An Objective-Based Stochastic Framework for Manipulation Planning

Steven M. LaValle
lavalle@cs.uiuc.edu

Seth A. Hutchinson
seth@cs.uiuc.edu

Dept. of Electrical and Computer Engineering
and
The Beckman Institute
University of Illinois, Urbana, IL 61801

Abstract

We consider the problem of determining robot manipulation plans when sensing and control uncertainties are specified as conditional probability densities. Traditional approaches are usually based on worst-case error analysis in a methodology known as *preimage backchaining*. We have developed a general framework for determining sensor-based robot plans by blending ideas from stochastic optimal control and dynamic game theory with traditional preimage backchaining concepts. We argue that the consideration of a precise loss (or performance) functional is crucial to determining and evaluating manipulation plans in a probabilistic setting. We consequently introduce a stochastic, *performance preimage* that generalizes previous preimage notions. We also present some optimal strategies for planar manipulation tasks that were computed by a dynamic programming-based algorithm.

1 Introduction

Uncertainty is inevitably involved in the planning and execution of robot tasks. Sources of such uncertainty include geometric model inaccuracies, limited or noisy sensing, and only partially predictable execution of robot commands. We address the latter two sources in this paper with a stochastic framework that is based on an explicit loss (or performance) functional. By a loss functional, we mean that we directly take into account explicit criteria such as the minimum path length, or the probability of success, when determining a robot plan. Many of the concepts introduced here are based on ideas used in stochastic optimal control theory [9] and dynamic game theory [1].

In robot motion planning, the general method of *preimage backchaining* constitutes a large body of work which assumes that sensing and control errors lie within bounded sets (e.g., [6, 11, 14]). Certain aspects of this method have been recently considered in a probabilistic context [3].

We briefly describe a manipulation planning model often used in the preimage backchaining context (see Figure 1), and further details are given in Section 3. The robot can be considered as a point moving in some subset of the plane, $\Re^2$, termed the *configuration space*. This could for instance correspond to a polygonal robot that is allowed to translate in a plane. There are subsets of the configuration space, called obstacles, which the robot is not allowed to enter. The robot does, however, have a force sensor that allows it detect collision, and move along an obstacle boundary if desired. The robot is equipped with another sensor that measures the position with a maximum error radius of ϵ_p. The robot can issue a command to move in a direction, specified as an angle between 0 and 2π; however, the actual direction that is selected is uncertain with an angular error that is bounded by ϵ_θ.

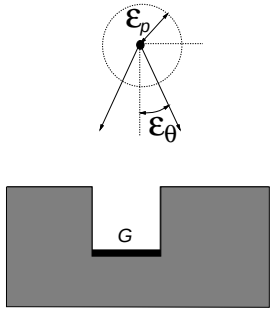

Figure 1. Accomplishing a goal under uncertainty in sensing and control.

A subset of the configuration space is defined as a *goal region*. The two primary concerns in determining a robot plan with preimage backchaining are: 1) getting the robot into the goal region (*reachability*), and 2) having the robot know to halt in the goal region (*recognizability*). We say that the goal is *achieved* if the robot successfully halts in the goal region. Using geometric reasoning techniques, a plan is constructed that guarantees that the robot will achieve the goal (see Figure 1). This plan is generally constructed using recursive subgoals, as a form of AI backchaining. For each subgoal, a *preimage* is formed that allows the robot to achieve the subgoal for a fixed command, starting from the subset of the configuration space attained from the previous subgoal. Classically, a preimage is defined as the set of all configurations form which a robot is guaranteed to achieve the goal. Figure 2 shows an example of a preimage when the robot issues a command to move down.

There are two basic questions we can ask about the performance of a particular robot plan or strategy:

- How likely is the goal to be achieved?
- If achieved, how efficient is the solution?

In geometric robot motion planning work, typically only the first question is precisely addressed, although there is often some weak preference for more efficient plans. In traditional preimage backchaining work, people have been interested in generating strategies that answer

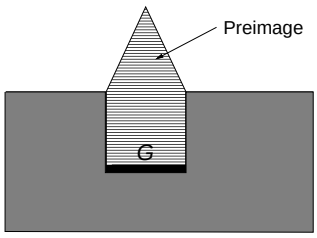

Figure 2. A simple preimage example.

the first question by guaranteeing that the goal will be achieved.

One motivation behind probabilistic backprojections [3], as well as our framework, is that worst-case analysis tends to eliminate the consideration of many reasonable robot strategies. The absolute requirement that the goal is achieved is too strong, particularly as the amount of uncertainty in control and sensing is increased. Furthermore, if bounded uncertainty models are replaced by smooth probability density functions such as a Gaussian, then it becomes impossible to *guarantee*[1] that the goal will be achieved in a fixed amount of time, except in very restricted cases. We note, however, that for applications in which probability densities are not available, then worst-case analysis may be appropriate [5].

When a probabilistic or stochastic formulation is considered, both of the previous questions need to be carefully considered. Many stochastic models will lead to guaranteed goal achievement for any possible set of bounded-velocity motion commands, even though the probability that the goal will be achieved in some reasonable finite time interval is very small. For example, continuous Brownian motion will eventually lead the robot to any nonzero-measure goal region, which indicates that Brownian motion achieves probabilistic completeness [2]. This fact forms the basis of incorporating specific diffusion processes into robotic plans in [5]. The problem in our context, however, is that the expected time to actually achieve the goal can be arbitrarily high, indicating that for some problems, at least, the probability that the goal will eventually be achieved is not useful in evaluating a strategy. Therefore the efficiency of the solution (e.g., the amount of time the robot is expected to take to achieve the goal) is of great importance in evaluating a robot strategy under general models of uncertainty. Sections 2, 4 and 5 show how objectives can be precisely defined that answer both of the questions above and guide the selection of a robot strategy.

Further benefits are provided by our stochastic framework. The relationship between sensor and action history and decision making has long been considered important for robot motion planning under uncertainty [6, 11, 14]. By using the concept of information state, as considered in stochastic systems, we provide a precise characterization of this relationship. A robot strategy will be defined in terms of this information state. Furthermore, the general structure of our framework provides insight into how certain aspects of a traditional manipulation planning problem might be generalized. For instance, if a better sensing model (in which the error is described in terms of a probability density) is determined for a given application, the appropriate probability density can be replaced, and much of the general approach remains the same. The loss functional can also easily be changed. Furthermore, this entire framework can be adapted to a more general multi-player, dynamic game theory, in which the interaction of potentially conflicting objectives can be analyzed [1, 12, 13].

Section 2 formally defines the general components that describe our version of the robot manipulation planning problem, with stochastic uncertainty in control and sensing. Section 3 applies our general formulation to express the models and assumptions used in the preimage planning approaches. By using this model we can compare the stochastic version of preimages with the traditional constructions. Section 4 introduces the performance preimage, which can be used for the evaluation of a given robot strategy. Section 5 presents a dynamic programming-based algorithm, which we have applied to the model from Section 3, that yields performance preimages and numerically optimal robot strategies. Several computed examples are presented. Section 6 provides some conclusions and possible extensions.

2 The General Framework

In this section we define the general concepts and terminology that form the basis for our framework. We consider manipulation planning with uncertainty as a dynamic game, played by a robot, $\mathcal{A}$, and nature. The robot has a general plan to achieve some goal, while nature performs some actions that potentially interfere with $\mathcal{A}$. At an abstract level, this general view of robotic manipulation tasks has been advocated in [15]. In this framework, we assume that nature chooses actions by sampling from a known probability density, $p(\theta)$; hence, nature can be represented as a random variable, Θ. For this density and the remaining probability densities in the paper, we implicitly assume there is some underlying probability space, and random variables with densities are constructed using appropriate measurability conditions.

The position of $\mathcal{A}$ in a workspace is represented by a point in an n-dimensional *configuration space*, $\mathcal{C}$, in which n is the number of degrees of freedom of $\mathcal{A}$. For manipulation planning a subset of $\mathcal{C}$, denoted as $\mathcal{C}_{valid}$, is usually defined. This corresponds to points in $\mathcal{C}$ at which either: 1) $\mathcal{A}$ does not touch an obstacle, or 2) the boundary of $\mathcal{A}$ is in contact with the boundary of some obstacle, but the interiors do not overlap. The second condition enables the possibility of guarded motion and compliance [16], which for instance allows the robot to execute a motion along the tangent of an obstacle boundary.

We associate a *state space*, X, with a given problem. We will usually take $X = \mathcal{C}_{valid}$, but in general xan allow X to represent additional parameters, such as robot velocity.

We consider a discretized representation of time as *stages*, with an index $k \in \{1, 2, \ldots, K\}$. Stage k refers to time $(k-1)\Delta t$. The state of $\mathcal{A}$ at stage k is denoted by x_k. We generally take Δt sufficiently small to approximate continuous paths. K is taken to be very large, and as will be seen in Section 5, the robot is expected to achieve the goal well before K (i.e., the specification of K is not required in our algorithm). We could also consider an infinite number of stages with only minor notation changes in

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

the remainder of this paper. The formalism could also be defined in sufficient generality without discretizing time, and consequently defining controlled diffusions [8]; however, the definitions that would follow require the use of continuous stochastic processes and substantial measure theory. Furthermore, a real robot will be limited to some sampling rate for acquiring sensor information and executing motion commands.

An *action* (or command), which is denoted by u_k, can be issued to $\mathcal{A}$ at each stage k. We let U denote the *action space* for $\mathcal{A}$, requiring that $u_k \in U$. We also consider nature as choosing actions. Nature was represented by a random variable, Θ, with a known density, $p(\theta)$. The specific action of nature at stage k is denoted by θ_k, sampled from the random variable Θ_k. We consider θ_k to be a vector quantity that is divided into two subvectors, θ_k^a and θ_k^s (i.e., $\theta_k = [\theta_k^a \ \ \theta_k^s]$). As will be seen shortly, θ_k^a affects the outcome of $\mathcal{A}$'s actions, and θ_k^s affects the sensor observations of $\mathcal{A}$. We will use the notation θ to refer to the specification of θ_k for all k.

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

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

Hence, given a robot action, nature's action, and the current state, the next state is deterministically specified. During execution, however, $\mathcal{A}$ will not know the action of nature. Hence, we often consider X_{k+1} as a random variable with density function $p(x_{k+1}|x_k, u_k)$.

At stage k, $\mathcal{A}$ makes an *observation* that is governed by the equation,

$$y_k = h(x_k, \theta_k^s), \tag{2}$$

which we term the *observation equation*. The values, y_k, belong to a *sensor space*, denoted by Y. Since information about θ_k^s is specified in the form of a density, we can also consider Y_k as a random variable, with corresponding density $p(y_k|x_k)$. As an example, h could represent a position sensor that measures x_k with Gaussian noise. Then $h(x_k, \theta_k^s) = x_k + \theta_k^s$, and $p(\theta_k^s)$ is a Gaussian density. This equation represents the output equation used in control theory, as well as a stochastic version of the projection of world states onto sensor values, used in previous robotics contexts (e.g., [4]).

The following definitions precisely describe the sensing and action history that $\mathcal{A}$ has available. For a given stage k, let η_k denote some subset:

$$\eta_k \subseteq \{u_1, u_2, \ldots, u_{k-1}, y_1, y_2, \ldots, y_k\}. \tag{3}$$

The value η_k is a set of past actions and observations that are known to $\mathcal{A}$ at stage k, and is termed the *information state* of $\mathcal{A}$. For instance, we could consider a memoryless robot, in which $\eta_k = y_k$. As another example, we could have a sensorless robot as considered in [7], in which $\eta_k = \{u_1, \ldots, u_{k-1}\}$. We could also consider the stage index k as part of the information space for the purpose of developing robot strategies that involve timing; however, we will not explicitly consider k as part of η_k in this paper. The set of values that η_k can take on is denoted by N_k, and is termed the *information space*. We define an *information structure* as the set of N_k for all $1 \leq k \leq K$.

We now define the notion of a robot plan or strategy in our stochastic framework. At first it might seem appropriate to define some action u_k for each stage; however, we want plans that are conditioned on sensor and action history. Therefore, we define a *strategy at stage k* of $\mathcal{A}$ as a function $g_k : N_k \rightarrow U$. For each information state, η_k, a strategy yields an action $u_k = g_k(\eta_k)$. The set of mappings $\{g_1, g_2, \ldots, g_K\}$ is denoted by g and termed a *strategy* of $\mathcal{A}$. This concept is equivalent to a control law in stochastic control theory [9], and is similar to a conditional multi-step plan in manipulation planning [11].

The notion of a termination condition has been quite useful for formulating robot plans that tell the robot when to halt, based on its current, partial information [6, 11, 14]. The same concept is needed in our context, hence we define a *termination condition* TC_k at each stage by a binary-valued mapping,

$$TC_k : N_k \rightarrow \{true, false\}. \tag{4}$$

Hence at each stage, the robot decides whether or not to stop, based on its current information state. We use TC to denote the complete specification for all k. The termination condition is implemented so that the game terminates at some stage $k_{TC} < K$, making the specific choice of K not important, except that it is sufficiently large.

Some subset G of the state space X is defined as the *goal region*. We encode the objectives that are to be achieved by a nonnegative real-valued functional $L(x_1, ..., x_{K+1}, u_1, ..., u_K, \theta)$, called the *loss functional* of $\mathcal{A}$. The ultimate goal of the planner is to determine a strategy g and termination condition TC that causes L to be minimized in an expected sense. We will use the notation γ to denote the pair (g, TC). This pairing is analogous to the concept of a *motion command* as defined in [11]. Further details on loss functionals are given in Section 4.

There are a few possibilities for defining the starting place in the game. We can consider an initial density $p(x_1)$. We could also consider this conditioned on an initial observation, y_1, to obtain $p(x_1|y_1)$. In traditional preimage planning, the initial state, x_1, is constrained to lie in some bounded subset of X, which could be represented by a density. The initial condition will generally be represented by the conditional density $p(x_1|\eta_1)$. We can also, or course, consider cases in which x_1 is initially given.

2.1 The information state as a conditional density

Consider the case in which the robot has perfect memory. Then each η_k corresponds to complete history of previous robot actions and observations. If U is n_1-dimensional and Y is n_2-dimensional, then in general the dimension of N_k will be $[k(n_1 + n_2) + n_2]$-dimensional. A space that grows significantly with each stage (and becomes infinite-dimensional when $K = \infty$) is very unappealing for designing strategies. The information state can, however, be considered as a conditional density on the state space, denoted as $p(x_k|\eta_k)$. By using this approach, the information state density $p(x_{k+1}|\eta_{k+1})$ can be determined from $p(x_k|\eta_k)$, when u_k and y_{k+1} are given. This observation allows the development of several well-known stochastic control results, such as the Kalman filter, when all densities in the information space take some parametric form of fixed, low dimension [9]. Hence the density

can be thought of as an alternative representation of the robot's information state. Further, this representation is intuitively satisfying since we can think of $\mathcal{A}$'s uncertainty model as a density representing possible locations in the state space X. Using bounded uncertainty models this representation would correspond to identifying a subset of X within which the configuration of $\mathcal{A}$ is known to lie, based on history.

We briefly indicate how the information state density is obtained. Initially, we have $p(x_1|\eta_1)$. We can derive an expression for $p(x_{k+1}|\eta_{k+1})$ in terms of $p(x_k|\eta_k)$, u_k, and y_{k+1}. Begin with $p(x_k|\eta_k)$. First consider the effect on the state space of using the action, u_k. From the density from the state transition equation we obtain, through marginalization with respect to X_k,

$$p(x_{k+1}|u_k, \eta_k) = \int p(x_{k+1}|x_k, u_k) p(x_k|\eta_k) dx_k. \quad (5)$$

Note that η_{k+1} can be specified with η_k, u_k, and y_{k+1}. By using Bayes' rule on X_{k+1} and Y_{k+1}, the following can be obtained [9]:

$$p(x_{k+1}|\eta_{k+1}) = \frac{p(y_{k+1}|x_{k+1}) p(x_{k+1}|\eta_k, u_k)}{\int p(y_{k+1}|x_{k+1}) p(x_{k+1}|\eta_k, u_k) dx_{k+1}}, \quad (6)$$

which is a function of $p(y_{k+1}|x_{k+1})$ (which is inferred from the observation equation) and $p(x_{k+1}|\eta_k, u_k)$ (which is inferred from (5)). If $\mathcal{A}$ does not have perfect memory, then the condition $\{\eta_k, u_k\}$ is replaced in (6) by the appropriate subset of history.

3 Modeling a Manipulation Task

In this section we describe a specific stochastic model, based on the model used in [3, 11], in terms of our framework presented in Section 2. This basic model, and a number of variations, has been used extensively for analyzing manipulation tasks under uncertainty (e.g., [6, 11, 14]). Robot strategies are generated under this model in Section 5 via dynamic programming.

Other models could be defined in this stochastic framework. For instance, Goldberg has considered stochastic modeling on a finite state space, without sensors [7]. This led to the development of a backchaining algorithm for planning squeeze grasp operations of a parallel gripper on polygonal objects.

Our definitions are based on previous preimage backchaining approaches. For the state space we have $X = \mathcal{C}_{valid}$ as defined in [11]. The robot $\mathcal{A}$ is a polygon translating in the plane amidst polygonal obstacles. The configuration space $\mathcal{C}_{valid}$ can be partitioned into two sets, $\mathcal{C}_{free}$ and $\mathcal{C}_{contact}$. $\mathcal{C}_{free}$ is an open set in which $\mathcal{A}$ does not touch any obstacles, and $\mathcal{C}_{contact}$ represents the boundary of $\mathcal{C}_{free}$, in which the robot touches obstacles. The action set of $\mathcal{A}$ is a set of commanded velocity directions, which can be specified by an orientation, yielding $U = [0, 2\pi)$. The robot will attempt to move a fixed distance $\|v\|\Delta t$ (expressed in terms of a constant velocity-modulus) in the direction specified by u_k. The action space of nature is a set of angular displacements, θ_k^a, such that $-\epsilon_\theta \leq \theta_k^a \leq \epsilon_\theta$, for some maximum angle ϵ_θ. We assume that $p(\theta_k^a)$ is a stage-independent uniform density, which is zero outside of $[-\epsilon, \epsilon]$.

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

$$f(x_k, u_k, \theta_k^a) = x_k + \|v\|\Delta t \begin{bmatrix} cos(u_k + \theta_k^a) \\ sin(u_k + \theta_k^a) \end{bmatrix}. \quad (7)$$

If $x_k \in \mathcal{C}_{contact}$, with a distance of at least $\|v\|\Delta t$ from the edge endpoints, then a compliant motion is generated by using the generalized damper model (see e.g., [6]) for certain choices of u_k. If u_k points into the obstacle edge with a sufficient angle to overcome friction, then the robot moves a fixed distance parallel to the edge. Otherwise, the robot either remains fixed, or moves away into $\mathcal{C}_{free}$. The remaining cases describe when the robot moves from $\mathcal{C}_{free}$ to $\mathcal{C}_{contact}$, $\mathcal{C}_{contact}$ to $\mathcal{C}_{free}$, or from one edge in $\mathcal{C}_{valid}$ to another. These cases are straightforward to define, and the proper modeling of the motions in these cases becomes less important in terms of robot objectives as Δt becomes smaller.

This model of uncertainty actually deviates from the model used in [3, 6, 11, 14] since nature acts at *every* step. In the traditional model, if some $u_k = u_{k+1}$, then no additional uncertainty will be introduced. The uncertain direction is selected at the beginning of a constant command, and another uncertain direction is not selected until the action changes.

The robot $\mathcal{A}$ is equipped with a position sensor and a force sensor. Assume that the position sensor is calibrated in the configuration space, yielding values in $\Re^2$. The force sensor provides values in $[0, 2\pi) \cup \{\emptyset\}$, indicating either the direction of force, or no force.

We consider independent state observation equations: h^p for the position sensor, and h^f for the force sensor (which together form a 3-dimensional vector-valued function). We partition the sensing action of nature, θ_k^s into subvectors $\theta_k^{s,p}$ and $\theta_k^{s,f}$, which act on the position sensor and force sensor, respectively. The observation for the position sensor is $y_k^p = h^p(x_k, \theta_k^{s,p}) = x_k + \theta_k^{s,p}$ in which

$$p(\theta_k^{s,p}) = \begin{cases} \frac{2}{\pi \epsilon_p^2} & \text{for } \|\theta_k^{s,p}\| < \epsilon_p \\ 0 & \text{otherwise} \end{cases}, \quad (8)$$

for some prespecified radius ϵ_p, and $\theta_k^{s,p}$ is 2-dimensional.

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

$$p(\theta_k^{s,f}) = \begin{cases} \frac{1}{2\epsilon_f} & \text{for } |\theta_k^{s,f}| < \epsilon_f \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

for some positive prespecified constant $\epsilon_f < \frac{1}{2}\pi$.

We consider the random variables $\theta_k^{s,p}$ and $\theta_k^{s,f}$ to be independent over all stages.

A number of different information spaces are possible. Several variations are discussed in [6] with respect to developing a termination criterion. For our formulation, we

could consider η_k as the complete set of sensing history and action history. The limiting factor in the definition of an information space is certainly not space available for memory, but rather the effective dimensionality of N_k (whether represented by a density or directly by a history). In Section 5 we present dynamic programming solutions for the case in which $\eta_k = y_k$, meaning that the strategies and termination conditions are developed based on current observation feedback.

A loss functional is not a concept that is part of the traditional preimage backchaining approach. However, the loss functional (14) and probabilistic preimage $\pi(0)$ to be introduced in Section 4 will yield classical preimages. We will describe in the next section how a probabilistic backprojection [3] can also be obtained.

Traditional termination conditions can be considered as binary-valued functions of the information space variables; however, we can alternatively consider the posterior density $p(x_k|\eta_k)$ in expressing TC. The stage index, k, could also be used to incorporate time into a termination condition by defining η_k to include k. In Section 5, we describe how TC is optimally chosen, along with g, to form γ.

4 Evaluating Robot Strategies

In this section we introduce the concept of a performance preimage. A performance preimage describes a region in either the state space or information space from which the expected loss in achieving the goal lies within a set of values. This concept generalizes the notion of classical preimages to arbitrary performance measures, although they are derived from discretized time in our framework. We conclude the section by discussing performance preimages for two specific loss functionals.

To begin with, suppose that we wish to evaluate some $\gamma = (g, TC)$ with a given initial state, x_1. The expected loss that we incur if γ is implemented can be expressed using $p(\theta)$ as

$$\bar{L}(\gamma, x_1) = \int L(x_1, ...x_{K+1}, u_1, ..., u_K, \theta) p(\theta) d\theta. \quad (10)$$

The integral looks as each possible action sequence for nature, θ, weighted by the probability density $p(\theta)$. When θ is given (along with γ and x_1), then the action sequence, $\{u_1, \ldots, u_K\}$, and state trajectory, $\{x_1, \ldots, x_{K+1}\}$ can be completely determined, resulting in the evaluation of the loss functional. This is true because (1), (2), and η_k can be determined for every state when the value of nature, θ, is given.

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

$$\pi_x(C) = \{x_1 \in X | \bar{L}(\gamma, x_1) \in C\}. \quad (11)$$

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

In general, the robot $\mathcal{A}$ will begin in some uncertain initial state. Therefore, we also consider performance preimages on the robot's initial information space, N_1. The classes represent places in the initial information space where if $\mathcal{A}$ begins, the expected performance will lie within some $C \subseteq \Re$. This result specializes to (11) in the case of a given initial state, $\eta_1 = y_1 = x_1$. As a minor extension, one could also consider performance preimages on any information space N_k.

Using the information state density $p(x_1|\eta_1)$ on X we can obtain by marginalization:

$$\bar{L}(\gamma, \eta_1) = \int \bar{L}(\gamma, x_1) p(x_1|\eta_1) dx_1, \quad (12)$$

which depends on (10).

For a subset $C \subseteq \Re$, we define the *performance preimage on* N_1 as a subset of N_1, denoted by $\pi(C)$, that is given by

$$\pi(C) = \{\eta_1 \in N_1 | \bar{L}(\gamma, \eta_1) \in C\}. \quad (13)$$

We now describe some particular choices for C. Suppose that $C = [0, c]$ for some $c \geq 0$ (recall that L is nonnegative). The performance preimage yields places in N_1 (or alternatively X) from which the expected performance will be better than or equal to c. If $C = \{c\}$, for some point $c \geq 0$, then we obtain places in N_1 (or X) where equal expected performance will be obtained. We can consider partitioning N_1 (or X) into *isoperformance classes* by defining an equivalence class $\pi(\{c\})$ for each $c \in [0, \infty)$. To shorten notation, we denote an isoperformance class, $\pi(\{c\})$, by $\pi(c)$.

We now discuss performance preimages in terms of two particular loss functionals. The first one considers the probability of achieving the goal as the objective, while the other incorporates action cost. In general, loss functionals can be considered that incorporate both state and action cost, as is commonly done in stochastic optimal control [9].

Suppose that the following loss is specified:

$$L(x_1, ...x_{K+1}, u_1, ..., u_K, \theta) = \begin{cases} 0 & \text{if } x_{TC} \in G \\ 1 & \text{otherwise} \end{cases}. \quad (14)$$

Equation (14) implies that we are only interested in achieving the goal, without any notion of efficiency in the actual robot trajectory. The loss $\bar{L}(\gamma, \eta_1)$ now represents the probability that the goal will not be achieved using γ. We consider some $\pi([0, c])$ for $c \in [0, 1]$ as a *probabilistic preimage on* N_1. The probabilistic preimage thus indicates places in N_1 from which the goal will be achieved with probability of at least $1 - c$. We could also define the probabilistic preimage on X. We can also consider $\pi(c)$ as an *isoprobability class*. Furthermore, if we effectively remove the termination criterion by assigning $TC_k(\eta_k) = false$ for all $1 \leq k \leq K$ and $\eta_k \in N_k$, and replace the condition "if $x_{TC} \in G$" in (14) by "if $x_k \in G$ for some k" then $\pi([0, c])$ yields a *probabilistic backprojection*, quite similar to that appearing in [3]. The isoprobability class $\pi(0)$ corresponds to the classical preimage notion, in which the goal is guaranteed to be achieved.

This next loss functional directly considers cost associated with executing actions: $L(x_1, ...x_{K+1}, u_1, ..., u_K, \theta) =$

$$\begin{cases} \sum_{k=1}^{k(x_{TC})} l(u_k) & \text{if } x_{TC} \in G \\ C_f & \text{otherwise} \end{cases}. \quad (15)$$

Above, $l(u_k)$ denotes the cost associated with taking action u_k, and $k(x_{TC})$ represents the stage at which TC caused $\mathcal{A}$ to halt. We use C_f to express how important it is to achieve the goal. As C_f approaches infinity, the minimization of (15) becomes equivalent to minimizing (14), and trajectory length is essentially not considered.[2] As C_f becomes less than a typical aggregate action cost that achieves the goal, then strategies will be preferred that do not even expect to achieve the goal.

5 Determining Optimal Strategies

In this section we present computed examples of strategies with termination conditions, $\gamma = (g, TC)$, that optimize an expected loss using either (14) or (15). We also show several examples of performance preimages. The experiments were performed using the model presented in Section 3. The basis of our algorithm is dynamic programming, which is a general optimization concept that has been useful in a variety of contexts, both for producing analytic solutions and for numerical computation procedures.

We now describe the specific choices made for our current implementation. The information structure of $\mathcal{A}$ is chosen as $\eta_k = y_k$, implying that the information space and sensor space are identical, $N_k = Y$. Hence, the robot is memoryless and the robot strategy and termination condition at each stage are limited to functions of the current sensor observation. This choice maintains a low-dimensional information space. For the following expressions, the information space will be referred to as Y; however, the same general theory can be applied to arbitrary information spaces, N_k.

For each point in the information space, y_k, we consider the density, $p(x_k|y_k)$, as the subset of X in which the robot may lie, since (8) specifies a uniform density inside a circle.[3] Let $DISK(y_k^p, \epsilon_p)$ define an open disk in X with radius ϵ_p and center y_k^p. If $y_k^f = \emptyset$, implying that the robot is not in contact with an obstacle, then $p(x_k|y_k)$ can be described as a uniform density on $DISK(y_k^p, \epsilon_p) \cap \mathcal{C}_{free}$. If y_k^f provides a force direction, then $p(x_k|y_k)$ can be described as a uniform density on $DISK(y_k^p, \epsilon_p) \cap \mathcal{C}_{contact}$.

The space of possible subsets of X that correspond to points in the information space is very restricted. We either have some open 2-dimensional subset of $DISK(y^p, \epsilon_p)$ (bounded by edge constraints), or subsets of edges from obstacle boundaries. We currently assume that ϵ_f is sufficiently small so that non-parallel edges that overlap with $DISK(y^p, \epsilon_p)$ can be uniquely identified. In the implementation, we consider consider a 2-dimensional array for each possible sensed position value, y^p. If $DISK(y^p, \epsilon_p)$ does not intersect an obstacle boundary, then there is one entry. Otherwise, there is one element for each edge orientation that could be inferred (for typical problems there is usually one or two).

To ease the consideration of TC in the dynamic programming scheme, we will introduce an additional definition. We allow the robot to have a new action, $\emptyset$, that allows it to do nothing (i.e., perform no action and receive no observation). We have $f(x_k, \emptyset, \theta_k^a) = x_k$. Furthermore, the robot does not perform sensing at the next state when this action is taken, yielding an identical information state. If the action is taken at stage k, then we require that $u_{k+1} \ldots u_K$ all become $\emptyset$. The action $\emptyset$ simulates the effect of the termination condition by causing the robot to remain motionless until stage $K+1$ is reached. Using $\emptyset$, we define an *augmented action space* by $U' = U \cup \{\emptyset\}$, and denote an action at stage k by u'_k.

We will be interested in considering loss that is accumulated at each stage; hence, it is assumed that the loss functional can be expressed as $L(x_1, ...x_{K+1}, u_1, ..., u_K, \theta) =$

$$\sum_{k=1}^{K} l_k(u'_k) + l_{K+1}(x_{K+1}), \tag{16}$$

which includes the loss functionals from Section 4. We state that $l_k(x_k, \emptyset) = 0$, implying that there is no additional loss for choosing $\emptyset$. We could let l_k depend also on the state, x_k, but this extension will not be considered further. The dependency on x_{TC} in any of the loss functionals in Section 4 can replaced by x_{K+1}, when the augmented action space is considered.

The next several equations describe loss and actions in terms of the robot's information space. This leads to the dynamic programming equation, (20), which is a recursive formulation of optimality that is expressed on the information space. Recall the expected loss functionals (10) and (12) from Section 4. We will use a similar concept here; however, we consider the expected loss from that will be incurred from stage k until stage $K+1$. For a given k and one of the loss functionals from Section 4, this is represented using (16) as

$$\bar{L}_k(\gamma_{k..K}, x_k) = \int \left\{ \sum_{i=k}^{K} l_i(u'_i) + l_{K+1}(x_{K+1}) \right\} p(\theta) d\theta. \tag{17}$$

Note that u' depends on θ for a given γ. Using the information state density $p(x_k|y_k)$ on X we have

$$\bar{L}_k(\gamma_{k..K}, y_k) = \int \bar{L}_k(\gamma_{k..K}, x_k) p(x_k|y_k) dx_k. \tag{18}$$

Let $\bar{L}_k^*(y_k)$ denote $\bar{L}_k(\gamma_{k..K}^*, y_k)$ in which $\gamma_{k..K}^*$ is a choice for $\gamma_{k..K}$ that minimizes (18).

We want to consider the effect of taking an action u'_k from a point in the information space, y_k, in terms of a density on Y. This will be similar to the density, $p(x_{k+1}|x_k, u_k)$ that was inferred from the state transition equation in Section 2. The expression is
$p(y_{k+1}|y_k, u'_k) =$

$$\int \int p(y_{k+1}|x_{k+1}) p(x_{k+1}|x_k, u'_k) p(x_k|y_k) dx_k dx_{k+1}. \tag{19}$$

The density $p(x_{k+1}|x_k, u'_k)$ is inferred from the state transition equation. The density $p(x_k|y_k)$ is the conditional density representation of the robot's information state. We compute the integral by generating random samples from $p(x_{k+1}|x_k, u'_k)$ and $p(x_k|y_k)$, and averaging. This

[2] The isoperformance classes will remain the same, but have different indices.

[3] This can be shown by Bayes' rule, with appropriate uniform prior densities on the sensor and state spaces.

method will work for a variety of densities, and for the special case of uniform densities, some geometric computations could alternatively be performed to compute $p(y_{k+1}|y_k, u'_k)$.

Using the previous notation, the dynamic programming principle states that $\bar{L}^*_k(y_k)$ can be obtained from $\bar{L}^*_{k+1}(y_{k+1})$ by the following recurrence:

$$\bar{L}^*_k(y_k) = \min_{u'_k} \left\{ l(u'_k) + \int \bar{L}^*_k(y_{k+1}) p(y_{k+1}|y_k, u'_k) dy_k \right\}. \quad (20)$$

For the loss functional (14) we take $l(u'_k) = 0$. For the loss functional (15), we take $l(u'_k) = \|v\|\Delta t$ if $u'_k \in U$. If $u' = \emptyset$, then the action cost is zero.

At stage $K+1$, we can use the last term of (16) and compute

$$\bar{L}^*_{K+1}(y_{K+1}) = \int l_{K+1}(x_{K+1}) p(x_{K+1}|y_{K+1}) dx_{K+1}. \quad (21)$$

The loss functional $\bar{L}^*_K$ can be determined from $\bar{L}^*_{K+1}$ through (20). If the u'_K that minimizes (20) at y_K belongs to U, then we define $g^*_K(y_K) = u'_K$ and $TC^*_K(y_K) = false$. If the optimal action u'_K is $\emptyset$, then we define $TC^*_K(y_K) = true$. We then apply (20) again, using $\bar{L}^*_K(y_K)$ to obtain $\bar{L}^*_{K-1}$, g^*_{K-1}, and TC^*_{K-1}. This iteration continues until stage $k = 1$. Finally, we take $g^* = \{g^*_1, \ldots, g^*_K\}$ and $TC^* = \{TC^*_1, \ldots, TC^*_K\}$, which comprise γ^*.

The final-stage loss, $\bar{L}^*_{K+1}(y_{K+1})$ is computed from (21). If we are using (14), then the final-stage loss is the probability that $x_{K+1} \in G$ (which can be computed geometrically by considering the percentage of goal that is included in the subset of X in which the robot lies). If we are using (15), then the final-stage loss is product of C_f and the probability that $x_{K+1} \in G$.

If the dynamic programming recursion is iterated for K stages, then all trajectories of length K stages or fewer will be examined. Eventually, the loss values stabilize, and we terminate when $|\bar{L}^*_k(y) - \bar{L}^*_{k+1}(y)|$ becomes small for all y. After the algorithm terminates the resulting stage is designated as $k = 1$, and an explicit prior choice of K is not necessary.

The level sets of the resulting loss $\bar{L}^*_1(y_1)$ yield the isoperformance classes on N_1 for γ. The isoperformance classes on X can then be obtained by considering level sets of

$$\bar{L}^*(x_1) = \int \bar{L}^*_1(y_1) p(y_1|x_1) dy_1. \quad (22)$$

Instead of using (20) to determine the *optimal* γ, we can alternatively use the equation to recursively evaluate a *given* γ. Instead of selecting u'_k to minimize $\bar{L}^*_k$, we fix u'_k for each y_k.

We present several experimental examples in Figures 3 and 4. The box around each problem has dimensions 100×100 in the configuration space. We have chosen the velocity modulus, $\|v\|\Delta t$ to be 2. For the action error we take $\epsilon_\theta = 15$ degrees. The position sensor error radius, ϵ_p is 6. For the determination of optimal strategies, the action set was quantized into four values, similar to what was done in [11] for multi-step plans. For each problem, the obstacles are outlined with thin line segments, and the goal is indicated with a thick line. Isoperformance classes are indicated by a set of contours that correspond to incremental spacing of level sets of $\bar{L}^*_1(x_1)$.

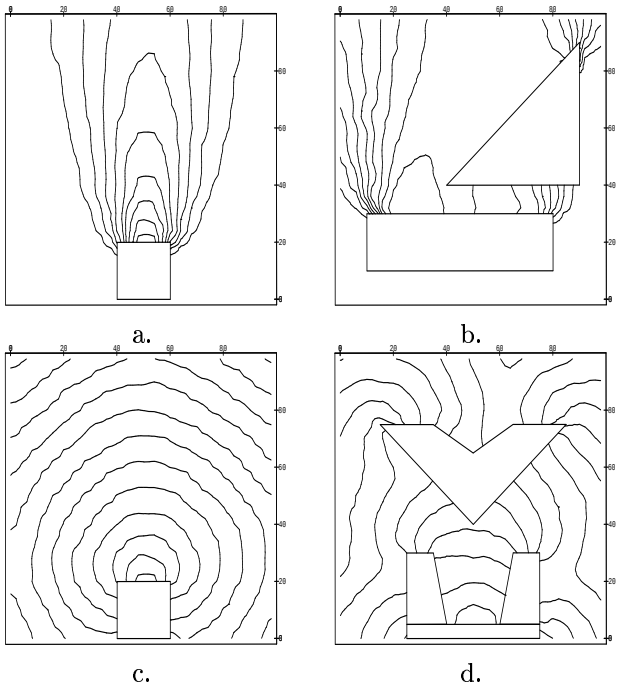

Figure 3. Isoperformance classes represented by contours.

Figures 3.a,b indicate isoperformance classes on X for a strategy that always executes $u_k = \frac{3\pi}{2}$ (i.e., move down). These experiments use (14), representing the probability of failing to terminate in the goal. The lines that are closer to the goal represent lower probabilities. The result in 3.a is similar to that obtained in [3]. For the example in Figure 3.b we observe how performance improves because of compliance, which causes separation to occur in the contours.

The remaining isoperformance contours, in Figures 3.c,d and Figures 4.b,d, correspond to the implementation of the optimal strategy γ^*. These lines tend to emanate radially outward from the goal, as expected loss increases. The results were obtained by using the loss functional (15). If (14) is alternatively used, the $\bar{L}^*_1(x_1)$ becomes approximately 0 (computed as 10^{-6}) for all x_1. Under the implementation of γ^* the behavior of the robot can be considered as a random process. In Figures 4.a and 4.c we show 20 simulated robot trajectories starting from $x_1 = (95, 50)$ that correspond to the optimal strategies depicted by the contours in Figures 4.b and 4.d, respectively. Each one corresponds to a sample path of the random process, and the goal is achieved each time.

6 Conclusions and Extensions

In this paper we described a stochastic framework for manipulation planning in the presence of probabilistic uncertainty in sensing and control. By blending ideas from stochastic optimal control and dynamic game the-

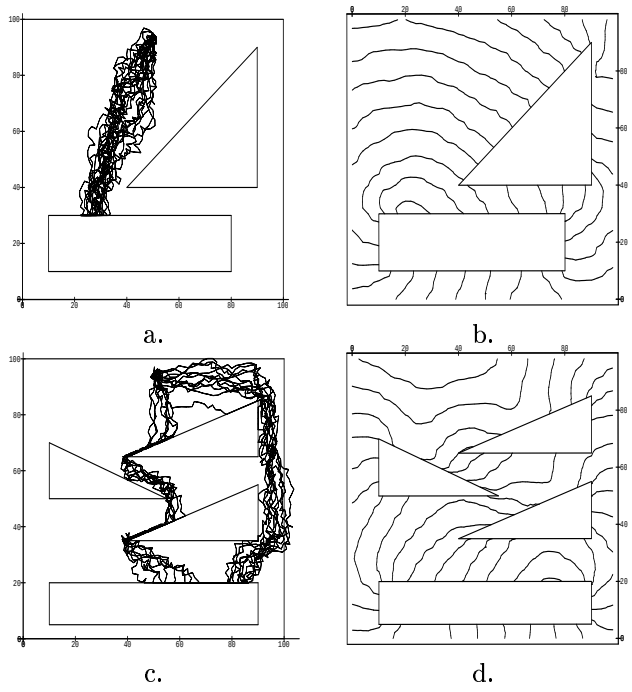

Figure 4. Dynamic programming strategies with sample paths and isoperformance classes.

ory with traditional preimage planning, we have been able to develop general stochastic notions of classical preimage concepts. Furthermore, under a memoryless information structure we have been able to compute optimal strategies and isoperformance classes for a popular robot model via dynamic programming.

A considerable amount of work remains to be done within this formalism. One interesting avenue to explore is to consider the analog of preimage backchaining and subgoals by using performance preimages. We can define G_1 as a subgoal for a larger problem, and define a g and TC that achieves G_1 in a satisfactory way. The resulting posterior density $p(x_{TC})$ would be used as the initial information state for the achievement of a second goal, G_2. We can consider abstract actions of the form $\{G_i, \gamma\}$ that attempt to achieve some original goal. Backchaining from G under explicit performance measures and a given abstract action set appears to be equivalent to dynamic programming. The reason for considering abstract actions and subgoals is the hope that a simple set of abstract actions exists that can be composed to provide quick and efficient solutions for a wide class of problems (as was the case with the backprojection approach to preimage backchaining [6]).

The model for which optimal strategies were obtained in Section 5 can be extended in a number of ways without altering the general approach to the computation. For instance, instead of using compliance, we could use the force sensor as a collision detector that forces the system to halt without necessarily achieving the goal. We can then determine strategies that minimize obstacle contact in an expected sense. Also, different sensor and action nature densities can be substituted; densities could be used that more accurately reflect the error distributions in a particular application. If we are able to efficiently increase the dimension of the information space by one or two degrees, then the state space could be augmented with orientation or velocity, or sensing history might be included.

Acknowledgements

This work is supported by NSF grant #IRI-9110270. We are grateful for the comments provided by the anonymous reviewers.

References

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

[2] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *IEEE Int. Conf. on Robotics and Automation*, pages 1712–1717, 1990.

[3] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A research agenda. In *IEEE Int. Conf. on Robotics and Automation*, pages 3:549–556, 1993.

[4] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *IEEE Int. Conf. on Robotics and Automation*, pages 190–197, Sacramento, CA, April 1991.

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

[6] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, MIT, Cambridge, MA, August 1984.

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

[8] N. V. Krylov. *Controlled diffusion processes.* Springer-Verlag, New York, NY, 1980.

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

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

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

[12] S. M. LaValle and S. A. Hutchinson. Game theory as a unifying structure for a variety of robot tasks. In *IEEE International Symposium on Intelligent Control*, pages 429–434, Chicago, August 1993.

[13] S. M. LaValle and S. A. Hutchinson. Path selection and coordination of multiple robots via Nash equilibria. In *IEEE International Conference on Robotics and Automation*, pages 1847–1852, May 1994.

[14] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic systhesis of fine-motion strategies for robots. *Int. J. of Robot. Res.*, 3(1):3–24, 1984.

[15] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In *Fourth International Symposium on Robotics Research*, pages 421–429, 1987.

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