed.

The next concern is to design an optimal strategy under imperfect environment information. We could define an *information state at stage k* as some subset:

$$\eta_k \subseteq \{u_1, u_2, \ldots, u_{k-1}, o_1, o_2, \ldots, o_k\}. \tag{3.26}$$

Recall from Section 2.3.3 that under probabilistic uncertainty, the information space can be considered as a function space of probability densities. Let $\mathcal{P}$ denote a function space of all probability distributions over E. Let $p_k \in \mathcal{P}$ denote the probability distribution over E obtained at stage k. The dimension of $\mathcal{P}$ is $|E| - 1$. A new state space for this problem can be defined as $X = \mathcal{C}_{free} \times \mathcal{P}$. Hence, at any given stage, the robot will be at some known configuration in $\mathcal{C}_{free}$, and we have a probability distribution for E that belongs to $\mathcal{P}$. This state space has dimension $n + |E| - 1$, in which n is the dimension of $\mathcal{C}_{free}$. A state transition distribution must be specified to determine x_{k+1} from x_k. The first n coordinates are given by the motion equation of the robot, as specified in Section 3.3.2. The state transition distribution in Section 3.3.2 required the environment transition probabilities, $P(e_{k+1}|e_k)$ and, analogously in this case, we are interested in $P(p_{k+1}|p_k)$. We also require the definition of a termination condition, which causes the robot to halt, because the goal region might depend on the environment mode, which is generally unknown. Using these components, the dynamic programming equation (3.11) can be applied to yield solutions.

To obtain $P(p_{k+1}|p_k)$ we first note that p_k represents the function $P(e_k|o_k, \ldots, o_1)$, and p_{k+1} represents the function $P(e_{k+1}|o_{k+1}, \ldots, o_1)$. The probability that p_{k+1} will be obtained in the next stage is equivalent to the probability that o_{k+1} will be observed. Therefore, we have

$$P(p_{k+1}|p_k) = P(o_{k+1}|o_k, \ldots, o_1) \tag{3.27}$$

which is given by (3.24).

One additional issue in the dynamic programming computations is that $\mathcal{P}$ must also be quantized and approximated. Considering the problem sizes that have already been computed, we can at least apply the current computation techniques to obtain optimal strategies for problems in which the dimension of $\mathcal{C}_{free}$ is two and $|E| = 2$.

3.7.2 Nonstationary motion planning problems

The strategies that have been considered up to this point are stationary in the sense that the robot actions depend only on the state. The optimal strategy for the robot does not depend on time, because the model components (such as the state transition distribution or the environment transition probabilities) do not depend on the particular stage index, $k \in \{1, \ldots, K\}$. In turns out that with little effort, the model components can be allowed to vary over time. This affords the opportunity to model many interesting problems, such as the incorporation of known moving obstacles. The tradeoff, however, is that more storage is required for representation of the optimal strategy (it is one dimension larger with the inclusion of time).

Figure 3.33 shows an example problem that results in a nonstationary strategy. In addition to a doorway that produces two environment modes, there is a moving obstacle in the workspace. It is assumed that the trajectory of this obstacle is known to the robot.

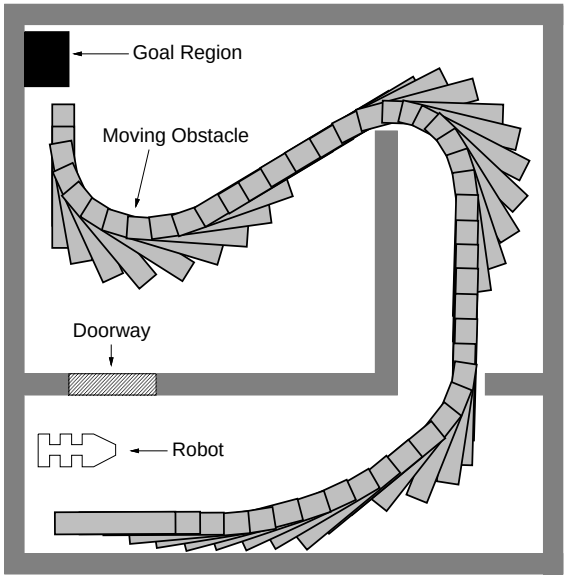


Figure 3.33 A motion planning problem that involves a doorway and a moving obstacle that has a known trajectory.

Suppose the objective is to bring the robot to the goal region in minimal time without colliding with the doorway or the moving obstacle.

We briefly describe the general time-varying components that can be defined to yield nonstationary solutions. Suppose the workspace contains obstacles $\mathcal{B}_1(t), \ldots, \mathcal{B}_q(t)$ that may possibly be in motion. This results in a time-varying free configuration space, $\mathcal{C}_{free}(t)$ [109]. To handle discrete time, at each stage k, we define a *stage-dependent* free configuration space

$$\mathcal{C}_{free}[k] = \bigcap_{t \in [(k-1)\Delta t, k\Delta t)} \mathcal{C}_{free}(t). \tag{3.28}$$

In addition, we can have moving dynamic regions $\mathcal{D}_1(t), \ldots, \mathcal{D}_m(t)$. In configuration space each of these becomes $\mathcal{CD}_i^c(t)$ or $\mathcal{CD}_i^e(t)$, and in the state space the dynamic regions are $X_1(t), \ldots, X_m(t)$. As done in (3.28), we can similarly define $X_1[k], \ldots, X_m[k]$ to be *stage-dependent* dynamic X-regions. To obtain the appropriate loss functional, we simply

replace (3.8) with

$$l_k(x_k, u_k) = \begin{cases} 0 & \text{If } x_k \in X_G \\ \\ c_u + \sum_{i=1}^{m} [c_i I_{X_i[k]}(x_k) + c_i' I_{X_i^c[k]}(x_k)] & \text{Otherwise} \end{cases} . \quad (3.29)$$

with the addition of an explicit dependency on k.

Optimal strategies can be computed by slightly modifying the algorithm in Section 3.4. These extensions do not increase the state space dimension. After each iteration of the dynamic programming, however, we must recall the optimal actions. The final stage index $K + 1$ is more significant in this case, because we do not expect the algorithm to terminate by yielding a stationary strategy; the algorithm terminates when $k = 1$. The optimal strategy will be $\gamma^* : X \times \{1, \ldots, K\} \rightarrow U$. The action taken at stage k is given by $u_k = \gamma_k^*(x_k)$.

In addition to the time-varying components discussed above, additional components can vary with time. By allowing the environment transition probabilities to vary, many more statistical processes can be modeled. For example, it might be known that the workspace is more likely to become hazardous after some prescribed time, or become increasingly more likely to be hazardous over time. We can also allow the goal region to move over time, to obtain $X_G[k]$. In this case, the robot must intercept the moving goal as a terminating condition for the strategy (as considered in [120]).

3.7.3 Nondeterministic uncertainty

In this chapter, uncertainties have been represented probabilistically; however, in many cases, a nondeterministic (or bounded-set) representation may be preferable. Expected case analysis is then replaced by *worst-case* analysis. Reasons for using this representation include modeling ease and problems that involve an extremely high risk of fail-

ure. As discussed in Chapter 2, nondeterministic and probabilistic representations could be easily interchanged when modeling configuration-predictability and configuration-sensing uncertainties. With environment-predictability and environment-sensing uncertainties, probabilistic representations can naturally be replaced by nondeterministic representations. In this section, we briefly provide a nondeterministic characterization of planning under environment-predictability uncertainty.

The environment process must first be redefined. Nature in this case is modeled nondeterministically. Instead of a Markov process with transition probabilities, let $F_k^e(x_k, u_k) \subset E$ be the set of possible next environment modes, when the system is at state x_k and action u_k is chosen. The state transition distribution is then replaced by $F_k(x_k, u_k)$, which represents the set of possible next states.

Using worst-case analysis, the ideal choice for a strategy satisfies

$$\inf_{\gamma \in \Gamma} \check{L}(x_1, \gamma) = \inf_{\gamma \in \Gamma} \sup_{\mathbf{e} \in \mathbf{E}} L(x_1, \gamma, \gamma^\theta) \tag{3.30}$$

for all $x_1 \in X$. Due to the nondeterministic uncertainty in prediction, $\mathbf{E}$ represents the set of possible environment mode sequences that could be obtained. This indicates that from any initial state, the strategy will guarantee the least possible loss given the worst-case environment mode sequence. The principle of optimality can be applied to successively compute $\check{L}_k^*(x_k)$, yielding an optimal strategy.

3.8 Conclusions

We have presented a method for analyzing and determining optimal robot motion strategies under a partially predictable, changing environment. This method is general and flexible for characterizing environment-predictability uncertainty. The concept of optimal *motion strategies* under performance criteria provides a useful characterization

of the desired behavior for the robot in this context. In addition, we have provided a computational approach, based on the principle of optimality, that determines optimal solutions to many interesting motion planning problems under environment-predictability uncertainty. The variety of computed examples that were presented in Sections 3.5 and 3.6.2 helps substantiate these conclusions. Section 5.3 will discuss how environment-predictability uncertainty can be combined with the other sources of uncertainty, in a unified approach.

CHAPTER 4

MOTION PLANNING FOR MULTIPLE ROBOTS

4.1 Introduction

This chapter describes two contributions to geometric motion planning for multiple robots: (1) motion plans are determined that simultaneously optimize an independent performance criterion for each robot; and (2) a general spectrum is defined between decoupled and centralized planning, along which we introduce the concept of coordination along independent roadmaps. By considering independent performance criteria, we introduce a form of optimality that is consistent with concepts from multiobjective optimization and game theory research. Previous multiple-robot motion planning approaches that consider optimality combine several individual criteria into a single criterion. As a result, these methods can fail to find many potentially useful motion plans. We present implemented, multiple-robot motion planning algorithms that are derived from the principle of optimality, for three problem classes along the spectrum between centralized and decoupled planning: (1) coordination along fixed, independent paths; (2) coordination along independent roadmaps; and (3) general, unconstrained motion planning. Several computed examples are presented for all three problem classes to illustrate the concepts and algorithms. This chapter does not consider aspects of uncertainty and is, therefore,

complementary to the material in Chapters 2 and 3. Section 5.3 discusses issues that result from the combination of multiple robots with aspects of uncertainty. Portions of the work in this chapter were also presented in [114].

The direct consideration of independent performance criteria differs from previous approaches to multiple-robot motion planning. Typically, if optimality is considered, performance measures for the individual robots are combined into a single scalar objective. For example, in [19], [177] the objective is to minimize the time taken by the last robot to reach the goal. In [193], the performance measures are aggregated to yield a single objective. When individual objectives are combined, certain information about potential solutions and alternatives is lost (for general discussions, see [85], [164], [205]). For example, the amount of sacrifice that each robot makes individually to accomplish its goals is not usually taken into account. It might be that one robot's goal is nearby, while the other robot has a distant goal. Combining the objectives by scalarization might produce a good plan for the robot that has the distant goal; however, the execution cost for the other robot would hardly be considered.

Consider the motion planning problem in Figure 4.1 as an illustrative example. Assume that both robots are capable of translating at the same fixed speed and can stop or start instantaneously, and that we want to minimize the time to achieve goals while avoiding collisions. If, for the moment Robot 2 is ignored, then Robot 1 has two choices that produce equivalent cost: using the lower corridor or the upper corridor. From the perspective of Robot 1, both paths seem equivalent, but choosing the upper corridor is clearly worse for Robot 2. Robot 2 has only a short distance to travel, but must wait if Robot 1 enters the upper corridor (assuming for the moment that Robot 1 has priority). If we consider minimizing the time required for the last robot to reach the goal, we are still left with two choices that may seem equivalent; however, one choice is much more costly for Robot 2. Aggregating the total times for the two robots would provide a cri-

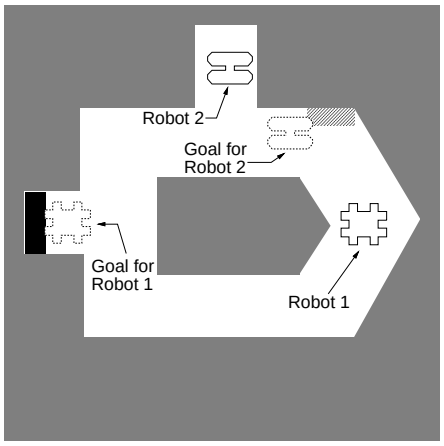


Figure 4.1 An illustrative example of potential difficulty in defining a combined objective.

terion that prefers Robot 1 taking the lower corridor, which is the natural preference. However, if we vary the initial position of Robot 1 slightly, so that the path through the upper corridor is shorter, then Robot 1 prefers to take the upper corridor, which suddenly conflicts with the preference of Robot 2. For a problem in which there is no natural priority, there seem to be two viable solutions. For these types of problems, the methods that we propose characterize the solution alternatives, by generating a small set of motion plans that are better than or equivalent to all others. An approach that minimizes a single, scalar objective will not determine these alternatives.

More formally, given a vector of independent objective functionals, we show that there exists a natural partial ordering on the space of motion plans, yielding a search for the set of *minimal* motion plans. For any other motion plan that can be considered, there will exist at least one minimal plan that is clearly better or equivalent, and the set of all minimal motion plans is typically small. Hence, our approach filters out all of the motion plans that are not worth considering, and presents a small set of reasonable alternatives. Within this framework, additional criteria, such as priority or the amount of sacrifice one robot makes, can be applied to automatically select a particular motion plan. If the

same tasks are repeated and priorities change, then a different minimal plan would be selected, as opposed to re-exploring the space of motion plans.

As further justification of this form of optimality, we show that the minimal strategies are consistent with certain optimality concepts from multiobjective optimization (e.g., [85], [164], [205]) and dynamic game theory (e.g., [6], [141]) literature. Within these subjects, considerable effort has yielded many solution concepts that pertain to optimal decision making when confronted with multiple objectives. Such issues as the number of decision makers, the degree of cooperation between decision makers, and prioritization, become important for determining which type of optimality concept is appropriate. These issues are relevant for a robotics context, and it is important to justify whether a solution concept is appropriate under the modeling assumptions.

We now present an overview of related previous work in motion planning for multiple robots. Approaches are often categorized as *centralized* or *decoupled*. A centralized approach typically constructs a path in a composite configuration space, which is formed by combining the configuration spaces of the individual robots. A decoupled approach typically generates paths for each robot independently, and then considers the interactions between the robots. The suitability of one approach over the other is usually determined by the tradeoff between computational complexity associated with a given problem and the amount of completeness that is lost.

With a centralized approach, a multiple-robot planning problem is viewed as a single-robot planning problem by considering the Cartesian product of the individual configuration spaces of the robots. As an example, the problem of planning the motion of N disks in the plane was solved in [167] by performing a critical curve analysis on the configuration spaces of the disks and planning a collision-free path through free cells in the composite configuration space. Collision-free motion control is generated for two specific, constrained manipulators by modeling the problem as a noncooperative game in

[3]. Randomized search is combined with a potential field in [11], [12], to centrally plan the motion of several translating objects in the plane.

A variety of decoupled planning approaches exist. *Prioritized planning* has been proposed, in which plans are generated sequentially for robots of decreasing priority, given the plans of the higher-priority robots [60]. In [194], prioritization is combined with a potential field that is defined on a time-varying configuration space. Issues for selecting priorities that improve performance are discussed in [29].

Several decoupled approaches involve the construction of a *coordination diagram*, which represents places along the robot paths in which a collision might occur. A collision-free path is determined in the diagram, which leads to a successful motion plan. In [137] robot paths are independently determined, and a coordination diagram is used to plan a collision-free trajectory along the paths. The coordination space for two manipulators is analyzed in [19], [33]. Scheduling issues are studied through the use of coordination diagrams in [119], and additional multiple-robot scheduling issues are presented in [103].

In addition to introducing multiple-objective optimality to the multiple-robot geometric motion planning, we expand the traditional view of centralized and decoupled planning by considering these two approaches as opposite ends of a spectrum. An approach that weakly constrains the robot motions before considering interactions between robots could be considered to lie somewhere in the middle of the spectrum. By utilizing this view, we show that many useful solutions can be obtained by constraining the robots to lie on independent, configuration space roadmaps. A roadmap is a one-dimensional network of curves, which can be constructed by one of many methods. The *visibility graph* approach generates a roadmap by connecting certain vertices of the boundary of the free configuration space, $\mathcal{C}_{free}$, and is primarily suitable for two-dimensional polygonal $\mathcal{C}$-space planning problems [109]. The topological *retraction* operation has been used in a roadmap generation approach that continuously retracts $\mathcal{C}_{free}$ onto its Voronoi dia-

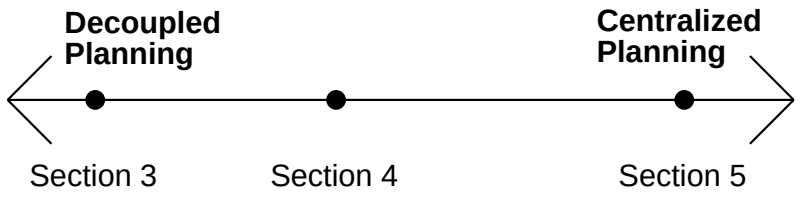


Figure 4.2 Differing degrees of centralization.

gram [138]. Other roadmap methods are described in [23], [31], [36]. A special form of coordination on a common network of paths has been suggested in [174].

Section 4.2 provides a general problem description, the models that are used, and the general concepts associated with multiple-robot optimality. Each of Sections 4.3 through 4.5 provides a method for multiple-robot motion planning, with increasing levels of centralization (see Figure 4.2). Section 4.3 presents a method that computes the minimal solutions, assuming that the robots are confined to move along fixed paths. Section 4.4 presents a method for determining minimal solutions, when each robot is allowed to traverse an independent, configuration-space roadmap. Section 4.5 presents a method for determining motion plans without imposing additional constraints on the motions of the robots; however, the computational cost is increased. Sections 4.3 through 4.5 each conclude by showing computed examples that illustrate the concepts. Finally, Section 4.6 summarizes the contributions of this work and presents some directions for future research.

4.2 Background and Motivation

In this section, we define a general multiple-robot motion planning problem in continuous time, under general, independent performance criteria. This can be considered as an extension to multiple decision makers of the optimal control encoding of the basic motion planning problem from Section 1.3. This continuous-time representation is used

to precisely characterize concepts of optimality in this general context. The methods presented in Sections 4.3, 4.4, and 4.5 can be considered as special versions of this general problem. Section 4.2.1 introduces the basic concepts and models that characterize our problem. Section 4.2.2 introduces our proposed optimality concept. Section 4.2.3 provides motivation for this concept by establishing relationships to solution concepts from multiobjective optimization and game theory literature.

4.2.1 Basic definitions

We use some of the configuration-space concepts that were described in Section 1.2.1. Let each robot, $\mathcal{A}^i$, be a rigid object, capable of moving in a workspace that is a bounded subset of $\Re^2$ or $\Re^3$. The position and orientation of the robot in the workspace are specified parametrically, by a point in an n-dimensional *configuration space*, $\mathcal{C}^i$. Static obstacles in the workspace (compact subsets of $\Re^2$ or $\Re^3$) prohibit certain configurations of the robot. The open subset of $\mathcal{C}^i$ that corresponds to configurations in which $\mathcal{A}^i$ does not intersect any obstacles is referred to as the *free configuration space*, and is denoted by $\mathcal{C}^i_{free}$. If the robot is permitted to "touch" obstacles when executing a motion plan, we alternatively use the *valid configuration space*, $\mathcal{C}^i_{valid}$, which is the closure of $\mathcal{C}^i_{free}$. We use $\mathcal{C}^i_{valid}$ in this work because optimality is more straightforward to consider. This distinction is primarily technical, because solutions that exist in $\mathcal{C}^i_{valid}$ can be considered as limit points for solutions in $\mathcal{C}^i_{free}$. We assume that each robot has complete knowledge of $\mathcal{C}^i_{valid}$, along with perfect configuration sensing and control.

A *state space*, X, is defined that simultaneously represents the configurations of all of the robots. Because collisions with obstacles are prohibited, a natural choice for the state space is

$$X = \mathcal{C}^1_{valid} \times \mathcal{C}^2_{valid} \times \cdots \times \mathcal{C}^N_{valid}, \tag{4.1}$$

in which $\times$ denotes the Cartesian product. In this chapter, we also consider two additional definitions of the state space that are more restrictive. Section 4.3 will consider motions of the robots that are restricted to fixed paths. The corresponding state space will be referred to as $X = \mathcal{S} = \mathcal{S}^1 \times \mathcal{S}^2 \times \cdots \times \mathcal{S}^N$. Section 4.4 will consider a more general case in which the robots are constrained to move along independent roadmaps. This yields a state space that we refer to as $X = \mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2 \times \cdots \times \mathcal{R}^N$. The state space used in this context differs from its typical use in robotics and control theory for modeling dynamics [183]; we do not incorporate dynamics because our primary concern is the geometric aspect of motion planning.

The concepts introduced in the remainder of this section apply to any of the above state space definitions. For this reason we generally refer to the state space as

$$X = X^1 \times X^2 \times \cdots \times X^N. \tag{4.2}$$

Each subspace, X^i, of the state space yields the configuration of $\mathcal{A}^i$. We use the notation $\mathcal{A}^i(x^i)$ to refer to the transformed robot, $\mathcal{A}^i$, at x^i.

In multiple robot motion planning problems, we are concerned not only about collision with obstacles, but also about collisions that occur between robots. Let $\mathcal{A}^{i\circ}$ denote the interior of $\mathcal{A}^i$ (i.e., the open set corresponding to the exclusion of the boundary of $\mathcal{A}^i$). We define (see Figure 4.3) as

$$X_{coll}^{ij} = \{x \in X \mid \mathcal{A}^{i\circ}(x^i) \cap \mathcal{A}^{j\circ}(x^j) \neq \emptyset\}, \tag{4.3}$$

which represents the set of states in which the two robots collide. The implication of using the interior of $\mathcal{A}^i$ is that we allow the robots to "touch." The *collision subset*, $X_{coll} \subset X$ is represented as the open set,

$$X_{coll} = \bigcup_{i \neq j} X_{coll}^{ij}. \tag{4.4}$$

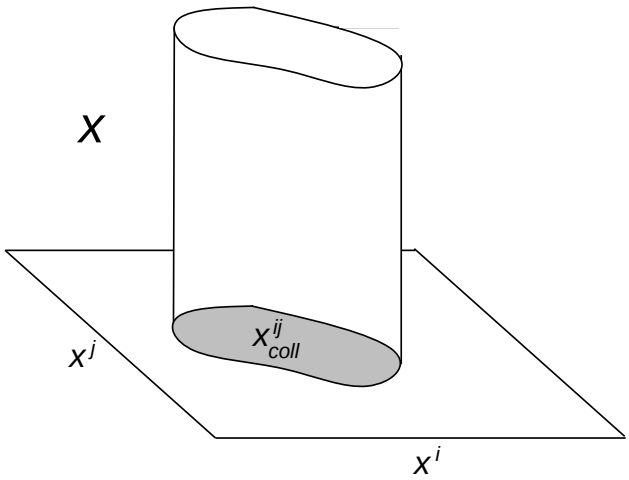


Figure 4.3 The set X^{ij}_{coll} and its cylindrical structure on X.

Hence, a state is in the collision subset if the interior of two or more robots intersect. We define X_{valid} as the closed set, $X - X_{coll}$.

Note the cylindrical structure of X_{coll} (depicted in Figure 4.3). Computationally, a representation of X^{ij}_{coll} can be constructed by considering only robot-pair collisions, and cylindrically lifting the result into all of X. The collision subset is then formed as the union of the X^{ij}_{coll}. This property is exploited by our algorithms when building a representation of the state space, allowing the number of collision detections to grow quadratically with N, as opposed to growing exponentially.

The basic task is to bring each robot from some initial state $x^i_{init} \in X^i$ to some goal state $x^i_{goal} \in X^i$. While achieving this task, each robot is not permitted to collide with obstacles or other robots (i.e., the state must remain within X_{valid}). In addition, explicit objectives must be taken into consideration when achieving this task.

We consider a *state trajectory* as a continuous mapping $x : [0, T] \rightarrow X$. A trajectory for an individual robot is represented as $x^i : [0, T] \rightarrow X^i$. An explicit choice for the final time, T, is usually not needed in practice. For some problems, a final time may naturally

exist, by which the robots must accomplish the basic task. Usually, however, we do not require a specific termination time, and can consider $T = \infty$.

The motion of an individual robot, $\mathcal{A}^i$, is specified through the *state transition equation*,

$$\dot{x}^i(t) = f^i(x^i(t), u^i(t)) \text{ for each } i \in \{1, \ldots, N\}, \tag{4.5}$$

in which $u^i(t)$ represents a *control function* for $\mathcal{A}^i$, which is chosen from a set of allowable controls.

Because we focus on the geometric aspect of a motion planning problem, we assume that a robot is capable of switching between a fixed, maximum speed, $\|v^i\|$, and remaining motionless (this represents a typical assumption in geometric motion planning [11], [60], [96], [137]). If, for example, a robot is allowed to translate and rotate, then finite bounds might be given that limit the translational and angular speeds. The particular form that (4.5) will take is made more explicit in Sections 4.3 through 4.5.

We next express the performance criteria for the robots. For each robot, $\mathcal{A}^i$, we define a *loss functional* of the form

$$L^i(x_{init}, x_{goal}, u^1, \ldots, u^N) = \int_0^T g^i(t, x^i(t), u^i(t))dt + \sum_{j \neq i} c^{ij}(x(\cdot)) + q^i(x^i(T)), \tag{4.6}$$

which maps to the extended reals, and

$$c^{ij}(x(\cdot)) = \begin{cases} 0 & \text{if } x(t) \in X_{valid} \text{ for all } t \\ \\ \infty & \text{otherwise} \end{cases} \tag{4.7}$$

and

$$q^i(x^i(T)) = \begin{cases} 0 & \text{if } x^i(T) = x^i_{goal} \\ \\ \infty & \text{otherwise} \end{cases} . \tag{4.8}$$

The function g^i represents a continuous cost function, which is a standard form that is used in optimal control theory. We additionally require, however, that

$$g^i(t, x^i(t), u^i(t)) = 0 \quad \text{if } x^i(t) = x^i_{goal}. \tag{4.9}$$

This implies that no additional cost is received while robot $\mathcal{A}^i$ "waits" at x^i_{goal} until time T.

The middle term in (4.6), $c^{ij}(x(\cdot))$ penalizes collisions between the robots. This has the effect of preventing any robots from considering strategies that lead to collision.

The function $q^i(x^i(T))$ in (4.6) represents the goal in terms of performance. If a robot, $\mathcal{A}^i$, fails to achieve its goal x^i_{goal}, then it receives infinite loss. We could alternatively associate a finite loss with failure to achieve a goal.

4.2.2 A proposed solution concept

Suppose that a coordination problem has been posed in which the state space, X, is defined, along with initial and goal states, x_{init} and x_{goal}. The goal of each robot is to choose some control function u^i that achieves the goal x^i_{goal} while considering the loss functional in Equation (4.6). We will use the notation γ^i to refer to a *robot strategy* for $\mathcal{A}^i$, which represents a possible choice of a control function that incorporates state feedback, represented as $u^i(t) = \gamma^i(x, t)$. In terms of control laws, this is equivalent to a closed-loop controller. In principle, extensions that incorporate incomplete or imperfect information feedback can be made (see, e.g., [6], [113], [117], [116]). The distinction between using γ^i and u^i will become more important in the coming sections. We refer to $\gamma = \{\gamma^1, \gamma^2, \ldots, \gamma^N\}$ as a *strategy*. Let Γ denote the set of all allowable strategies.

A *stationary strategy* is a special form of strategy that depends only on state, and not on a particular time. This concept has been used in both Chapters 2 and 3. For the motion planning problems that we consider in this chapter, the solutions are naturally

stationary. If f^i and g^i are time invariant, then the resulting solution strategies will be stationary. This is true because both the objectives expressed in Equation (4.6) and the effects of the control $u^i(t)$ on the system remain invariant through the passage of time. The algorithms that we present in Sections 4.3 through 4.5 can be extended to handle time-varying strategy solutions, but we present no examples of this type.

For a given x_{init} and strategy γ, the entire trajectory, $x(t)$, can be determined. If we assume that x_{init} and x_{goal} are given, then $L^i(\gamma)$ can be written, instead of using the form $L^i(x_{init}, x_{goal}, u^1, \ldots, u^N)$. Unless otherwise stated, we assume in the remainder of the chapter that $L^i(\gamma)$ refers to the loss associated with implementing γ, to bring the robot from some fixed x_{init} to x_{goal}. Hence, we can consider the loss functional as a function on Γ.

In general, there will be many strategies in Γ that produce equivalent losses. Therefore, we define an equivalence relation, $\sim_L$, on all pairs of strategies in Γ. We say that $\gamma \sim_L \gamma'$ if and only if $L^i(\gamma) = L^i(\gamma') \ \ \forall i$ (i.e., γ and γ' are equivalent). The equivalence relation, $\sim_L$, induces a partition of Γ into classes that represent equivalent losses. We denote the *quotient strategy space* by $\Gamma/\sim$, whose elements are the induced equivalence classes. An element of $\Gamma/\sim$ will be termed a *quotient strategy* and will be denoted as $[\gamma]_L$, indicating the equivalence class that contains γ.

For the special case in which there is only one robot, the goal would be to select a strategy that minimizes the loss in Equation (4.6). For multiple robots, however, a total ordering on the space of strategies does not, in general, exist. Consider a strategy, γ, which produces $L^1(\gamma) = 1$ and $L^2(\gamma) = 2$, and another strategy, γ', which produces $L^1(\gamma') = 2$ and $L^2(\gamma') = 1$. From a global perspective, it is not clear which strategy would be preferable. Robot $\mathcal{A}^1$ would prefer γ, while $\mathcal{A}^2$ would prefer γ'. Both robots would, however, prefer either strategy to a third alternative that produced $L^1(\gamma'') = 5$ and $L^2(\gamma'') = 5$. These comparisons suggest that there exists a natural partial ordering

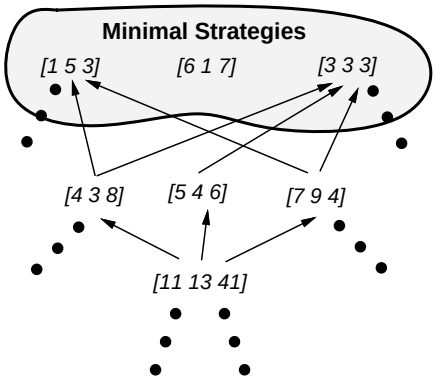


Figure 4.4 We are interested in obtaining strategies that are minimal with respect to the partial ordering that exists on $\Gamma/\sim$.

on the space of strategies (see Figure 4.4). Our interest is in finding the set of strategies that are minimal with respect to this partial ordering; these comprise all of the useful strategies, because any other strategies would not be preferred by any of the robots.

We define a partial ordering, $\preceq$, on the space $\Gamma/\sim$. The minimal elements with respect to $\Gamma/\sim$ will be considered as the solutions to our problem. For a pair of elements $[\gamma]_L, [\gamma']_L \in \Gamma/\sim$ we declare that $[\gamma]_L \preceq [\gamma']_L$ if $L^i(\gamma) \leq L^i(\gamma')$ for each i. If it further holds that $L^j(\gamma) < L^j(\gamma')$ for some j, we say that $[\gamma]_L$ is *better* than $[\gamma']_L$. Two quotient strategies, $[\gamma]_L$ and $[\gamma']_L$, are *incomparable* if there exists some i, j such that $L^i(\gamma) < L^i(\gamma')$ and $L^j(\gamma) > L^j(\gamma')$. Hence, we can consider $[\gamma]_L$ to be either better than, *worse* than, equivalent to, or incomparable to $[\gamma']_L$. We can also apply the terms *worse* and *better* to representative strategies of different quotient strategies; for example we could say that γ is better than γ' if $[\gamma]_L \preceq [\gamma']_L$. We say that $[\gamma^*]_L$ is a *minimal* strategy if, for all $[\gamma]_L \neq [\gamma^*]_L$ such that $[\gamma]_L$ and $[\gamma^*]_L$ are not incomparable, we have $[\gamma^*]_L \preceq [\gamma]_L$.

4.2.3 Relationships to established forms of optimality

In this section, we show how the minimal strategies relate to optimality concepts from multiobjective optimization and dynamic game theory. These concerns are relevant

because assumptions that form the basis of these decision-making contexts can also arise in a robotics context. Multiobjective optimization applies to a single decision maker which is confronted with a vector of objectives. In a motion planning context, this could correspond to providing a centralized controller for all of the robots. By applying a game-theoretic view to the multiple-robot planning problem, one could assume that the robots independently and autonomously implement motion plans. This situation can become more complex, because such issues as communication and cooperation become important to ensure that such problems such as deadlock do not arise. Such issues have been considered for multiple robots in [152], [179].

If the robots are working to accomplish a single task, then it seems natural to model cooperation. In fact, it may be possible to convert the vector of objectives into a single criterion; this form of combination leads to *team theory* [86], [99], which has been applied to multisensor planning in [53], [79]. In some situations, however, the robots could be working to achieve independent tasks. Suppose, for example, that several robots transport materials in a warehouse or factory floor, for independent processing. The formulation of a single criterion for this type of problem may be difficult, or have little significance. These issues could be further complicated if communication is limited or nonexistent.

In multiobjective decision making, the usual approach is to determine a small set of alternatives to present to the decision maker. A partial ordering is defined on the space of strategies that indicates pairwise preferences between strategies (if the ordering were linear, standard optimization could be applied). Using the partial ordering, any strategy that is *nondominated* [205] is considered as a solution. This usually results in the set of strategies that are minimal with respect to the partial ordering. Under the partial ordering in Section 4.2.2, our proposed solutions are the nondominated strategies in $\Gamma/\sim$.

One popular optimality concept that is considered in cooperative game theory is *Pareto optimality*. This concept assumes that a high degree of cooperation is available among the decision makers (other models that involve partial cooperation, or the forming and competition of coalitions, could be defined [141]). A robot strategy, γ^*, is considered Pareto optimal, if for any other strategy $\gamma \in \Gamma$, no robot can improve its loss without directly causing the loss to increase for another robot. As shown in [184], Pareto optimal solutions are equivalent to nondominated solutions in a multiobjective context. Therefore, our minimal solutions can also be considered as Pareto optimal.

In noncooperative game theory, however, it could be the case that for a given γ^*, robot $\mathcal{A}^i$ can further reduce its loss by selecting a different γ^i, given the γ^{j*} chosen by the other robots, even though $[\gamma^*]_L$ is minimal. In other words, robot $\mathcal{A}^i$ might not be satisfied with the outcome, given the actions taken by the other robots. This implies that if we treat the robots as individual agents with independent objectives, a minimal quotient strategy might not be a fair solution, or might not represent an appropriate concept of optimality. We will show, however, that the minimal quotient strategies share an additional property that represents a widely used optimality concept in noncooperative game theory [6]: the Nash equilibrium condition.

A given strategy $\gamma^* = \{\gamma^{1*} \dots \gamma^{N*}\}$, is termed a *Nash equilibrium* if the following holds for each i and each $\gamma^i \in \Gamma^i$:

$$L^i(\gamma^{1*}, \dots, \gamma^{i*}, \dots \gamma^{N*}) \leq L^i(\gamma^{1*}, \dots, \gamma^{i}, \dots \gamma^{N*}). \tag{4.10}$$

If in addition, there is no better strategy that is also a Nash equilibrium, then γ^* is an *admissible Nash equilibrium*.

The following proposition establishes that for our context, minimal quotient strategies are equivalent to admissible Nash equilibria.

Proposition 1 *A minimal quotient strategy, $[\gamma^*]_L$, is an admissible Nash equilibrium if and only if $[\gamma^*]_L$ is minimal in $\Gamma/\sim$.*

Proof: Suppose that $[\gamma^*]_L$ is a minimal strategy, but not a Nash equilibrium. To violate the Nash condition of Equation (4.10), for some i there must exist a strategy $\gamma \in \Gamma$, such that $\gamma = \{\gamma^{1*}, \ldots, \gamma^{i-1*}, \gamma^i, \gamma^{i+1*}, \ldots, \gamma^{N*}\}$ and $L^i(\gamma) < L^i(\gamma^*)$. If $[\gamma]_L \preceq [\gamma^*]_L$, then a contradiction would be reached. Since $L^i(\gamma) < L(\gamma^*)$, then we would have $[\gamma]_L \preceq [\gamma^*]_L$ if $L^j(\gamma) = L_j(\gamma^*)$ for all $j \neq i$. We will establish that this is indeed true by analyzing the loss functional definition in (4.6), (4.7), and (4.8). Consider any $j \neq i$. We argue that each of the three additive terms in (4.6) remains fixed when γ^* is replaced by γ. The function $g^j(t, x^j(t), u^j(t))$ depends only on the robot strategy γ^{j*}, and not on the other robot strategies. Since γ^{j*} remains the same in γ and γ^*, $g^j(t, x^j(t), u^j(t))$ remains constant. We must have $c^{ij}(x(\cdot)) = 0$ under the implementation of γ; otherwise, we would have $L^i(\gamma) = \infty$, which implies that $\mathcal{A}^i$ and $\mathcal{A}^j$ collide. The trajectories $x^j(\cdot)$ of the other robots do not change, which implies that $q^j(x^j(T))$ remains unchanged. Hence, we must have $L^j(\gamma) = L^j(\gamma^*)$ for all $j \neq i$ (i.e., (4.6) remains constant). This implies, however, that $[\gamma]_L \preceq [\gamma^*]_L$, which is a contradiction to the minimality assumption. Since $[\gamma^*]_L$ is both minimal and a Nash equilibrium, there does not exist another Nash equilibrium that is better; therefore. $[\gamma^*]_L$ is an admissible Nash equilibrium.

Suppose that $[\gamma^*]_L$ is an admissible Nash equilibrium, but not minimal in $\Gamma/\sim$. Then there exists a minimal quotient strategy $[\gamma]_L \in \Gamma/\sim$ such that $[\gamma]_L \preceq [\gamma^*]_L$, and $[\gamma]_L \neq [\gamma^*]_L$. Since $[\gamma]_L$ is minimal in $\Gamma/\sim$, it must be an admissible Nash equilibrium by the first part of this proof. This contradicts the assumption that $[\gamma^*]_L$ is an admissible Nash equilibrium. □

We can also consider the relationship between our minimal strategies and scalar optimization. Formal treatment of this relationship has been considered in multiobjective

optimization literature and is referred to as *scalarization* [164]. The goal of scalarization is to determine a mapping that projects the losses to a scalar value, while guaranteeing that optimizing the scalar loss produces a minimal strategy. This is advantageous because standard optimization techniques can then be applied to produce a minimal strategy. The tradeoff, however, is that the scalarizing function selects only a single minimal strategy. This makes the particular choice of a scalarizing function crucial, and information about the solution alternatives is lost.

We present a linear scalarizing function, and then show that optimizing the scalar objective yields a minimal strategy. This function is used in Section 4.5, in an algorithm that determines minimal solutions. Consider a set of positive, real-valued constants, $\beta = \{\beta_1, \beta_2, \ldots, \beta_N\}$, such that

$$\sum_{i=1}^{N} \beta_i = 1. \tag{4.11}$$

We can scalarize the objectives by performing a linear projection,

$$H(\gamma, \beta) = \sum_{i=1}^{N} \beta_i L^i(\gamma). \tag{4.12}$$

By taking a linear combination of the robot objectives, we can consider the scalarizing function as giving weighted importance to different robots. If we take $\beta_i = \frac{1}{N}$ for all $i \in \{1, \ldots, N\}$, then the scalarizing function produces the average loss among the robots. In principle, this scalarizing function could be considered as a flexible form of prioritization.

The scalarizing function in (4.12) produces a minimal strategy, which is stated in the following proposition:

Proposition 2 *For a fixed β, if γ^* is a strategy that minimizes $H(\gamma, \beta)$, then the quotient strategy $[\gamma^*]_L$ is minimal.*

Proof: Suppose to the contrary that $[\gamma^*]_L$ is not minimal. Then there exists some γ, such that $\gamma \preceq \gamma^*$. This implies that $L^i(\gamma) \leq L^i(\gamma^*)$ for each $i \in \{1, \ldots, N\}$, and

there exists some i for which this inequality is strict. By comparing the terms in (4.12), we determine that $H(\gamma, \beta) < H(\gamma^*, \beta)$. This contradicts the fact that the choice of γ^* minimizes H, which establishes the proposition. □

This implies that $H(\gamma, \beta)$ can be optimized to determine a minimal quotient strategy; however, in addition, we can apply H to the set of all minimal quotient strategies (which can be obtained by our algorithms) to select a single strategy. Once the minimal strategies have been obtained, different values of β can be used, which only requires a different selection from the small set of minimal quotient strategies, as opposed to re-exploring Γ. This would be useful if the robots were to repeatedly perform the same tasks, with preferences or priorities that change over time.

4.3 Motion Planning Along Fixed Paths

In this section, we consider the problem of coordinating the motions of multiple robots, when each robot is independently constrained to traverse a fixed path. As discussed in Section 4.1, many variations of this problem have previously been considered in motion planning research. The method developed in this section is perhaps most related to [137], because both approaches construct an approximate cellular representation of a coordination space.

This section describes some new contributions to the problem of coordinating multiple robots along fixed paths. First, we generalize the coordination space to more than two robots by exploiting the cylindrical structure of X_{coll}. The principle of optimality is then applied to the problem of determining all minimal quotient strategies. Using homotopy we first show that few minimal quotient strategies will typically exist. We then develop an algorithm that determines the minimal quotient strategies and that can be considered as a form of dynamic programming on a partially-ordered space of strategies. For a

scalarized objective, we describe how both dynamic programming and A^* search [198] can be applied to the coordination space representation to obtain single solutions in a straightforward manner. To conclude the section, computed minimal strategies are presented.

4.3.1 Concepts and definitions

We assume that each robot $\mathcal{A}^i$ is given a path, τ^i, which is a continuous mapping $[0,1] \rightarrow C^i_{valid}$. Without loss of generality [139], assume that the parameterization of τ^i is of constant speed. Let $\mathcal{S}^i = [0,1]$ denote the set of parameter values that place the robot along the path τ^i. We define a *path coordination space* as $\mathcal{S} = \mathcal{S}^1 \times \mathcal{S}^2 \times \cdots \times \mathcal{S}^N$.

A strategy, $\gamma \in \Gamma$, must be provided in which $s_{init} = (0, 0, \ldots, 0)$ and $s_{goal} = (1, 1, \ldots, 1)$, and the robots do not collide. This corresponds to moving each robot from $\tau^i(0)$ to $\tau^i(1)$. We assume that a robot, $\mathcal{A}^i$, monotonically moves toward $\tau^i(1)$; waiting at a particular $\tau^i(s)$ for some $s^i \in [0,1]$ is also allowed. It is assumed that the robots do not collide with static obstacles, implying that each given path τ^i is a solution to the basic motion planning problem for $\mathcal{A}^i$ (with the other robots removed).

We perform a discrete-time analysis of this problem. The development of continuous-time, analytical solutions would require detailed analysis for specific models and geometric representations; however, with discrete time, we can readily compute approximate solutions to a variety of motion planning problems. This choice is often made in motion planning research (e.g., [11], [137]). We assume that we can send an action (or motion command) to each robot every Δt seconds. The selection Δt could be governed in practice by one of two constraints: (1) a limiting sampling rate for a robotic system; or (2) the computational resources needed to determine the minimal strategies.

With the discretization of time, we partition $[0, T]$ into *stages*, denoted by $k \in \{1, \ldots, K\}$. Stage k refers to time $(k-1)\Delta t$. Discretized time allows $\mathcal{S}$ to be represented by a discrete set of locations, which corresponds to the set of possible positions along the paths at time $k\Delta t$ for some k. For each robot, say $\mathcal{A}^1$, we partition the interval $\mathcal{S}^1 = [0, 1]$ into values that are indexed by $i^1 \in \{0, 1, \ldots, i^1_{max}\}$, in which i^1_{max} is given by $\lfloor length(\tau^1)/\|v^1\|\Delta t \rfloor$. Each indexed value yields $\tau^1(i^1\|v^1\|\Delta t/length(\tau^1))$. We denote the discrete-time version of the path coordination space as $\tilde{\mathcal{S}} = \tilde{\mathcal{S}}^1 \times \tilde{\mathcal{S}}^2 \times \cdots \times \tilde{\mathcal{S}}^N$. Each $\tilde{\mathcal{S}}^i$ represents a finite set of points in $[0, 1]$. This yields a restricted space of strategies $\tilde{\Gamma} \subseteq \Gamma$. In Section 4.4, the notation $\tilde{\Gamma}$ will also be used to reprepresent a restricted strategy space that is due to discretization on $\mathcal{R}$.

The points in $\tilde{\mathcal{S}}$ can be used to delineate rectangular cells in $\mathcal{S}$ (Figure 4.5 shows an example that depicts this relationship). We can thus define $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}_{valid}$ as continuous subsets of $\mathcal{S}$ that represent cellular approximations of $\mathcal{S}_{coll}$ and $\mathcal{S}_{valid}$. We assume that a point is in $\tilde{\mathcal{S}}_{coll}$ if any point within its corresponding continuous cell is in S_{coll}, which corresponds to a conservative approximation. The set $\tilde{\mathcal{S}}_{valid}$ is then defined as the complement of $\tilde{\mathcal{S}}_{coll}$.

During the time interval $[(k-1)\Delta t, k\Delta t]$, each robot can decide to either remain motionless, or to move a distance $\|v^i\|\Delta t$ along the path. The choice taken by a robot, $\mathcal{A}^i$, is referred to as an *action*, which is denoted at stage k as u^i_k. The set of actions for the robots at a given stage is denoted by $u_k = \{u^1_k, \ldots, u^N_k\}$. The choices for u^i_k can be represented as 0 for no motion, and 1 to move forward. The next state from $\tau^i(s^i_k)$, with action $u^i_k = 1$, is given by $\tau^i(s^i_k + \|v^i\|\Delta t/length(\tau^i))$.

We can approximate (4.6), in discrete time as

$$L^i(\gamma) = \sum_{k=1}^{K} \left\{ l^i_k(x^i_k, u^i_k) + \sum_{j \neq i} c^{ij}_k(x(\cdot)) \right\} + q^i(x^i_{K+1}), \tag{4.13}$$

in which

$$l_k^i(x_k^i, u_k^i) = \int_{(k-1)\Delta t}^{k\Delta t} g^i(t, x^i(t), u^i(t))dt \tag{4.14}$$

and

$$c_k^{ij}(x(\cdot)) = \begin{cases} 0 & \text{if } x(t) \notin S_{coll}^{ij} \quad \forall t \in [(k-1)\Delta t, k\Delta t] \\ \infty & \text{otherwise} \end{cases} . \tag{4.15}$$

The l_k^i and q^i terms of (4.13) comprise the standard terms that appear in a discrete-time dynamic optimization context [6]. The middle term, c_k^{ij} represents the interaction between the robots, by penalizing collision. As will be seen shortly, l_k^i will typically be considered as a constant, which measures time, for example.

Before discussing the algorithm in Section 4.3.2, we will provide a proposition that characterizes the quantity of minimal quotient strategies that can exist in $\tilde{\Gamma}/\sim$ for the fixed-path coordination problem. Numerous minimal quotient strategies might seem to exist, even for only two robots. Suppose there were strategies that produced losses $L^1 = i$ and $L^2 = 10000 - i$ for each $i \in \{1, \ldots, 10000\}$. No pair of these strategies is comparable and, hence, all strategies could be minimal. In multiobjective optimization, the existence or numerous, or even an infinite number of solutions, often causes difficulty [205]. We show that at least for the case in which time-optimality is of interest (i.e., $l_k^i(x_k^i, u_k^i) = \Delta t$ for all i, k), there are very few minimal quotient strategies because each must be obtained from a distinct path class in $\tilde{\mathcal{S}}_{valid}$.

We formally consider path classes in $\tilde{\mathcal{S}}_{valid}$. A given strategy, $\gamma \in \tilde{\Gamma}$, yields a trajectory $\alpha_\gamma : [0, T] \rightarrow \tilde{\mathcal{S}}_{valid}$ through the coordination space. A different strategy, $\gamma' \in \tilde{\Gamma}$, yields a trajectory $\alpha_{\gamma'}$. The two paths α_γ and α'_γ are *homotopic* in $\tilde{\mathcal{S}}_{valid}$ (with endpoints fixed) if there exists a continuous map $h : [0, T] \times [0, 1] \rightarrow \tilde{\mathcal{S}}_{valid}$ with $h(t, 0) = \alpha_\gamma(t)$ and $h(t, 1) = \alpha_{\gamma'}(t)$ for all $t \in [0, T]$, and $h(0, s) = h(0, 0)$ and $h(1, s) = h(1, 0)$ for all

$s \in [0,1]$. This homotopy determines an equivalence relation, $\sim_h$, on the trajectories and, hence, on the space of strategies, $\tilde{\Gamma}$. We will use $[\alpha_\gamma]_h$ to denote the equivalence class that contains α_γ. Because α_γ is monotone, the path classes defined here do *not* represent the fundamental group from homotopy theory [87]; there are far fewer path classes in this context. Note that $\sim_h$ is distinct from $\sim_L$, which induces equivalence classes by losses.

Using these path classes we have the following proposition:

Proposition 3 *If $l_k^i(x_k^i, u_k^i) = \Delta t$ for all $i \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, K\}$, then there exists at most one minimal quotient strategy per path class in $\tilde{\mathcal{S}}_{valid}$.*

The proof of this proposition relies on the algorithm in Figure 4.6, and is given in Appendix A. One implication of this proposition is that the number of solutions is quite restricted (in practice there typically are only a few minimal quotient strategies).

4.3.2 Algorithm presentation

In this section, we present an algorithm that determines all of the minimal quotient strategies in $\tilde{\Gamma}/\sim$. We apply the principle of optimality, but in this context there is only a partial ordering on the space of strategies; therefore, the principle of optimality is replaced by a principle of minimality with respect to the partial ordering. In other words, any portion of a minimal strategy is itself minimal. Because multiple minimal strategies will be maintained, the computational issues will become more complex than for the standard dynamic programming that results from the principle of optimality.

We represent both $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$ using N-dimensional arrays. A strategy $\gamma \in \tilde{\Gamma}$ must ensure that the robots do not collide during the transitions from x_k to x_{k+1} (i.e., $x(t)$ does not produce a collision $\forall t \in [(k-1)\Delta t, k\Delta t]$). In practice, this computation depends on the type of curve τ^i, the geometry of $\mathcal{A}^i$, and the type of transformation that is performed to obtain $\mathcal{A}^i(x^i)$. In our current implementation, we have detected collisions

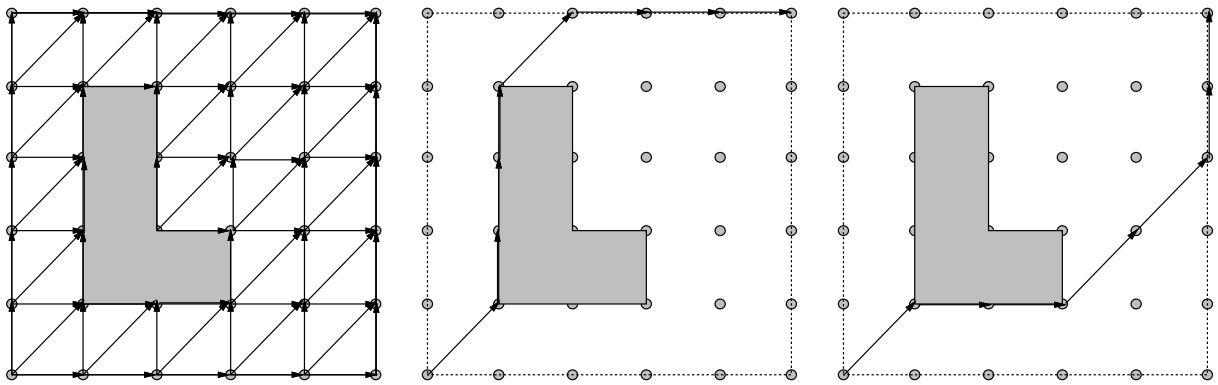


Figure 4.5 Part (a) shows an illustrative example of the coordination space representation. Parts (b) and (c) depict two minimal strategies.

only at the endpoints of the cells. The structure of this representation for $N = 2$ is shown in Figure 4.5(a). Each shaded square represents a rectangular subset of $\mathcal{S}$ in which a collision might occur (i.e., a subset of $\tilde{\mathcal{S}}_{coll}$). The points in Figure 4.5(a) indicate the values of $\tilde{\mathcal{S}}$, and the arrows indicate the possible values for u_k. There are at most three possibilities for the $N = 2$ case: (1) $u_k^1 = 1$ and $u_k^2 = 1$; (2) $u_k^1 = 1$ and $u_k^2 = 0$; or (3) $u_k^1 = 0$ and $u_k^2 = 1$. In general, there are $2^N - 1$ possibilities (assuming that we do not allow situations in which all of the robots remain motionless). For this simple example, two distinct minimal strategies are shown in Figures 4.5(b) and 4.5(c), for minimum-time objectives.

We construct a data structure that maintains the complete set of minimal quotient strategies from each discretized value $\tilde{s} \in \tilde{\mathcal{S}}$. Each position $\tilde{s} = (s^1, s^2, \ldots, s^N)$ in the coordination space $\tilde{\mathcal{S}}$ will contain a list of minimal strategies $M(\tilde{s})$, which reach $(1, 1, \ldots, 1)$ from $\tilde{s}$. In $M(\tilde{s})$, we have only one representative strategy for each class in $\tilde{\Gamma}/\sim$. Equivalent strategies could in theory be maintained, however, the storage required would be computationally infeasible. Each element $m \in M(\tilde{s})$ is of the form:

$$m = \langle\, u_k\,,\, [L^{1*}\;\; L^{2*}\;\; \cdots\;\; L^{N*}]\,,\, j\,\rangle. \tag{4.16}$$

Above, u_k denotes the vector of actions that are to be taken by the robots, in the first step of the strategy represented by m. Each L^{i*} represents the loss that the robot $\mathcal{A}^i$ receives under the implementation of the minimal strategy that m represents. The actions u_k will bring the system to some $\tilde{s}'$. At this state location, a set of strategies will be represented, $M(\tilde{s}')$, and j in (4.16) indicates which element in $M(\tilde{s}')$ will continue the strategy.

For a given state, $\tilde{s}$, it will be useful to represent the set of all states that can be reached by trying the various combinations of robot actions that do not yield a collision (one can easily check the array representation of $\tilde{\mathcal{S}}$). Define $\mathcal{N}(\tilde{s}) \subset \tilde{\mathcal{S}}$ as the *neighborhood* of the state $\tilde{s}$, which corresponds to these immediately reachable states. Formally, we have

$$\mathcal{N}(\tilde{s}_k) = \{\tilde{s}' = w(\tilde{s}, u_k) \mid u_k \in U \text{ and } w(\tilde{s}, u_k) \in \tilde{\mathcal{S}}_{valid}\}, \tag{4.17}$$

in which w represents the next state that is obtained for the vector of robot actions u_k, and U denotes the space of possible action vectors.

Consider the algorithm in Figure 4.6. Only a single iteration is required over the coordination space. This algorithm does not require a specification of K, and it can be effectively assumed that $K = \infty$. In Step 1, all states are initially empty, expect for the goal state. Lines 5 through 8 are iterated over the entire coordination space, starting at the goal state, and terminating at the initial state. At each element, $\tilde{s}$, the minimal strategies are determined by extending the minimal strategies at each neighborhood element.

Consider the extension of some $m \in M(\tilde{s}')$ in which $\tilde{s}' \in \mathcal{N}(\tilde{s})$. Let u_k be the action such that $\tilde{s}' = f(\tilde{s}, u_k)$. Suppose that m is the i^{th} element in $M(\tilde{s}')$. The loss for the

```
1  Let M(s̃_goal) = {⟨∅, [0, 0, ..., 0], ∅⟩}, and all other M(s̃) be ∅
2  For each i^1 from i^1_max down to 0 do
3      For each i^2 from i^2_max down to 0 do
           ...
4              For each i^N from i^N_max down to 0 do
5                  Let s̃ = (i^1, i^2, ..., i^N)
6                  Let M_u be a set of strategies that is the union of M(s̃')
                       for each s̃' ∈ N(s̃)
7                  Construct a set M'_u by extending the strategies in M_u
8                  Let M(s̃) consist of all unique-loss minimal elements of M'_u
9  Return M(s̃_init)
```

Figure 4.6 For a stationary problem, this algorithm finds all of the minimal quotient strategies in $\tilde{\mathcal{S}}^1 \times \tilde{\mathcal{S}}^2 \times \cdots \times \tilde{\mathcal{S}}^N$.

extended strategy is given by

$$L_k^i = \begin{cases} 0 & \text{if } \tilde{s}^{i\prime} = 1 \\ & \\ L^{i\prime} + l_k^i(\tilde{s}^i, u_k^i) & \text{otherwise} \end{cases}, \tag{4.18}$$

for each $i \in \{1, \ldots, N\}$. Suppose that m is the j^{th} element in $M(\tilde{s}')$. The third element of m (recall (4.16)) represents an index, j, which selects a strategy in $M(\tilde{s}')$.

We now discuss how to execute a strategy represented as $m \in M(\tilde{s})$. If the action u_k is implemented, then a new state $\tilde{s}'$ will be obtained. The index parameter j is used to select the j^{th} element of $M(\tilde{s}')$, which represents the continuation of the minimal strategy. From the j^{th} element of $M(\tilde{s}')$, another action is executed, and a coordination state $\tilde{s}''$ is obtained. This iteration continues until the goal state $(1, 1, \ldots, 1)$ is reached.

The following proposition establishes the correctness of the algorithm.

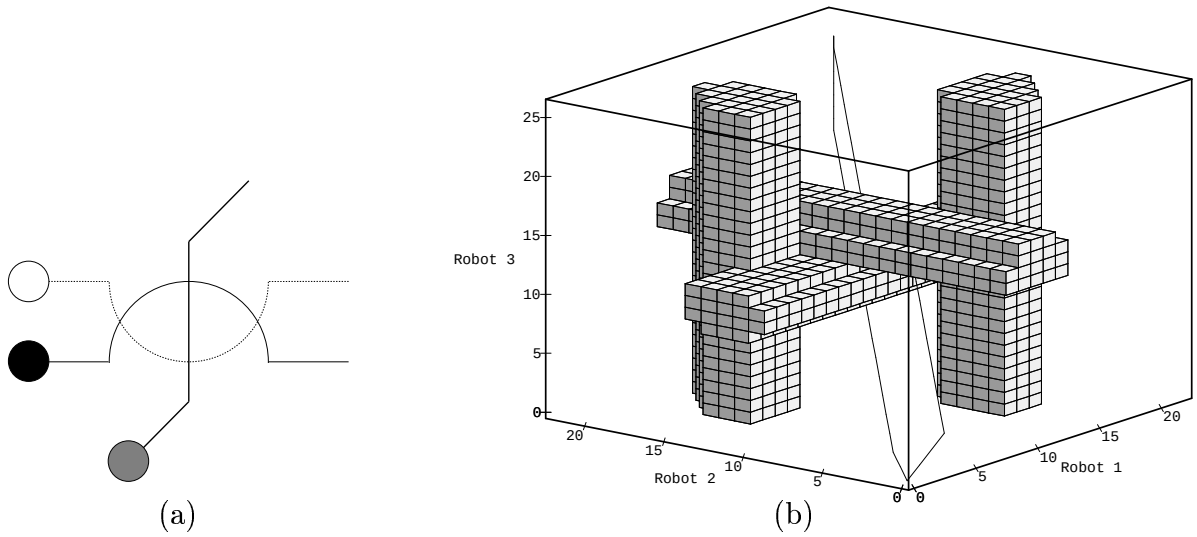
Proposition 4 *For a stationary problem, the algorithm presented in Figure 4.6 determines the complete set of minimal quotient strategies in* $\tilde{\Gamma}/\sim$ *for* $X = \tilde{\mathcal{S}} = \tilde{\mathcal{S}}^1 \times \tilde{\mathcal{S}}^2 \times \cdots \times \tilde{\mathcal{S}}^N$.

Proof: For any strategy that begins in a state $(s^1, s^2, \ldots, s^N)$, the trajectory in $\tilde{\mathcal{S}}$ will lie in the region bounded by $s^{i\prime} \geq s^i$ since the robots can only move forward along the path. Since the strategies depend only on state, it is argued inductively that the minimal strategies are maintained. At each inductive step, the extended strategies are functions of states for which minimal strategies have already been obtained (i.e., in the upper right portion of the coordination space). This type of induction forms the basis of Dijkstra's algorithm, for example, for single-source shortest paths [40]. □

The algorithm in Figure 4.6 can be specialized in a straightforward manner to provide a single strategy by minimizing the scalarized loss, which is shown in Equation (4.12). In this case, the problem has been reduced to the application of standard dynamic programming on a finite graph (the vertices are the discrete states, and the edges are the action combinations u_k), which is equivalent to Dijkstra's algorithm [40]. It is well known that A^* search can be performed on a problem of this type to potentially reduce by a substantial amount the number of vertices that are considered [149]. The only requirement is that an admissible heuristic is used, that is, it provides an underestimate of the remaining loss required to reach the initial vertex from a certain vertex. One reasonable choice for a heuristic would be to determine the remaining loss for each robot by neglecting the presence of the other robots (this is guaranteed to be an underestimate). If time optimality is considered, then the remaining loss is simply proportional to the remaining path length.

4.3.3 Computed examples

In this section, we present examples that illustrate the fixed-path coordination concepts and the algorithm from Figure 4.6. In our simulation experiments, we have considered problems in which there are from two to four robots. Many two-robot problems



(a) (b)

Figure 4.7 Part (a) shows a three-robot fixed-path problem, and part (b) shows the corresponding coordination space.

can be solved in a few seconds on a SPARC 10 workstation. Three-robot problems can take minutes or hours of computation, depending on both the resolution chosen to represent the coordination space and the number of minimal quotient strategies. We typically divide each axis of the coordination space into 25 to 100 cells.

Figure 4.7(a) shows an example that has three robots. The initial positions are indicated in Figure 4.7(a): $\mathcal{A}^1$ is black, $\mathcal{A}^2$ is white, and $\mathcal{A}^3$ is gray. Figure 4.7(b) shows the computed representation of $\tilde{\mathcal{S}}$. The axes show distances along the paths. The cylindrical structure in $\tilde{\mathcal{S}}_{coll}$ can be clearly observed in this example. The two vertical columns correspond to the two collisions that can occur between $\mathcal{A}^1$ and $\mathcal{A}^2$. Each of the two horizontal columns represents collisions of $\mathcal{A}^3$ with $\mathcal{A}^1$ or $\mathcal{A}^2$. There are two minimal quotient strategies for this problem, for which representative strategies are depicted as paths in the coordination space.

Figure 4.8 shows a three-robot example in which two robots move along "S"-curves, and the third robot moves horizontally. There are four minimal quotient strategies for this problem:

Strategy	Loss 1	Loss 2	Loss 3
γ_1^*	81	75	30
γ_2^*	79	73	82
γ_3^*	83	73	41
γ_4^*	73	80	30

.

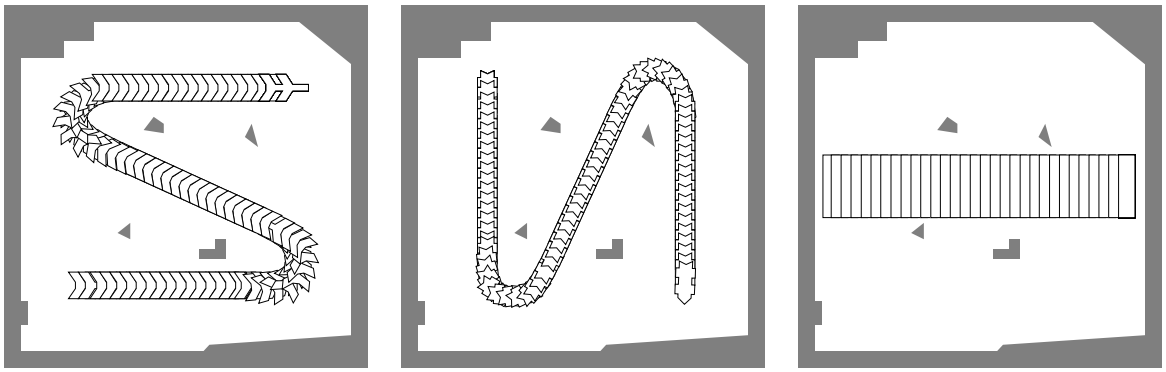
Each integer represents the number of stages required to reach x^i_{goal}. In the lower portion of Figure 4.6, we show four sets of timing diagrams, each of which corresponds to a representative minimal quotient strategy that was computed. Each graph indicates whether a robot is moving or waiting, as a function of time.

4.4 Motion Planning Along Independent Roadmaps

In this section we present a method that determines minimal strategies for the case in which the robots are restricted to independent roadmaps. This allows many more strategies to be considered than for fixed path coordination, while causing only a modest increase in computation. Many of the general concepts are similar to those from the last section; however, the complicated topological structure of a Cartesian product of roadmaps makes this problem more complex. At the end of the section, several computed minimal strategies are shown.

4.4.1 Concepts and definitions

We consider a *roadmap* for $\mathcal{A}^i$ to be a collection of curves, $\mathcal{T}^i$, such that for each $\tau^i_j \in \mathcal{T}^i$, $\tau^i_j : [0,1] \rightarrow \mathcal{C}^i_{valid}$. We assume (without loss of generality [139]) that each parametrization is of constant speed. The endpoints of some paths coincide in $\mathcal{C}^i_{valid} = X^i$ to form a network.



Robot 1 Robot 2 Robot 3

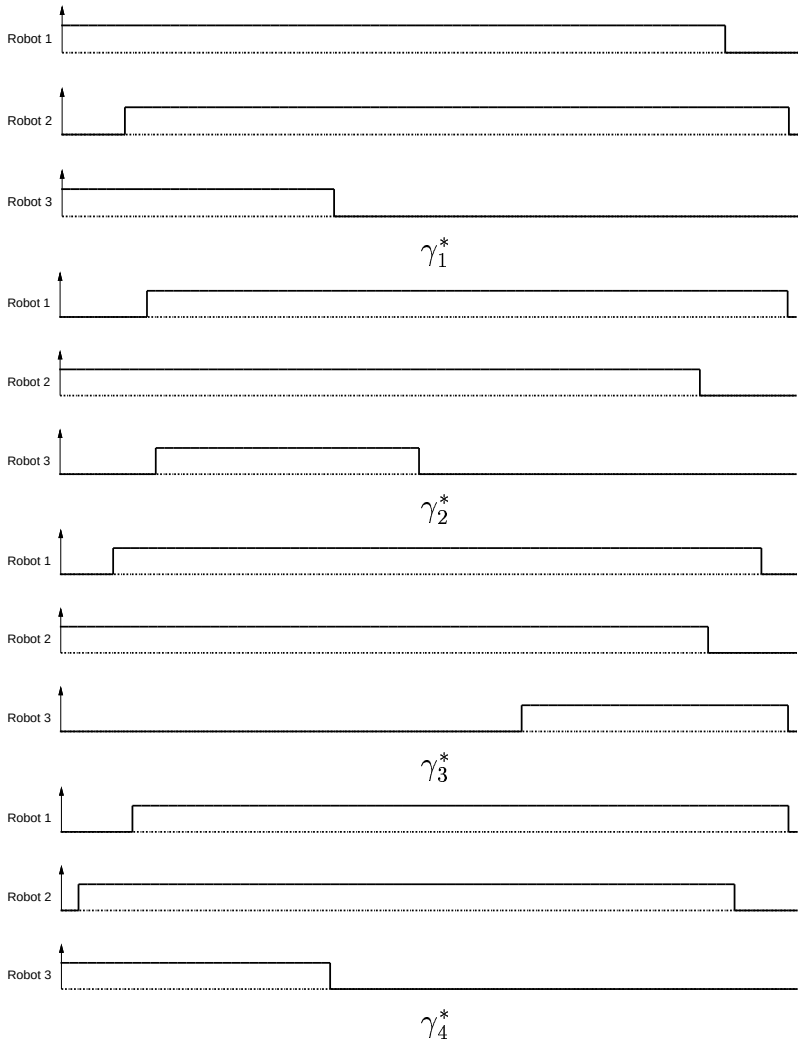


Figure 4.8 An example that yields four minimal quotient strategies.

In the Section 4.3 we considered robot coordination on the Cartesian product of unit intervals, which represented the domains of the paths. For the roadmap coordination problem, we will coordinate the robots on the domain of the functions in $\mathcal{T}^i$. We let $\mathcal{R}^i$ denote a set that represents the union of transformed domains of the paths in $\mathcal{T}^i$. Using the $\mathcal{R}^i$'s, we can describe a *roadmap coordination space*, $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2 \times \cdots \times \mathcal{R}^N$. We can specify a position $r \in \mathcal{R}$ by $r = (\pi^1, \pi^2, \ldots, \pi^N; s^1, s^2, \ldots, s^N)$. Each π^i specifies the path in $\mathcal{R}^i$ that $\mathcal{A}^i$ has chosen, while each s^i specifies the position of the robot along that path.

A problem is specified by providing an initial configuration, $r^i_{init} \in \mathcal{R}^i$, and a goal configuration, $r^i_{goal} \in \mathcal{R}^i$ for each robot $\mathcal{A}^i$. As a minor extension, we could also consider initial and goal configurations in $\mathcal{C}^i_{free}$; however, a roadmap can usually be extended in a straightforward manner to include the particular initial and goal states of a given problem [109].

During the time interval $[(k-1)\Delta t, k\Delta t]$ each robot can decide to either remain motionless or to move a distance $\|v^i\|\Delta t$ in either direction along a path. If the robot moves into a roadmap junction, then a new path must be chosen.

4.4.2 Algorithm presentation

In this section we present an algorithm that determines all of the minimal quotient strategies that arise from a Cartesian product of discretized roadmaps, $\tilde{\mathcal{R}}$. We consider the case in which $l^i_k(x^i_k, u^i_k) = \Delta t$ for all i, k. This means that the only performance concern is the time that the robots take to complete their tasks. Variations and extensions of this algorithm are mentioned.

We construct the discrete representations $\tilde{\mathcal{R}}_{coll}$ and $\tilde{\mathcal{R}}$, which are similar to $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$. We build one array for each combination of path choices for the robots, each of which

can be constructed in the same manner as for $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$. This representation can be intuitively be thought of as a network of coordination spaces that exists in $\Re^N$. Let $\tilde{\Gamma}/\sim$ denote the quotient strategy space that results from the discretization of $\tilde{\mathcal{R}}$, in that same manner in which $\tilde{\Gamma}/\sim$ was defined for $\tilde{\mathcal{S}}$.

There are two primary differences between the roadmap coordination problem and the fixed path coordination problem in terms of the algorithm development. The first difference is that robots on $\mathcal{R}$ are allowed to move in either direction. For fixed paths, we assumed that the robots could only move forward along a path. Allowing the robots to move in either direction, provides $3^N - 1$ choices for u_k, as opposed to $2^N - 1$ (there are additional choices when one or more robots moves into a junction, because a new path must be selected). This leads to more complicated strategies; for example, if $l_k^i(x_k^i, u_k^i) = \Delta t$, then, in general, minimal strategies will exist in Γ that cause one or more robots to oscillate on a path. The second major difference is the complicated topology of $\mathcal{R}$, as opposed to $\mathcal{S}$, which is a unit cube.

Both of these differences increase the difficulty of defining the neighborhood of a state. For an example of a neighborhood in the roadmap coordination problem in which $N = 2$, consider Figures 4.9 and 4.10. For this example, the second robot is approaching a junction, while the first robot is in the middle of a path. The white circles in Figure 4.9 indicate the positions of the robots at state $\tilde{r}$, and the black circles indicate possible locations of the robots at the next state, $\tilde{r}'$. One representation of this situation in $\tilde{\mathcal{R}}$ is illustrated in Figure 4.10. For this problem, there are 11 possible choices for $\tilde{r}'$. For each representation of some $m \in M(\tilde{r})$ (here M represents minimal strategies on the roadmap coordination space), in addition to the components in Equation (4.16), we store an index when necessary that indicates which new paths are chosen by the robots.

We now describe the algorithm in Figure 4.11. A set of roadmap coordination states, termed a *wavefront* $\mathcal{W}_i$ is maintained in each iteration. During an iteration, the complete

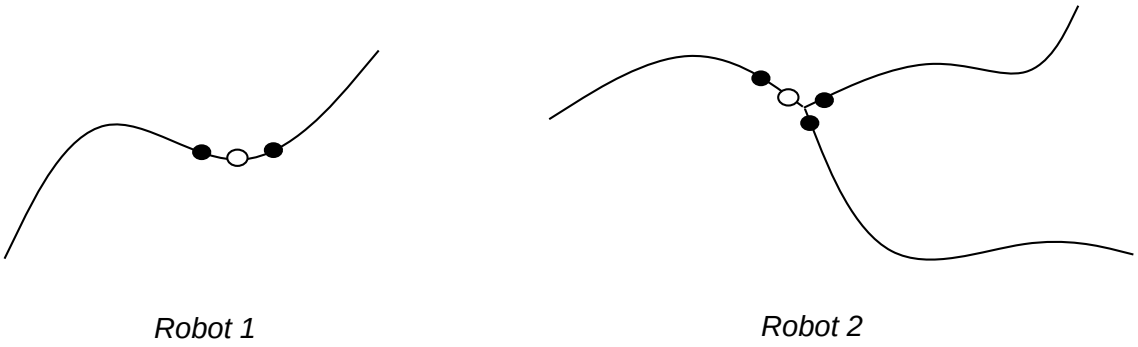


Figure 4.9 A two-robot example in which one of the robots can make a decision about which path to continue along. The white circles indicate current locations, and the black circles indicate potential next locations.

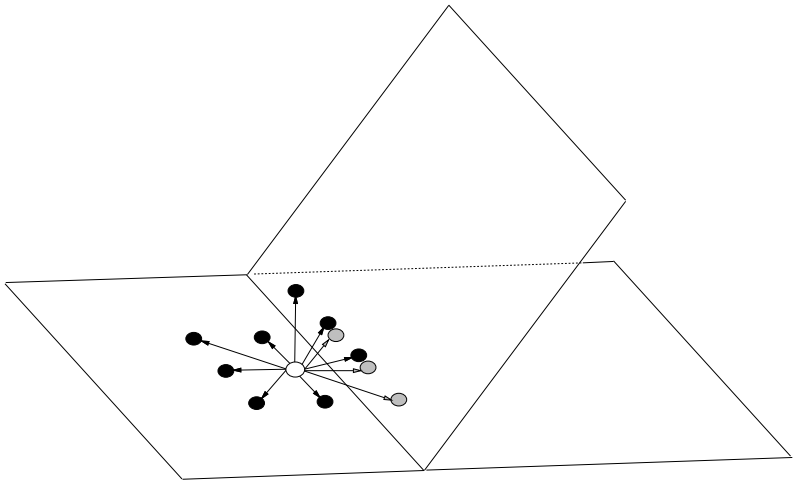


Figure 4.10 The corresponding path branch in the representation of $\tilde{\mathcal{R}}$.

set of minimal strategies is determined for each element of $\mathcal{W}_i$. The initial wavefront, $\mathcal{W}_0$, contains only the goal state. Each new wavefront $\mathcal{W}_i$ is defined as the set of all states that: (1) can be reached in one stage from an element in $\mathcal{W}_i$; and (2) are not included in any of $\mathcal{W}_{i-1}, \ldots, \mathcal{W}_0$. The algorithm terminates when all states have been considered. This algorithm could be viewed as a multiple-objective extension of the wavefront algorithm that is used in [14].

The following proposition establishes the correctness of the algorithm:

```
1   Initialize R̃
2   Let W_0 = {r̃_init}
3   i = 0
4   Until W_i = ∅ do
5       For each r̃ ∈ W_i do
6           Let M_u be a set of strategies that is the union of M(r̃')
                for each r̃' ∈ N(r̃)
7           Construct a set M'_u by extending the strategies in M_u
8           Let M(r̃) consist of all unique-loss minimal elements of M'_u
9           Let i = i + 1
10      Let W_i be set of all neighbors of W_{i-1} that have not yet been processed
11  Return M(r̃_init)
```

Figure 4.11 Suppose that $l_k^i(x_k^i, u_k^i) = \Delta t$ for all $k \in \{1, \ldots, K\}$ and $i \in \{1, \ldots, N\}$. This algorithm finds all of the minimal quotient strategies in $\tilde{\mathcal{R}}^1 \times \tilde{\mathcal{R}}^2 \times \cdots \times \tilde{\mathcal{R}}^N$.

Proposition 5 *The algorithm presented in Figure 4.11 determines the complete set of minimal quotient strategies in* $\tilde{\Gamma}/\sim$*, when* $X = \tilde{\mathcal{R}} = \tilde{\mathcal{R}}^1 \times \tilde{\mathcal{R}}^2 \times \cdots \times \tilde{\mathcal{R}}^N$.

Proof: We use an inductive argument that is based on the principle of minimality. After the i^{th} iteration of the algorithm, all minimal strategies that complete in time less than $i\Delta t$ are represented. After the iteration for $\mathcal{W}_1$, all of the single-stage minimal strategies are determined (corresponding to all of the elements of $\mathcal{W}_1$), forming the basis of the induction. Consider the wavefront $\mathcal{W}_i$ under the assumption that minimal strategies have been determined for all elements in the wavefronts $\mathcal{W}_{i-1}, \ldots, \mathcal{W}_0$. Any minimal strategy for a state $\tilde{r} \in \mathcal{W}_i$ must require exactly i stages to reach the goal. If it were possible to achieve the goal in fewer stages, then $\tilde{r}$ would have appeared in an earlier wavefront. By the principle of minimality over time, any minimal strategy that requires i stages must be an extension of some substrategy that required $i - 1$ stages, which has already been considered in a previous wavefront. Hence, the extension constructs the minimal strategies in $\mathcal{W}_i$, which completes the inductive step. □

The algorithm in Figure 4.11 can be scalarized in the same manner as discussed in Section 4.3.2. In addition, A^* search can be performed to obtain a single minimal

solution. We have successfully implemented an algorithm that performs A^* search on the roadmap coordination space. An extension can also be considered to include general loss functions and nonstationary strategies.

4.4.3 Computed examples

In this section, we present some computed examples that were obtained with the algorithm in Figure 4.11. In our simulation experiments for roadmap coordination, we have considered problems in which there are either two or three robots. Computation times range from a few minutes to several hours on a SPARC 10 workstation. The amount of computation time required depends on the resolution of the representation and the number of minimal strategies.

Figure 4.12 shows the two unique-loss minimal strategies side by side, for an "H"-shaped roadmap coordination problem in which two robots attempt to reach opposite corners. Intuitively, for this problem, we would expect two symmetric possibilities to exist: either $\mathcal{A}^1$ must wait, or $\mathcal{A}^2$ must wait. These two situations are precisely what are obtained in the two minimal quotient strategies.

Figures 4.13 and 4.14 present one minimal strategy in a roadmap coordination problem that involves three robots in $\Re^3$, with different roadmaps for each robot.

Figure 4.15 presents an example in which there are two robots in the plane that move along independent roadmaps. The configuration spaces of the individual robots is three dimensional in this case because the robots can rotate while moving along the roadmap. There are five minimal quotient strategies for this problem, and the two that are shown do not require either robot to wait. Quite distinct routes, however, are taken by the robots in the different strategies.

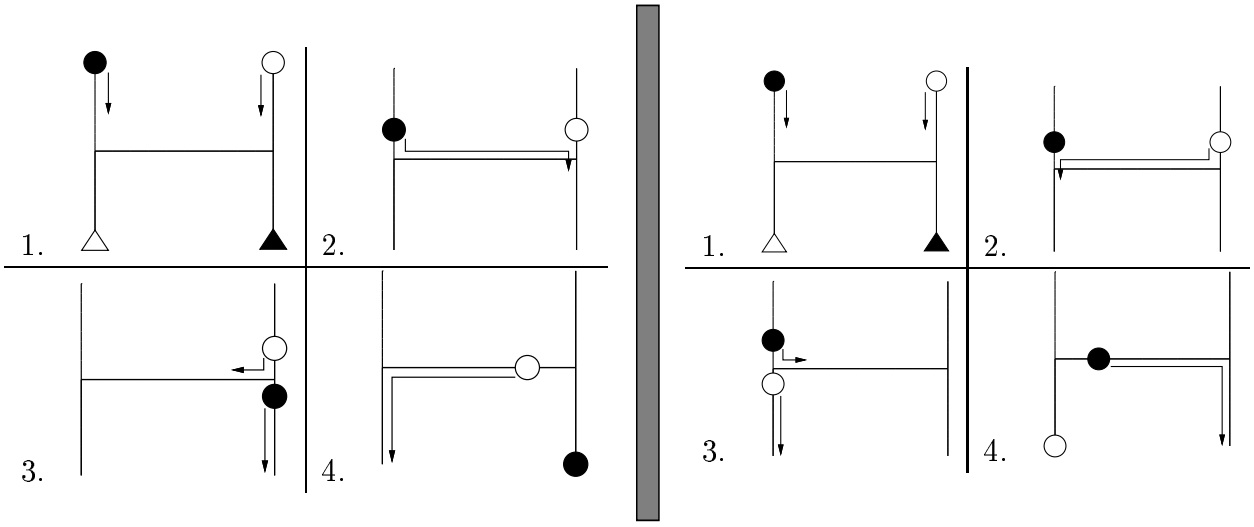


Figure 4.12 There are two symmetric minimal quotient strategies. The black and white discs represent $\mathcal{A}^1$ and $\mathcal{A}^2$, respectively. The black and white triangles indicate the goal configurations.

We briefly provide some statistics for the entire representation of $\tilde{\mathcal{R}}$ for this problem. This provides an indication of how few minimal quotient strategies exist in the state space. The collision region comprises only 9.89%; 83.0% of $\tilde{\mathcal{R}}$ corresponds to states in which there is only one minimal strategy. Also, 6.52% holds two solutions; 0.602% holds three solutions; 0.0265% holds four solutions; and 0.00212% holds five strategies, which is the maximum for this problem.

Figure 4.16 presents an example in which there are three rotating robots in the plane that move along independent roadmaps.

Figure 4.17 shows another "H"-shaped roadmap coordination problem; however, in this case there are three robots, and they rotate along the roadmaps. This problem is perhaps one of the most complex in terms of solution alternatives; one minimal quotient strategy out of 16 is represented in the figure.

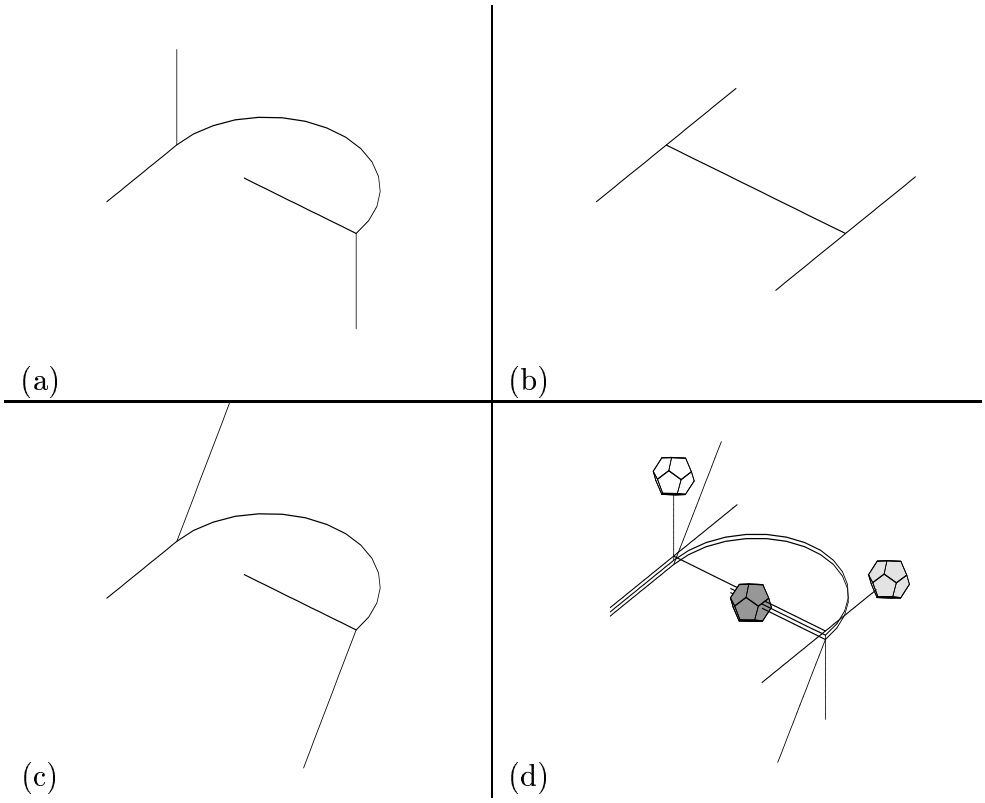


Figure 4.13 Parts (a), (b), and (c) show the independent roadmaps for $\mathcal{A}^1$, $\mathcal{A}^2$, and $\mathcal{A}^3$, respectively. Part (d) shows the initial positions on the roadmaps.

4.5 Unconstrained Motion Planning

Sections 4.3 and 4.4 have provided methods for coordinating multiple robots when the robots are restricted to fixed paths or independent roadmaps. These restrictions were motivated by savings from the complexity of the complete composite configuration space. In some applications, however, it may be desirable to analyze the complete state space. In this section, we present an algorithm that determines one discrete-time minimal strategy on the unconstrained state space, $X = \mathcal{C}^1_{valid} \times \mathcal{C}^2_{valid} \times \cdots \times \mathcal{C}^N_{valid}$. We present some computed examples for the case in which there are two translating robots in $\Re^2$.

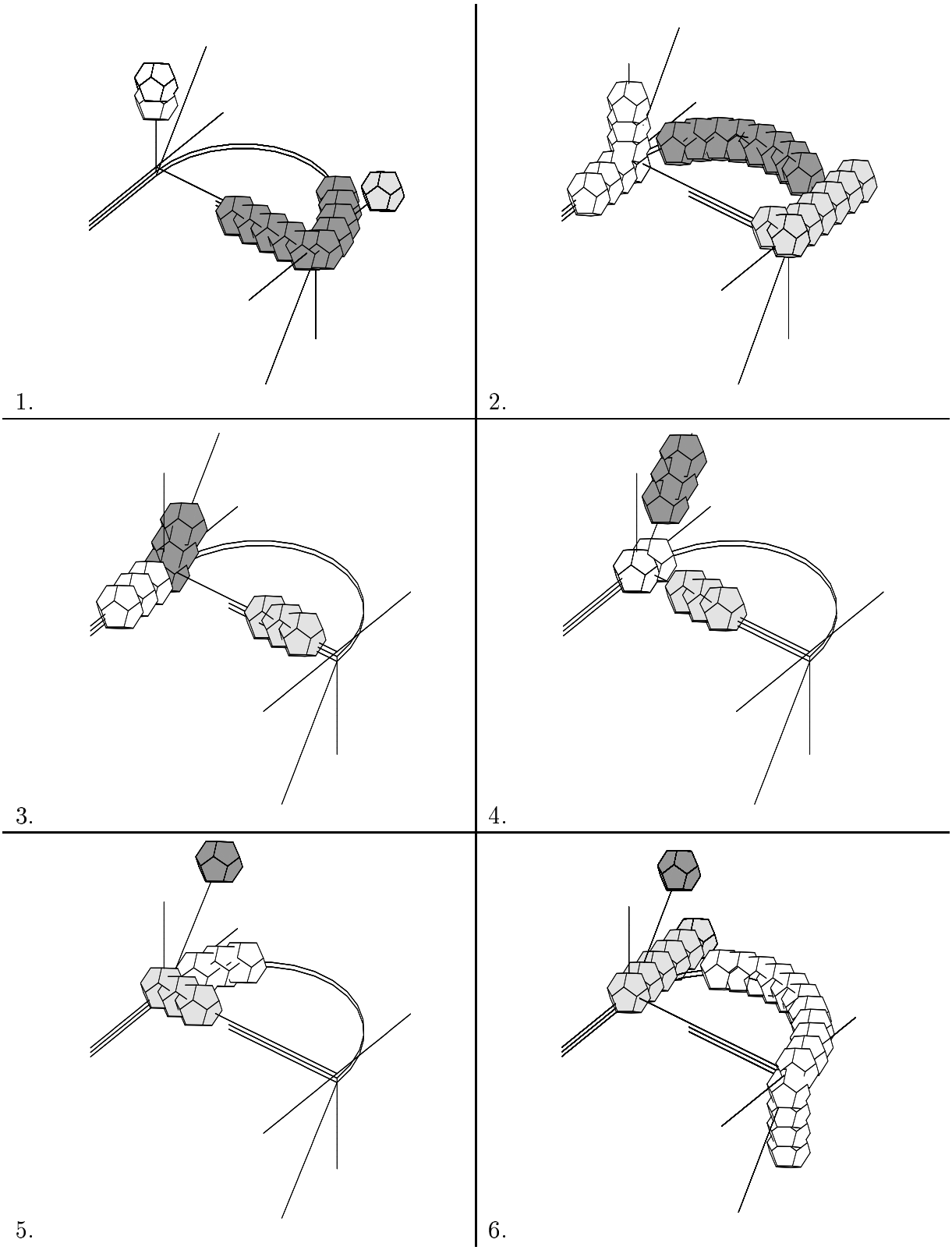


Figure 4.14 A representative of one of four minimal quotient strategies.

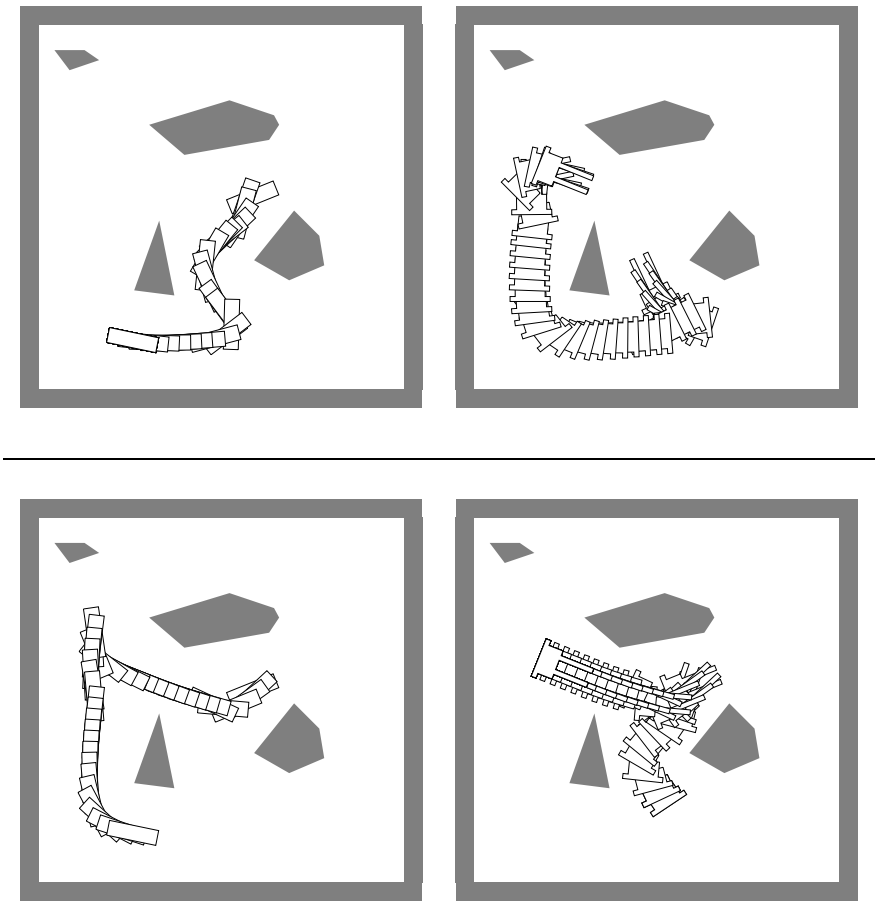


Figure 4.15 Two of five minimal quotient strategies for a two-robot problem with rotation.

4.5.1 Concepts and definitions

We first choose a vector, β, such that a linear scalarizing function, H, is defined using Equation (4.12). We consider discrete time, as in Sections 4.3 and 4.4, which results in stages. As opposed to a point goal in X, we allow each robot goal to be a subset, $X_G^i \subset X^i$.

For each $\mathcal{A}^i$, a discrete-time state transition equation is used:

$$x_{k+1}^i = \begin{bmatrix} x_k^i[1] + \|v^i\| \Delta t \; cos(u_k) \\ x_k^i[2] + \|v^i\| \Delta t \; sin(u_k) \end{bmatrix}. \tag{4.19}$$

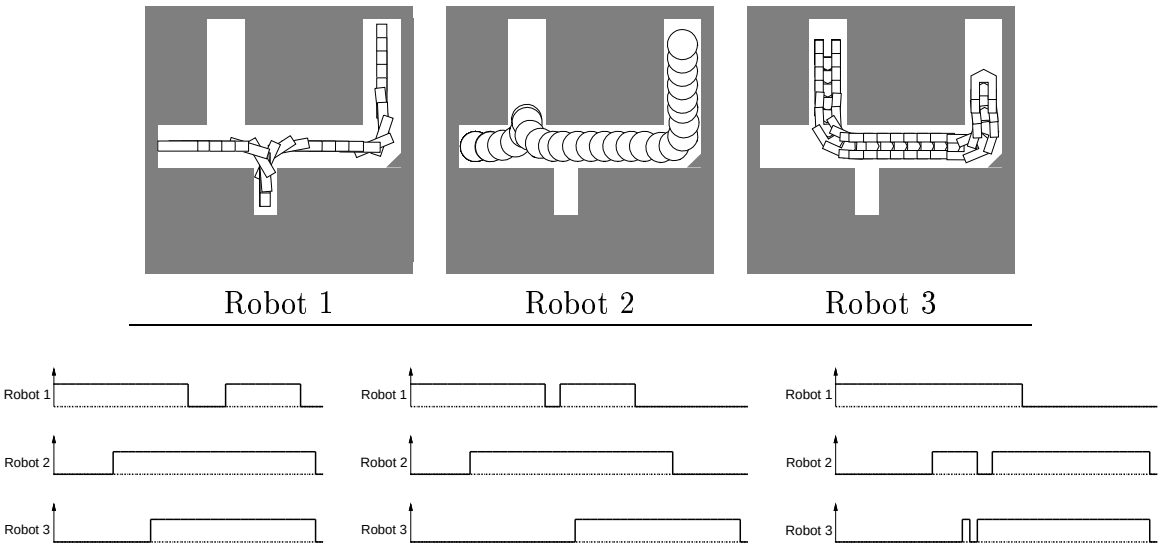


Figure 4.16 An example that has three minimal quotient strategies.

This is equivalent to the motion model in Section 3.3.2, and in this context, $U^i = [0, 2\pi) \cup \emptyset$, and $\|v^i\|$ represents the speed.

Suppose that at some stage k, the optimal strategy is known for each stage $i \in \{k, \ldots, K\}$. The loss obtained by starting from stage k, and implementing the portion of the optimal strategy $\{\gamma_k^*, \ldots, \gamma_K^*\}$, can be represented as

$$L_k^{i*}(x_k) = \sum_{k'=k}^{K} \left\{ l_{k'}^i(x_{k'}^i, u_{k'}^i) + \sum_{j \neq i} c_{k'}^{ij}(x(\cdot)) \right\} + q^i(x_{K+1}^i). \tag{4.20}$$

The function $L_k^{*i}(x_k)$ represents the *cost-to-go*. For this context, we modify the definition of $q^i(x_{K+1}^i)$ in (4.8), by replacing $x^i(T) = x_{goal}^i$ with $x^i(T) \in X_G^i$.

We can convert the cost-to-go functions into a scalar function by applying $H(\gamma, \beta)$ to obtain

$$H_k^* = \sum_{i=1}^{N} \beta_i L_k^{i*}(x_k). \tag{4.21}$$

Above, H_k^* represents a single cost-to-go function, which implicitly assumes that β is given.

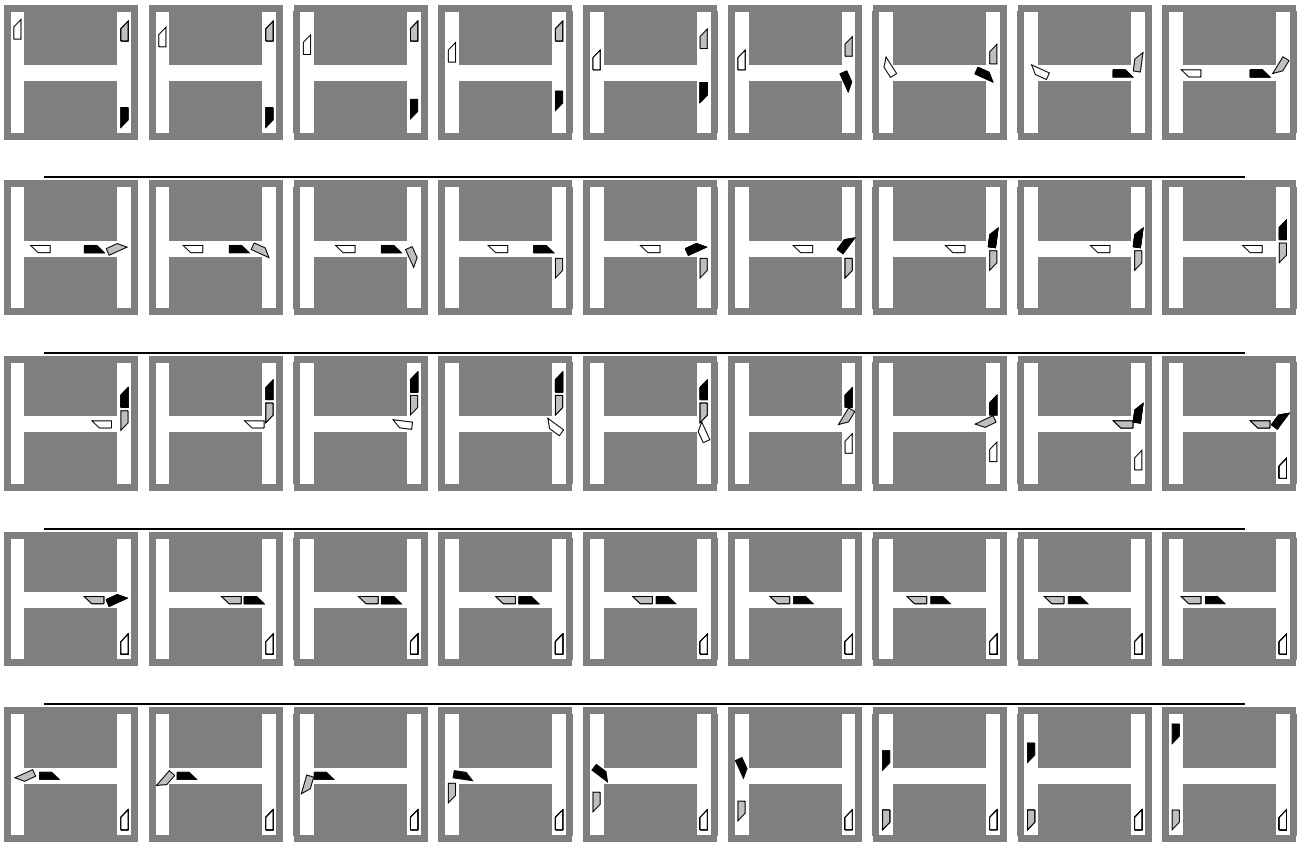


Figure 4.17 One solution out of 16 is shown for three rotating robots.

The principle of optimality [105] implies that H_k^* can be obtained from H_{k+1}^* by selecting an optimal value for u_k. The following recurrence represents the principle of optimality for our context:

$$H_k^*(x_k) = \min_{u_k \in U} \left\{ \sum_{i=1}^{N} \beta_i \, l_k^i(x_k, u_k) + \sum_{i=1}^{N} \sum_{j \neq i} \beta_i \, c_k^{ij}(x^i(\cdot)) + H_{k+1}^*(x_{k+1}) \right\}. \tag{4.22}$$

For each choice of u_k, x_{k+1} is obtained by applying f_k^i for each $i \in \{1, \ldots .N\}$. The boundary condition for this recurrence is given by

$$H_{K+1}^*(x_{K+1}) = \sum_{i=1}^{N} \beta_i q^i(x_{K+1}^i). \tag{4.23}$$

4.5.2 Algorithm presentation

The computational issues are similar to those discussed in Sections 2.6.3 and 3.4.2. Optimal strategies are determined numerically, by successively building approximate representations of H_k^*, and by using linear interpolation. We begin with stage $K+1$, and repeatedly apply (4.22) to obtain the optimal actions. Due to stationarity, $\gamma^* = \{\gamma_1^*, \ldots, \gamma_K^*\}$ after some number of iterations. The conditions of stabilization of H_k^* are similar to those for L_k^*, which are given in Section 3.4.2. To execute a strategy, the robots use (4.22) and the final cost-to-go representation, which we call H_1^*.

The complexity analysis is straightforward. Let $|Q|$ denote the number of cells per dimension in the representation of X. Let n denote the dimension of X. Let $|U|$ denote the number of actions per robot that are considered. The space complexity of the algorithm is $O(|Q|^n)$, and the time complexity of each iteration is $O(|Q|^n \ |U|^N)$. The number of iterations required is directly proportional to the number of stages required for the longest (in terms of stages) optimal strategy that reaches the goal.

4.5.3 Computed examples

In this section, we present two computed examples that were obtained with the algorithm described in this section. Because the algorithm is similar to that used in Chapter 2, the computational performance issues in Section 2.6.5 are relevant in this section. In this context, however, the dimension of the state space increases linearly with the number of robots, leading to poor computational performance (which provides motivation for using roadmap coordination or fixed-path coordination). For the computational examples that we have considered, the state space is four-dimensional, and each axis is divided into 20 cells.

Both examples involve motion planning for two robots, which are allowed to independently translate in $\Re^2$ (without restriction to a path or roadmap). For the problem in Figure 4.18, (4.12) was used with $\beta_1 = \beta_2 = \frac{1}{2}$. In the solution, neither robot is required to wait; they move around each other. Figure 4.19 concludes the computed examples of this chapter by returning to the example from Section 4.1. The top solution is found when $\beta_1 = \beta_2 = \frac{1}{2}$, which is equivalent to minimizing the total aggregate time. The lower example shows the solution when $\beta_1 = \frac{19}{20}$ and $\beta_2 = \frac{1}{20}$. For this case $\mathcal{A}^1$ (which is initially the rightmost robot) receives a greater priority, and is allowed to execute its time-optimal solution, while $\mathcal{A}^2$ is forced to wait. These represent the solution possibilities that were discussed in Section 4.1.

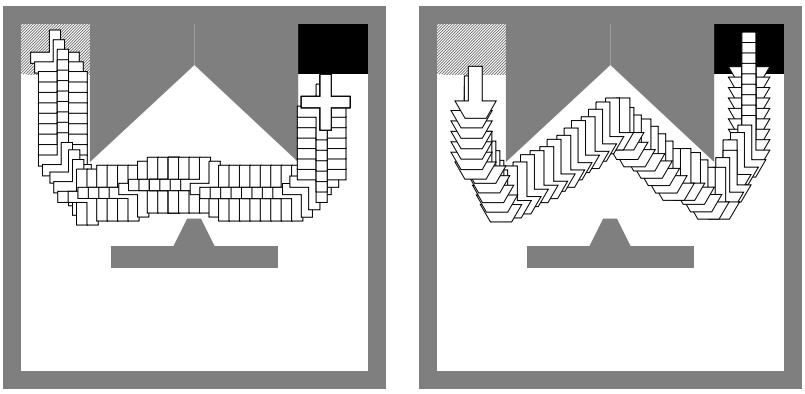


Figure 4.18 One representative minimal quotient strategy is given for two robots, allowed to translate in $\Re^2$.

4.6 Discussion and Conclusions

We have presented a general method for multiple-robot motion planning that is centered on a concept of optimality with respect to independent objectives. Strategies are determined that simultaneously optimize an independent performance criterion for each

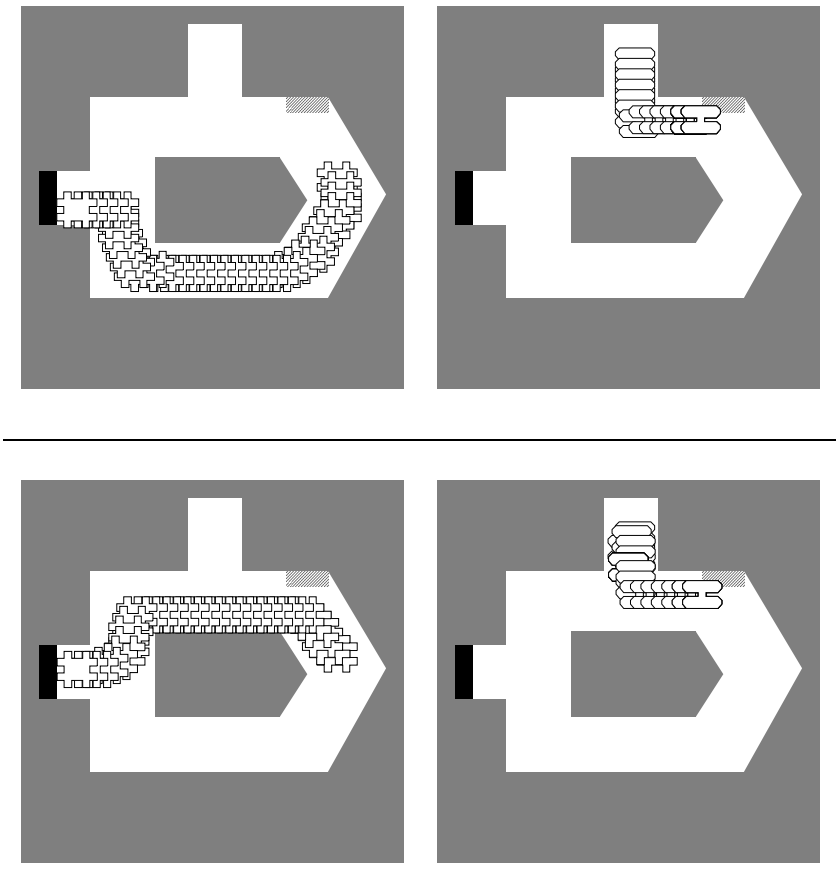


Figure 4.19 Two alternative solutions are presented for the example problem from Section 4.1.

robot. In addition, a general spectrum is defined between decoupled and centralized planning, in which we introduce optimal coordination along independent roadmaps.

It might now be questioned whether there could be multiple-robot motion planning methods that are between roadmap coordination and unconstrained motion planning, along the spectrum discussed in Section 4.1. To describe this middle ground, we borrow some concepts from algebraic topology [162]. A path can be considered as a 1-simplex, and each roadmap can be considered as a one-dimensional (singular) complex. The simplexes in this complex are connected by combinations of their boundaries, which are the endpoints of the paths. The coordination space is formed by planning on the Cartesian product of 1-complexes, which results in a N-complex for N robots. One natural generalization would be to replace each roadmap by an m-dimensional complex, which represents a connected structure in the configuration of each robot. For example, instead of a network of curves, we could consider a 2-complex of 2-manifolds, in which combinations of boundaries are connected. The resulting coordination space is of dimension Nm (assuming all complexes are of the same dimension). The principles presented in this chapter could then be applied by treating the complex as a state space; however, the larger challenge for this problem would be to construct these higher-dimensional complexes in the configuration space.

We next discuss several points that could provide a basis for future research. One useful benefit of the algorithms presented in this chapter is that the minimal quotient strategies from all initial states are represented (for a fixed goal). This could be useful if we are interested in repeatedly returning the robots to some goal positions without colliding, if the initial locations vary. We could alternatively exchange the initial state and goal states in the algorithms. This would produce a representation of minimal quotient strategies to all possible goals, from a fixed initial state. This initial state can be interpreted as a "home" position for each of the robots. After running the algorithm, the

robots can repreatedly solve different goals and return to the home position by reversing the strategy.

Coordination on roadmaps provides enough maneuverability for most problems; however, in general, completeness with respect to the original problem is lost when restricted to roadmaps. Roadmaps have traditionally been determined for motion planning of a single robot, and some additional issues can be considered when constructing roadmaps for the purpose of coordination. For example, if each roadmap contains at least one configuration that is reachable by the robot, and the robot avoids collisions with the other robots, regardless of their configurations, then completeness should be maintained (with the possible exception of some pathological cases). For example, we could give each robot an initial configuration in a home position or "garage," in which other robots are not allowed to enter. In [60], prioritization is introduced, and successive motion plans are constructed to prevent collision with robots of higher priority. One could extend prioritized path planning to *prioritized roadmap construction*. Consider, for example, the greater amount of coordination flexibility that arises in multiple-lane streets for automobiles, as opposed to one-lane streets. A similar principle could be applied to the construction of roadmaps for multiple robots.

To reduce the computational cost at the expense of losing completeness, traditional prioritization can be generalized within our framework. Suppose that we wish to coordinate nine robots along fixed paths. Rather than directly prioritizing the motions or building a nine-dimensional coordination space, consider dividing the robots into three groups of three. For each group, the algorithm in Figure 4.6 (or a variation of it) can be applied to determine a strategy that coordinates the three robots. These three strategies could be constructed successively, by interpreting the higher-priority robots as moving obstacles, and providing nonstationary strategies. A better approach might be to consider each of the strategies as a single path that simultaneously moves three robots (which

are then considered as a single robot). The algorithm in Figure 4.6 could then be directly applied to coordinate each of the three strategies, considered as fixed paths. Such issues such as the choice of groupings, and choices between prioritizing and coordinating, must be addressed. The analog of prioritization in noncooperative game theory is the Stackelberg equilibrium [6], which assumes a hierarchical structure of leaders in which higher-level decision makers can make decisions before others. This solution concept could also be relevant for some multiple-robot motion planning problems.

We next discuss a possible improvement to the computational performance of the algorithms. Hierarchical decomposition is often used for motion planning problems, to reduce the computations by varying the sizes of collision-free cells. The same principle might be applicable to representations of a coordination space, or state space. Because we are concerned with optimality, special concern must be given to ensure that the hierarchical representation yields a solution that is approximately optimal. For example, H_k^* from Section 4.5.2 may or may not be successively well-approximated by linear interpolation and a hierarchical representation. It remains to be seen whether nonuniform decompositions can be exploited.

In Section 4.2.3, we argued the equivalence of minimal quotient strategies to other established forms of optimality. However, we can change the general loss functional of Equation (4.6), to yield criteria for which some of the equivalence disappears. For example, the equivalence between admissible Nash equilibria and minimal quotient strategies has occurred because of the minimal form of interaction that occurs between the robots, through c^{ij}. If a robot received a penalty that is inversely proportional to its distance to other robots, then the two solution concepts become distinct. Fortunately, the principle of optimality can be applied to obtain solutions to this and other forms of optimality [6].

We have presented a general method for multiple-robot motion planning that is centered on a concept of optimality with respect to independent objectives. Strategies are

determined that simultaneously optimize an independent performance criterion for each robot. In addition, a general spectrum is defined between decoupled and centralized planning, in which we introduce optimal coordination along independent roadmaps.

CHAPTER 5

TOWARD BROADER MOTION PLANNING CONCEPTS

5.1 Introduction

Each of the previous three chapters presented analysis and computation methods for a particular class of motion planning problems. Specific contributions were indicated with respect to previous research for each class. This chapter shows how the game-theoretic concepts can be used to generalize and unify the analysis and computation methods across different classes of motion planning problems. The emphasis here is on defining models and formulating concepts, as opposed to presenting computed examples. The discussion in this chapter provides a basis for future research, under the direction taken in this dissertation.

Section 5.2 presents the most general discrete-stage game structure that is defined in this dissertation. Section 5.3 defines a game that incorporates all of the essential concepts from Chapters 2 through 4. Section 5.4 presents several game formulations that characterize additional robotics problems. Section 5.5 concludes this chapter by discussing issues that result from generalizing and comparing new motion planning problems within a game-theoretic framework.

5.2 Principle Modeling Components

This section provides a generic game formulation that can apply to a wide variety of robotics problems. This formulation is a variation of that presented in a dynamic non-cooperative game context [6], and in a robotics context [112]. Continuous-time versions exist, some of which are discussed in [6].

The components in this formulation will be enumerated, and a description of the components will follow. Most of this structure has been utilized in the previous chapters in this dissertation. Consider the following:

1. An index set, $\mathbf{N} = \{1, 2, \ldots, N\}$, of N decision makers

2. An index set, $\mathbf{K} = \{1, 2, \ldots, K\}$, that denotes the *stages* of the game

3. A set, X, called the *state space*. The state of the game, x_k, at stage k, belongs to X.

4. A set, U_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}$, which is called the *action set* of the i^{th} decision maker at stage k. The *action*, u_k^i, at stage k, belongs to U_k^i.

5. A set, Θ_k^a, defined for each $k \in \mathbf{K}$, which is called the *control action set for nature* at stage k. The *control action for nature*, θ_k^a, at stage k, belongs to Θ_k^i.

6. A function, $f_k : X \times U_k^1 \times \ldots \times U_k^N \times \Theta_k^a \rightarrow X$, defined for each $k \in \mathbf{K}$ so that

$$x_{k+1} = f_k(x_k, u_k^1, \ldots, u_k^N, \theta_k^a), \tag{5.1}$$

   is a *state transition equation.*

7. A set, Y_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}$, and called the *sensor space* of the i^{th} decision maker at stage k, to which the sensed observation y_k^i belongs at stage k.

8. A set, $\Theta_k^{s,i}$, defined for each $i \in bfN$ and $k \in \mathbf{K}$, which is called the *sensing action set for nature* at stage k. The *sensing action for nature*, $\theta_k^{s,i}$, at stage k, belongs to $\Theta_k^{s,i}$.

9. A function, h_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}$, so that

$$y_k^i = h_k^i(x_k, \theta_k^{s,i}), \tag{5.2}$$

which is the instantaneous observation equation of the i^{th} decision maker concerning the value of x_k.

10. A finite set, η_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}$ as a subset of all actions and observations made by decision makers at any previous stage, $\{u_1^1, \ldots, u_k^N, y_1^1, \ldots, y_k^N\}$.

11. A set of all possible values for η_k^i, denoted by N_k^i, which is called the *information space* for the i^{th} decision maker at stage k.

12. A set, Γ_k^i, of mappings $\gamma_k^i : N_k^i \rightarrow U_k^i$, which are the *strategies* available to the i^{th} decision maker at stage k. The combined mapping $\gamma^i = \{\gamma_1^i, \gamma_2^i, \ldots, \gamma_K^i\}$ is a *strategy* for the i^{th} decision maker, and the set Γ^i of all such mappings γ^i is the *strategy space* of the i^{th} decision maker. A *game strategy*, γ, represents a simultaneous specification of the strategy for each decision maker, and the space of game strategies is denoted by $\Gamma = \Gamma^1 \times \cdots \times \Gamma^N$.

13. An extended real-valued functional $L^i : (X \times U_1^1 \times \ldots \times U_1^N) \times (X \times U_2^1 \times \ldots \times U_2^N) \times \ldots \times (X \times U_K^1 \times \ldots \times U_K^N) \times \Theta \rightarrow \Re^+$, defined for each $i \in \mathbf{N}$, and called the *loss functional* of the i^{th} decision maker. The Cartesian product of all of nature's actions spaces is represented here as Θ.

Item 1 enumerates the decision makers in the game. For the problems considered in the previous chapters, each robot has been considered as a decision maker (nature can

also be considered as a decision maker, but we define nature in separate items). If several robots are controlled in unison, then we might consider the system of robots as a single, combined decision maker. In general, any agent that is capable of making decisions and interfering with the other decision makers should be considered as a decision maker.

Item 2 defines stages that correspond to times at which decisions are made during the play of the game. For the problems in the previous chapters, decisions were made at each Δt time increment. In general, decision making at regular intervals is not required. Suppose for instance that the decisions correspond to very high-level operations, that may have unpredictable completion times. This was implicitly assumed when fine-motion planning was taken into account with the part-carrying problem presented in Section 3.6. A continuum of stages can be considered, which results in a continuous-time differential game (e.g., [94]). In this case, the action spaces are replaced by a control input that is a function of time.

The state space is defined in Item 3. Configuration spaces have been embedded in all of the state space definitions presented thus far in this dissertation. It is well known in the literature (e.g., [109], [123]) that configuration space concepts provide a compact and useful characterization of motion planning problems. In general, however, the state space could incorporate additional information, such as environment modes, or could encode completely different information that is relevant to a particular robotic task. Some of the examples in Section 5.4 will define state spaces that do not explicitly encode a configuration space.

Items 4 through 6 define how changes in state can occur. Each decision maker has a set of actions available at every stage. The i^{th} decision maker has influence over the state x_k by applying the action u_k^i. The precise characterization of this influence is given in Item 6, the state transition equation. By choosing a control action θ_k^a from

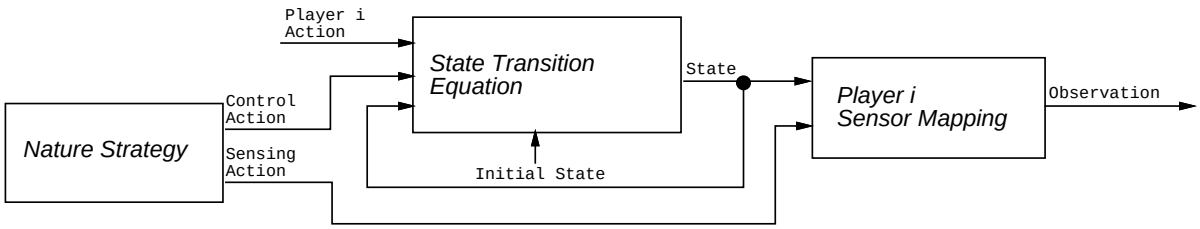


Figure 5.1 The effect of nature on the game.

the set Θ_k^a, nature can also influence the state transitions, either nondeterministically or probabilistically.

Definitions that are used for handling imperfect state information are presented in Items 7 through 11. Each decision maker at each stage has a *sensor space*, Y_k^i (or observation space), which encodes information regarding the state that is observed during stage k. In addition to a projection from the state space to the sensor space, this information is potentially corrupted by a sensing action, $\theta_k^{s,i}$, of nature, which is chosen from $\Theta_k^{s,i}$. Figure 5.1 depicts how nature can affect the decision-making problem through both control actions and sensing actions. Differential sensor observations have been considered in [142], and depend on the state at different times. Sensors of this type can be included in the formulation by replacing $h_k^i(x_k, \theta_k^{s,i})$ with $h_k^i(x_1, \ldots, x_k, \theta_k^{s,i})$.

Item 12 defines a strategy for each decision maker. Each decision maker conditions its actions on its information state at the given stage. This represents a deterministic (or pure) strategy; however, we can alternatively define randomized (or mixed) strategies. In this case a pdf of the form $p(u_k^i | \eta_k^i)$ is specified as the strategy. During each execution, the action u_k^i is chosen by sampling. With probabilistic representations, nature can be imagined as a decision maker that implements a randomized strategy.

Item 13 defines a loss functional for each of the decision makers, which guides the selection of strategies. The loss can generally be based on actions taken by any decision maker at any stage, and on the state trajectory. In this general form, the loss functionals

can also depend on nature. Recall that the sensing models are functions of state and nature. Because the information space is generated by a set of sensor observation and actions, the inclusion of nature in the definition of loss allows loss functionals that depend on information states. In some applications, it may be practical to express the desired performance in terms of information space. Suppose that a problem involves probabilistic uncertainty in configuration sensing; the task might simply be to minimize the variance of $p(x_k|\eta_k)$, which is a statistic (or function) of the information state. Examples of loss functionals that depend on nature are given in Section 5.4.

It has been assumed thus far that each decision maker knows all game components, including the loss functionals, of other decision makers. Another item could be introduced that reflects imperfect information that each decision maker has about the game itself. Problems of this type are quite realistic, yet are very difficult to model [73], [83]. The information of each decision maker could be represented, for example, as a probability density over a set of possible games. To make appropriate decisions, each decision maker must speculate about the knowledge that other decision makers have regarding the game. This type of second-guessing can progress for an infinite number of layers, which leads to a formidable modeling task. One approach to problems of this type is the Recursive Modeling Method, which finds strategies that are optimal in the expected sense by averaging over a finite number of layers [73].

5.3 Unifying the Concepts from Chapters 2-4

This section specializes the structure from Section 5.2 to minimally encompass the essential concepts from Chapters 2 through 4. A multiple-robot motion planning problem is defined in which each robot experiences uncertainty in configuration predictability, configuration sensing, environment predictability, and environment sensing. This pro-

vides an illustration of the generalization capabilities that arise from the mathematical framework presented in this dissertation.

5.3.1 Defining the game components

Suppose there are N robots, $\{\mathcal{A}^1, \ldots, \mathcal{A}^N\}$, that operate in a common workspace, $\mathcal{W}$. Each robot $\mathcal{A}^i$ might be a rigid robot (such as a mobile robot) or an articulated robot (such as a manipulator). Each robot is characterized by its n_i-dimensional configuration space, $\mathcal{C}^i$. There are static obstacles in the common workspace, and a free configuration space, $\mathcal{C}^i_{free}$ can be defined for each robot.

It will be assumed that for each robot, all four sources of uncertainty described in Section 1.2.2 are present. The combined effects of all these uncertainties on all robots can be considered as actions of a single nature player. However, to make the presentation more precise, we will define four nature players for each robot, each of which corresponds to a distinct source of uncertainty. The nature actions for the i^{th} robot will be represented as $\theta_k^{cp,i}$, $\theta_k^{cs,i}$, $\theta_k^{ep,i}$, and $\theta_k^{es,i}$.

The state space will be defined next. Recall that in Chapter 3, the set E of environment modes was assumed to be finite. In this section, suppose that the set of environment modes is $E = \Re^m$ for some m. The state space is formed by taking the Cartesian product of the configuration spaces of the robots and the set of environment modes. This results in a state vector representation of the form

$$x_k = \left[\begin{array}{cccccccccc} \mathbf{q}_k^1[1] & \cdots & \mathbf{q}_k^1[n_1] & \cdots & \mathbf{q}_k^N[1] & \cdots & \mathbf{q}_k^N[n_N] & e_k[1] & e_k[2] & \cdots & e_k[m] \end{array} \right]. \tag{5.3}$$

There are many locations in the state space in which two or more robots intersect in the workspace. The state space is therefore restricted to X_{valid}, which is obtained after the removal of all pairwise collision regions, as constructed in Chapter 4.

The state transition equation for robot i can be defined as

$$x_{k+1}^i = f_k^i(x_k, u_k^i, \theta_k^{cp,i}, \theta_k^{ep,i}), \tag{5.4}$$

in which f_k^i depends on x_k, not x_k^i. Thus, one robot is generally allowed to influence another robot's motions, or any robot can influence the environment mode.

The observation equation for robot i is $y_k^i = h_k^i(x_k, \theta_k^{cs,i}, \theta_k^{es,i})$. This equation can model sensing of both configurations and environments. For example, suppose $p(o_k^i|e_k)$ describes a sensing model that contains information about the environment, and $p(z_k^i|\mathbf{q}_k^i)$ is a sensing model that contains information about the current configuration. If these models are independent, then $y_k^i = [z_k^i \;\; o_k^i]$ simultaneously represents both sensing models and

$$p(y_k^i|x_k^i) = p(z_k^i, o_k^i|\mathbf{q}_k^i, e_k) = p(z_k^i|\mathbf{q}_k^i)p(o_k^i|e_k), \tag{5.5}$$

which is specified in terms of the individual sensing models.

An information state can be constructed as

$$\eta_k^i \subseteq \left(\bigcup_{i \in \mathbf{N}} \{u_1^i, \ldots, u_{k-1}^i, y_1^i, \ldots, y_k^i\} \right). \tag{5.6}$$

With this definition, each robot might have access to the actions and sensor observations that were made by the other robots.

Once the information space has been fixed, the definition of a strategy for each robot is simply the set of mappings $\gamma_k^i : N_k^i \rightarrow U_k^i$ for each k. A termination condition, TC_k^i, for the i^{th} robot at each stage is defined as a binary-valued mapping,

$$TC_k^i : N_k^i \rightarrow \{true, false\}. \tag{5.7}$$

We require that if $TC_k^i = true$, then $TC_{k+1}^i = true$. The game terminates when $TC_k^i = true$ for every robot. Let TC denote the set of termination conditions for all of the decision makers.

The next step is to define the loss functionals,

$$L^i(x_1, \ldots, x_{K+1}, u_1^1, \ldots, u_K^1, \ldots, u_1^N, \ldots, u_K^N, TC) =$$

$$\sum_{k=1}^{K} l_k(x_k, u_k^1, \ldots, u_k^N, TC_k) + l_{K+1}(x_{K+1}). \tag{5.8}$$

Let $\mathcal{L}$ represent a vector of N loss values, in which the i^{th} element corresponds to the loss for the i^{th} robot.

5.3.2 Generalizing planning concepts

This section provides generalizations of some of the planning concepts that were introduced in Chapters 2 through 4.

Forward Projections The forward projection concepts from Section 2.4 can be generalized to the current context. Suppose there is perfect state information and nondeterministic uncertainty. The single-step forward projection is

$$F_{k+1}(x_k, u_k^1, \ldots, u_k^N) = \{f_k(x_k, u_k^1, \ldots, u_k^N, \theta_k^{cp}, \theta_k^{ep}) \in X | \theta_k^{cp} \in \Theta_k^{cp}, \theta_k^{ep} \in \Theta_k^{ep}\}, \tag{5.9}$$

which is an extension of Equation (2.3). By using the state transition equation, the next state can be represented by a pdf, $p(x_{k+1}|x_k, u_k^1, \ldots, u_k^N)$.

Next consider the case of imperfect information. Let $\mathcal{N}_k$ denote the Cartesian product of information spaces

$$\mathcal{N}_k = N_k^1 \times N_k^2 \times \cdots \times N_k^N. \tag{5.10}$$

Let $\boldsymbol{\eta_k}$ denote $\{\eta_k^1, \ldots, \eta_k^N\}$, in which $\boldsymbol{\eta_k} \in \mathcal{N}_k$.

The (information) forward projection $\tilde{F}_{k+1}(\boldsymbol{\eta_k}, u_k^1, \ldots, u_k^N)$ can be defined as the set of all $\boldsymbol{\eta_{k+1}} \in \mathcal{N}_{k+1}$ such that if $y_{k+1}^i \in \eta_{k+1}^i$ then $y_{k+1}^i = h_k^i(x_{k+1}, \theta_{k+1}^{cs,i}, \theta_{k+1}^{es,i})$ for some $x_{k+1} \in F_k(x_k, u_k^1, \ldots, u_k^N)$, $\theta_{k+1}^{cs,i} \in \Theta_{k+1}^{cs,i}$, $\theta_{k+1}^{es,i} \in \Theta_{k+1}^{es,i}$, and $x_k \in F_k(\boldsymbol{\eta_k})$. The term $F_k(\boldsymbol{\eta_k})$

is a vector version of (2.7). For this case, $F_k(\boldsymbol{\eta_k})$ represents a subset of the Cartesian product of N state spaces.

Next consider a probabilistic version of the forward projection. Let $\mathbf{X}$ represent the Cartesian product of N copies of X. The information state can be described as a density on $\mathbf{X}$ of the form $p(\mathbf{x_k}|\boldsymbol{\eta_k})$. At stage k, the density on $\mathbf{X}$ after starting at $\boldsymbol{\eta_1}$ is given by $p(\mathbf{x_k}|\boldsymbol{\eta_1}, \gamma) =$

$$\int p(\mathbf{x_k}|\boldsymbol{\eta_{k-1}}, \gamma_{k-1}(\boldsymbol{\eta_{k-1}}))p(\boldsymbol{\eta_{k-1}}|\boldsymbol{\eta_{k-2}}, \gamma_{k-2}(\boldsymbol{\eta_{k-2}})) \cdots p(\boldsymbol{\eta_2}|\boldsymbol{\eta_1}, \gamma_1(\boldsymbol{\eta_1}))d\boldsymbol{\eta_{k-1}} \cdots d\boldsymbol{\eta_2}, \tag{5.11}$$

in which $\gamma_k(\boldsymbol{\eta_k})$ specifies the actions $\{u_k^1, \ldots, u_k^N\}$ at stage k. The i^{th} subspace of the first term in the integrand can be determined using

$$p(x_{k+1}|\eta_k^i, u_k^1, \ldots, u_k^N) = \int p(x_{k+1}|x_k, u_k^1, \ldots, u_k^N)p(x_k|\eta_k^i)dx_k, \tag{5.12}$$

which depends on the state transition equation and the density representation of the information state η_k^i.

Each of the remaining terms of (5.11) can be reduced to

$p(\boldsymbol{\eta_{k+1}}|\boldsymbol{\eta_k}, \gamma_k(\boldsymbol{\eta_k})) =$

$$p(y_1^1, \ldots, y_{k+1}^N, u_1^1, \ldots, u_k^N|y_1^1, \ldots, y_k^N, u_1^1, \ldots, u_k^1) = p(y_{k+1}^1, \ldots, y_{k+1}^N|\boldsymbol{\eta_k}, u_k^1, \ldots, u_k^N). \tag{5.13}$$

This reduction occurs because most of the sensing and action history appears on both sides of the density expression. With the assumption that the robots receive independent sensor observations, the right side of (5.13) becomes

$$p(y_{k+1}^1, \ldots, y_{k+1}^N|\boldsymbol{\eta_k}, u_k^1, \ldots, u_k^N) = \prod_{i \in \mathbf{N}} p(y_{k+1}^i|\boldsymbol{\eta_k}, u_k^1, \ldots, u_k^N), \tag{5.14}$$

in which each term can be further reduced to

$p(y_{k+1}^i|\boldsymbol{\eta_k}, u_k^1, \ldots, u_k^N) = \int p(y_{k+1}^i|x_{k+1})p(x_{k+1}|\eta_k^i, u_k^1, \ldots, u_k^N)dx_{k+1} =$

$$\int \int p(y_{k+1}^i|x_{k+1})p(x_{k+1}|x_k, u_k^1, \ldots, u_k^N)p(x_k|\eta_k^i)dx_k dx_{k+1}, \tag{5.15}$$

and all three terms in the final integrand are known. The density $p(y^i_{k+1}|x_{k+1})$ is inferred from the sensing model; $p(x_{k+1}|x_k, u^1_k, \ldots, u^N_k)$ is inferred from the control model; $p(x_k|\eta^i_k)$ is the density representation of the current information state.

These expressions have provided generalizations of the forward projection concepts from Section 2.4 by applying the game-theoretic framework.

Performance Preimages The performance preimage concepts from Section 2.5 can also be generalized. Suppose there is perfect information and the nondeterministic representation is considered. Under the implementation of a strategy, the following vector value can be obtained:

$$\check{\mathcal{L}}(x_1, \gamma) = \left[\sup_{\gamma^\theta \in \Gamma^\theta} L^1(x_1, \gamma, \gamma^\theta) \quad \cdots \quad \sup_{\gamma^\theta \in \Gamma^\theta} L^N(x_1, \gamma, \gamma^\theta) \right]. \tag{5.16}$$

The i^{th} component of $\check{\mathcal{L}}(x_1, \gamma)$ corresponds to the worst-case loss for the i^{th} robot. Note that since the sup is applied to each component individually, there might not exist a single nature strategy γ^θ that produces $\check{\mathcal{L}}(x_1, \gamma)$ as a vector of losses. The vector $\check{\mathcal{L}}(x_1, \gamma)$ represents the individual worst-case losses, from the independent perspectives of each decision maker.

Let V denote a subset of $\Re^N$. A performance preimage can be defined on the state space as

$$\check{\pi}_x(\gamma, V) = \{x_1 \in X | \check{\mathcal{L}}(x_1, \gamma) \in V\}. \tag{5.17}$$

With the probabilistic representation of uncertainty and perfect state information, the following can be defined:

$$\bar{\mathcal{L}}(x_1, \gamma) = \int \mathcal{L}(x_1, \gamma, \theta) p(\theta) d\theta = \left[\int L^1(x_1, \gamma, \theta) p(\theta) d\theta \quad \cdots \quad \int L^N(x_1, \gamma, \theta) p(\theta) d\theta \right]. \tag{5.18}$$

For a choice of $V \subseteq \Re^N$, a performance preimage can be defined as

$$\bar{\pi}_x(\gamma, V) = \{x_1 \in X | \bar{\mathcal{L}}(x_1, \gamma) \in V\}. \tag{5.19}$$

The performance preimage can be specialized to evaluate a strategy with respect to the loss of a single robot. Let $R \subset [0, \infty)$. Suppose that V is chosen as that set of all elements in $\Re^N$ such that the i^{th} coordinate lies in R. The performance preimage in this case yields the maximal subset of the state space from which the (expected or worst-case) loss of the i^{th} robot lies in R. The interactions between two or more robots can be evaluated in a similar way.

Next suppose that there is imperfect information, but the information spaces for different robots are identical. Hence, there is a common information space, which can be denoted as N_k. With a nondeterministic representation, the i^{th} component of $\check{\mathcal{L}}(\eta_1, \gamma)$ is

$$\sup_{\gamma^\theta \in \Gamma^\theta} L^i(\eta_1, \gamma, \gamma^\theta), \tag{5.20}$$

and

$$\check{\pi}(\gamma, V) = \{\eta_1^i \in N_1 | \check{\mathcal{L}}(\eta_1, \gamma) \in V\}, \tag{5.21}$$

for some $V \subseteq \Re^N$. With a probabilistic representation, the i^{th} component of $\bar{\mathcal{L}}(\eta_1, \gamma)$ is

$$\int L^i(\eta_1, \gamma, \theta) p(\theta) d\theta, \tag{5.22}$$

and

$$\bar{\pi}(\gamma, V) = \{\eta_1 \in N_1 | \bar{\mathcal{L}}(\eta_1, \gamma) \in V\} \tag{5.23}$$

for some $V \subseteq \Re^N$.

The most complex case is now considered, in which there is imperfect information and independent information spaces. These preimages will be defined as subsets of $\mathcal{N}_1$.

Each L^i can be expressed as $L^i(\boldsymbol{\eta_1}, \gamma, \gamma^\theta)$, and the vector losses can be expressed as $\mathcal{L}(\boldsymbol{\eta_1}, \gamma, \gamma^\theta)$.

With a nondeterministic representation, the i^{th} component of $\check{\mathcal{L}}(\boldsymbol{\eta_1}, \gamma)$ can be expressed as

$$\sup_{\gamma^\theta \in \Gamma^\theta} L^i(\eta_1^1, \eta_1^2, \ldots, \eta_1^N, \gamma, \gamma^\theta), \tag{5.24}$$

and

$$\check{\pi}(\gamma, V) = \{\boldsymbol{\eta_1} \in \mathcal{N}_1 | \check{\mathcal{L}}(\boldsymbol{\eta_1}, \gamma) \in V\} \tag{5.25}$$

for some $V \subseteq \Re^N$. Finally, with a probabilistic representation, the i^{th} component of $\bar{\mathcal{L}}(\boldsymbol{\eta_1}, \gamma)$ is

$$\int \mathcal{L}(\boldsymbol{\eta_1}, \gamma, \theta) p(\theta) d\theta, \tag{5.26}$$

and

$$\bar{\pi}(\gamma, V) = \{\boldsymbol{\eta_1} \in \mathcal{N}_1 | \bar{\mathcal{L}}(\boldsymbol{\eta_1}, \gamma) \in V\} \tag{5.27}$$

for some $V \subseteq \Re^N$.

These expressions indicate that the performance preimage concept can be used in very general contexts.

5.3.3 Determining solutions

Several different types of optimality concepts have been considered in this dissertation. Because the mathematical structure in this section encompasses the motion planning problems considered in the previous chapters, many of same concepts are relevant. For a single robot that faces all four sources of uncertainty with a probabilistic representation, a suitable strategy is one that minimizes the loss in the expected sense. With

the nondeterministic representation, a strategy that provides the least worst-case loss is most desirable. With multiple robots that face uncertainties, the situation becomes more complicated. With a probabilistic representation, the situation is mathematically equivalent to a Markov game [147]. One possible solution in this context is to define a partial ordering on strategies by comparing vectors of expected losses, and to take the minimal elements with respect to the partial ordering. For problems of this type, a computational approach can still be developed that utilizes the principle of optimality. Straightforward application of the computational techniques in this dissertation, however, would probably lead to prohibitive computational expense for many interesting problems. Hence, the development of improved computational techniques remains an important direction for future research.

5.4 Additional Problem Formulations

This section indicates different types of robot planning problems that can be characterized with the general formulation from Section 5.2. These are provided to indicate the expressive power of the formulation and potential uses of it in future research. These characterizations, therefore, do not represent proposed solutions to problems, and it is generally expected that additional issues and more detailed analysis would have to be considered in each case.

A meta-level motion planning problem Suppose a robot is confronted with the task of selecting a computation method (or methods) that will lead to the solution of a basic motion planning problem, and is given statistical information about the likelihood that a method will succeed and its expected computational cost. The goal is to produce a solution (e.g., find a collision-free path) in a minimal amount of time. One motivation

is that there are simple ways to attack a given problem that often provide a solution. For example, consider connecting initial and goal configurations with a straight line. This method requires little computation time and may solve a significant number of problems. It seems nearly useless as a general motion planning approach; however, as one method among many to choose from, it can be quite powerful when used appropriately. This low execution cost was the motivation behind the randomized construction of configuration-space graphs in [97]. The view presented here is inspired by the case-based approach taken in [143] and is related to the decision-making approaches in [7], [55].

Let $\mathcal{M}$ denote a space of basic motion planning problems, which can be generated by considering combinations of possible free configuration spaces (environments), initial configurations, and goal configurations. In practice, this set need not be enumerated. A state space, X, is defined as $\mathcal{M} \times \{0, 1\}$. Let 1 represent the condition that the problem has not been solved; otherwise, it has been solved. Initially, the system is in some state, $x_1 = (m, 1)$, which implies that a motion planning problem, $m \in \mathcal{M}$, has been chosen, and it is not yet solved. The action set U corresponds to a collection of algorithms that might solve the motion planning problem. The control action set for nature is binary, $\Theta_k^a = \{0, 1\}$. If $\theta_k^a = 0$, then nature allows the problem to be solved; otherwise, nature chooses $\theta_k^a = 1$, and the problem remains unsolved. Hence, the state transition equation is

$$f(x_k, u_k, \theta_k^a) = \begin{cases} (m, 0) & \text{if } \theta_k^a = 0 \\ (m, 1) & \text{otherwise} \end{cases} . \tag{5.28}$$

Nature represents a form of uncertainty, which is modeled probabilistically. Hence, the state transitions can be considered as $P(x_{k+1}|x_k, u_k)$.

Assume that the final component of the state space can be directly observed, and a sensing model is used on $\mathcal{M}$. The sensor space Y has the same interpretation as a feature space in statistical pattern recognition [46], [51]. Rather than making decisions directly

from $\mathcal{M}$, a projection is formed into a feature space with h. The information space, N, is generated by the sensor space Y and the set of previous actions.

A loss functional can be defined as

$$L = \sum_{i=1}^{k} l_k(u_k), \tag{5.29}$$

in which $l_k(u_k)$ represents the time required to apply u_k.

Successfully using this approach requires that several issues still be addressed. A *sensor space* must be defined for which $h(m)$ can be efficiently computed and for which appropriate, informed decisions can be made. The extraction of useful features for a motion planning problem is challenging, and some features were obtained in [143]. Note that $p(\theta_k|x_k, u_k)$ must be determined. This could, for example, be determined by constructing a cellular partition of Y, and using the statistical average of successes versus total trials in each cell. We generally allow any motion planning method, u_k, to be attempted multiple times. Complete methods, such as cylindrical algebraic decomposition [4], [39], will always successfully terminate; however, the cost is usually extremely high. For deterministic motion planning methods, we know that the probability of success is zero in subsequent stages if it has failed once. Randomized methods, such as pure Brownian motion, might produce nearly stage-independent probabilities.

Pursuit and evasion Problems of pursuit and evasion have been considered throughout game theoretic literature [81], [94], [199], [200], [201] and have particular relevance in some robotic applications. These problems involve interactions between two decision makers, a pursuer and an evader, that have opposing interests. The pursuer views the evader as a moving goal that must be intercepted, which is similar to tracking [64], [126], [146] or target acquisition [15], [43], [100]. Game-theoretic concepts have recently been used to analyze and determine solutions to a class of linear tracking problems [168].

From the point of view of the evader, the situation becomes a type of collision avoidance problem.

A brief description of a pursuit-evasion problem is given. Suppose that the pursuer and evader are robots that are capable of translating in the plane. Let X represent the four-dimensional composite configuration space. If there are no static obstacles, the state space can be reduced to two dimensions by viewing the evader with respect to a moving coordinate system that is attached to the pursuer [94]. A state transition equation can be defined by combining independent local motion models for each of the robots. Suppose, for example, that the pursuer has control over the first two coordinates in X, and the evader has control over the final two coordinates in X. The loss functional for the pursuer can be defined in terms of the distance to the evader, or a cost can be applied at the final stage if the robot has not captured the evader. Incremental costs can also be added to induce a preference for faster captures. The loss functional of the evader can be the negation of the pursuer's loss functional. The pursuit-evasion scenario can thus be viewed as a zero-sum game [6].

Motion planning with vision feedback It has been recognized in the literature that a vision system can provide useful information for planning [64], [66], [84]. We briefly indicate how information space concepts can be applied to formulate this type of problem, by characterizing the imaging system with a sensor model.

Let X be the configuration space of a multiple-link manipulator, such as the one indicated in Figure 5.2. Suppose that the task is to move the end effector to some specified subset of the workspace. This workspace subset could be mapped into the configuration space, as done for the problems in Section 3.6; however, in many applications it is preferable to avoid calibrations of the end effector in the workspace. One way to avoid this

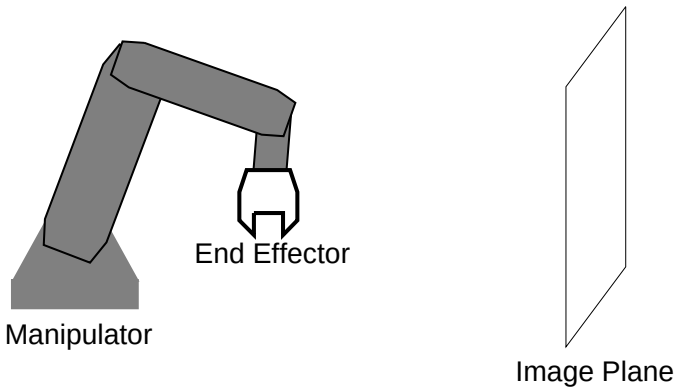


Figure 5.2 Motion planning with vision feedback.

is to directly monitor the accomplishment of the task by tracking image features that depend on the location of the end effector.

Let Y represent the feature space. A mapping can be constructed from the joint space directly into the feature space. This mapping results from a composition of the mappings that includes the projection model for the camera. This mapping can be expressed as $y_k = f(x_k, \theta_k^a)$. In general this mapping can be quite complicated, even without the consideration of a sensing action for nature that interferes with the observation. The mapping f typically has many singularities that directly affect the relationship between transitions in the state space and transitions in the feature space [84].

A goal region could be defined in the state space; however, it may be preferable to define the goal region directly in the feature space. The information space for this problem is generated by the set of action and sensing history. The approximation techniques discussed in Section 2.6.4 could be applied to successfully plan on the information space. For example, the information space could be approximated with moments, or limited memory could be used, as for the computed examples in Section 2.6.6.

Multiple exploring robots This problem involves multiple decision makers that have imperfect information about the game, and is adapted from [73]. Suppose there are two

robots, $\mathcal{A}^1$ and $\mathcal{A}^2$, that are both dedicated to gathering information about unknown terrain. This type of problem is conceptually similar to reducing environment-sensing uncertainty by making observations about the environment.

Suppose the initial configurations of the robots are as depicted in Figure 5.3. Region 1 and Region 2 designate possible vantage points from which the robots can make observations about the environment. A single-stage decision problem will be defined. Suppose that $\mathcal{A}^1$ has three possible actions: (1) move to Region 1; (2) move to Region 2; or (3) do nothing. Suppose that L^1 is represented as

$$\begin{bmatrix} 4 & 0 & 4 \\ 1 & 3 & 3 \\ 3 & 1 & 5 \end{bmatrix}. \tag{5.30}$$

The columns of (5.30) represent the three possible actions available to $\mathcal{A}^1$, and rows represent the same possible choices for $\mathcal{A}^2$. Each entry represents the loss for the combination of actions, which takes into account both the distance traveled and the value of the information obtained. Region 2 represents a better vantage point, and at that location, a lower loss is received.

The primary difficulty with choosing the best action for $\mathcal{A}^1$ is that it has imperfect information about $\mathcal{A}^2$'s decision-making process. The shaded area in Figure 5.3 indicates an obstruction that may potentially block $\mathcal{A}^2$'s view of Region 2. Robot $\mathcal{A}^1$ does not know whether $\mathcal{A}^2$ knows about Region 2. Suppose that $\mathcal{A}^1$ believes that L^2 is

$$\begin{bmatrix} 5 & 1 & 5 \\ 0 & 2 & 2 \\ 3 & 1 & 5 \end{bmatrix} \tag{5.31}$$

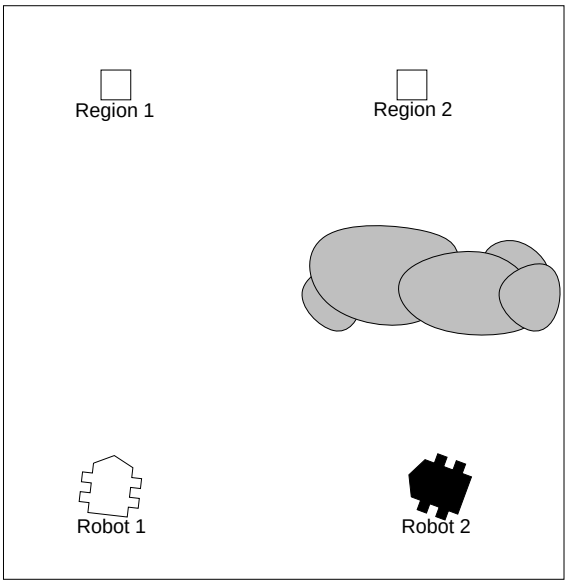


Figure 5.3 A two-robot terrain exploration problem.

with probability 0.1, which assumes that $\mathcal{A}^2$ knows about Region 2, and $\mathcal{A}^1$ believes that L^2 is

$$\begin{bmatrix} 5 & 5 \\ 3 & 5 \end{bmatrix} \tag{5.32}$$

with probability 0.9, in which the third row and column have been removed, indicating that $\mathcal{A}^2$ is not aware of that possible action.

Robot $\mathcal{A}^1$ could additionally speculate about how $\mathcal{A}^2$ models $\mathcal{A}^1$'s loss. The presented example is fairly simple, yet there can generally be an infinite recursion of modeling to fully handle this form of imperfect information. In [73], solutions were computed by assuming after some number of layers that the other decision maker has a mixed strategy. The implications are that many multiple-robot planning problems quickly become difficult when there are limitations in communication and information.

Part orienting This problem formulation is inspired by the approach considered in [74], [75], [76]. Suppose that planar parts appear on a conveyor belt at unknown orienta-

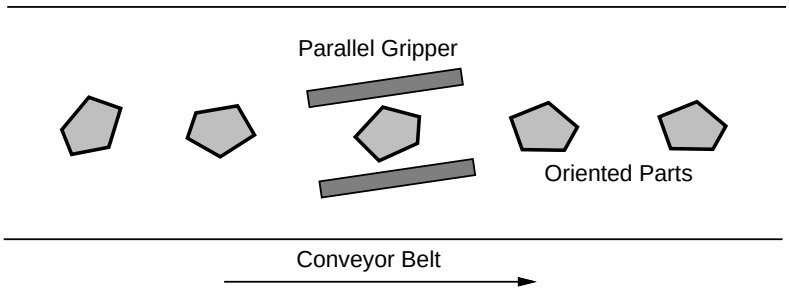


Figure 5.4 A parallel gripper that squeezes parts for alignment.

tions (see Figure 5.4). The task is to use a parallel gripper to perform squeeze operations that force the parts into a prescribed orientation. Let $X \subset S^1$, which corresponds to possible planar orientations of a known part. Let $U = S^1$, which corresponds to an angle at which a parallel robot gripper can squeeze. Let TC_k be a termination condition, as defined in Chapter 2. Suppose that there is probabilistic configuration-predictability uncertainty; hence, $x_{k+1} = f(x_k, u_k, \theta_k^a)$, and $P(\theta_k^a | x_k, u_k)$ is given. This is consistent with the modeling assumptions in [74], in which experimental methods were used to determine $P(\theta_k^a | x_k, u_k)$ (although it was not represented in this way).

Suppose that there are no sensor observations that can be utilized by the planner. Initially a uniform distribution is assumed on the space of possible orientations. An information state is represented as $\eta_k = \{u_1, \ldots, u_k\}$. The information space can also be considered as the set of all probability densities on X. Hence, an information state is characterized by $P(x_k | u_1, \ldots, u_k)$.

Designate a subset, G, of X as the goal region. The task is to place the part in an orientation that is in G. A loss functional can be defined as

$$L(u_1, ..., u_K) = \sum_{k=1}^{K+1} l(u_k, TC_k) + l_{K+1}(x_{K+1}), \tag{5.33}$$

in which l_{K+1} represents a cost of failure. Hence, $l_{K+1}(x_{K+1}) = 0$ if $x_{K+1} \in G$. Let $l(u_k, TC_k) = 0$ of $TC_k = true$, and $l(u_k, TC_k) = 1$, otherwise. The task can now be specified as determining the strategy that optimizes (5.33) in the expected sense.

The termination condition becomes important because of uncertainty in configuration sensing. If some of the probability mass from $P(x_k|u_1, \ldots, u_k)$ lies outside of G, then l_{K+1} influences the stage at which the termination condition is applied.

Several extensions to the basic formulation can also be readily made. Sensor measurements can be defined that estimate the distance between the sides of the gripper. One might also consider a problem in which there are multiple parts to be both sorted and grasped, which additionally involves planning under uncertainty in environment predictability.

5.5 Utilizing Formulations

This section defines some concepts for evaluating and comparing game formulations. These concepts capture a few intuitive notions about how a general formulation can be utilized to relate problems and methods. A game space will be defined in which each point corresponds to a completely represented problem. A subset of the game space can be considered as a generic formulation that can be specialized to a particular problem; Section 5.3 presented one such subset.

Let all combinations of actions of nature be represented as

$$\Theta = \Theta_1^a \times \cdots \times \Theta_K^a \times \Theta_1^{s,1} \times \cdots \times \Theta_K^{s,N} \times \Theta_1^{d,1} \times \cdots \times \Theta_K^{d,N}. \tag{5.34}$$

A particular game, g, can be defined as

$$\begin{aligned} g = \quad & \{N, K, X, U_1^1, \ldots, U_K^N, \Theta, f_1, \ldots, f_K, Y_1^1, \ldots, Y_K^N, h_1^1, \ldots, h_K^N, \\ & N_1^1, \ldots, N_K^N, \Gamma, L^1, \ldots, L^N, T\} \end{aligned}, \tag{5.35}$$

which corresponds to choices of functions and spaces that are defined in Section 5.2. Here Γ could represent a space of pure (deterministic) or mixed (randomized) strategies.

For the final component, $T = 0$ if nature is modeled nondeterministically, and $T = 1$ if nature is modeled probabilistically.

Let G denote a *game space*, which is generated by elements of the form of Equation (5.35). The generation of this space technically requires spaces to be defined for each component of (5.35). For instance, N can range over the set of positive integers. Each loss functional, L^i, generally belongs to some suitably chosen space of functionals. It will be assumed that spaces are defined for each component that are sufficiently general for the discussion. Technical considerations beyond this level of detail are not necessary for the treatment provided here.

Each of the formulations presented in Chapters 2 through 4 can be considered as subsets of G. Let G_c represent a set of single-robot problems that involve uncertainty in configuration predictability and configuration sensing, which were the subject of Chapter 2. One subset of G_c represents the case of perfect information; another subset represents nondeterministic uncertainty. The formulation in Section 5.3 represents a subset of G that contains the formulations from Chapters 2-4.

Imperfect information about the game itself could be considered by defining a space of probability densities on G, with appropriate measurability conditions are assumed. Each element in this space could, for example, represent the uncertainty that one decision maker has about another decision maker's models, which occurs in the exploration problem of Section 5.4.

Specializations Suppose a certain problem type has been encoded as a set of games, G_1, and analysis and computation methods are known for a set of games G_2 with $G_1 \subseteq G_2$. This implies that the same techniques apply to the problems in G_1; however, it might be the case that improvements could be made to the techniques due to the additional restrictiveness of G_1. For example, let G_2 represent the set of basic motion planning prob-

lems that can be described with algebraic constraints, and that have 0-1 loss functionals. Let G_1 represent the subset of G_2 that corresponds to problems in which $\mathcal{C}_{free}$ can be described with piecewise-linear constraints, and in which $\mathcal{C}_{free}$ is two-dimensional. The numerical dynamic programming method from Chapter 2 can in principle be applied to the problems in G_2. However, utilizing the additional information that the configuration space is two-dimensional with piecewise-linear constraints permits the visibility graph method (e.g., [125]) to quickly provide an exact solution.

Generalizations Suppose that analysis and reasonable computation methods are known for some set of games, G_1. One hope is that the same or similar methods apply for some larger set, G_1', such that $G_1 \subseteq G_1'$. One example of this type of generalization was the inclusion of nondeterministic uncertainty in Chapter 3; analysis and a computation method were presented for a particular game formulation that involved probabilistic representations of uncertainty. It can be expected that problems that alternatively use nondeterministic uncertainty can be attacked with a similar approach, which leads to larger set of games that can be solved, because the dependency on T is not critical.

Suppose that another set of games, G_2 has been analyzed and that computational methods are known. One natural question is whether the same concepts or a combination of the concepts can be applied to solve problems in some larger set G_3 for which $G_1 \cup G_2 \subseteq G_3$. This type of reasoning led to the game formulation and discussion in Section 5.3.

Similarities In many cases, it may be important to consider equivalence relations on G. Many points in the game space might lead to identical or nearly identical problems. Some situations are obvious: in any formulation that has multiple decision makers, all of the components for any pair of decision makers can be interchanged. This operation does little to change the problem, because it only causes the decision makers to be re-indexed.

More generally, it can be important to identify similarities between problems. Suppose, for example that three sets of games differ only by the state transition equation. One definition might include a class of nonholonomic motion models, another might include unconstrained motions, and a third might include models appropriate for independent control of manipulator joints. Each of these problems is very similar, because all other game components remain fixed. In this case, the computation approach may vary only slightly from one game set to another. This occurred for the problems in Chapter 3, in which results were obtained for these types of motion models, while the computational considerations were nearly unchanged. As another example, suppose that a set of games is partitioned into subsets by the dimension of the state space. We can expect that a computation method in one of these classes can most likely be applied to another. Computational expense, however, might drive the search for alternative approaches.

Conclusion The concepts provided in this chapter indicate a general approach for obtaining new insights and solution methods for various types of robotics problems. The unified mathematical structure assists in understanding the relationships between problems, which leads to a greater potential for progress. Relationships between certain robotics problems and game-theoretic and control-theoretic literature can also be utilized. Problems that consider state space constraints that are equivalent to those obtained in a motion planning problem are, however, uncommon in game-theoretic literature. (As an exception, [82] considers geometric constraints on the state space.) For this reason many formulations from the literature may have to be modified and adapted to apply to a particular robotics problem.

CHAPTER 6

CONCLUSIONS

This final chapter summarizes the contributions made in this dissertation and provides a concluding perspective.

6.1 Summary of Contributions

The general purpose throughout this dissertation has been to provide a unified and systematic way to analyze motion planning problems through the use of a dynamic game-theoretic mathematical structure. Many benefits are obtained by encoding types of motion planning problems in this structure. The concepts have provided a useful, compact characterization of several extended motion planning problems. For example, a general probabilistic characterization of motion planning under uncertainty in sensing and control had not been previously obtained; however, applying the general formulation allowed a clear objective to be defined: to select an expected optimal information-feedback strategy. Such formulations are useful, even in situations for which a practical solution cannot be directly obtained. For example, cylindrical algebraic decomposition techniques [4], [39] provided a means for solving the basic motion planning problem in a very general form, although the computational complexity made obtaining solutions to most problems

prohibitive. The tradeoffs offered by more efficient computational approaches can be analyzed by using the general solution as a basis for comparison, as done in [109]. In a similar manner, the information-space formulation of optimal strategies in Chapter 2 characterized solutions to the problem of motion planning under sensing and control uncertainties, and this led to a computational approach that approximates the information space.

The mathematical structure also offers flexibility when varying the modeling assumptions or generalizing concepts. The unifying nature of configuration space concepts has offered similar benefits, as rigid and articulated robots are analyzed with the same geometric concepts [109]. To illustrate the ease in varying modeling assumptions, Chapter 3 presented examples for a nonholonomic car-like robot, a constrained rotating robot, translating robots, and three-link manipulators, with little variation in the approach. The forward projection and performance preimage concepts provided an example of how the structure can be used to obtain generalizations; their traditional formulations were substantially generalized both in Chapters 2 and 5.

Another benefit is that similarities between different problems can easily be identified once the problems are embedded in the mathematical structure. This can provide useful insights for the development of computational approaches. For example, the principle of optimality holds true for a very general formulation, and formed the basis for all of the computation methods in this thesis. Similarities between the concepts in Chapters 2 and 3 led to similar computation methods.

Chapters 2 through 4 each provided modeling, analysis, algorithms, and computed examples for a distinct class of motion planning problems. The reason for these three separate investigations is twofold. First, each chapter independently made contributions to a class of problems that were motivated by previous research interests and limitations of existing approaches. Second, the work reported in these chapters supports the con-

clusion that a wide class of problems can be formulated and analyzed within the unified framework. The remainder of this section summarizes the independent contributions that were made in Chapters 2 through 4.

The treatment of uncertainties has been one of the primary concerns in advancing the state of the art of robot motion planning. In Section 1.2.2 four major sources of motion planning uncertainty were defined: (1) configuration sensing, (2) configuration predictability, (3) environment sensing, and (4) environment predictability. Chapter 2 addressed uncertainty in configuration sensing and configuration predictability. Chapter 3 addressed uncertainty in environment predictability, with some additional consideration of environment-sensing uncertainty. It was shown in Section 5.3 that all four sources of uncertainty can be treated in a single system using the concepts in this dissertation.

Uncertainty in sensing and control Chapter 2 introduced a characterization of motion planning under uncertainty in configuration sensing and predictability in which the goal is to select an information-feedback strategy that optimizes a loss functional. Nondeterministic representations could be used, which led to *worst-case* analysis, or probabilistic representations could be used, which led to *expected-case* analysis. The discussion and simulation experiments demonstrated that the efficiency of a robot motion strategy is crucial in planning under uncertainty, which emphasizes the importance of defining a loss functional. General forward projections and performance preimages were introduced to evaluate strategies. The *termination condition* concept was observed to be equivalent to the concept of *optimal stopping* from stochastic control theory. The information space concepts provided a useful and formal means to capture the relationship between history and the strategy of the robot. The specification of strategies in terms of information provided a novel formulation of optimality in the context of motion planning under un-

certainty. Finally, a computational approach was presented, and computed examples of forward projections, performance preimages, and optimal strategies were shown.

Environment uncertainties Chapter 3 introduced a characterization of motion planning under uncertainty in environment sensing and predictability. Many similarities can be drawn between this problem and the one in Chapter 2; therefore, many of the contributions are similar. Most of the chapter focused on determining optimal strategies under environment-predictability uncertainty with probabilistic representations, and extensions that include environment-sensing uncertainty and nondeterministic uncertainty were presented. One unique contribution of this research is the integration of motion planning with a separate process that models the changing environment. A general formulation of environment-predictability uncertainty had not previously existed in the motion planning literature. An extension to the basic concepts was presented in which the robot carries parts, and the optimal strategy minimized the expected time that parts wait to be delivered between partially-predictable sources and destinations. Several computed examples were presented for problems that involved environment uncertainties.

Multiple Robots Chapter 4 made two general contributions to motion planning for multiple robots: (1) motion plans were determined that simultaneously optimize an independent performance criterion for each robot by considering minimality with respect to the partial ordering of strategies; and (2) a general spectrum was defined between decoupled and centralized planning. Previous multiple-robot motion planning approaches that consider optimality combined several individual criteria into a single criterion. It was shown that these methods can fail to find many potentially useful motion plans. The minimal strategies characterize a small set of strategies that are better than or equivalent to all others. It was shown that this solution concept is equivalent to Nash equilibria,

Pareto optimality, and multiobjective optimality in this context; therefore, the reasonable strategies have been represented, regardless of some additional assumptions about cooperation and communication. A spectrum was defined between decoupled and centralized planning that amounted to varying the degree in which the individual configuration spaces of the robots were constrained. The optimality concepts could then be applied to both fixed-path coordination and centralized coordination. Coordination along independent roadmaps was introduced as a useful tradeoff between the completeness lost by constraining the state space and the computational cost of a coordination space with increased dimension. Algorithms were developed by applying the principle of optimality on a space of partial orderings. Computed examples were given for coordination on fixed paths, fixed roadmaps, and unconstrained planning.

6.2 Perspective

Many advances have been made in robotics research in the past few decades. As the field continues to mature, fundamental issues and problems are identified that are inherent in many robotic systems. This has led to a greater understanding of such fundamental issues as planning, control, sensing, manipulation, and grasping. Due to the complexity of engineering problems in the field of robotics, much effort is expended to improve a single aspect of the general problem. Motion planning represents one such aspect that has evolved greatly in the past decade or two. It is expected that such efforts will ultimately lead to a synergism of different ideas and issues, and to a coherent approach to robotics problems in general. One hope is that the material in this dissertation can assist in such a synthesis by providing a coherent and unifying mathematical characterization of the motion planning aspect.

APPENDIX A

PROOF OF PROPOSITION 3

The following proposition was stated in Section 4.3:

Proposition 3 *If* $l_k^i(x_k^i, u_k^i) = \Delta t$ *for all* $i \in \{1, \ldots, N\}$ *and* $k \in \{1, \ldots, K\}$, *then there exists at most one minimal quotient strategy per path class in* $\tilde{\mathcal{S}}_{valid}$.

Proof: First, consider the case in which $N = 2$. Let α denote the path that is obtained in the coordination space from a minimal strategy γ. Suppose to the contrary that there exists some $\alpha' \in [\alpha]_h$ (with strategy γ') such that $[\gamma]_L$ and $[\gamma']_L$ are distinct and minimal. The goal of this portion of the proof is to construct another path, $\alpha^* \in [\alpha]_h$ such that both $[\gamma^*]_L \preceq [\gamma]_L$ and $[\gamma^*]_L \preceq [\gamma']_L$, and $[\gamma^*]_L \neq [\gamma]_L$ and $[\gamma^*]_L \neq [\gamma']_L$. This will contradict the hypothesis, implying that the proposition holds for $N = 2$.

The images of α and α' in $\tilde{\mathcal{S}}_{valid}$ intersect in at least two places (including $(0, 0)$ and $(1, 1)$). Let V be the points of intersection in $\tilde{\mathcal{S}}_{valid}$. If the paths coincide from some stage k until stage $k' > k$, then we add only two intersection points to V, corresponding to when the paths initially coincide at stage k, and when the coincidence terminates at stage k'. This yields a finite set, $V = \{v_1, v_2, \ldots, v_m\}$ of intersection points. These points are ordered according to the occurrence of the intersection along the paths. Note that we always have $v_1 = (0, 0)$ and $v_m = (1, 1)$.

The path α^* that we will construct will intersect α and α' at every point in V. Let $\alpha_{i,j}$, for $i < j$, denote the portion of the path α that lies between v_i and v_j. For $1 \leq i < m-1$, compare the lengths in $\tilde{\mathcal{S}}_{valid}$ of $\alpha_{i,i+1}$ and $\alpha'_{i,i+1}$. A shorter path length will always cause the robots to reach v_{i+1} from v_i in less time. Because the passage of time produces the same loss for all robots, any strategy that reaches v_{i+1} from v_i in less time is better than or equal to a strategy that takes more time. For this reason, we let $\alpha^*_{i,i+1} = \alpha_{i,i+1}$ whenever $\alpha_{i,i+1}$ is shorter than $\alpha'_{i,i+1}$; otherwise, $\alpha^*_{i+1,i+1} = \alpha'_{i,i+1}$.

If we were to complete the construction of α^* by taking $\alpha^*_{m-1,m} = \alpha_{m-1,m}$ or $\alpha^*_{m-1,m} = \alpha'_{m-1,m}$, then the resulting strategy γ^* would be better than or equivalent to either γ or γ'. To contradict the hypothesis, however, we are required to construct a γ^* that is better than or equivalent to both γ and γ'.

For this final piece of α^*, consider Figure A.1. The lower left corner represents the intersection point v_{m-1}, and the upper right corner is the goal $v_m = (1, 1)$. There are two thick black lines that connect v_{m-1} to v_m and represent some $\alpha_{m-1,m}$ and $\alpha'_{m-1,m}$. We will determine the final piece of α^* without leaving the region formed by the two paths (hence the exterior is shaded in the figure). Both strategies are in the same path class; therefore, it is known that this region is free of collisions.

We will use the principle of minimality to construct the final path segment. The algorithm that was presented in Figure 4.6 produces the complete set of minimal strategies, and is sufficient to show that the algorithm produces only one minimal strategy at v_m. The path that corresponds to this minimal strategy will be designated as $\alpha^*_{m-1,m}$. The algorithm begins in the upper right corner and progresses from right to left, and top-down. Along the upper and rightmost boundaries, there are unique minimal strategies. These serve as initial conditions, and it will be argued inductively that each $M(\tilde{s})$ will contain only one element. At each iteration, there are at most three minimal strategies that can be constructed, which correspond to the three possible choices for u_k. If, from a

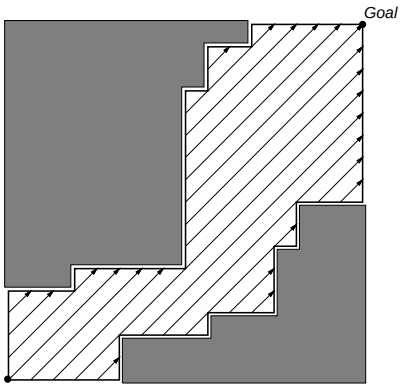


Figure A.1 See the proof of Proposition 3.

given state, the actions $u_k^1 = 1$ and $u_k^2 = 1$ do not produce a collision, then the resulting extended strategy will always be better than the other two choices. If these actions do produce a collision, then there is only one allowable action set (either $u_k^1 = 0$ and $u_k^2 = 1$, or $u_k^1 = 1$ and $u_k^2 = 0$) that does not produce a collision and, hence, there will be only one minimal strategy. If there were two possible action sets, then, due to the monotonicity of α^*, the two choices would lead to two different path classes, which contradicts the initial hypothesis. At the final iteration, $M(v_{m-1})$ will contain only one minimal strategy. The path corresponding to the minimal strategy is used to complete α^*, resulting in the contradicting strategy.

We now consider the case in which $N > 2$. Suppose again that there exists some $\alpha' \in [\alpha]_h$, such that $[\gamma]_L$ and $[\gamma']_L$ are distinct and minimal. This implies that for some pair of indices i, j, we have $L_i(\gamma) < L_i(\gamma')$ and $L_j(\gamma) > L_j(\gamma')$. Consider $\tilde{\mathcal{S}}_{valid}^{ij}$ as the coordination space generated by considering only τ^i and τ^j. The path α^{ij} on $\tilde{\mathcal{S}}_{valid}^{ij}$ that corresponds to the implementation of $\{\gamma^i, \gamma^j\}$ is obtained by the projection of the path α down to $\tilde{\mathcal{S}}_{valid}^{ij}$. This is true, because f^i, as given in (4.5), depends only on the configuration and control of $\mathcal{A}^i$. The same is true for the path $\alpha^{ij'}$ under the implementation of $\{\gamma^{i'}, \gamma^{j'}\}$. Hence, robots other than $\mathcal{A}^i$ and $\mathcal{A}^j$ do not interfere with the projected path in $\tilde{\mathcal{S}}_{valid}^{ij}$.

From the previous part of the proof (for $N = 2$), it follows that projected paths α^{ij} and $\alpha^{ij\prime}$ are in distinct path classes in $\tilde{\mathcal{S}}^{ij}_{valid}$. We consider lifting this projected space $\tilde{\mathcal{S}}^{ij}_{valid}$ back up to $\tilde{\mathcal{S}}_{valid}$. We note that $\tilde{\mathcal{S}}^{ij}_{valid}$ (two-dimensional) and $\mathcal{S} - \tilde{\mathcal{S}}^{ij}_{coll}$ (N-dimensional) are homeomorphic, due to the cylindrical property of $\tilde{\mathcal{S}}^{ij}_{coll}$. Because homeomorphic spaces are homotopically equivalent [87], the paths α and α' are in distinct path classes in $\mathcal{S} - \tilde{\mathcal{S}}^{ij}_{coll}$. Because $\tilde{\mathcal{S}}_{valid} \subset \mathcal{S} - \tilde{\mathcal{S}}^{ij}_{coll}$, and the image of the paths α and α' lie in $\tilde{\mathcal{S}}_{valid}$, they consequently belong to distinct path classes in $\tilde{\mathcal{S}}_{valid}$. $\square$

REFERENCES

[1] R. Alami, T. Siméon, and J. P. Laumond. A geometrical approach to planning manipulation tasks. In *5th Int. Symp. Robot. Res.*, pages 113–119, 1989.

[2] B. D. Anderson and J. B. Moore. *Optimal Control: Linear-Quadratic Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[3] M. D. Ardema and J. M. Skowronski. Dynamic game applied to coordination control of two arm robotic system. In R. P. Hämäläinen and H. K. Ehtamo, editors, *Differential Games - Developments in Modelling and Computation*, pages 118–130. Springer-Verlag, Berlin, 1991.

[4] D. S. Arnon. Geometric reasoning with logic and algebra. *Artif. Intell.*, 37(1-3):37–60, 1988.

[5] T. Başar and P. R. Kumar. On worst case design strategies. *Comput. Math. Applic.*, 13(1-3):239–245, 1987.

[6] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[7] M. Barbehenn, P. C. Chen, and S. Hutchinson. An efficient hybrid planner in changing environments. In *IEEE Int. Conf. Robot. & Autom.*, pages 2755–2760, 1994.

[8] M. Barbehenn and S. Hutchinson. Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees. *IEEE Trans. Robot. & Autom.*, 11(2):198–214, April 1995.

[9] J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 1235–1242, 1994.

[10] J. Barraquand and P. Ferbach. Motion planning with uncertainty: The information space approach. In *IEEE Int. Conf. Robot. & Autom.*, pages 1341–1348, 1995.

[11] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst., Man, Cybern.*, 22(2):224–241, 1992.

[12] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *IEEE Int. Conf. Robot. & Autom.*, pages 1712–1717, 1990.

[13] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 2328–2335, 1991.

[14] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1991.

[15] K. Basye, T. Dean, J. Kirman, and M. Lejter. A decision-theoretic approach to planning, perception, and control. *IEEE Expert*, 7(4):58–65, August 1992.

[16] R. E. Bellman and S. E. Dreyfus. *Applied Dynamic Programming.* Princeton University Press, Princeton, NJ, 1962.

[17] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[18] W. F. Bialas. Cooperative n-person Stackelberg games. In *IEEE Conf. Decision & Control*, pages 2439–2444, Tampa, FL, December 1989.

[19] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Trans. Robot. & Autom.*, 8(3):414–418, June 1992.

[20] D. Blackwell and M. A. Girshik. *Theory of Games and Statistical Decisions.* Dover Publications, New York, NY, 1979.

[21] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robot. Res.*, 4(3):3–17, 1985.

[22] A. Briggs. An efficient algorithm for one-step compliant motion planning with uncertainty. In *5th ACM Symp. Comp. Geom.*, 1989.

[23] R. A. Brooks. Solving the find-path problem by good representation of free space. *IEEE Trans. Syst., Man, Cybern.*, 13(3):190–197, 1983.

[24] R. C. Brost. Automatic grasp planning in the presence of uncertainty. *Int. J. Robot. Res.*, 7(1):3–17, 1988.

[25] R. C. Brost. *Analysis and Planning of Planar Manipulation Tasks.* Ph.D. thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.

[26] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A research agenda. In *IEEE Int. Conf. Robot. & Autom.*, volume 3, pages 549–556, 1993.

[27] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A computational framework. Technical Report SAND92-2033, Sandia National Laboratories, Albuquerque, NM, January 1994.

[28] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York, NY, 1975.

[29] S. J. Buckley. Fast motion planning for multiple moving robots. In *IEEE Int. Conf. Robot. & Autom.*, pages 322–326, 1989.

[30] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. IEEE Conf. on Foundations of Computer Science*, pages 49–60, 1987.

[31] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[32] J. F. Canny. On computability of fine motion plans. In *IEEE Int. Conf. Robot. & Autom.*, pages 177–182, 1989.

[33] C. Chang, M. J. Chung, and B. H. Lee. Collision avoidance of two robot manipulators by minimum delay time. *IEEE Trans. Syst., Man, Cybern.*, 24(3):517–522, 1994.

[34] S.-C. Chang and D.-T. Liao. Scheduling flexible flow shops with no setup effects. *IEEE Trans. Robot. & Autom.*, 10(2):99–111, 1994.

[35] F. L. Chernousko, N. N. Bolotnik, and V. G. Gradetsky. *Manipulation Robots.* CRC Press, Ann Arbor, MI, 1994.

[36] H. Choset and J. Burdick. Sensor based planning, part I: The generalized Voronoi graph. In *IEEE Int. Conf. Robot. & Autom.*, pages 1649–1655, 1995.

[37] K.-C. Chu. Team decision theory and information structures in optimal control problems-part II. *IEEE Trans. Autom. Control*, 17(1):22–28, February 1972.

[38] C. K. Chui and G. Chen. *Kalman Filtering.* Springer-Verlag, Berlin, 1991.

[39] G. E. Collins. *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1975.

[40] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *An Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[41] M. R. Cutkosky. *Robotic Grasping and Fine Manipulation.* Kluwer Academic Publishers, Boston, MA, 1985.

[42] J. Gil de Lamadrid and J. Zimmerman. Avoidance of obstacles with unknown trajectories: locally optimal paths and path complexity, part I. *Robotica*, 11:299–308, 1993.

[43] T. Dean, K. Kanazawa, and J. Shewchuk. Prediction, observation, and estimation in planning and control. In *IEEE Symp. on Intelligent Control*, pages 645–650, 1990.

[44] T. L. Dean and M. P. Wellman. *Planning and Control.* Morgan Kaufman, San Mateo, CA, 1991.

[45] M. H. DeGroot. *Optimal Statistical Decisions.* McGraw Hill, New York, NY, 1970.

[46] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall Publications, Englewood Cliffs, NJ, 1982.

[47] B. R. Donald. *Error Detection and Recovery for Robot Motion Planning with Uncertainty.* Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1987.

[48] B. R. Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artif. Intell.*, 37:223–271, 1988.

[49] B. R. Donald. Planning multi-step error detection and recovery strategies. *Int. J. Robot. Res.*, 9(1):3–60, 1990.

[50] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *IEEE Int. Conf. Robot. & Autom.*, pages 190–197, Sacramento, CA, April 1991.

[51] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, NY, 1973.

[52] H. F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Planning Systems.* Kluwer Academic Publishers, Boston, MA, 1988.

[53] H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Trans. Robot. & Autom.*, 4(1):23–31, February 1988.

[54] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Berlin, 1987.

[55] D. Einav and M. R. Fehling. Computationally-optimal real-resource strategies for independent, uninterruptible methods. In *Uncertainty in Artificial Intelligence 6*, pages 145–158. North-Holland, Amsterdam, 1991.

[56] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, June 1989.

[57] M. Erdmann. Randomization in robot tasks. *Int. J. Robot. Res.*, 11(5):399–436, October 1992.

[58] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[59] M. Erdmann. On a representation of friction in configuration space. *Int. J. Robot. Res.*, 13(3):240–271, 1994.

[60] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Int. Conf. Robot. & Autom.*, pages 1419–1424, 1986.

[61] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[62] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1984.

[63] M. A. Erdmann. *On Probabilistic Strategies for Robot Tasks.* Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[64] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. Robot. & Autom.*, 8(3):313–326, June 1992.

[65] J. T. Feddema and O. R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. Robot. & Autom.*, 5(5):691–700, October 1989.

[66] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans. *IEEE Trans. Robot. & Autom.*, 1(11):56–71, February 1995.

[67] Th. Fraichard and A. Scheuer. Car-like robots and moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 64–69, 1994.

[68] K. Fujimura. On motion planning amidst transient obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 1488–1493, 1992.

[69] K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63, 1993.

[70] S. Geman. Experiments in Bayesian image analysis. *Bayesian Statistics*, 3:159–172, 1988.

[71] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 11–19, 1987.

[72] E. G. Gilbert and D. W. Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Trans. Robot. & Autom.*, 1(1):21–30, March 1985.

[73] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision-theoretic approach to coordinating multi-agent interations. In *Proc. Int. Joint Conf. on Artif. Intell.*, pages 62–68, 1991.

[74] K. Y. Goldberg. *Stochastic Plans for Robotic Manipulation.* Ph.D. thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1990.

[75] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[76] K. Y. Goldberg and M. T. Mason. Bayesian grasping. In *IEEE Int. Conf. Robot. & Autom.*, 1990.

[77] S. Gottschlich, C. Ramos, and D. Lyons. Assembly and task planning: A taxonomy. *IEEE Robotics and Automation Magazine*, 1(3):4–12, 1994.

[78] S. N. Gottschlich and A. C. Kak. AMP-CAD: Automatic assembly motion planning using CAD models of parts. *Robotics and Autonomous Systems*, 13:245–289, 1994.

[79] G. Hager and H. G. Durrant-Whyte. Information and multi-sensor coordination. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 381–393. Elsevier, New York, NY, 1988.

[80] G. D. Hager. *Task-Directed Sensor Fusion and Planning.* Kluwer Academic Publishers, Boston, MA, 1990.

[81] O. Hájek. *Pursuit Games.* Academic Press, New York, 1975.

[82] K. Haji-Ghassemi. On differential games of fixed duration with phase coordinate restrictions on one player. *SIAM J. Control & Optimization*, 28(3):624–652, May 1990.

[83] J. C. Harsanyi. Games with incomplete information played by Bayesian players. *Management Science*, 14(3):159–182, November 1967.

[84] J. Hervé, P. Cucka, and R. Sharma. Qualitative visual control of a robot manipulator. In *Proc. of the DARPA Workshop on Image Understanding*, pages 895–908, 1991.

[85] K. W. Hipel, K. J. Radford, and L. Fang. Multiple participant-multiple criteria decision making. *IEEE Trans. Syst., Man, Cybern.*, 23(4):1184–1189, 1993.

[86] Y.-C. Ho and K.-C. Chu. Team decision theory and information structures in optimal control problems-part I. In *IEEE Trans. Autom. Control*, pages 15–22, 1972.

[87] J. G. Hocking and G. S. Young. *Topology.* Dover Publications, New York, NY, 1988.

[88] H. Hu and M. Brady. A Bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots*, 1(1):69–92, 1994.

[89] H. Hu, M. Brady, and P. Probert. Coping with uncertainty in control and planning for a mobile robot. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 1025–1030, Osaka, Japan, November 1991.

[90] Y.-R. Hu and A. A. Goldenberg. Dynamic control of multiple coordinated redundant robots. *IEEE Trans. Syst., Man, Cybern.*, 22(3):568–574, May/June 1992.

[91] Y. K. Hwang and N. Ahuja. Gross motion planning–A survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[92] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Trans. Robot. & Autom.*, 8(1):23–32, February 1992.

[93] K. Ikeuchi and T. Kanade. Modeling sensors: Toward automatic generation of object recognition program. *Comp. Vision, Graphics, and Image Process.*, 48:50–79, 1989.

[94] R. Isaacs. *Differential Games.* Wiley, New York, NY, 1965.

[95] M. E. Kahn and B. E. Roth. The near minimum-time control of open-loop articulated kinematic chains. *Trans. ASME J. Dyn. Sys., Meas., & Contr.*, 93(3):164–172, 1971.

[96] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Robot. Res.*, 5(3):72–89, 1986.

[97] L. E. Kavraki. *Random Networks in Configuration Space for Fast Path Planning.* Ph.D. thesis, Department of Computer Science, Stanford University, 1994.

[98] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.

[99] K. H. Kim and F. W. Roush. *Team Theory.* Ellis Horwood Limited, Chichester, England, 1987.

[100] J. Kirman, K. Basye, and T. Dean. Sensor abstraction for control of navigation. In *IEEE Int. Conf. Robot. & Autom.*, pages 2812–2817, 1991.

[101] D. Koditschek. Robot planning and control via potential functions. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, Cambridge, MA, 1989.

[102] V. S. Kouikoglou and Y. A. Phillis. Discrete event modeling and optimization of unreliable production lines with random rates. *IEEE Trans. Robot. & Autom.*, 10(2):153–159, 1994.

[103] V. G. Kountouris and H. E. Stephanou. Dynamic modularization and synchronization for intelligent robot coordination: The concept of functional time-dependency. In *IEEE Int. Conf. Robot. & Autom.*, pages 508–513, Sacramento, CA, April 1991.

[104] N. V. Krylov. *Controlled diffusion processes.* Springer-Verlag, Berlin, 1980.

[105] P. R. Kumar and P. Varaiya. *Stochastic Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1986.

[106] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems.* Wiley, New York, NY, 1972.

[107] R. E. Larson. *State Increment Dynamic Prgramming.* Elsevier, New York, NY, 1968.

[108] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II.* Dekker, New York, NY, 1982.

[109] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[110] J.-C. Latombe, A. Lazanas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artif. Intell.*, 52:1–47, 1991.

[111] J.-P. Laumond. Singularities and topological aspects in nonholonomic motion planning. In Z. Li and J. F. Canny, editors, *Nonholonomic Motion Planning*, pages 149–200. Kluwer Academic Publishers, Boston, MA, 1993.

[112] S. M. LaValle and S. A. Hutchinson. Game theory as a unifying structure for a variety of robot tasks. In *Proc. IEEE Int'l Symp. on Intelligent Control*, pages 429–434, August 1993.

[113] S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems*, pages 1772–1779, September 1994.

[114] S. M. LaValle and S. A. Hutchinson. Path selection and coordination of multiple robots via Nash equilibria. In *Proc. 1994 IEEE Int'l Conf. Robot. & and Autom.*, pages 1847–1852, May 1994.

[115] S. M. LaValle and S. A. Hutchinson. A framework for constructing probability distributions on the space of segmentations. *Computer Vision and Image Understanding*, 61(2):203–230, March 1995.

[116] S. M. LaValle and R. Sharma. Motion planning in stochastic environments: Applications and computational issues. In *IEEE Int'l Conf. on Robotics and Automation*, pages 3063–3068, 1995.

[117] S. M. LaValle and R. Sharma. Motion planning in stochastic environments: Theory and modeling issues. In *IEEE Int'l Conf. on Robotics and Automation*, pages 3057–3062, 1995.

[118] Z. Li and J. F. Canny. *Nonholonomic Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1993.

[119] C.-F. Lin and W.-H. Tsai. Motion planning for multiple robots with multi-mode operations via disjunctive graphs. *Robotica*, 9:393–408, 1990.

[120] Z. Lin, V. Zeman, and R. V. Patel. On-line robot trajectory planning for catching a moving object. In *IEEE Int. Conf. Robot. & Autom.*, pages 1726–1731, 1989.

[121] J.-S. Liu, L.-S. Wang, and L.-S. Tsai. A nonlinear programming approach to nonholonomic motion planning with obstacle avoidance. In *IEEE Int. Conf. Robot. & Autom.*, pages 70–75, 1994.

[122] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Trans. Syst., Man, Cybern.*, 11(10):681–698, 1981.

[123] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. on Comput.*, C-32(2):108–120, 1983.

[124] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic systhesis of fine-motion strategies for robots. *Int. J. Robot. Res.*, 3(1):3–24, 1984.

[125] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[126] V. J. Lumelsky and T. Skewis. A paradigm for incorporating vision in the robot navigation function. In *IEEE Int. Conf. Robot. & Autom.*, pages 734–739, 1988.

[127] V. J. Lumelsky and A. A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[128] K. M. Lynch and M. T. Mason. Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. *Int. J. Robot. Res.*, 14(2):174–183, 1995.

[129] N. Mahadevamurty, T.-C. Tsao, and S. Hutchinson. Multi-rate analysis and design of visual feedback digital servo control systems. *Trans. ASME J. Dyn. Sys., Meas., & Contr.*, pages 45–55, March 1994.

[130] M. T. Mason. Compliance and force control for computer controlled manipulators. In B. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion: Planning and Control*, pages 373–404. MIT Press, Cambridge, MA, 1982.

[131] M. T. Mason. Automatic planning of fine motions: Correctness and completeness. In *Proc. IEEE Int. Conf. Robot. & Autom.*, pages 492–503, 1984.

[132] M. T. Mason. Mechanics and planning of manipulator pushing operations. *Int. J. Robot. Res.*, 5(3):53–71, 1986.

[133] S. McMillan, P. Sadayappan, and D. E. Orin. Parallel dynamic simulation of multiple manipulator systems: Temporal versus spatial methods. *IEEE Trans. Syst., Man, Cybern.*, 24(7):982–990, July 1994.

[134] J. S. B. Mitchell. *Planning Shortest Paths.* Ph.D. thesis, Department of Operations Research, Stanford University, 1986.

[135] J. Miura and Y. Shirai. Planning of vision and motion for a mobile robot using a probabilistic model of uncertainty. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 403–408, Osaka, Japan, May 1991.

[136] B. K. Natarajan. The complexity of fine motion planning. *Int. J. Robot. Res.*, 7(2):36–42, 1988.

[137] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *IEEE Int. Conf. Robot. & Autom.*, pages 484–489, 1989.

[138] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[139] B. O'Neill. *Elementary Differential Geometry.* Academic Press, New York, NY, 1966.

[140] J. B. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case. *IEEE J. of Robot. & Autom.*, 3(6):672–681, 1987.

[141] G. Owen. *Game Theory.* Academic Press, New York, NY, 1982.

[142] L. A. Page and A. C. Sanderson. Robot motion planning for sensor-based control with uncertainties. In *IEEE Int. Conf. Robot. & Autom.*, pages 1333–1340, 1995.

[143] S. Pandya and S. A. Hutchinson. A case-based approach to robot motion planning. In *Proc. IEEE Int. Conf. on Syst., Man, & Cybern.*, pages 492–497, Chicago, October 1992.

[144] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[145] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Math. of Oper. Res.*, 12(3):441–450, August 1987.

[146] N. P. Papanikolopoulos and P. K. Khosla. Shared and traded telerobotic visual control. In *IEEE Int. Conf. Robot. & Autom.*, pages 878–883, 1992.

[147] T. Parthasarathy and M. Stern. Markov games: A survey. In *Differential Games and Control Theory II*, pages 1–46. Marcel Dekker, New York, 1977.

[148] R. P. Paul and B. Shimano. Compliance and control. In *Proc. of the Joint American Automatic Control Conference*, pages 1694–1699, 1976.

[149] J. Pearl. *Heuristics.* Addison-Wesley, Reading, MA, 1984.

[150] J. Pertin-Troccaz. Grasping: A state of the art. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, Cambridge, MA, 1989.

[151] L. A. Petrosjan. *Differential Games of Pursuit.* Singapore, River Edge, NJ, 1993.

[152] S. Premvuti and S. Yuta. Consideration on the cooperation of multiple autonomous mobile robots. In *IEEE Int. Workshop on Intelligent Robots and Systems*, pages 59–63, 1990.

[153] F. P. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag, Berlin, 1985.

[154] M. H. Raibert and J. J. Craig. Hybrid position/force control of manipulators. *Trans. ASME J. Dyn. Sys., Meas., & Contr.*, 102, 1981.

[155] N. S. V. Rao, S. S. Iyengar, J. B. Oommen, and R. L. Kashyap. On terrain model acquisition by a point robot amidst polyhedral obstacles. *IEEE J. of Robot. & Autom.*, 4:450–455, 1988.

[156] S. Ratering and M. Gini. Robot navigation in a known environment with unknown moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 25–30, 1993.

[157] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. of IEEE Symp. on Foundat. of Comp. Sci.*, pages 144–154, 1985.

[158] E. Rimon and D. E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *IEEE Int. Conf. Robot. & Autom.*, pages 21–26, May 1989.

[159] E. Rimon and D. E. Koditschek. Exact robot navigation in geometrically complicated but topologically simply spaces. In *IEEE Int. Conf. Robot. & Autom.*, pages 1937–1942, May 1990.

[160] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. & Autom.*, 8(5):501–518, October 1992.

[161] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23:71–76, 1986.

[162] J. J. Rotman. *Introduction to Algebraic Topology*. Springer-Verlag, Berlin, 1988.

[163] N. C. Rowe and R. F. Richbourg. A new method for optimal path planning through nonhomogeneous free space. Technical Report NPS52-87-003, Naval Postgraduate School, 1987.

[164] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization*. Academic Press, New York, NY, 1985.

[165] G. K. Schmidt and K. Azarm. Mobile robot navigation in a dynamic world using an unsteady diffusion equation strategy. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 642–647, 1992.

[166] J. T. Schwartz and M. Sharir. On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[167] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies. *Int. J. Robot. Res.*, 2(3):97–140, 1983.

[168] U. Shaked and C. E. de Souza. Continuous-time tracking problems in an h_∞ setting: A game theory approach. *IEEE Trans. Autom. Control*, 40(5):841–852, May 1995.

[169] Y. Shan and Y. Koren. Obstacle accommodation motion planning. *IEEE Trans. Robot. & Autom.*, 1(11):36–49, February 1995.

[170] R. Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *IEEE Trans. Robot. & Autom.*, 8(1):105–110, February 1992.

[171] R. Sharma. A probabilistic framework for dynamic motion planning in partially known environments. In *IEEE Int. Conf. Robot. & Autom.*, pages 2459–2464, Nice, France, May 1992.

[172] R. Sharma, S. M. LaValle, and S. A. Hutchinson. Optimizing robot motion strategies for assembly with stochastic models of the assembly process. In *IEEE Int'l Symp. on Assembly and Task Planning*, 1995.

[173] R. Sharma, D. M. Mount, and Y. Aloimonos. Probabilistic analysis of some navigation strategies in a dynamic environment. *IEEE Trans. Syst., Man, Cybern.*, 23(5):1465–1474, September 1993.

[174] T. Shibata and T. Fukuda. Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system. In *IEEE Int. Conf. Robot. & Autom.*, pages 1:760–765, 1993.

[175] C. L. Shih, T.-T. Lee, and W. A. Gruver. A unified approach for robot motion planning with moving polyhedral obstacles. *IEEE Trans. Syst., Man, Cybern.*, 20:903–915, 1990.

[176] K. G. Shin and N. D. McKay. Minimum-time control of robot manipulators with geometric path constraints. *IEEE Trans. Autom. Control*, 30(6):531–541, 1985.

[177] K. G. Shin and Q. Zheng. Minimum-time collision-free trajectory planning for dual-robot systems. *IEEE Trans. Robot. & Autom.*, 8(5):641–644, October 1992.

[178] J. F. Silverman and D. B. Cooper. Bayesian clustering for unsupervised estimation of surface and texture models. *IEEE Trans. Pattern Anal. Machine Intell.*, 10(4):482–496, July 1988.

[179] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *IEEE Int. Conf. Robot. & Autom.*, pages 254–258, 1993.

[180] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Int. J. Robot. Res.*, 5(4):56–68, 1986.

[181] R. Spence and S. A. Hutchinson. Dealing with unexpected moving obstacles by integrating potential field planning with inverse dynamics control. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 1485–1490, 1992.

[182] R. Spence and S. A. Hutchinson. An integrated architecture for robot motion planning and control in the presence of obstacles with unknown trajectories. *IEEE Trans. Syst., Man, Cybern.*, 25(1):100–110, 1995.

[183] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* Wiley, New York, NY, 1989.

[184] W. Stadler. Fundamentals of multicriteria optimization. In W. Stadler, editor, *Multicriteria Optimization in Engineering and in the Sciences*, pages 1–25. Plenum Press, New York, NY, 1988.

[185] A. Stentz. Optimal and efficient path planning for partially-known environments. In *IEEE Int. Conf. Robot. & Autom.*, pages 3310–3317, 1994.

[186] S.-H. Suh and K. G. Shin. A variational dynamic programming approach to robot-path planning with a distance-safety criterion. *IEEE Trans. Robot. & Autom.*, 4(3):334–349, June 1988.

[187] K. Sutner and W. Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26:93–122, 1988.

[188] H. Takeda, C. Facchinetti, and J.-C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(10):1002–1017, October 1994.

[189] H. Takeda and J.-C. Latombe. Sensory uncertainty field for mobile robot navigation. In *IEEE Int. Conf. Robot. & Autom.*, pages 2465–2472, Nice, France, May 1992.

[190] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In *Fourth International Symposium on Robotics Research*, pages 421–429, 1987.

[191] J. C. Trinkle and D. C. Zeng. Prediction of the quasistatic planar motion of a contacted rigid body. *IEEE Trans. Robot. & Autom.*, 11(2):229–246, April 1995.

[192] C. van Delft. Approximate solutions for large-scale piecewise deterministic control systems arising in manufacturing flow control models. *IEEE Trans. Robot. & Autom.*, 10(2):142–152, 1994.

[193] F.-Y. Wang and P. J. A. Lever. A cell mapping method for general optimum trajectory planning of multiple robotic arms. *Robots and Autonomous Systems*, 12:15–27, 1994.

[194] C. W. Warren. Multiple robot path coordination using artificial potential fields. In *IEEE Int. Conf. Robot. & Autom.*, pages 500–505, 1990.

[195] D. Whitney. Force feedback control of manipulator fine motions. *Trans. ASME J. of Dyn. Sys., Meas., & Contr.*, 99:91–97, 1977.

[196] D. E. Whitney. Historical perspectives and the state of the art in robot force control. In *IEEE Int. Conf. Robot. & Autom.*, pages 262–268, 1985.

[197] P. Williams. *Probability with Martingales.* Cambridge Press, New York, NY, 1991.

[198] P. H. Winston. *Artificial Intelligence.* Addison-Wesley, Reading, MA, 1992.

[199] Y. Yavin and M. Pachter. *Pursuit-Evasion Differential Games.* Pergamon Press, Oxford, England, 1987.

[200] J. Yong. On differential evasion games. *SIAM J. Control & Optimization*, 26(1):1–22, January 1988.

[201] J. Yong. On differential pursuit games. *SIAM J. Control & Optimization*, 26(2):478–495, March 1988.

[202] L. S. Zaremba. Differential games reducible to optimal control problems. In *IEEE Conf. Decision & Control*, pages 2449–2450, Tampa, FL, December 1989.

[203] Y. F. Zheng, J. Y. S. Luh, and P. F. Jia. Integrating two industrial robots into a coordinated system. *Computers in Industry*, 12:285–298, 1989.

[204] Q. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Trans. Robot. & Autom.*, 7(3):390–397, June 1991.

[205] S. Zionts. Multiple criteria mathematical programming: An overview and several approaches. In P. Serafini, editor, *Mathematics of Multi-Objective Optimization*, pages 227–273. Springer-Verlag, Berlin, 1985.

VITA

Steven M. LaValle was born in St. Louis, Missouri, on June 14, 1968. He graduated from John F. Kennedy High School in Manchester, Missouri, in 1986. He received a B.S. degree with highest honors in Computer Engineering in 1990, an M.S. degree in Electrical Engineering in 1993, and a PhD degree in Electrical Engineering in 1995, all from the University of Illinois at Urbana-Champaign.

From 1987 to 1989 LaValle participated in the James Scholar program in the College of Engineering, University of Illinois. In 1989, he was designated an Amoco Foundation Scholar. He was nominated several times for the Oleson Award for Outstanding Teaching. He was listed seven times in the *Daily Illini's* List of Teachers Ranked Excellent by Their Students between 1988 and 1991. He received a Distinguished Student Paper Award at the Ninth Conference on Uncertainty in Artificial Intelligence in 1993. LaValle received a Beckman Institute Research Assistantship award for 1994-1995. In 1994 he was awarded the Mavis Fellowship from the College of Engineering, University of Illinois. In July 1995, his doctoral dissertation was selected to represent the University of Illinois in the national ACM Outstanding Dissertation Award competition. He is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, and the American Association for Artificial Intelligence.