Robot Motion Planning: A Game-Theoretic Foundation

Steven M. LaValle
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
lavalle@cs.iastate.edu

Abstract

Analysis techniques and algorithms for basic path planning have become quite valuable in a variety of applications such as robotics, virtual prototyping, computer graphics, and computational biology. Yet, basic path planning represents a very restricted version of general motion planning problems often encountered in robotics. Many problems can involve complications such as sensing and model uncertainties, nonholonomy, dynamics, multiple robots and goals, optimality criteria, unpredictability, and nonstationarity, in addition to standard geometric workspace constraints. This paper proposes a unified, game-theoretic mathematical foundation upon which analysis and algorithms can be developed for this broader class of problems, and is inspired by the similar benefits that were obtained by using unified configuration-space concepts for basic path planning. By taking this approach, a general algorithm has been obtained for computing approximate optimal solutions to a broad class of motion planning problems, including those involving uncertainty in sensing and control, environment uncertainties, and the coordination of multiple robots.

1 Introduction

Two decades of development has led to many successful analysis techniques and algorithms for planning collision-free paths of rigid or articulated robots in a known, cluttered environment. These algorithms have enjoyed great success across many different kinds of robotic platforms in both research and industrial applications, and in applications beyond robotics, such as virtual prototyping, graphical animation, architecture, and computational biology. While most of us are aware that basic path planning represents a very restricted version of motion strategy problems often encountered in robotics, existing algorithms that handle more difficult problems such as unpredictability, sensing uncertainty, nonholonomy, and dynamics seem to fall short of the same widespread applicability and use currently enjoyed by some basic path planning techniques. We propose the development of motion strategy algorithms that are built on a game-theoretic framework [72, 78, 81], which will hopefully lead to some of the same benefits enjoyed by current path planning algorithms, but instead apply to a broader class of problems that are of interest to the robotics community.

To achieve success in the development of these algorithms, we believe that it is paramount to follow some of the same principles which led to the success of the path planning paradigm:

- A well-formulated problem was isolated and abstracted away from a broad class of robotics tasks (e.g., the Piano Mover's problem).

- A powerful mathematical framework was developed that enabled a unified development of algorithms (e.g., configuration space).

- Algorithms were introduced that were general and adaptable (e.g., probabilistic roadmaps [1, 61]).

Once the path planning problem was identified as a fundamental operation that would need to be solved if robots were expected to achieve many common tasks, it received considerable research attention from the robotics and algorithms communities. Configuration space concepts served as a powerful representational tool for both the development and analysis of path planning algorithms [68, 84]. For example, the comparison of seemingly disparate approaches, such as cell decomposition methods, roadmap methods, and artificial potential field methods, was greatly facilitated [68], which increased our ability to measure progress in this area. Once the configuration space framework was in place, it became possible to develop algorithms that were generalizable to adaptable to a wide variety of applications. For example, the randomized potential field principles have been applied to a wide variety of problems that appear quite different on the surface, but all reduce to finding a continuous path that traverses the collision-free portion of the configuration space.

We now make a distinction between the basic *path planning problem* and the general *motion strategy problem* (see Figure 1). The path planning problem (or Piano Mover's problem) involves the following components: 1) a two or three dimensional Euclidean space, $\mathcal{W}$, is defined as the workspace or world in which a robot lives; 2) a rigid or multibody robot that is completely described using polygonal, polyhedral, or algebraic models; 3) a configuration space, $\mathcal{C}$, is a manifold in which each element represents a transformation that completely specifies the position and orientation of all parts of the robot (for example, $\mathcal{C} = \Re^2 \times S^1$) represents the set of all ways in which a 2D robot can be translated or rotated in the plane); 4) static obstacles are defined as subsets of $\mathcal{W}$, and are completely completely characterized using polygonal, polyhedral, or algebraic models; 5) an initial configuration, $q_{init} \in \mathcal{C}$, and goal configuration, $q_{goal} \in \mathcal{C}$, are specified. The task is to compute a continuous, collision-free path in $\mathcal{C}$ that connects the initial and goal configurations.

A motion strategy problem will generally involve more complications and components, in addition to those from the path planning problem. The general motion strategy problem refers to the broad class of particular problems that include any or all of these complications. Some of these additional complications and components are listed below:

- The set of allowable velocities is locally constrained (nonholonomy). In the basic path planning problem, the robot is permitted to move locally in any direction. Suppose, however, that a mobile robot behaves like a car (see Figure 5), which can roll forward or backwards but not sideways. A wide variety of examples exist that have nonholonomic constraints, and a solution must satisfy these constraints in addition to avoiding collision [71].

- The dynamics of the robot significantly affect the desired motion strategy. Suppose, for example, that the car-like robot is unable to turn sharply at higher speeds due to skidding. One might be interested

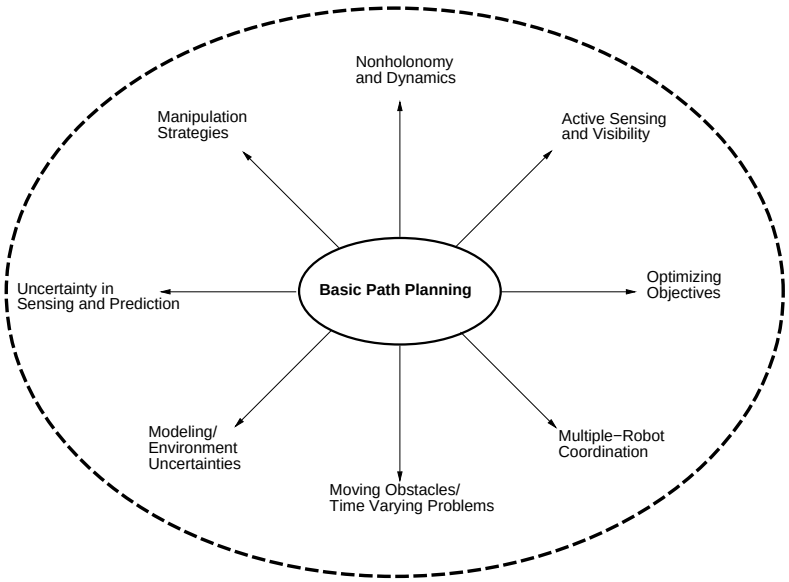


Figure 1: The general motion strategy problem encompassed by the game-theoretic framework can be viewed as a generalization of the basic path planning problem.

in driving the car-like robot to a goal region in minimum time. In general, dynamics can be modeled as nonholonomic constraints on a higher-dimensional state space.

- The robot is allowed to contact objects or obstacles in the world, and might even be required to manipulate them into some desired configuration. For example, a manipulator might be used to force parts into the same orientation for an assembly application.

- The world changes over time in a predictable way, as in the case of a moving obstacle or another robot with a known trajectory.

- The robot is unable to infer its exact configuration at a current time (uncertainty in configuration sensing). For example, a mobile robot might use dead-reckoning information and limited sonar data to estimate its location.

- Motion commands are given to the robot, but future configurations are not completely predictable. For example, an underwater robot might unexpectedly drift due to flow and turbulence. As another example, the wheels on a mobile robot might slip, making it difficult to predict future positions.

- A complete model of the world is not given. For example, a robot might use a laser range sensing device to incrementally build a representation of the world while attempting to achieve some task.

- The world changes in a way that is not completely predictable. For example, an indoor mobile robot might base its motion strategy on whether certain doors in the building are open or closed, even if these doors can change during execution.

- Several robots attempt to achieve tasks in a common environment.
- One robot has a visual sensor (e.g., a camera) that it must use to keep line-of-sight visibility with another robot or human.

In spite of the successful path planning model, there has been little attempt to obtain further benefits by broadening configuration space framework into a common mathematical structure that encompasses many of these important, well-studied extensions of the path planning problem. The intent of the current research is not to provide an alternative formulation of motion planning, but instead to present an expanded foundation that is built on previous geometric concepts, while characterizing and unifying a broader class of problems.

In the area of motion planning under uncertainty in sensing and prediction, many interesting concepts have been developed, such as preimages and forward projections [42, 69, 85, 88]); however, they are often tied to a particular set of uncertainty models that are based on crisp, geometric constructions (e.g., uncertainty cones and disks). In [78] it is shown how these concepts can be generalized to a broad class of problems that involve a variety of models and assumptions, and in [72, 81] the concepts are transported to other types of uncertainty problems. There are generally two forms of uncertainty that appear in motion planning and are addressed in this paper: uncertainty in predictability and uncertainty in sensing. With uncertainty in predictability, motion commands are issued to the robot(s), but future configurations or the future environment might be unknown. In this case, path planning is generally insufficient because a motion strategy must cause the robot(s) to respond appropriately to these unknown future states. As stated in [97], "...it remains a fundamental problem to develop dynamic movement planning algorithms," which motivated that research and lends support to the strategy concepts presented in this paper. State-feedback control laws are advocated in this work as a representation of a motion strategy, which can yield a distinct motion command for each possible state (representing the configuration, environment, velocity, time, etc.). The task is to select feedback strategies that takes into account a goal region and some performance concerns such as the distance traveled. With uncertainty in sensing, current information such as the configuration of the robot(s) or the environment representation, might not be known. Because the current state is not available, an information space that is based on sensing and control history can be defined, and motion strategies are consequently developed that use information feedback.

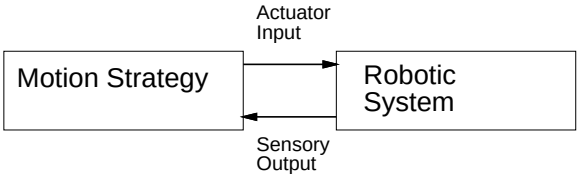
Specialized techniques that have been developed for the coordination of multiple robots that have independent goals can also be unified. Methods often vary significantly based on the use of decoupled planning (planning the paths independently and then coordinating the robot trajectories [15, 19, 40, 83]) or centralized planning (planning occurs in a composite configuration space [3, 8, 101]). For example, the decoupled planning representations constructed in [68, 90] can be generalized to a broad class of state spaces, including coordination along a configuration-space roadmap for each robot. Also, by recognizing the fact that the fixed-path coordination problem is equivalent to a simple form of planar nonholonomic planning, a straightforward Dijkstra-like algorithm can be adapted to quickly compute optimal coordination strategies [77]. In

general, feedback motion strategies can be determined that optimize one or more criteria in an appropriate game-theoretic sense (such as Pareto optimality or a Nash equilibrium), applying to a wide variety of multiple-robot coordination problems.

Since dynamics and control issues are traditionally decoupled from motion planning, limited solutions are obtained for many problems. Once one is confined to path planning and trajectory tracking, much of the interesting interaction between the geometric and dynamics issues is lost (along with optimal solutions to the original problem). This paper advocates using control-theoretic state space formulations of dynamics, which results in a nonholonomic planning problem (only first-order derivative constraints result) in which the configuration space has been replaced by a higher-dimensional state space that has similar geometric constraints. Using this approach, a method that computes solutions for a sufficiently broad class of nonholonomic planning problems can be directly applied to problems involving dynamics (this observation was exploited to immediately obtain a kinodynamic plan using a nonholonomic planner in [43]). In general, open-loop or feedback motion strategies can be designed that directly take into account dynamics and geometric constraints while optimizing some performance criterion.

1.1 Moving Beyond the Path Planning Paradigm

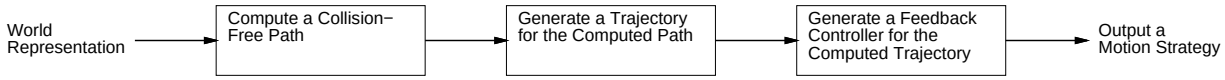
To provide some of the general motivation for this research, the path planning paradigm will be revisited and partially criticized (while acknowledging that the paradigm should also be praised in many aspects). Suppose that we wish to automate some task, such as placing a part in a particular orientation, or moving a mobile robot to a particular location. The following diagram depicts a conceptual relationship between robotic hardware, and software that provides some kind of "motion strategy" that controls the hardware:



It is, of course, not always wise to decouple these two components in the development of a robotic system; however, we can at least conceptually imagine being faced with two, interdependent tasks that involve designing a mechanical system and designing an algorithm. We know that the design of robotic systems that achieve great levels of autonomy and robustness is an extremely challenging and interdisciplinary effort, which places limitations on our abilities to analyze and design such systems.

One of the most common ways to handle difficult engineering problems is to use modular reasoning. It is always tempting to reduce a larger problem into subproblems, such that if each subproblem can be solved, a solution to the original problem can be assembled. For example, in classical computer vision research, the object recognition problem was modularized into segmentation (or feature extraction) and feature matching. In robotics, the motion strategy problem has been modularized for roughly two decades as shown below

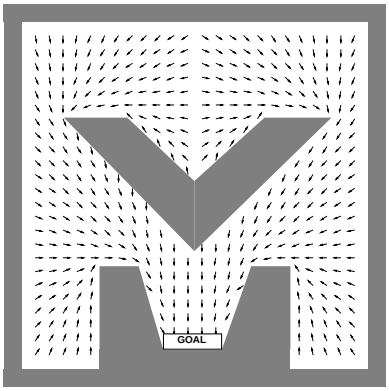
(although many exceptions exist in the literature):



The motion strategy problem has been typically divided into three modules: 1) first, a path is computed; 2) then, a trajectory is determined; 3) finally, a tracking controller is used to follow the trajectory. Although this approach is useful in many cases, it is important not to assume that this is the *only* way in which the motion strategy problem can be approached. It is well known that a carefully-constructed path might fail in a real system because it requires motions that do not necessarily satisfy kinematic and dynamic constraints. An integrated approach might attempt to tweak the path to obtain a viable solution; however, the overall approach is still limited due to modular reasoning.

Part of the approach taken in this research is to return to the basics, and develop algorithms that avoid the traditional modularization into path planning, trajectory following, and a "low-level" controller. Instead, a *motion strategy* is designed that attempts to achieve the task optimally, under the assumption that the models are valid. There is no need for the motion strategy to take the form of a path; the traditional need for a solution path in the configuration space is actually an artifact of modularization. An important concern should be: How does one generally represent a motion strategy if it is not a path in the configuration space?

The motion strategy problem can be modeled as a dynamic game that is played with one or more decision makers, which could correspond to robots. A special decision maker called "nature" is capable of interfering with the game, ultimately leading to uncertainties in prediction and sensing. The goal within this framework is to design a *strategy* that optimizes the choices of the decision makers under all possible contingencies with which they could be confronted. If there is uncertainty in prediction, then instead of a path, a navigation function can be used to represent a motion strategy. For example, with a car-like robot we might want to specify the best steering angle to choose from any possible configuration that the robot might be confronted with during execution (this is equivalent to a feedback control law [18]). A motion strategy for a 2D mobile-robot navigation problem might take the form of a vector field that indicates the best direction to travel from any (x,y) position:



The configuration space has been a powerful conceptual tool because it seems to be the natural space

where the path planning problem "lives." This is mainly because any transformation of a rigid or articulated body becomes a point in the configuration space. The configuration space does not sufficiently represent the problem in cases where additional complications such as dynamics or sensing uncertainty are present. The next concern should be: What is the natural space where the motion strategy problem "lives"? The game-theoretic mathematical foundation leads to the following spaces and their relationships:

$\mathcal{W}$	$\rightarrow$	$\mathcal{C}$	$\rightarrow$	$\mathcal{X}$	$\rightarrow$	$\mathcal{I}$
World or Workspace		Configuration Space		State Space		Information Space

The first step of transforming the world or workspace into the configuration space has already been achieved by the path planning framework [68, 84, 100]. In many problems, however, we might want to use a *state space* that encodes additional information. For example, if we would like to compute a motion strategy for a car-like robot that reaches a goal region in minimum time, the state space should include both configuration parameters and velocities. This kind of state space representation is common in modern control theory [24]. In some applications, the state space might encode information that represents the status of the environment [81]. In general, if there is sensing uncertainty (i.e., the current state is unknown during execution), the state space can be replaced by an *information space* [5]. This space is generally spanned by the history of previous sensor data and motion commands during execution, and can usually be transformed into either a space of probability density functions or subsets of the state space. Even though the state is not known, the information state (a point in the information space) is always known. Thus, the information space seems to be a natural place where the general motion strategy problem "lives."

If there is perfect prediction (or we design a sensorless system), then a motion strategy naturally takes the form of a path in either $\mathcal{C}$, $\mathcal{X}$, or $\mathcal{I}$. If, however, there is uncertainty in prediction, then a motion strategy takes the form of a feedback mapping on $\mathcal{C}$, $\mathcal{X}$, or $\mathcal{I}$, which gives a motion command to the robot from any location in the space. If the following was available, all of our problems would be solved: A practical algorithm that computes an optimal feedback motion strategy on any information space. It appears impractical to work toward this ideal; even the basic path planning problem is PSPACE-hard [95] (the information space is generally infinite dimensional). Nevertheless, we now have practical algorithms for path planning that have been very successful in practice, even though they fall short of the ideal of being complete, general algorithms. The intention is to develop adaptable algorithms that each incorporate some aspects of the general motion strategy problem. It is hoped that these steps can lead toward a greater understanding of the motion strategy problem and toward the development of practical algorithms that have broad applicability both in robotics and beyond. The same philosophy has helped advance path planning research, and it inspires the current research.

Section 2 presents a general game-theoretic structure that can be specialized to a variety of motion strategy problems. Section 3 shows how the game-theoretic concepts can be used as a representational tool to obtain new motion strategy concepts and generalizations of existing concepts. Section 4 presents a general approach for computing feedback motion strategies that are approximately optimal (at a given resolution),

and discusses a variety of related algorithmic issues. Conclusions are summarized in Section 5.

2 Formulating the Motion Strategy Problem

Before characterizing the motion strategy problem, an overview of the "game-theoretic" perspective is given. 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 paper *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 formulation presented in this section share similarities with concepts from statistical decision theory (e.g., [16, 29, 28]), optimal control theory (e.g., [2, 18, 64]), dynamic noncooperative game theory (e.g., [5, 55]).

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 [51, 99, 113]. In a situation in which there is a common loss functional and all decision makers wish to act cooperatively, *team theory* is obtained [25, 52, 63]. A *cooperative game* refers to the case in which some subsets of the decision makers can choose their actions in unison, so that a mutually beneficial outcome can be obtained [14, 92]. 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 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 [49, 55, 97, 109, 110, 111].

Game theory can also model a situation in which some decision makers represent disturbances that must be overcome by the other decision makers. As will be discussed shortly, such problems can be viewed as a *game against nature* [16, 94]. If this uncertainty is represented probabilistically, the game against nature becomes a problem of *stochastic optimal control theory* [13, 64]. If the uncertainty is represented nondeterministically and worst case analysis is performed, then the game against nature can be considered as a form of robust controller design [4].

Before considering a formulation of the general motion strategy problem, first consider making small extensions to the basic path planning problem. The basic problem is to find a continuous path $x : [0, t_f] \rightarrow \mathcal{C}_{free}$ such that $x(0) = q_{init}$ and $x(t_f) = q_{goal}$. Here, $\mathcal{C}_{free}$ refers to the set of configurations in which the robot is not in collision with static obstacles in the world, as defined in [68]. Usually, $\mathcal{C}_{free}$ implicitly incorporates all of the constraints due to the robot geometry and static obstacles in the workspace.

Suppose that there are nonholonomic constraints. To facilitate upcoming concepts, let $\mathcal{C}_{free}$ be renamed as a generic state space, $\mathcal{X} = \mathcal{C}_{free}$. It is well known that the nonholonomic constraints can be expressed as

$\dot{x} = f(x(t), u(t))$, which constrains the allowable vector fields on $\mathcal{X}$ [56]. Instead of directly choosing $x(t)$, one is forced to interact with the system using the input (or action) $u(t)$. This occurs, for example, when manipulating an object through pushing [87]. If $f(x(t), u(t)) = u(t)$ for $\|u(t)\| \leq 1$, the original holonomic, path planning problem is obtained since any desired, collision-free path in the state space can be obtained by selecting an appropriate input.

Suppose that optimality with respect to some criterion, such as path length or execution time, is important. A *loss functional* can be defined that evaluates any state trajectory and input:

$$L(x(\cdot), u(\cdot)) = \int_0^{t_f} l(x(t), u(t))dt + Q(x(t_f)). \tag{1}$$

The integrand $l(x(t), u(t))$ allows the specification of a cost that will accumulate during execution and will depend in general on the state trajectory and the input. The final term $Q(x(t_f))$ can indicate the importance of achieving the goal. The basic problem can be considered as a special form of optimal control [44]. Suppose $l(x(t), u(t)) \equiv 0$, and $Q(x(t_f)) = 0$ if $x(t_f) = q_{goal}$ and $Q(x(t_f)) = 1$ otherwise. This corresponds to the original case in which optimality is not important. The space of possible inputs to the system in this case is partitioned into two classes: those that lead to the goal region, and those that fail.

Next, consider moving to a mathematical structure for the general motion strategy problem, which is based on concepts from dynamic noncooperative game theory [5] and stochastic optimal control [64]. This structure will be formulated in discrete time to ease the specification of uncertainty aspects; however, continuous time can alternatively be used with some minor modifications and measure-theoretic restrictions. Thirteen components are first listed, and a discussion of how each relates to the motion strategy problem follows. Also, Figure 2 indicates both continuous-time and discrete-time specializations of the framework that can be used to formulate problems that have been considered in previous motion planning research.

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, $\mathcal{X}$, called the *state space*. The state of the game, x_k, at stage k, belongs to $\mathcal{X}$.

4. A set, U_k^i, defined for each $k \in \mathbf{K}$ and $i \in \mathbf{N}$, which is called the *action space* of the i^{th} decision maker at stage k. The *action*, u_k^i, at stage k, belongs to U_k^i. Generally, one can allow state-dependent action spaces of the form $U_k^i(x_k)$.

5. A set, Θ_k^a, defined for each $k \in \mathbf{K}$, which is called the *control action space for nature* at stage k. The *control action for nature*, θ_k^a, at stage k, belongs to Θ_k^a.

6. A function, $f_k : \mathcal{X} \times U_k^1 \times \ldots \times U_k^N \times \Theta_k^a \rightarrow \mathcal{X}$, defined for each $k \in \mathbf{K}$ so that

$$x_{k+1} = f_k(x_k, u_k^1, \ldots, u_k^N, \theta_k^a), \tag{2}$$

is a *state transition equation.*

7. A set, $\mathcal{Y}_k^i$, 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 $i \in \mathbf{N}$ and $k \in \mathbf{K}$, which is called the *sensing action space 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

$$y_k^i = h_k^i(x_k, \theta_k^{s,i}), \tag{3}$$

   which is the *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, $\{u_1^1, \ldots, u_{k-1}^N, y_1^1, \ldots, y_k^N\}$.

11. A set of all possible values for η_k^i, denoted by $\mathcal{I}_k^i$, which is called the *information space* for the i^{th} decision maker at stage k.

12. A set, Γ_k^i, of mappings $\gamma_k^i : \mathcal{I}_k^i \to U_k^i$, 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. A scalar-valued functional $L^i : (\mathcal{X} \times U_1^1 \times \ldots \times U_1^N) \times (\mathcal{X} \times U_2^1 \times \ldots \times U_2^N) \times \ldots \times (\mathcal{X} \times U_{K+1}^1 \times \ldots \times U_{K+1}^N) \times \Theta \to \Re^+$, 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 action spaces is represented here as Θ.

State transitions and control Item 1 defines the decision makers, which each typically refers to an independent, controllable robot, although other types of decision makers are possible. For example, one decision maker might be an unpredictable target that is attempting to avoid being located by another robot. Item 2 defines stages that correspond to times at which decisions are made. For standard discrete-time analysis, decisions are made at each Δt time increment. The limiting case of $K = \infty$ can be defined. In general, decision making at regular intervals is not required. Suppose, for instance, that the decisions correspond to very high-level operations which may have unpredictable completion times. This case is discussed in more detail in [102], for modeling the completion of a fine-motion operation.

The state space is defined in Item 3. At the very least, the state space can be used to represent the free configuration space, $\mathcal{C}_{free}$. In the case of multiple robots, it can represent the composite configuration space that is formed by taking the Cartesian product of the configuration spaces of the individual robots. In general, however, the state space could incorporate additional information. For instance, dynamics can

Problem	State Space	Motion Model	Sensing Model	Strategy
Basic MP [9, 91, 100]	$\mathcal{C}$	$\dot{x} = u(t)$ $x_{k+1} = x_k + u_k$	$y(t) = x(t)$ $y_k = x_k$	$[0,1] \rightarrow \mathcal{C}_{free}$
Nonholonomic [10, 20, 70, 82]	$\mathcal{C}$	$\dot{x} = f(x(t), u(t))$ $x_{k+1} = f(x_k, u_k)$	$y(t) = x(t)$ $y_k = x_k$	$u(t),\ 0 \leq t \leq t_f$
Dynamics [21, 34, 35, 79]	$T(\mathcal{C})$	$\dot{x} = f(x(t), u(t))$ $x_{k+1} = f(x_k, u_k)$	$y(t) = h(x(t))$ $y_k = h(x_k)$	$u(t),\ 0 \leq t \leq t_f$
Moving obstacles [41, 58]	$\mathcal{C}(t)$	$\dot{x} = f(x(t), u(t), t)$ $x_{k+1} = f_k(x_k, u_k)$	$y(t) = x(t)$ $y_k = x_k$	$u(t),\ 0 \leq t \leq t_f$
Pursuit-evasion [97]	$\mathcal{C}^1 \times \mathcal{C}^2$	$\dot{x} = f(x(t), u^1(t), u^2(t))$ $x_{k+1} = f(x_k, u_k^1, u_k^2)$	$y^i(t) = h^i(x(t))$ $y_k^i = h^i(x_k)$	$\gamma^1, \gamma^2 : \mathcal{X} \rightarrow U$
Multiple robots [19, 90, 106]	$\mathcal{C}^1 \times \cdots \times \mathcal{C}^N$	$\dot{x}^i = f^i(x^i(t), u^i(t))$ $x_{k+1}^i = f^i(x_k^i, u_k^i)$	$y^i(t) = x(t)$ $y_k^i = x_k$	$\gamma^i : \mathcal{X} \rightarrow U^i$
CP uncertainty [17, 31, 42]	$\mathcal{C}$	$\dot{x} = f(x(t), u(t), \theta^a(t))$ $x_{k+1} = f(x_k, u_k, \theta_k^a)$	$y(t) = x(t)$ $y_k = x_k$	$\gamma : \mathcal{X} \rightarrow U$
EP uncertainty [27, 81, 112]	$\mathcal{C} \times E$	$\dot{x} = f(x(t), u(t), \theta^a(t))$ $x_{k+1} = f(x_k, u_k, \theta_k^a)$	$y(t) = x(t)$ $y_k = x_k$	$\gamma : \mathcal{X} \rightarrow U$
CS and CP unc. [42, 69, 85]	$\mathcal{C}$	$\dot{x} = f(x(t), u(t), \theta^a(t))$ $x_{k+1} = f(x_k, u_k, \theta_k^a)$	$y(t) = h(x(t), \theta^s(t))$ $y_k = h(x_k, \theta_k^s)$	$\gamma : \mathcal{I}_k \rightarrow U$
ES uncertainty [30, 33, 57, 86]	$\mathcal{C} \times E$	$\dot{x} = u(t)$ $x_{k+1} = x_k + u_k$	$y(t) = h(x(t), \theta^s(t))$ $y_k = h(x_k, \theta_k^s)$	$\gamma : \mathcal{I}_k \rightarrow U$

Figure 2: Specific game components are shown for several different kinds of motion strategy problems that have received previous attention. For each case, the state space is identified. Both continuous-time and discrete-time models are given for motions and for sensing. An appropriate form of the strategy is shown in the final column. CP, EP, CS, and ES, refer to configuration predictability, environment predictability, configuration sensing, and environment sensing, respectively. The tangent bundle of $\mathcal{C}$ is denoted by $T(\mathcal{C})$, and E represents a space of environments.

be included by expanding the state space to include configuration time derivatives. This corresponds to the standard use of state space representations in optimal control theory. The state space can also include any parameters that can be completely or partially controlled through the operation of the robot(s). In [81] the state space includes *environment modes* that characterize varying conditions in the environment that potentially affect the robot.

Item 4 defines the set of actions that are available to each decision maker at each stage.

Item 5 is used to model sources of uncertainty. Two common representations of uncertainty have been applied to motion strategy problems. With a *nondeterministic* (or bounded-set) representation parameter uncertainties are restricted to lie within a specified set. A motion strategy is then generated that is based on *worst-case* analysis (e.g., [23, 42, 69, 85]). With a *probabilistic* representation the parameter uncertainties are characterized with a probability density function (pdf). This often leads to the construction of motion strategies through *average-case* or *expected-case* analysis (e.g., [17, 47, 48]).

One key aspect of the proposed mathematical foundation is a general capacity to model uncertainties by defining a nature player. This view of uncertainty was advocated for manipulation planning in [107]. It will be assumed that no decision maker has control over actions that are chosen by nature; however, models can

be constructed to partially predict nature's actions. Nature can introduce nondeterministic or probabilistic uncertainties into the game by applying either *control actions* or *sensing actions*. Item 5 defines the set of control actions that are available to nature, and Item 8 will define the set of sensing actions that are available to nature.

Item 6 defines how changes in state are effected. The next state x_{k+1}, at stage $k+1$, is obtained as a function of the current state x_k and the actions chosen by all decision makers, including nature. If nature is omitted from the state transition equation, then perfect prediction of future states is possible, given the actions of the decision makers. Under nondeterministic uncertainty, a set of possible future states can be derived from the state transition equation as:

$$F_{k+1}(x_k, u_k^1, \ldots, u_k^N) = \{f(x_k, u_k^1, \ldots, u_k^N, \theta_k^a) \in \mathcal{X} | \theta_k^a \in \Theta_k^a\}. \tag{4}$$

Under probabilistic uncertainty, it is assumed that $p(\theta_k^a)$, is known. By using the state transition equation, the next state is represented by a pdf, $p(x_{k+1}|x_k, u_k^1, \ldots, u_k^N)$.

Note that the discrete-time formulation can be considered as an approximation to a continuous-time system [11, 12]. For example, suppose a system of the form $\dot{x} = f(x(t), u^1(t), \ldots, u^N(t))$ is given. This can be approximated as

$$\frac{x(t+\Delta t) - x(t)}{\Delta t} = f(x(t), u^1(t), \ldots, u^N(t)) \tag{5}$$

or

$$x(t+\Delta t) = f(x(t), u^1(t), \ldots, u^N(t))\Delta t + x(t), \tag{6}$$

which is

$$x_{k+1} = x_k + f(x_k, u_k^1, \ldots, u_k^N)\Delta t, \tag{7}$$

in which $x_k \equiv x(t)$, $x_{k+1} \equiv x(t+\Delta t)$, and $u_k^i \equiv u^i(t)$. During the interval $[t, t+\Delta t)$, each control $u^i(t)$ remains constant. Many other discrete-time approximations, such as Runge-Kutta integration, are possible.

Sensing uncertainty Items 7 through 11 characterize the information that can be used for the basis of decision making when there is not direct access to the state. This can be considered as a general form of the sensing problem in robotics. Each decision maker at each stage has a *sensor space*, $\mathcal{Y}_k^i$ (or observation space), which is the set of possible measurements or observations that can be read at stage k. The function h_k^i relates the current state to the particular observation that is received at stage k. For example, an outdoor vehicular robot might receive a position estimate from a global positioning system (GPS). The function h_k^i relates latitude/longitude numbers from the GPS to the configuration of the robot. An another example, suppose a sonar sensor measures the distance of the robot to a wall along a particular direction. This distance depends on the configuration of the robot, and it provides partial information regarding the robot's configuration (although the exact configuration cannot be completely determined in general using this information). The relationship given by h_k^i is used in optimal control theory to define system outputs, and has also been

considered in robot sensing problems (see, for instance, [32, 33]). 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 $\Theta_k^{s,i}$. This can be used to account for noise and other unpredictable disturbances in the measurements.

Under nondeterministic uncertainty, the possible current states from a single sensor observation are

$$F_k^i(y_k^i) = \{x_k \in \mathcal{X} | y_k^i = h_k^i(x_k, \theta_k^{s,i}), \theta_k^{s,i} \in \Theta^{s,i}\}. \tag{8}$$

Under probabilistic uncertainty the current state, assuming only a single observation, is represented by a pdf, $p(x_k|y_k)$.

The sensing model can be generalized to include state history, $y_k^i = h_k^i(x_1, \ldots, x_k, \theta_k^{s,i})$.

Information spaces Items 10 and 11 characterize the history that is available for decision making. The relationship between sensor and action history and decision making has long been considered important in planning under uncertainty (e.g., [42, 85]). Generally one would like to optimize the performance of a 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, a useful characterization of this relationship is provided. When there is perfect state information, decisions can be made on the basis of state. However, with imperfect state information, the decisions are conditioned on information states. The information state concept is similar to the definition of knowledge states, considered in [38], and has also recently been proposed in [7].

The information space can be considered as a replacement for the state space in the case of imperfect state information (i.e., planning occurs in the information space instead of the state space). Since the information space is generated by a growing history, its dimension can increase linearly with the number of stages. This motivates the consideration of alternative representations when possible. In the case of nondeterministic uncertainty, the information space can be alternatively represented as an algebra of subsets of $\mathcal{X}$ that are obtained by performing set intersections that maintain consistency with the history. Thus for each possible action and sensing history, a set of possible current states can be identified. For example, in pursuit-evasion problems for which the evader position is unknown, the information state can represent possible locations of the evader, given the initial conditions and history of pursuer motions [80]. With probabilistic uncertainty, the information space can be alternatively represented as a pdf on $\mathcal{X}$ that is obtained through the repeated application of Bayes' rule. In the case of a linear Gaussian system with no geometric constraints, all possible pdfs are Gaussian, and the information space can be spanned by specifying only the mean and covariance. In general, the pdfs do not admit a low-dimensional parameterization; however, moments of the distributions can be considered as an approximate representation [72]. For practical purposes, one might have to consider low-dimensional transformations of the history, such as limiting the number of stages included in the history, or estimate some portion of the state vector.

The strategy concept Item 12 defines a strategy for each decision maker. The goal is to compute a strategy that will lead to the accomplishment of some robotic task, such as moving the robot to some prescribed region. In the most general form, each decision maker conditions its actions on its information state; however, various specializations of this are useful in particular contexts. For example, suppose Item 5 is dropped, implying that there is perfect prediction. In this case, for a given x_1 and sequence of actions, $u_1^i, \ldots, u_K^i$ for each $i \in \mathbf{N}$, the complete trajectory $x_2, \ldots, x_{K+1}$ can be determined using (2). In this case, feedback is not strictly necessary, and the solution strategy can be completely characterized by the action sequence. This corresponds to the case of open-loop control, and is equivalent to the type of motion strategy that is considered in basic path planning. Although prediction uncertainty is not explicitly modeled with nature, many applications exist for which the state-feedback solution can greatly improve performance by responding to on-line execution errors.

If there is perfect sensing information but uncertainty in predictability, then the strategy takes the form of a state-feedback mapping: $\gamma_k^i : \mathcal{X} \rightarrow U_k^i$. This occurs because the information space reduces to the state space under the perfect sensing model. Thus, for an n-dimensional state space, the domain of the strategy is n-dimensional (as opposed to a single dimension for a fixed-path strategy). Note that this represents quite a different solution concept that what is path planning. In this case, the robot is capable of responding to any unpredictable changes in state once a strategy is specified. This offers significant advantages over a preplanned path or trajectory. Instead of being forced to track a path, the robot can respond optimally from whichever states it finds itself in during execution.

Item 12 defines a deterministic (or pure) strategy; however, a randomized (or mixed) strategy can alternatively be defined. In this case a pdf of the form $p(u_k^i | \eta_k^i)$ is specified as the strategy, and actions are chosen by sampling. Randomized have been particularly useful for improving robustness in manipulation tasks [37].

Encoding preferences Item 13 defines a loss functional, which guides the selection of strategies, for each of the decision makers. The loss can generally be based on actions taken by any decision maker at any stage, on the state trajectory, and on nature.

One form that is often used in discrete-time optimal control theory is the stage-additive loss functional (indicated for the case a single decision maker):

$$L(x_1, \ldots, x_{K+1}, u_1, \ldots, u_K) = \sum_{k=1}^{K} l_k(x_k, u_k) + l_{K+1}(x_{K+1}), \tag{9}$$

in which $l_k(x_k, u_k)$ represents a cost that can accumulate (such as time, distance, or energy). The final cost, $l_{K+1}(x_{K+1})$, can be used to encode the goal. Desired final states can yield zero loss, and other final states can yield a large positive or infinite loss.

The general task is to determine strategies that optimize the losses in some appropriate sense. In the case of a single decision maker without nature, the task is to select a strategy that minimizes L. In the case

of nondeterministic actions from nature, the task is to select a strategy that minimizes the worst-case loss. In the probabilistic case, one natural choice is to minimize the expected loss. For cases in which there are multiple, independent decision makers, a number of different concepts may be appropriate. For instance, in a cooperative game in which there is a certain amount of trust, *Pareto optimality* may be appropriate [92]. In a noncooperative setting, a Nash equilibrium condition might be appropriate [5]. This corresponds to a game strategy that minimizes the loss of each decision maker, given that the strategies of the other decision makers cannot be changed.

3 Concepts for Analyzing Motion Strategy Problems

This section presents several conceptual tools that have been developed using the game-theoretic framework by building on concepts that have been introduced in previous motion planning contexts. The emphasis is placed on general motion strategy aspects; however, by using the game-theoretic formulation as a representational tool, particular models, analysis, algorithms, and computed solutions have so far been obtained for four particular classes of problems: (1) motion strategies under uncertainty in sensing and control [78, 76]; (2) motion strategies under environment uncertainties [72, 81]; (3) multiple-robot motion strategies [72, 77]; and (4) maintaining visibility of a predictable target in a cluttered workspace [75]. For the first problem class, a general method for determining feedback strategies is developed by blending ideas from 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 class, robot strategies are analyzed and determined for situations in which the environment is changing and not completely predictable. For the third problem class, dynamic game-theoretic concepts are applied to computer motion strategies for multiple robots that have independent goals. Several versions of the formulation have been considered: fixed-path coordination, coordination on independent configuration-space roadmaps, and centralized planning. For the fourth problem class, motion strategies are planned for a robot that must avoid obstacles, maintain visibility of second, predictable robot, and optimizes a loss functional that can take into account the total distance traveled, the distance between the two robots, and the amount of time that visibility is lost.

Modeling sources of uncertainty Several types of uncertainty will be discussed for the single-robot case. It is straightforward to extend the discussion to multiple robots. All types are modeled with nature, which can be assumed to be either nondeterministic or probabilistic.

Suppose that $\mathcal{X} = \mathcal{C}_{free}$, and let q_k denote the configuration (or state) at stage k. For the case of a single decision maker, the decision-maker superscripts will be dropped. The state transition equation (2) can be specialized to $q_{k+1} = f_k(q_k, u_k, \theta_k)$. This represents uncertainty in *configuration predictability*. Suppose further that the observation equation is of the form $y_k = h_k(q_k, \theta_k^s)$. This represents uncertainty in *configuration sensing*.

Other sources of uncertainty can be considered in addition to configuration uncertainties. Suppose, for example, that $\mathcal{C}_{free}$ is not exactly known, but is instead known to be one of several possibilities. In this case there is uncertainty in the robot's environment. A set E can be used to index the alternatives, and a state space is defined as some subset $\mathcal{X} \subset \mathcal{C} \times E$ [81]. Thus, for every $e \in E$, a different free configuration space can be obtained. Let $[q_k \; e_k]$ represent the state at stage k. A state transition equation can be defined in two portions. Suppose that the future configurations are obtained deterministically from $q_{k+1} = f_k'(q_k, u_k)$, and future environments are obtained from $e_{k+1} = f_k''(e_k, \theta_k^a)$. In this case nature causes uncertainty in *environment predictability*. More generally, the future environments can be conditioned on the robot's configuration (which occurs, for instance in a manipulation problem) and the action, to yield $e_{k+1} = f_k''(x_k, \theta_k^a)$, in which $x_k = [q_k \; e_k]$.

If the current environment is unknown, then there is uncertainty in *environment sensing*, which is a problem that has been considered in robotics from several different perspectives (e.g., [30, 36, 53, 67, 104]). This can be modeled by defining $y_k = h_k(x_k, \theta_k^s)$, in which $x_k = [q_k \; e_k]$.

In general, sensing and predictability uncertainties can be defined for any state space, including those that include dynamics. Also, a set of parameters could characterize variations in the model, and used to form models of uncertainty in predictability and sensing, in the same way that E was used.

It has been assumed thus far that each decision maker knows all game components, including the loss functionals, of other decision makers. Another sensing model 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 [46, 50]. The information of each decision maker could be represented, for example, as a pdf 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.

An example of uncertainty in configuration predictability A variety of motion models with uncertainty can be specified. As an example, consider characterizing the uncertainty model that is used for motion control in preimage planning research (e.g., [42, 69, 85]). Suppose there is a single decision maker that is a polygonal robot translating in the plane amidst polygonal obstacles. The action space defines 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 Gaussian density with zero mean and standard deviation ϵ_θ. If the robot chooses action u_k from state x_k, and nature chooses θ_k^a, then x_{k+1} is given by

$$f(x_k, u_k, \theta_k^a) = x_k + \|v\|\Delta t \begin{bmatrix} cos(u_k + \theta_k^a) \\ sin(u_k + \theta_k^a) \end{bmatrix}. \tag{10}$$

Thus, the game-theoretic interpretation of this model is that a nature player interferes with the commanded direction, causing the robot's heading to be unpredictable. A variety of other models can be easily encoded using this representation.

An example of uncertainty in configuration sensing Suppose that a single robot is equipped with a position sensor. Assume that the sensor is calibrated in a planar configuration space, yielding values in $\Re^2$. The observation equation is $y_k = h(x_k, \theta_k^s) = x_k + \theta_k^s$. Suppose the true configuration is known to lie within ϵ of the sensed configuration. In this case, the nondeterministic uncertainty model is appropriate, and θ_k^s is a 2D vector of magnitude ϵ. Alternatively, a probabilistic uncertainty model can be defined using a pdf. For example, a Gaussian model can be defined as

$$p(\theta_k^s) = \frac{1}{\sqrt{2\pi|\Sigma|}} e^{-\frac{1}{2}(\theta_k^s)^T \Sigma^{-1} \theta_k^s} \tag{11}$$

in which Σ represents the 2D covariance matrix of the error.

Thus, a nature player interferes with the sensor observations that are available to the robot, making it incapable of determining the true position.

An example of uncertainty in environment predictability Figure 7.a shows a problem in which a rigid robot is placed in a 2D world that corresponds to an indoor environment. There are two doors in the environment that might open or close at any point in time. When a door changes, the topology of $\mathcal{C}_{free}$ changes, and the robot must react appropriately. In fact, the robot must make its decisions based on whether certain doors *might* close during execution. One can design a motion strategy that minimizes the expected time that the robot takes to reach the goal if probabilistic models are available for the doors. For this problem, $E = \{1, 2, 3, 4\}$, which corresponds to the four possible cases in which the doors are open or closed. If the behavior of the doors does not depend on the configuration of the robot, then the portion of the state transition equation that corresponds to the environment can be defined as a probability $P(e_{k+1}|e_k)$ (assuming a Markov model).

An example of uncertainty in environment sensing Figure 3 shows an example that involves uncertainty in environment sensing. Suppose that a mobile robot is placed an unknown, indoor environment. The robot has an omnidirectional sensor that measures the distance to the nearest wall along each orientation in S^1. The value returned by the sensor can be considered as a function, $y_k : S^1 \rightarrow \Re$. The observation space, $\mathcal{Y}$, becomes a function space. Under nondeterministic uncertainty, the set of all possible environments that are consistent with y_k can be inferred. An information state corresponds to the set of possible environments after some history of sensor observations and actions.

Forward projections In preimage planning research [42, 85], the useful notion of a forward projection was introduced for characterizing robot execution when there is uncertainty in configuration predictability

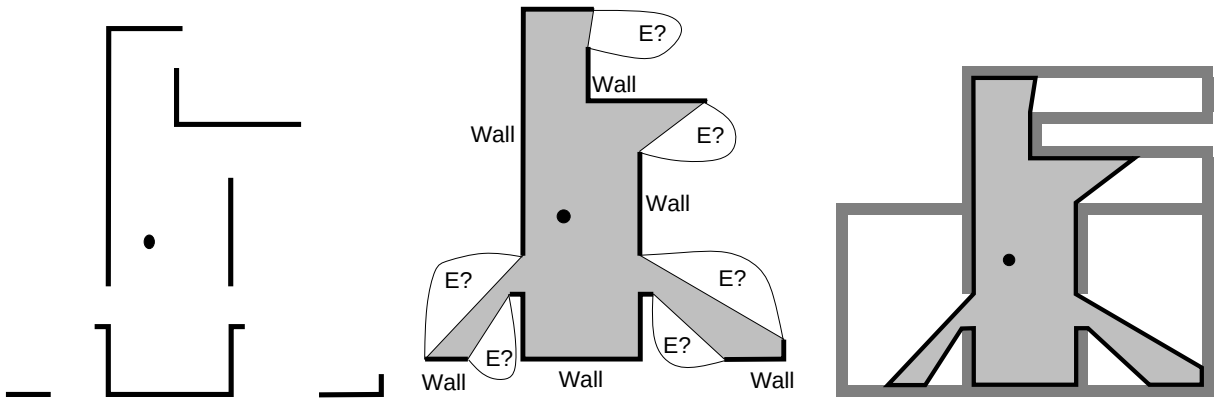


Figure 3: a) Typical appearance of data using omnidirectional sensing in an indoor environment; b) An on-line interpretation of the data: Discontinuities correspond to portions of the environment that are unknown; c) One possible environment that is consistent with the data. The information state for this problem represents the set of environments that are consistent with the observed data.

and sensing. This concept can be substantially generalized and applied to other motion strategy problems with similar benefits.

The forward projection characterizes future states under the implementation of a strategy. For a given strategy, initial state, and with no uncertainties, the state trajectory that can be inferred from (2). In many cases, one would like to consider the set of states that are reachable or likely to be reachable, given some set of possible actions or strategies. For example, in nonholonomic planning feasibility is a primary concern, which indicates the set of states that can be reached for at least one possible motion strategy.

When there are uncertainties in the actions taken by a decision maker, a forward projection can be used by other decision makers to assess future states (in a game of chess such future states are considered). In a pursuit-evasion problem, the pursuer might want to consider possible states that the evader could move to. This concept can also be applied to assess the effects of uncertainty due to nature. Suppose there is nondeterministic uncertainty, as in the model expressed in (10). Suppose that the strategy γ_k is fixed for all k, and that there is no sensing uncertainty. Under nondeterministic uncertainty, a subset of $\mathcal{X}$ in which the system state will lie can be inferred. Consider the state at stage x_{k+2}, if x_k is known. From (4), it is already known that $x_{k+1} \in F_{k+1}(x_k, u_k^1, \ldots, u_k^N)$, and $u_k^i = \gamma_k(x_k)$. The nondeterministic action of nature at stage $k+1$ must next be taken into account to yield

$$F_{k+2}(x_k, \gamma^1, \ldots, \gamma^N) = \{f(x_{k+1}, u_{k+1}^1, \ldots, u_{k+1}^N, \theta_{k+1}^a) \in \mathcal{X} | x_{k+1} \in F_{k+1}(x_k, \gamma^1, \ldots, \gamma^N), \;\; \theta_{k+1}^a \in \Theta^a\}. \tag{12}$$

This defines the forward projection at stage $k+2$ in terms of the projection at stage $k+1$. By induction, forward projections can be constructed to any future stage.

Forward projections can be analogously constructed for probabilistic uncertainty. For instance, the pdf

at stage $k+2$ is

$$p(x_{k+2}|x_k, \gamma^1 \ldots, \gamma^N) = \int p(x_{k+2}|x_{k+1}, \gamma^1_{k+1}(x_{k+1}), \ldots, \gamma^N_{k+1}(x_{k+1}))p(x_{k+1}|x_k, \gamma^1_k(x_k), \ldots, \gamma^N_k(x_k))dx_{k+1}. \tag{13}$$

Termination conditions The decision to halt a robot has been given careful attention in motion planning research that involves configuration-sensing uncertainty. A motion strategy might bring the robot into a goal region (*reachability*), but the robot may not halt if it does not realize that it is in the goal region (*recognizability*) [42]. 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 [42, 69, 85]. The same concept can be introduced in a game-theoretic formulation by defining a binary-valued mapping (as part of a strategy),

$$TC^i_k : \mathcal{I}^i_k \rightarrow \{true, false\}, \tag{14}$$

and enforcing the constraint that if $TC^i_k = true$, then $TC^i_{k+1} = true$. The *true* condition indicates that the robot should halt, and can be considered as a special action that can be considered by a decision maker (and hence incorporated into a strategy that uses information feedback). The loss functional can be defined so that $l^i_k = 0$ if $TC^i_k = true$. Thus, the robot has the opportunity to accept the loss at the current stage, as opposed to attempting to improve its loss. This termination condition, in the determination of an optimal strategy, is equivalent to an *optimal stopping rule*, which has been studied for problems such as deciding when to stop gambling [13]. The termination condition represents a special action that can be considered by a robot, and can be applied in a variety of contexts for designing motion strategies.

Performance preimages One concept that is complementary in many ways to the forward projection is the *preimage* [42, 69, 85]. A preimage is classically defined as the set of all configurations from which a robot is guaranteed to halt in the goal region under a constant motion command. This principle can be significantly generalized within the game-theoretic framework to yield a *performance preimage*, which is the set of all states (or information states) from which the performance lies within some specified bounds under the implementation of a fixed feedback strategy. This concept additionally relates closely to navigation functions [9, 98] and progress measures [39].

Assume that a strategy encodes a termination condition in addition to motion control, and that there is only a single decision maker (other than nature). Suppose that there is nondeterministic uncertainty, which is standard in preimage planning research. Consider some subset of the reals, $\mathcal{R} \subseteq \Re$. The *performance preimage on* X is the subset of $\mathcal{X}$ that is given by

$$\check{\pi}_x(\gamma, \mathcal{R}) = \{x_1 \in \mathcal{X} | \check{L}(x_1, \gamma) \in \mathcal{R}\}, \tag{15}$$

in which $\check{L}(x_1, \gamma)$ represents the worst-case loss that could be obtained under the implementation of γ with an initial state x_1. The set $\check{\pi}_x(\gamma, \mathcal{R}) \subseteq \mathcal{X}$ indicates places in the state space from which if the robot begins,

the loss will lie in $\mathcal{R}$.

The state space, $\mathcal{X}$, can be partitioned into *isoperformance classes* by defining an equivalence class $\check{\pi}_x(\gamma, \{r\})$ for each $r \in [0, \infty)$. For a 0-1 loss functional (zero if the goal is achieved), $\check{\pi}_x(\gamma, \{0\})$ yields the classical preimage. With a general loss functional, and $\mathcal{R} = [0, m)$ a performance preimage is obtained that indicates all $x_1 \in \mathcal{X}$ from which the goal will be achieved with a loss that is guaranteed to be less than m. If the termination condition is neglected, then $\check{\pi}(\gamma, \{0\})$ yields a *backprojection* similar to that in [42].

Suppose probabilistic uncertainty is considered instead of nondeterministic uncertainty. The performance preimage becomes

$$\bar{\pi}_x(\gamma, \mathcal{R}) = \{x_1 \in \mathcal{X} | \bar{L}(x_1, \gamma) \in \mathcal{R}\}, \tag{16}$$

in which $\bar{L}(x_1, \gamma)$ represents the *expected* loss that is obtained under the implementation of γ from x_1. Suppose that $\mathcal{R} = [0, r]$ for some $r \geq 0$. The performance preimage yields places in $\mathcal{X}$ from which the expected performance will be less than or equal to r. If $\mathcal{R} = \{r\}$ for some point $r \geq 0$, then places in $\mathcal{X}$ are obtained in which equal expected performance will be obtained. With a 0-1 loss functional and ignoring the termination condition, the performance preimages can give isoprobability curves which are equivalent to the probabilistic backprojections in [17].

Some examples of preimages are shown in Figure 4 (see also [78]). Figure 4(a) shows a performance preimage under nondeterministic uncertainty and a loss functional that returns 0 when the goal is achieved, and 1 otherwise. The curve shown in Figure 4(a) corresponds closely to the classical preimage that has been determined for this problem in previous manipulation planning research (e.g., [42, 68]). Figure 4(b) assumes probabilistic uncertainty, and shows probabilistic backprojections that are quite similar to those that appear in [17]. Figure 4(c) shows performance preimages for a case in which a Gaussian error model is used to represent the uncertainty in control, as opposed to a bounded uniform pdf as in [17]. Figure 4(d) shows performance preimages of a computed optimal strategy. These results were all computed using variants of the algorithm discussed in Section 4.

Performance preimages can also be defined on the information space to account for sensing uncertainty, and for multiple decision makers [72].

Decoupling multiple robots Consider the problem of coordinating multiple robots that have independent goals. Approaches to multiple-robot motion coordination 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 (e.g., [8, 101]). A decoupled approach typically generates paths for each robot independently, and then considers the interactions between the robots (e.g., [41, 58, 90]). 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.

A variety of multiple-robot coordination problems can be formulated by defining appropriate state spaces

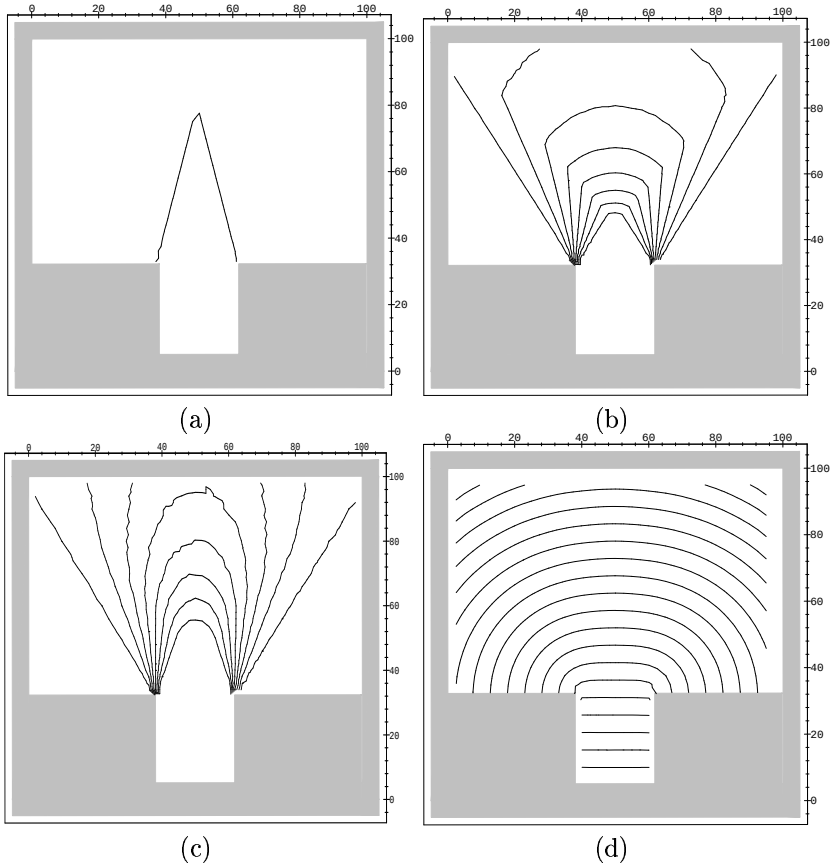


Figure 4: Several performance preimages for the classic peg-in-hole problem using the motion model given in Section 3: (a) a classical preimage; (b) a single-stage probabilistic preimage for a uniform pdf; (c) a single-stage probabilistic preimage for a truncated Gaussian state transition pdf; and a computed optimal strategy for a different problem: (d) performance preimages for the optimal solution using a minimum-distance criterion.

[72]. Suppose there is a collection of N robots that share a common workspace and have free spaces $\mathcal{C}^1_{free}, \ldots, \mathcal{C}^N_{free}$. The state space can be defined as the Cartesian product

$$\mathcal{X} = \mathcal{C}^1_{free} \times \mathcal{C}^2_{free} \times \cdots \times \mathcal{C}^N_{free}. \tag{17}$$

The subset of $\mathcal{X}$ in which two or more robots collide is avoided in a successful motion strategy. The dimensionality of this composite space has previously prompted many approaches that decouple the problem. Motion strategies are more or less constructed for each robot independently, and then combined to coordinate the robots.

For fixed-path coordination, it is assumed that a collision-free path $\tau^i : [0,1] \rightarrow \mathcal{C}^i_{free}$ is given for each robot, and the state space is defined as the Cartesian product $[0,1]^N$. Instead of a single collision-free path, suppose that each robot is given a network of collision-free paths, referred to as a *roadmap*. Let $\mathcal{R}^i$ denote a space that is formed by combining the domains of the roadmap paths for the i^{th} robot. A roadmap

coordination space can be defined as

$$\mathcal{X} = \mathcal{R}^1 \times \mathcal{R}^2 \times \cdots \times \mathcal{R}^N. \tag{18}$$

In general, many other combinations of constrained spaces are possible to define the state, leading to a variety of ways to define decoupled planning problems.

Multiple-robot optimality Little concern has been given in previous research to optimality for multiple-robot coordination problems. Previous approaches that consider optimality project the vector of individual losses onto a scalar loss [15, 103, 108]. As a result, these methods can fail to find many potentially useful motion strategies. There are many well-studied optimality concepts from game-theory and multiobjective optimization literature that can be applied in this case. An optimality concept will be briefly described for the multiple-robot planning problem that results in a small set of alternative strategies that are guaranteed to be better than or equivalent to (in terms of losses) any other possible strategy.

For each robot, assume there are no uncertainties and define a loss functional of the form

$$L^i(x_{init}, x_{goal}, u^1, \ldots, u^N) = \int_0^T l^i(t, x^i(t), u^i(t))dt + \sum_{j \neq i} c^{ij}(x(\cdot)) + q^i(x^i(T)), \tag{19}$$

which maps to the extended reals, and

$$c^{ij}(x(\cdot)) = \begin{cases} 0 & \text{if } x(t) \in \mathcal{X}_{valid} \text{ for all } t \\ \\ \infty & \text{otherwise} \end{cases} \tag{20}$$

and

$$q^i(x^i(T)) = \begin{cases} 0 & \text{if } x^i(T) = x^i_{goal} \\ \infty & \text{otherwise} \end{cases} . \tag{21}$$

The variables x_{init} and x_{goal} represent the initial and goal configurations for all of the robots.

The integrand l^i represents a continuous cost function, which is a standard form that is used in optimal control theory. It is additionally required, however, that

$$l^i(t, x^i(t), u^i(t)) = 0 \quad \text{if } x^i(t) = x^i_{goal}. \tag{22}$$

This implies that no additional cost is received while the i^{th} robot "waits" at x^i_{goal} until time T. The term (20) penalizes collisions between the robots. The subset $\mathcal{X}_{valid} \subset \mathcal{X}$ represents the (closed) set of all states at which no robots or obstacles are in collision. This has the effect of preventing any robots from considering game strategies that lead to collision. The term (21) represents the goal in terms of performance. If the i^{th} robot fails to achieve its goal, x^i_{goal}, then it receives infinite loss.

Suppose that the initial state is given. For each game strategy, $\gamma = \{\gamma^1, \ldots, \gamma^N\}$, a vector of losses will be obtained. A partial ordering, $\preceq$, can be defined on the space of game strategies, Γ. For a pair of elements $\gamma, \gamma' \in \Gamma$ let $\gamma \preceq \gamma'$ if $L^i(\gamma) \leq L^i(\gamma')$ for every i. The minimal game strategies with respect to $\preceq$ are better

than or equal to all other game strategies in Γ, and it is shown in [72] that very few minimal game strategies typically exist (ignoring those that produce equivalent losses).

These solutions can be generated using algorithms that are based on the dynamic programming principle. For other applications this was observed in [26]. For the criterion (19) it is shown that minimal solutions are consistent with other well-established forms of optimality from optimization literature [72]. The minimal game strategies are equivalent to the *nondominated* strategies used in multiobjective optimization and *Pareto optimal* game strategies used in cooperative game theory. Furthermore, it can be shown that the minimal game strategies satisfy the Nash equilibrium condition from noncooperative game theory, which implies that for a game strategy $\gamma^* = \{\gamma^{1*} \dots \gamma^{N*}\}$, the following holds for each i and each $\gamma^i \in \Gamma^i$:

$$L^i(\gamma^{1*}, \dots, \gamma^{i*}, \dots \gamma^{N*}) \leq L^i(\gamma^{1*}, \dots, \gamma^{i-1*}, \gamma^i, \gamma^{i+1*} \dots \gamma^{N*}). \tag{23}$$

Thus, the minimal game strategies represent reasonable coordination strategies for multiple robots under a variety of different interpretations.

Moving obstacles and other nonstationary systems It has been assumed so far that the system is not time-varying. From a control perspective, this corresponds to a *stationary* problem. Optimal solutions to problems of this type depend only on state (or the information state with sensing uncertainty) and not on time or the stage index.

By allowing time-varying models, many interesting motion strategy problems can be defined. Suppose, for instance, that several moving obstacles exist in the workspace. For a single-robot problem, this leads to a time-varying free configuration space $\mathcal{X}(t) \equiv \mathcal{C}_{free}(t)$ [68], which can be approximated in discrete time as

$$\mathcal{X}[k] = \bigcap_{t \in [(k-1)\Delta t, k\Delta t)} \mathcal{X}(t). \tag{24}$$

In general, many game items from Section 2 can encode time-dependent models. In these cases, the motion strategies γ_k^i and γ_{k+1}^i will generally be different due to changes in the model.

Representing nonholonomic constraints A simple example is presented to illustrate how the state space formulation can be utilized for encoding nonholonomic constraints and dynamics. Suppose that one would like to design a motion strategy for a car-like robot that has a bounded turning radius. Consider the classical case in which only kinematics are taken into account. Let $x = [x_1 \ x_2 \ x_3]$, which spans a three-dimensional state space ($\mathcal{X} = \Re^2 \times S^1$). Let x_1 and x_2 represent the translational position of the car in the plane, and let $x_3 \in [0, 2\pi)$ represent the orientation. Let u represent the steering angle. The velocity constraints can be specified in the form $\dot{x} = f(x, u)$:

$$\dot{x} = s \begin{bmatrix} cos(x_3) \\ sin(x_3) \\ tan(u)/L \end{bmatrix}, \tag{25}$$

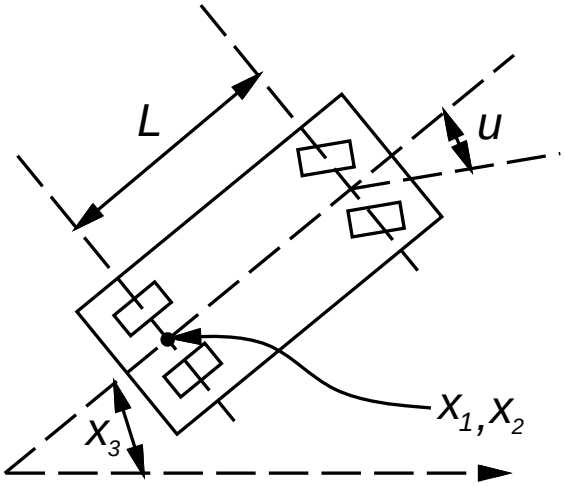


Figure 5: A car-like robot.

in which L is the distance between the front and rear axles, and s is the vehicle speed. A discrete-time representation is straightforward to obtain [72]. In this formulation, the car is only allowed to drive forward.

Suppose that some dynamic constraints must additionally be satisfied. A very naive model is given here as an example; vehicle dynamics models are usually much more complicated. Let x_4 denote a fourth state variable, which represents the speed of the car. Let f_{max} be the maximum centrifugal force that the car can withstand before laterally slipping or toppling. Let ϕ_{max} be the maximum steering angle (based on mechanical limits), and let s_{max} be the maximum speed the car can go with steering angle ϕ_{max}. Let m be the mass of the car and L, be the distance between the front and rear axles of the car. Thus,

$$f_{max} = \frac{m s_{max}^2 \tan \phi_{max}}{L}. \tag{26}$$

If the car is going at a speed x_4, then the limit on the steering angle, u_1, is

$$m s^2 L \tan u_1 \leq f_{max}. \tag{27}$$

Using Equation 26,

$$u_1 \leq arctan(s_{max}^2 \tan \phi_{max} / s^2). \tag{28}$$

A second input, u_2, represents the acceleration of the car (which is assumed to be bounded). This results in

$$\dot{x} = \begin{bmatrix} x_4 \, cos(x_3) \\ x_4 \, sin(x_3) \\ x_4 \, tan(u_1)/L \\ u_2 \end{bmatrix}, \tag{29}$$

in which u_1 must obey (28), resulting in a state-dependent action space, $U_1(x_4)$. In this new state space, there are standard geometric obstacle constraints on the variables x_1, x_2, x_3, and an additional constraint on the speed, x_4, can exist. This dynamical model is quite simple, and a variety of other, more sophisticated models [45] could be formulated in state-space terms.

In general, robot dynamics can be expressed in terms of q, $\dot{q}$, and $\ddot{q}$. If the state vector, x, is defined as $x = [q \;\; \dot{q}]$, the dynamics can be written in the form $\dot{x} = f(x(t), u(t))$. This is a common representation from control theory, which converts higher-order differential equations into higher-dimensional first-order differential equations. In the case of robot dynamics on an n-dimensional configuration space, the state space is $2n$-dimensional.

4 Computing Optimal Strategies

This section presents general algorithm issues that result from computing approximate optimal motion strategies using the game-theoretic models. The intention is not to present an algorithm that advances the state-of-the-art for a particular problem, but instead to indicate some of the general computational issues that arise for a broad class of problems. For many particular problems that fall within the game-theoretic framework, it might be possible to exploit special properties to develop an algorithm that is far superior to the general methods stated here. For example, in [80], a combinatorial representation of the information space led to a complete algorithm for computing a motion strategy for a pursuer in a 2D environment, which is guaranteed to lead to line-of-sight visibility of an evading target.

The quality of the approximate solutions computing by the method described in this section depends on the resolution of the representation chosen for the state space and action space. In basic path planning, a single algorithm can be often applied with only minor modification to a variety of specific problems. For example, the randomized path planner in [8] has been applied to many examples including manipulator systems and rigid robots. Part of the ease of this applicability is due to the common configuration space representation. To make the game-theoretic ideas convincing, algorithms developed within the broader mathematical foundation should have similar portability. Indeed this is the case with the algorithm that is presented and discussed in this section. The particular case of computing optimal feedback strategies for a robot that has perfect sensing and probabilistic uncertainty in predictability is discussed; however, variants of this approach have been applied to other forms of uncertainty and to coordinating multiple robots. Related details and dozens of computed examples for a variety of motion strategy problems are presented in [72].

The efforts are restricted to obtaining approximate solutions for three primary reasons: 1) known lower-bound hardness results for basic path planning and a variety of extensions; 2) exact methods often depend strongly on specialized analysis for a specific problem class; and 3) the set of related optimal-control and dynamic-game problems for which analytical solutions are available is quite restrictive. The computational hardness results have curbed many efforts to find efficient, complete algorithms to general motion strategy problems. In [95] the basic path planning problem was shown to be PSPACE-hard for polyhedral robots with n links. In [22] is was shown that computing minimum-distance paths in a 3-D workspace is NP-hard. It was also shown that the compliant motion control problem with sensing uncertainty is nondeterministic exponential time hard. In [96] it was shown that planning the motion of a disk in a 3-D environment with

rotating obstacles is PSPACE-hard. In [97], a 3-D pursuit-evasion problem is shown to be exponential time hard, even though there is perfect sensing information. Such results have turned efforts toward approximate techniques. For example, a polynomial-time algorithm is given in [93] for computing epsilon approximations of minimum-distance paths in a 3-D environment. Also, randomized techniques are used to compute solutions for high degree-of-freedom problems that are unapproachable by complete methods [1, 9, 59, 106].

The second motivation for considering approximate solutions is to avoid specialized analysis of particular cases, with the intent of allowing the algorithms to be adaptable to other problem classes. Of course, in many cases there is great value in obtaining an exact solutions to a specialized class of problems. The approach described in this paper can be considered as a general way to approximate solutions that might be sufficient for a particular application, or the approach might at least provide some understanding of the solutions.

The final motivation for considering approximate solutions is that the class of related optimal-control and dynamic-game problems that can be solved directly is fairly restrictive. In both control theory and dynamic game theory, the classic set of problems that can be solved are those with a linear state transition equation and quadratic loss functional [2, 5, 18, 64]. Because few problems can be solved analytically, there has been a large focus on numerical dynamic optimization procedures [12, 13, 65, 66], particularly in robotics applications [6, 54, 89, 105].

The algorithm description is organized into three parts. First, the general principle of optimality is described, which greatly reduces the amount of effort that is required to compute optimal strategies. The next part describes how cost-to-go functions are computed as an intermediate representation of the optimal strategy. The third part describes how the cost-to-go is used as a navigation function to execute the represented strategy (i.e., selecting optimal actions during on-line execution). Following this, basic complexity assessments are given.

Exploiting the principle of optimality Because the decision making expressed in Item 6 of Section 2 is iterative, the dynamic programming principle can generally be employed to avoid brute-force enumeration of alternative strategies, and it forms the basis of the general algorithm. Although there are obvious connections to dynamic programming in graph search, it is important to note the distinctions between Dijkstra's algorithm and the usage of the dynamic programming principle in this paper. In optimal control theory, the dynamic programming principle is represented as a differential equation (or difference equation in discrete time) that can be used to directly solve a problem such as the linear-quadratic Gaussian regulator [64], or can be used for computing numerical approximations of optimal strategies [65]. In the general case, the differential equation is expressed in terms of time-dependent *cost-to-go* functions. The cost-to-go is a function on the state space (or information space if there is imperfect sensing) that expresses the cost that is received under the implementation of an optimal strategy from that particular state and time. In some cases, the time index can be eliminated, as in the special case of values stored at vertices (states) in the execution of Dijkstra's algorithm.

For the discrete-time model in Section 2, the dynamic programming principle is expressed as a difference equation. First, consider the special case of $x_{k+1} = f(x_k, u_k)$ (i.e., a single robot and there are no uncertainties), and the task is to compute an optimal state-feedback strategy (i.e., perfect sensing). The cost-to-go function at stage k is defined as

$$L_k^*(x_k) = \min_{u_k, \ldots, u_K} \left\{ \sum_{i=k}^{K} l_i(x_i, u_i) + l_{K+1}(x_{K+1}) \right\}. \tag{30}$$

Since x_k is given, the choice of u_k locally specifies the strategy γ_k (i.e., $u_k = \gamma_k(x_k)$). The cost-to-go can be separated:

$$L_k^*(x_k) = \min_{u_k} \min_{u_{k+1}, \ldots, u_K} \left\{ l_k(x_k, u_k) + \sum_{i=k+1}^{K} l_k(x_k, u_i(x_i)) + l_{K+1}(x_{K+1}) \right\}. \tag{31}$$

The second *min* does not affect the l_k term; thus, it can be removed to obtain

$$L_k^*(x_k) = \min_{u_k} \left[l_k(x_k, u_k) + \min_{u_{k+1}, \ldots, u_K} \left\{ \sum_{i=k+1}^{K} l_k(x_k, u_i(x_i)) + l_{K+1}(x_{K+1}) \right\} \right]. \tag{32}$$

The second portion of the *min* represents the cost-to-go function for stage $k+1$, yielding [11]

$$L_k^*(x_k) = \min_{u_k} \left\{ l_k(x_k, u_k(x_k)) + L_{k+1}^*(x_{k+1}) \right\}. \tag{33}$$

This final form represents a powerful constraint on the set of optimal strategies. The optimal strategy at stage k and state x depends only cost-to-go values at stage $k+1$. Furthermore, only the particular cost-to-go values that are reachable from the state transition equation, $x_{k+1} = f(x_k, u_k)$, need to be considered. The dependencies are local; yet, the globally-optimal strategy is characterized.

Expressions similar to (33) can be obtained for many extensions, variants, and optimality concepts. For example, suppose that there is probabilistic uncertainty in predictability, resulting in the state transition equation $x_{k+1} = f(x_k, u_k, \theta_k)$. The actions for nature are sampled from the probability density $p(\theta_k)$. Using this model, a probability density function can be derived for next states of the form $p(x_{k+1}|x_k, u_k)$. The task is to design a strategy that is optimal in the expected sense. In this case the cost-to-go is defined as [12]

$$\bar{L}_k^*(x_k) = E \left\{ \sum_{i=k}^{K} l_i(x_i, u_i) + l_{K+1}(x_{K+1}) \right\}, \tag{34}$$

in which $E\{\}$ denotes expectation taken over the actions of nature. The dynamic programming principle yields

$$\bar{L}_k^*(x_k) = \min_{\gamma_k \in \Gamma_k} \left\{ l_k(x_k, u_k) + \int \bar{L}_{k+1}^*(x_{k+1}) p(x_{k+1}|x_k, u_k) dx_{k+1} \right\}, \tag{35}$$

which is analogous to (33). Other variations include finding multiple Nash equilibria, worst-case strategies, and determining cost-to-go functions directly on the information space when there is uncertainty in sensing [72].

Iteratively approximating cost-to-go functions An optimal strategy can be computed by successively building approximate representations of the cost-to-go functions. One straightforward way to represent a cost-to-go function is to specify its values at each location in a discretized representation of the state space. Note that this requires visiting the entire state space to determine a strategy. Although this is normally unacceptable for a basic path planning problem, for the extensions considered in this paper the solution is a feedback mapping instead of a fixed path (i.e., the domain of the function in the solution is n-dimensional instead of one-dimensional). Since a strategy must produce an appropriate action from any state, it is quite reasonable to visit the entire state space (the dynamic programming principle avoids brute-force exploration of the *strategy space*). Additionally note that the cost-to-go function is encoding a globally-optimal solution which must take into account all of the appropriate geometric and topological information at a given resolution. Artificial potential functions have often been constructed very efficiently in path planning approaches; however, these approaches heuristically estimate the cost-to-go and are typically prone to have local minima [9, 62].

Suppose there are no uncertainties, and an optimal state-feedback strategy is sought using (33). Assume that the problem is *stationary*, which implies that no model components are time varying. The first step is to construct a representation of L^*_{K+1}. The final term, $l_{K+1}(x_{K+1})$, of the loss functional is directly used to assign values of $L^*_{K+1}(x_{K+1})$ at discretized locations. Typically, $l_{K+1}(x_{K+1}) = 0$ if x_{K+1} lies in the goal region, and $l_{K+1}(x_{K+1}) = \infty$ otherwise. This only permits trajectories that terminate in the goal region.

The dynamic programming equation (33) is used to compute the next cost-to-go function, L^*_K, and subsequent cost-to-go functions. For each quantized state, x_k, a quantized set of actions $u_k \in U$ are evaluated. For a given action u_k, the next state obtained by $x_{k+1} = f(x_k, u_k)$ generally might not lie on a quantized state. Linear interpolation between neighboring quantized states can be used, however, to obtain the appropriate loss value without restricting the motions to the grid (see Figure 6.a). Other schemes, such as quadratic interpolation, can be used to improve numerical accuracy at the expense of computation time [66]. Convergence properties of the quantization and interpolation are discussed in [11, 12]. For a motion planning problem, the obstacle constraints must additionally be taken into account. The constraints can be directly evaluated each time to determine whether each x_{k+1} lies in the free space, or a bitmap representation of the configuration space can be used for quick evaluations (an efficient algorithm for building a bitmap representation of $\mathcal{C}_{free}$ is given in [60]).

Note that L^*_K represents the cost of the optimal one-stage strategy from each state x_K. More generally, L^*_{K-i} represents the cost of the optimal $(i+1)$-stage strategy from each state x_{K-i}. For a motion strategy problem, one is typically concerned only with strategies that require a finite number of stages before terminating in the goal region. For a positive $\delta \approx 0$ the dynamic programming iterations are terminated when $|L^*_k(x_k) - L^*_{k+1}(x_{k+1})| < \delta$ for all values in the state space. This assumes that the robot is capable of selecting actions that halt it in the goal region. The resulting stabilized cost-to-go function can be considered as a representation of the optimal strategy. Note that no choice of K is necessary. Also, only the representation

of L^*_{k+1} is retained while constructing L^*_k; earlier representations can be discarded to save storage space.

The general applicability of these kinds of computations was noted long ago in [65]: 1) extremely general types of system equations, performance criteria, and constraints can be handled; 2) particular questions of existence and uniqueness are avoided; 3) a true feedback solution is directly generated.

These same advantages apply to motion strategy problems that are formulated in this paper. For example, suppose that there is probabilistic uncertainty in predictability, which results in the dynamic programming equation (35). The computation of $L^*_k(x_k)$ was previously computed by trying choices of u_k; however, in the probabilistic case, for each action u_k, the actions of nature, θ^a_k, are attempted. The cost-to-go is computed by selecting the action that produces the minimum expectation over the actions of nature. Using nondeterministic uncertainty and worst-case analysis, the action is selected that produces the least loss, over all of the actions of nature. In the case of uncertainty in sensing, cost-to-go functions are instead defined and computed on the information space (or an approximation of the information space). Some problems involve nonstationary information, such as tracking a predictable target in a cluttered environment; in this case optimal solutions can be computed by retaining all cost-to-go functions from stages 1 to $K+1$ [75]. In some problems, multiple "optimal" solutions are possible. This occurs, for instance, in the coordination of multiple robots that have independent goals [77]. One might want to compute the set of Nash equilibria, as discussed in Section 3. Instead of storing a scalar loss, the cost-to-go at a state can be expanded to include a set of alternative strategies and corresponding vectors of losses. Several variations are discussed in further detail in [72]. In all of these cases, feedback strategies are determined that can respond quickly to online changes, without necessarily making the traditional assumption that the motion strategy and on-line control are decoupled.

Using the cost-to-go as a navigation function To execute the optimal strategy, an appropriate action must be chosen using the cost-to-go representation from any given state (see Figure 6.b). One approach would be to simply store the action that produced the optimal cost-to-go value, for each quantized state. The appropriate action could then be selected by recalling the stored action at the nearest quantized state. This method could cause errors, particularly since it does not utilize any benefits of interpolation. A preferred alternative is to select actions by locally evaluating (33) (or the appropriate dynamic programming equation) at the exact current state. Linear interpolation can be used as before. Note that although the approach to select the action is local (and efficient), the global information is still taken into account (it is encoded in the cost-to-go function). This concept is similar to the use of a numerical navigation function in previous motion planning literature [9, 98], and the cost-to-go is a form of *progress measure*, as considered in [39]. When considering the cost-to-go as a navigation function, it is important to note that it does not contain local minima because it is constructed as a by product of determining the optimal solution. Once the optimal action is determined, an exact next state is obtained. This form of iteration continues until the goal is reached or a termination condition is met. During the time between stages, the state trajectory can be

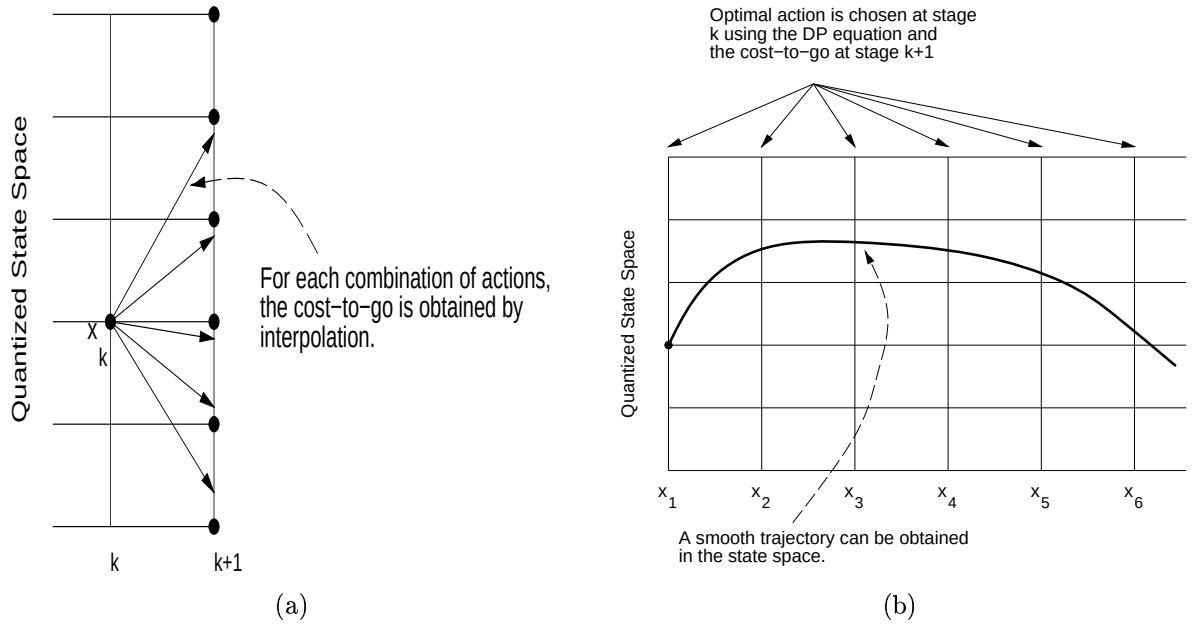


Figure 6: The computations are illustrated with a one-dimensional state space. (a) The cost-to-go is obtained from at the next stage by interpolation of the values at the neighboring quantized states. (b) During execution, interpolation can also be used to obtain a smooth trajectory.

linearly interpolated between the endpoints given by the discrete-time state transition equation, or can be integrated using an original continuous-time state transition equation.

Computational expense Consider the computation time for the dynamic programming algorithm for the basic case modeled by (33). Let c denote the number of quantized values per axis of the state space. Let n denote the dimension of the state space. Let a denote the number of quantized actions. Each stage of the cost-to-go computations takes time $O(c^n a)$, and the number of stages before stabilization is nearly equal to the longest optimal trajectory (in terms of the number of stages) that reaches the goal. The space complexity is obviously $O(c^n)$. The algorithm is efficient for fixed dimension, yet suffers from the exponential dependence on dimension that appears in most deterministic path planning algorithms. The utilization of the cost-to-go function during execution requires $O(a)$ time in each stage. These time complexities assume constant evaluation time of the cost-to-go at the next stage; however, if multilinear interpolation is used, then additional exponential-time computation is added because 2^n neighbors are evaluated. Consider the case of uncertainty in predictability. Let θ denote the number of actions available to nature. In this case, the time complexity of each cost-to-go computation stage is $O(c^n a\,\theta)$, and the time to execute the strategy is $O(a\theta)$ per stage. The space complexity remains unchanged. Related dynamic programming algorithms that apply to variants such as nonstationarity, multiple robots, and sensing uncertainty are also straightforward to analyze.

Execution times for practical examples vary dramatically depending on the resolutions, but computation times typically range from a few seconds for a basic 2-D problem up to several hours for a challenging 3-D or 4-D problem with uncertainties, on a typical workstation [72]. It is important to note, however, that this algorithm is not competing with known algorithms that apply to the basic path planning problem, since the algorithm described in this paper computes a state-feedback strategy.

Computed examples To indicate the level of difficulty that can be handled by the dynamic programming approach described in this section, examples of two motion planning problems that can be solved appear in Figure 7. These particular problems involve an environment that changes over time and is not completely predictable (more details appear in [81]). Figure 7.a shows a problem for which there is a single rigid robot that can rotate in place or translate along its major axis. There are two doors that can become open or closed at various points in the future, and the behavior of the doors is modeled with a Markov process. The state space for this problem is the Cartesian product of the configuration space of the robot and a set of four possible combinations of open and closed doors. Figures 7.b and 7.c show two simulated executions under the implementation of a computed strategy that minimizes the expected time to reach the goal. Different trajectories are taken in different executions because the openings and closings of doors vary; however, both behaviors are obtained from the same strategy. Figure 7.d shows a problem in which there is a nonholonomic car robot that is capable of only moving in a forward direction and has a limited turning radius. There are two regions in the workspace that are designated as service areas. In this case, the robot interacts with the environment by processing service requests that can occur at various points in the future (again modeled with a Markov process). Figures 7.e and 7.f show two simulated executions under the implementation of the strategy that minimizes the expected time to reach the goal region while there are no outstanding requests. Other examples, which illustrate the breadth of the approach, appear in [72].

Algorithm improvements By making some restrictions of the problems considered, several improvements can be made to the dynamic programming algorithm. As it is formulated in Section 4, the entire state space is explored at each stage; however, in practice, only a small portion of the cost-to-go function actually changes in each iteration. Dijkstra's algorithm is able to find optimal paths by making a quick pass over the graph because the cost-to-go is guaranteed to be stabilized once a vertex is visited. A similar idea can be applied for continuous-state dynamic programming by identifying the "frontier" in the state space at which the loss values become stabilized [73]. For problems that involve uncertainty, this region of interest could be quite narrow or potentially large enough to span the entire space, depending on the problem. One might also be able to use admissible heuristics to preclude part of the space from consideration as in the A^*-search extension of Dijkstra's graph algorithm.

Even with such improvements, however, the time complexity would still be exponential in the dimension of the state space (or information space). In the case of basic path planning, randomized search algorithms

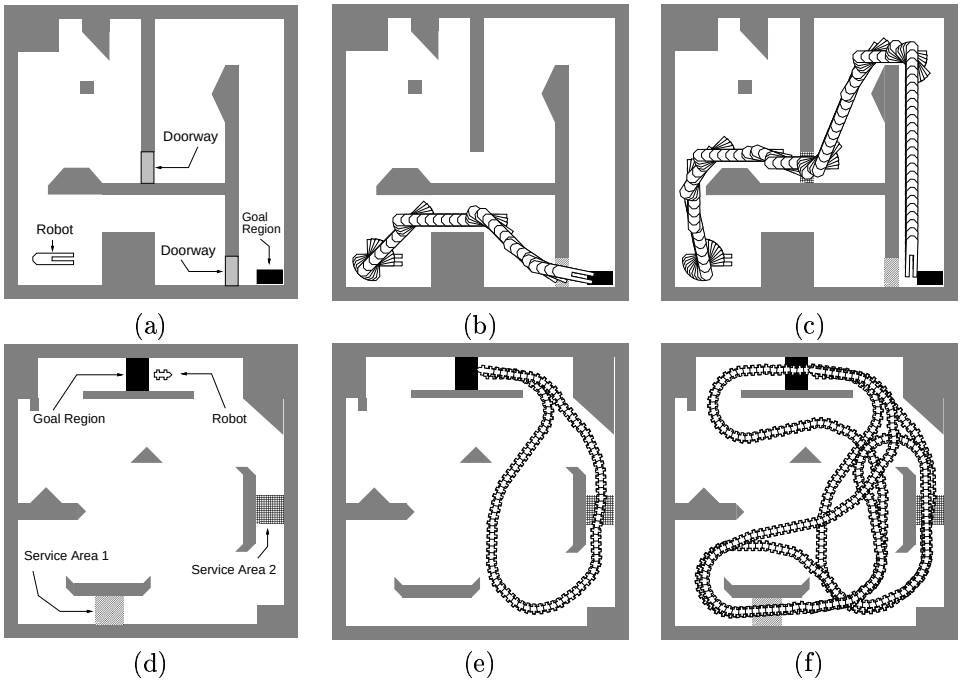


Figure 7: Optimal solutions are indicated for two problems that involve uncertainty in environment predictability.

have been developed that perform well in practice for many high degree-of-freedom problems by relying on the notion of probabilistic completeness [1, 9, 59, 106]. These planners are formulated for problems in which path optimality is not a central concern and in which there is perfect configuration predictability (i.e., the solutions are open-loop paths in the configuration space). One could also consider developing randomized search techniques using the formulation presented in Section 2 to obtain solution trajectories in the state space for nonholonomic, kinodynamic, or other problems in which there is perfect predictability and optimality is a not a central concern (or if solutions within some factor of optimal are acceptable). Following the general framework in this paper, a randomized data structure known as a Rapidly-exploring Random Tree (RRT) [74] has been developed recently for nonholonomic and kinodynamic path planning in high-dimensional state spaces [79].

Although the framework has been applied so far to several classes of problems, one important direction for future research will be to characterize and analyze additional problems. In many cases, useful concepts from the existing literature can be combined with the mathematical structure, such as in the case of using preimage planning research to develop the performance preimage. Such constructions are useful for developing algorithms, and are compatible with the dynamic game-theoretic concepts.

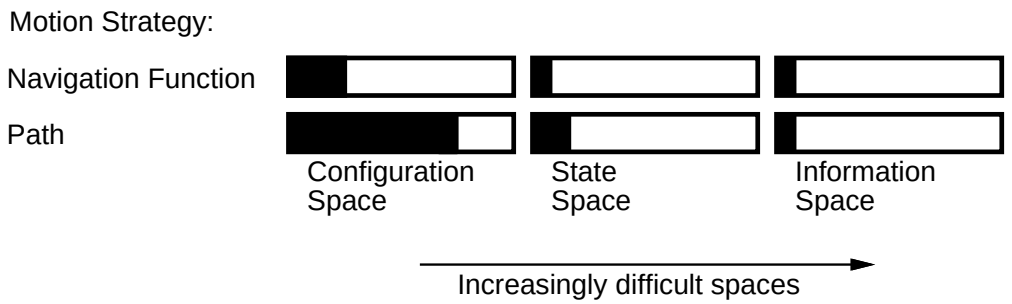


Figure 8: A taxonomy of motion strategy problems.

5 Conclusion

A dynamic game-theoretic framework has been proposed in this paper to serve as a mathematical foundation for a broad class of motion planning problems. Results obtained by following this perspective were presented with the intent of indicating the general utility of this foundation. By no means is it intended to provide a general solution to this broad class of problems, but instead it provides a useful characterization upon which motion strategy algorithms can be developed. In this way, it can serve the same purpose that configuration space concepts served for basic path planning problems.

This foundation can provide several key advantages for future research: (1) A common, unified structure facilitates the comparison of techniques. Just as configuration space concepts provided a precise, ideal formulation of basic path planning, the dynamic game-theoretic concepts provide a formulation of the ideal (or optimal) strategies that can be achieved. For many difficult problems, tradeoffs are inevitably made to improve computational performance. As approximate or incomplete methods are proposed, it is useful for the purposes of analysis to have precise, ideal formulations. (2) Clear directions are provided along which the concepts and methods can be generalized. For example, the preimage and forward projection concepts have been shown to apply in very general settings by generalizing their definitions within the framework. This has provided a clear relationship between nondeterministic and probabilistic uncertainty models, and numerical navigation functions and preimages. (3) A variety of different models can be incrementally tested. One of the greatest difficulties in determining motion strategies under uncertainties is determining appropriate models of uncertainty, while previous algorithms have often applied to very specific uncertainty models. The framework allows the substitutions of a variety of different models while many of the principles remain unchanged. This is particularly true of the algorithm discussed in Section 4, which makes few restrictions on the models.

Figure 8 shows a taxonomy of motion strategy problems that is based on the game-theoretic framework. Each bar indicates a particular class of problems. The black area depicts the author's subjective interpretation of the amount of progress to date relative to the size or importance of the problem. The bar at the lower left represents the path planning problem, which is the most basic in the taxonomy. Each column corresponds

to a different space, increasing in difficulty from left to right: the configuration space, the state space, and the information space. Each row represents a different solution concept, and computing a navigation function (or feedback motion strategy) is considered more difficult than computing a path. Most existing research on the motion strategy problem falls into the category of path planning in configuration space; however, it is hoped that eventually general algorithms are developed that further advance the state-of-the-art for some of the other challenging categories.

Acknowledgments

I thank Narendra Ahuja, Tamer Başar, Bruce Donald, Mike Erdmann, Ken Goldberg, Dan Koditschek, Vijay Kumar, Jean-Claude Latombe, Max Mintz, Jean Ponce, and Mark Spong, for their helpful comments and suggestions regarding this research. David Hsu and James Kuffner made helpful comments on this manuscript. I especially thank Seth Hutchinson and Rajeev Sharma, who each made significant contributions to some of the ideas described in this paper. This work was sponsored while the author was at the University of Illinois, sponsored by a Beckman Institute research assistantship, and a Mavis Fellowship, and also while the author was at Jean-Claude Latombe's research lab at Stanford University. The author is support in part by an NSF CAREER Award IRI-9875304.

References

[1] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 113–120, 1996.

[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] T. Başar and P. R. Kumar. On worst case design strategies. *Comput. Math. Applic.*, 13(1-3):239–245, 1987.

[5] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[6] J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 1235–1242, 1994.

[7] J. Barraquand and P. Ferbach. Motion planning with uncertainty: The information space approach. In *IEEE Int. Conf. Robot. & Autom.*, pages 1341–1348, 1995.

[8] 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.

[9] 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.

[10] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[11] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[12] D. P. Bertsekas. Convergence in discretization procedures in dynamic programming. *IEEE Trans. Autom. Control*, 20(3):415–419, June 1975.

[13] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[14] W. F. Bialas. Cooperative n-person Stackelberg games. In *IEEE Conf. Decision & Control*, pages 2439–2444, Tampa, FL, December 1989.

[15] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Trans. Robot. & Autom.*, 8(3):414–418, June 1992.

[16] D. Blackwell and M. A. Girshik. *Theory of Games and Statistical Decisions.* Dover Publications, New York, NY, 1979.

[17] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A computational framework. *Int. J. Robot. Res.*, 15(1):1–23, February 1996.

[18] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York, NY, 1975.

[19] S. J. Buckley. Fast motion planning for multiple moving robots. In *IEEE Int. Conf. Robot. & Autom.*, pages 322–326, 1989.

[20] L. G. Bushnell, D. M. Tilbury, and S. S. Sastry. Steering three-input nonholonomic systems: the fire truck example. *Int. J. Robot. Res.*, 14(4):366–381, 1995.

[21] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461–484, 1991.

[22] 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.

[23] J. F. Canny. On computability of fine motion plans. In *IEEE Int. Conf. Robot. & Autom.*, pages 177–182, 1989.

[24] C.-T. Chen. *Linear System Theory and Design.* Holt, Rinehart, and Winston, New York, NY, 1984.

[25] 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.

[26] H. W. Corley. Some multiple objective dynamic programs. *IEEE Trans. Autom. Control*, 30(12):1221–1222, December 1985.

[27] 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.

[28] M. H. DeGroot. *Optimal Statistical Decisions.* McGraw Hill, New York, NY, 1970.

[29] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall Publications, Englewood Cliffs, NJ, 1982.

[30] B. R. Donald. *Error Detection and Recovery for Robot Motion Planning with Uncertainty.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1987.

[31] B. R. Donald. Planning multi-step error detection and recovery strategies. *Int. J. Robot. Res.*, 9(1):3–60, 1990.

[32] B. R. Donald. On information invariants in robotics. *Artif. Intell.*, 72:217–304, 1995.

[33] 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.

[34] B. R. Donald and P. G. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open-chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

[35] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, November 1993.

[36] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, June 1989.

[37] M. Erdmann. Randomization in robot tasks. *Int. J. Robot. Res.*, 11(5):399–436, October 1992.

[38] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[39] M. Erdmann. Understanding action and sensing by designing action-based sensors. *Int. J. Robot. Res.*, 14(5):483–509, 1995.

[40] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Int. Conf. Robot. & Autom.*, pages 1419–1424, 1986.

[41] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[42] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1984.

[43] P. Ferbach. A method of progressive constraints for nonholonomic motion planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 2949–2955, 1996.

[44] 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.

[45] T. N. Gillespie. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, Warrendale, PA, 1992.

[46] 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.

[47] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[48] K. Y. Goldberg and M. T. Mason. Bayesian grasping. In *IEEE Int. Conf. Robot. & Autom.*, 1990.

[49] O. Hájek. *Pursuit Games*. Academic Press, New York, 1975.

[50] J. C. Harsanyi. Games with incomplete information played by Bayesian players. *Management Science*, 14(3):159–182, November 1967.

[51] 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.

[52] 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.

[53] H. Hu and M. Brady. A Bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots*, 1(1):69–92, 1994.

[54] 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.

[55] R. Isaacs. *Differential Games*. Wiley, New York, NY, 1965.

[56] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, Berlin, 1989.

[57] I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[58] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Robot. Res.*, 5(3):72–89, 1986.

[59] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for path planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 2138–2139, 1994.

[60] L. E. Kavraki. Computation of configuration-space obstacles using the Fast Fourier Transform. *IEEE Trans. Robot. & Autom.*, 11(3):408–413, 1995.

[61] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[62] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.

[63] K. H. Kim and F. W. Roush. *Team Theory.* Ellis Horwood Limited, Chichester, England, 1987.

[64] P. R. Kumar and P. Varaiya. *Stochastic Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1986.

[65] R. E. Larson. A survey of dynamic programming computational procedures. *IEEE Trans. Autom. Control*, 12(6):767–774, December 1967.

[66] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II.* Dekker, New York, NY, 1982.

[67] R. E. Larson and W. G. Keckler. Optimum adaptive control in an unknown environment. *IEEE Trans. Autom. Control*, 13(4):438–439, August 1968.

[68] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[69] J.-C. Latombe, A. Lazanas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artif. Intell.*, 52:1–47, 1991.

[70] 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.

[71] J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Plannning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[72] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[73] S. M. LaValle. Numerical computation of optimal navigation functions on a simplicial complex. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective.* A K Peters, Wellesley, MA, 1998.

[74] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Computer Science Dept., Iowa State University. <http://janowiec.cs.iastate.edu/papers/rrt.ps>, Oct. 1998.

[75] S. M. LaValle, H. H. González-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 731–736, 1997.

[76] 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.

[77] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. In *Proc. IEEE Int'l Conf. Robot. & and Autom.*, pages 2847–2852, April 1996.

[78] S. M. LaValle and S. A. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *International Journal of Robotics Research*, 17(1):19–42, January 1998.

[79] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1999.

[80] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 737–742, 1997.

[81] S. M. LaValle and R. Sharma. On motion planning in changing, partially-predictable environments. *International Journal of Robotics Research*, 16(6):775–805, December 1997.

[82] Z. Li and J. F. Canny. *Nonholonomic Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1993.

[83] C.-F. Lin and W.-H. Tsai. Motion planning for multiple robots with multi-mode operations via disjunctive graphs. *Robotica*, 9:393–408, 1990.

[84] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. on Comput.*, C-32(2):108–120, 1983.

[85] 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.

[86] 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.

[87] 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.

[88] M. T. Mason. Automatic planning of fine motions: Correctness and completeness. In *Proc. IEEE Int. Conf. Robot. & Autom.*, pages 492–503, 1984.

[89] 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.

[90] 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.

[91] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[92] G. Owen. *Game Theory.* Academic Press, New York, NY, 1982.

[93] C. H. Papadimitriou. An algorithm for shortest-path planning in three dimensions. *Information Processing Letters*, 20(5):259–263, 1985.

[94] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[95] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of IEEE Symp. on Foundat. of Comp. Sci.*, pages 421–427, 1979.

[96] 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.

[97] J. H. Reif and S. R. Tate. Continuous alternation: The complexity of pursuit in continuous domains. *Algorithmica*, 10:157–181, 1993.

[98] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. & Autom.*, 8(5):501–518, October 1992.

[99] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* Academic Press, New York, NY, 1985.

[100] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniqies for computing topological properties of algebraic manifolds. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[101] 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.

[102] R. Sharma, S. M. LaValle, and S. A. Hutchinson. Optimizing robot motion strategies for assembly with stochastic models of the assembly process. *IEEE Trans. on Robotics and Automation*, 12(2):160–174, April 1996.

[103] 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.

[104] A. Stentz. Optimal and efficient path planning for partially-known environments. In *IEEE Int. Conf. Robot. & Autom.*, pages 3310–3317, 1994.

[105] 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.

[106] P. Svestka and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *IEEE Int. Conf. Robot. & Autom.*, pages 1631–1636, 1995.

[107] 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.

[108] 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.

[109] Y. Yavin and M. Pachter. *Pursuit-Evasion Differential Games.* Pergamon Press, Oxford, England, 1987.

[110] J. Yong. On differential evasion games. *SIAM J. Control & Optimization*, 26(1):1–22, January 1988.

[111] J. Yong. On differential pursuit games. *SIAM J. Control & Optimization*, 26(2):478–495, March 1988.

[112] Q. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Trans. Robot. & Autom.*, 7(3):390–397, June 1991.

[113] 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.