On Motion Planning in Changing, Partially-Predictable Environments

Steven M. LaValle
(lavalle@cs.stanford.edu)
Dept. of Computer Science
Stanford University
Stanford, CA 94305

Rajeev Sharma
(rajeev@cs.uiuc.edu)
The Beckman Institute
University of Illinois
Urbana, IL 61801

Abstract

We present a framework for analyzing and computing motion plans for a robot that operates in an environment that both varies over time and is not completely predictable. We first classify sources of uncertainty in motion planning into four categories, and argue that the problems addressed in this paper belong to a fundamental category that has received little attention. We treat the changing environment in a flexible manner by combining traditional configuration space concepts with a Markov process that models the environment. For this context, we then propose the use of a *motion strategy*, which provides a motion command for the robot for each contingency that it could be confronted with. We allow the specification of a desired performance criterion, such as time or distance, and determine a motion strategy that is optimal with respect to that criterion.

We demonstrate the breadth of our framework by applying it to a variety of motion planning problems. Examples are computed for problems that involve a changing configuration space, hazardous regions and shelters, and processing of random service requests. To achieve this, we have exploited the powerful principle of optimality, which leads to a dynamic programming-based algorithm for determining optimal strategies. In addition, we present several extensions to the basic framework that incorporate additional concerns, such as sensing issues or changes in the geometry of the robot.

1 Introduction

Substantial interest in the field of robot motion planning has led to a variety of approaches that use different models of the robot and its environment (see [30, 37, 59] for surveys). For many problems, the success of a motion planning approach depends to a large extent on the manner in which various forms of uncertainty are modeled and treated. One important uncertainty source

There are two popular representations of uncertainty that have been applied to geometric motion planning problems. One representation restricts parameter uncertainties to lie within a specified set. A motion plan is then generated that is based on *worst-case analysis* (e.g., [9, 38, 43]). We refer to this representation as *bounded uncertainty.* The other popular representation expresses uncertainty in the form of a posterior probability density. This often leads to the construction of motion plans through *average-case analysis* (e.g., [14, 18, 23, 60])

Uncertainty can be introduced into a motion planning problem in a number of ways. We organize this uncertainty into four basic sources for discussion:

- *Uncertainty in configuration sensing* **(Type CS)**
- *Uncertainty in configuration predictability* **(Type CP)**
- *Uncertainty in environment sensing* **(Type ES)**
- *Uncertainty in environment predictability* **(Type EP)**

Figure 1 depicts the relationship between the sources.

We will now describe each of the sources of uncertainty, and the final source will be the primary focus of this paper. For the discussion, we will consider each type of uncertainty in isolation, although in general any combination of these types can be considered simultaneously in a motion planning formulation.

> **Type CS uncertainty.** Suppose that the space of collision-free configurations, $\mathcal{C}_{free}$ is given. Under uncertainty in configuration sensing, incomplete or imperfect information is utilized by the robot to make an inference about its configuration. This information could come from sensor measurements or motion history. With a nondeterministic uncertainty model, the robot might have sufficient information to infer that $\mathbf{q}$ lies in some subset $Q \subset \mathcal{C}_{free}$. For example, in [9], [17], [38], [43] this representation of uncertainty is used to guarantee that the robot recognizably terminates in a goal region. With a probabilistic model, the robot might infer a posterior probability density over configurations, $p(\mathbf{q})$, that is conditioned on sensor observations, initial conditions, or additional knowledge. Examples that handle

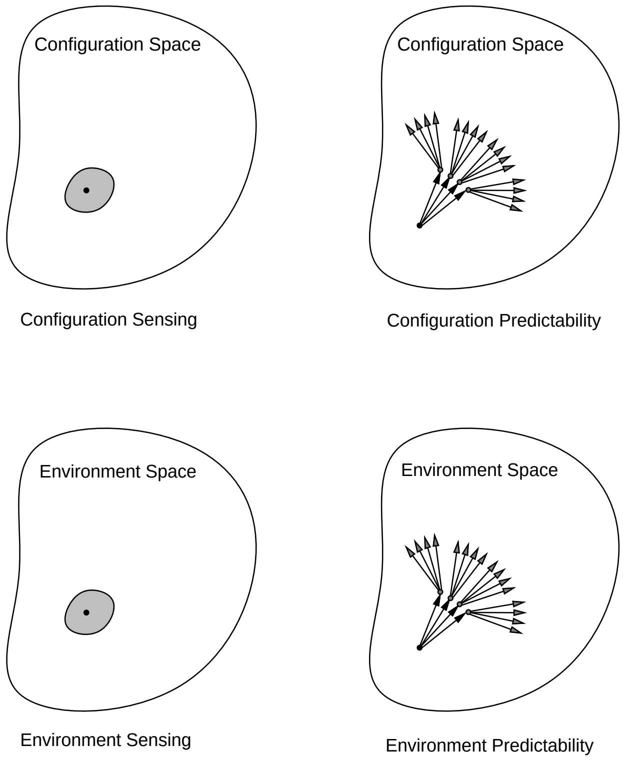


Figure 1: Four sources of uncertainty in the motion planning problem.

configuration-sensing uncertainty with probabilistic representations include [6], [71].

Type CP uncertainty. Suppose that both $\mathcal{C}_{free}$ and the current configuration, $\mathbf{q} \in \mathcal{C}_{free}$ are given. Motion commands can be given to the robot, but with Type CP uncertainty the future configurations cannot, in general, be completely predicted. With nondeterministic uncertainty, the robot may infer that some future configuration will belong to a subset $Q \subset \mathcal{C}_{free}$. The method of *preimage backchaining* constitutes of large body of work in which bounded uncertainties are propagated and combined with configuration-sensing uncertainty, to guarantee that the robot will achieve a goal (e.g., [9], [13], [17], [20], [38], [43]). With a probabilistic model, future configurations can be described by a posterior density over configurations, $p(\mathbf{q})$, that is conditioned on the initial configuration and the executed motion command. Examples of work that include a probabilistic representation of configuration-predictability uncertainty include [4], [6], [23].

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

Under Type ES uncertainty, incomplete or imperfect information is utilized by the robot to make an inference about its environment. With a nondeterministic uncertainty model, the robot might have sufficient information to infer that the environment e belongs to some subset $F \subset E$. For example, in [50], [53], the environment is restricted to a plane populated with unknown polygonal obstacles, which are then discovered using visual "scans" to build a visibility graph for motion planning. In [44], unknown obstacles are allowed to be of arbitrary shape, and the sensor data consists of "tactile" information for a point robot. With a probabilistic model, the robot might infer a posterior probability density, $p(e)$, over environments, which is conditioned on sensor observations, initial conditions, or additional knowledge (e.g., [14], [15], [28], [68]).

Type EP uncertainty. Suppose again that the space of environments, $\mathcal{E}$, is known by the robot; however, in addition, the robot knows its current environment $e \in \mathcal{E}$. Predictable motion commands might be given to the robot, but with Type EP uncertainty future environments cannot be completely predicted. With bounded uncertainty, the robot may infer that some future environment will belong to a subset $F \subset \mathcal{E}$. With a probabilistic model, future environments can be described by a posterior density over environments, $p(e)$, that can be conditioned on the initial environment, the robot configuration, or an executed motion command.

Type EP represents a fundamental source of uncertainty in robot motion planning that has received little attention. Previous approaches that have dealt with this difficult source have made limiting assumptions. In most cases, the robot is expected to react on-line to changes that occur in the environment. The past work is mainly concerned with local collision avoidance (e.g.,[75]) or incorporating unexpected moving objects locally into the updating of the motion plan. Thus, many of the predictive aspects of the changing environment is not utilized, and the approach is similar to the incorporation of Type ES uncertainty. For example, in the approach to motion planning with artificial potential fields it is possible to incorporate moving obstacles with unknown trajectories using on-line sensor data without considering future changes in the environment [3, 54, 65]. As another example, [12] presents an incremental planning scheme for collision avoidance with unknown moving obstacles that are restricted to linear trajectories.

In this paper a general framework is provided for Type EP uncertainty, when the environment is partially predictable. Motion strategies will be determined that provide specific motion commands that take into account environment changes that may occur at some time in the (possibly distant) future. Problems in which $\mathcal{E}$ is restricted to a finite number of environments are the primary focus in this paper. This covers a wide variety of motion planning problems, and leads to practical computational solutions. Our mathematical characterization of the problem, however, applies to more general environment spaces.

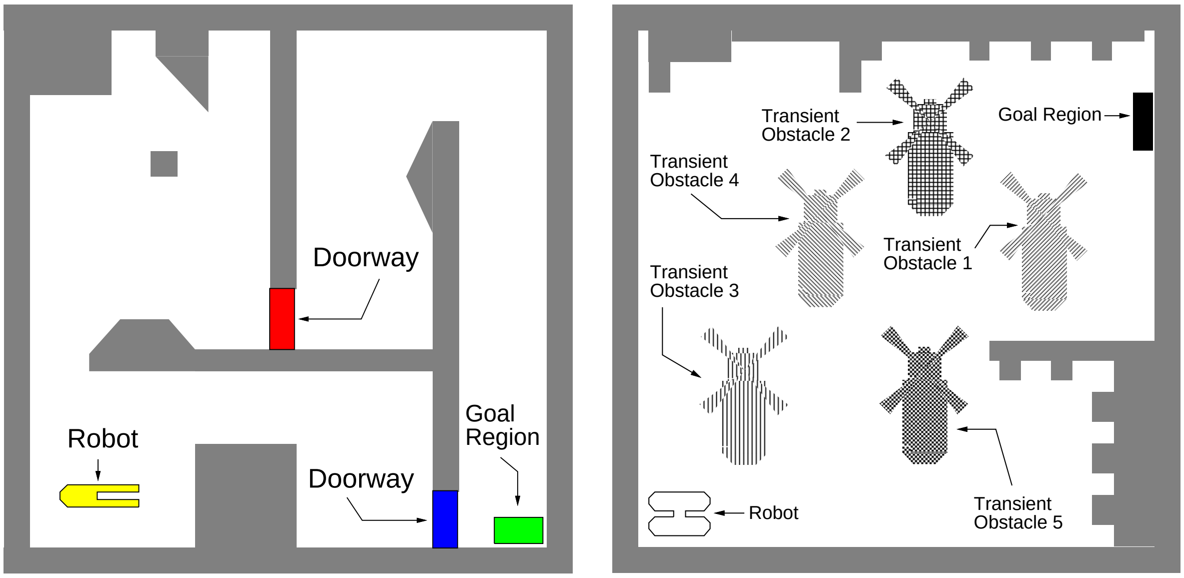
The rest of the paper is organized as follows. Section 2 provides motivation, and introduces classes of motion planning problems that involve Type EP uncertainty, which are addressed by this paper. Section 3 develops the mathematical model for the changing environment, the robot, and performance criteria. Section 4 explains the computational technique that determines optimal strategies. Section 5 presents computed examples that are obtained by applying the framework from Section 3 and the computation method from Section 4. Section 6 presents additional applications and extensions of our framework. Section 7 summarizes the key contributions of the paper, and provides a basis for future work.

2 Problem Description and Motivation

In this section we discuss three classes of motion planning problems that involve Type EP uncertainty (environment predictability), which is the main focus of this paper. This discussion provides motivation for the framework that is presented, by describing example problems. These examples will be referred to again when computed results are presented in Section 5.

We first consider the following motion planning problem in an environment depicted in Figure 2.a. While the robot is moving, "doors" that the robot has no control over could close or open. Suppose that the robot has bounded velocity, and wishes to reach a goal region in a minimal amount of time. Should the robot always try to move through the lower door? Should it adjust its path depending on which of the two doors are open? What happens when the robot is moving toward a door and the door closes? Should it just wait for it to open or should it head toward the other door if that door is open? One would like to define a formal basis for deciding on the best actions to take, given that the robot does not know exactly when certain changes will occur in the workspace. Note that these types of questions will exist even if the robot has perfect information at a given time regarding the status of the doors. We will refer to problems such as this as the class of *changing configuration space* problems. To the best of our knowledge there is no formal treatment of this form of uncertainty in the literature. By using the framework presented in this paper, we can analyze and determine a solution to a problem of this type.

Motion planning problems for which the changing configuration space is completely predictable have been previously considered in motion planning. However, in even the simplest cases of moving obstacles, the computational complexity of the motion planning a problem becomes prohibitive, as demonstrated by various complexity results reported in the literature. In [55], for example, Reif and Sharir show that motion planning in a three-dimensional environment is PSPACE-hard when the robot's velocity is bounded, and NP-hard when there is no such bound. In [8] Canny and Reif prove the NP-hardness of a more restricted problem—that of motion planning for a point robot in the plane with bounded velocity, when the moving obstacles are convex polygons moving at constant linear velocities without rotation. These intractability results have encouraged heuristic approaches, especially as applied to more limited geometric models—for example, in [16, 22, 32, 64]. We model problems in which the changing environment is not completely predictable; however, we only consider a finite number of possibilities for the environment.



(a) (b)

Figure 2: A changing environment in which the workspace changes over time, by (a) the opening and closing of "doors," and (b) appearance and disappearance of "transient" obstacles.

While accounting for the changing environment, problems such as the one in Figure 2.a must be described geometrically. A standard way of describing a geometric motion planning problem of a robot $\mathcal{A}$ in a workspace $\mathcal{W}$ is in terms of its n-dimensional *configuration space*, $\mathcal{C}$, in which n is the number of degrees of freedom of $\mathcal{A}$ [37]. In terms of the configuration space, the robot is represented as a point. If $\mathcal{W}$ contains a set of fixed obstacles, $\{\mathcal{B}_1, \ldots, \mathcal{B}_q\}$, the goal of a traditional motion planning algorithm is to find a path, from an initial configuration to a goal configuration, whose image lies in $\mathcal{C}_{free}$ (or C_{valid} [37]).

A problem such as that in Figure 2.a can be described with a discrete set of collision-free configuration spaces. At a given time, the robot is in one of these spaces, and the environment can cause the robot to move to a different collision-free configuration space. Another example of a changing configuration space is shown in Figure 2.b, and it contains transient obstacles.

In [21] transient obstacles were introduced for a motion planning problem in which polygonal obstacles exist only for a given period of time, and an efficient algorithm was proposed to determine reasonable solutions when the transient obstacles are completely predictable. Since the obstacles appear and disappear at the same position, the motion planning problem can again be described in terms of a discrete set of collision-free configuration spaces.

In classical motion planning approaches, the output of algorithm is usually a "motion plan" for a given description of the $\mathcal{C}$, the initial and the goal positions. When unpredictable changes occur in the workspace, *dynamic replanning* is often used. This has been used, for example, in the context of error-detection and recovery [13, 46], and task-level reasoning [19, 27]. Alternatively, a fixed command might be given to the robot, and local collision avoidance is performed to handle unexpected aspects of the environment [3, 54, 58, 65, 75]. In the probabilistic framework that we propose, a "motion strategy" provides a motion command for the robot for each contingency that it could be confronted with. This motion strategy can be considered as a state-feedback stochastic controller [35], on a state space that simultaneously considers the environment and the robot configuration. Replanning is not needed when the environment changes, because the robot responds appropriately for different regions of the state space during execution. In addition, a state-feedback controller is advantageous, since it will typically be robust to small modeling errors that can develop during execution. To select a motion strategy, we formulate an explicit performance criterion (or loss functional) that evaluates a trajectory executed by the robot. This allows a variety items, such as time or distance, to be optimized through the selection of a strategy.

In general, the changing environment is characterized with a stochastic model. Consider developing a model for the above motion planning problem with doors; one suitable means of modeling the opening of a door could be in terms of a Poisson process with arrival frequency λ_o, so that the time until a closed door opens is an exponential random variable with mean λ_o. Similarly, the closing of a door, given that it is open could be modeled as a Poisson process with arrival frequency λ_c. The behavior of a door is thus characterized by two parameters, which eases the specification of a model. These parameters could in practice be estimated by observing a many samples of such changes in a particular workspace.

The Poisson process is a reasonable choice for many problems because it captures several realistic properties of a changing environment: (i) The probabilities that a door will open in two non-overlapping time intervals are independent of each other; (ii) The probability that a door will open in an interval is proportional to the length of the interval. This implies that events are uniformly distributed in time, and thus do not favor one epoch of time over another. (iii) The probability that a door will open in an interval becomes arbitrarily small if the interval is made sufficiently small. These Poisson processes can be considered as special cases of a general Markov process has much greater expressive power, and will be further developed in Sections 3 and 5.

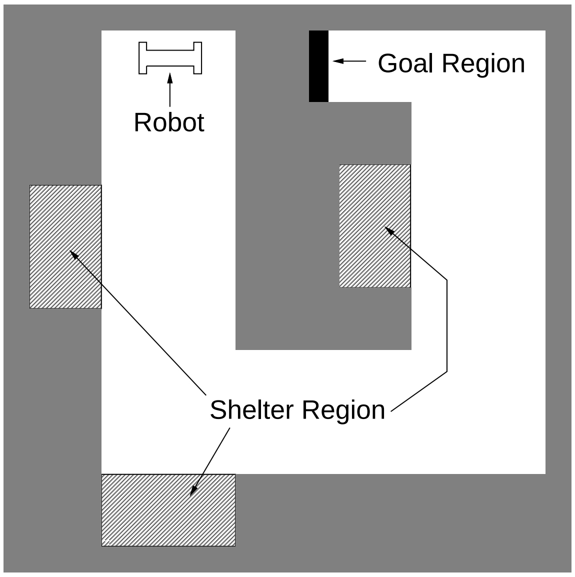


Figure 3: A problem that involves safe and hazardous regions in addition to obstacles.

We can also move beyond the concept of a changing configuration space to model new types of motion planning problems. For instance, in some situations, it may be appropriate to *not consider* the individual moving obstacles in the environment, and instead associate a cost of traversing a region that could have moving obstacles. This leads to a class of problems that we refer to as *hazardous region and shelter problems.* The cost could, for instance, directly correspond to the risk involved in traversing a hazardous region. In a similar context, this has been referred to as a *weighted region problem* [31, 47, 57]. Assigning a cost associated with the traversal of a region provides a way of dealing with the complexity of motion planning in an environment that has several moving obstacles, particularly when their motion is unknown. Similar treatment of dynamic environments in [60, 61] led to the idea of *shelters* (regions which have low cost of traversal) and *alarms* (events that cause the cost of traversing a region to change from low to high). The treatment considered in this work, however, is substantially more general.

As an example of a hazardous region and shelter problem, consider the motion planning problem shown in Figure 3, for a mobile robot in a factory floor in which there might be other moving robots, vehicles, or people in the corridors. If the robot receives a signal that indicates that there are other motions in the corridor, it can be considered to be at risk of collision. Such a signal could be transmitted via a wireless modem from a set of motion detectors or from other robots. *Shelters* designate areas in which the robot is guaranteed to avoid collisions. These shelters become relevant only when the environment is hazardous, and appropriate motion commands must be given to the robot that take into account the possibility of a receiving a signal that indicates a hazard. By proper modeling of hazardous regions and shelters in the workspace, the need for explicitly considering multiple moving objects (as in [8, 16, 55, 70]) can be avoided, while implicitly factoring in the effect of moving obstacles in the determination of motion plans.

As it becomes clear from the previous problem, simple geometric changes in the configuration space may not be sufficient to capture all the desired features for motion planning in a changing, partially-predictable environment. One expanded concept, that we will formalize in the next section is that of an *environment mode.* The changing status of the environment (and not just of the geometry of the configuration space) can then be described in terms of a finite set of environment modes. Uncertainty is represented in terms of transition probabilities between the different possible environment modes.

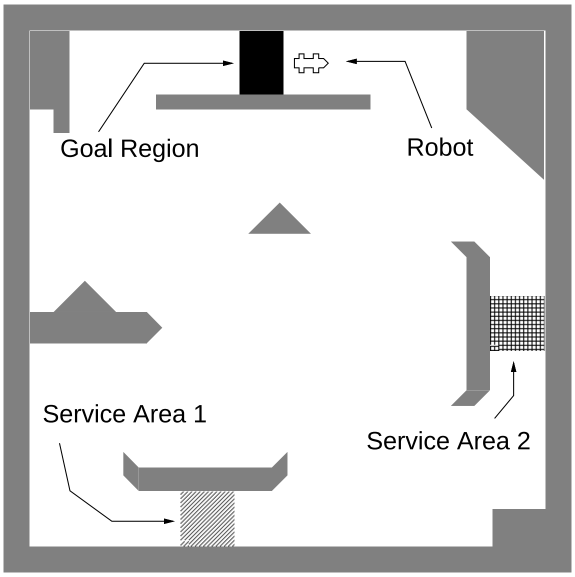


Figure 4: A problem of processing random service requests in the workspace.

The concept of environment modes permits us to describe motion planning problems that have a greater scope than just the problem of moving from a initial position to a goal region. For example, the robot could influence the changes in the environment mode by being at particular points in its configuration space. We describe a class of problems that we call *servicing problems*, which involve interaction between the robot and the environment. Consider, for example the environment shown in Figure 4, which contains service areas 1 and 2. Whenever there is a request for one type of service, the robot should head toward that service area and process the service request. In practice, this could correspond to mobile robot that performs tasks in an office or other work area, and receives requests to perform services directly through a radio ethernet connection. We can define four environment modes, which correspond to the combinations of the two different types of service requests. One can associate a cost for the waiting time before a request is processed. Additionally, the robot directly affects the environment by causing the mode to change once the robot arrives at a service area. Our framework can model interactions of this type, between the robot and the environment, which occur during motion planning.

The purpose of this background discussion has been to motivate the need for a general framework for motion planning in changing, partially-predictable environments (with Type EP uncertainty). The next few sections develop the details of the framework and show its applicability to the examples discussed in this section. The implementation of the algorithm that determines optimal strategies, and the results presented in Section 5 further demonstrate the utility of our approach.

3 Mathematical Formulation

In this section we develop the mathematical concepts that model the problems discussed in Section 2. Section 3.1 introduces the finite-state Markov process that is used to model the changing environment, and the relationship of this model to the configuration space of the robot. In Section 3.2 we define a model of robot motion, which accepts a motion command and produces a next configuration. A fixed behavior for the robot is specified in the form of a strategy. In contrast to the traditional specification of a planned path, a strategy specifies the motion that will be executed by the robot for any given combination of robot configuration and environment mode. This allows different trajectories to be executed by the robot under different environment modes.

Section 3.3 introduces the concept of dynamic regions in the robot's configuration space. These regions are used to explicitly define interaction that occurs with the environment and the robot, which is effected through the definition of a performance criterion. The performance criterion (termed a loss functional) evaluates the executed trajectory of the robot. This criterion could, for instance, could be chosen to measure the total distance traveled by the robot. Since the environment is a stochastic process, the actual trajectory taken by the robot under a given strategy is also a stochastic process. Therefore the goal is to determine a strategy for the robot that is optimal in an *expected sense*.

3.1 The Environment Process

For geometric motion planning problems without uncertainty, the space of possible situations that can occur is sufficiently characterized by $\mathcal{C}_{free}$ (or $\mathcal{C}_{valid}$ [37]). In our context, the environment can additionally interfere with the motion plans of $\mathcal{A}$, compelling us to define a finite set, E, of *environment modes.*

Since we are modeling problems in which the environment changes, we require an explicit representation of time in many subsequent definitions. We define a discretized representation of time by a set of *stages*, with an index $k \in \{1, 2, \ldots, K\}$. Stage k refers to time $(k-1)\Delta t$. At a given stage, k, the environment is in some mode $e_k \in E$, which is known to the robot. We generally take Δt sufficiently small to approximate continuous trajectories. This appropriately reflects a situation in which a real robot is limited to some sampling rate for acquiring sensor information and executing motion commands. A stage-limit, K, is defined only to preclude a special treatment of infinite stages. In practice, an explicit choice of K does not need to be determined, which will be discussed in Section 4.

We additionally consider the environment as a finite-state Markov process, which we call the *environment process*. We consider a Markov process since the representation is powerful enough to encode many important stochastic processes, such as a Wiener process (Brownian motion) or a Poisson process [67]. As another example of using a Markov model in the analysis of motion planning, see [75]. At the initial stage ($k = 1$) the environment mode, $e_1 \in E$, is given. For a given environment mode, e_k, the next environment mode, e_{k+1}, is specified with a probability distribution over E. This probability distribution is defined by a vector P_i such that $P_i[j] = P(e_{k+1} = j | e_k = i)$.

We now present an example of a four-mode environment process that can model the problem from Figure 2.a. More general models and examples are presented in Section 5. We define the following four environment states:

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

.

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

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

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

The probability that it will stay closed is $P_{11} = 1 - P_{01}$. For a door that is initially open, we similarly obtain $P_{10} = 1 - e^{-\lambda_c \Delta t}$, and $P_{00} = 1 - P_{10}$.

The environment transition probabilities can be generated by taking products of pairs of P_{00},

P_{01}, P_{10}, and P_{11}:

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} P_{00}^2 & P_{00}P_{10} & P_{10}P_{00} & P_{10}^2 \\ P_{00}P_{01} & P_{00}P_{11} & P_{10}P_{01} & P_{10}P_{11} \\ P_{01}P_{00} & P_{01}P_{10} & P_{11}P_{00} & P_{11}P_{10} \\ P_{01}^2 & P_{01}P_{11} & P_{11}P_{01} & P_{11}^2 \end{bmatrix}. \tag{2}$$

The four-mode process is depicted in Figure 5, in which we take $\lambda_o = \lambda_c = 0.10101354$ (approximately one expected arrival every ten seconds), and $\Delta t = 0.2$. This results in $P_{10} = P_{01} = 0.02$ and $P_{00} = P_{11} = 0.98$.

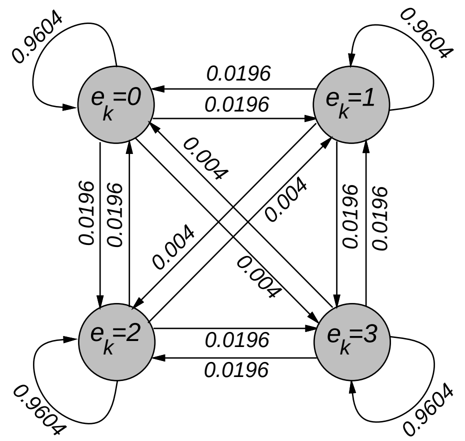


Figure 5: The environment process can be considered as a finite-state Markov process with state transition probabilities.

For this example, the environment process is independent of the robot configuration. In general, however, we allow the robot to have influence over the environment by conditioning the probabilities on the configuration of the robot. The servicing problem, discussed in Section 2, is an example in which this extension is needed.

In general, to uniquely identify all of the possible situations that can occur in our problem, we define a *state space* as the cartesian product, $X = \mathcal{C}_{free} \times E$. This is similar to the view taken in [13], in which the space for motion planning is a cartesian product of $\mathcal{C}_{free}$ with a single parameter that characterizes a hole width for a peg-in-hole task. The state at stage k is denoted by x_k, which simultaneously represents both a configuration of $\mathcal{A}$ in the geometric sense, and an environment mode, e_k. The environment modes form a partition of the state space, X. Each time the environment mode changes, the robot is forced into a different layer of X (see Figure 6).

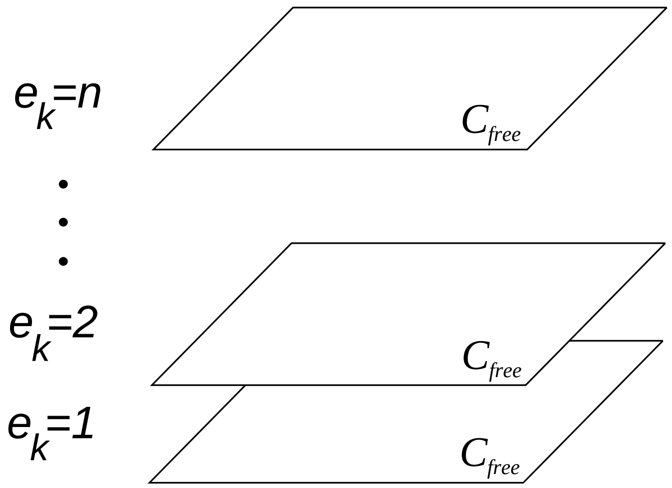


Figure 6: The environment modes can form a partition of X.

3.2 Defining the Robot Behavior

In this section we present several concepts that lead to the definition of a strategy, which characterizes a fixed behavior for the robot. We begin by defining an *action*, u_k, (or command), which can be issued to $\mathcal{A}$ at each stage, k. We let U denote the *action space* for $\mathcal{A}$, while requiring that $u_k \in U$. We define a *state transition distribution* as $P(x_{k+1}|x_k, u_k)$. This represents a probability distribution over a finite set of next states, given x_k as the initial state, and an action u_k. This relationship is probabilistic because the final component of the state vector (which corresponds to the environment mode) cannot be completely predicted. Since the remaining components of the state space correspond to the configuration space of the robot, we assume that these can be predicted once x_k and u_k are given. This implies that we have perfect control of the robot (i.e., the response of the robot to a given command is assumed to be exactly executed by a deterministic relationship). In addition, the use of x_k in the conditional of the state transition distribution implies that the robot has perfect information regarding its state. In other words, the CS, CP, and ES forms of uncertainty, as discussed in Section 1, are suppressed. Sections 6.1 and 7, however, indicate how these forms of uncertainty could be incorporated.

We present a state transition distribution that applies to the case in which $\mathcal{C} \subseteq \Re^2$, and the robot is limited to translational motion. More complicated motions, which apply to the examples in Section 5, are presented in Appendix A. Dynamic robot constraints could also be introduced; however, $\mathcal{C}_{free}$ would have to be replaced by its tangent bundle in the state space definition, and the robot constraints would have to be specified in state-space form. We define the action space as $U = [0, 2\pi) \cup \{\emptyset\}$. If $u_k \in [0, 2\pi)$, then $\mathcal{A}$ attempts to move a distance $\|v\|\Delta t$ toward a direction in $\mathcal{C}$, in which $\|v\|$ denotes some fixed speed for $\mathcal{A}$. If $u_k = \emptyset$, then the robot remains motionless.

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

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\|\Delta t \; cos(u_k) \\ x_k[2] + \|v\|\Delta t \; sin(u_k) \\ e_{k+1} \end{bmatrix}, \qquad (3)$$

in which the environment mode e_{k+1} is known to be sampled from $P(e_{k+1}|x_k, u_k)$. We can thus consider a finite-valued random variable X_{k+1} with corresponding distribution $P(x_{k+1}|x_k, u_k)$, which can be inferred from the given model. If $u_k = \emptyset$, then $x_k[1] = x_{k+1}[1]$ and $x_k[2] = x_{k+1}[2]$; however, e_{k+1} is not necessarily equal to e_k because the environment transition equation determines e_{k+1}. We prohibit the robot from considering actions that produce an obstacle collision; however, one could also consider compliant or constrained motions [40, 45, 51, 73].

We now define the notion of a robot strategy for our context. At first it might seem appropriate to define some action u_k for each stage; however, we want a motion plan that is prepared for the various contingencies presented by the changing environment. Therefore, we define a *strategy at stage* k of $\mathcal{A}$ as a function $\gamma_k : X \to U$. For each state, x_k, the function γ_k yields an action $u_k = \gamma_k(x_k)$. The set of mappings $\{\gamma_1, \gamma_2, \ldots, \gamma_K\}$ is denoted by γ and termed a *strategy.* This is equivalent to a control law or policy in control theory [35]. For the examples that we present in this paper, γ_k will be the same for all k (i.e., each robot action depends only on the current state, and not the particular stage). Section 6.2 presents a discussion of time varying strategies, in which this assumption is relaxed.

We can represent a desired performance criterion by a nonnegative real-valued functional $L(x_1, ..., x_{K+1}, u_1, ..., u_K)$, called the *loss functional.* A strategy that produces a lower loss will be considered preferable. The ultimate goal of the planner is to determine an optimal strategy $\gamma^* = \{\gamma_1^*, \gamma_2^*, \ldots, \gamma_K^*\}$ that causes L to be minimized in an expected sense.

We use the notation $w(\gamma, x_1, \mathbf{e})$ to refer to the path taken through the state space by the implementation of γ, an initial state, x_1, and a given environment mode sequence $\mathbf{e} = \{e_1, e_2, \ldots, e_K\}$. We refer to $w(\gamma, x_1, \mathbf{e})$ as a *sample path* for γ (given x_1 and $\mathbf{e}$). We also define $W(\gamma, x_1)$, which is a random process that takes on values of sample paths once $\mathbf{e}$ is known. Note that the probability distribution of $\mathbf{e}$ can be directly determined from γ, x_1, and the environment transition distribution; therefore, the probability distribution over the sample paths (which defines $W(\gamma, x_1)$) is known.

3.3 Defining Performance with Dynamic Regions

Sections 3.1 and 3.2 have introduced the environment process and a model of the robot behavior. This section discusses the key concepts that are used to model the effect that the environment

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

has on the robot. In particular, costs that appear in a loss functional directly depend on dynamic regions in the state space. If the robot enters a particular dynamic region, the amount of loss received might increase or decrease. For instance, a dynamic region might correspond to the robot's collision with a closed door, which would incur a very high loss.

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

A dynamic region $\mathcal{D}_i$ in $\mathcal{W}$ can map into the region $\mathcal{CD}_i^c \subset \mathcal{C}_{free}$, which is given by (see Figure 7):

$$\mathcal{CD}_i^c = \{\mathbf{q} \in \mathcal{C}_{free} | \ \mathcal{A}(\mathbf{q}) \cap \mathcal{D}_i \neq \emptyset\}. \tag{4}$$

We call $\mathcal{CD}_i^c$ a *contact (dynamic) $\mathcal{C}$-region.* This yields configurations in which the robot is in contact with $\mathcal{D}_i$. A contact $\mathcal{C}$-region is useful for problems such as that in Figure 2a, in which contact with an obstacle must be determined.

In some situations, we will be interested in determining whether the robot is completely contained in some $\mathcal{D}_i$. For instance, in the example in Figure 3, the robot is only considered safe if it is completely inside the shelter region. For this situation, a dynamic region $\mathcal{D}_i$ in $\mathcal{W}$ maps into the region $\mathcal{CD}_i^e \subset \mathcal{C}_{free}$, which is given by (see Figure 8):

$$\mathcal{CD}_i^e = \{\mathbf{q} \in \mathcal{C}_{free} | \ \mathcal{A}(\mathbf{q}) \subseteq \mathcal{D}_i\}. \tag{5}$$

We call $\mathcal{CD}_i^e$ an *enclosure (dynamic) $\mathcal{C}$-region.* One could alternatively define $\mathcal{CD}_i^e$ as the subset of configuration space in which the robot and dynamic region interiors overlap, and also the containment in the definition of $\mathcal{CD}_i^e$ could be made strict. Note that $\mathcal{CD}^e \subseteq \mathcal{CD}^c$.

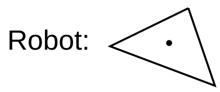
We now want to map the dynamic regions into the state space, since the loss functional depends on the state trajectory. Recall Figure 6. Since the dynamic regions have been mapped into $\mathcal{C}_{free}$, the mapping into X can be considered as lifting the contact $\mathcal{C}$-region (or enclosure $\mathcal{C}$-region) into different layers of X. We want the dynamic region to only influence the robot at certain layers. For instance, with the example in Figure 2.a, we only want the dynamic region to exist in X in environment modes that correspond to the door being closed. In other modes, the door should not interfere with the robot. For each $i \in \{1, \ldots, m\}$ we select a subset, E_i, of environment states, $E_i \subseteq E$.

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

$$X_i = \{(\mathbf{q}, e) \in X | \ q \in \mathcal{CD}_i^c \text{ and } e \in E_i\}. \tag{6}$$

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

$$X_i = \{(\mathbf{q}, e) \in X | \ q \in \mathcal{CD}_i^e \text{ and } e \in E_i\}. \tag{7}$$



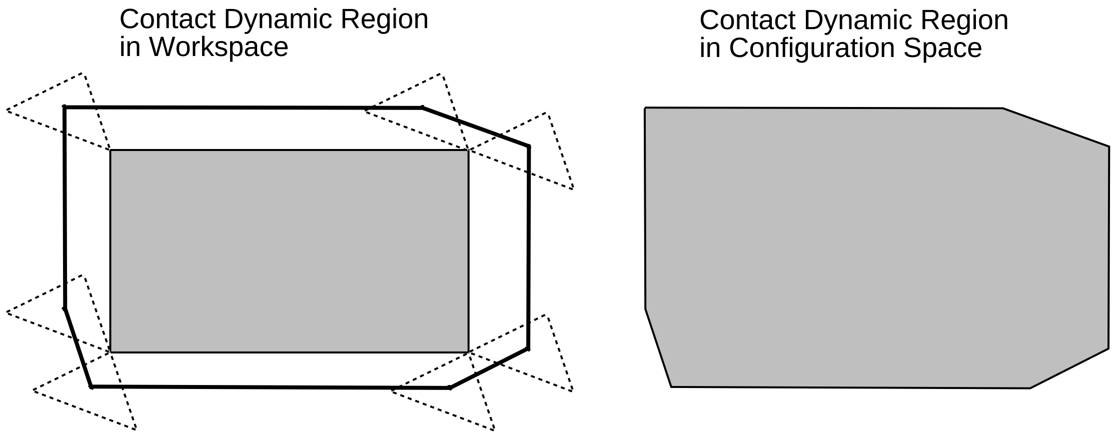


Figure 7: A contact dynamic region.

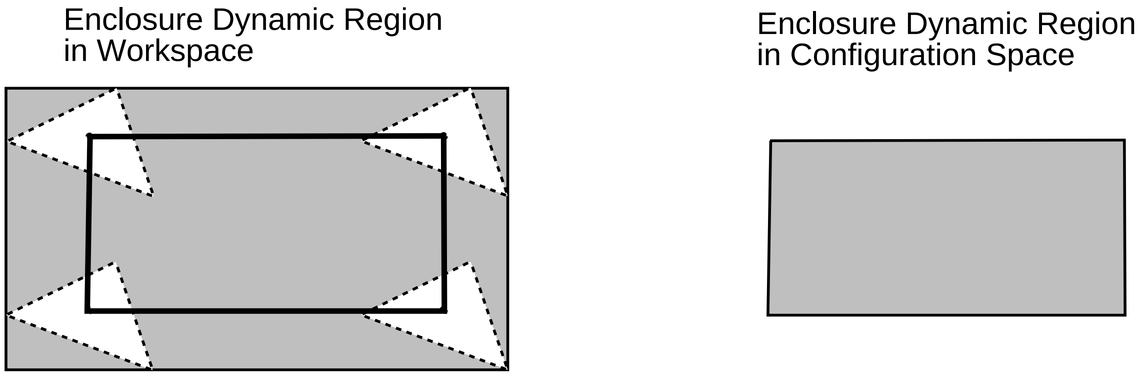


Figure 8: An enclosure dynamic region.

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

We now define a *goal region* as a special kind of dynamic region (in addition to $\mathcal{D}_i, i \in \{1, \ldots, m\}$). We first define a subset $G \subseteq \mathcal{W}$ as a goal region in the workspace. We next consider G as a *contact goal region* (or *enclosure goal region*), and apply (4) (or (5)) with $\mathcal{D}_i = G$ to obtain the goal region in configuration space. A subset $E_g \subseteq E$ is selected, and we obtain X_G by applying (6) (or (7)). The termination condition for a given motion planning problem will be to bring the system to any state in X_G.

We next present a general loss functional. Although this form will not be the most general that is possible, it is sufficient to encompass the examples that are discussed in this paper.

Specific applications of this loss functional are presented in Section 5.

We assume that a loss functional is of the following additive form, which is often used in optimal control theory [35]:

$$L(x_1, ...x_{K+1}, u_1, ..., u_K) = \sum_{k=1}^{K} l_k(x_k, u_k) + l_{K+1}(x_{K+1}) \tag{8}$$

The first K terms correspond to costs that are received at each step during the execution of the strategy. The final term, l_{K+1}, is a terminal cost that will be used to ensure that the robot achieves the goal (if the goal is reachable).

For a given set A, let I_A denote its characteristic function: $I_A(a) = 1$ if $a \in A$, and $I_A(a) = 0$ otherwise. The term l_k in (8) is defined as

$$l_k(x_k, u_k) = \begin{cases} 0 & \text{If } x_k \in X_G \\ & \\ c_u + \sum_{i=1}^{m} [c_i I_{X_i}(x_k) + c_i' I_{X_i^c}(x_k)] & \text{Otherwise} \end{cases} . \tag{9}$$

The constant $c_u \geq 0$ corresponds to the cost for choosing an action. This cost will often be the same for every $u_k \in U$, but in general can be dependent on the particular action. For instance, if time optimality is considered, then $c_u = \Delta t$. However, if distance optimality is considered, then one might choose $c_u = 0$ if $u_k = \emptyset$, and $c_u = \|v\|\Delta t$ otherwise.

The constant $c_i \geq 0$ is a penalty that is added if $x_k \in X_i$. The constant $c_i' \geq 0$ is a penalty that is added if $x_k \notin X_i$. In (9), X_i^c denotes $X \setminus X_i$. For the case of a changing configuration space, for instance, these constants could become $c_i = \infty$, to indicate that a collision has occurred, and $c_i' = 0$ otherwise. The specific loss functionals for applications are presented in Section 5.

The term l_{K+1} in (8) is defined as

$$l_{K+1}(x_{K+1}) = c_f I_{X_G^c}(x_{K+1}), \tag{10}$$

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

4 Determining Optimal Strategies

One of the primary advantages of our framework is that a straightforward computation procedure can be used to determine optimal strategies. In Section 4.1 we show how the principle of optimality can be applied to our problem to obtain solutions through dynamic programming. Section 4.2 briefly discusses computational issues.

4.1 Applying the Principle of Optimality

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

$$\bar{L}_k^*(x_k) = E\left\{\sum_{i=k}^{K} l_i(x_i, \gamma_i^*(x_i)) + l_{K+1}(x_{K+1})\right\}, \tag{11}$$

in which $E\{\}$ denotes expectation. The expectation is taken over the possible environment sequences, $\mathbf{e}$. The function $\bar{L}_k^*(x_k)$ is sometimes referred to as the *cost-to-go* function in dynamic optimization literature [5].

The principle of optimality [35] states that $\bar{L}_k^*(x_k)$ can be obtained from $\bar{L}_{k+1}^*(x_{k+1})$ by the following recurrence:

$$\bar{L}_k^*(x_k) = \min_{u_k}\left\{ l_k(x_k, u_k) + \sum_{x_{k+1}} \bar{L}_{k+1}^*(x_{k+1}) P(x_{k+1}|x_k, u_k)\right\}. \tag{12}$$

Note that the sum in (12) is taken over a finite number of states, which can be reached using (3).

Suppose that the goal is to determine the optimal action, u_k, for every value of x_k, and every stage $k \in \{1, \ldots, K\}$. One can begin with stage $K+1$, and repeatedly apply (12) to obtain the optimal actions. At stage $K+1$, we can use the last term of (8) to obtain $\bar{L}_{K+1}^*(x_{K+1}) = l_{K+1}(x_{K+1})$. The cost-to-go, $\bar{L}_K^*$, can be determined from $\bar{L}_{K+1}^*$ through (12). Using the $u_K \in U$ that minimizes (12) at x_K, we define $\gamma_K^*(x_K) = u_K$. We then apply (12) again, using $\bar{L}_K^*$ to obtain $\bar{L}_{K-1}^*$ and γ_{K-1}^*. These iterations continue until $k = 1$. Finally, we take $\gamma^* = \{\gamma_1^*, \ldots, \gamma_K^*\}$.

The loss function $\bar{L}_1^*(x_1)$ shares similarities with the concept of a global navigation function in motion planning [37, 56]. Also, various forms of dynamic programming have been successfully applied in several other motion planning contexts [2, 29, 48, 69]; for instance, the *wavefront expansion method* that is described in [37] can be viewed as a special form of dynamic programming.

It turns out that the optimal action u_k, does not depend on the stage index, k, for the problems that we consider (see Section 6.2 for a discussion of an extension that includes stage dependency in the optimal strategy). In addition, we do not need to choose K. These points are discussed in Section 4.2.

4.2 Computational Issues

We determine optimal strategies numerically, by successively building approximate representations of $\bar{L}_k^*$. This offers flexibility, since analytical solutions are very difficult to obtain, and have only been previously obtained by considering very specific cases [61, 60]. Each dynamic programming iteration can be considered as the construction of an approximate representation of $\bar{L}^*_k$. We decompose the state space into cells of uniform size; however, it is important to note the differences between the use of this decomposition in our context and the traditional use of decompositions in geometric motion planning (see, for example, [37]). Our primary interest in using the decomposition is to construct a good approximation of the function $\bar{L}^*_k$.

We obtain the value for $\bar{L}^*_k(x_k)$ by computing the right side of (12) for various values of u_k, including $u_k = \emptyset$. The value for $\bar{L}^*_k(x_k)$ is obtained by linear interpolation (see Figure 9). Other schemes, such as quadratic interpolation, can be used to improve numerical accuracy [36].

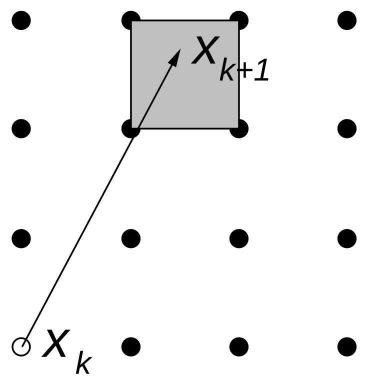


Figure 9: Interpolation is performed on the shaded region to obtain a more accurate value for $\bar{L}^*_{k+1}$.

Note that $\bar{L}^*_K$ represents the cost of the optimal one-stage strategy from each state x_K. More generally, $\bar{L}^*_{K-i}$ represents the cost of the optimal $i+1$-stage strategy from each state x_{K-i}. For a motion planning problem, we are only concerned with strategies that require a finite number of stages, before terminating in the goal region. For each position in the state space, one of the following occurs after some finite number of iterations: (i) The state, x_k, is in the goal region, in which case $\bar{L}^*_k(x_k) = 0$; (ii) The losses $\bar{L}^*_k(x_k)$ and $\bar{L}^*_{k+1}(x_{k+1})$ become equal for $x_k = x_{k+1}$; (iii) The loss $\bar{L}^*_k(x_k)$ continues to be greater than $\bar{L}^*_{k+1}(x_{k+1})$ for $x_k = x_{k+1}$. The second condition occurs when the optimal strategy from x_{k+1} has already been completely determined, and an additional stage accomplishes nothing (this additional stage can the considered as transpiring in the goal region, in which no additional loss is received). The third condition occurs when the goal cannot be reached from x_{k+1}. If we continue to perform the dynamic programming iterations until one of the three conditions is met for every $x_k \in X$, then the optimal strategy from all initial states will be represented. The resulting strategy is formed from the optimal actions in the final iteration. The optimal strategy is considered *stationary*, since it only depends on the state, as opposed to additionally requiring the stage index. Note that no choice of K is necessary. Also, at each iteration of the dynamic programming algorithm, we only retain the representation of $\bar{L}^*_{k+1}$ while constructing $\bar{L}^*_k$.

To execute a strategy, the robot uses the final cost-to-go representation, which we call $\bar{L}^*_1$. The robot is not confined to move along the quantization grid that is used for determining the cost-to-go functions. The optimal action can be obtained from any real-valued location $x \in X$ though the use of (12), linear interpolation, and the approximate representation of $\bar{L}^*_1$. A real-valued initial state is given (the final component represents the environment mode, and is an integer). The application of the optimal action will yield a new real-valued configuration for the robot. This form of iteration continues until the goal region X_G is entered (assuming that it can be reached).

We briefly discuss the computational performance of the algorithm. Let Q denote the number of cells per dimension in the representation of $\mathcal{C}_{free}$. Let n denote the dimension of $\mathcal{C}_{free}$. Let E denote the number of environment states. Let U denote the number of actions that are considered. The space complexity of the algorithm is $O(Q^n E)$, which is proportional to the size of the state space. For each iteration of the dynamic programming, the time complexity is $O(Q^n E^2 U)$, and the number of iterations is proportional to the robot velocity and the complexity of the solution strategy. The number of iterations required is directly proportional to the number of stages required for the longest (in terms of stages) optimal strategy that reaches the goal. The computation at each cell (in the application of (12)), has time complexity $O(EU)$, with n fixed.

Although the computational cost of dynamic programming increases exponentially in the dimension of the state space (as is the case with most algorithms for the basic motion planning problem without uncertainty [37]), for a given dimension the algorithm is quite efficient. The computational approach is reasonable for problems as large as a three dimensional configuration space with several environment modes. This dimensionality includes many interesting motion planning problems (see [37]); however, for more difficult problems some additional techniques may need to be developed.

In our simulation experiments, we have considered problems in which the dimension of $\mathcal{C}_{free}$ is two or three, and we have considered up to 32 environment modes. For two-dimensional configuration space, we typically divide the space into $50 \times 50 \times |E|$ cells, and use from 16 to 64 quantized actions (excluding $\emptyset$) to approximate translational motion. For three-dimensional configuration space, we typically divide the space into $50 \times 50 \times 64 \times |E|$ cells. These levels of resolution produce very reasonable results for most motion planning problems (see the computed examples in Section 5).

The computation times vary dramatically, depending on the resolution of the representation, number of environment states, and dimension of the configuration space. For the examples that we present in this paper, the times to compute the optimal strategy vary from about one minute to a few hours, on a SPARC 10 workstation. Execution of an optimal strategy is sufficiently fast for most applications; motion commands can be provided every few milliseconds. The computation of the optimal strategy can in many applications be considered as an off-line precomputation. The stored strategy can then be quickly executed on-line with a fast sampling rate for state feedback (up to hundreds of Hz on a typical workstation or a mobile robot-based processor board).

One improvement for computing optimal strategies would be to parallelize the implementation of the dynamic programming equation (12). The computation of the optimal action at each location x_k, depends only on a very local portion of the representation of $\bar{L}^*_{k+1}(x_{k+1})$, and on no portion of $\bar{L}^*_k(x_k)$. For a large class of problems, it may also be possible to determine the optimal strategy by a type of wavefront expansion on the state space. This can be considered as a generalization of Dijkstra's algorithm, and would require only a single pass over the state space, potentially improving the computational performance by a couple orders of magnitude.

5 Specific Models with Computed Examples

In this section we present computed examples from each of the three problem classes that were discussed in Section 2, by using the computation method discussed in Section 4. The mathematical models from Section 3 are specialized to model specific problem types. Section 5.1 presents computed examples that involve a changing configuration space. Section 5.2 presents examples that involve hazardous regions and shelters. Section 5.3 presents examples that involve servicing. The incremental motion equations for these examples are provided in Section 3.2 and Appendix A.

5.1 Changing Configuration Space

We first describe how the concepts in Section 3 specialize to this problem. Suppose there are m regions in the workspace that can appear or disappear, and we wish to prohibit collisions if they appear. We define m dynamic regions, $\mathcal{D}_1, \ldots, \mathcal{D}_m$.

We will assume that the stochastic processes that govern these regions are independent. In general, we have 2^m environment modes, which correspond to each possible subset of obstacles that can appear. If the region processes are dependent, several of these subsets of regions might not be possible (for one reason or another in practice), thereby reducing the number of environment modes. Our framework supports dependent processes by defining the appropriate environment transition probabilities; however, we use independent processes to ease the modeling, through the use of Poisson processes.

Recall the example process given in Section 3.1. The complete specification of the environment process is given for $m = 2$ and identical Poisson processes that govern the doors. A straightforward extension can be made to m dynamic regions, with distinct Poisson processes.

We define two Poisson arrival rates for each dynamic region, $\mathcal{D}_i$: λ_0^i and λ_1^i. Using equations similar to (1), probabilities of a region appearing or disappearing can be derived, to yield: P_{00}^i, P_{01}^i, P_{10}^i, and P_{11}^i. Recall from (2) that environment transition probabilities could be constructed from products of pairs of the probabilities P_{ij}. To generalize this each environment transition probability is given by

$$P(e_{k+1}|e_k) = \prod_{i=1}^{m} P_{kl}^i, \tag{13}$$

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

We now describe how the loss functional is built, by applying definitions from Section 3.3. Each $\mathcal{D}_i$ is considered as a contact dynamic region, from which m dynamic X-regions are formed. We define the terms in (9) as $c_u = \Delta t$, $c_i = \infty$, and $c_i' = 0$. By setting c_u, we obtain time-optimal solutions when the goal is reached without collision. The constant c_i provides a penalty for colliding with a dynamic region that has appeared, which precludes this alternative from the space of reasonable strategies. We also let $c_f = \infty$ in (10).

We additionally need to describe the behavior of the appearing and disappearing regions when the robot approaches them or is in a configuration in which a region might suddenly appear on top of the robot. We allow the state transition probabilities to additionally by conditioned on the robot's configuration. We assume that the probability is zero that a region will appear in the same location as the robot. We can, of course, remove this restriction; however, in this case the robot will choose to always avoid the dynamic region because there is a nonzero probability of obtaining an infinite penalty.[2].

We now present several computed examples. A simple example is first presented in Figure 10 that illustrates many of the concepts. Figure 10.a shows a problem in which there is a point robot that translates in $\Re^2$ using (3). A single doorway exists in the workspace; therefore, there are two environment states, $e = 0$ and $e = 1$. The outer dimensions of the workspace for this and all other examples are 100×100. For this example, $\|v\|\Delta t = 2$, $P_{00} = P_{11} = 0.98$. The goal region, X_G for this problem and others in this class exist in all layers of X (i.e., the goal does not depend on the environment mode).

Figure 10.b depicts 20 sample paths from a fixed initial location to the goal region, under the implementation of the computed optimal strategy. Initially $e_1 = 0$, indicating that the door is open. Recall from Section 3.2 that $W(\gamma^*, x_1)$ represents the random process yielded by the strategy γ^*, starting from $x_1 \in X$. Each of the 20 sample paths is obtained by sampling an environment mode sequence, $\mathbf{e}$, from the Markov process, to obtain $w(\gamma^*, x_1, \mathbf{e})$, which corre-

[2]In the implementation, we represent infinities as large, positive real values, and avoid technicalities of assigning infinite cost to a set of measure zero.

sponds to one fixed trajectory in X that terminates in the goal. Figure 10.b clearly illustrates different sample paths that can result during execution, even though the strategy is fixed. These differences are caused by different $\mathbf{e}$. Some sample paths go through the doorway, and others take a longer way to the goal. In one sample path, the robot begins to go around because the door closed; however, the robot changes its direction and heads toward the doorway again because the door reopened.

Figures 10.c and 10.d depict the optimal strategy γ^*. The direction of each arrow indicates the direction of motion (specified as $u_k = \gamma^*(x_k)$) for the robot, from that particular state location. The state space was quantized into $75 \times 75 \times 2$ locations for determining the optimal strategy; however, for clarity we show actions at fewer locations in the figures. When $e = 0$, a sharp division is observed between places in the state space that lead to the doorway, places that lead to the open corridor. When $e = 1$, the robot is lead through the open corridor, to the goal region.

Figures 10.e and 10.f show 20 level-set contours of the cost-to-go function, $L_1^*(x_1)$. This function increases as the distance from the goal increases. For translational motion, the negative gradient of the cost-to-go function represents the direction of motion for the robot. Hence, the cost-to-go function is similar to a numerical navigation function [33, 37, 56]; however, in our work, we obtain the representation of $L_1^*(x_1)$ as a by-product of determining the optimal strategy.

We next show some results for a more complex example, in which there are 18 doorways, in Figure 11. We assume a point robot, in which $\|v\|\Delta t = 3$. There are three different classes of doors, which open and close simultaneously (see Figure 11.a). This results in three disconnected dynamic regions and eight environment modes. Each class of doors is governed by the same Poisson parameters as the previous example.

Figure 11.b shows 20 sample paths under the implementation of the optimal strategy, when $e_1 = 0$ (all doors are initially open). It is observed that many different sample paths are obtained, under the optimal strategy, γ^*. For this example, there are places in the state space in which the optimal action is $\gamma_k^*(x_k) = u_k = \emptyset$ (i.e., the robot waits for some door(s) to open). Figure 12 shows 30 level-set contours of the cost-to-go function, $L_1^*(x_1)$. When some of the doors close, wells form in the cost-to-go function. Figure 13 shows two portions of $L_1^*(x_1)$, which correspond to $e = 0$ and $e = 7$. When $e = 7$, the robot could be trapped in any of the nine square compartments, and must wait for a door to open; this causes nine wells to appear in the cost-to-go function.

Figure 14 shows results from the problem discussed in Figure 2.a. Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 1$ (the lower door is closed, and the upper door is open). Graphs of $\mathbf{e}$ are also given. For the lower door, we have $P_{00} = P_{11} = 0.99$, and for the upper door, we have $P_{00} = P_{11} = 0.98$. The incremental motion model for the robot is constrained rotation with reverse, as described in

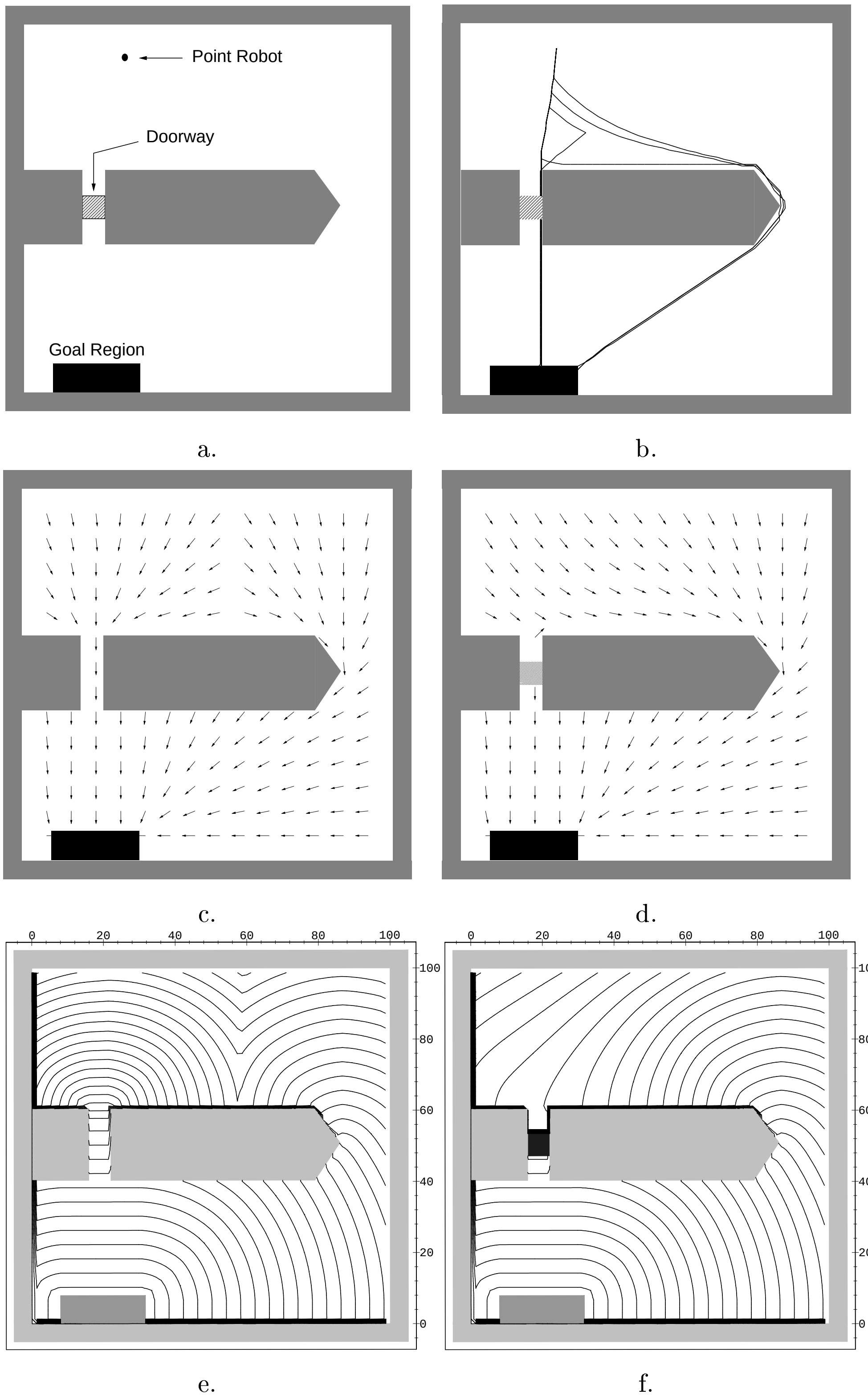


Figure 10: a) A door problem; b) 20 sample paths; c) γ^* at $e = 0$; d) γ^* at $e = 1$; e) isoperformance curves at $e = 0$; f) isoperformance curves at $e = 1$.

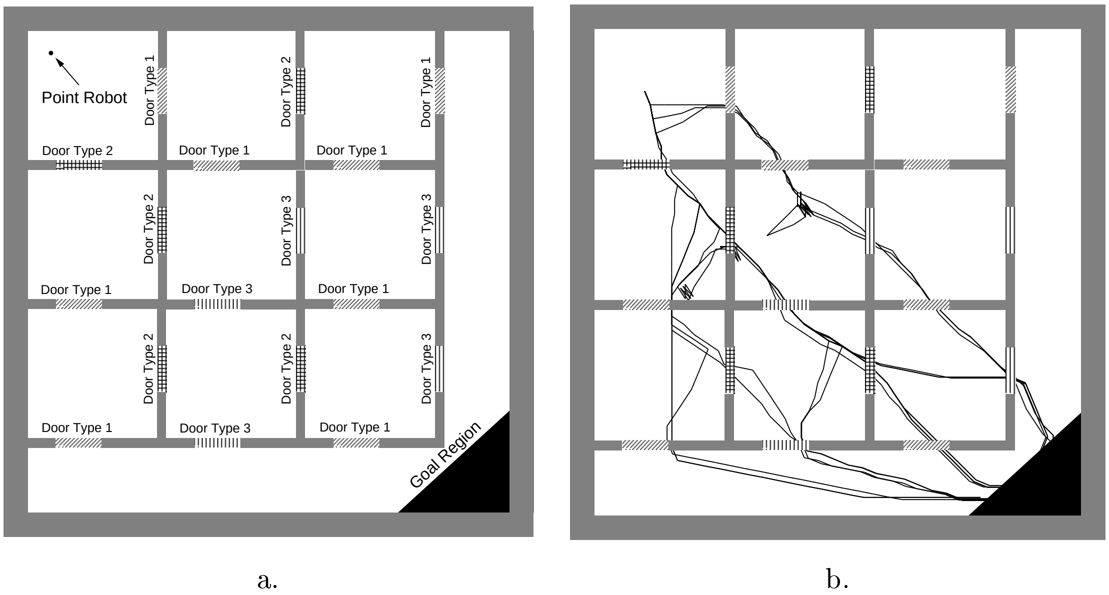


Figure 11: a) A problem that has 18 doors; b) 20 sample paths.

Appendix A, in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$. Four very different sample paths are shown. In the first sample the lower door opens, and the robot efficiently moves to the goal region. In the second sample the lower door remains closed for a long period of time, and the robot chooses to move through the upper doorway, taking a much longer route. In the third sample the robot starts to head for the upper doorway, and then changes its heading when the lower door opens. In the fourth sample the lower door opens and then closes again. The robot decides to wait for the door to open again, instead of taking the longer route.

Figure 15 shows results from the problem discussed in Figure 2.b. Four sample paths are shown under the implementation of the optimal strategy. There are five dynamic regions, which each correspond to a transient obstacle. For each transient obstacle we have $P^i_{00} = P^i_{11} = 0.98$. Initially, $e = 0$, which corresponds to the existence of none of the five transient obstacles. The robot can translate in the workspace through (3), in which $\|v\|\Delta t = 3$. Again, we observe many different sample paths as the free-configuration space changes in different ways. If a transient obstacle appears at any time during the execution, it is shown in the figure (i.e., it may appear that the robot collides with the transient obstacle in some of the figures, but the obstacle disappears in time).

5.2 Hazardous Regions and Shelters

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

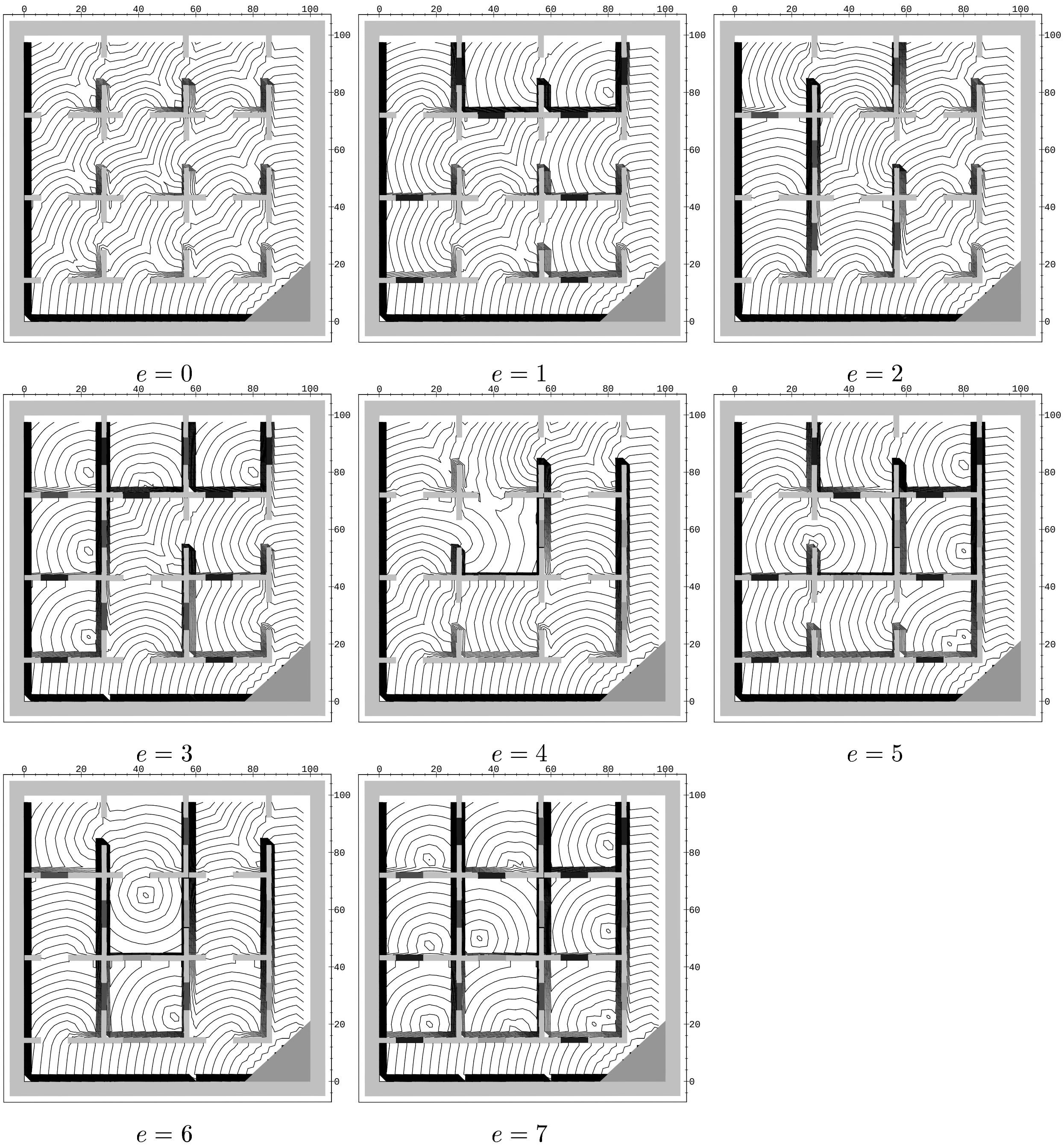
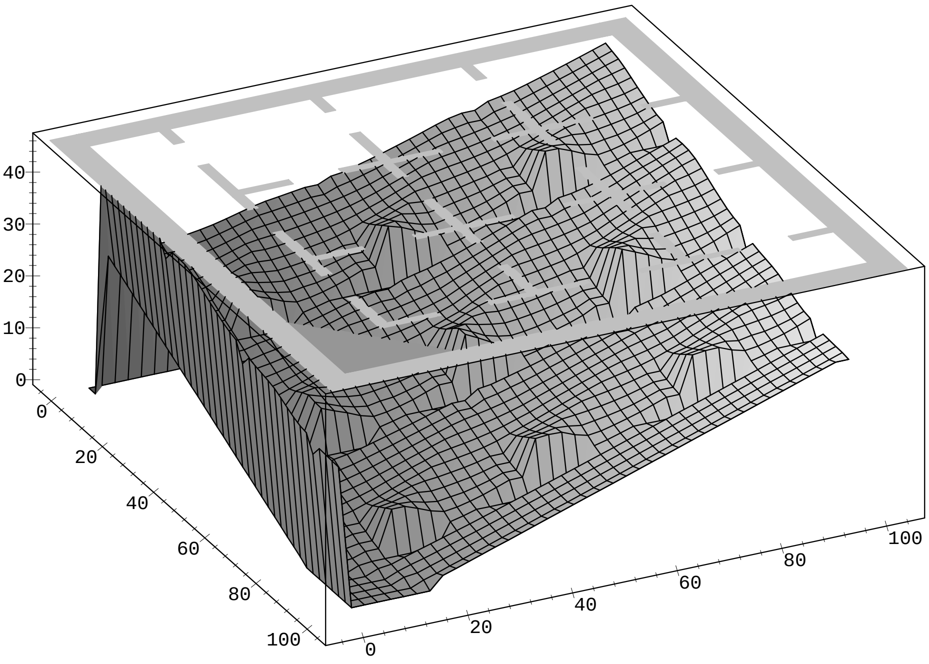
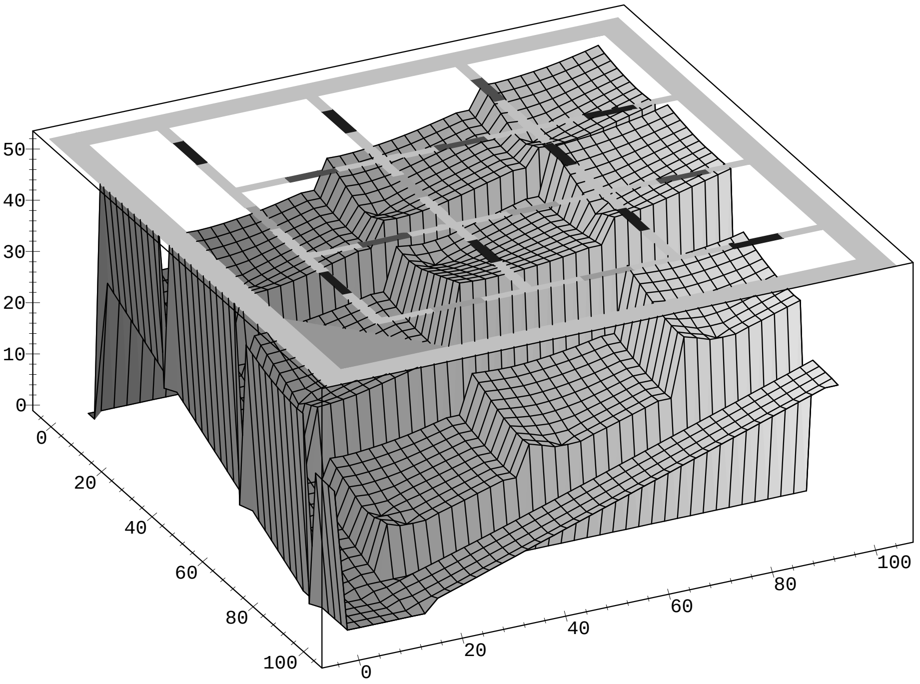


Figure 12: The isoperformance curves for γ^*.



b.



a.

Figure 13: Cost-to-go functions for: a) $e = 0$ and b) $e = 7$.

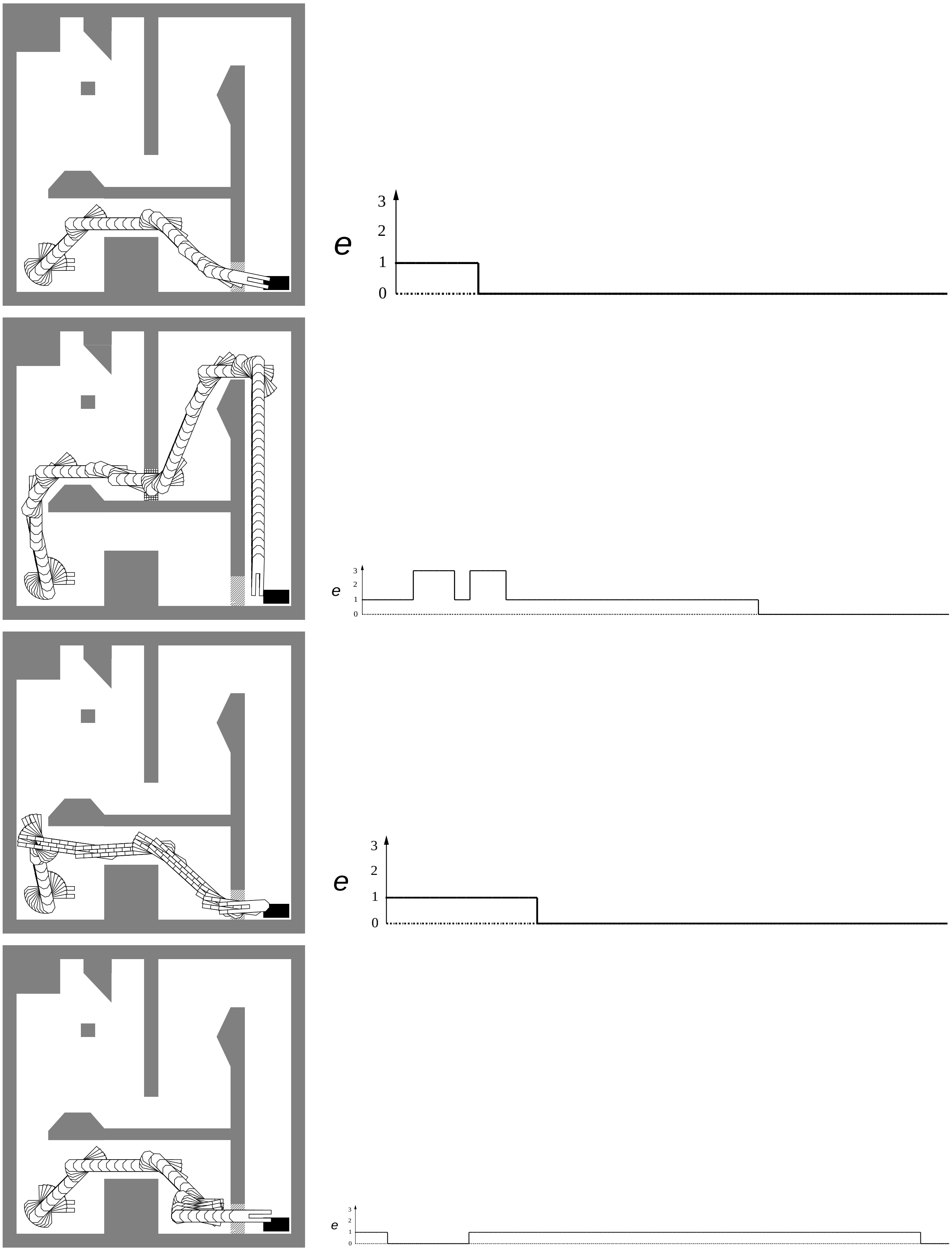


Figure 14: Four sample paths for a changing configuration space problem with two doors.

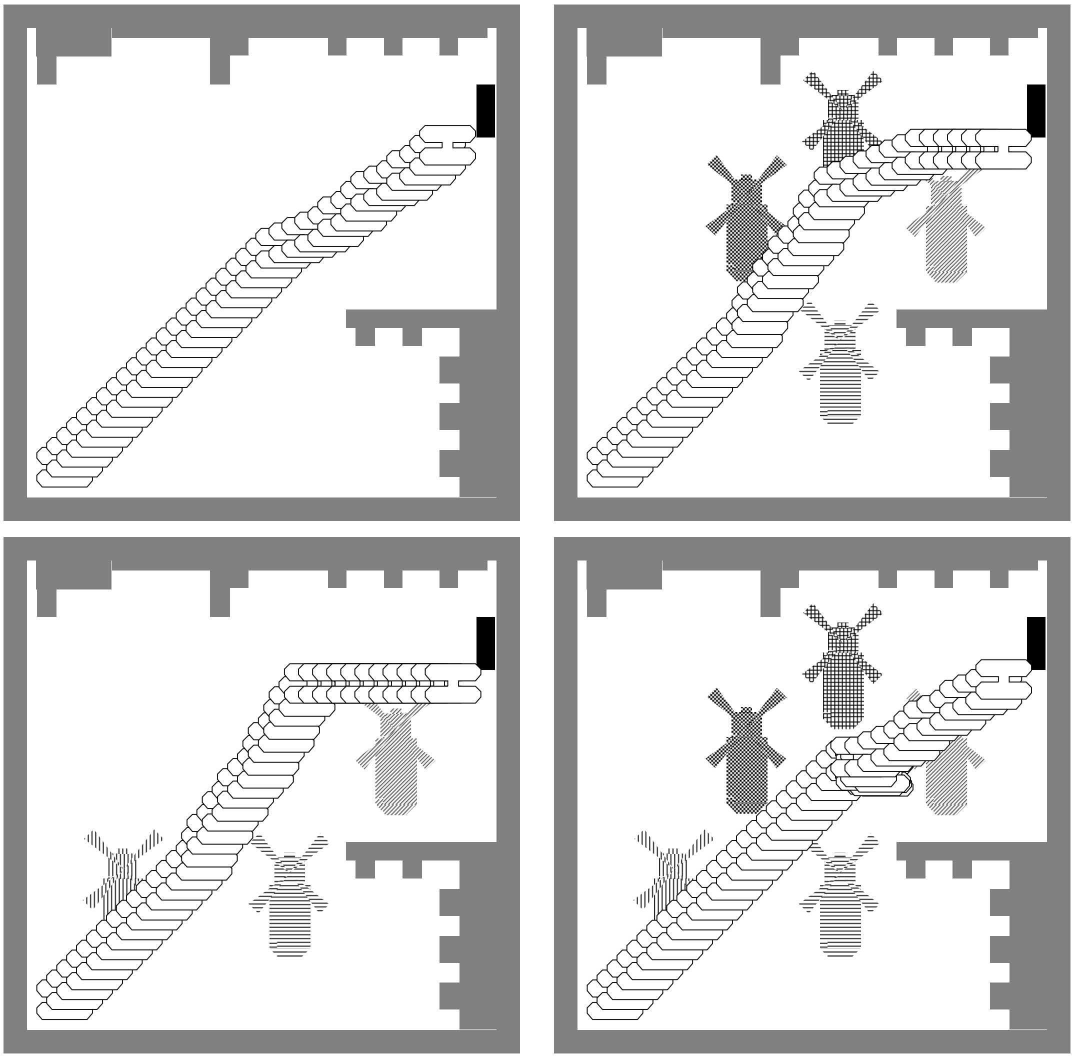


Figure 15: Four sample paths for a transient obstacle problem.

levels of danger, or different shelters for different types of hazards. We have a single dynamic region, $\mathcal{D}_1$. We let $c_f = \infty$, $c_u = \Delta t$, $c_i = 0$, and $c_i' > 0$. To construct the loss functional, $\mathcal{D}_1$ is considered as an enclosure dynamic region, as defined in Section 3.3.

Figure 16.a shows a basic example that illustrates the shelter and hazardous region concepts. There is a point robot that translates in $\Re^2$ using (3), and four thin horizontal regions that are designated as shelters. For this example, $\|v\|\Delta t = 2$, $P_{00} = 0.75$ and $P_{11} = 0.98$. The loss function is defined with $c_0 = 0$ and $c_0' = 5$. This is a generalization of a local path optimization problem defined in [60], which involved a single horizontal shelter region, with analogy to the problem of crossing a street. The current problem can be seen as analogous to the problem of crossing a multi-lane street with multiple median shelters. For the one lane case, an analytical

solution was presented in [60] but the analysis cannot be generalized easily to the multiple lane case.

Figure 16.b shows 20 sample paths under the implementation of the optimal strategy. One can see clearly the use of the shelters during the times when the rest of the "street" becomes hazardous ($e = 1$). During the environment mode $e = 1$, the best strategy seems to be to head toward the next shelter (median) and move along the shelter until the environment mode changes back to 0. This intuitive observation about the robot's behavior is further supported by the results shown in Figures 16.e and 16.f, which show 20 level-set contours of the cost-to-go function, $L_1^*(x_1)$. For $e = 0$, the contours are arranged to draw the robot across the street and directly toward the goal. For $e = 1$, on the other hand, the robot is forced to approach and move along the shelters.

Figure 17 shows results from the problem discussed in Figure 3. Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 0$ (the environment is not hazardous). We have $P_{00} = P_{11} = 0.98$. The incremental motion model for the robot is constrained rotation with reverse, as described in Appendix A, in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$. Very different sample paths are obtained during execution, which reflect the responses due to the environment becoming hazardous. In the first sample, the environment does not become hazardous, and the robot never moves into a shelter (although it travels close to the shelters). In the remaining sample paths, the robot responds to the hazardous environment by moving into a shelter. In the final sample path, the environment became hazardous three times, causing the robot to take shelter each time. After the robot moves into a shelter, it remains there until the environment mode, e switches back to 0. Further, while it is waiting inside a shelter the robot chooses an orientation that points along the remaining optimal path for $e = 0$. We have observed this behavior more clearly through animations of the robot moving along the sample paths shown in Figure 17.

5.3 Servicing Problems

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

We now define the environment probability distribution. If $x_k \in \mathcal{D}_i$ then $P_{11}^i = P_{10}^i = 0$, and

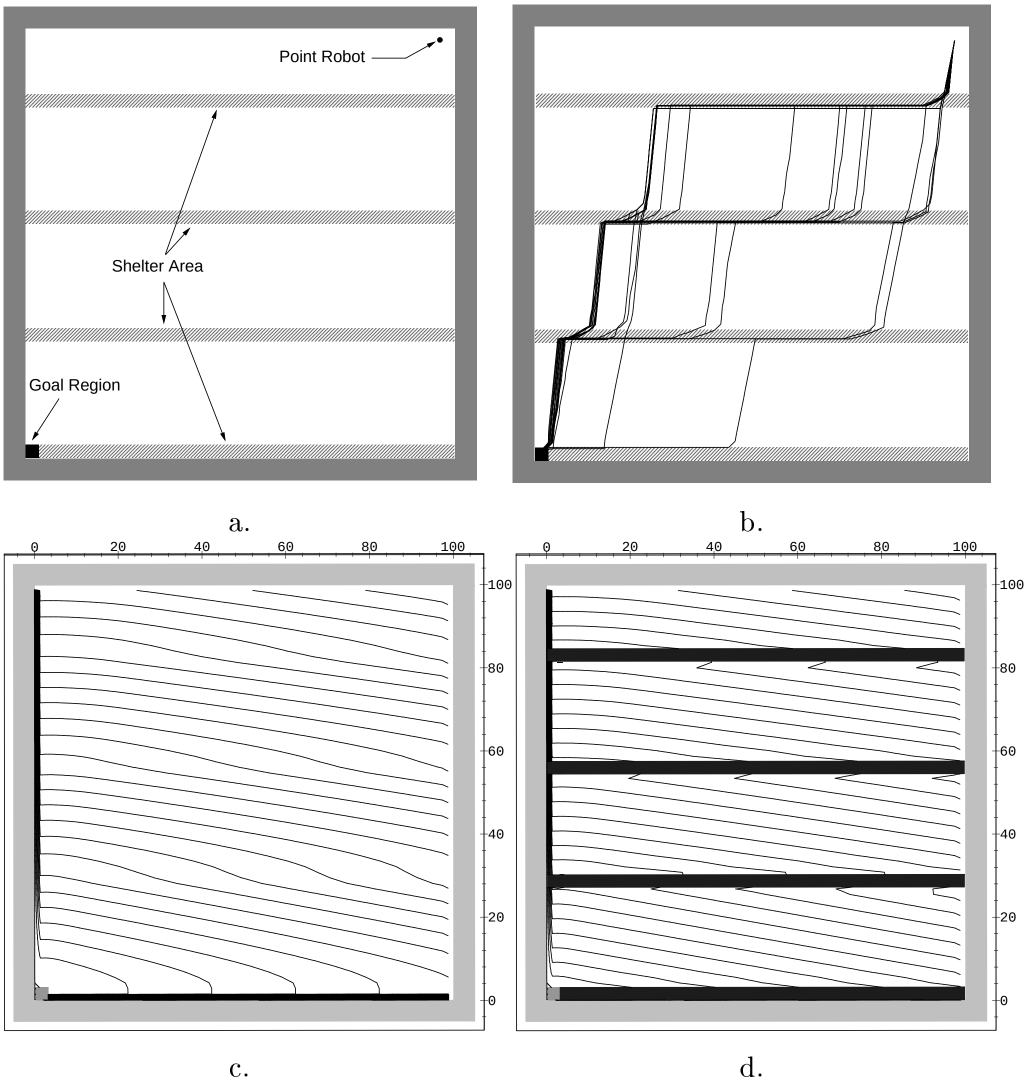


Figure 16: a) A hazardous region and shelter problem; b) 20 sample paths; c) isoperformance curves at $e = 0$; d) isoperformance curves at $e = 1$.

$P^i_{00} = P^i_{01} = 1$; otherwise, we have $P^i_{11} = 1$ and

$$P_{10} = \int_0^{\Delta t} \lambda_s e^{-\lambda_s t_a} dt_a = 1 - e^{-\lambda_s \Delta t}, \tag{14}$$

in which λ_s is the Poisson arrival rate for the i^{th} service request. The elements of the environment transition distribution are obtained by forming products as in (13).

We let $c_f = \infty$, $c_u = \Delta t$, $c_i = 0$, and $c_i' > 0$. To construct the loss functional, $\mathcal{D}_1$ is considered as an enclosure or contact dynamic region, as defined in Section 3.3.

Figure 18.a shows a basic example in which there is a translating point robot, and five small regions in which a single type of service can be performed. There is a single dynamic region,

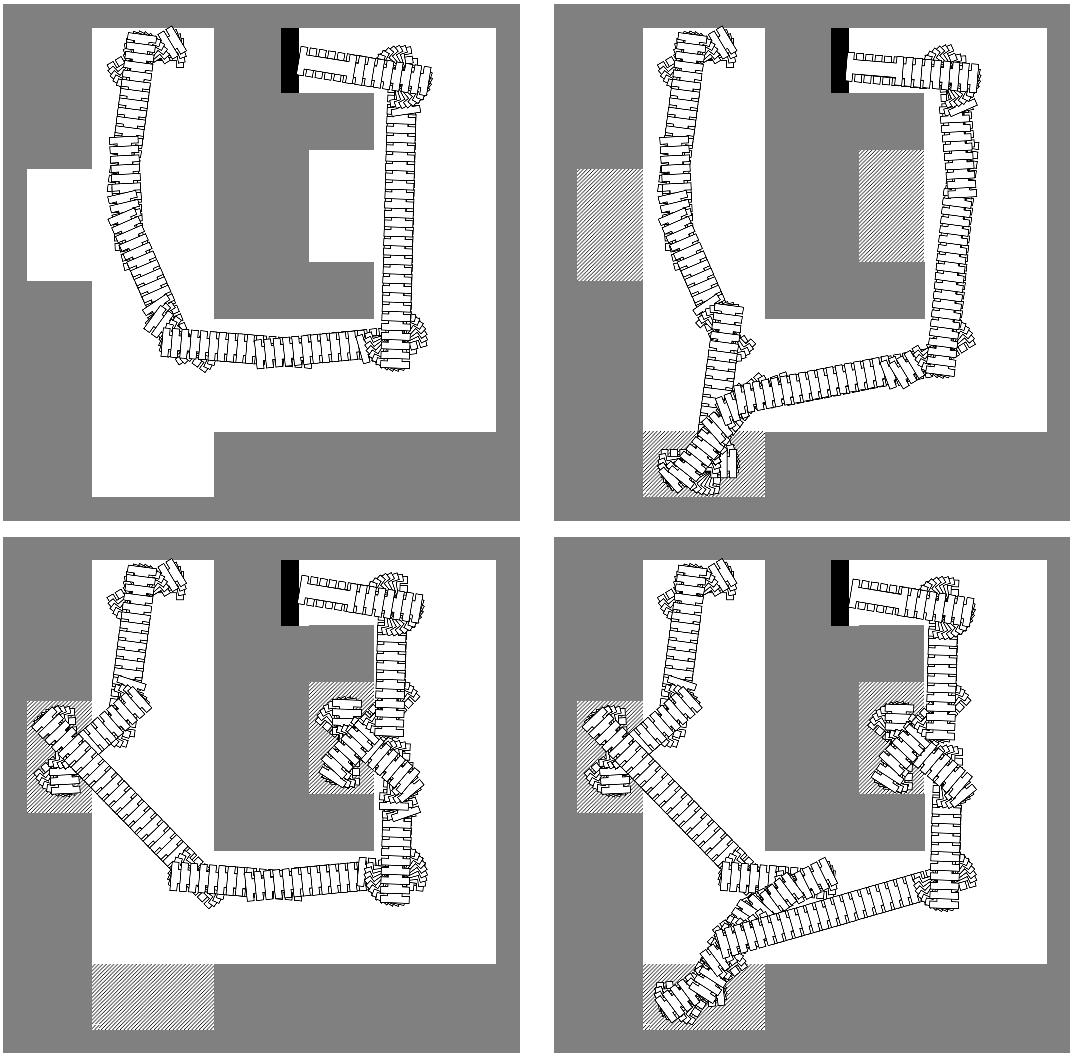


Figure 17: Four sample paths for a hazardous region and shelter problem.

which corresponds to the existence of a service request. The goal region, X_G, exists for both layers of X. This is similar to the motion planning problem analyzed in [63] with points distributed on a plane. For that model some properties of the behavior of optimal paths where presented although the actual optimal solution was not derived. Here we consider a more general problem as discussed in Section 2. Figure 18.b shows shows 20 sample paths under the implementation of the optimal strategy. One can see clearly the “detours” that the robot has to make to process the service requests. Figures 18.c and 18.d show the computed optimal strategies for the two modes 0 and 1 respectively. When there is a service request, the robot heads toward a nearby region to service the call, except as it approaches the goal region. This general behavior is brought out dramatically in the level-set contours of the cost-to-go function shown in Figures

18.e and 18.f. For the environment mode $e = 1$ the contours form wells that draw the robot toward a nearby region. This general behavior is supported by the theoretical analysis in [63], in which by analogy a Delaunay path (a path formed by edges from the Delaunay graph on the plane) would be formed from the initial to goal position when the environment mode is 1.

Figure 19 shows results from the problem discussed in Figure 4. Four sample paths are shown under the implementation of the optimal strategy, in which the initial environment mode is $e = 2$ (there is a request for the second service only). For the first service type, we have $P_{00} = P_{11} = 0.99$, and for the second type, we have $P_{00} = P_{11} = 0.98$. The incremental motion model for the robot is the nonholonomic fixed-radius motion from Appendix A, in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$. Each service region is an enclosure dynamic region. The goal region in the state space, X_G, only exists for $e = 0$; this implies that the robot must reach the goal region while there are no requests for servicing. Very different sample paths are obtained because the robot must process any request that appears in order to reach X_G.

6 Additional Models and Applications

This section presents several additional models and applications that illustrate the flexibility and extendibility of our approach. Section 6.1 discusses an extension in which the robot does not receive perfect information about the current environment mode. This form of uncertainty can be combined with the changing environment to yield strategies that are conditioned on sensor observations made by the robot. Section 6.2 discusses an extension that incorporates any time-varying, completely-predictable aspect of the motion planning problem into the motion strategy. The resulting strategy is a function of both state and time, as opposed to only state as in Section 3.2. Section 6.3 describes applications that assume that the *robot* is changing in some manner over time that is not completely predictable. This change could be in the robot geometry or in the robot motion equation, and is straightforward to incorporate into our framework.

6.1 Imperfect Environment Information: Incorporating Type ES Uncertainty

It has been assumed so far that at stage k the robot knows the environment mode, e_k. In general, it could be the case that the robot has limited sensing, and cannot perfectly determine the current environment mode. In this section we present some expressions that indicate how uncertainty in the current environment mode can be modeled and incorporated into the optimal strategy.

We assume that the environment transition probabilities depend only on the previous environment mode, and hence can be written $P(e_{k+1}|e_k)$. As before, we also assume that the robot has perfect control and configuration sensing. None of these assumptions is necessary to formulate an information space; however, a detailed discussion of information spaces is well beyond the

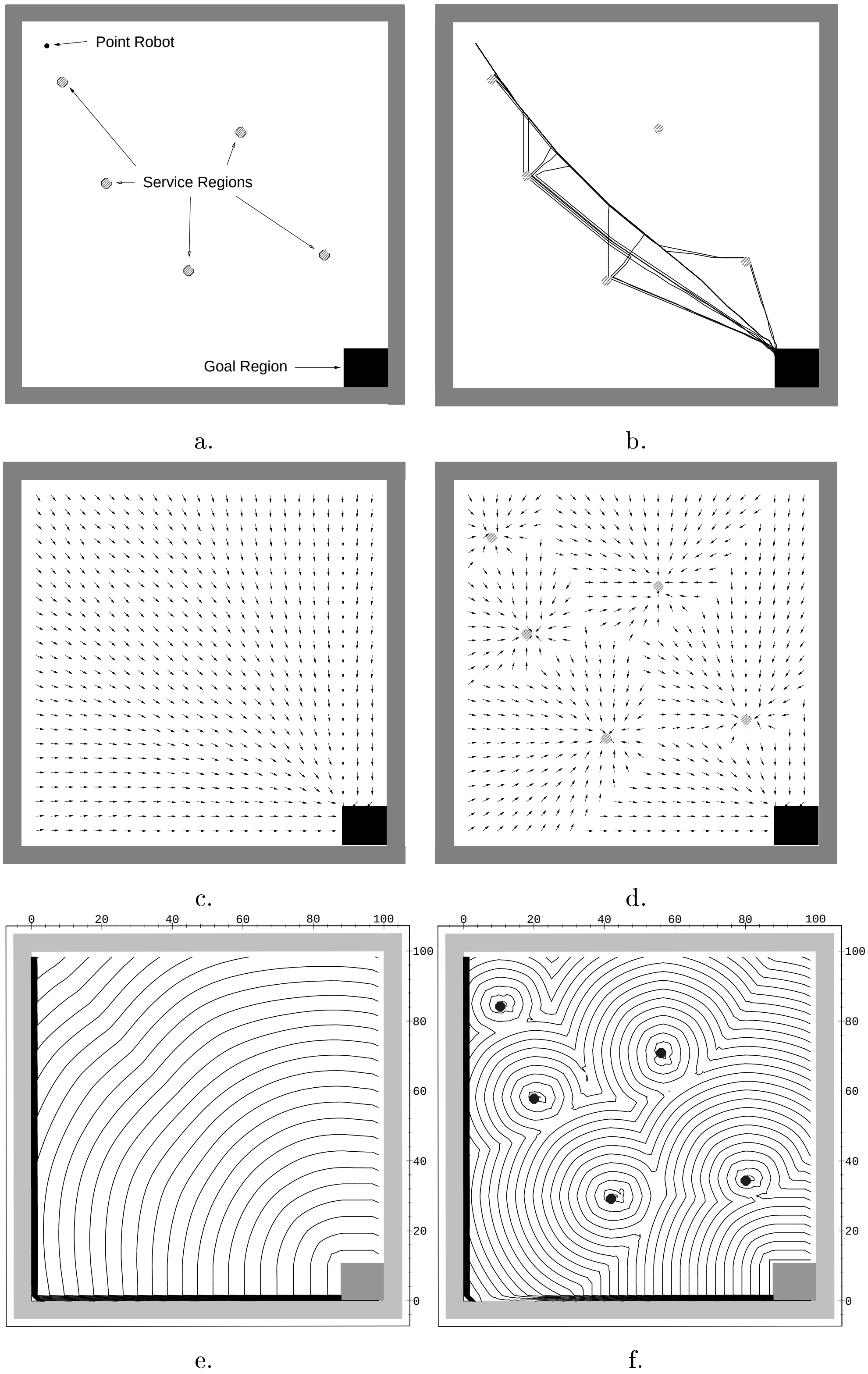


Figure 18: a) A servicing problem; b) 20 sample paths; c) γ^* at $e = 0$; d) γ^* at $e = 1$; e) isoperformance curves at $e = 0$; f) isoperformance curves at $e = 1$.

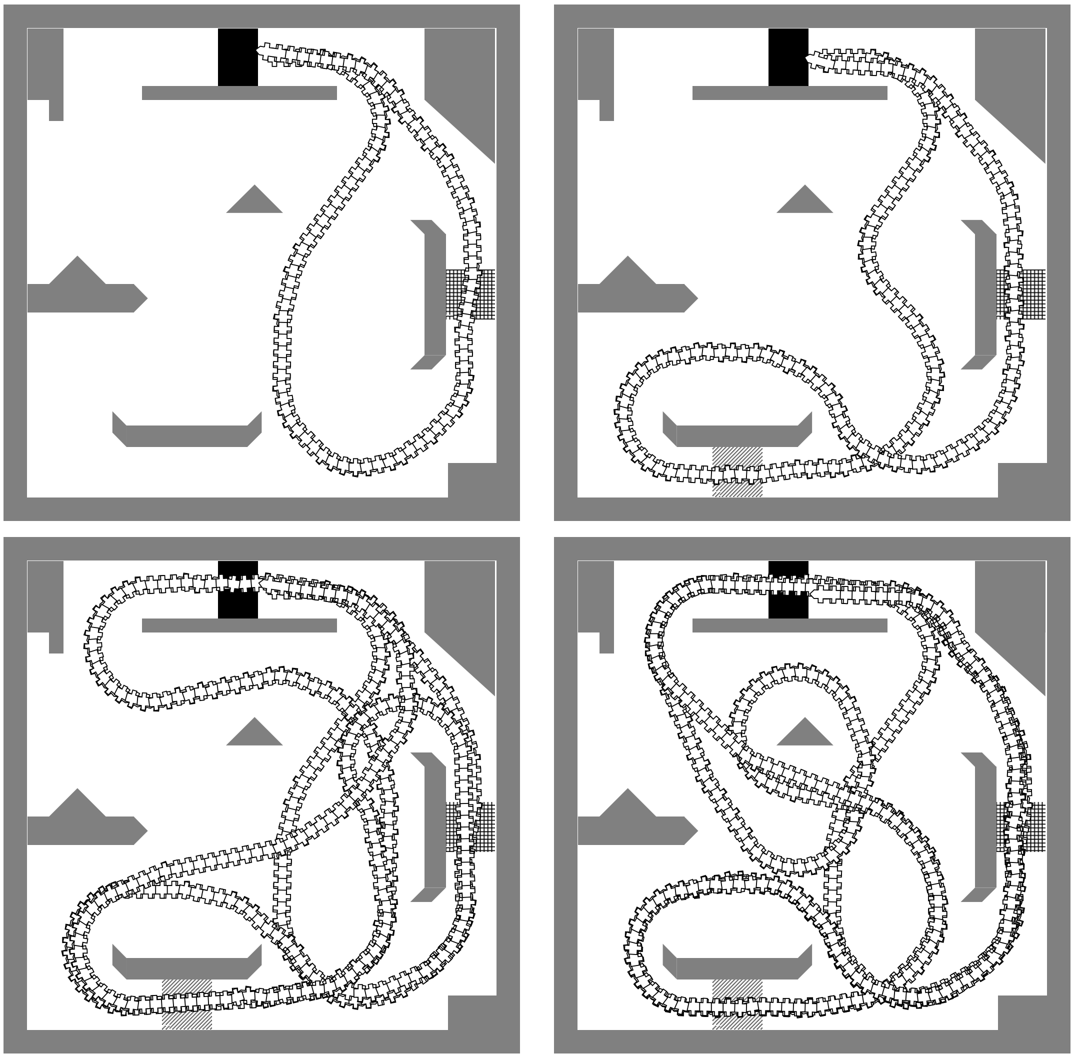


Figure 19: Four sample paths for a servicing problem with a nonholonomic car robot.

scope of this presentation. Detailed treatment of information spaces in optimal control theory can be found in [1, 35], and their application to motion planning with uncertainty in control and sensing appears in [39].

Suppose that the robot is equipped with a sensor that produces an observation o_k at each stage, $k \in \{1, \ldots, K\}$. We assume that a noise or error model for the sensor can be specified as $P(o_k|e_k)$. This characterizes the observations that are likely to be made for a given environment mode. The form $P(o_k|e_k)$ is typically used in a variety of robotics applications that involve statistical sensor error [25, 26, 41], and in general for stochastic control theory [35].

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

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

$$P(e_{k+1}|o_k, \ldots, o_1) = \sum_{e_k \in E} P(e_{k+1}|e_k) P(e_k|o_k, \ldots, o_1). \tag{15}$$

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

$$P(e_{k+1}|o_{k+1}, \ldots, o_1) = \frac{P(o_{k+1}|e_{k+1}, o_k, \ldots, o_1) P(e_{k+1}|o_k, \ldots, o_1)}{P(o_{k+1}|o_k, \ldots, o_1)} \tag{16}$$

in which

$$P(o_{k+1}|o_k, \ldots, o_1) = \sum_{e_{k+1} \in E} P(o_{k+1}|e_{k+1}, o_k, \ldots, o_1) P(e_{k+1}|o_k, \ldots, o_1). \tag{17}$$

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

$$P(e_{k+1}|o_{k+1}, \ldots, o_1) = \frac{P(o_{k+1}|e_{k+1}) \sum_{e_k \in E} P(e_{k+1}|e_k) P(e_k|o_k, \ldots, o_1)}{\sum_{e_k \in E} \sum_{e_{k+1} \in E} P(o_{k+1}|e_{k+1}) P(e_{k+1}|e_k) P(e_k|o_k, \ldots, o_1)}. \tag{18}$$

Equation (18) defines $P(e_{k+1}|o_{k+1}, \ldots, o_1)$ in terms of the following probabilities: $P(e_{k+1}|e_k)$, $P(e_k|o_k, \ldots, o_1)$, and $P(o_{k+1}|e_{k+1})$, which are given. Hence, at each stage during the execution of a strategy, a new posterior distribution can be computed.

The next concern is to design an optimal strategy under imperfect environment information. Let $\mathcal{P}$ denote a function space of all probability distributions over E. Let $p_k \in \mathcal{P}$ denote the probability distribution over E obtained at stage k. The dimension of $\mathcal{P}$ is $|E| - 1$. A new state space for this problem can be defined as $X = \mathcal{C}_{free} \times \mathcal{P}$. Hence, at any given stage, the robot will be at some known configuration in $\mathcal{C}_{free}$, and we have a probability distribution for E that belongs to $\mathcal{P}$. This state space has dimension $n+|E|-1$, in which n is the dimension of $\mathcal{C}_{free}$. A state transition distribution must be specified to determine x_{k+1} from x_k. The first n coordinates are given by the motion equation of the robot, as specified in Section 3.2. The state transition distribution in Section 3.2 required the environment transition probabilities, $P(e_{k+1}|e_k)$, and analogously in this case we are interested in $P(p_{k+1}|p_k)$. Using this, the dynamic programming equation (12) can be applied to yield a practical solution.

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

$$P(p_{k+1}|p_k) = P(o_{k+1}|o_k,\ldots,o_1) \tag{19}$$

which is given by (17).

The only additional issue in the dynamic programming computations is that $\mathcal{P}$ must also be quantized and approximated. Considering the problem sizes that we have already been computed, we can at least apply the current computation techniques to obtain optimal strategies for problems in which the dimension of $\mathcal{C}_{free}$ is two, and $|E| = 2$. This results in a three-dimensional information space in which the first two coordinates represent the location of the robot, and the final coordinate represents $P(e_k = 1)$, if $E = \{0,1\}$.

Figure 20 shows an example problem in which there is a translating robot that must reach a goal region in minimum time. There is one door in the workspace that can be open or closed, yielding two environment modes. It is assumed that the robot is equipped with a binary sensor that produces an observation $o_k \in \{0,1\}$. If the robot is in a portion of the workspace in which it can view the door, then $P(e_k = 1|o_k = 1) = 1$ and $P(e_k = 0|o_k = 0) = 1$. If the door is not visible, then $P(e_k = 1|o_k = 1) = P(e_k = 0|o_k = 1)$ and $P(e_k = 1|o_k = 0) = P(e_k = 0|o_k = 0)$, which implies that the sensor provides no information. For this problem, it is assumed that $P(e_{k+1}|e_k) = 0.98$ if $e_k = e_{k+1}$. Intuitively this means that the door will tend to remain in the same position.

Figure 6.1 shows four samples that result from the execution of the optimal strategy, which took a few minutes to compute. In each case it was initially given that $P(e_1 = 1) = \frac{1}{2}$. For each sample the initial configiration of the robot is the same; however, different environment mode sequences were supplied. In the first sample, the door was initially open, and the robot takes a direct route to the goal region. In the second sample the door was initially closed, but the robot moved into a region in which it can determine whether the door is open. The information that the robot has available at execution is identical for the two samples before the door becomes visible, hence the two samples are the same up to this point. Once the current environment mode is known, the robot takes an alternate route. If the model was specified such that there is a high probability that the door will open at some future stage, then the optimal strategy would

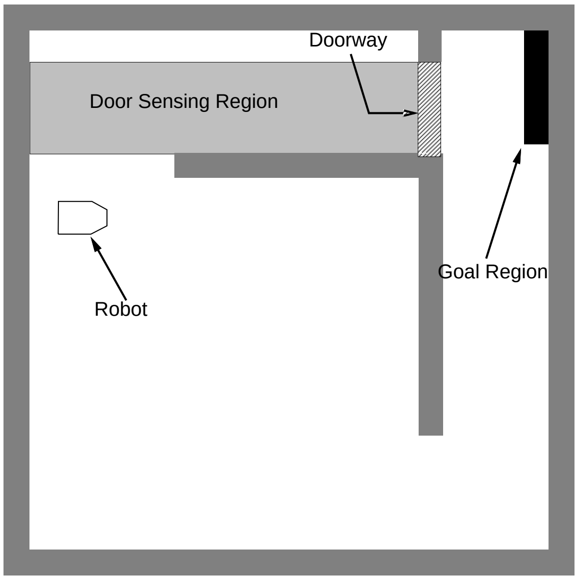


Figure 20: A problem that involves a changing configuration space and uncertainty in environment sensing. The robot can only determine whether the door is open from a specified subset of the workspace.

cause the robot to move to the doorway and wait. In the third sample, the door is initially open, but closes as the robot approaches. The fourth sample appears to be identical to the second one; however, the environment modes are different. The behaviors are the same because the robot does not know that the door opened in the fourth sample because of imperfect environment sensing.

6.2 Time-Varying Strategies

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

Figure 22 shows an example problem that results in a nonstationary strategy. In addition to a doorway that produces two environment modes, there is a moving obstacle in the workspace. It is assumed that the trajectory of this obstacle is known to the robot. Suppose that the goal is to bring the robot to the goal region in minimal time without colliding with the doorway or

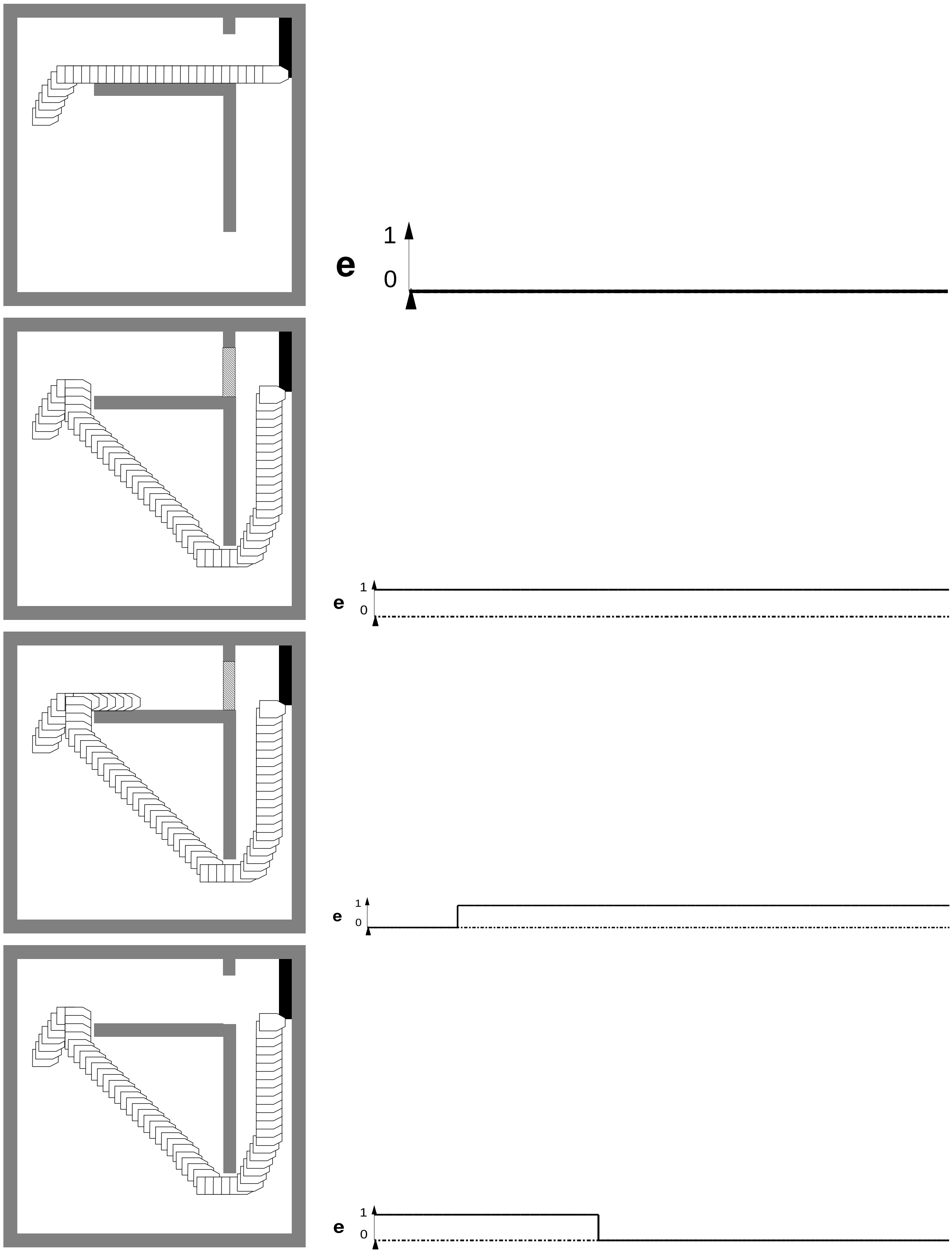


Figure 21: Four sample paths for a changing configuration space problem with one door and environment sensing uncertainty.

the moving obstacle.

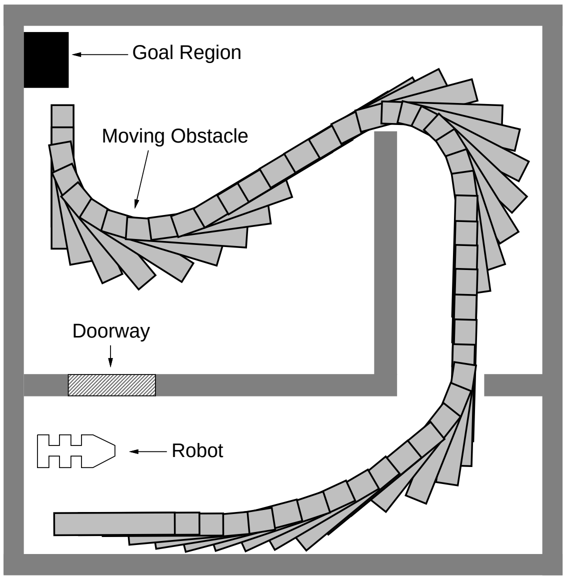


Figure 22: A motion planning problem that involves a doorway and a moving obstacle that has a known trajectory.

We briefly describe the general time-varying components that can be defined to yield non-stationary solutions. Suppose that the workspace contains obstacles, $\mathcal{B}_1(t), \ldots, \mathcal{B}_q(t)$, that may possibly be in motion. This results in a time-varying free configuration space, $\mathcal{C}_{free}(t)$ [37]. To handle discrete time, at each stage, k, we define a *stage-dependent* free configuration space

$$\mathcal{C}_{free}[k] = \bigcup_{t \in [(k-1)\Delta t, k\Delta t)} \mathcal{C}_{free}(t). \tag{20}$$

In addition, we can have moving dynamic regions $\mathcal{D}_1(t), \ldots, \mathcal{D}_m(t)$. In configuration space each of these becomes $\mathcal{CD}_i^c(t)$ or $\mathcal{CD}_i^e(t)$, and in the state space we have $X_1(t), \ldots, X_m(t)$. As done in (20), we can similarly define $X_1[k]$, $\ldots$, $X_m[k]$ to be *stage-dependent* dynamic X-regions. To obtain the appropriate loss functional, we simply replace (9) by

$$l_k(x_k, u_k) = \begin{cases} 0 & \text{If } x_k \in X_G \\ \\ c_u + \displaystyle\sum_{i=1}^{m} [c_i I_{X_i[k]}(x_k) + c_i' I_{X_i^c[k]}(x_k)] & \text{Otherwise} \end{cases}. \tag{21}$$

with the addition of an explicit dependency on k.

Optimal strategies can be computed by slightly modifying the algorithm in Section 4. Note that these extensions do not increase the state space dimension. After each iteration of the dynamic programming, however, we need to recall the optimal actions. The final stage index $K+1$ is more significant in this case, since we do not expect the algorithm to terminate by yielding a stationary strategy; the algorithm terminates when $k = 1$. The optimal strategy will be $\gamma^* : X \times \{1, \ldots, K\} \rightarrow U$. The action taken at stage k is given by $u_k = \gamma_k^*(x_k)$.

In addition to the time-varying components discussed above, additional components can vary with time. By allowing the environment transition probabilities to vary, many more statistical processes can be modeled. For instance, it might be known that the workspace is more likely to become hazardous after some prescribed time, or become increasingly more likely to be hazardous over time. We can also allow the goal region to move over time, to obtain $X_G[k]$. In this case, the robot must intercept the moving goal as a terminating condition for the strategy (as considered in [42]).

6.3 A Changing, Partially-Predictable Robot

In all the specific models that we have considered so far, the robot has been influenced by an external, changing environment. However, we can use the general framework presented in Section 3 to model situations in which the *robot* is changing. These changes could correspond, for example, to changing the geometry of the robot, or to changing its motion equation. We briefly discuss how these can be modeled with the help of environment modes, and be easily incorporated in our framework for dealing Type EP uncertainty.

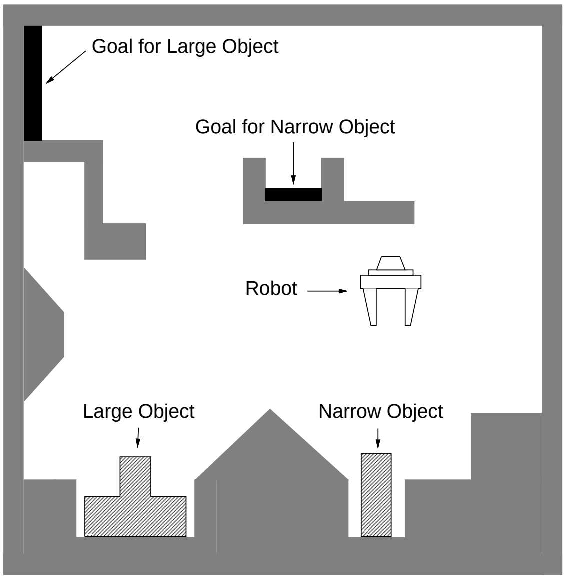


Figure 23: A "pick-and-place" motion planning problem with uncertainty that can be modeled as a servicing problem (see text).

Consider the problem depicted in Figure 23. The robot needs to pick up parts that appear at random times in one of two source bins and deliver them to a specified destination bin. We idealize the object grasping problem (for relevant issues in grasping, see [11, 52]), and assume that the robot can immediately pick up and carry an object through the workspace. While the robot is carrying an object, the geometry of the motion planning problem changes. In addition, the motion equation for the robot might be altered to compensate for the additional load [66], resulting in a different state transition distribution. We could assume that no terminating goal region is defined, and the objective is to move the robot in a manner that minimizes the amount of time that objects wait before being brought to their destinations. This problem contains the form of uncertainty addressed in this paper because the time that parts will appear cannot be predicted by the robot. For single source and destination bins, there will be three environment modes: (i) There is no object that needs to be moved; (ii) An object is waiting to be moved; (iii) The robot is carrying the object. With two source bins, as depicted in Figure 23, there are eight environment modes, assuming that the robot cannot simultaneously carry two objects.

Several extensions to the above "pick-and-place" problem are possible that can additionally be modeled by environment modes: different objects may appear in source bins, the destination bin is given only when the object is picked up, or the robot may carry multiple objects. The solutions to these problems would be useful in assembly planning. A similar approach could also be applied to motion planning of a mobile robot with *movable* objects in the workspace [74].

There are other interesting applications of the model, which involve a robot operating under different "modes" that change over time in a stochastic manner. The state transition distribution, (3), could change. For instance, we could consider a situation in which $\|v\|\Delta t$ is forced to change due to environment conditions such as a fluctuating power supply, or when the surface on which the robot is navigating suddenly becomes slippery.

Another interesting application of the model would be for dealing with motor *failures* in motion planning for a robot manipulator, particularly when it has redundant degrees of freedom [7, 49]. Such failure handling might be crucial in applications that involve carrying hazardous material, or for space operation. For example, if a motor error causes a joint to suddenly become immobile, this can be modeled in terms of a change in the state transition equation, which gives rise to null actions in a failure mode. Motion planning can then incorporate the probabilistic information of the joint failures to optimize the cost functional, which could include the probability of safely completing an ongoing task, time, or just the path length. The motion plans thus generated would be robust to joint failures.

7 Conclusions

We have presented a framework for analyzing and determining optimal robot motion strategies under a partially predictable, changing environment. This framework is general and flexible for characterizing Type EP uncertainty, by modeling the environment as a Markov process. The concept of optimal *motion strategies* under performance criteria provides a useful characterization of the desired behavior for the robot in this context. In addition, we have provided a computational approach, based on the principle of optimality, that determines optimal solutions to many motion planning problems that involve Type EP uncertainty. The variety of computed examples that were presented in Section 5 help substantiate these conclusions.

In related work, these techniques have also been applied to a problem that involves the delivery of parts from source locations to destination locations in the workspace [62]. The robot is capable of manipulating and carrying the parts, which can cause the motions and geometry to change. The particular part, the particular source location, and particular destination location are *a priori* unknown, but are modeled with a stochastic process. This problem can be considered as a component in a flexible assembly or manufacturing system (e.g., [10], [24], [34], [72]). For this problem, there is no goal region in which the robot must terminate, and strategies are selected that minimize the expected time that parts wait to be delivered. Optimal solutions have been computed for three degree-of-freedom manipulators and rigid robots.

As the number of environment modes or the dimension of the configuration space increases, it will become important to focus on appropriate tradeoffs that can be made between computational expense and the quality of solutions. In the part delivering problem, we have computed solutions for problems that involve up to 145 environment modes; however, more complex environment variations can be imagined. Furthermore, if there is uncertainty in environment sensing, planning essentially occurs in a high-dimensional information space. Dynamic replanning approaches have often been able to handle a large number of variations in the changing environment by assuming that the robot can provide a suitable response at the point in which a change is detected [13, 19, 27, 46]. One possible hybrid approach to problems with complex environments might be to design on-line optimal strategies that only take into account some local portion of the state space and time.

In this work we have focused primarily on Type EP uncertainty. In a broader setting, however, the combination of the additional sources of uncertainty from Section 1 must be addressed. We argue that the framework presented here can facilitate such a combination. In Section 6.1, we described how Type ES uncertainty can be incorporated. In addition, however, Type CP and Type CS can also be incorporated in a straightforward manner. A treatment of these forms of uncertainty in motion planning that uses concepts similar to those presented here can be found in [39]. The incremental motion model can be defined stochastically, to reflect Type CP uncertainty. The information space concepts from Section 6.1 can be expanded to include complete sensing history that characterizes uncertainty in the robot configuration. Extensions to multiple-robot coordination problems can also be considered [40, 39]. Computational issues involved in these complex combinations will depend mostly on the dimensions of the state and information spaces for the problem.

Acknowledgments

We thank Steve Sullivan for providing some of the numerical dynamic programming code.

A Incremental Motion Models

We have experimented with five different incremental motion models. Simple planar translation is described by (3). The remaining incremental motion models are presented here.

Constrained Rotation

There are four possible actions for the robot in time Δt. The first action allows the robot to remain motionless: $x_{k+1} = x_k$. The robot can either turn clockwise or counter-clockwise, in which

$$x_{k+1} = \begin{bmatrix} x_k[1] \\ x_k[2] \\ (x_k[3] \pm \theta_m \Delta t)_{mod 2\pi} \\ e_{k+1} \end{bmatrix}. \tag{22}$$

Above, θ_m represents a fixed angular velocity. Finally, the robot can translate along the orientation x_{k+1}, yielding

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\| \Delta t \; cos(x_k[3]) \\ x_k[2] + \|v\| \Delta t \; sin(x_k[3]) \\ x_k[3] \\ e_{k+1} \end{bmatrix}. \tag{23}$$

Constrained Rotation with Reverse

For this motion model, there are five possible actions. The first four are the same as for constrained rotation, and the additional action produces

$$x_{k+1} = \begin{bmatrix} x_k[1] - \|v\| \Delta t \; cos(x_k[3]) \\ x_k[2] - \|v\| \Delta t \; sin(x_k[3]) \\ x_k[3] \\ e_{k+1} \end{bmatrix}, \tag{24}$$

which allows to robot to translate in the reverse direction.

Nonholonomic Fixed-Radius Motion

For this motion model, there are four possible actions. We obtain these by replacing the two rotation actions from constrained rotation by

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\| \Delta t \; cos(x_k[3] \pm \theta_m \Delta t) \\ x_k[2] + \|v\| \Delta t \; sin(x_k[3] \pm \theta_m \Delta t) \\ (x_k[3] \pm \theta_m \Delta t)_{mod 2\pi} \\ e_{k+1} \end{bmatrix}, \tag{25}$$

which requires the robot to translate while rotating. This incrementally implements a fixed turning radius constraint that is based on $\|v\|\Delta t$ and θ_m.

Nonholonomic Fixed-Radius Motion with Reverse

This motion model simply adds the action (24) to the nonholonomic fixed-radius motion, to allow reverse translation.

References

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

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

[3] 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.

[4] K. Basye, T. Dean, J. Kirman, and M. Lejter. A decision-theoretic approach to planning, perception, and control. *IEEE Expert*, 7(4):58–65, August 1992.

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

[6] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A research agenda. In *IEEE Int. Conf. Robot. & Autom.*, volume 3, pages 549–556, 1993.

[7] J. W. Burdick. *Kinematic Analysis and Design of Redundant Manipulators.* PhD thesis, Stanford University, 1988.

[8] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. IEEE Conference on Foundations of Computer Science*, pages 49–60, 1987.

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

[10] S.-C. Chang and D.-T. Liao. Scheduling flexible flow shops with no setup effects. *IEEE Trans. Robot. & Autom.*, 10(2):99–111, 1994.

[11] M. R. Cutkosky. *Robotic Grasping and Fine Manipulation.* Kluwer Academic Publishers, Boston, MA, 1985.

[12] 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.

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

[14] H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Trans. Robot. & Autom.*, 4(1):23–31, February 1988.

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

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

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

[18] M. A. Erdmann. *On Probabilistic Strategies for Robot Tasks.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[19] R. J. Firby. An investigation into reactive planning in complex domains. In *Proc. National Conference on Artificial Intelligence*, 1987.

[20] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans. *IEEE Trans. Robot. & Autom.*, 1(11):56–71, February 1995.

[21] K. Fujimura. On motion planning amidst transient obstacles. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1488–1493, 1992.

[22] K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63, 1993.

[23] K. Y. Goldberg. *Stochastic Plans for Robotic Manipulation.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 1990.

[24] S. Gottschlich, C. Ramos, and D. Lyons. Assembly and task planning: A taxonomy. *IEEE Robotics and Automation Magazine*, 1(3):4–12, 1994.

[25] G. Hager and H. G. Durrant-Whyte. Information and multi-sensor coordination. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 381–393. Elsevier, New York, NY, 1988.

[26] G. D. Hager. *Task-Directed Sensor Fusion and Planning.* Kluwer Academic Publishers, Boston, MA, 1990.

[27] J. A. Hendler and J. C. Sanborn. Planning and reaction in dynamic domains. In *Proc. DARPA Workshop on Knowledge-Based Planning Systems*, 1987.

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

[29] 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.

[30] Y. K. Hwang and N. Ahuja. Gross motion planning–A survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[31] S. T. Jones. Solving problems involving variable terrain, part i: A general algorithm. *BYTE*, 5(2), 1980.

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

[33] D. Koditschek. Robot planning and control via potential functions. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, 1989.

[34] V. S. Kouikoglou and Y. A. Phillis. Discrete event modeling and optimization of unreliable production lines with random rates. *IEEE Trans. Robot. & Autom.*, 10(2):153–159, 1994.

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

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

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

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

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

[40] 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.

[41] S. M. LaValle and S. A. Hutchinson. A Bayesian segmentation methodology for parametric image models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):211–218, February 1995.

[42] Z. Lin, V. Zeman, and R. V. Patel. On-line robot trajectory planning for catching a moving object. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1726–1731, 1989.

[43] 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.

[44] 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.

[45] M. T. Mason. Compliance and force control for computer controlled manipulators. In B. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion: Planning and Control*, pages 373–404. MIT Press, Cambridge, MA, 1982.

[46] A. C. Meng. Dynamic motion replanning for unexpected obstacles. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1848–1849, 1988.

[47] J. S. B. Mitchell. *Planning Shortest Paths.* PhD thesis, Stanford University, 1986.

[48] 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.

[49] Y. Nakamura. *Advanced Robotics: Redundancy and Optimization.* Addison-Wesley, 1991.

[50] J. B. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. part I: The disjoint convex obstacle case. *IEEE J. of Robot. & Autom.*, 3(6):672–681, 1987.

[51] R. P. Paul and B. Shimano. Compliance and control. In *Proc. of the Joint American Automatic Control Conference*, pages 1694–1699, 1976.

[52] J. Pertin-Troccaz. Grasping: A state of the art. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, 1989.

[53] N. S. V. Rao, S. S. Iyengar, J. B. Oommen, and R. L. Kashyap. On terrain model acquisition by a point robot amidst polyhedral obstacles. *IEEE J. of Robot. & Autom.*, 4:450–455, 1988.

[54] S. Ratering and M. Gini. Robot navigation in a known environment with unknown moving obstacles. In *Proc. IEEE International Conference on Robotics and Automation*, pages 25–30, 1993.

[55] 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.

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

[57] N. C. Rowe and R. F. Richbourg. A new method for optimal path planning through nonhomogeneous free space. Technical Report NPS52-87-003, Naval Postgraduate School, 1987.

[58] G. K. Schmidt and K. Azarm. Mobile robot navigation in a dynamic world using an unsteady diffusion equation strategy. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 642–647, 1992.

[59] J. T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37:157 – 169, 1988.

[60] R. Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *IEEE Transactions on Robotics and Automation*, 8(1):105–110, February 1992.

[61] R. Sharma. A probabilistic framework for dynamic motion planning in partially known environments. In *Proc. of the 1992 IEEE International Conference on Robotics and Automation*, pages 2459–2464, May 1992.

[62] 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*. To appear in Special Issue on "Assembly and Task Planning for Manufacturing," April, 1996.

[63] R. Sharma, D. M. Mount, and Y. Aloimonos. Probabilistic analysis of some navigation strategies in a dynamic environment. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(5):1465–1474, September 1993.

[64] C. L. Shih, T-T. Lee, and W. A. Gruver. A unified approach for robot motion planning with moving polyhedral obstacles. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:903–915, 1990.

[65] R. Spence and S. A. Hutchinson. Dealing with unexpected moving obstacles by integrating potential field planning with inverse dynamics control. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1485–1490, 1992.

[66] M. Spong and I. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, 1989.

[67] H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1986.

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

[69] 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.

[70] K. Sutner and W. Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26:93–122, 1988.

[71] H. Takeda and J.-C. Latombe. Sensory uncertainty field for mobile robot navigation. In *IEEE Int. Conf. Robot. & Autom.*, pages 2465–2472, Nice, France, May 1992.

[72] C. van Delft. Approximate solutions for large-scale piecewise deterministic control systems arising in manufacturing flow control models. *IEEE Trans. Robot. & Autom.*, 10(2):142–152, 1994.

[73] D. Whitney. Force feedback control of manipulator fine motions. *Trans. ASME J. of Dyn. Sys., Meas., & Contr.*, 99:91–97, 1977.

[74] G. Wilfong. Motion planning in the presence of movable obstacles. In *Proc. ACM Symposium on Computational Geometry*, pages 279–288, June 1988.

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