March 1988 Report No. STAN-CS-88-1196

Motion Planning With Uncertainty: The Preimage Backchaining Approach

by

Jean-Claude Latombe

Department of Computer Science

Stanford University

Stanford, California **94305**

Motion Planning With Uncertainty: The Preimage Backchaining Approach

Jean-Claude Latombc
Robotics Laboratory, Computer Science Department
Stanford University

Abstract

This paper addresses the problem of planning robot motions in the presence of uncertainty. It explores an approach to this problem, known as the *preimage backchaining approach.*. Basically, a preimage is a region in space, such that if the robot executes a certain motion command from within this region, it is guaranteed to attain a target and to terminate into it. Preimagc backchainin*g* consists of reasoning backward from a given goal region, by computing prcimages of the goal, and then recursively prcimages of the prcimages, until-some preimages include the initial region where it is known at planning time that t.he robot will be before executing the motion plan. In the paper, we first give a rigorous formalization of the problem of planning motions in the presence of uncertainty; such a formalization is necessary because in many regards reasoning with uncertainty is not reducible to straightforward intuition. Then, we investigate in detail the theory of the preimage backchaining approach; we give a new presentation of preimages, we explore the notion of maximal preimages, and we extend the framework to the generation of conditional motion strategies. Finally, we describe a complete set of algorithms that makes it possible implementing the approach in a simplified two-dimensional world, which we call the *mini-world*. The restrictions imposed on the mini-world arc essentially aimed at reducing the conceptual and computational complexity of the geometric computations required by the preimage backchaining approach. Ncvcrthclcss, the mini-world is still appropriate to handle realistic navigation problems with omni-directional mobile robots.

Key-Words: Spatial Reasoning, Robot Planning, Motion Planning, Planning in the Presence of Uncertainty, Preimagc Backchaining.

Acknowledgements: This paper describes research done at the Robotics Laboratory of Stanford University (Computer Science Department). Support for the research as been provided in part by the Stanford Institute for Manufacturing and Automation (SIMA), and in part by the Center of Integrated Systems (CIS). The author also thanks Shashank Shckhar and Tom Strat for their useful comments on an carlier draft of the paper.

Contents

1 Introduction

In this paper, we address the problem of planning robot motions in the presence of uncertainty. In principle, the robot may be any kind of rigid or articulated object capable of controlling its motions within a workspace. In particular, it may be a manipulator arm, a multi-joint multi-finger hand, a wheeled vehicle, or a free-flying vehicle. In practice, however, the complexity of the motion planning problem augments exponentially with the number of degrees of freedom of the robot system [45,48,6].

Motion planning in the presence of uncertainty is one of the important problems that we have to solve in order to create autonomous robots [20]. By autonomous robots we mean robots that are both automatic – i.e., that can execute tasks in the physical workspace without human intervention –, and taskable – i.e., that accept high level task descriptions. Such a description typically specifies *what* the user wants done rather than how to do it. Therefore, at some level of reasoning, an autonomous robot has to plan the motion commands and the sensing acts that are appropriate to achieve the goals, and it must monitor their execution. Examples of sub-tasks that usually require motion plans taking uncertainty into consideration are: grasping an object with the end-effector of a manipulator robot, mating two mechanical parts in an assembly process, and navigating from one location to another in an in-door environment. In this paper, we consider the generic task of planning the motions of a single controlled object (i.e., the robot) among fixed, un-movable, and rigid obstacles, from an initial region (a single location if there was no uncertainty), where it is known that the moving object will be before executing the plan, to a goal region. We distinguish among three types of uncertainty: uncertainty on robot control (the robot does not execute motions exactly as they are specified), uncertainty on dimensions and locations of obstacles in the initial world, and uncertainty on on-line sensing. However, most of the paper concentrates on uncertainty on control and on sensing.

The solution to a motion planning problem without uncertainty is the geometrical description of a collision-free path of the robot from its initial location to a goal one. The solution to a motion planning problem with uncertainty *is a motion strategy*. Typically, a motion strategy is an algorithm including motion and sensing commands, which takes advantage of various sources of information (e.g., model of the motions, prior model of the world, on-line sensing) in order to reduce uncertainty and lead the robot to the goal position. Thus, a strategy may include motion commands merely aimed at acquiring new pieces of information. However, reaching a goal position is the only imposed goal. Reducing uncertainty is important only when it is a prerequisite to achieving this goal. Although in this paper we consider motion strategies using sensing, this is not always a requirement. For instance, Erdmann and Mason [18] investigate sensorless strategies capable of dealing with uncertainty. However, such strategies require reasoning about operations, such as pushing and sliding, which involve several independent moving objects. Planning sensory-based strategies requires reasoning at planning time about pieces of information that will be known (with some uncertainty) only at execution time.

In this paper, we focus on an approach to motion planning with uncertainty known as the *preimage backchaining approach*. Basically, a preimage of a target for a given motion command is a region in space, such that if the robot executes the motion command from within this region, it is guaranteed to attain the target and to terminate into it despite uncertainty; terminating the motion in the target typically requires sensory-based recognition capabilities. Preimage backchaining consists of reasoning backward from a given goal. A search graph is built and explored by computing preimages of the goal for different motion commands, and then preimages of the preimages, until some preimages include the initial region. The preimage backchaining approach has been first introduced by Lozano-Pérez, Mason and Taylor [35], with subsequent contributions by Mason [39], Erdmann [17,18], and Donald [10,12].

The contribution of this paper is threefold. First, it gives a rigorous formalization of the problem of planning motions in the presence of uncertainty (Sections 2 through 5); such a formalization is necessary because in many regards reasoning with uncertainty is not reducible to straightforward intuition. Second, it brings new fundamental insights in the theory of the preimage backchaining approach (Sections 6 through 13, and Section 17); in particular, it introduces a new formal definition of preimages (Section 8), which, we believe, is clearer than the one used in previous papers; based on this definition, it explores the notion of maximal preimage (Sections 9 through 12); it also extends the formal framework of preimage backchaining to the generation of conditional strategies (Section 17). Third, the paper describes a complete set of algorithms that makes it possible implementing the approach in a simplified two-dimensional world, which we call the *mini-world* (Sections 14 through 16); although rather simple, the mini-world is still realistic enough for some applications: for instance, it can be the world of an omni-directional mobile robots with a polygonal outline moving among obstacles bounded by polygonal outlines. Throughout the paper, we use examples in the mini-world to illustrate our presentation; the restrictions imposed on the mini-world are essentially aimed at reducing the conceptual and computational complexity of the required geometric computations. A final section (Section 18) relates our presentation to previous work.

During the last five years, a trend in research on autonomous agents interacting with a dynamic and/or uncertain external world has been toward “reactive planning” (e.g., see [22]). This trend grew up in reaction against the more traditional approach to planning, which tends to decompose planning and execution between two successive phases. A new extreme position related to this trend is to use no prediction of future states at all. A criticizable effect of such a position is to produce planners that produce plans with no provable properties relative to their correctness (but is there any more a plan?). We think that planning is an essential capability of an autonomous agent in order to display intelligent behavior, and whether it is performed off-line or on-line, it should produce plans with well-defined properties, so that if their execution fails, it is possible to diagnose why. Such a diagnosis is not important for correcting the plan (since it already failed), but to determine the assumptions which were wrong, i.e. to learn from failure.

In. this paper, we concentrate our presentation on the planning of *strongly guaranteed* strategies. Strategies of this class arc guaranteed to succeed whenever errors on control, model, and sensing remain within predefined bounds specifying uncertainty. If such a strategy fails, it means that one error exceeded these bounds during execution. Despite some drawbacks (e.g., some motion planning problems may admit no strongly guaranteed solution, or only complex ones), these strategies are most appropriate when off-line planning is prefered (e.g. in the context of industrial manufacturing [28]), or when on-line interaction between the controller and the planner is limited (e.g. by the bandwidth of a radio link). They also can be used on-line to plan motions to *achieve* short-term goals. In addition, from **a** theoretical point of view, strongly guaranteed strategies raise many interesting questions leading to study theoretical concepts with broader relevance. In the conclusion, we will introduce a weaker (i.e., larger) class of motion strategies, which still has provable properties, while being more adapted to reactive planning.

2 Modeling Task Geometry

Let us consider an object, $\mathcal{A}$[1], moving in a euclidean space called **workspace.** Any list of parameters that completely specifies the position of every point on $\mathcal{A}$ at any instant t with respect to a fixed Cartesian coordinate system $\mathcal{W}$ in the workspace defines a space called the **configuration space** of $\mathcal{A}$ [34]. Any point in this latter space (i.e., any instantiated list of parameters) is called a **configuration** of A.

There is an infinity of possible configuration spaces for A. We assume that one of them, denoted C, has been arbitrarily selected as *the* configuration space of **A.** At every instant, the mapping of $\mathcal{A}$ into C is a point, P, called the **effector point.** In the following, d(c) denotes the region occupied by $\mathcal{A}$ in the workspace, when P's position in C (i.e., d's configuration) is c.

Example 1: Figure 1 shows several examples of configuration spaces:

- Figure 1 a: The configuration space of **a** two-dimensional rigid object $\mathcal{A}$ that can only **translate** in the plane is $\Re^2$ (more precisely, **it** is isomorphic to $\Re^2$). A configuration (x, y) consists of the coordinates of **a** fixed reference point on $\mathcal{A}$ with respect to $\mathcal{W}$.

- Figure **1** *b*: The configuration space of a two-dimensional rigid object $\mathcal{A}$ that can both translate and rotate in the plane is $\Re^2$ x S', where S^1 is the unit circle. A configuration (x, y, θ) consists of the two coordinates of **a** fixed reference point on $\mathcal{A}$ and the orientation

[1] A table of symbols is given in Appendix at the end of the paper.

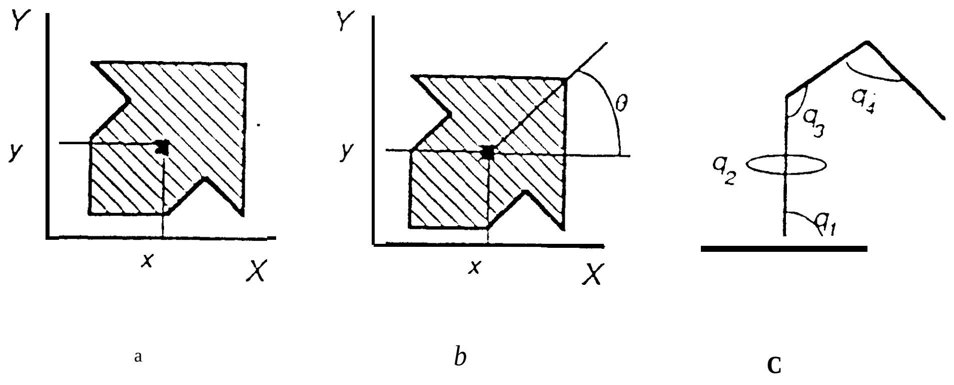


Figure 1: **Configuration Space Examples**

of a fixed reference axis with respect to $\mathcal{W}$. Similarly, if $\mathcal{A}$ is a three-dimensiona. rigid object allowed to translate and rotate without restriction, C = $\Re^3$ x $SO(3)$, where SO(3) is the three-dimensional Special Orthogonal Group. Then, if orientation is represented using the Euler angles (ψ, θ, ϕ), a configuration is the list $(x, y, z, \psi, \theta, \phi)$.

- Figure 1 c: The configuration space of an articulated object with N rotating joints is a subspace of $(S^1)^N$ = S^1 x . . . x S'. A configuration is a N-dimensional list $(q_1, q_2, \ldots . q_N)$, each parameter q_i specifying one joint angle. ∎

At this stage, a trajectory of $\mathcal{A}$ in the workspace can be described as a mapping τ : $t \in \Re \rightarrow c \in \mathcal{C}$. It can also be represented as *a* curve in configuration space x **time c x $\Re$.**

Now, let us assume that d's workspace includes fixed obstacles $\mathcal{B}_i$, $i = 1, 2, \ldots$. The region occupied by each obstacle $\mathcal{B}_i$ in the workspace maps into C as another region called a **C-obstacle** and denoted $\mathcal{CB}_i$[2]. By definition, $\mathcal{CB}_i = \{c \in C \ / \ d(c) \cap \mathcal{B}_i \neq \emptyset\}$.

Example 2: Figure 2 illustrates the case where both $\mathcal{A}$ and $\mathcal{B}_i$ are convex polygona. regions, $\mathcal{A}$ being only allowed to translate. The configuration of $\mathcal{A}$ is defined as the coordinates of point **P** (when $\mathcal{A}$ is a rigid object only allowed to translate, the effector point and the reference point coincide). The curve followed by **P** when $\mathcal{A}$ slides in contact with $\mathcal{B}_i$'s boundary, without overlapping of d's and $\mathcal{B}_i$'s interiors is the boundary of $\mathcal{CB}_i$. **It** can be proved that $\mathcal{CB}_i$ is also **a** convex polygon [34].

If we also **allow $\mathcal{A}$ to** rotate, then $\mathcal{CB}_i$ is *a* volume in $\Re^2$ x S^1 = $\{(z, y, \theta)\}$. Each cut through $\mathcal{CB}_i$ perpendicular to the θ-axis of C is a convex polygon. However, $\mathcal{CB}_i$ is bounded by curved surface patches (more precisely, ruled surfaces) [34]. I

[2]A connected region $\mathcal{B}_i$ may map into C as a non-connected region *CB*;.

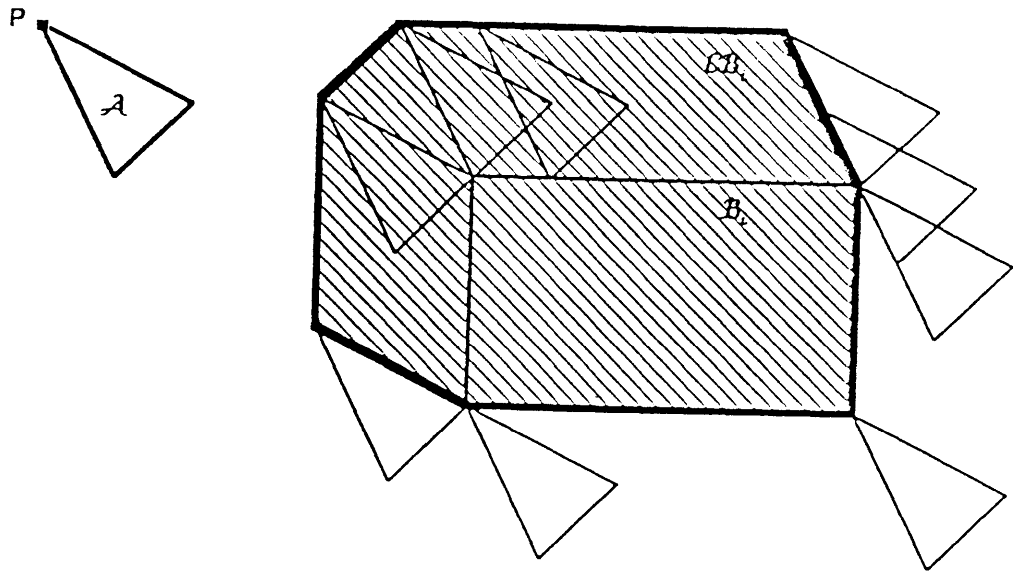


Figure **2: Mapping an Obstacle in Configuration Space**

Several practical methods exist for computing either the exact or an approximate representation of a C-obstacle, when both $\mathcal{A}$ and $\mathcal{B}_i$ are polyhedral (or polygonal) objects [34,3,23,9,30]. In particular, Donald [9] describes a method for computing the mapping of polyhedral obstacles, when $\mathcal{A}$ is a rigid polyhedral object allowed to both translate and rotate.

If the obstacles $\mathcal{B}_i$ are mobile obstacles, then it is possible to map the regions they occupy in $\mathcal{A}$'s workspace into regions of configuration space x time $\mathcal{C} \times \Re$. Each cut $\mathcal{C} \times \{t\}$ perpendicular to the time axis includes the C-obstacle $\mathcal{CB}_i$ at time t. In the rest of the paper, we only consider fixed un-movable obstacles.

Let $\mathcal{C}_{free} = \{c \in \mathcal{C} \,/\, \mathcal{A}(c) \cap (\bigcup \mathcal{B}_i) = \emptyset\} = \mathcal{C} - \bigcup \mathcal{CB}_i$. $\mathcal{C}_{free}$ is called **free space.** Whenever the effector point **P** is in free space, it means that $\mathcal{A}$ has no contact with any obstacle $\mathcal{B}_i$. Let $\mathcal{C}_{contact} = \{c \in \mathcal{C} \,/\, \mathcal{A}(c) \cap (\bigcup \mathcal{B}_i) \neq \emptyset$ and $\mathcal{A}(c) \cap (\bigcup \mathcal{B}_i) \subseteq \partial(\bigcup \mathcal{B}_i)\}$, where $\partial \mathcal{S}$ denotes the boundary of the closed region $\mathcal{S}$ [3]. $\mathcal{C}_{contact}$ is called **contact space.** Whenever **P** is in contact space, it means that $\mathcal{A}$ has made a contact with one or several obstacles $\mathcal{B}_i$. We always have $\partial(\bigcup \mathcal{CB}_i) \subseteq \mathcal{C}_{contact}$, but, as illustrated by Figure 3 (there is no clearance between $\mathcal{A}$ and $\mathcal{B}$'s hole), it may happen that $\mathcal{C}_{contact} \neq \partial(\bigcup \mathcal{CB}_i)$.

Mapping the geometry of the task into configuration space allows us to transform the problem of planning the motion of a dimensioned object into that of planning the motion of **a** point, P, from an **initial region** $\mathcal{I}$ to a **goal region** $\mathcal{G}$. Both $\mathcal{I}$ and $\mathcal{G}$ are subsets

[3]We assume that physical objects occupy closed bounded regions in the workspace. We use the same symbols, $\mathcal{A}$ and $\mathcal{B}_i$, to denote both the physical objects and the regions they occupy.

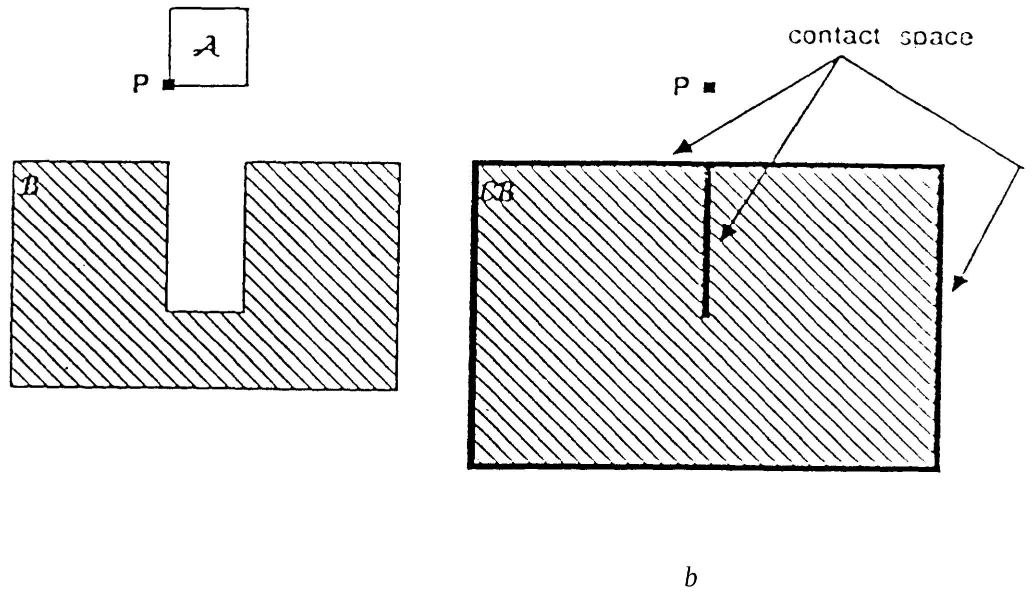


Figure **3: Contact Space**

of C. The motion path of **P** is constrained to lie entirely in $C_{free} \cup C_{contact}$. Thus, configuration space makes explicit the geometrical constraints imposed on the motion of $\mathcal{A}$ by the obstacles. However, it is easy to verify that different geometries in the workspace may result in the same geometry **in** configuration space; so, the mapping between workspace and configuration space is not a bijective one.

In the rest of this paper, we mainly consider **a** simple two-dimensional (2D) configuration space (x, y), called the **mini-world,** to which some restrictions apply. In particular, there is a finite number of C-obstacles in C, and every C-obstacle *CB;* is a polygonal region; $C_{contact}$ consists of a finite number of finite straight segments. The other restrictions will be stated when relevant in further sections. Our presentation of the preimage backchaining approach directly applies to the mini-world, and all the illustrating examples take place in the mini-world (eventually with slight indicated differences). Although most of this presentation remains valid in higher-dimensional configuration spaces, certain modeling aspects and geometrical computations, not treated in this paper, are made considerably more complex by increasing C's dimension. Some geometrical computation problems in higher-dimensional spaces are even still completely unexplored.

3 Modeling Task Physics

We are interested **in** planning motions with uncertainty. In particular (see Section 5), the motion of the effector point **P may** not be controlled perfectly. In addition, the geometry of the workspace, and so the geometry of $\mathcal{C}$, may not be known exactly. Due to uncertainty, it may be useful (or necessary) to include sensing acts other than position sensing in motion plans. However, the use of some sensors requires dealing with more that just geometry. Certain physical properties of the workspace have to be modeled, and mapped into constructs in configuration space. For instance, using visual sensing may entail representing reflectance properties of obstacle surfaces.

In this paper, we assume that the robot is equiped with two sensors only, the *position sensor,* which gives the current configuration of $\mathcal{A}$, and the force *sensor*, which measures the reaction force generated by obstacles when d pushes on them. Using force sensing requires mapping forces into configuration space. The rest of this section describes how wrenches (combination of forces and moments) resulting from the contact of $\mathcal{A}$ with actual obstacles can be mapped into C as generalized force vectors resulting from the corresponding contact of P with C-obstacles. Our description is inspired from Erdmann's work [17], where more detail can be found.

A wrench *(F, M)* applied to (or by) $\mathcal{A}$ is mapped into C as a force vector f applied to (or by) P. The component of f along each parameter axis of C is proportional to the acceleration of $\mathcal{A}$ caused by the wrench along the degree of freedom corresponding to this axis. For instance, in the configuration space C = (x, y, θ) of a rigid 2D object, a force vector is made of three components respectively proportional to the linear acceleration of $\mathcal{A}$ along the x- and y-axes, and to the rotational acceleration about the Q-axis.

Let us assume that d's and $\mathcal{B}_i$'s boundaries are both frictionless. When there is no contact between $\mathcal{A}$ and any of the $\mathcal{B}_i$, then $\mathcal{A}$ cannot exert any force on its environment, so the reaction force on $\mathcal{A}$ is null. Correspondingly, when P is in free space, the reaction force on P is null. When there is a contact between $\mathcal{A}$ and an obstacle $\mathcal{B}_i$, if $\mathcal{A}$ pushes *on* $\mathcal{B}_i$, then $\mathcal{B}_i$ pushes back. It turns out that, in configuration space, P and $\mathcal{CB}_i$ behave in the same manner. The generalized force exerted by $\mathcal{A}$ on $\mathcal{B}_i$ is mapped into C as a vector $\mathbf{f}_{appl}$ applied by P. It can be proved that the reaction wrench exerted by $\mathcal{B}_i$ on $\mathcal{A}$ maps into C as a force vector, $\mathbf{f}_{react}$, which is perpendicular to the boundary of $\mathcal{CB}_i$ at the current position of P. We say that $\mathcal{CB}_i$ reacts to $\mathbf{f}_{appl}$ by generating $\mathbf{f}_{react}$. If $\mathbf{f}_{appl}$ is perpendicular to the boundary of $\mathcal{CB}_i$, then $\mathbf{f}_{react} = -\mathbf{f}_{appl}$.

Let us now consider the case when the surfaces produce friction. A classical mode1 of friction on a surface in the workspace, based on Coulomb's law, is known as the *friction cone* (in fact, it is a half-cone). The cone's ***axis is*** normal to the surface at the considered point (see Figure 4 *a*); its extreme rays make an angle $tan^{-1}\mu$ with this axis, where μ is the coefficient of friction (we assume the same value for the static and dynamic coefficients). An applied force that points toward the surface inside the cone causes the generation of an opposite reaction force having the same intensity (see Figure 4 6). An applied force that points toward the surface outside the friction cone results in a reaction force along one extreme ray of the friction cone (see Figure 4 c); then, the resulting net force is tangent to

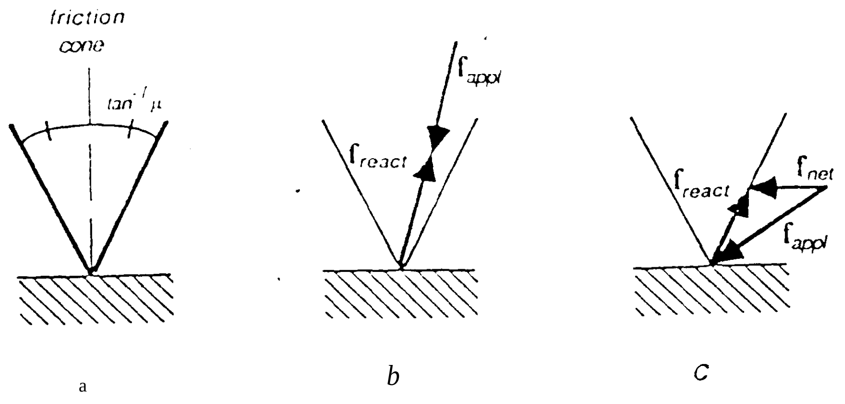


Figure **4: Illustration of the Friction Cone**

the surface.

The notion of friction cone in the workspace extends easily to the configuration space of a translating object A. In such a space, at any point on the surface of a C-obstacle CO;, friction can be modeled using a friction cone. The angle of this cone derives from the friction coefficients of the actual surfaces in contact. The applied force, the reaction force, and the friction cone in configuration space are related in the same fashion as in the workspace. The friction cone at a point on a C-obstacle boundary thus specifies the range of possible orientations of the reaction force on P at that point. Erdmann [17] discusses friction representation when d can also rotate. $\forall c^* \in \mathcal{C}_{free} \cup \mathcal{C}_{contact}$, we denote $\mathcal{F}^*(c^*)$ the range of reaction force that can be generated at position c^* [4]. If $c^* \in \mathcal{C}_{free}$, then $\mathcal{F}^*(c^*) = \{0\}$.

In our mini-world, the friction coefficient is constant along every edge of every C-obstacle. Thus, both the angle and the orientation of the friction cone remain constant along an edge. If the edge is frictionless, then $\mu = 0$ and the cone reduces to its axis. The friction cone associated with every C-obstacle vertex is the cone, the sides of which are the two most extreme rays of the cones associated with the adjacent edges. Thus, WC assume that when P is in $\mathcal{C}_{contact}$ at a C-obstacle's vertex, the reaction force generated by the C-obstacle can be any non-negative linear combination of the reaction forces that can be generated by the two adjacent edges. Figure 5 illustrates friction cones in the mini-world. $\forall c^* \in \mathcal{C}_{contact}$: $\nu(c^*)$ denotes the unit vector pointing along the axis of the friction cone, and $2\phi(c^*)$ denotes the angle of the friction cone. Let $\mathcal{E}$ be an edge in $\mathcal{C}_{contact}$; $\nu(\mathcal{E})$ denotes both the unit outgoing normal vector to $\mathcal{E}$, and the unit vector pointing along the axis of the friction cone at any position on $\mathcal{E}$. In the mini-world, $\forall c^* \in \mathcal{C}_{contact}$, $\mathcal{F}^*(c^*)$ **is** the set of all vectors included in the friction cone at c^*.

[4]As it will be explained further, c^* denotes an actual position of P in C, while c denotes a measured position.

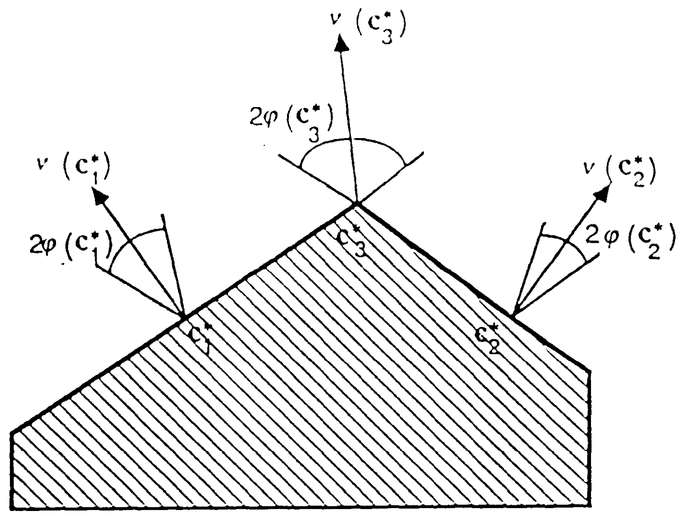


Figure **5: Friction Cones in the Mini-World**

We denote **f** the value of the reaction force on **P,** which is measured by the force sensor.

4 Motion Commands .

A solution to a motion planning problem in configuration space is a plan including motion commands expressing intended motions of the effector point **P.** If there were no uncertainty, one could consider formulating motion commands as geometrical paths (continuous sequence of configurations) to be followed by **P** at execution time. However, since we address the motion planning problem in the presence of uncertainty, we consider slightly more sophisticated commands, called generalized motion commands.

A generalized motion command M is one of the form M = **(CS,TC),** where:

- **CS** is the **control statement** specifying the (possibly infinite) trajectory along which the controller executing the command has to move **P,**
- **TC is** the **termination condition** specifying the condition upon which the controller should terminate the motion of P.

The concept of control statement is illustrated by the following two examples.

Example 3: One **type** of control statement is *pure* **velocity** *control.* **A** velocity v is specified, and executing the motion command causes **P** to move along a straight line in **C,** with constant velocity v, when **P** is in free space and when **P** is in contact space with v either pointing toward the outside of the C-obstacle or tangent to **its** boundary. The

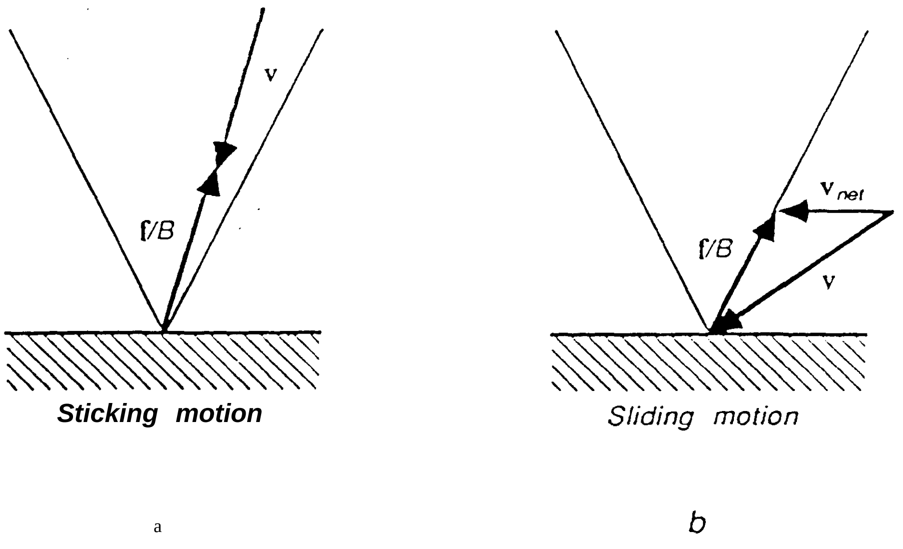


Figure **6: Sticking** and **Sliding Generalized Damper Motions**

motion command causes no motion of P when it is in contact space, with v pointing toward the inside of the C-obstacle. ∎

Example 4: Another type of control statement, which makes use of force sensing, is generalized *damper* control. The corresponding equation in C for this type of control **is** $f = B(\mathbf{v}_{net} - v)$, where v is the specified commanded velocity, **f** is the reaction force on P, and $\mathbf{v}_{net}$ is the net velocity of P; *B* is a constant, called the damper constant, which relates velocities to forces. When P is in free space and when P is in contact space with v either pointing toward the outside of the C-obstacle or tangent to its boundary, $\mathbf{f} = 0$, so that $\mathbf{v}_{net} = v$. Then, as with pure velocity control, generalized damper control along v causes P to move along a straight line with constant velocity v. When P is in contact space with v pointing toward the inside of the C-obstacle, $\mathbf{f} \neq 0$, so that $\mathbf{v}_{net} \neq v$. Then, two cases are possible: either *v points inside*[5] the friction cone at the current position of P (see Figure 6 *a*), and P sticks to the boundary of the C-obstacle (no motion); or v points outside the friction cone (see Figure 6 *b*), and P slides tangentially to the C-obstacle boundary. ∎

Notice that both pure velocity control and generalized damper control, as we described them, are ideal approximations of the behavior of actual controllers. Indeed, both suppose that the controller can change the robot's velocity instantaneously. Obviously this is impossible with an object that has non-null mass. This approximation is one source of error resulting in control uncertainty.

There are many other possible types of control statements than those presented in the above examples. However, in our mini-world, we only consider those two. We denote pure velocity control with commanded velocity v by V(v), and generalized damper control

[5]We say that v points inside the friction cone if the vector $-\mathbf{v}$ originating at the cone's apex is contained in the friction cone.

with commanded velocity $\mathbf{v}$ by $GD(\mathbf{v})$. Generalized damper is one sort of force-based *compliance*, which has received considerable attention in the Robotics literature (e.g., see [44,38,55,26] for more detail). In particular, Mason [38] analyzes generalized damper control in configuration space. Khatib [26] introduces the concept of "operational space", which is similar to configuration space in the case of a rigid moving object, and formalizes the dynamic equations of a manipulator arm in this space; he applies this formalization to define a hybrid position/force motion controller. Buckley [5] investigates generalized spring control, another type of force-based compliant control, in configuration space; he applies this type of control for planning motions of objects from contact to contact.

The termination condition TC is an expression of the general form $tp(\delta t, \mathbf{c}_{[0,\delta t]}, \mathbf{f}_{[0,\delta t]})$, where: tp is a predicate; δt is the elapsed time since the beginning of the motion; $\mathbf{c}_{[0,\delta t]}$ and $\mathbf{f}_{[0,\delta t]}$ are the records of position and force sensing since the beginning of the motion. Examples of termination conditions are $[\delta t > T_0]$ and $[\mathbf{c}(\delta t) \in S$ and $angle(n_0, \mathbf{f}(\delta t)) = 0]$. $[\delta t > T_0]$ means that the motion has to be terminated when its duration exceeds T_0. $[\mathbf{c}(\delta t) \in S$ and $angle(n_0, \mathbf{f}(\delta t)) = 0]$ means that the motion has to be terminated when the measured configuration is in region S and the measured force makes a null angle with the given vector n_0.

Notice that a termination condition may not be guaranteed to ever terminate a motion. A particular case occurs when the motion physically stops by sticking against an obstacle, while the termination condition does not recognize this situation (because it was not anticipated at planning time). Then, although there is no more motion in the physical sense, the controller does not know it and does not execute the next step in the motion plan.

5 Modeling Uncertainty

When considering a real robot operating in a real world, one has to take into account possible errors arising from many different sources. It has become rather common to group possible errors into three different types: *control* errors, *model* errors, and *sensing errors*. Control errors result from the fact that no robot controller is perfect; for instance, executing a motion command with CS = V(v) does not cause P to move exactly along v in free space. Model errors arise from our inability to have an exact model of the workspace (e.g., we cannot know the exact dimensions of the objects in the workspace). Sensing errors are inherent to the fact that sensors are measuring devices that have limited precision.

Let us consider that every error applies to the value of a parameter. Given the nominal value $\mathbf{p}$ of a parameter p, the actual value $\mathbf{p}^*$ of this parameter belongs to a set $\mathcal{U}_p(\mathbf{p})$. We call this set the **uncertainty** on the value of p. The set may be bounded or not, discrete or not, finite or not. We assume **a** uniform probabilistic distribution of the actual value of p over this set[6]. In **our** notations, we distinguish the actual value of a parameter from its

[6]This assumption is directly related to our focus ***on*** (strongly) guaranteed strategies. Other types of prob-

nominal value by using a star (*) as exponent.

In the following, $\mathcal{U}_{CS}(CS)$, $\mathcal{U}_{\mathcal{C}}(\mathcal{C})$, $\mathcal{U}_c(\mathbf{c})$, and $\mathcal{U}_f(\mathbf{f})$ denote the functions specifying uncertainty on control, model, position sensing, and force sensing. Below, we specify a possible representation of these functions in the mini-world. In the rest of the paper, however, we will assume no model error, i.e. $\mathcal{U}_{\mathcal{C}}(\mathcal{C}) = \{\mathcal{C}\}$.

Control uncertainty: Let $\mathbf{v}$ be the specified (i.e., nominal) commanded velocity in either V(v) or $\mathbf{GD(v)}$. At any instant during the execution of the motion command, the actual commanded velocity $\mathbf{v}^* \in \mathcal{U}_v(\mathbf{v})$, such that:
- $angle(\mathbf{v}^*, \mathbf{v}) \leq \theta_v$,
- $\|\mathbf{v}^*\| \in \Delta_v(\mathbf{v})$, an interval including $\|\mathbf{v}\|$,
where $angle(\mathbf{v}^*, \mathbf{v})$ evaluates to the angle between $\mathbf{v}^*$ and v, and $\|\mathbf{v}\|$ evaluates to the module of v. (Note that $\mathbf{v}^*$ may not he constant during the motion.)

Thus, at each instant, the orientation of $\mathbf{v}^*$ is within a half-cone, called the velocity cone. This cone's apex is at the current position of P; its axis points along the direction of v; its extreme rays make an angle θ_v with this axis.

If $\mathbf{CS} =$ V(v) or $\mathbf{GD(v)}$ and P is in free space, the actual velocity of $\mathbf{P}$ is $\mathbf{v}^*$.

If CS = GE)(V) and P is in contact space, the actual velocity of $\mathbf{P}$ is $\mathbf{v}^*_{net} = \mathbf{f}^*/B + \mathbf{v}^*$ **(see** Example 4). Let the *negative velocity cone* be the half-cone symctrical to the velocity cone with respect to the apex. If the negative velocity cone is contained in the friction cone at **P's** current position, sticking is guaranteed (i.e., $\mathbf{v}^*_{net} = 0$), because $\mathbf{P}$ is guaranteed to push against the C-obstacle within the friction cone; if the two cones have no intersection [except their common apex), then sliding or moving away is guaranteed; in all other cases, sticking is possible, but not certain. Notice that testing whether P may slide or stick on a C-obstacle's edge in the mini-world is made particularly simple by the fact that the orientation of the friction cone remains constant along the edge. The test is illustrated by Figure 7.

In the mini-world, both $\mathcal{U}_{CS}(\mathbf{V(v)})$ and $\mathcal{U}_{CS}(\mathbf{GD(v)})$ arc denoted $\mathcal{U}_v(\mathbf{v})$.

Model **uncertainty:** Consider the configuration space with a single C-obstacle $\mathcal{CB}$, as shown at Figure 8 a. Assume that one of the dimensions of $\mathcal{CB}$, d, is not precisely known. Uncertainty on d can be defined by $\mathcal{U}_d = [d_{min}, d_{max}]$. One way to represent such uncertainty **is** to extend the configuration space into a *generalized configuration space (see* Figure 8 6) by adding one extra-dimension corresponding to the d parameter. Each cut perpendicular to the d-axis corresponds to **a** possible configuration space. The problem is still to move **a** point, P. However, the difference with regular configuration space is not so

abilistic distributions have been used to model uncertainty in Robotics (e.g., see [16,49]). More sophisticated distributions could be used in addition to the uniform one in order for example to evaluate the probability that a strongly guaranteed strategy will perform correctly or to guide trouble-shooting if the guaranteed strategy happens to fail.

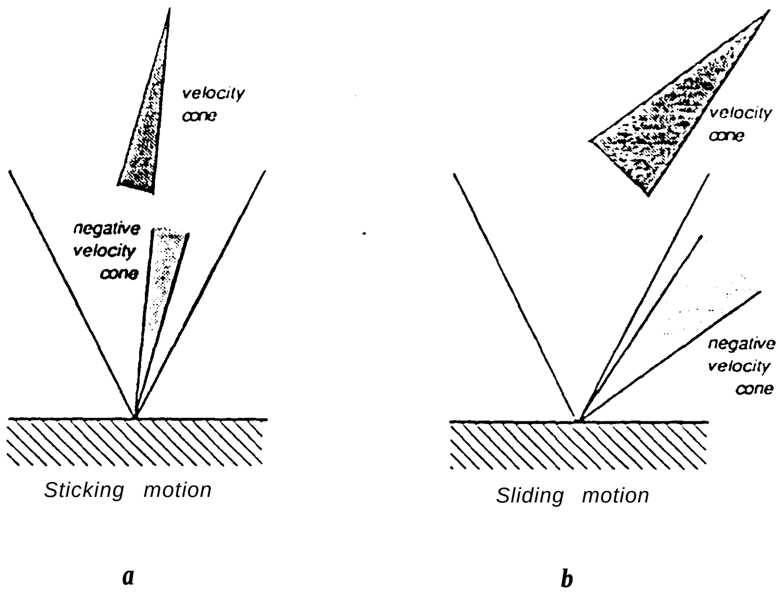


Figure **7: Sticking Test With Control** Uncertainty

much that the new space is three-dimensional (x, y, d); it is that **P** can only be controlled along two of its axes (x and y). Indeed, as long as C-obstacles are rigid and un-movable, **P** is constrained to move within one single plane perpendicular to the d-axis; but we do not know the d-coordinate of this plane within the range $[d_{min}, d_{max}]$.

This technique for representing uncertainty on configuration space geometry can be applied to N parameters ($N \geq 1$), by adding N axes to configuration space. Parameters need not be continuous ones. They may also take their values from discrete and finite sets. Let $\mathcal{GC}$ be the resulting generalized configuration space; $\mathcal{U}_{\mathcal{C}}(\mathcal{C}) = \mathcal{GC}$.

As investigated' by Donald [10,12], most of the preimage backchaining approach can be extended to such a generalized configuration space. However, in the rest of the paper, and in the mini-world in particular, we assume that C's geometry is perfectly known.

It is easy to model uncertainty on the friction cone, by defining a small cone and a large cone. The only impact is on the sticking test illustrated by Figure 7. Sticking is guaranteed only if the negative velocity cone is contained in the small friction cone; sliding is guaranteed only if the negative velocity cone and the large cone have null intersection. In the following, we only consider cases where sliding has to be guaranteed. Therefore, we assume no uncertainty on friction cones (alternatively, we can think of using the large friction cone only).

Sensing uncertainty: Let **c** and **f** be the position and force measured by the sensors at some instant. At the same instant, the actual position and force, $\mathbf{c}^* \in \mathcal{U}_c(\mathbf{c})$ and $\mathbf{f}^* \in \mathcal{U}_f(\mathbf{f})$,

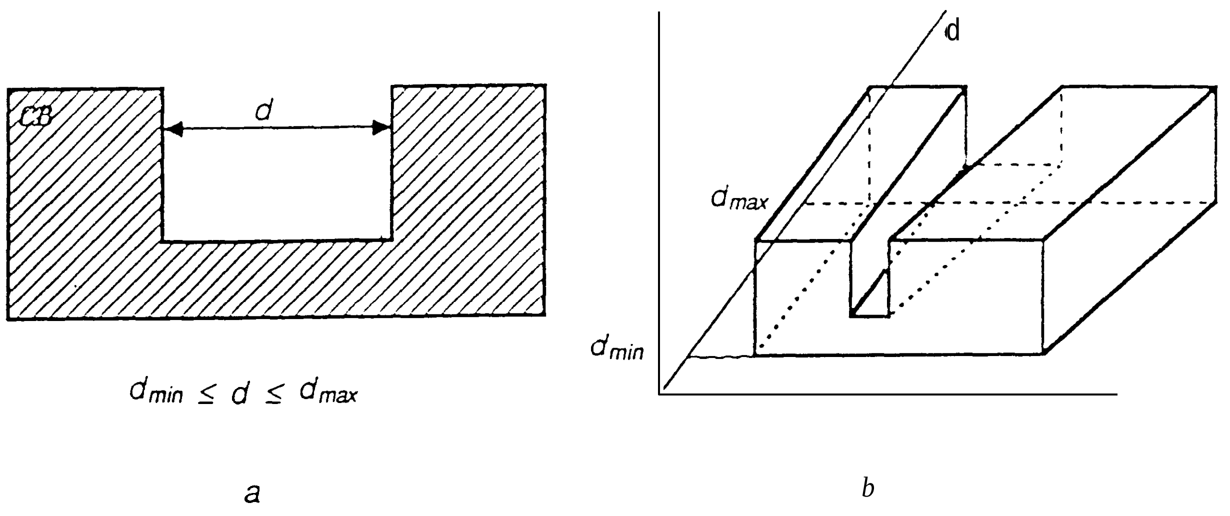


Figure **8: Representation** of Model Uncertainty

such that:
- $\mathcal{U}_c(\mathbf{c})$ = C,,(c), the closed disk of radius ρ_c centered at c,
- $\|\mathbf{f}^*\| \in [\|\mathbf{f}\| - \epsilon_f, \|\mathbf{f}\| + \epsilon_f]$,
- If $\|\mathbf{f}\| \geq$ E;, then $angle(\mathbf{f}^*, \mathbf{f}) \leq \theta_f$; otherwise, the orientation of **f** has no significance.

Throughout the paper, we assume that time measurement is perfect. This is not quite exact, because a real controller discretizes time. We also assume that the termination condition of an executed motion command is continuously monitored, and that the motion is instantaneously stopped (both in the control sense and the physical sense) when the condition becomes true. Again, this is not exact. In fact, in first approximation, errors on time measurement and on motion termination can be blended with other errors, by enlarging control and sensing uncertainties. However, a more realistic approach would be to treat them differently. An approach to the representation and the treatment of uncertainty on time measurement, in the context of motion planning, can be found in [41].

6 Preimage Backchaining

Let $\mathcal{T}$ be a region in configuration space C. Let M = **(CS,TC)** be a generalized motion command. Let $\mathcal{T}$ be the *target* of M, that is we want to bring the effector point **P** into $\mathcal{T}$ by executing M. Uncertainty on control and sensing is specified by $\mathcal{U}_{CS}$, $\mathcal{U}_c$, and $\mathcal{U}_f$. Wc assume no model error.

We call **preimage** of $\mathcal{T}$ for M any region $\mathcal{P}$ in C such that: if the effector point **P** is actually in $\mathcal{P}$ at the time when the execution of **M** starts, then, despite uncertainty, it is guaranteed both that the resulting motion will terminate and that **P** will be in $\mathcal{T}$ when

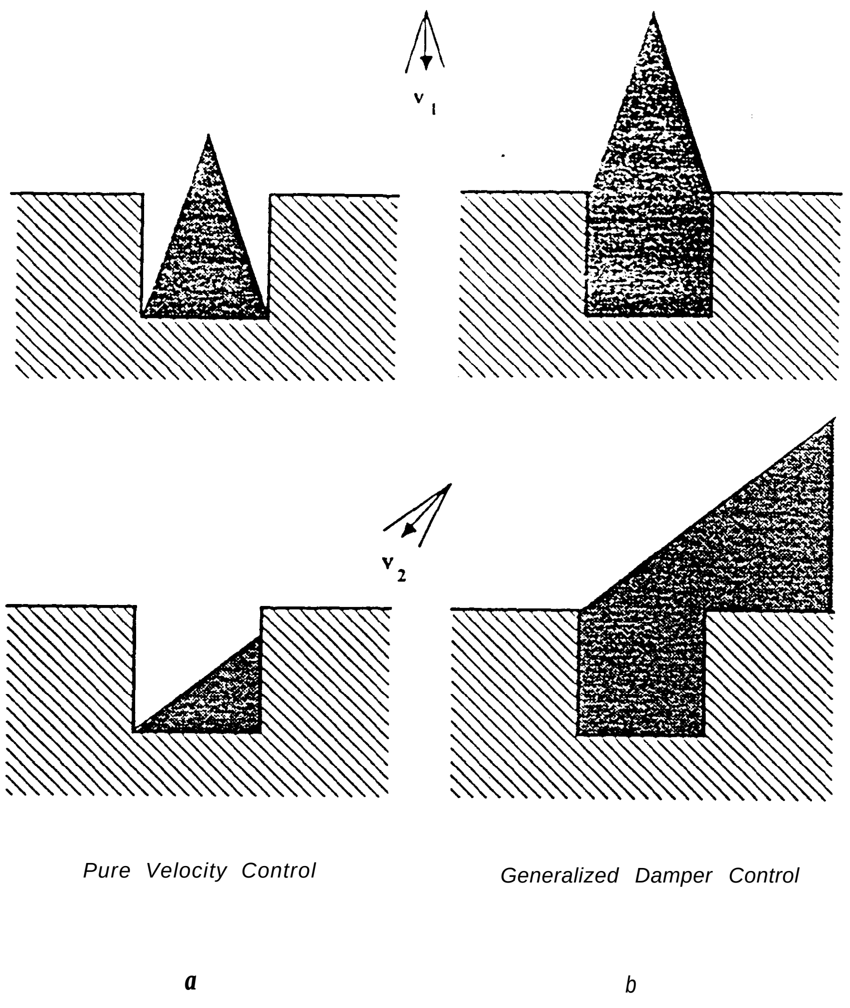


Figure **9: Examples of Preimages**

the motion terminates. In other words, if the precondition $\mathbf{P} \in \mathcal{P}$ holds before executing **M,** then the postcondition $\mathbf{P} \in \mathcal{T}$ will hold after executing the motion command. We will give a more formal definition of a preimage later in Section 8.

Example 5: Figure 9 (a_1 and a_2) shows examples of preimages of a target 7 for velocity controlled motion command with two commanded velocities $\mathbf{v}_1$ and $\mathbf{v}_2$. Figure 9 (b_1 and b_2) shows examples of preimages of $\mathcal{T}$ for generalized damper motion commands with the same **two** commanded velocities. In every example, **a** possible termination condition is: $[c(\delta t) \in$ 7 $\oplus \Sigma_{\rho_c}(0)$ and $angle(\mathbf{f}(\delta t),$ 47)) $\leq \phi(\mathcal{T}) + \theta_f]$, where $\oplus$ is the Minkowski's operator for set addition[7]. Preimages for generalized damper control are larger than those obtained with velocity control, because generalized damper control has some limited capabilities to

[7]$\mathcal{T} \oplus \Sigma_{\rho_c}(0)$ is the edge $\mathcal{T}$ grown by ρ_c.

comply with the obstacle geometry, by sliding along edges. ∎

Now, suppose that an algorithm is avaiiabie for computing prcimages of a target $\mathcal{T}$ for a motion command $\mathbf{M}$ (we will investigate preimage computation in Sections 13 through 16). Let us consider **a** motion planning problem specified by $\mathcal{G}$, the goal region in which **P** has to be moved, and $\mathcal{I}$, the region that is guaranteed to contain the initial position $\mathbf{c}^*_{init}$ of P. **Preimage backchaining** consists of constructing a sequence of preimages $\mathcal{P}_1, \mathcal{P}_2, \ldots . \mathcal{P}_q$, such that:

- $\mathcal{P}_i$, $\forall i \in [1, q]$, is a preimage of $\mathcal{P}_{i-1}$ for **a** selected motion command $\mathbf{M}_i$ (with $\mathcal{P}_0 = \mathcal{G}$);
- $\mathcal{I} \subseteq \mathcal{P}_q$.

The inverse sequence of the motion commands which have been selected to produce the prcimages, $[\mathbf{M}_q, \mathbf{M}_{q-1}, \ldots . \mathbf{M}_1]$, is the generated motion strategy. We say that this strategy is **strongly guaranteed** because its execution is guaranteed to achieve the goal successfully, whenever the control errors and sensing errors remain within the ranges determined by $\mathcal{U}_{CS}$, $\mathcal{U}_c$, and $\mathcal{U}_f$. As mentioned in the introduction, this paper focuses on this type of strategy.

A motion planner is said to be **strongly complete** if it is guaranteed to generate a strongly guaranteed strategy whenever such a strategy exists. At the eventual expense of strong completeness, the problem of generating the sequence of preimages can be transformed into' the combinatorial problem of searching a graph by selecting motion commands from a predefined discretized set. The root of this graph is the goal region $\mathcal{G}$, and each other node is a preimage region; each arc is **a** motion command, connecting a region to a preimage for this command. Construction of this graph requires discretizing the set of possible control statements. For instance, with velocity control and generalized damper control, it requires discretizing the set of velocity orientations.

Example 6: Figure 10 illustrates the application of the preimage backchaining approach to a simple example. Figure 10 *a* displays the initial region $\mathcal{I}$ and the goal region $\mathcal{G}$. Figure 10 *b* shows a preimage $\mathcal{P}_1$ of $\mathcal{G}$ for the motion command $\mathbf{M}_1 = (\mathbf{GD}(\mathbf{v}_1), \mathbf{TC}_1)$, where $\mathbf{v}_1$ is as shown in the figure and $\mathbf{TC}_1$ detects contact against $\mathcal{G}$ by measuring the horizontal component of the measured reaction force. $\mathcal{P}_1$ has no intersection with the initial region $\mathcal{I}$. Assume that we consider edge $\mathcal{E}$, which is a subset of $\mathcal{P}_1$, as an intermediate target. Figure 10 c shows a preimage $\mathcal{P}_2$ of $\mathcal{E}$ for $\mathbf{M}_2 = (\mathbf{GD}(\mathbf{v}_2), \mathbf{TC}_2)$. $\mathbf{v}_2$ **is** shown in the figure. $\mathbf{TC}_2$ detects contact against $\mathcal{E}$ by measuring both the vertical component of the reaction force. Since $\mathcal{E}$ is part of $\mathcal{P}_1$, $\mathcal{P}_2$ is also a preimage of $\mathcal{P}_1$. $\mathcal{P}_2$ includes $\mathcal{I}$; so the problem is solved. The generated strategy is $[\mathbf{M}_2, \mathbf{M}_1]$. ∎

Some motion planning problems only admit conditional strategies (i.e. strategies with conditional branching statements), or are more easily solved by generating such strategies. The application of the **preimage** backchaining approach to the generation of conditional strategies does not raise major difficulties, and will be considered in Section 17.

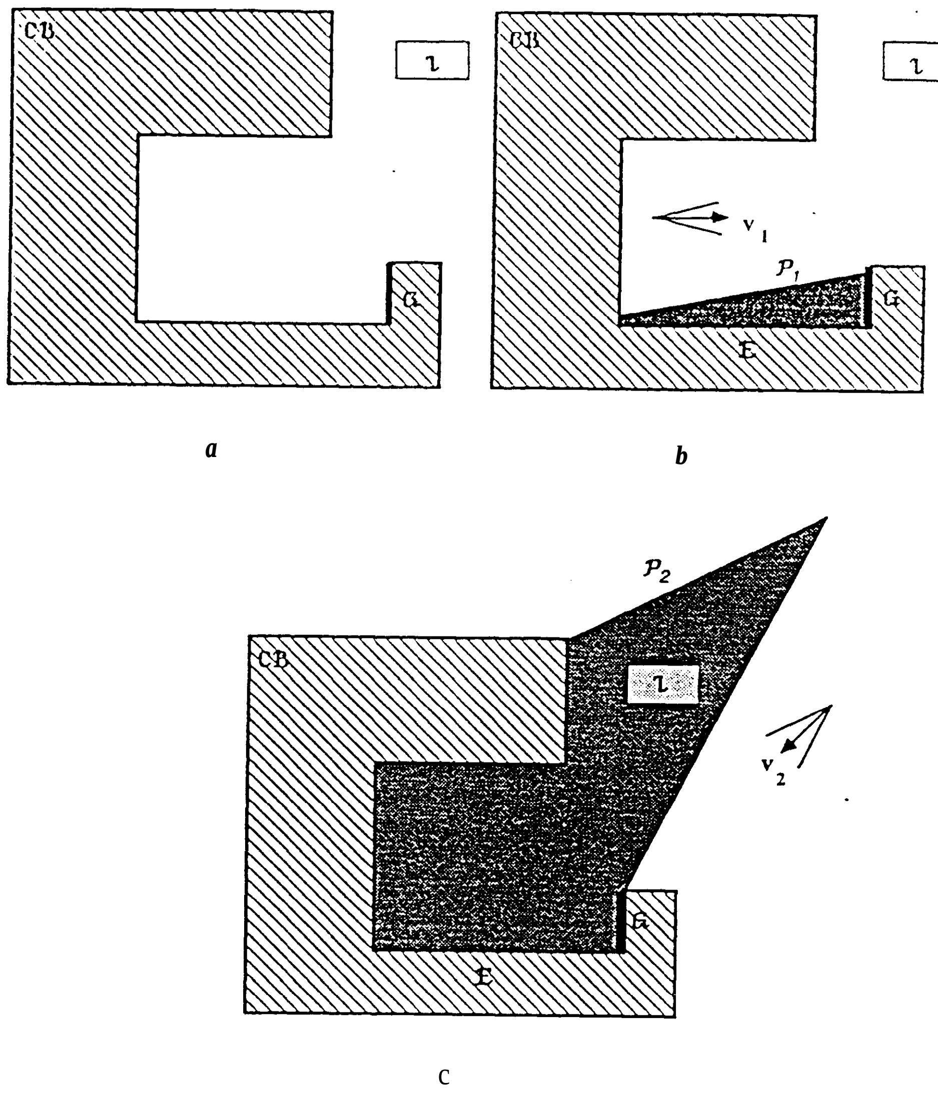


Figure 10: **Illustration of the Preimage Backchaining Approach**

A strategy with no iteration or recursion can only result in the execution of a bounded number of motion commands. It may be insufficient in the presence of infinitely many C-obstacle algebraic surfaces. However, since this is unlikely to happen in real world problems, we will not consider strategies with iteration or recursion in the rest of the paper. The completeness of **a** planner restricted to the class of motion problems that can be solved by executing a bounded number of motion commands is called **bounded completeness.**

Notice that the preimage backchaining approach can also be useful to plan motions even when there is no uncertainty. Although there exists more efficient path planning techniques applicable to such situations, the approach may still present some interest. Indeed, since the outcome of the planner is a *channel* formed by successive preimages, which is less constrained than a unique path, it **leaves** the controller more opportunities for facing contingencies (e.g., unexpected obstacles) [8]. However, we will not explore this aspect of the approach further in this paper.

Note also that the relevance of the preimage backchaining approach is not limited to motion planning. The principle of the approach may also be of interest for other types of action planning problems, including the "robot planning" problems traditionally considered in Artificial Intelligence [42]. Indeed, a preimage is nothing else than a precondition of a given postcondition (ideally, it is the weakest precondition). In motion planning, it has a strong geometric flavor; but, in other domains, it may well have a more logic-oriented flavor. In particular, *goal regression,* as presented in [54] and in [42] (Chapter S), is a similar technique for planning in worlds represented as sentences in the first-order predicate language. It resembles preimage backchaining in that it consists of propagating goals backward by computing the weakest logical conditions whose satisfaction before executing a sequence of actions guarantees the achievement of each of the goals after the actions have been executed.

In the following sections, we explore in detail the theory of the preimage backchaining approach applied to motion planning. One of our underlying preoccupations is to attempt to reduce the cost of searching the preimage graph, by analyzing the notion of *maximal preimage* (with respect to set inclusion). Indeed, intuitively, a large preimage has more chance to include the initial region $\mathcal{I}$ than **a** small one; in addition, if it is considered recursively as an intermediate target, a large preimage has more chance to admit large preimages than a small one. Thus, considering larger preimages may reduce the size of the search graph; in addition, it may have the side-effect of producing simpler strategies (i.e., strategies with less motion commands). Another way of dealing with combinatorial complexity would be to use heuristics for guiding the search; We will not explore it in this paper because, except for simple cases, it seems that motion planning with uncertainty tends to defy intuition and straightforward heuristics.

7 Actual and Observed Trajectories

Given a starting position c_i^*, the control statement CS in a motion command M = (CS,TC) specifies a nominal trajectory of P. However, due to control errors, executing the motion command may produce another trajectory, τ^*, called the **actual trajectory.** Furthermore, due to sensing errors, the termination condition TC may observe τ^* as another trajectory, τ, called the **observed trajectory.**

We represent an actual trajectory, τ^*, as $(c_{\tau^*}^*, f_{\tau^*}^*)$, where $c_{\tau^*}^*$ and $f_{\tau^*}^*$ are functions mapping the elapsed time δt since the beginning of the motion into the actual position of P and the actual reaction force on P at this instant. Notice that this representation is redundant. For example, in the mini-world, if CS = GD(v), $f_{\tau^*}^*(\delta t)$ is completely determined by the actual velocity $\dot{c}_{\tau^*}^*$ (the first derivative of $c_{\tau^*}^*$), the friction cone on each C-obstacle edge, and the damper constant. In particular, $\forall \delta t \geq 0$:
- $c_{\tau^*}^*(\delta t) \in \mathcal{C}_{free} \Rightarrow f_{\tau^*}^*(\delta t) = 0$,
- $c_{\tau^*}^*(\delta t) \in \mathcal{C}_{contact} \Rightarrow \|f_{\tau^*}^*(\delta t)\| = 0$ or $angle(\nu(c_{\tau^*}^*(\delta t)), f_{\tau^*}^*(\delta t)) \leq \phi(c_{\tau^*}^*(\delta t))$.

We represent an observed trajectory, τ, as (c_τ, f_τ), where c_τ and f_τ are functions mapping the elapsed time since the beginning of the motion into the measured position and the measured force at this instant.

When planning *a* motion command to achieve some target, the motion planner has to consider the set of possible actual trajectoriesthat can result from the execution of the command, since they must all attain the target. It also has to consider the set of possible observed trajectories, so that it can plan termination conditions that will guarantee the controller to terminate the motion when the goal is attained.

This leads us to introduce several notions, which are useful to formalize and explore the preimage backchaining approach.

One notion is the directory of actual trajectories, which contains a description of all the possible actual trajectories that can be generated by executing a motion according to a commanded control statement CS from a region S [39]:

-DEFINITION 1: *The* **directory of actual trajectories** *for a region S in C and a control statement* CS *is the set, denoted $\mathcal{D}^*(S, CS)$, of all the trajectories τ^* of* P *that would be generated by an ideal controller executing every motion command M^* = $(CS^*, false)$, with $CS^* \in \mathcal{U}_{CS}(CS)$, according to the exact* specification *of CS^*, starting from every position c_s^* in* S (i.e., $c^*(0) = c_s^* \in S$). *(The termination condition 'false'* is *the constant termination condition which is never satisfied.)*

The second notion is consistency between actual and measured data:

DEFINITION 2: *A pair (c^*, f^*) of actual position and force, and a pair (c, f) of measured*

position and force are **consistent** *if and only if* $(c^*, f^*) \in \mathcal{U}_c(c) \times \mathcal{U}_f(f)$.
An actual trajectory τ^* *and an observed trajectory* τ *are* **consistent** *if and only if,* $\forall \delta t \geq 0$: $(c^*_{\tau^*}(\delta t), f^*_{\tau^*}(\delta t)) \in \mathcal{U}_c(c_\tau(\delta t)) \times \mathcal{U}_f(f_\tau(\delta t))$. $\mathcal{K}_{traj}(\tau^*) \overset{\text{def}}{=} \{\tau \ / \ \tau^*$ *and* τ *are consistent*$\}$.

Since the range of possible reaction forces at a position c^* is $\mathcal{F}^*(c^*)$, we can also define the consistency between an actual position and a pair of measured position and force:

DEFINITION 3: *An actual position* c^* *and a pair* (c, f) *of measured position and force are* **consistent** *if and only if* $\exists f^* \in \mathcal{F}^*(c^*)$: (c^*, f^*) *and* (c, f) *are consistent.*
$\mathcal{K}^*_{pos}(c, f) \overset{\text{def}}{=} \{c^* \ / \ c^*$ *and* (c, f) *are consistent*$\}$.

In the mini-world, $\mathcal{K}^*_{pos}(c, f)$ can be computed as follows:

- if $||f|| \leq \varepsilon_f$ then: $\mathcal{K}^*_{pos}(c, f) = \Sigma_{\rho_c}(c) \bigcap (\mathcal{C}_{free} \cup \mathcal{C}_{contact})$;
- if $||f|| > \varepsilon_f$ then: $\mathcal{K}^*_{pos}(c, f) = \Sigma_{\rho_c}(c) \bigcap \{c'^* \in \mathcal{C}_{contact} \ / \ angle(\nu(c'^*), f) \leq \phi(c'^*) + \theta_f\}$.

The third notion is confusability between actual data:

DEFINITION 4: *Two pairs* (c_1^*, f_1^*) *and* (c_2^*, f_2^*) *are* **confusable** *if and only if* $\exists (c, f)$ *such that* $(c_1^*, f_1^*) \in \mathcal{U}_c(c) \times \mathcal{U}_f(f)$ *and* $(c_2^*, f_2^*) \in \mathcal{U}_c(c) \times \mathcal{U}_f(f)$. *Otherwise they are* **distinguishable.**
Two actual trajectories τ_1^* *and* τ_2^* *are* **confusable** *if and only if,* $\forall \delta t \geq 0$: $(c^*_{\tau_1^*}(\delta t), f^*_{\tau_1^*}(\delta t))$ *and* $(c^*_{\tau_2^*}(\delta t), f^*_{\tau_2^*}(\delta t))$ *are confusable. Otherwise they are* **distinguishable.**

In the mini-world, two actual trajectories τ_1^* and τ_2^* are confusable if and only if the following two conditions hold simultaneously, $\forall \delta t \geq 0$:
- $distance(c^*_{\tau_1^*}(\delta t), c^*_{\tau_2^*}(\delta t)) \leq 2\rho_c$,
- if $||f^*_{\tau_1^*}(\delta t)|| > 2\varepsilon_f$ and $||f^*_{\tau_2^*}(\delta t)|| > 2\varepsilon_f$, then $angle(f^*_{\tau_1^*}(\delta t), f^*_{\tau_2^*}(\delta t)) \leq 2\theta_f$.

If two trajectories are confusable, the motion planner cannot be certain that the controller will be able to distinguish between them at execution time.

8 Formal Definition of Preimages

Given a target[8] $\mathcal{T}$ in C and a motion command **M** = **(CS,TC),** a preimage $\mathcal{P}$ is such that any possible motion of **P** executed according to CS, starting from within P, follows a trajectory τ that is guaranteed to attain $\mathcal{T}$ *(target attainment)* in such a **way** that **TC** stops

[8]We use two different words, *target* and goal, which the reader may consider rather indistinctively. However, our convention is to use the world *target* when we are only interested in a single step of preimage backchaining. We use the word *goal* when we are interested in a complete motion planning problem, which may, or may not, require multiple-step recursion.

P into the target (*target recognition*). We formalize these two concepts – target attainment and target recognition – below, by defining two predicates, `Attain` and `Achieve`.

Let us denote **Attain**($\mathcal{T}$,CS,$\mathcal{S}$) the condition that a motion executed according to CS is guaranteed to attain 7 if the initial position of P is in $\mathcal{S}$. This condition can be formalized as follows:

DEFINITION 5: Attain($\mathcal{T}$, **CS,** $\mathcal{S}$) $\stackrel{\text{def}}{=} [\forall \tau^* \in \mathcal{D}^*(\mathcal{S}, CS), \exists t \geq 0 : c^*_{\tau^*}(t) \in \mathcal{T}]$.

Obviously: **Attain**($\mathcal{T}$,CS,$\mathcal{S}$) $\Leftrightarrow [\forall c^*_i \in \mathcal{S}$: Attain(7, CS, $\{c^*_i\})]$.

A preimage $\mathcal{P}$ of 7 for M = **(CS,TC)** must satisfy **Attain**($\mathcal{T}$, CS, P), since any motion from within $\mathcal{P}$ must attain 7. However, it is only a necessary condition. A region $\mathcal{S}$ satisfying Attain(7, CS, $\mathcal{S}$) may not be a prcimage of 7 for M because executing M from within $\mathcal{S}$ may not be guaranteed to terminate in 7 (it may even not be guaranteed to terminate at all!). Appropriate termination of M is under the responsability of the termination condition **TC,** which plays no role in the definition of `Attain`.

The termination condition TC = **tp**(δt, $c_{[0,\delta t]}$, $f_{[0,\delta t]}$) only applies to observed trajectories. $\mathcal{D}^*(\mathcal{S}$, CS) contains possible actual trajectories. Each such trajectory, τ^*, may be observed by **TC** as any trajectory τ in $\mathcal{K}_{traj}(\tau^*)$.

Let us denote Achieve(7, **M,** $\mathcal{P}$) the condition that the execution of M is guaranteed to terminate in $\mathcal{T}$ if the initial position of P is inside $\mathcal{P}$. It is formalized as follows:

DEFINITION 6:
Let M = (CS,TC) and TC = **tp**(δt, $c_{[0,\delta t]}$, $f_{[0,\delta t]}$).
Achieve($\mathcal{T}$, M, $\mathcal{P}$) $\stackrel{\text{def}}{=} [\forall \tau^* \in \mathcal{D}^*(\mathcal{P}, CS), \forall \tau \in \mathcal{K}_{traj}(\tau^*)$:
. $\exists t \geq \mathbf{0}$: **tp**(t, $c_{[0,t]}$, $f_{[0,t]}$) = **true;**
· *let* $t_0 = Inf\ \{t/\mathbf{tp}(t, c_{[0,t_0]}, f_{[0,t_0]}) = true\}$; $c^*_{\tau^*}(t_0) \in \mathcal{T}]$.

The formal definition of a preimage derives directly from the specification of `Achieve`:

***DEFINITION 7:** *A* **preimage** *of* **7** *for the motion command* M *is any region* $\mathcal{P}$ *such that* **Achieve**($\mathcal{T}$, M, $\mathcal{P}$).

This formal definition does not provide an immediate practical method for constructing preimages. However, we can easily derive the following properties:

PROPERTY 1:
a- **Achieve**($\mathcal{T}$, M, $\mathcal{P}$) $\Leftrightarrow \forall c^*_i \in \mathcal{P}$: **Achieve**($\mathcal{T}$, M, $\{c^*_i\}$).

b- *If* $\mathcal{P}$ *is a preimage of* **7** *for* M, *then any subset of* $\mathcal{P}$ *is also a preimage* ***of*** $\mathcal{T}$ ***for*** **M.**

c- *If $\mathcal{P}_1$ and $\mathcal{P}_2$ are both preimages of $\mathcal{T}$ for* M, *then $\mathcal{P}_1 \cup \mathcal{P}_2$ is a preimage of $\mathcal{T}$ for* M.

These properties naturally lead to the notion of maximal preimage:

DEFINITION 8: *The region $\mathcal{P}^{max}(\mathcal{T}, M) \overset{def}{=} \{c_i^* / Achieve(\mathcal{T}, M, \{c_i^*\})\}$ is the* **maximal preimage of $\mathcal{T}$ *for* M.**

As mentioned in the preceding section, the size of preimages is an important factor to consider both for reducing the cost of searching the preimage graph, and for producing simpler strategies. Since every preimage of a target $\mathcal{T}$ is included in a maximal preimage $\mathcal{P}^{max}(\mathcal{T}, M)$, for some M, we are conducted to investigate the parameters in M influencing the size of $\mathcal{P}^{max}(\mathcal{T}, M)$.

The size of $\mathcal{P}^{max}(\mathcal{T}, M)$ depends on both the ability of the control statement CS to attain $\mathcal{T}$ and the ability of the termination predicate TC to recognize achievement of $\mathcal{T}$. Dependence on CS is an important topic relating motion control to motion planning. Because there is currently no substantial results (either theoretical or practical), we will not discuss it further in this paper. In the next four sections, we address the dependence of the maximal preimage on the termination condition.

9 Power of a Termination Condition

The following definition specifies a partial ordering on termination conditions for a given target $\mathcal{T}$ and a given control statement CS:

DEFINITION 9: *Let $M_1 = (CS, TC_1)$ and $M_2 = (CS, TC_2)$ be two motion commands that only differ by their termination conditions. TC_1 is said to be* more powerful *than TC_2* ***for*** *CS and $\mathcal{T}$ if and only if $\mathcal{P}^{max}(\mathcal{T}, M_2) \subseteq \mathcal{P}^{max}(\mathcal{T}, M_1)$.*

Therefore, if TC_1 is more powerful than TC_2 for CS and $\mathcal{T}$, then $\forall \mathcal{P} \subseteq \mathcal{C}$: $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M_2 implies that it is also a preimage of $\mathcal{T}$ for M_1.

Example 7: Consider the point-into-hole example shown at Figure 11 a. The two horizontal edges on the sides of the hole are semi-infinite lines[9]. The target $\mathcal{T}$ is the edge at the bottom of the hole and CS = $GD(\mathbf{v})$. Assume perfect control ($v' = v$), but no position feedback ($\rho_c = \infty$) and no force feedback ($\varepsilon_f = \infty$). The termination condition can only recognize achievement of the target by measuring the elapsed time since the beginning of the motion. Therefore, any finite region $\mathcal{P}$ inside the shaded area displayed in Figure 11 a is a preimage for the termination condition $\delta t \geq T$, where T is the maximal amount

[9]Since some of the edges are not finite, this example occurs in a space that slightly differs from the mini-world. However, all the other mini-world specifications apply.

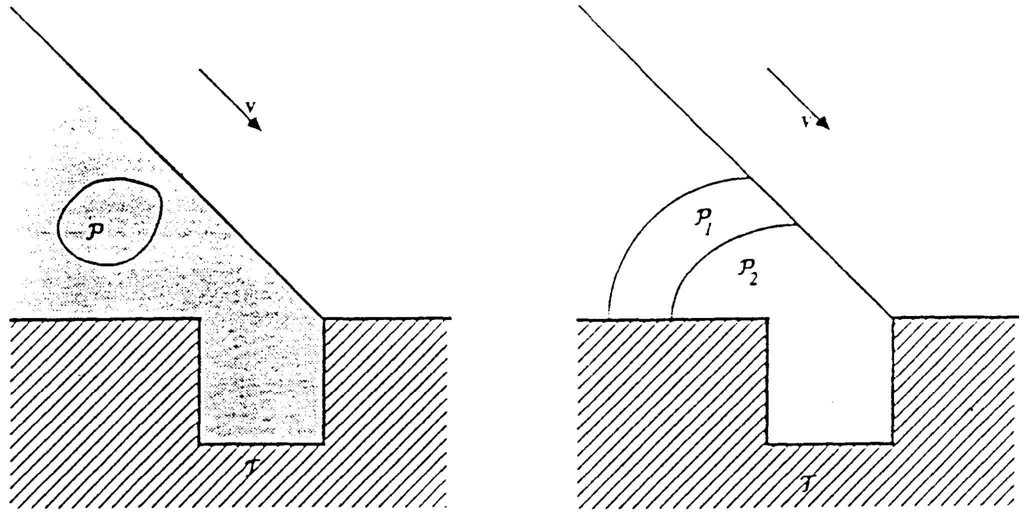


Figure 11: **Illustration of the Power of a Termination** Condition

of time required for attaining $\mathcal{T}$ from within $\mathcal{P}$. Consider the maximal preimage $\mathcal{P}_1$ with $\mathbf{TC}_1 = [\delta t \geq T_1]$, and the maximal preimage $\mathcal{P}_2$ with $\mathbf{TC}_2 = [\delta t \geq T_2]$, where $T_2 \leq T_1$ (Figure **11** *b*). Clearly, $\mathbf{TC}_1$ is more powerful than $\mathbf{TC}_2$ for CS and $\mathcal{T}$. ∎

The power of **a** termination condition depends on both its arguments – i.e. the information it has access to during motion – and the knowledge embedded in its predicate – i.e. the information that is transmitted by the planner. We analyze these dependences in the following two sections.

10 Role of Arguments in a Termination Condition

The general form of a termination condition includes the following arguments: δt, the elapsed time since the beginning of the motion; $\mathbf{c}_{[0,\delta t]}$, the record of position sensing since the beginning of the motion; and $\mathbf{f}_{[0,\delta t]}$, the record of force sensing since the beginning of the motion. **However,** a particular termination condition may use only **a** subset of these arguments. The following definition characterizes several types of termination conditions depending on the arguments they actually use [17][10]:

DEFINITION 10:

[10] Actually, our terminology slightly differs from the definition given by Erdmann.

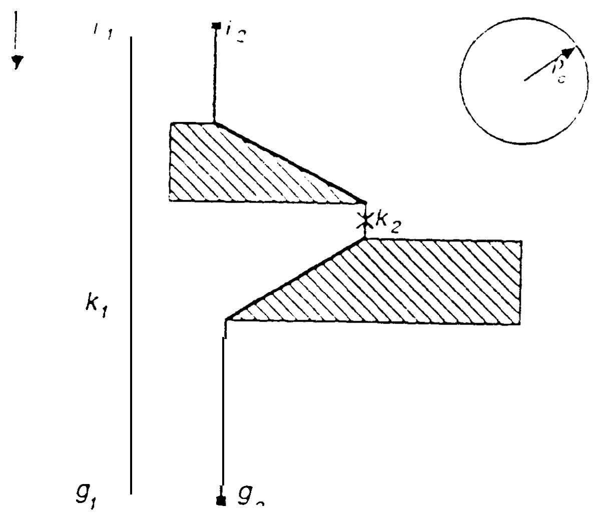


Figure 12: **Illustration of the Role of History in a Termination Condition**

A termination condiiion using c(δt) *and/or* f(δt) is *called a ierminaiion condition* **with instantaneous sensing.**
A terminatiòn condiiion *using St is called a termination condition* **with time.**
A termination condition using c(t) and/or f(t), for some $t \in [0, \delta t[$, *is called a termination condition* **with history.**

In Example 5, we used a termination condition with instantaneous sensing, but without time and history to illustrate the notion of preimage. Indeed, instantaneous position and force measurements were sufficient to reliably recognize entry into the target. However, as we show below, there are situations where time and history are useful (or needed!).

For instance, consider Example 7 again. Since no position and no force sensing are available, the motion command can only rely on the elapsed time δt to recognize achievement of the target. Note that in this case the only termination conditions without time are the constant conditions *true* and *false*. Only *true* can stop the execution of the motion command, and the largest preimage of $\mathcal{T}$ for (CS,*true*) is $\mathcal{T}$ itself!

The example below illustrates the role of history in a termination condition:

Example 8: Consider the motion planning problem depicted at Figure 12 (this example is extracted from [39]): the region $\mathcal{I}$ of possible initial positions of **P** consists of two points i_1 and i_2; the region $\mathcal{G}$ of goal positions of **P** consists of two points g_1 and g_2. Assume that CS = **GD(v)**, with v pointing downward, perfect control, imperfect position sensing (ρ_c is as shown in the upper right corner of the figure), and no force measurement. Thus, there are two possible actual trajectories depending on the initial position of **P.** Uncertainty on position sensing makes the two trajectories distinguishable only during ... interval where they are distant from each other by more than $2\rho_c$. Thus, by remembering sensing history, the termination condition can terminate the motion inside the target. A specification for such a termination condition is the following:

```
if $\delta t \leq T_k$
    then if $xc(\delta t)$ - $xc_1^* \leq 2\rho_c$
        then $flag \leftarrow 1$;
        else $flag \leftarrow 2$;
    else if $((flag = 1) \wedge (\delta t = T_1)) \vee ((flag = 2) \wedge (\delta t = T_2))$
        then return(true);
        else return(false);
```

where:

- $xc(\delta t)$ is the x-coordinate of the measured position of P along the horizontal x-axis;
- xc_1^* is the x-coordinate of the actual position of i_1 along the x-axis;
- T_k is the amount of time to attain point k_1 or k_2 (see Figure 12), depending on whether the motion starts from i_1 or from i_2;
- T_i ($i = 1$ or 2) is the amount of time necessary to travel from i_i to g_i. ∎

Notice that the above example could not be solved by a three-motion strategy (left-down-right) avoiding all the obstacles, because the relative position of i_2 and g_2 is different from the relative position of i_1 and g_1. It could also be solved, however, by generating a conditional strategy (see Section 17) with two motion commands, none of them including a termination condition with history. However, the important event (that the two possible trajectories become distinguishable) is used to build the branching statement in the strategy. Thus, history is incorporated in the control structure of the motion strategy rather than in the termination condition. In fact, this seems to be a general way of remembering sensing history, removing the absolute need for termination conditions with history.

Termination conditions with instantaneous sensing, but without time and history, can be at best as powerful as termination conditions with instantaneous sensing and either time, or history, or both. Nevertheless, they seem sufficient for solving many realistic motion-planning problems.

11 Termination Conditions With State

In this section, we explore how the planner can transmit some knowledge to the controller in the termination predicate.

Remember that given a goal region $\mathcal{G}$ and an initial region $\mathcal{I}$, preimage backchaining consists of constructing a sequence of preimages $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_q$ such that: (1) $\mathcal{P}_i$ is a preimage of $\mathcal{P}_{i-1}$ for a selected motion command $\mathbf{M}_i$; and (2) $\mathcal{I} \subseteq \mathcal{P}_q$. When $\mathcal{P}_i$ is constructed, it is known by rccurrcncc that if the recursion upwinds successfully, P will be inside $\mathcal{P}_i$ before $\mathbf{M}_i$ is executed. Thus, if the planner was able to construct $\mathcal{P}_i$ and $\mathbf{M}_i$ simultaneously, it. could embed this knowledge into the termination predicate $\mathbf{tp}_i$ of TC, (the termination condition of $\mathbf{M}_i$). This knowledge might contribute to augmenting the power of $\mathbf{TC}_i$.

This is the idea analyzed in this section. Although the outcome is not a practical means for constructing the resulting termination predicate, it is useful to establish limits on termination conditions and preimages, before we explore techniques for constructing them.

Let us introduce the notion of termination condition with state:

DEFINITION 11: *Let $\mathcal{S}$ be a region in C, $\mathcal{T}$ a target, and CS a control statement.* $\mathbf{TC}^+_{\mathcal{S}} = \mathbf{tp}^+_{\mathcal{S}}(\delta t, \mathbf{c}_{[0,\delta t]}, \mathbf{f}_{[0,\delta t]})$ is *specified as* follows:

1. $\mathcal{L} \leftarrow \mathcal{D}^*(\mathcal{S}, \mathrm{CS})$.

2. *For every $\delta t \geq 0$ do:*

   - *For every τ^** ***in*** *$\mathcal{L}$,* ***if*** $(\mathbf{c}^*_{\tau^*}(\delta t), \mathbf{f}^*_{\tau^*}(\delta t)) \notin \mathcal{U}_c(\mathbf{c}(\delta t)) \times \mathcal{U}_f(\mathbf{f}(\delta t))$, *then remove τ^** ***from*** *$\mathcal{L}$.*
   - $\dot{\mathcal{Q}} \leftarrow \{\mathbf{c}^*_{\tau^*}(\delta t) \ / \ \tau^* \in$ [illegible] ***If*** *$\mathcal{Q} \subseteq \mathcal{T}$, then evaluate $\mathbf{TC}^+_{\mathcal{S}}$ to true; otherwise evaluate* ***to false.***

$\mathbf{TC}^+_{\mathcal{S}}$ is *called a* **termination condition with state,** *and* $\mathbf{tp}^+_{\mathcal{S}}$ a **termination predicate with state.** *We denote* $\mathbf{M}^+_{\mathcal{S}} = (\mathrm{CS}, \mathbf{TC}^+_{\mathcal{S}})$.

$\mathbf{TC}^+_{\mathcal{S}}$ embeds in its predicate the knowledge that the only possible actual trajectories are those which may **be** produced by CS starting from within S. Indeed, the evaluation of the termination condition does not consider trajectories that are not in $\mathcal{D}^*(\mathcal{S}$, CS), while some of these trajectories might be confusable with trajectories in $\mathcal{D}^*(\mathcal{S}$, CS). Thus, there may be cases where a termination condition not embedding the above knowledge is not able to recognize achievement of 7, while $\mathbf{TC}^+_{\mathcal{S}}$ can.

Now suppose that the planner considers a region $\mathcal{P}$ as a candidate preimagc of $\mathcal{T}$ for a motion command whose control statement is CS. Using the termination condition with state, it **may** attempt to construct the preimage of 7 relative to $\mathcal{P}$:

DEFINITION **12:** *The region* $\Pi_{\mathcal{P}}(\mathcal{T}, \mathbf{M}^+_{\mathcal{P}}) = \{\mathbf{c}^*_s \in \mathcal{P}/\mathtt{Achieve}(\mathcal{T}, \mathbf{M}^+_{\mathcal{P}}, \{\mathbf{c}^*_s\})\}$ is *called the preimage* ***of*** *$\mathcal{T}$* **relative to** *$\mathcal{P}$* ***(for the control*** *statement CS in* $\mathbf{M}^+_{\mathcal{P}}$).

Obviously, if $\mathcal{P} = \Pi_{\mathcal{P}}(\mathcal{T}, \mathbf{M}^+_{\mathcal{P}})$, then $\mathcal{P}$ is a preimage of 7 for $\mathbf{M}^+_{\mathcal{P}}$. Furthermore, we prove the following lemma:

LEMMA 1: *If $\mathcal{P} \neq \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$, then there exists no termination condition* TC *such that $\mathcal{P}$ is a preimage of $\mathcal{T}$ for* M = (CS,TC).

Proof: First, note that by definition: $\Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}}) \subseteq \mathcal{P}$. So, if $\mathcal{P} \neq \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$, it implies that $\mathcal{P} - \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}}) \neq 0$.

Now assume the existence of a termination condition TC = $\mathbf{tp}(\delta t, \mathbf{c}_{[0,\delta t]}, \mathbf{f}_{[0,\delta t]})$ such that $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M = (CS,TC), while $\mathcal{P} \neq \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$. Consider a sample motion, commanded according to CS from an initial position inside $\mathcal{P} - \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$. Assume that **TC** terminates this motion (in $\mathcal{T}$), while $\mathbf{TC}^{+}_{\mathcal{P}}$ would not have terminated it. Such a sample motion necessarily exists, otherwise $\mathcal{P} = \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$. Let us denote τ the observed trajectory and t_0 the instant when **TC** becomes true. Thus $t_0 = Inf\ \{t/\mathbf{tp}(\delta t, \mathbf{c}_{\tau[0,\delta t]}, \mathbf{f}_{\tau[0,\delta t]}) = true\}$.

Since $\mathbf{TC}^{+}_{\mathcal{P}}$ would not have terminated the motion at t_0: $\exists \tau^{*}_{\lambda} \in \mathcal{D}^{*}(\mathcal{P}, \mathrm{CS})$ such that $\tau \in \mathcal{K}_{traj}(\tau^{*}_{\lambda})$, while $\mathbf{c}^{*}_{\tau^{*}_{\lambda}}(t_0) \notin \mathcal{T}$. This falsifies the condition Achieve($\mathcal{T}$, M, $\mathcal{P}$), and therefore contradicts the initial assumption that $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M. ∎

An immediate consequence of the above lemma is the following theorem:

THEOREM 1: *A region $\mathcal{P}$ is a preimage **of** a target $\mathcal{T}$ **for a** control statement CS **if** and only* if $\mathcal{P} = \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$, *where* $\mathrm{M}^{+}_{\mathcal{P}}$ = *(CS,* $\mathbf{TC}^{+}_{\mathcal{P}}$*). The equation* $\mathcal{P} = \Pi_{\mathcal{P}}(\mathcal{T}, \mathrm{M}^{+}_{\mathcal{P}})$ is *called the* **characteristic equation** ***of*** *preimages.*

This theorem means intuitively that there is no way to provide a termination condition with more useful knowledge than is in $\mathbf{TC}^{+}_{\mathcal{P}}$. Note however that we cannot say that $\mathbf{TC}^{+}_{\mathcal{P}}$ is the most powerful termination condition for CS and $\mathcal{T}$. Indeed, $\mathbf{TC}^{+}_{\mathcal{P}}$ does not denote just *one* termination condition, but an *infinity* of them (one for each region in C). This is due to the fact that the termination predicate $\mathbf{tp}^{+}_{\mathcal{P}}$ is the value of a function of $\mathcal{P}$. This value (i.e. the predicate itself) is fixed only when $\mathcal{P}$ is known.

The notion of termination condition with state can easily be generalized to termination conditions without instantaneous sensing, history or time. For instance, a termination condition $\mathbf{TC}_{\mathcal{S}}$ with **state** and time, but without sensing, i.e. without history and instantaneous sensing, is specified as follows:

1. $\mathcal{L} \leftarrow \mathcal{D}^{*}(\mathcal{S}$, cs).
2. For every $\delta t \geq 0$ do:
    - $\mathcal{Q} \leftarrow \{\mathbf{c}^{*}_{\tau^{*}}(\delta t)/\tau^{*} \in \mathcal{L}\}$. If $\mathcal{Q} \subseteq \mathcal{T}$, then evaluate $\mathbf{TC}_{\mathcal{S}}$ to true; otherwise evaluate to false.

In Example 7, the termination condition $\delta t \geq T$ is equivalent to $\mathbf{TC}_{\mathcal{S}}$ for a certain S easily related to δt.

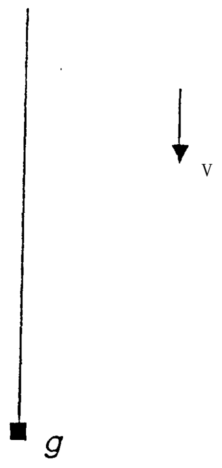


Figure 13: **The Point-onto-Point** Problem

12 Maximality of Preimages

Consider two regions $\mathcal{P}_1$ and $\mathcal{P}_2$ that satisfy the characteristic equation of preimages for a target $\mathcal{T}$ and a control statement CS. In general, their union is not a preimage of 7. Indeed, $\mathcal{P}_1$ and $\mathcal{P}_2$ may be preimages for different termination predicates, and there may exist no termination predicate capable of recognizing achievement of $\mathcal{T}$, if the initial position of **P** is only known to be within $\mathcal{P}_1 \cup \mathcal{P}_2$.

In correlation with this fact, preimages of a target $\mathcal{T}$ for **a** given control statement CS do not admit **a** unique maxima1 element (with respect to set inclusion) *over all possible termination conditions* (as we saw in Section 8, a unique maximal preimage exists when the termination condition is given in addition to the control statement). The following two examples show that: (I) there may exist no maximal prcimage, and (2) if there exists one, there may be an infinity of them.

Example 9: Let us consider again the point-into-hole problem under the same conditions as in Example 7. There exists a non-countable infinity of preimages, none being maximal. The union of all these preimages, which is an un-bounded region, is not a preimage; indeed, no termination condition can terminate reliably a motion starting from anywhere in this region, just by waiting **a** predefined finite amount of time. ∎

Example 10: Now consider the point-onto-point problem depicted at Figure 13, with perfect generalized damper control, commanded velocity pointing downward, and no position sensing. The goal region consists of a single point g. Each point on the vertical half-line above g is a maximal preimage. There is a continuous infinity of them. ∎

It is reasonable however to expect that if free space is bounded and if there are a finite number of algebraic constraints on the motion of P imposed by the surfaces of the obstacles[11] then there exists at least one maximal preimage over a11 possible termination conditions.

13 Preimage Computation

Within the preimage backchaining framework, we would like to be able to compute a *complete description* of the preimages of a target $\mathcal{T}$ for a control statement CS, i.e., a finite description of all the preimages. For example, if there exists a unique maximal preimage, then its description subsumes all other preimages, since any preimage is a subset of the maximal preimage; thus a description of the maximal preimage is a complete description of the preimages. If there exist no maximal preimages or several of them, then we may hope that the set of preimages still admit a finite description usable by the preimage backchaining algorithm for building the search graph; for instance, in Example 9, it would be "every region included in the infinite shaded area"; in Example 10, it would be "every point along a half-line drawn upward from the goal point g".

Unfortunately, neither the specification of the predicate Achieve, nor the characteristic equation of preimages, provide an algorithm for computing preimages, maximal or not. In fact, we know no generally applicable algorithm for computing a complete description of preimages, at least for a realistic type of control statement. We even do not know whether it is possible to produce such an algorithm. In order to realize the difficulty of computing preimages, one may consider the supposedly simpler problem of constructing an algorithm for verifying that an input candidate region $\mathcal{P}$ is a preimage of a target $\mathcal{T}$ for a motion command M. Even this problemstill has no known general solution. In principle, it requires to check that when any observable trajectory τ in $\{\tau \ / \ \exists \tau^* \in \mathcal{D}^*(\mathcal{P}, \text{CS}) : \tau \in \mathcal{K}_{traj}(\tau^*)\}$ terminates, then every actual trajectory τ^* in $\{\tau^* \in \mathcal{D}^*(\mathcal{P}, \text{CS}) \ / \ \tau \in \mathcal{K}_{traj}(\tau^*)\}$ has attained 7; but, there may be a non-countable infinity of possible actual and observable trajectories. In general, it is not kndwn how to characterize them finitely.

Despite the above remarks, there exist algorithms for constructing preimages. Some obvious ones work under very restrictive assumptions on control and/or sensing, for example that control is perfect. We will describe no such algorithms, but it is easy to imagine simple ones by looking to some of the examples given above. Their applicability is very limited. Instead, in the next three sections, we present an algorithm, which imposes no such limitative assumptions, but which does not usually produce complete description of preimages. Using this algorithm may result in a non-optimal overall backchaining preimage program. It may also augment the incompleteness of this program[12]. However, its

[11]Erdmann [17] gives an example showing that in the presence of an-infinite number of C-obstacle algebraic surfaces in a bounded free space there may exist no maximal preimage.

[12]Another source of incompleteness is the discretization of control statements (see Section 6).

applicability is quite general.

14 Backprojection from Target Kernel

This section and the next two present a technique for constructing preimages, which may not be maximal. Below, we introduce and formalize the basic ideas underlying this technique. The two subsequent sections describe algorithms performing the required geometric computations in the mini-world.

The technique presented consists of: (1) identifying a subset of the target (the *kernel*) such that if it is attained then achievement of the target is recognizable by a computable termination condition without state, history, and time; and (2) determining a region (the *backprojection)* from which a given motion command is guaranteed to attain that subset.

We already used $\mathcal{F}^*(\mathbf{c}^*)$ to denote the range of reaction forces that can be generated at position $\mathbf{c}^*$. Let us now denote $\mathcal{F}^*_{\mathrm{CS}}(\mathbf{c}^*)$ the range of reaction forces that can be generated at position c* when the specified control statement is CS. $\forall \mathrm{CS} : \mathcal{F}^*_{\mathrm{CS}}(\mathbf{c}^*) \subseteq \mathcal{F}^*(\mathbf{c}^*)$. In particular, let CS = $\mathbf{GD}(\mathbf{v})$; if $\mathbf{c}^* \in \mathcal{C}_{free}$, then $\mathcal{F}^*_{\mathbf{GD(v)}}(\mathbf{c}^*) = \{0\}$; if $\mathbf{c}^* \in \mathcal{C}_{contact}$, then $\mathcal{F}^*_{\mathbf{GD(v)}}(\mathbf{c}^*) = \{\mathbf{f}^* / \exists \mathbf{v}^* \in \mathcal{U}_v(\mathbf{v}) : \mathbf{f}^* = \mathbf{f}^*_{react}(\mathbf{c}^*, \mathcal{B}\mathbf{v}^*)\}$, where $\mathbf{f}^*_{react}(\mathbf{c}^*, \mathcal{B}\mathbf{v}^*)$ is the reaction force to $\mathcal{B}\mathbf{v}^*$ at $\mathbf{c}^*$. $\mathbf{f}^*_{react}(\mathbf{c}^*, \mathcal{B}\mathbf{v}^*)$ depends on the friction cane at $\mathbf{c}^*$ as follows (see Figure **4): if** $\mathcal{B}\mathbf{v}^*$ points inside the friction cone, then $\mathbf{f}^*_{react} = -\mathcal{B}\mathbf{v}^*$; otherwise $\mathbf{f}^*_{react}$ is equal to the projection of $-\mathcal{B}\mathbf{v}^*$, perpendicular to the cone's axis, onto the closest extreme ray of the cone.

We can define the confusability of two actual positions for a given control statement as follows:

DEFINITION 13: *Let* $\mathbf{c}^*_1$ *and* $\mathbf{c}^*_2$ *be two actual positions in* $\mathcal{C}_{free} \cup \mathcal{C}_{contact}$, *and CS a control statement.* $\mathbf{c}^*_1$ *and* $\mathbf{c}^*_2$ are **CS-confusable** *if and only* if $\exists \mathbf{f}^*_1 \in \mathcal{F}^*_{\mathrm{CS}}(\mathbf{c}^*_1)$ *and* $\mathbf{f}^*_2 \in \mathcal{F}^*_{\mathrm{CS}}(\mathbf{c}^*_2)$ *such that* $(\mathbf{c}^*_1, \mathbf{f}^*_1)$ *and* $(\mathbf{c}^*_2, \mathbf{f}^*_2)$ *are confusable. Otherwise they are* **CS-distinguishable.**

Notice the role of CS in this definition. If both $\mathbf{c}^*_1$ and $\mathbf{c}^*_2$ are in $\mathcal{C}_{contact}$, we may expect detectable reaction forces, which may make the two positions distinguishable. However, a position in $\mathcal{C}_{contact}$ entails a detectable reaction force only if P is guaranteed to push sufficiently hard on the C-obstacle's boundary at that position. In order to know if it is the case, CS must be taken into consideration. For instance, if CS = $\mathbf{GD}(\mathbf{v})$, the reaction force on P **at a** position $\mathbf{c}^*$ in $\mathcal{C}_{contact}$ is guaranteed to be detectable if and only if, $\forall \mathbf{v}' \in \mathcal{U}_v(\mathbf{v}) : \|\mathbf{f}^*_{react}(\mathbf{c}^*, \mathcal{B}\mathbf{v}^*)\| > 2\varepsilon_f$. Thus, two positions $\mathbf{c}^*_1$ and $\mathbf{c}^*_2$, which are closer than $2\rho_c$ from each other, are GD(v)-distinguishable if and only if, $\forall \mathbf{v}' \in \mathcal{U}_v(\mathbf{v})$, the following three conditions hold:

- $\|\mathbf{f}^*_{react}(\mathbf{c}^*_1, \mathcal{B}\mathbf{v}^*)\| > 2\varepsilon_f$,

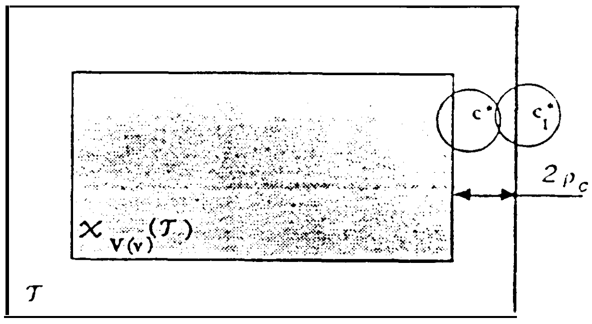


Figure **14: Kernel of a Rectangular Region in Free Space**

- $||\mathbf{f}^*_{react}(\mathbf{c}^*_2, \mathcal{B}\mathbf{v}^*)|| > 2\varepsilon_f$,
- $angle(\mathbf{f}^*_{react}(\mathbf{c}^*_1, \mathcal{B}\mathbf{v}^*), \mathbf{f}^*_{react}(\mathbf{c}^*_2, \mathcal{B}\mathbf{v}^*)) > 2\theta_f$.

The third condition is equivalent to: $angle(\nu(\mathbf{c}^*_1), \nu(\mathbf{c}^*_2)) > 2\theta_f + \phi(\mathbf{c}^*_1) + \phi(\mathbf{c}^*_2)$.

If two positions are CS-distinguishable, then it is guaranteed that the controller will be able to distinguish between them during a motion according to CS.

The kernel of **a** region for a given control statement is defined as follows:

DEFINITION 14: Let $\mathcal{S}$ *be a region in* $\mathcal{C}_{free} \cup \mathcal{C}_{contact}$ *and CS a control staiement. The* **kernel** ***of*** **S** ***for*** **CS** is *the subset* ***of*** *S defined as:* $\mathcal{X}_{\mathrm{CS}}(\mathcal{S}) \overset{\mathrm{def}}{=}$ *{c' ∈ S /* $\forall \mathbf{c}'^* \in \mathcal{C} - \mathcal{S}$: **c*** *and* **c'*** *are CS-distinguishable).*

Thus, $\mathcal{X}_{\mathrm{CS}}(\mathcal{S})$ is the subset of S which consists of every point in S that, given CS, cannot produce a measured position and a measured force consistent with those produced by a point outside S. The dependence of the kernel of a region on the control statement must be emphasized, since it seems to have been ignored by previous authors (e.g., [17]).

Example 11: Consider the target 7 in Figure **14.** It is a rectangular region in free space. $\mathcal{X}_{\mathrm{V(v)}}(\mathcal{T})$, **Vv,** is obtained by shrinking $\mathcal{T}$ by $2\rho_c$. Note that it is important that $\mathcal{T}$ be shrunk by $2\rho_c$, and not just by ρ_c. Indeed, as illustrated by the figure, any position **c*** closer than $2\rho_c$ from the boundary of 7 may produce a measured position consistent with that produced by a position $\mathbf{c}^*_1$ outside 7. ▮

Example 12: The target 7 in Figure 15 ***a*** is an edge in contact space adjacent to two other edges $\mathcal{E}_1$ and $\mathcal{E}_2$. The angle between 7 and $\mathcal{E}_1$ is smaller than $2\theta_f$, while the angle between 7 and $\mathcal{E}_2$ is greater than $2\theta_f$. CS = **GD(v)**, with v pointing downward. Assume that, $\forall \mathbf{v}^* \in \mathcal{U}_v(\mathbf{v})$, $\forall \mathbf{c}^* \in$ 7 $\cup \mathcal{E}_1 \cup \mathcal{E}_2$: $||\mathbf{f}^*_{react}(\mathbf{c}^*, \mathcal{B}\mathbf{v}^*)|| > 2\varepsilon_f$. $\mathcal{X}_{\mathrm{GD(v)}}(\mathcal{T})$ is drawn in bold line in Figure 15 6. The portion of 7, which is closer from $\mathcal{E}_1$ than $2\rho_c$ has been removed

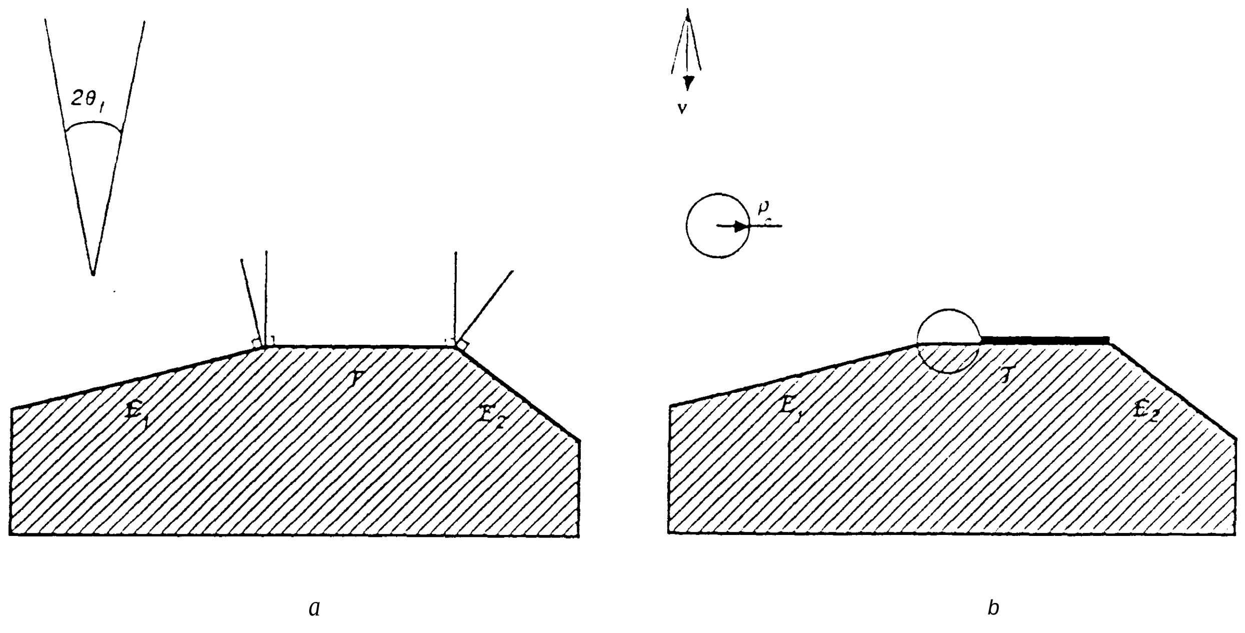


Figure 15: Kernel of an Edge **in** Contact Space

because actual reaction forces generated by contacts with $\mathcal{T}$ and with $\mathcal{E}_1$ can produce the same values on the force sensor. ∎

PROPERTY 2:
- $\forall \mathcal{S}_1 and \mathcal{S}_2 \subseteq \mathcal{C}_{free} \cup \mathcal{C}_{contact}$: $\mathcal{K}_{CS}(\mathcal{S}_1) \cup \mathcal{K}_{CS}(\mathcal{S}_2) \subseteq \mathcal{K}_{CS}(\mathcal{S}_1 \cup \mathcal{S}_2)$.
- *If* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are non-connected, then* $\mathcal{K}_{CS}(\mathcal{S}_1) \cup \mathcal{K}_{CS}(\mathcal{S}_2) = \mathcal{K}_{CS}(\mathcal{S}_1 \cup \mathcal{S}_2)$.

If a motion command is guaranteed to attain **a** point in the target kernel, then it is possible to reliably recognize achievement of the target when the only positions that are consistent with instantaneous sensing are in the target. This is illustrated by the following example and formalized further.

Example 13: Let us consider Example 11 again. If a motion is guaranteed to enter the kernel $\mathcal{K}_{V(v)}(\mathcal{T})$, then it is also guaranteed that at some instant $\delta t \geq 0$ during the motion, the measured position $c(\delta t)$ belongs to the region denoted $\mathcal{T}_{-\rho_c}$ in Figure 16. This region is obtained by shrinking $\mathcal{T}$ by ρ_c. When $c(\delta t) \in \mathcal{T}_{-\rho_c}$ is true, it is guaranteed that the target $\mathcal{T}$ has been achieved, since no actual position of P outside 7 is consistent with $c(\delta t)$ (do not confuse consistency and confusability!). I

Now, in order to characterize the motions which are guaranteed to attain the target kernel, let us introduce the notion of backprojection:

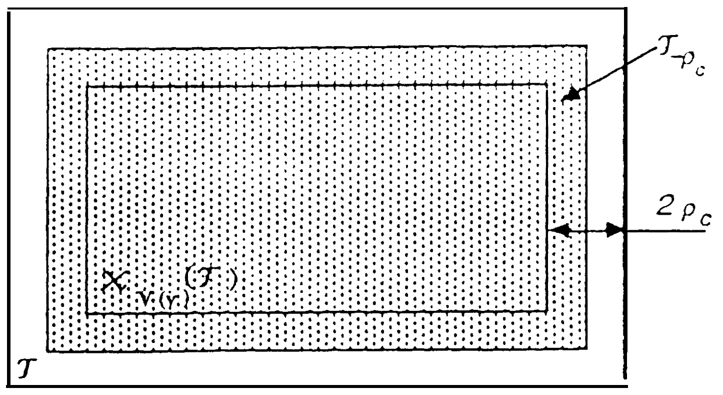


Figure 16: Recognition **of Achievement of a** Target in Free Space

DEFINITION 15: *Let 7 be a target in C and CS be a control statement. A* **back-**projection *from 7 for CS* is *any region $\mathcal{B}$ such that* Attain(7, CS, $\mathcal{B}$). $\mathcal{B}^{max}(\mathcal{T}$, CS) $\overset{def}{=}$ $\{c_s^*/\text{Attain}(\mathcal{T}$, CS, $\{c_s^*\})\}$ is *the maximal backprojection from 7 for CS.*

The notion of backprojection differs from the notion of preimage because it does not address -the target recognition issue. In the following, we only consider backprojections that are maximal. Thus, often, we do not mention that they are maximal.

Example 14: Figure 17 a shows the backprojection from a segment[13] $\mathcal{T}_1$ in free space. Any motion according to **V(v)** starting from within the backprojection is guaranteed to attain the target segment, although, due to position sensing uncertainty, no termination condition will be able to recognize it. Figure 17 6 shows the backprojection from an edge $\mathcal{T}_2$ in contact space, for GD(v). I

The following theorem can easily be proven from the previous two definitions:

THEOREM 2: $\mathcal{B}^{max}(\mathcal{X}_{CS}(\mathcal{T})$, *CS)* **is** *a preimage of $\mathcal{T}$ for* M = (CS,TC), *where* TC = $[\mathcal{K}^*_{pos}(c(\delta t), f(\delta t)) \subseteq \mathcal{T}]$.

Proof: According to the definition of a backprojection, any execution of M from within $\mathcal{B}^{max}(\mathcal{X}_{CS}(\mathcal{T})$, CS) is guaranteed to enter $\mathcal{X}_{CS}(\mathcal{T})$. In addition, whenever the effector point **P** is actually in $\mathcal{X}_{CS}(\mathcal{T})$, the termination condition TC specified in the theorem is guaranteed to be satisfied. Thus, any execution of M from within $\mathcal{B}^{max}(\mathcal{X}_{CS}(\mathcal{T})$, CS) is guaranteed to terminate before the motion has traversed the target. Since the termination condition cannot be satisfied as long as P is not actually in 7, no execution of M **can** terminate before entering 7. Thus, $\mathcal{B}^{max}(\mathcal{X}_{CS}(\mathcal{T})$, CS) is a preimage of 7 for M =

[13]Our convention is to use the word *segment* in free space and the word edge in contact space.

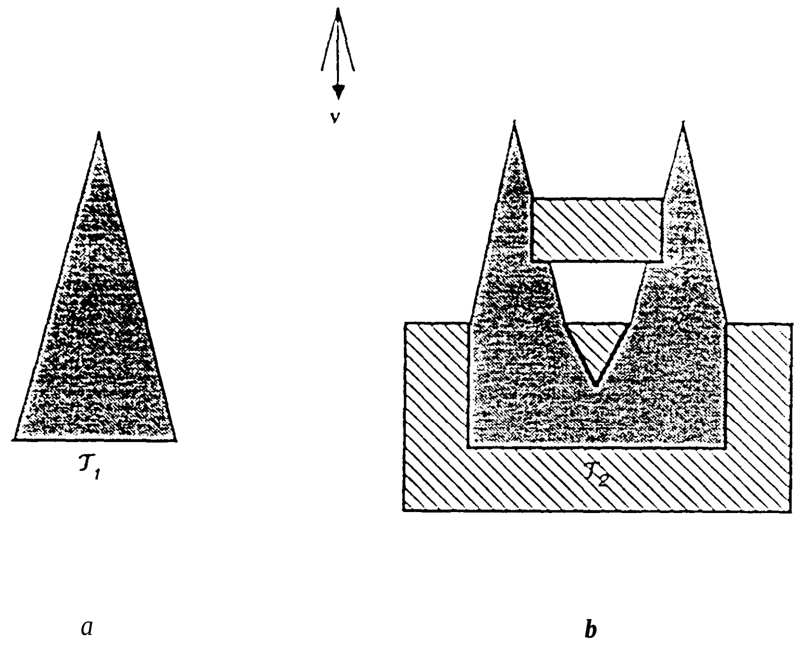


Figure **17: Examples of Backprojections**

(CS,TC). ▮

It is interesting to remark that the termination predicate is independent from the control statement.

If we have at our disposal an algorithm for computing target kernels (see Section 15), another one for computing backprojections (see Section 16), and a third one for computing $\mathcal{K}^{*}_{pos}(\mathbf{c}, \mathbf{f})$ (see Section 7), the above theorem directly provides a technique for computing preimages. Obviously, however, the technique is not guaranteed to construct a prcimage whenever one exists. For instance, in example 10, the goal region consists of a single point g in free space. $\mathcal{X}_{\mathbf{V}(\mathbf{v})}(\{g\}) = \emptyset$ and $\mathcal{B}^{max}(\mathcal{X}_{\mathbf{V}(\mathbf{v})}(\{g\}), \mathrm{V(v)}) = 0$. More generally, in the mini-world, any target in free space having one of its dimensions smaller than $2\rho_c$ has an empty kernel. Furthermore, when this technique generates a preimage, this preimage may not be-maximal as illustrated by the following example.

Example 15: Let the target 7 be the edge in contact space shown in bold line at Figure **18** *a*. Assume CS = $\mathbf{GD(v)}$, where v points vertically downward. Figure 18 *b* displays the backprojection $\mathcal{P}_1$ (shaded region) from the kernel of 7. Figure 18 c displays the backprojection $\mathcal{P}_2$ (shaded region) from 7. $\mathcal{P}_2$, which includes $\mathcal{P}_1$, is a preimage for M = $(\mathbf{GD(v)},[\|\mathbf{f}(\delta t)\| > \varepsilon_f])$. ▮

One ad-hoc way to improve the backprojection-from-kernel technique is to complement

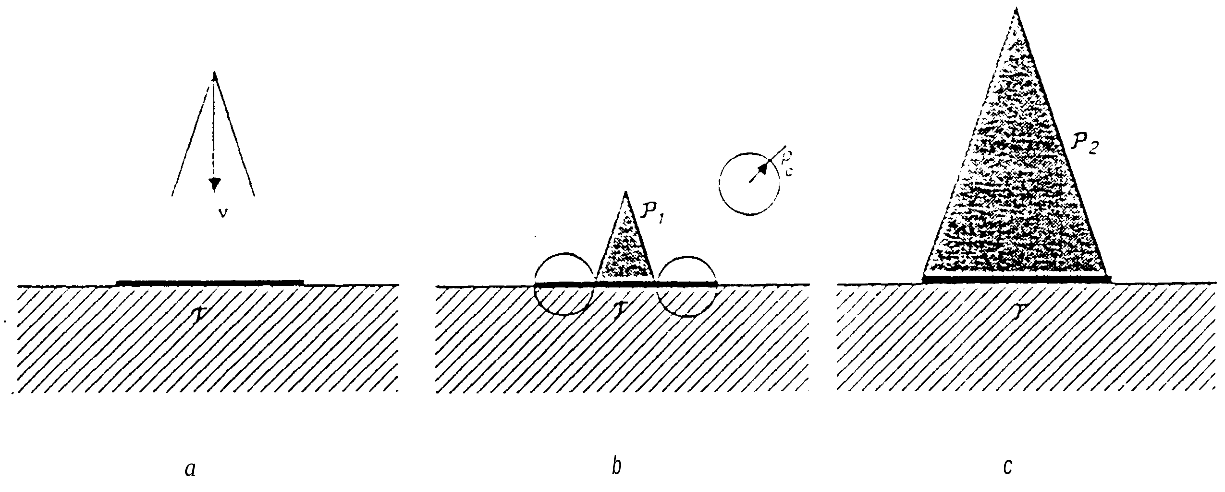


Figure 18: **Backprojection from the** Kernel of **an Edge in Contact Space**

it by predcfined solutions for well-identified particular cases such as those presented in the above two examples. Whenever such **a** case is identified, the corresponding solution is retrieved and selected; in all other cases, the more general backprojection-from-kernel technique is applied.

In the next two sections, we give two algorithms, one for computing the kernel of a region, the other for computing the maximal backprojection from a region. The applicability of both algorithms is limited to the mini-world with generalized damper control. The computation of $\mathcal{K}^{*}_{pos}$ (c, **f)** in the mini-world has already been presented in Section 7.

15 Computation of Region Kernels

In this section we describe an algorithm for computing the kernel $\mathcal{X}_{\mathbf{v}}(\mathcal{S}) = \mathcal{X}_{\mathbf{GD(v)}}(\mathcal{S})$ of a closed polygonal region $\mathcal{S}$ in the mini-world for generalized damper control. Examples **11** and 12 shown **above** already illustrated the c&es of a region in free space and an edge in contact space. Here, Figure 19 is used to illustrate the computation carried out by the algorithm with a region lying both in free space and in contact space.

The algorithm consists of the following two major steps. Comments are printed in italics. The example shown at Figure 19 is commented next to the description of the algorithm.

Algorithm TK:

1.. *(See Figure 19 b)*

- Decompose $\mathcal{S}$ into convex polygonal regions $\mathcal{S}_i$ $(i = 1, 2, ...)$ such that $\bigcup_i \mathcal{S}_i = \mathcal{S}$.

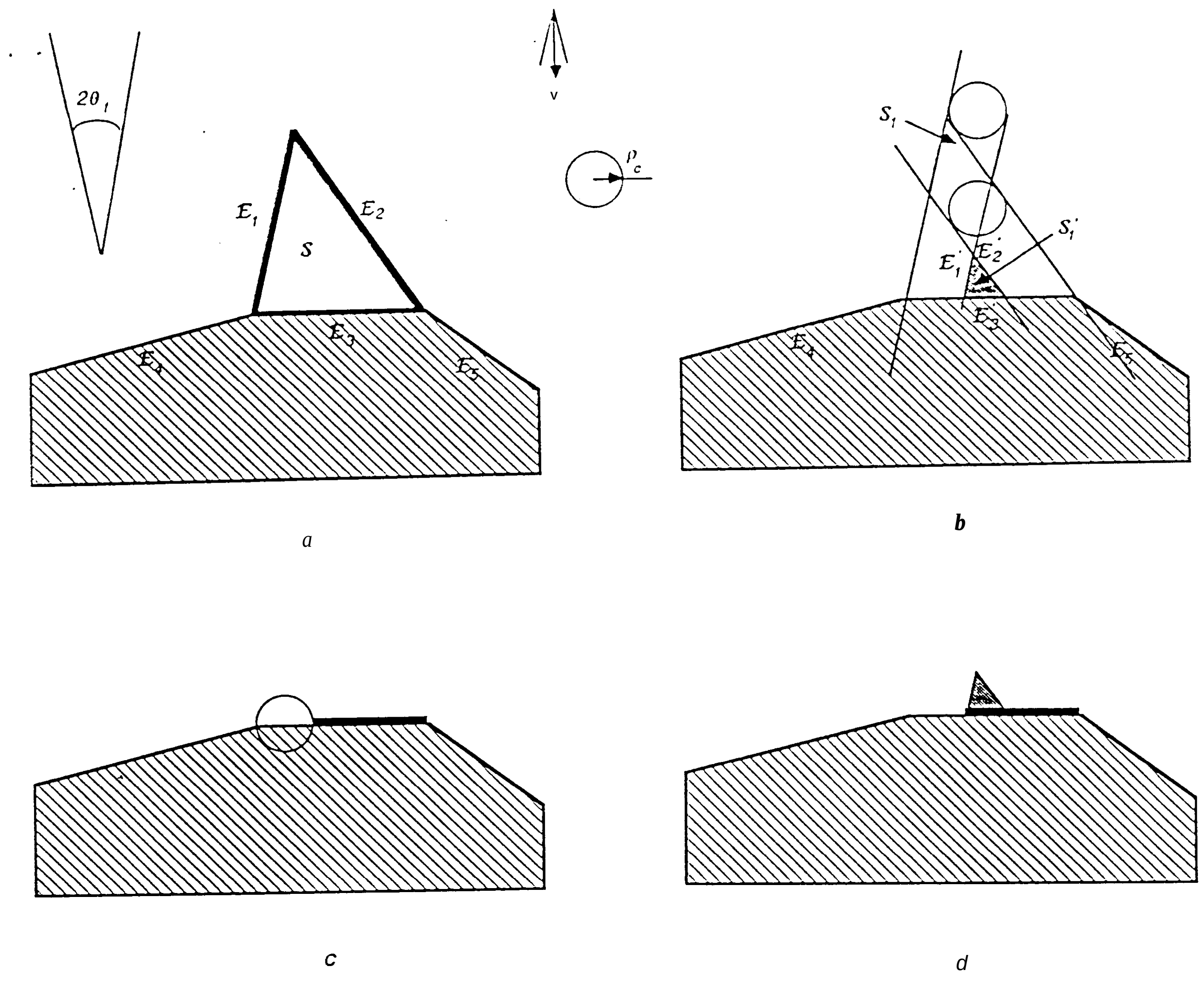


Figure 19: Kernel of a Region Lying in Both Free Space and Contact Space

- Represent every $\mathcal{S}_i$ as the conjunction of the linear constraints imposed by [illegible] line supporting a segment of $\mathcal{S}_i$'s contour, i.e.:
  $\mathcal{S}_i \overset{\text{rep}}{=} \bigwedge_{k_i}[x \cos\alpha_{k_i} + y \sin \alpha_{k_i} \leq d_{k_i}]$.
- $\mathcal{E}_{i,k_i} \leftarrow$ segment of $\mathcal{S}_i$'s contour supported by the line $[x \cos \alpha_{k_i} + y \sin \alpha_{k_i} = d_{k_i}]$.
- $\mathcal{S}_i' \leftarrow \bigwedge_{k_i}[x$ [illegible] $\alpha_{k_i} +$ [illegible] $\alpha_{k_i} \leq d_{k_i} - \beta_{k_i}]$, where $\beta_{k_i} = 2\rho_c$ if $\mathcal{E}_{i,k_i}$ C $\mathcal{C}_{free}$, and $\beta_{k_i} = 0$ if $\mathcal{E}_{i,k_i} \subset \mathcal{C}_{contact}$.

  (This operation results in *shifting in by* $2\rho_c$ *every segment of* $\mathcal{S}_i$ *'s contour, if it lies in free space. Indeed, every position* in $\mathcal{S}_i$ *closer than* $2\rho_c$ *from such a segmeni is confusable with a position in free space on the oiher side* ***of*** *the segmeni. Th*is is *not the case* if *the segment is an edge* in *contact space, since the oiher side* ***of*** *such an edge* is *inierior to a C-obstacle and so, is not accessible".)*
- . $\mathcal{X}_v(\mathcal{S}) \leftarrow \bigcup_i \mathcal{S}_i'$.

2. *(See Figure 19 c)*

- Mark every edge $\mathcal{E}$ C $\mathcal{C}_{contact}$ such that:
  $\forall c^* \in \mathcal{E} \Rightarrow \forall v^* \in \mathcal{U}_v(v) : ||f^*_{react}(c^*, Bv^*)|| > 2\varepsilon_f$.
- $\mathcal{E}_j$ (j $= 1, 2, . ..) \leftarrow$ edges of the polygon bounding S, which are in $\mathcal{C}_{contact}$ and are marked.
- Compute $\mathcal{E}_j'$ as follows:
  - $\mathcal{E}_j'$ t $\mathcal{E}_j$
  - for every marked edge $\mathcal{E}$ **C** $\mathcal{C}_{contact}$ -S, such that $angle(\nu(\mathcal{E}), \nu(\mathcal{E}_j)) \leq 2\theta_f$, do:
  $\mathcal{E}_j' \leftarrow \mathcal{E}_j' - \mathcal{E} \oplus \Sigma_{2\rho_c}(0)$.
- $\mathcal{X}_v(\mathcal{S}) \leftarrow \mathcal{X}_v(\mathcal{S})$ U $(\bigcup_j \mathcal{E}_j')$.

Example 16: Consider Figure 19. The region S is a triangle (Figure 19 a) and v is pointing downward.

'At step 1, a single region $\mathcal{S}_1$ = S is considered. Two segments of $\mathcal{S}_1$'s contour, $\mathcal{E}_1$ and $\mathcal{E}_2$, are situated in free space; the other segment, $\mathcal{E}_3$, is an edge in-contact space. Step 1 shifts $\mathcal{E}_1$ and $\mathcal{E}_2$ in by $2\rho_c$, while it leaves $\mathcal{E}_3$ unchanged (the thickness of the obstacle is greater than $2\rho_c$). Figure 19 *b* displays $\mathcal{S}_1'$.

At step **2,** three edges in contact space, $\mathcal{E}_3$, $\mathcal{E}_4$ and $\mathcal{E}_5$ are marked; indeed, given v, they are the only ones guaranteed to produce **a** detectable reaction force. The angle made by $\mathcal{E}_3$ and $\mathcal{E}_4$ is less than $2\theta_f$; the angle made by $\mathcal{E}_3$ and $\mathcal{E}_5$ is greater than $2\theta_f$. Step 2 removes the portion of $\mathcal{E}_3$ that is closer from $\mathcal{E}_4$ than $2\rho_c$ (see Figure 19 c).

"'Actually, this is true only if the thickness of the C-obstacle is greater than $2\rho_c$.

Figure 19 *d* shows the computed kernel of $\mathcal{S}$ for GD(v). ∎

Let n be the number of vertices in contact space, and q the number of vertices of $\mathcal{S}$. The time complexity of Step 1 of algorithm TK is the complexity of the decomposition of $\mathcal{S}$ into convex polygons. The complexity of a non-optimal decomposition is $\mathcal{O}(q \log q)$ [52]. The time complexity of Step 2 is $\mathcal{O}(n \times q)$. In general $n \gg q$, and the overall complexity of TK is $\mathcal{O}(n \times q)$.

16 Computation of Maximal Backprojections

In this section we describe an algorithm for computing the maximal backprojection from a region S in the mini-world for generalized damper control. We first present an algorithm applicable when $\mathcal{S}$ is either an edge in contact space or a segment in free space. Then, we propose an algorithm for treating the case when S is a collection of edges and segments, or a two-dimensional region. The first algorithm is basically a more detailed version of the algorithm described by Erdmann [17,18].

Let us first consider a region S, which is either an edge in contact space[15], or a segment in free space. The control statement is GD(v). The algorithm below computes $\mathcal{B}^{max}(\mathcal{S}, GD(v))$. It consists of five major steps. Step 1 eliminates some non-interesting cases. Steps 2 through 5 actually compute $\mathcal{B}^{max}(\mathcal{S}, GD(v))$ and they are illustrated by Figure 20.

Algorithm MB1:

1. *(This step is here for completeness. It treats some non-interesting cases, resulting in* B-‘(S, GD(v)) = S.)

   If one of the following two conditions is not satisfied:

   (1) the negative velocity cone at any position on $\mathcal{S}$ lies entirely in one of the two sides (open half-planes) of the line supporting $\mathcal{S}$,

   (2) if $\mathcal{S} \subset \mathcal{C}_{contact}$, then the negative velocity cone at any point on $\mathcal{S}$ lies within the side pointed out by the outgoing normal to S,

   then $\mathcal{B}^{max}$(S, GD(v)) $\leftarrow \mathcal{S}$, and exit.

2. ***(This steps consists** of **marking every C-obstacle's vertex where*** P *could either stick,* ***or slide away from S. It is illustrated by Figure 20 a.)***

[15]If we want $\mathcal{S}$ to be a portion of an edge, we first partition the edge into shorter colinear edges, such that one of the new edges is exactly S.

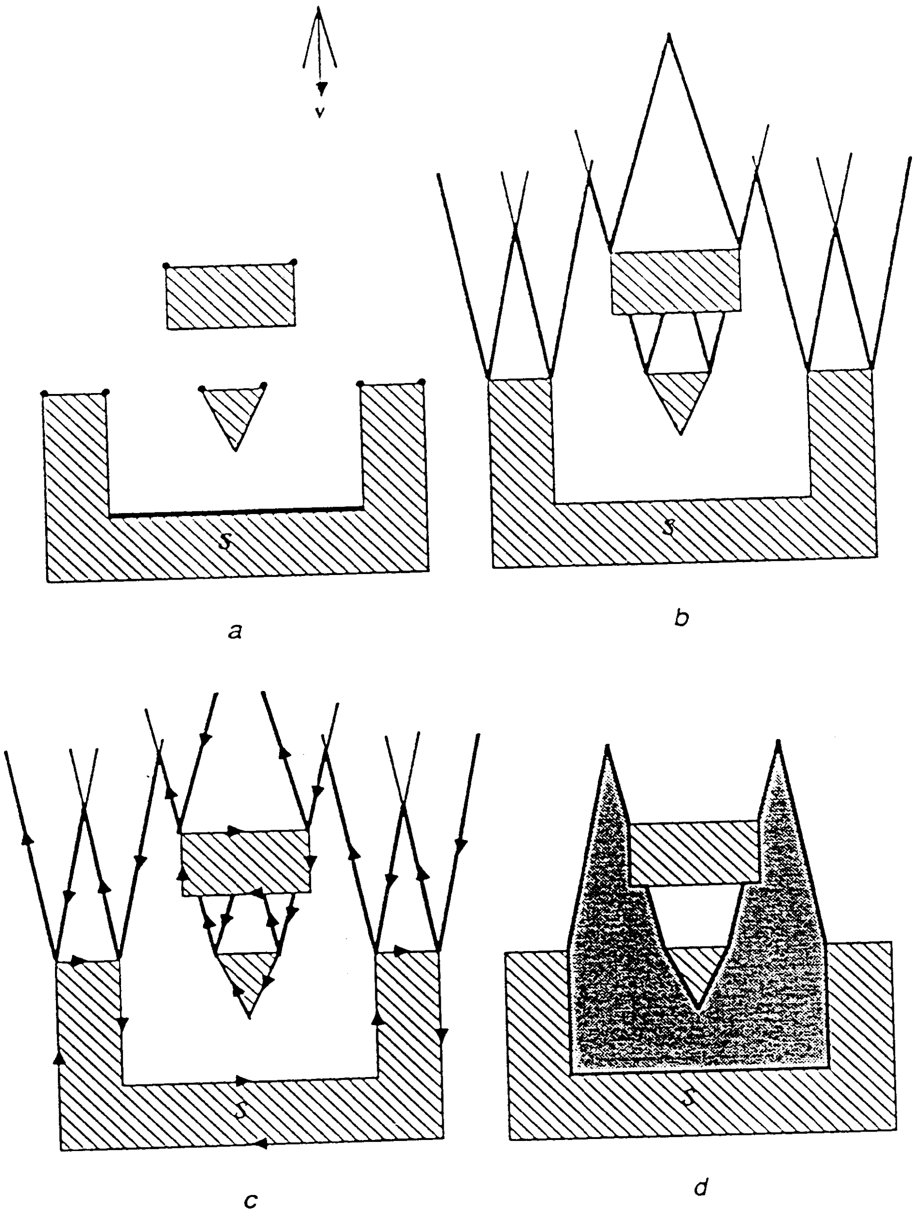


Figure 20: Illustration of Steps 2 through 5 of Algorithm MB1

- Mark every vertex X of every C-obstacle, which satisfies either one **of** the following two conditions:

  (1) $\exists \mathbf{v}^* \in \mathcal{U}_v(\mathbf{v})$: $\mathbf{v}^*$ points inside the friction cone at X, and X is not an extremity of $\mathcal{S}$,

  (2) X is the extremity of both $\mathcal{S}$ and another edge $\mathcal{E}$, and $\exists \mathbf{v}^* \in \mathcal{U}_v(\mathbf{v})$: $\mathbf{v}^*$ points outside the friction cone at any position on $\mathcal{E}$, and its projection on $\mathcal{E}$ points away from $\mathcal{G}$.

- If $\mathcal{S}$ is a segment in free space, then treat its extremities as fictitious vertices, and mark both of them.

3. *(The contour of the backprojeciion from $\mathcal{S}$ is made of $\mathcal{S}$ itself, portions of C-obstacles' edges, and poriions of the extreme rays of the negaiiue velocity cones erected at the marked vertices. This step contributes in building the contour by erecting the extreme rays and determining their interesting poriions. Ii is illustrated by Figure 20 b.)*

- Erect the two extreme rays of the negative velocity cone at every marked vertex; $\mathcal{LR} \leftarrow$ list of erected rays; activate every ray in $\mathcal{LR}$.

  (Portions of a ray can be active or inactive. Activating a ray makes the whole half-infinite line aciive.)

- While $\mathcal{LR} \neq \emptyset$ do:

  - $\mathcal{R} \leftarrow$ first ray in $\mathcal{LR}$; remove $\mathcal{R}$ from CR.
  - If $\mathcal{R}$ intersects either S, or the active portion of another ray, or a C-obstacle's edge, then:

    - $Z_1 \leftarrow$ nearest intersection point from the marked vertex;

    - inactivate $\mathcal{R}$ beyond Z_1; denote Z_1 as an extremity of $\mathcal{R}$; if the intersection at Z_1 occurs with a ray R', remember $\mathcal{R}'$ as the reason for inactivating $\mathcal{R}$ beyond Z_1;

    - if $\mathcal{R}$ is remembered as the reason for inactivating a ray $\mathcal{R}''$ beyond point Z_2, and if Z_2 is situated on the now inactive portion of $\mathcal{R}$, then: erase Z_2 as an extremity of $\mathcal{R}''$; reactivate $\mathcal{R}''$ beyond Z_2; and reinsert $\mathcal{R}''$ into LX.

- Whenever the extremity Z of the active portion of a ray is located on a C-obstacle's edge, partition this edge into two colinear edges adjacent at Z.

(Figure 20 b shows the erected rays. The aciive portions of them are represented as bold lines. The lowest edge of the upper C-obstacle is decomposed into smaller colinear edges.)

4. *(This step gives an orientation to each of the lines that may participate to the contour of the backprojcciion from $\mathcal{S}$. It prepares step 5, which consists of tracing along these lines. See Figure 20 c.)*

   - Orient S in such a way that any $\mathbf{v}^* \in \mathcal{U}_v(\mathbf{v})$ points toward the right of this orientation.
   - Orient each ray so that the interior of the negative velocity cone lies on the right-hand side.
   - Orient every edge of every C-obstacle's contour so that the ingoing normal to the edge points toward the right.

5. *(This' step consists of tracing oui the backprojection region by tracing along some of the lines according to their orientafion. During this process, the backprojection always lies on the left side of the line that is currently traced. We get a list of the successive vertices on the contour of the backprojection; this list is denoted $\partial\mathcal{B}$. See Figure 20 d.)*

   - $F_0 \leftarrow$ initial extremity of S (according to the orientation given to $\mathcal{S}$);
     $F_1 \leftarrow$ final extremity of S;
     $\mathcal{E} \leftarrow$ s;
     $i \leftarrow 1$; $\partial\mathcal{B} \leftarrow (F_0, F_1)$.
   - While $F_i \neq F_0$ do:

     - $\mathcal{E}$ t first active portion of a ray or C-obstacle's edge starting from F_i on the left of $\mathcal{E}$;

     - i t $i + 1$; $F_i \leftarrow$ final extremity of $\mathcal{E}$; insert F_i at the end of $\partial\mathcal{B}$.
   - $\mathcal{B}^{max}(\mathcal{S}, \mathbf{GD}(\mathbf{v})) \leftarrow \mathcal{P}olygon(\partial\mathcal{B})$, where $\mathcal{P}olygon(\partial\mathcal{B})$ is the function that evaluates to the closed polygonal region bounded by the contour linking the vertices listed in $\partial\mathcal{B}$[16].

Let n be the number of vertices in contact space. Step **2** of **MB1** marks $\mathcal{O}(n)$ vertices. At Step 3, $\mathcal{O}(\ )$rays are erected. Each one has O(n) intersections with other rays. These intersections can all be computed at the beginning of the iteration in the second operation of Step 3. During the iteration, each ray is reinserted $\mathcal{O}(n)$ times in CR. Thus, the complexity of Step 3 is $\mathcal{O}(n^2)$. This is also the complexity of the overall algorithm.

Consider now a region S = $\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_q$, where $\mathcal{S}_i$, Vi $\in [1, q]$, is either an edge in contact space, or a segment in free space. For all i = 1 to q, we can compute $\mathcal{B}^{max}(\mathcal{S}_i, \mathbf{GD}(\mathbf{v}))$ using

"Computing the maximal backprojection as a *closed* polygonal region results in inserting some portions of rays abutting at sticking edges. However the probability that a motion reaches such a vertex is zero.

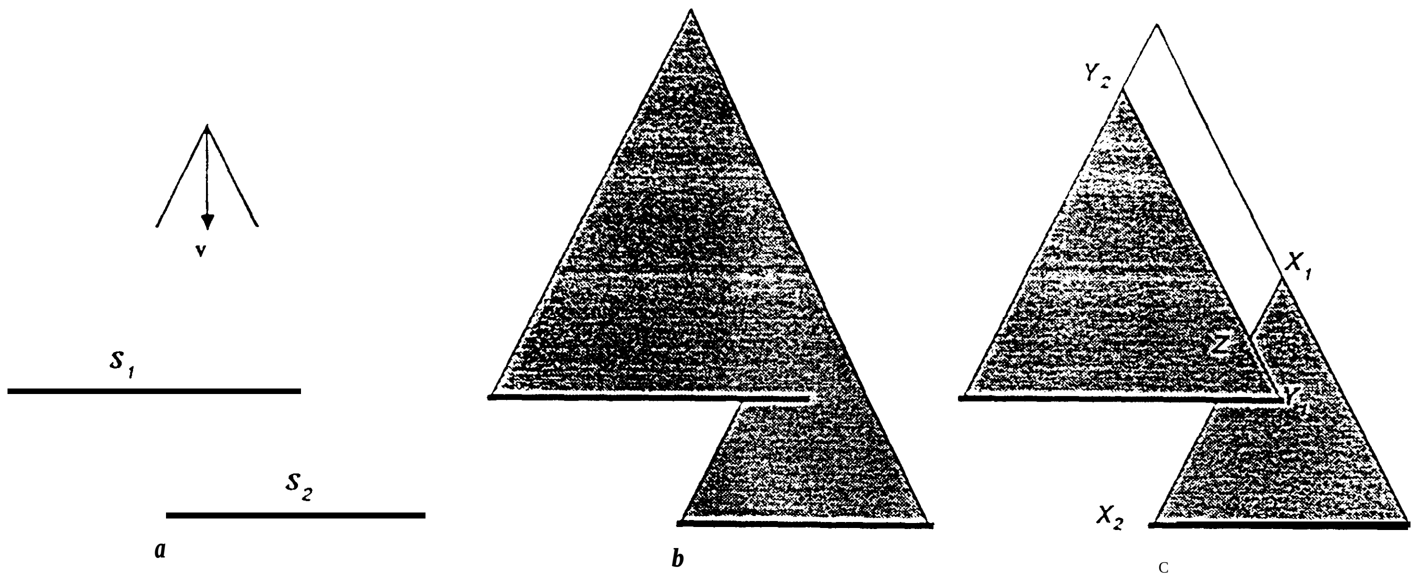


Figure 21: $\bigcup_i \mathcal{B}^{max}(\mathcal{S}_i, \mathrm{GD}(\mathbf{v})) \neq \mathcal{B}^{max}(\bigcup_i \mathcal{S}_i, \mathrm{GD}(\mathbf{v}))$

algorithm **MB1**. Obviously, $\bigcup_i \mathcal{B}^{max}(\mathcal{S}_i, \mathrm{GD}(\mathbf{v}))$ is a backprojection from S for $\mathrm{GD}(\mathbf{v})$. But, as shown by the following example, it may not be the maximal one. Indeed, there may exist positions in C from where we are certain that a motion according to $\mathrm{GD}(\mathbf{v})$ will attain either one of the $\mathcal{S}_i$, without knowing which one. Such positions are in $\mathcal{B}^{max}(\mathcal{S}, \mathrm{GD}(\mathbf{v}))$, but in none of the $\mathcal{B}^{max}(\mathcal{S}_i, \mathrm{GD}(\mathbf{v}))$.

Example **17:** Consider the case where $\mathcal{C}_{free}$ = C. Let S be made of two segments $\mathcal{S}_1$ and $\mathcal{S}_2$ as shown in Figure 21 a. Obviously the maximal backprojection from $\mathcal{S}_1 \cup \mathcal{S}_2$ with v pointing downward (shaded region in Figure 21 *b*) is larger than the union of the maximal backprojections from $\mathcal{S}_1$ and $\mathcal{S}_2$ (shaded regions in Figure 21 c). ∎

When a region S consists of several edges and/or segments $\mathcal{S}_i$, the backprojection from S for some commanded velocity v is a strict superset of the union of the backprojections from the individual edges/segments $\mathcal{S}_i$, if and only if there exist i and j such that: a portion X_1X_2 of a right ray[17] bounding $\mathcal{B}^{max}(\mathcal{S}_i, \mathrm{GD}(\mathrm{CS}))$ and a portion Y_1Y_2 of a left ray bounding $\mathcal{B}^{max}(\mathcal{S}_j, \mathrm{GD}(\mathbf{v}))$ intersect at a point Z, within the two portions or at one of their extremities X_2 and/or Y_1 (see Figure 21 *c*). Let us denote $\widehat{X_1ZY_2}$ the combination of the two segments X_1Z and ZY_2, which do not lie in the interior of $\mathcal{B}^{max}(\mathcal{S}_j, \mathrm{GD}(\mathbf{v}))$ and $\mathcal{B}^{max}(\mathcal{S}_i, \mathrm{GD}(\mathbf{v}))$. In order to obtain the maximal backprojection from S, we have to complete the union of the maximal backprojections from the individual edges/segments $\mathcal{S}_i$ by the maximal backprojection from $\widehat{X_1ZY_2}$ (white region in Figure 21 c), for every such intersections.

[17]We *can always* distinguish between the right and left extreme rays of the negative velocity *cone.*

The computation of the maximal preimage of $X_1\widehat{Z}Y_2$ is achieved by the algorithm **MB2** below, which is a direct adaptation of **MB1**.

Algorithm MB2:

1. • Mark every vertex of every C-obstacle as in Step 2 of **MB1**.
   • Mark X_1 and Y_2.

2. Erect rays, activate portions of rays, and partition edges exactly as in Step 3 of **MB1**.

3. Orient lines exactly **as** in Step **4** of **MB1**.

4. • $F_0 \leftarrow Y_2$; $F_1 \leftarrow Z$; $F_2 \leftarrow X_1$;
   $\mathcal{E} \leftarrow F_1F_2$;
   $i \leftarrow 2$; $\partial\mathcal{B} \leftarrow (F_0, F_1, F_2)$;
   • While $F_i \neq F_0$ do:

     - $\mathcal{E} \leftarrow$ first active portion of a ray or C-obstacle's edge starting from F_i on the left of $\mathcal{E}$;

     - $i \leftarrow i + 1$; $F_i \leftarrow$ final extremity of I; insert F_i at the end of $\partial\mathcal{B}$.

   • $\mathcal{B}^{max}(X_1\widehat{Z}Y_2, \mathbf{GD(v)}) \leftarrow \mathcal{P}olygon(\partial\mathcal{B})$, where ***Polygon(%)*** is the function that evaluates to the closed polygonal region bounded by the contour linking the vertices listed in $\partial\mathcal{B}$.

The complexity of **MB2** is the same that the complexity of **MB1**, i.e. $\mathcal{O}(n^2)$, where n is the number of vertices in contact space. However, since MB1 is applied before MB2, results of computations done by **MB1** can be saved, and re-used at Steps 1, 2, and 3 of MB2.

Let $\mathcal{S} = \bigcup \mathcal{S}_i$ be a finite union of segments in free space and edges in contact space. The maximal backprojection from S is computed by the algorithm **MB3** below.

Algorithm MB3:

1. Compute $\mathcal{B} = \bigcup \mathcal{B}^{max}(\mathcal{S}_i, \mathbf{GD(v)})$ using **MB1**.

2. While the boundary of $\mathcal{B}$ includes two successive segments X_1Z and ZY_2, such that X_1Z is supported by a right ray and ZY_2 by a left ray, do:

$$\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}^{max}(X_1\widehat{Z}Y_2, \mathbf{GD(v)})$$

   where $\mathcal{B}^{max}(X_1\widehat{Z}Y_2, \mathbf{GD(v)})$ is computed using MB2.

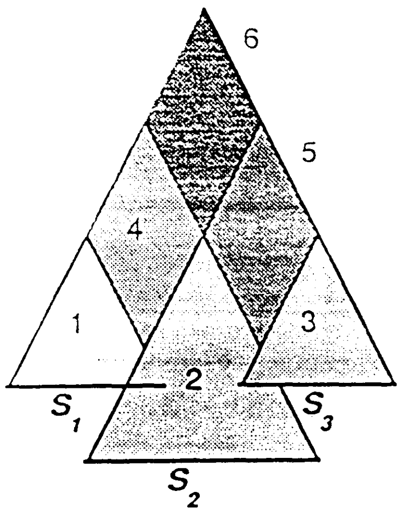


Figure 22: **Illustration of Algorithm MB3**

When **MB3** terminates, $\mathcal{B} = \mathcal{B}^{max}(S, GD(\mathbf{v}))$. Let n be the number of vertices in contact space and q the number of edges and segments in S. The time complexity of MB_3 **is** $\mathcal{O}(q \times n^2)$.

Example 18: Figure 22 illustrates the computations performed by **MB3.** The region $\mathcal{S}$ consists of three edges $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$. The commanded velocity points downward. Step **1** computes the' regions marked I, 2 and 3; these are the backprojections from $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, respectively. Step 2 first computes the regions marked 4 and 5. This creates two intersecting right and left **rays,** so that Step 2 iterates and produces the region marked 6. The resulting backprojections is the union of regions **1** through 6. ∎

Finally, if S is a closed region bounded by straight edges and/or segments $\mathcal{S}_i$, i = 1 to q (see for instance the kernel region at Figure 19), then $\mathcal{B}^{max}(\mathcal{S}, GD(\mathbf{v})) = \mathcal{B}^{max}(\cup\mathcal{S}_i, GD(\mathbf{v}))$, and can be computed using **MB3.**

Remark: MB3 is to be applied to a region $\mathcal{S}$, even if it is made of a single segment/edge, as illustrated below. Figure 23 a shows the backprojection from a single edge S computed by MB1, and Figure 23 b shows the backprojection computed by **MB3.** In this example, the-difference between the two backprojections is computed at Step 2 of MB3. This case can happen only when **a** vertex is the extremity of two edges along which the motion is guaranteed to slide. Including the difference into the backprojection is consistent with computing backprojections as closed regions.

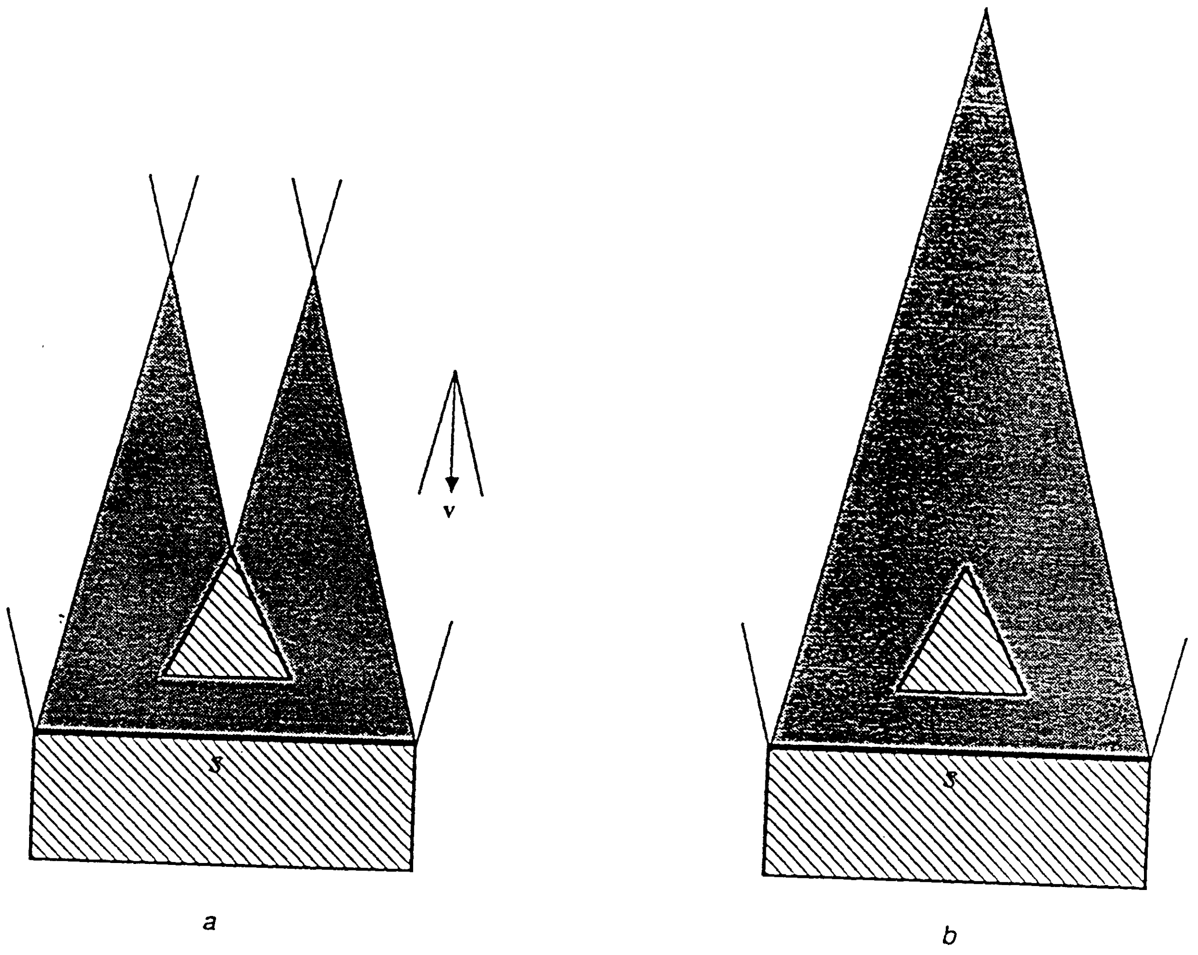


Figure 23: A Difference Between MB1 and MB3

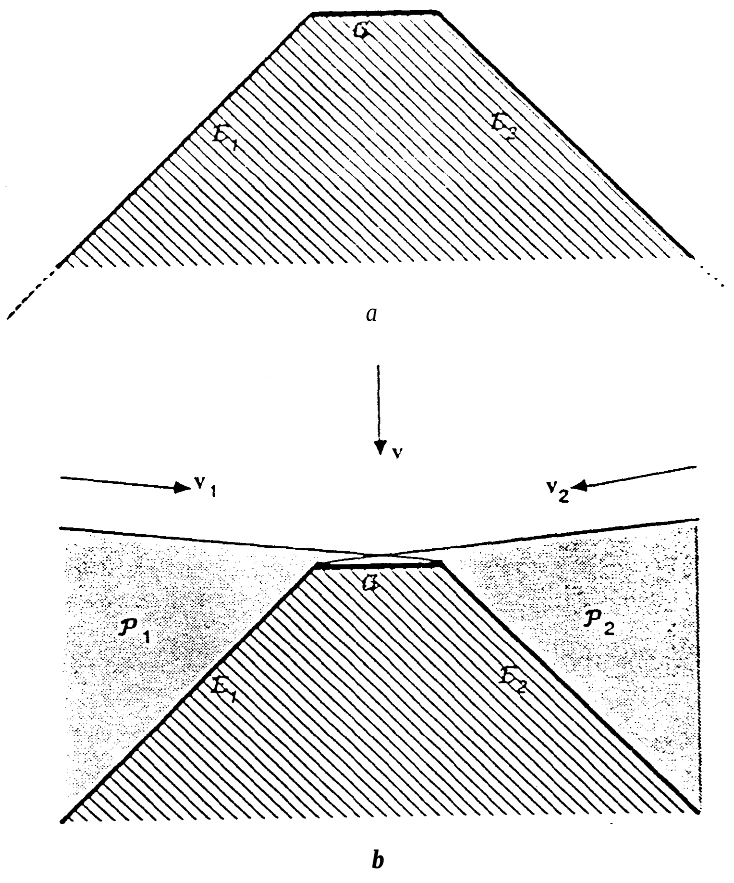


Figure **24: The Point-onto-Hill Problem**

17 Generation of Conditional Strategies

The simple kind of graph searching algorithm used in Section 6 to introduce the preimage backchaining approach can only generate linear strategies, i.e. sequences of motion commands. However, some motion planning problems only admit conditional strategies for solutions. In addition, as suggested in Section 10, conditional strategies are an alternative to the use of termination conditions with history. In this section, we extend the original framework in order to make it **possible** generating conditional plans.

Let us start with an example illustrating the need for conditional strategies. We use this example to sketch an approach for planning such strategies. We give a systematic presentation of the approach next to the example.

Example 19: Consider the two-dimensional point-onto-hill problem depicted in Figure 24 a (this example is drawn from [30]). The "hill" consists of three edges, the top edge $\mathcal{G}$, the left edge $\mathcal{E}_1$, and the right edge $\mathcal{E}_2$. Both the left and the right edges are infinite half-lines abutting to the top edge[8]. The problem is to move the reference point, **P** onto $\mathcal{G}$. The initial region $\mathcal{I}$ is all free space. We assume perfect generalized damper control ($\mathbf{v}^* = \mathbf{v}$), no position feedback ($\rho_c = \infty$), perfect force sensing ($\varepsilon_f = 0$ and $\theta_f = 0$), and frictionless edges ($\forall c^* \in \mathcal{C}_{contact} : \phi(c^*) = 0$).

A motion of P with a commanded velocity pointing downward until contact (i.e. until f > 0) is guaranteed to terminate in $\mathcal{G} \cup \mathcal{E}_1 \cup \mathcal{E}_2$. Then, the orientation of the reaction force makes it possible determining which of the three edges has been actually attained. If it is the top edge, the goal is achieved; if it is the left edge, then P must be moved by sliding along $\mathcal{E}_1$ towards the right until the orientation of the measured force shows that **P** is in $\mathcal{G}$; if it is the right edge *a* sliding motion towards the left is needed.

A conditional strategy is necessary for solving this problem. Ideally, it may be generated as follows:

- First, the planner considers the goal $\mathcal{G}$ and generates two preimages of $\mathcal{G}$, $\mathcal{P}_1$ and $\mathcal{P}_2$, for two motion commands, $\mathbf{M}_1$ and $\mathbf{M}_2$ (see Figure **24** b); $\mathbf{M}_1 = (\mathbf{GD}(\mathbf{v}_1), \mathrm{TC})$, where $\mathbf{v}_1$ points toward the right, slightly downward, as shown in the Figure. $\mathbf{M}_2 = (\mathbf{GD}(\mathbf{v}_2), \mathrm{TC})$, where $\mathbf{v}_2$ points toward the left, slightly downward. In both motion commands, $\mathbf{TC} = [angle(\mathbf{f}(\delta t), \nu(\mathcal{G})) = 0]$. It turns out that $\mathcal{E}_1 \subset \mathcal{P}_1$ and $\mathcal{E}_2 \subset \mathcal{P}_2$. So, $\mathcal{E}_1$ is a preimage of $\mathcal{G}$ for $\mathbf{M}_1$, **and** $\mathcal{E}_2$ is a preimage of $\mathcal{G}$ for $\mathbf{M}_2$.

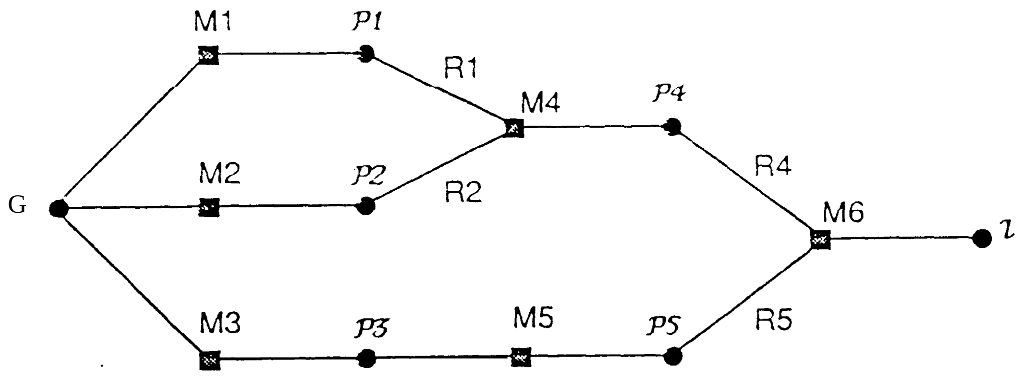
- Then, **the** planner considers $\{\mathcal{G}, \mathcal{E}_1, \mathcal{E}_2\}$ as **a** set of targets. It generates **a** preimage $\mathcal{P}$ of $\mathcal{G} \cup \mathcal{E}_1 \cup \mathcal{E}_2$ for the motion command $\mathrm{M} = (\mathbf{GD}(\mathbf{v}), \|\mathbf{f}(\delta t)\| > 0)$, where v points downward. Not only $\mathcal{P} = \mathcal{I}$, but the three conditions $angle(\mathbf{f}, \mathrm{n}) = 0$, with $\mathrm{n} = \nu(\mathcal{G})$, $\nu(\mathcal{E}_1)$, and $\nu(\mathcal{E}_2)$, are guaranteed to make it possible recognizing which target has actually been achieved at execution time. Thus, the planner can generate the following strategy:

```
execute $\mathrm{M} = (\mathbf{GD}(\mathbf{v}), \|\mathbf{f}\| > 0)$;
if $angle(\mathbf{f}, \nu(\mathcal{E}_1)) = 0$
      then execute $\mathbf{M}_1 = (\mathbf{GD}(\mathbf{v}_1), [angle(\mathbf{f}(\delta t), \nu(\mathcal{G})) = 0])$;
      else if $angle(\mathbf{f}, \nu(\mathcal{E}_2)) = 0$
            then execute $\mathbf{M}_2 = (\mathbf{GD}(\mathbf{v}_2), [angle(\mathbf{f}(\delta t), \nu(\mathcal{G})) = 0])$;
```

$\mathbf{RC}_1 = [angle(\mathbf{f}, \nu(\mathcal{E}_1)) = 0]$, $\mathbf{RC}_2 = [angle(\mathbf{f}, \nu(\mathcal{E}_2)) = 0]$, and $\mathbf{RC}_3 = [angle(\mathbf{f}, \nu(\mathcal{G})) = 0]$ are called ***recognition conditions.*** They allow the robot controller to identify which target has actually been achieved after the first motion in the strategy. ($\mathbf{RC}_3$ does not appear in the strategy because, if it evaluates to true, no action has to be taken.) ∎

Let us now formalize the approach outlined above. We define the preimage of a set of targets as follows:

[8] This example occurs in a space that slightly differs from the mini-world.



```
if R4 then execute M4;
                   if R1 then execute M1 ;
                          else execute M2;
                   exit;
if R5 then execute M5;
                  execute M3;
                  exit;
```

Figure **25: Representation of a Conditional Strategy as a Graph**

DEFINITION 16: *Let* $\mathcal{ST} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ be a *set of targets,* M = **(CS,TC) a** *motion command, and* $\mathbf{RC}_1, \mathbf{RC}_2, \ldots, \mathbf{RC}_n$ *n conditions, called* **recognition conditions.** $\mathbf{TC} = \mathbf{tp}(\delta t, \mathrm{c}_{[0,\delta t]}, \mathrm{f}_{[0,\delta t]})$. $\mathbf{RC}_i = \mathbf{rp}_i(\Delta, \mathrm{c}_{[0,\Delta]}, \mathrm{f}_{[0,\Delta]})$, *where* Δ is *the argument evaluating to the duration of the execution of* M, *when the execution terminates.*
A **preimage** *of* $\mathcal{ST}$ *for* M *and* $\{\mathbf{RC}_1, \ldots, \mathbf{RC}_n\}$ is *any region* $\mathcal{P}$ *in C such that executing* M *from within* $\mathcal{P}$ is *guaranteed to attain and terminate in* $\bigcup_i \mathcal{T}_i$, *in such a way that when the motion terminates the following two conditions are satisfied:*
- $\exists i \in [1, n]$: $\mathbf{RC}_i$ *evaluates to 'true',*
- $\forall i \in [1, n]$: $\mathbf{RC}_i$ *evaluates to 'true'* $\Rightarrow \mathcal{T}_i$ *has been achieved.*

The definition does not impose that the $\mathcal{T}_i$ be disjoint, so that several conditions $\mathbf{RC}_i$ may evaluate to *true* when the motion terminates.

In the following, we represent **a conditional** motion strategy as a labeled graph. Figure **25** shows an example of such a graph and the corresponding strategy. Nodes are alternatively *region nodes* and *motion nodes*. Each motion node N_M has a single parent $N_\mathcal{P}$ (a region node), and one or several children $N_{\mathcal{T}_1}$ through $N_{\mathcal{T}_n}$ (region nodes). Every arc connecting N_M to $N_{\mathcal{T}_i}$ is labeled by a recognition condition $\mathbf{RC}_i$. The region $\mathcal{P}$ labeling $N_\mathcal{P}$ is a preimage of $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ (the regions labeling $N_{\mathcal{T}_1}$ through $N_{\mathcal{T}_n}$) for M (the motion command labeling $\mathcal{N}_{\mathbf{M}}$) and $\{\mathbf{RC}_1, \ldots, \mathbf{RC}_n\}$. The graph has a root labeled by the initial region $\mathcal{I}$, with a unique child labeled by the null motion (the fictitious motion that does not move **P**). This node makes it possible selecting the first motion command from the sensory data at the initial position $\mathbf{c}^*_{init}$ of P. If every motion node in the graph has a single child, then the strategy is a linear one. We assume that the graph contains no cycle, but several nodes may have the same label.

Note that this graph may not define **a** unique strategy. Indeed, the conditions labeling the arcs originating at the same motion node need not be exclusive (i.e., $\forall i \neq j$: $\mathbf{RC}_i$ evaluates to *true* $\not\Rightarrow$ $\mathbf{RC}_j$ evaluates to false). We assume however that the arcs are scanned sequentially by the conditional branching statement according to some predefined order, as illustrated in Figure 25.

Now we can reformulate the preimage backchaining search algorithm as follows:

Algorithm PB:

1. Create the region nodes $\mathcal{N}_{\mathcal{I}}$ and $\mathcal{N}_{\mathcal{G}}$ labeled by the initial region $\mathcal{I}$ and the goal region $\mathcal{G}$, and the motion node $\mathcal{N}_{null}$ labeled by the null motion. Create an arc connecting $\mathcal{N}_{\mathcal{I}}$ to $\mathcal{N}_{null}$.
2. $ST \leftarrow \{\mathcal{G}\}$; $\mathcal{I}' \leftarrow \mathcal{I}$.
3. While $\mathcal{I}' \neq 0$ do:
   - Select a subset *st* of ST, a motion command M = **(CS,TC)** and *n* recognition conditions $\mathbf{RC}_i$, where *n is* the number of targets'in *st*. Compute a preimage $\mathcal{P}$ of *st* for M and $\{\mathbf{RC}_1, \ldots, \mathbf{RC}_n\}$.
   - Create a motion node $\mathcal{N}_{\mathbf{M}}$ labeled by M, *n* region nodes $\mathcal{N}_{\mathcal{T}_i}$ labeled by the targets in *ts,* and a region node $\mathcal{N}_{\mathcal{P}}$ labeled by $\mathcal{P}$. Create an arc labeled by $\mathbf{RC}_i$ from $\mathcal{N}_{\mathbf{M}}$ to every $\mathcal{N}_{\mathcal{T}_i}$, and an arc from $\mathcal{N}_{\mathcal{P}}$ to $\mathcal{N}_{\mathbf{M}}$.
   - $\mathcal{Z} \leftarrow \{\mathbf{c}^* \in \mathcal{I}' \cap \mathcal{P} \,/\, \forall \mathbf{c}'^* \in \mathcal{C} - \mathcal{I}' \cap \mathcal{P} : \not\exists \mathbf{c}, \mathbf{c}^* \in \mathcal{U}_c(\mathbf{c}) \text{ and } \mathbf{c}'^* \in \mathcal{U}_c(\mathbf{c})\}$.

     (We assume that we do not know how **P** *has reached* $\mathcal{I}$*, or will reach* $\mathcal{I}$*. Thus, even if a portion of* $\mathcal{I}$ *is in contact space, it is not guaranteed to produce a detectable reaction force. Since position sensing is the only sensing-data with which we can reliably plan, we define* $\mathcal{Z}$ *as the subset of the positions in* $\mathcal{I}' \cap \mathcal{P}$ *that cannot be confused with positions outside* $\mathcal{I}' \cap \mathcal{P}$ *using position sensing only.)*
   - If $\mathcal{Z} \neq \mathbf{0}$ then create an arc from $\mathcal{N}_{null}$ to $\mathcal{N}_{\mathcal{P}}$, and label it by the condition $[\mathcal{K}^*_{pos}(\mathbf{c}(0), \mathbf{f}(0)) \subseteq \mathcal{I}' \cap \mathcal{P}]$.

     (The arguments of $\mathcal{K}^*_{pos}$ *are* $\mathbf{c}(0)$ *and* $\mathbf{f}(0)$ *since the duration of the null motion is 0.)*

- $\mathcal{ST} \leftarrow \mathcal{S} \cup \{\mathcal{P}\}$; $\mathcal{I}' \leftarrow \mathcal{I}' - \mathcal{Z}$.

4. Return the strategy described by the subgraph made of all the nodes and arcs accessible from $\mathcal{N}_{\mathcal{I}}$.

At this point it remains the problem of computing the preimage of a set of targets. This problem requires being able both to compute the preimage of the union of the targets and to generate recognition conditions. Thus it is at least as difficult as the problem of computing the preimage of a single target. However, the backprojection method presented in the previous section can easily be adapted to handle a set of targets. Given the algorithms described in Section 15 and 16, the following theorem, which is an extension of Theorem 2, directly provides a technique for computing the preimage of a set of targets in the mini-world:

THEOREM 3: *Let* $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ *be the set of targets and CS a control statement,* $\mathcal{B}^{max}(\bigcup_i \mathcal{K}_{CS}(\mathcal{T}_i), \mathrm{CS})$ *is a preimage of* $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ *for* M = (CS,TC) *and* $\{\mathbf{RC}_1, \ldots, \mathrm{RC}_n\}$, *where* $\mathrm{RC}_i = [\mathcal{K}^*_{pos}(\mathbf{c}(\delta t), \mathbf{f}(\delta t)) \subseteq \mathcal{T}_i]$ *and* $\mathrm{TC} = [\bigvee_i \mathbf{RC}_i]$.

The technique provided by this theorem consists of backprojecting from the union of the kernels of the targets. Note that backprojecting from the kernel of the union of the targets could produce a larger preimage of the union of the targets, but then there would be no guarantee that the targets are distinguishable from each other.

This technique combined with the PB algorithm significantly augments the range of motion planning problems in the mini-world that we can solve automatically.

18 Bibliographical Notes

Research on robot motion planning has become active in the mid-seventies, when the goal of automatically programming robots from a geometrical description of the task was first considered attainable [32,33,50]. Since the early eighties, a great deal of effort has been devoted to this domain. Part of this effort was motivated, on the one side by the difficulties encountered in using explicit robot programming systems [28,29], and on the other side by the goal of introducing autonomous robots in hazardous environments (e.g. nuclear sites, space, undersea, mines). Although automating robot programming has turned out much more difficult than it first appeared, significant results with practical relevance have recently been obtained. Mazer's thesis [40] includes a chapter detailing why robot programming is difficult.

During the last ten years, most of the effort has been oriented toward solving the *path finding* problem, i.e. the problem of planning motions without uncertainty. Over the last few years, it has produced several major results, both theoretical and practical. Theoretical results mostly concern lower and upper bounds of the complexity of multiple variants of the path finding *problem* (e-g., see [48,6]). In particular, it has been shown that planning the motion of a robot with arbitrarily many degrees of freedom is PSPACE-hard [45], and that its time complexity is polynomial in the number of algebraic surfaces bounding the objects if the number of degrees of freedom is fixed [47]. Some path-finding methods have been produced as a side-effect of these results, but most of them involve very large constants and polynomial exponents. Another important result is the development of the notion of Configuration Space used throughout this paper, both as a conceptual tool and as a technique for exploring motion planning problems. This notion was popularized by Lozano-Pérez in the early eighties [34] and has given birth to many techniques for computing C-obstacles and finding collision-free paths among obstacles (e.g., [3,23,30,12,37]). Finally, relatively fast path-finding algorithms have been defined and implemented. Although these algorithms are not complete (they may fail to find a path while one exists), they can solve many practical problems. In particular, Faverjon and Tournassoud [21] reports a system using an adaptation of Khatib's *Potential Field* method [25] for planning the motion of *a* manipulator with eight degrees of freedom, operating in the complex environment of a nuclear reactor. Lozano-Pérez et al. [36] and Mazer [40] describe an impressive system, Handey, capable of planning all the motions required for assembling two polyhedral parts, in the absence of significant uncertainty. These practical techniques could bring substantial improvement to the programming of opkrations such as painting, welding, and riveting.

The problem of planning motions in the presence of uncertainty is conceptually more difficult than the path finding problem. It has attracted less attention so far, and less results have been produced. Three basic approaches to this problem have been developed to some extent.

The first has been proposed simultaneously by Lozano-Pérez [33] and Taylor [50], and is known *as* the *skeleton refining* approach. It consists of: first, retrieving a plan skeleton appropriate to the task at hand; and second, iteratively modifying the skeleton by inserting complements (typically sensor-based readings). Complements are decided after checking the correctness of the skeleton, either by propagating uncertainty through the steps of the plan skeleton [50], or by simulating several possible executions [33]. Subsequent contributions *to* the approach has been brought by Brooks [2], who developed a symbolic computation technique for propagating uncertainty forward and backward through plan skeletons, and by Pertin-Troccaz and Puget [43], who proposed techniques for verifying the correctness of a plan and amending incorrect plans. Backward propagation of uncertainty in this approach can be regarderd as a particular case of preimage backchaining with known motion commands.

The second approach to motion planning with uncertainty has been proposed by Dufay and Latombe [15], and is known *as* the *inductive learning* approach. It consists of assembling input partial strategies into a global one. First, during a training phase, the system uses the partial strategies to make on-line decisions and execute several instances of the task at hand. Second, during an induction phase, the system combines the execution traces generated during the training phase. and generalizes them into a global strategy. In fact, the training phase and the induction phase arc interweaved. The generation of a strategy for the task ends when new executions do not modify the current strategy. A system based on these principles has been implemented, and experimented successfully on several part mating tasks. Some aspects of this approach have been extended by Andreae [l].

Both the skeleton refining and inductive learning approaches deal with uncertainty in a second phase of planning. The plan skeleton and the local strategies used during the first phase could be produced using path-finding methods assuming null uncertainty. The second phase takes uncertainty into account, either by analyzing the correctness of the current plan, or by directly experimenting with the local strategies and combining them into execution traces shaped by actual errors. In contrast, the rationale of the third approach, preimage backchaining, is that uncertainty may affect the overall structure of a plan, in such a way that a motion strategy may not be generated by modifying or composing plans generated assuming no uncertainty. This can be illustrated by several examples. Consider the task of inserting a peg into a hole; in the absence of uncertainty (or equivalently with large clearance), the best strategy is to position the peg above the hole, to align the two axes, and to move the peg downward; in the presence of uncertainty, and with no chamfer, the best strategy is to tilt the peg before insertion, in order to be certain to generate a contact between the tip of the peg and the entrance of the hole. In a navigation task, with no uncertainty, the shortest route is the best; with uncertainty, a route providing enough landmarks to make execution monitorable is necessary; it may be very different from the first one.

In addition to be based on a different rationale, preimage backchaining is a much more rigorous approach to motion planning with uncertainty, than the other two approaches. Consequently, it is natural to expect that preimage backchaining raises new theoretical issues, which were not considered in the other approaches. It does not mean that these issues are not present in the other approaches, but that they are hidden by their ad-hocness. Conversely, solving these issues is a prerequisite to implementing the preimage backchaining approach, but not to implementing the other approaches. This expresses the fact that in general it is easier to build ad-hoc implementations of ad-hoc approaches than ad-hoc implementation of rigorous approaches. It explains why preimage backchaining has not yet been implemented, although as shown in this paper an implementation in a two-dimensional world is possible.

The preimage backchaining approach was first presented by Lozano-Pérez, Mason, and Taylor [35]. This paper set up most of the basic framework. It directly introduced a definition of preimages based on the use of termination condition with state. We think that our definition is simpler. It allows us to analyze theoretical issues related to the maximality of preimages in a step-by-step fashion. Key concepts prior to this definition are the directory of actual trajectories and the notions of consistency between actual and measured data. The concept of trajectory directory was previously used by Mason [39] as a tool to specify a termination condition with state.

Mason [39] investigated several control schemes for searching the graph of preimages. He proved the strong bounded-completeness of the original scheme presented in [35].

Erdmann [17,18] contributed to the approach in several ways. He separated the problem of computing *a* preimage into two sub-problems, reachability and recognizability. By considering reachability alone, he introduced the notion of backprojection, and used it for computing non-maximal preimage. Algorithm **MB1** is a detailed variant of Erdmann's algorithm. Algorithms **MB2** and **MB3** are improvements allowing to backproject from multiple edges/segments and from a polygonal region. Donald [12] presents another technique based on a plane sweep algorithm for computing the backprojection from a polygonal region. In order to compute preimages as backprojections, Erdmann introduced the notion of *first entry set*, which seems to be more powerful than the notion of target kernel. It is not clear however how this notion could be implemented in a program. An extension of the algorithm for computing backprojections to a three-dimensional configuration space is proposed in [17]. An investigation of friction modelling in configuration spaces with rotational axes is made in the same publication.

Donald [10,12] extended the preimage backchaining approach to model uncertainty by introducing the notion of generalized configuration space. He also inroduced the notion of *Error Detection and Recovery (EDR)* strategies, which *may* fail. Such strategies, however, either succeed or failed recognizably.

Buckley [5] proposed an application of preimage backchaining to the analysis of the correctness of a given motion plan. He also described a procedure for planning motion strategies in the forward direction. This procedure is based on the notion of *forward projection* (a more appropriate terminology would probably be *post-image*). The procedure requires to discretize configuration space into atoms and builds a transition graph between the atoms. It is not clear however how to select the resolution of the discretization. Buckley implemented a planner operating in a three-dimensional configuration space with translational axes.

Hopcroft and Wilfong [24], Valade [53], Laugier and Théveneau [31], and Koutsou [27] analyzed motions in contact space, without paying special attention to uncertainty. Within the preimage backchaining approach, their work could contribute in defining heuristics for searching the preimage graph.

The complexity of problems of planning compliant motions with uncertainty have been analyzed in a few papers (see [41,13,6,7]). Canny and Reif [6,7] have proven that the three-dimensional compliant motion planning problem is non-deterministic exponential time hard. Donald [13] has shown that planning a guaranteed planar multi-step strategy with sticking termination conditions can be done in time polynomial in the number of vertices in the polygonal environment, and roughly simply exponential in the number of steps in the strategy. The method presented in this paper corroborates this theoretical result.

19 Conclusion

In this paper, we have addressed the problem of planning motions with uncertainty. Autonomous robots need motion planning capabilities, and subtasks such as part mating and navigation in cluttered environments require Icing able to deal with uncertainty.

Wc have focused the paper on the preimage backchaining approach to motion planning in the presence of uncertainty. First, we have given a detailed formalization of the class of problems we *are* interested in (models of task geometry, task physics, motion commands, and uncertainty). Then, we have defined preimage backchaining and analyzed several underlying theoretical issues related to the power of termination conditions and the maximality of preimages. Finally, we have proposed the first complete set of algorithms making possible implementing preimage backchaining in a simplified world, the mini-world. These algorithms are based on the two concepts of target kernel and backprojection. These algorithms are certainly the most important outcome of this paper.

Although rather simple, the mini-world is still realistic enough for some applications. For instance, it can be the world of an omni-directional mobile robot, with a polygonal outline (typically a rectangular or hexagonal vehicle), moving among obstacles bounded by polygonal outlines (e.g., pieces of furniture, machines). Possible application tasks for such a robot is the transferring of objects in office, clean room, and shop-floor environments. We are currently implementing the proposed algorithm for a similar robot. Our goal with this implementation is not only to give an experimental validation of these algorithms. It is also to show that sophisticated methods for dealing with uncertainty, such as preimage backchaining, can make it possible building low-cost smart robots.

There-are many directions in which the prcimage backchaining approach could bc usefully extended. These are some of the questions we would like to answer in the future. How to build practical procedures for computing prcimages and solve realistic motion planning problems in spaces of dimension higher than 2 with rotational axes? What control schemes are the most appropriate to the preimage backchaining approach (for instance, Shekhar and Khatib [46] proposed a compliant scheme with selectable compliance center, which might result in larger preimages, but in a higher-dimensional space)? How to efficiently generate weak guaranteed strategies such as those proposed by Donald [12,14], in order -to build *a* reactive motion planner with provable properties? How to associate a monitoring plan to a motion plan, so that if during the execution of the motion plan an error exceeds uncertainty bounds, possible failure of the motion plan can be recognized by the monitoring plan executed in parallel? Answering these questions will require a lot more research.

References

[1] P.M.Andreae (1986). Justified *Generalization: Acquiring Procedure3 from Examples.* Ph.D. Dissertation, Technical Report 834, Artificial Intelligence Laboratory, MIT.

[2] R.A.Brooks (1982). *Symbolic Error Analysis and Robot Planning.* Intern. Journal of Robotics Research (IJRR) 1, 4.

[3] R.A.Brooks, T.Lozano-Perez (1983). *A Subdivision Algorithm in Configuration Space for Findpath with Rotation.* Proc. of the 8th Intern. Joint Conf. on Artificial Intelligence (IJCAI), Karlsruhe, FRG, August 1983.

[4] R.C.Brost (1986). *Aufomafic Grasping **in the** Presence **of** Uncertainty.* Proc. of the IEEE Intern. Conf. on Robotics and Automation, San Francisco, CA, April 1986.

[5] S.J.Buckley (1986). *Planning and Teaching Compliant Mofion* **Sfrafegies.** Ph.D. Disskrtation, Dept. of Elec. Eng. and Comp. SC., MIT, January 1986.

[6] J.F.Canny (1987). *The Complexity of Robot* **Motion** *Planning.* Ph.D. Dissertation, Dept. of Elec. Eng. and Comp. SC., MIT.

[7] .J.F.Canny, J.Reif (1987). *New Lower Bound Techniques **for** Robot Motion Planning Problems.* Proc. IEEE Symposium FOCS.

[8] W.Choi, J.C.Latombe (1988). *Planning and Ezecufing Robot Motion3 in the Presence **of** Contingencies.* In preparation.

[9] B.R.Donald (1984). *Mofion Planning With Six Degrees **of** Freedom.* Tech. Report 791, AI Lab., MIT, May 1984.

[10] B.R.Donald (1986). *A Theory **of** Error Detection and Recovery **for** Robot Mofion Planning with Uncertainty.* Preprints of the Intern. Workshop on Geometric Reasoning, Oxford, UK, June 1986.

[11] B.R.Donald (1987). *A Search Algorifhm for **Mofion Planning with Six Degrees of Freedom.*** Artificial Intelligence 31, 3, March 1987.

[12] B.R.Donald (1987). ***Error Deiecfion and Recovery** for **Robot Motion Planning wifh** Uncerfainfy.* Ph.D. Dissertation, Dept. of Elec. Eng. and Comp. SC., MIT, July 1987.

[13] B.R.Donald *(1987). The Complexity of Planar Compliant Motion Planning Under Uncertainty.* Tech. Report 87-889, Dept. of Comp. Sc.. Cornell University, Ithaca, NY, December 1987.

[14] B.R.Donald (1988). *A Geometric Approach to Error Defection and Recovery for Robot Motion Planning with Uncertainty.* To appear in Artificial Intelligence.

[15] B.Dufay, J.C.Latombe (1984). *An Approach to Automatic Robot Programming Based on Inductive Learning.* Intern. Journal of Robotics Research (IJRR) 3, 4.

[16] H.F.Durrant-Whyte (1987). *Uncertain Geometry in Robotics.* Proc. *of* the IEEE Intern. Conf. on Robotics and Automation, Raleigh, NC, March-April 1987.

[17] M.Erdmann (1984). *On Motion Planning With Uncertainty.* Tech. Report 810, AI Lab., MIT.

[18] M.Erdmann (1986). *Using Backprojections for the Fine Motion Planning With Uncertainty.* Intern. Journal of Robotics Research (IJRR) 5, 1, Spring 1986.

[19] M.Erdmann, M.T.Mason (1986). *An Exploration of Sensorless Manipulation.* Proc. of the IEEE Intern. Conf. of Robotics and Automation, San Francisco, CA, April 1986.

[20] M.R.Genesereth, J.C.Latombe (1987). *A Proposal for Research on Autonomous Construction Robots.* Logic Group Report Logic-87-14, Dept. of Comp. Sc., Stanford Universi ty, October 1987.

[21] B.Faverjon, P.Tournassoud (1987). *A Local Based Approach for Paih Planning of Manipulators with a High Number of Degrees of Freedom.* Proc. of the IEEE Intern. Conf. on Robotics and Automation, Raleigh, NC, March-April 1987.

[22] R.J.Firby (1987). *A Invesiigation* into *Reactive Planning in Complex Domains.* Proceedings of the 6th National Conference on Artificial Intelligence (AAAI87), Seattle, WA, July 1987.

[23] L.Gouzènes *(1984). Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots.* Intern. Journal of Robotics Research (IJRR) 3, 1.

[24] J.Hopcroft, G.Wilfong (1986). *Moiion of Objects in Contact.* Intern. Journal of Robotics Research (IJRR) 4, 4, Winter 1986.

[25] O.Khatib (1986). *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.* Intern. Journal of Robotics Research (IJRR) 5, 1, Spring 1986.

[26] O.Khatib (1987). *A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation.* IEEE Journal of Robotics and Automation, RA-3, 1, February 1987.

[27] A.Koutsou (1986). *Planning Motion in Contact to Achieve Parts Mating.* Ph.D. Dissertation, Dept. of Comp. SC., University of Edinburgh.

[28] J.C.Latombe *(1984). Automatic Synihesis of Robot Programs from CAD Specifications.* In M.Brady, L.A.Gerhardt, and H.F.Davidson (cds.), *Robotics and Artificial Intelligence.* NATO AS1 Series, Springer-Veriag.

[29] J.C.Latombe et al. (1984). *Deaiyn and Implementaiion of a Robot Programming System Combining Tezfual and Graphical Facilities.* Proc. of the 2nd Intern. Symp. on Robotics Research (IsRR), Kyoto, Japan, August 1984.

[30] C.Laugier, F.Germain (1985). *An Adaptative Collision-Free Trajectory Planner,* Proc. of Intern. Conf. on Advanced Robotics (ICAR), Tokyo, Japan, September 1985.

[31] C.Laugier, P.Théveneau (1986). *Planning* Sensor-Baaed *Mofiona for Part-Mating Using Geometric Reasoning Techniquea.* Proc. of European Conf. on Artificial Intelligence (ECAI), Brighton, UK, July 1986.

[32] L.I.Liebermann, M.A. Wesley *(1977). A UTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly.* IBM Journal of Research and Development, July 1977.

[33] T.Lozano-Pérez (1976). *The* Design *of a Mechanical Assembly System.* S.M. Dissertation, Dept. of Elec. Eng. and. Comp. SC., MIT.

[34] T.Lozano-Pérez (1981). *Automatic Planning of Manipulator Transfer Movements.* IEEE Tr. on Systems, Man and Cybernetics, SMC-11.

[35] T.Lozano-Pérez, M.T.Mason, R.H.Taylor (1984). *Automatic Syniheaia of Fine-Motion Strategies for Robots.* Intern. Journal of Robotics Research (IJRR), 3, 1.

[36] T.Lozano-Pérez *et* al. (1987). *Handey: A Robot System That Recognizes, Plans, and Manipulates.* Proc. of the IEEE Intern. Conf. on Robotics and Automation, Raleigh, NC, March-April 1987.

[37] T.Lozano-Perez (1987). *A Simple Motion-Planning Algorifhm for General Robot Manipulators.* IEEE Journal of Robotics and Automation, RA-3, 3, June 1987.

[38] M.T.Mason (1981). *Compliance and Force Control for Computer Controlled Manipulators.* Proc. of the IEEE Intern. Conf. on Robotics and Automation, San Francisco, CA, April 1981.

[39] M.T.Mason (1984). *Automatic Planning of Fine* Motions: *Correctness and Complete*neaa. Proc. of the IEEE Intern. Conf. on Robotics and Automation, Atlanta, GA.

[40] E.Mazer *(1987). Handey: A Planner for Task-Level Robot Command.* Thesis Dissertation, University of Grenoble, December 1987.

[41] B.K.Natarajan (1986). *The Complexity of Fine Motion Planning.* Tech. Report 86-*734,* Cornell University, Ithaca, NY, February 1986.

[42] N.J.Nilsson (1980). *Principles of Artificial Intelligence.* Morgan Kaufmann, Los Altos, CA.

[43] J.Pertin-Troccaz, P.Puget (1987). *Dealing with Uncertainty in Robot Planning Using Program Prouing Techniques.* Preprints of the 4th Intern. Symp. on Robotics Research (ISRR), Santa-Cruz, CA, August 1987.

[44] M.H.Raibert, J.J.Craig (1981). *Hybrid Position/Force Control of Manipulators.* Journal of Dynamic Systems, Measurement and Control, 102, June 1981.

[45] J.Reif (1979). *Complexity of the Mover's Problem and Generalizations.* Proc. of the 20th IEEE Symposium on FOCS.

[46] S.Shekhar, O.Khatib (198'7). *Force Strategies* in *Real-Time Fine Motion Assembly.* Proceedings of the ASME Winter Annual Meeting, Boston, MA.

[47] J-Schwartz, M.Sharir (1982). *On the Piano Movers' Problem, II. General techniques for Compuiing Topological Properties of Real Algebraic Manifolds.* Tech. Report No. 41, Computer Science Department, Courant Institute of Mathematical Sciences, New York University.

[48] J.Schwartz, J.Hopcroft, M.Sharir (1987). *Planning, Geometry and Complexity of Robot Motion Planning.* Albex Publishing Co., New- Jersey.

[49] R.Smith, M.Self, P.Cheeseman *(1987). Estimating Uncertain Relationships in Robotics.* Intern. Journal of Robotics Research (IJRR), Winter 1987.

[50] R.H.Taylor (1976). *Synthesis of Manipulator Control Programs from Task-Level Specifications.* Ph.D. Dissertation, AIM 228, AI Laboratory, Stanford, July 1976.

[51] R.H.Taylor, M.T.Mason, K.Y.Goldberg (1987). *Sensor- Based Manipulation Planning as a Game with Nature.* Preprints of the 4th Intern. Symp. on Robotics Research (ISRR), Santa Cruz, CA, August 1987.

[52] G.Toussaint, ed. (1985). *Computational Geometry.* Machine Intelligence and Pattern Recognition, Vol. *2,* North-Holland.

[53] J.Valade (1984). *Automatic Generation of Trajectories for Assembly Tasks.* Proc. of the 6th European Conf. on Artificial Intelligence (ECAI), Pisa, Italy, September 1984.

[54] R.Waldinger (1975). *Achieving Several Goals Simultaneously.* In E.Elcock and D.Michie(eds.), *Machine Intelligence 8*, Ellis Horwood, Chichester, UK.

[55] D.E. Whitney (1985). *Historical Perspectives and Slate of the Art tn Robot Force Control.* Proc. of IEEE Intern. Conf. on Robotics and Automation, St. Louis, MO, March . 1985.

Appendix: Table of Symbols

$\mathcal{A}$	Moving object
$\mathcal{B}_i$	Obstacle (i = 1, 2, . ..)
C	Configuration space of $\mathcal{A}$
P	Effector point (mapping of $\mathcal{A}$ in C)
$\mathcal{A}(\mathbf{c})$	Region occupied by $\mathcal{A}$ when **P**'s position in C is c
CB_i	C-obstacle (mapping of $\mathcal{B}_i$ in C)
$\mathcal{C}_{free}$	Free space
$C_{contact}$	Contact Space
$\mathcal{I}$	Intial region of P in a motion planning problem
$\mathcal{G}$	Goal region of P in a motion planning problem
$\mathcal{T}$	Target of a motion command
$\mathbf{c}^*_{init}$	Initial actual position of P before executing a motion plan
$\mathbf{c}^*_s$	Initial actual position of P before executing a motion command
M	Generalized motion command
c s	Control statement
T C	Termination condition
$\mathbf{V}(\mathbf{v})$	Pure velocity control statement
$\mathbf{GD}(\mathbf{v})$	Generalized damper control statement
B	Damper constant (in generalized damper control)
$\mathbf{v}_{net}$	Net velocity (in generalized damper control)
$\mathbf{tp}(\delta t, \mathbf{c}_{[0,\delta t]}, \mathbf{f}_{[0,\delta t]})$	General form of the termination condition
$\mathcal{U}_{CS}(\mathrm{CS})$	Control uncertainty
$\mathcal{U}_C(\mathcal{C})$	Model uncertainty
$\vec{\mathbf{v}}$	Specified commanded velocity
$\mathbf{v}^*$	Actual commanded velocity
$\mathcal{U}_v(\mathbf{v})$	Uncertainty on commanded velocity
c	Measured position of P (configuration of A)
$\mathbf{c}^*$	Actual position of P
$\mathcal{U}_c(\mathbf{c})$	Uncertainty on position sensing
$\mathbf{f}$	Measured reaction force on P
$\mathbf{f}^*$	Actual reaction force on P
$\mathcal{U}_f(\mathbf{f})$	Uncertainty on force sensing

$\nu(\mathbf{c}^*)$	Unit vector pointing along the friction cone at $\mathbf{c}^*$ ($\forall \mathbf{c}^* \in \mathcal{C}_{contact}$)
$\nu(\mathcal{E})$	Unit outgoing normal vector to edge $\mathcal{E}$
$2\phi(\mathbf{c}^*)$	Angle of the friction cone at $\mathbf{c}^*$ ($\forall \mathbf{c}^* \in \mathcal{C}_{contact}$)
$2\phi(\mathcal{E})$	Angle of the friction cone along edge $\mathcal{E}$
θ_v	Uncertainty on the orientation of the commanded velocity
ρ_c	Radius of the position uncertainty disk
ε_f	Uncertainty on the module of the measured force
θ_f	Uncertainty on the orientation of the measured force
τ	Observed trajectory
τ^*	Actual trajectory
$\mathcal{D}^*(\mathcal{S}, \text{CS})$	Directory of actual trajectories
δt	Elapsed time since the beginning of the execution of a motion
$\mathbf{c}_\tau$	Function mapping δt into the measured position along trajectory τ
$\mathbf{f}_\tau$	Function mapping δt into the measured force along trajectory τ
$\mathbf{c}^*_{\tau^*}$	Function mapping δt into the actual position along trajectory τ^*
$\mathbf{f}^*_{\tau^*}$	Function mapping St into the actual force along trajectory τ^*
$\mathcal{K}_{traj}(\tau^*)$	Set of observed trajectories consistent with actual trajectories τ^*
$\Pi_{\mathcal{P}}(\mathcal{T}, \mathbf{M}^+_{\mathcal{P}})$	Preimage of $\mathcal{T}$ relative to $\mathcal{P}$
$\mathcal{P}^{max}(\mathcal{T}, \mathbf{M})$	Maximal preimage of 7 for M
$\mathcal{F}^*(\mathbf{c}^*)$	Range of reaction forces that can be generated at position $\mathbf{c}^*$
$\mathcal{F}^*_{\text{CS}}(\mathbf{c}^*)$	Range of forces that can be generated at position $\mathbf{c}^*$ under CS
$\mathbf{f}^*_{react}(\mathbf{c}^*, \mathbf{f}^*_{appl})$	Reaction force to $\mathbf{f}^*_{appl}$ at position $\mathbf{c}^*$
$\mathcal{X}_{\text{CS}}(\mathcal{S})$	Kernel of region S for CS
$\mathcal{K}^*_{pos}(\mathbf{c}, \mathbf{f})$	Set of actual positions of P consistent with measurements c and f
$\mathcal{B}^{max}(\mathcal{T}, \text{CS})$	Maximal backprojection from 7 for CS
$distance(c_1, c_2)$	Euclidean distance between two points c_1 and c_2
$angle(v_1, v_2)$	Un-signed angle between two vectors v_1 and v_2
$\|\|v\|\|$	Module of vector v
$\Sigma_{\rho_c}(\mathbf{c})$	Position uncertainty disk centered at c
$\oplus$	Minkowski's operator for set addition