A Framework for Motion Planning in Stochastic Environments: Applications and Computational Issues

Steven M. LaValle Rajeev Sharma
lavalle@cs.uiuc.edu *rajeev@cs.uiuc.edu*
The Beckman Institute
University of Illinois, Urbana, IL 61801

Abstract

In a companion paper [7], we presented a framework for analyzing motion plans for a robot that operates in an environment that changes over time in an uncertain manner. In this paper, we demonstrate the utility 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. Several computed examples are presented and discussed.

1 Introduction

In this paper we handle motion planning problems in which an environment changes over time, and is not completely predictable. In a companion paper [7], we characterized this as Type EP uncertainty (environment predictability), for which a general framework was presented. The focus of this paper is to discuss motion planning applications that can be characterized by our framework, and to present an implemented computational method for determining optimal motion strategies; several computed optimal strategies will be shown.

First, consider a motion planning problem in which "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 a particular doorway? Should it adjust its path depending on which of several 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 another doorway? 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. 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 type of problem in the literature. We can analyze and determine a solution to a problem of this type.

While accounting for the changing environment, problems such 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 $\mathcal{A}$ [6]. 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} [6]).

We can consider a problem that has 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. This concept has been considered in [3], in which transient obstacles were introduced for a motion planning problem. In this approach 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.

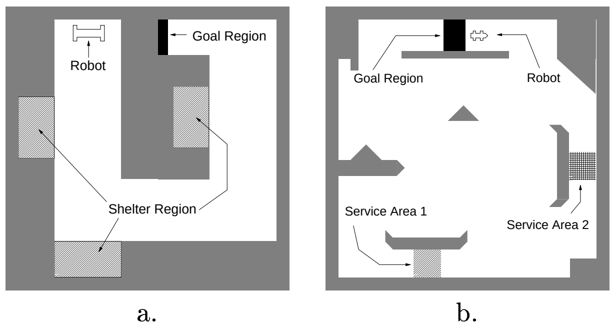


Figure 1. A problem that involves safe and hazardous regions in addition to obstacles.

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* [8]. 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 [11] 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 1.a, for a mobile robot in a factory floor in which there might be other moving robots, vehicles, or people in the corridors. Whenever the robot detects the presence of some impending collision in its path, it is directed toward *shelters* along the sides of the corridor. These shelters become relevant only when the environment is hazardous. By proper modeling of hazardous regions and shelters in the workspace, the need for explicitly considering multiple moving objects (as in [1, 2, 9]) can be avoided, while implicitly factoring in the effect of moving obstacles in the determination of motion plans.

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. 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 1.b, 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. 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.

2 Review of General Concepts

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, $X = \mathcal{C}_{free} \times E$, in which E denotes a set of environment modes. The environment mode e_{k+1} (corresponding to stage k) is known to be sampled from $P(e_{k+1}|x_k, u_k)$, which is conditioned on the previous robot action, u_k.

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.*

We 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.* 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. This loss function can be defined in terms of contact dynamic regions (mapped to X_i from $\mathcal{D}_i^c \subseteq \mathcal{C}_{free}$) and/or enclosure dynamic regions (mapped to X_i from $\mathcal{D}_i^e \subseteq \mathcal{C}_{free}$). A goal region is considered as a special kind of dynamic region, denoted by X_G.

In computed examples, we will use $w(\gamma, x_1, \mathbf{e})$ 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 Determining Optimal Strategies

We present a method for computing optimal strategies that is based on 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

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

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.

The principle of optimality [4] states that $\bar{L}_k^*(x_k)$ can be obtained from $\bar{L}_{k+1}^*(x_{k+1})$ by the following recurrence: $\bar{L}_k^*(x_k) =$

$$\min_{u_k}\left\{ l_k(x_k, u_k) + \sum_{x_{k+1}} \bar{L}_{k+1}^*(x_{k+1}) P(x_{k+1}|x_k, u_k) \right\}. \quad (2)$$

Note that the sum in (2) is taken over a finite number of states, which can be reached using the state transition distribution.

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 (2) to obtain the optimal actions. At stage $K+1$, we 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 (2). Using the $u_K \in U$ that minimizes (2) at x_K, we define $\gamma_K^*(x_K) = u_K$. We then apply (2) 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 [6, 10], and the *wavefront expansion method* that is described in [6].

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 [11]. 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, [6]). 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 (2) for various values of u_k, including $u_k = \emptyset$. The value for $\bar{L}_k^*(x_k)$ is obtained by linear interpolation on X, for a given x_{k+1}. Other schemes, such as quadratic interpolation, can be used to improve numerical accuracy [5].

Note that the $\bar{L}_K^*$ 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 (2), 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).

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.

The computation times to determine γ^* vary dramatically, depending on the resolution of the representation, number of environment states, and dimension of the configuration space. For each iteration of the dynamic programming, the time complexity is $O(Q^n E^2 U)$ (E is the number of environment modes, Q is the number of cells per dimension of X, and U is the number of actions), and the number of iterations is proportional to the robot velocity and the complexity of the solution strategy. For the examples that we present in this paper, the computation times vary from about one minute to a few hours, on a SPARC 10 workstation. It is important to note that the dynamic programming equation (2) is highly parallelizable. 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 $\bar{L}_k^*(x_k)$. Also note that once the state-feedback controller has been constructed, the optimal action can be obtained very quickly for a given state, permitting a real-time application.

4 Computed Results

4.1 Changing Configuration Space

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.

We define two Poisson arrival rates for each dynamic region, $\mathcal{D}_i$: λ_0^i and λ_1^i. The probabilities of a region appearing or disappearing can then be derived, to yield: P_{00}^i, P_{01}^i, P_{10}^i, and P_{11}^i. Each environment transition probability is given by

$$P(e_{k+1}|e_k) = \prod_{i=1}^{m} P_{kl}^i, \tag{3}$$

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 additionally assume that the probability is zero that a region will appear in the same location as the robot.

We now describe how the loss functional is built. Each dynamic region, $\mathcal{D}_i$, is considered as a contact dynamic region, from which m dynamic X-regions are formed. We define $c_u = \Delta t$, $c_i = \infty$, 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 strategies. We also let $c_f = \infty$.

We now present several computed examples. A simple example is first presented in Figure 2 that illustrates many of the concepts. Figure 2.a shows a problem in which there is a point robot that translates in $\Re^2$. 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 2.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 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 corresponds to one fixed trajectory in X that terminates in the goal. Figure 2.b clearly illustrates different sample paths that can result during execution, even though the strategy is fixed.

Figures 2.c and 2.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 2.e and 2.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.

We next show some results for a more complex example, in which there are 18 doorways, in Figure 3. 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 3.a). This results in three disconnected dynamic regions and eight environment modes. Each class of doors is governed by the same Poisson parameters as the previous example.

Figure 3.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 4 shows results from the problem discussed in [7]. 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). For the lower door, we have $P_{00} = P_{11} = 0.99$, and for the upper door, we have

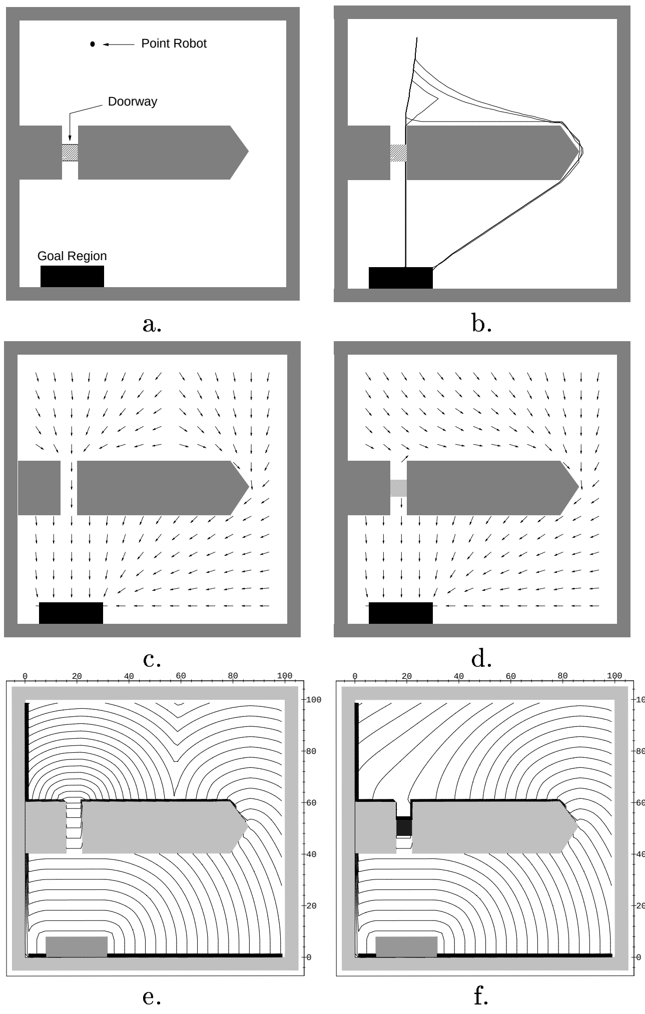
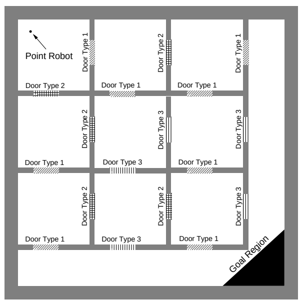


Figure 2. a) A door problem; b) 20 sample paths; c) γ^* at $e = 0$; d) γ^* at $e = 1$; e) isoperformance curves at $e = 0$; f) isoperformance curves at $e = 1$.

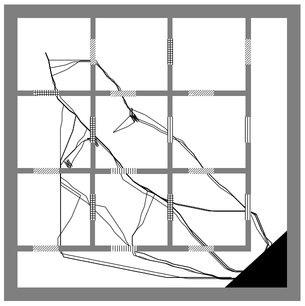
$P_{00} = P_{11} = 0.98$. At a stage the robot is allowed to either rotation, or translate in the direction of orientation (reverse allowed) in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$. In the first sample the lower door opens, and the robot efficiently moves to the goal region. In the 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.

4.2 Hazardous Regions and Shelters

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



a.



b.

Figure 3. a) A problem that has 18 doors; b) 20 sample paths.

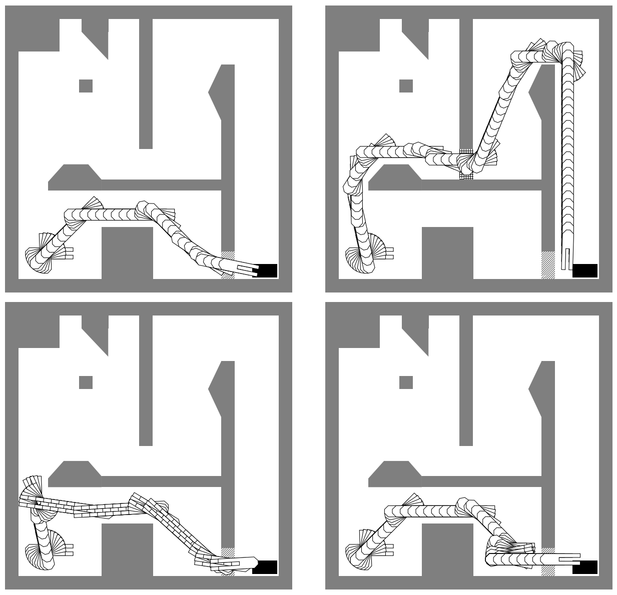
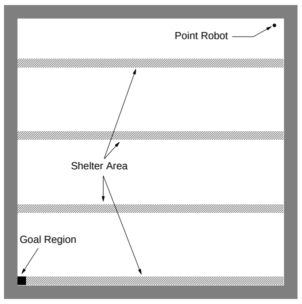


Figure 4. Four sample paths for a changing configuration space problem with two doors.

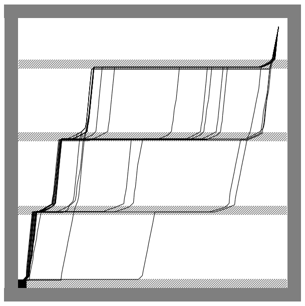
shelters for different types of hazards. We have a single dynamic enclosure region, $\mathcal{D}_1$. We let $c_f = \infty$, $c_u = \Delta t$, $c_i = 0$, and $c'_i > 0$.

Figure 5.a shows a basic example that illustrates the shelter and hazardous region concepts. There is a point robot that translates in $\Re^2$, 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 [11], which involved a single horizontal shelter region, with analogy to the problem of crossing a street.

Figure 5.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.



a.



b.

Figure 5. a) A hazardous region and shelter problem; b) 20 sample paths.

Figure 6 shows results from the problem discussed in Figure 1. 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 motion model is the same as in the example from Figure 4, in which $\|v\|\Delta t = 3$ and $\theta_m \Delta t = 0.2$. 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 6.

4.3 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 $P_{00}^i = P_{01}^i = 1$; otherwise, we have $P_{11}^i = 1$ and

$$P_{10} = \int_0^{\Delta t} \lambda_s e^{-\lambda_s t_a} dt_a = 1 - e^{-\lambda_s \Delta t}, \qquad (4)$$

in which λ_s is the Poisson arrival rate for the i^{th} service request. The elements of the environment transition dis-

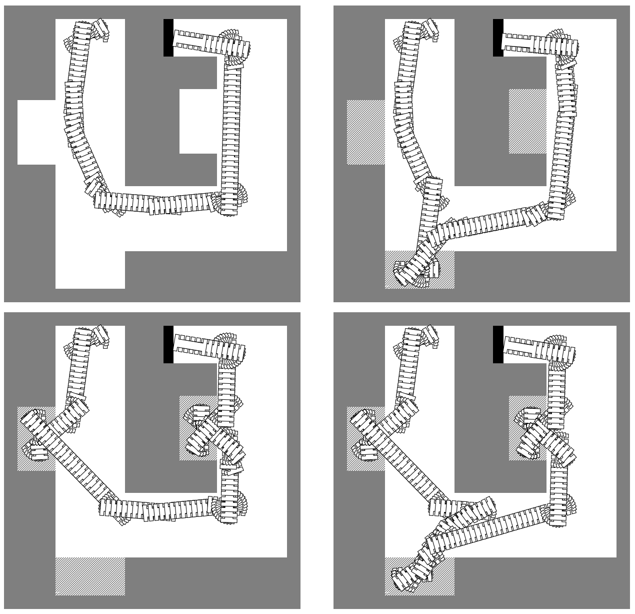


Figure 6. Four sample paths for a hazardous region and shelter problem.

tribution are obtained by forming products as in (3). 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 [7].

Figure 7 shows results for the problem discussed in Figure 1.b. 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 robot is allowed a nonholonomic, fixed-radius motion, 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.

5 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 interesting motion planning problems under Type EP uncertainty. The variety of computed examples that were presented in Section 4 help substantiate these conclusions. We are presently investigating problems in which the *robot* changes over time, for instance while carrying objects.

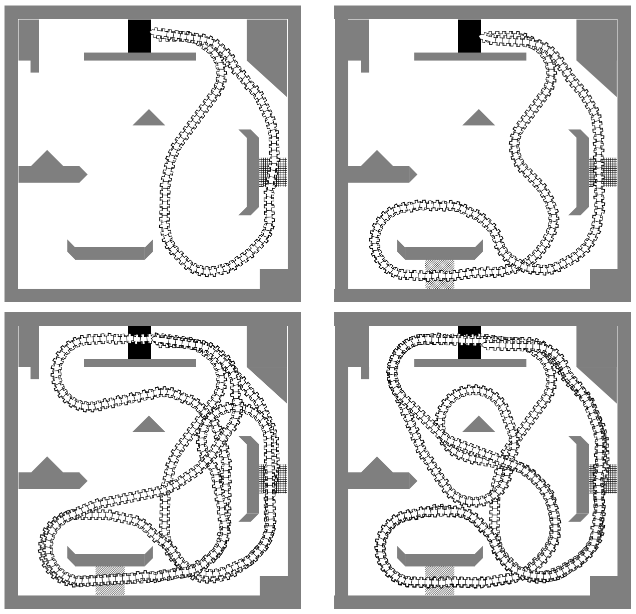


Figure 7. Four sample paths for a servicing problem with a nonholonomic car robot.

References

[1] 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.

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

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

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

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

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

[7] S. M. LaValle and R. Sharma. A framework for motion planning in stochastic environments: Modeling and analysis. *Proc. 1995 IEEE International Conference on Robotics and Automation.*

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

[9] 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.

[10] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. on Robotics and Automation*, 8(5):501–518, October 1992.

[11] 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.