A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty*

Bruce Donald

88-895
January 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This paper was presented at the International Workshop on Geometric Reasoning in Oxford, England, July 3, 1986. It will also appear in *Artificial Intelligence*.

Presented at the
International Workshop on Geometric Reasoning
Oxford University, England
July 3, 1986

A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty

Bruce R. Donald
Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract: Robots must plan and execute tasks in the presence of uncertainty. Uncertainty arises from sensing errors, control errors, and uncertainty in the geometric models of the environment and of the robot. The last, which we will call *model uncertainty*, has received little previous attention. In this paper we present a formal framework for computing motion strategies which are guaranteed to succeed in the presence of all three kinds of uncertainty. We show that it is effectively computable for some simple cases. The motion strategies we consider include sensor-based gross motions, compliant motions, and simple pushing motions.

We show that model uncertainty can be represented by position uncertainty in a generalized configuration space. We describe the structure of this space, and how motion strategies may be planned in it.

It is not always possible to find plans that are guaranteed to succeed. In the presence of model error, such plans may not even exist. For this reason we investigate *Error Detection and Recovery (EDR) strategies.* We characterize what such strategies are, and propose a formal framework for constructing them. Our theory represents what is perhaps the first systematic attack on the problem of error detection and recovery based on geometric and physical reasoning.

To appear in *Artificial Intelligence.*

Acknowledgements. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Office of Naval Research under Office of Naval Research contract N00014–81–K–0494 and in part by the Advanced Research Projects Agency under Office of Naval Research contracts N00014–80–C–0505 and N00014–82–K–0334. The author was funded in part by a NASA fellowship administered by the Jet Propulsion Laboratory.

PART I: INFORMAL DEVELOPMENT

1. Introduction

Robots must plan and execute tasks in the presence of uncertainty. Uncertainty arises from sensing errors, control errors, and uncertainty in the geometric models of the environment and of the robot. The last, which we will call *model uncertainty*, has received little previous attention. In this paper we present a formal framework for computing motion strategies which are guaranteed to succeed in the presence of all three kinds of uncertainty. We show that it is effectively computable for some simple cases. The motion strategies we consider include sensor-based gross motions, compliant motions, and simple pushing motions.

We show that model uncertainty can be represented by position uncertainty in a generalized configuration space. We describe the structure of this space, and how motion strategies may be planned in it.

It is not always possible to find plans that are guaranteed to succeed. In the presence of model error, such plans may not even exist. For this reason we investigate *Error Detection and Recovery (EDR) strategies.* We characterize what such strategies are, and propose a formal framework for constructing them.

This paper offers two contributions to the theory of manipulation. The first is a framework for planning motion strategies with model error. Model error is a fundamental problem in robotics, and we have tried to provide a principled, theoretical approach. Our framework can be described very compactly, although many algorithmic and implementational questions remain.

The second contribution is a formal, geometric approach to EDR. While EDR is largely motivated by the problems of uncertainty and model error, its applicability may be quite broad. EDR has been a persistent but ill-defined theme in both AI and robotics research. Typically, it is viewed as a kind of source-to-source transformation on robot programs: for example, as a method for robustifying them by introducing sensing steps and conditionals. We have taken the view that if one can actually plan to sense an anomalous event, and to recover from it, then it is not an error at all. When such plans can be guaranteed, they fall out of a strategy such as LMT [21]. In our view of EDR, an "error" occurs when the goal cannot be recognizably achieved given the resources of the executive and the state of the world. The EDR framework fills a gap when guaranteed plans cannot be found or do not exist: it provides a technology for constructing plans that might work, but fail in a "reasonable" way when they cannot. Our theory represents what is perhaps the first systematic attack on the problem of error detection and recovery based on geometric and physical reasoning.

The first part of this paper develops the framework informally. A more detailed development may be found in part two. At the end of part one, we describe an implementation in a restricted domain.

1.1. Application and Motivation

A Simple Example

Consider fig. 1, which depicts a peg in hole insertion task. One could imagine a manipulation strategy derived as follows: The initial plan is to move the peg straight down towards the bottom of the hole. However, due to uncertainty in the initial position of the peg, the insertion may fail because the peg contacts to the left or right of the hole. Either event might be regarded as an "error." The "recovery" action is to move to the right (if the peg contacted to the left) and to move to the left (if the peg contacted to the right). Thus a plan can be obtained by introducing sensing steps and conditional branches.

Suppose that this conditional plan can be guaranteed—that is, it is a complete manipulation strategy for this simple task. In this case, it seems strange to view the contact conditions as "errors." We do not regard these events as "errors." Our reasoning is that if they can be detected and planned for, then they are simply events in a guaranteed plan.

We are interested in a different class of "errors." Now suppose that there is uncertainty in the width of the hole. If the hole is too small, we will consider this an error, since it causes all plans to fail. Similarly, if some object blocks the hole, and cannot be pushed aside, this is also an error, since makes the goal unreachable. If either error is possible, there exists no guaranteed plan, for there is no assurance that the task can be accomplished. Since no guaranteed plan can be found, we are left with the choice of giving up, or of considering a broader class of manipulation strategies: plans that might work, but fail in an "reasonable" way when they cannot. Specifically, we propose that EDR strategies should achieve the goal when it exists and is recognizably reachable, and should signal failure when it is not.

Application

The theory developed here is quite general. However, it may help to keep in mind a particular application task. We will later return to this task in describing an implementation and experiments.

An interesting application domain for EDR is gear meshing. Let us consider a simplified instance of this problem. In fig. 2 there are two planar gear-like objects, A and B. The task is to plan a manipulation strategy which will mesh the gears. The state in which the gears are meshed is called the *goal*.

We will consider two variants of this problem. In the first, we assume that the manipulator has grasped A, and that neither A nor B can rotate. However, A can slide along the surfaces of B. In the second, B is free to rotate about its center, but this rotation can only be effected by pushing it with A. In both cases, the initial orientation of B is unknown. We regard A as the moving object and B as the environment; hence the uncertainty in B's orientation is a form of model uncertainty.

In the first case, the system has only two degrees of motion freedom. In the second, there are three degrees of motion freedom, one of which is rotational, since B can be pushed.

In both variations, there is uncertainty in control, so when a motion direction is commanded, the actual trajectory followed is only approximately in that direction. There is also uncertainty in position sensing and force sensing, so that the true position and reaction forces are only known approximately. The magnitude of these uncertainties are represented by error balls.

In general, a commanded motion of A may cause A to move through free space, and contact B, possibly causing B to rotate. Our EDR theory is a technique for analyzing these outcomes geometrically to generate strategies that achieve the goal when it is recognizably reachable, and signal failure when it is not. After developing the EDR theory in general, we will return and apply it to the gear example.

1.2. Examples of Model Error

We will begin developing the EDR theory by examining some very simple planning problems with model error.

Example (1). Consider fig. 3. There is position sensing uncertainty, so that the start position of the robot is only known to lie within some ball in the plane. The goal is to bring the robot in contact with the right vertical surface of A.

We will simplify the problem so that the computational task is in configuration space [2,7,9,10,19,20]. This transformation reduces the planning task for a complicated moving object to navigating a point in configuration space. Consider fig. 4. The configuration point starts out in the region R, which is the position sensing uncertainty ball about some initial sensed position. To model sliding behavior, we will assume coulomb friction and generalized damper dynamics, which allows an identification of forces and velocities. Thus the actual commanded velocity v_0 is related to the effective velocity v by $f = B(v - v_0)$ where f is the commanded force and B is a scalar. Given a nominal commanded velocity v_0^*, the control uncertainty is represented by a cone of velocities (B_{ec} in the figure). The actual commanded velocity v_0 must lie within this cone.

The goal in fig. 4 is to move to the region G. Now, with coloumb friction, sticking occurs on a surface iff the (actual) commanded velocity points into the friction cone. We assume the friction cones are such that sliding occurs (for all possible commanded velocities in B_{ec}) on all surfaces save G, where all velocities stick. We will assume that the planner can monitor position and velocity sensors to determine whether a motion has reached the goal. Velocity sensing is also subject to uncertainty: for an actual velocity v, the sensed velocity lies in some cone B_{ev} of velocities about v.

Now we introduce simple model error. The shape of A and B are known precisely, and the position of A is fixed. However, the position of B, relative to A is not known. B's position is characterized by the distance α. If $\alpha > 0$ the goal is reachable. But if $\alpha = 0$, then the goal vanishes. No plan can be guaranteed to succeed if $\alpha = 0$ is possible. Suppose we allow α to be negative. In this case the blocks meet and fuse. Eventually, for sufficiently small α, B will emerge on the other

side of A. In this case, the goal "reappears," and may be reachable again. Let us assume that α is bounded, and lies in the interval $[-d_0, d_0]$.

Our task is to find a plan that can attain G in the cases where it is recognizably reachable. Such a plan is called a *guaranteed strategy in the presence of model error.* But the plan cannot be guaranteed for the α where the goal vanishes. In these cases we want the plan to signal failure. Loosely speaking, a motion strategy which achieves the goal when it is recognizably reachable and signals failure when it is not is called an *Error Detection and Recovery (EDR) strategy.* Such strategies are more general than guaranteed strategies, in that they allow plans to fail.

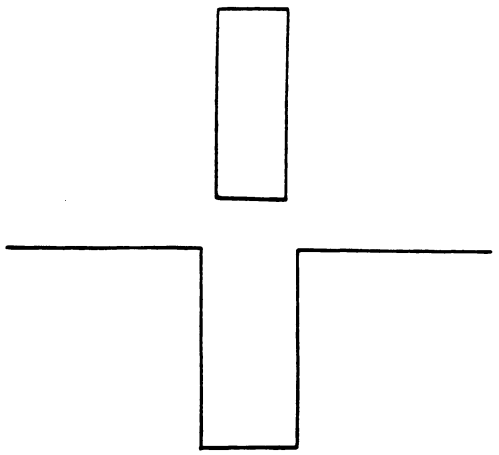

Fig. 1. The goal is to insert the peg in the hole. No rotation of the peg is allowed. One can imagine a strategy which attempts to move straight down, but detects contact on the top surfaces of the hole if they occur. If the peg sticks on the top surfaces, the manipulator tries to move to the left or right to achieve the hole. Are these contact conditions "errors"? We maintain that they are not, since they can be planned for and verified.

Example (2). Before we attack the problem of constructing guaranteed strategies and EDR strategies (both in the presence of model error), let us look at a second example. Fig. 5 illustrates the peg-in-hole (or more precisely, point-in-hole) problem with model error. Here, there are five, interacting types of model error. Our framework will also cover this case. Although in these examples model error has been represented by a kind of "tolerancing", the framework can represent arbitrary model error. For example, we could represent CAD surfaces with real coefficients, and allow the coefficients to vary. Discrete and discontinuous model error may also be represented. Finally, note that we permit gross topological changes in the environment—for example, the goal can vanish.

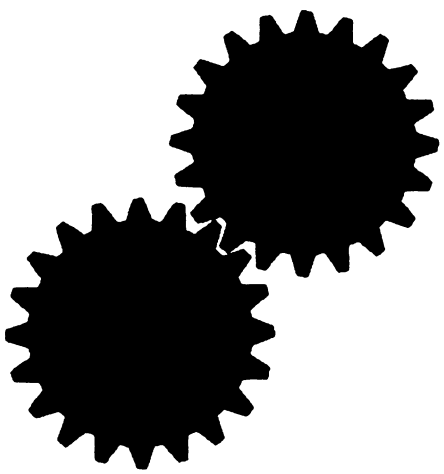

Fig. 2. Geometric models of two gear-like planar objects A and B. A is grasped and can translate but not rotate. B can rotate about its center if pushed. The orientation of B is unknown. The task is to generate a motion strategy to mesh the gears.

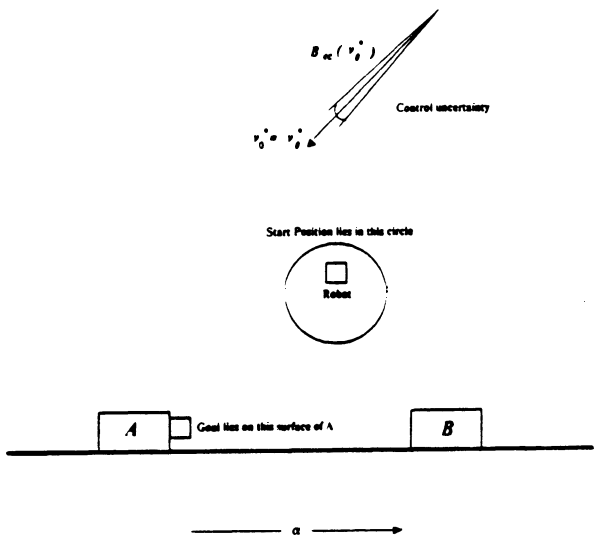

Fig. 3. The goal is to bring the robot into contact with the right vertical surface of A. (For example, the "robot" could be a gripper finger). There is position sensing uncertainty, so in the start position the robot is only known to lie within some uncertainty ball. There is also control uncertainty in the commanded velocity to the robot. It is represented as a cone, as shown.

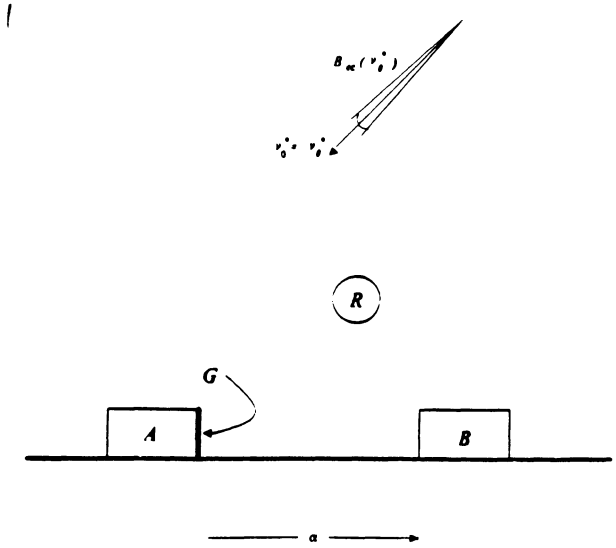

Fig. 4. The equivalent problem in configuration space. The blocks A and B, the distance between the blocks α, and the commanded velocity $v_\theta = v_0^*$ with control error cone $B_{ec}(v_0^*)$. The position of A is fixed.

1.3. Review of Previous Work

For previous work on configuration space, see [1,2,19,20,21,22,29,9,7,10]. For previous work on motion planning with uncertainty, see [32,24,25,4,21,13,14,5,6]. We will exploit the LMT notion of *preimages* [21,13,14] to construct guaranteed strategies. There is little previous work on planning with model uncertainty. However, see [28], [30], [6], and [3]. There has been almost no formal analysis of the EDR problem. [4,31,16,33,18,17,26,34] consider the problem, and the classical planning literature [8] is also relevant. This paper is an expansion of [11], based on [12].

1.4. Representing Model Error

To represent model error, we will choose a parameterization of the possible variation in the environment. The degrees of freedom of this parameterization are considered as additional degrees of freedom in the system. For example, in fig. 4, we have the x and y degrees of freedom of the configuration space. In addition, we have the model error parameter α. A coordinate in this space has the form (x, y, α). The space itself is the cartesian product $\Re^2 \times [-d_0, d_0]$. Each *α-slice* of the space for a particular α is a configuration space with the obstacles A and B instantiated at distance α apart. Fig. 4 is such a slice.

More generally, suppose we have a configuration space C for the degrees of freedom of the moving object. Let J be an arbitrary index set which parameterizes the model error. (Above, J was $[-d_0, d_0]$). Then the *generalized configuration space* with model error is $C \times J$. One way to think

of this construction is to imagine a collection of possible "universes", $\{C_\alpha\}$ for α in J. Each C_α is a configuration space, containing configuration space obstacles. The ambient space for each C_α is some canonical C. $C \times J$ is simply the natural product representing the ambient space of their disjoint union. There is no constraint that J be finite or even countable. In fig. 5, C is again the cartesian plane, and J is a five-dimensional product space. One of the J dimensions is circular, to parameterize the angular variation represented by α_2.

In fig. 6 we show the generalized configuration space for example (1). Note that the goal in generalized configuration space becomes a 2-dimensional surface, and the obstacles are 3-dimensional polyhedra.

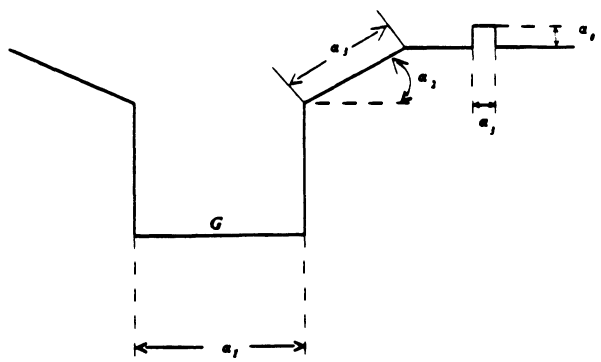

Fig. 5. A peg-in-hole environment with model error. The width of the hole (α_1), angle of chamfer (α_2), length of chamfer (α_3), and height and width of bump (α_4 and α_5) are the model parameters. The hole is allowed to close up.

Given a configuration space corresponding to a physical situation, it is well known how to represent motions, forces, velocities, and so forth in it (eg., see [2]). The representations for classical mechanics exploit the geometry of differentiable manifolds. We must develop a similar representation to plan motions, forces, and velocities in generalized configuration space . Henceforth, we will denote the generalized configuration space $C \times J$ by $\mathcal{G}$. We develop the following "axioms" for "physics" in $\mathcal{G}$.

(1) At execution time, the robot finds itself in a particular slice of $\mathcal{G}$, (although it may not know which). Thus we say there is only one "real" universe, α_0 in[1] J. This α_0 is fixed. However, α_0 is not known *a priori*. Thus all motions are confined to a particular (unknown) α_0-slice, such as fig. 4. This is because motions cannot move between universes. In fig. 6, any legal motion in $\mathcal{G}$ is everywhere orthogonal to the J-axis and parallel to the x-y plane.

[1] α_0 is a multi-dimensional point of J.

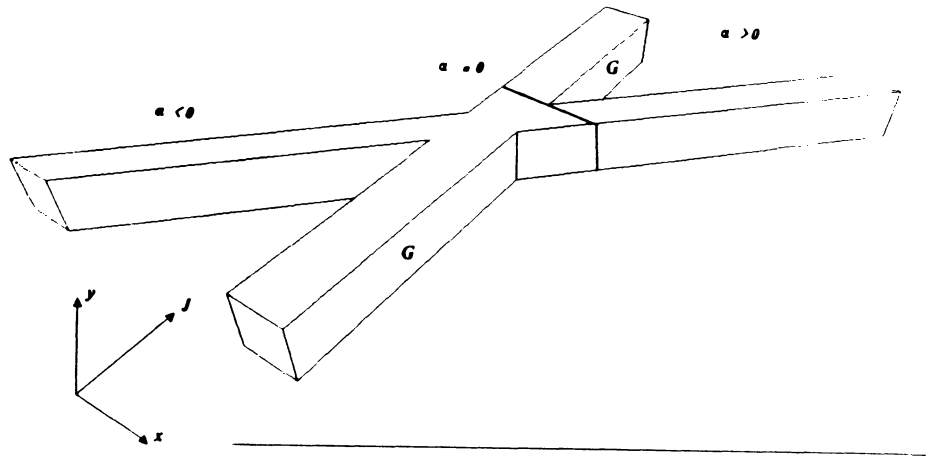

Fig. 6. The generalized configuration space obstacles for example (1). The generalized configuration space is three dimensional, having x and y degrees of motion freedom, and an α degree of model error freedom. Legal motions are parallel to the x-y plane, and orthogonal to the J axis.

(2) Suppose in any α-slice the position sensing uncertainty ball about a given sensed position is some set B_{ep}. The set R in fig. 4 is such a ball. We cannot sense across J: position sensing uncertainty is infinite in the J dimensions.[2] Thus the position sensing uncertainty in $\mathcal{G}$ is the cylinder $B_{ep} \times J$. In figs. 4,6, this simply says that x and y are known to some precision, while α is unknown. The initial position in fig. 4 is given by $R \times [-d_0, d_0]$. This cylinder is a 3-dimensional solid, orthogonal to the x-y plane and parallel to the J-axis in fig. 6.

(3) Suppose in the configuration space C, the velocity control uncertainty about a given nominal commanded velocity is a cone of velocities B_{ec}. Such a cone is shown in fig. 4. This cone lies in the *phase-space* for C, denoted TC. (Phase space is simply Position-space $\times$ Velocity-Space. A point in phase space has the form (x, v), and denotes an instantaneous velocity of v at configuration x). Phase space represents all possible velocities at all points in C. The phase space for $\mathcal{G}$ is obtained by indexing TC by J to obtain $TC \times J$. All velocities in generalized configuration space lie in $TC \times J$. For Ex. (1) $TC \times J$ is $\Re^4 \times [-d_0, d_0]$. The generalized velocity uncertainty cones are two-dimensional, parallel to the x-y plane, and orthogonal to the J axis.

(4) Generalized damper dynamics extend straight fowardly to $\mathcal{G}$, so motions satisfy $f = B(v - v_0)$ where f, v, and v_0 lie in $TC \times J$. Thus friction cones from configuration space (see [13,14]) naturally embed like generalized velocity cones in $TC \times J$.

These axioms give an intuitive description of the physics of $\mathcal{G}$. A formal axiomatization is given

[2] One generalization of the framework would permit and plan for sensing in J. In this case one would employ a bounded sensing uncertainty ball in the J dimensions.

in [12]. We have captured the physics of $\mathcal{G}$ using a set of *generalized uncertainties*, friction, and control characteristics (1-4). These axioms completely characterize the behavior of motions in $\mathcal{G}$.

1.5. Representing Pushing Operations in Generalized Configuration Space

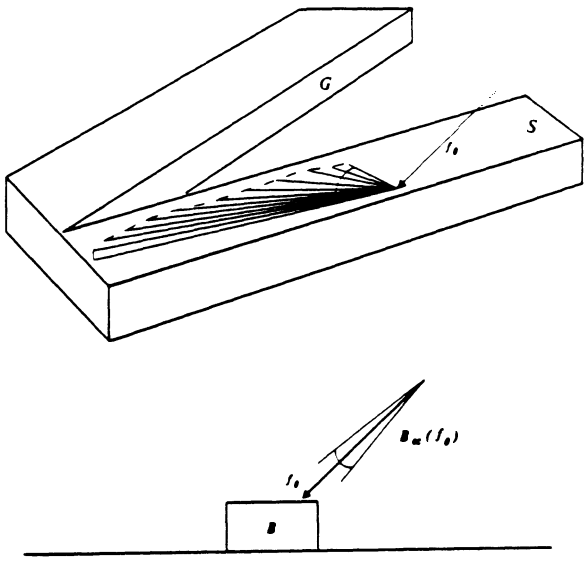

Fig. 7. A force f_θ applied to the top surface of B can cause sliding (or sticking) on the top of B, coupled with motion of B on the table. This corresponds to a pushing motion in $\mathcal{G}$. By giving the right geometric structure to the surface S, we can predict the resulting cone of motions in $\mathcal{G}$, given a commanded velocity f_θ subject to control uncertainty. A planner could generate a motion along S in order to plan pushing operations.

By relaxing axiom (1), above, we can consider a generalization of the model error framework, in which pushing motions are permitted, as well as compliant and gross motions. We relax the assumption that motion between universes is impossible, and permit certain motions across J. Consider example (1). Observe that a displacement in J corresponds to a displacement in the position of the block B. Thus a motion in J should correspond to a motion of B. Suppose the robot can change the position of B by pushing on it, that is, by exerting a force on the surface of B. The key point is that pushing operations may be modeled by observing that commanded forces to the robot may result in changes in the environment. That is, a commanded force to the robot can result in motion in C (sliding) as well as motion in J (pushing the block). Let us develop this notion further.

Our previous discussion assumed that motion across J was impossible. That is, all motion is confined to one α-slice of generalized configuration space. In example (1), this is equivalent to the axiom that B does not move or deform under an applied force. Such an axiom makes sense for applications where B is indeed immovable, for example, if A and B are machined tabs of a connected

metal part. However, suppose that B is a block that can slide on the table. See fig. 7. Then an applied force on the surface of the block can cause the block to slide. This corresponds to motion in J. In general, the effect of an applied force will be a motion which slides or sticks on the surface of B, and which causes B to slide or stick on the table. This corresponds to a coupled motion in both C and J. When the motion maintains contact, it is tangent to a surface S in generalized configuration space.

Our goal is to generalize the description of the physics of $\mathcal{G}$ to permit a rigorous account of such motions. This model can then be employed by an automated planner. *Such a planner could construct motion strategies whose primitives are gross motions, compliant motions, and pushing motions.*[3].

The description of the physics should embrace the following observations:

The phase space for C corresponds to forces exerted at the center of mass of the robot. The phase space for J corresponds to forces acting at the center of mass of B. When pushing is allowed, the phase space for generalized configuration space is not $TC \times J$ but $TC \times TJ$. In the pushing application, all forces are exerted in C, but may be "transferred" to J via the contact. In other words, the applied forces we consider will have zero component along J. However, they may result in a motion in J, via the transferred pushing force.

In free space, or on surfaces generated by immovable objects, all differential motions lie within one α-slice. This is because objects can only be pushed when the robot is in contact with them.

Along surfaces generated by objects that can be pushed, the differential motions are tangent to the surface in $\mathcal{G}$, and may move along J as well as C. See fig. 7.

A motion in free space corresponds to a gross motion. A motion on a surface staying within one α-slice corresponds to a compliant motion. A motion on a surface which moves across J corresponds to a pushing motion.

Configuration space surfaces share many properties with real space surfaces. When pushed on, they push back. In particular, they have a normal. In the absence of friction, they can exert reaction forces only along this normal direction. We must define what the normals to generalized configuration space surfaces are. For example, see fig. 8. The normal is transverse to J, so that even when the applied force lies exclusively in C, the surface exerts a reaction force with a J component. Thus the resultant force can cause a motion across J, tangent to S. In fig. 8 this implies that pushing on the side of B results in a transferred force to J, causing B to slide. In generalized configuration space, this is simply viewed as applying a force to a surface S, which exerts a reaction force across J. Since the resultant force is across J, the motion in $\mathcal{G}$ will be in that direction.

The physics is complicated by the introduction of friction. Given an applied force, one of four qualitative outcomes are possible. (1) The motion may slide in C and J. This corresponds to pushing while sliding[4] at the point of contact. (2) The motion may stick in C and slide in J. This corresponds to pushing with no relative motion. (3) The motion may slide in C and stick in J.

[3]Our model of pushing is less general than [24], since it requires knowledge of the center of friction. See [12] for details

[4]Or rotating.

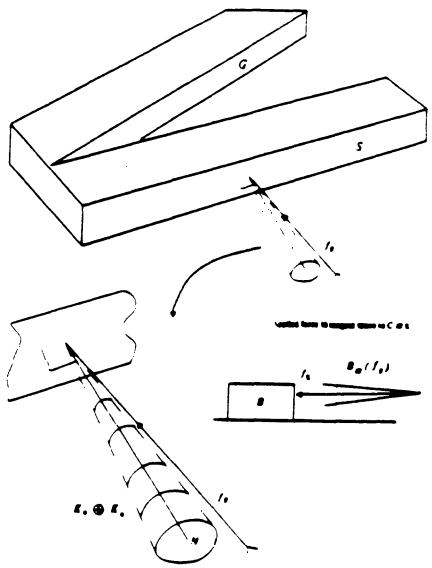

Fig. 8. Pushing on the side of B can cause B to slide, even in the absence of friction. This behavior can be modeled by giving the surface S a normal which points across J. The surface can exert reaction forces along this normal. Thus, applying a force in C results in a reaction force with a J component. The resulting motion moves across J, tangent to S. That it, it pushes the block. Friction can also be introduced on S. A picture of the friction cone developed in [12] is shown. It represents the range of reaction forces the surface S can exert.

This corresponds to compliant motion in one α-slice. (4) The motion may break contact. This corresponds to the initiation of gross motion in one α-slice.

In order to generalize physical reasoning to generalized configuration space, we must provide a generalization of the Erdmannian configuration space friction cone [13,14] for generalized configuration space. The friction cone represents the range of reaction forces that a surface in generalized configuration space can exert. A picture of this generalized cone is shown in fig. 8. Using the friction cone, it is possible to specify a geometrical computation of reaction forces. Such an algorithm is necessary for a planner to predict the possible resulting motions from an uncertainty cone of commanded applied forces. For example, see fig. 7.

By characterizing the physics of pushing and sliding via geometrical constraints in generalized configuration space, it appears that a unified planning framework for gross-, compliant-, and pushing motions emerges. However, certain aspects of the physics require elaboration and simplification before a practical planner for pushing operations can be implemented; see [12] for details.

1.6. Guaranteed Plans in Generalized Configuration Space

A motion strategy [21] is a commanded velocity (such as v_0^* in fig. 4) together with a *termination predicate* which monitors the sensors and decides when the motion has achieved the goal. Given

a goal G in configuration space, we can form its *preimage* [21]. The preimage of G is the region in configuration space from which all motions are guaranteed to move into G in such a way that the entry is recognizable. That is, the preimage is the set of all positions from which all possible trajectories consistent with the control uncertainty are guaranteed to reach G recognizably. For example, see fig. 9. The entry is recognized by monitoring the position and velocity sensors until the goal is attained. Fig. 9 is a *directional* preimage: only one commanded velocity v_θ^* is considered. Here all preimage points reach the goal recognizably under this particular v_θ^*. The *non-directional* preimage is the union of all directional preimages.

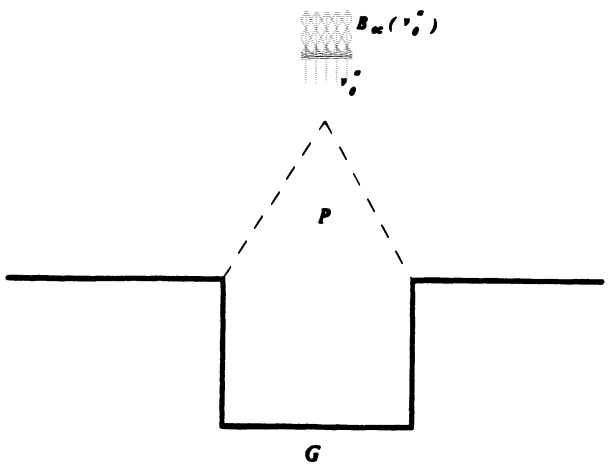

Fig. 9. The goal is the region G. Sliding occurs on vertical surfaces, and sticking on horizontal ones. The commanded velocity is v_θ^*, and the control uncertainty is $B_{ec}(v_\theta^*)$. The *preimage* of the G with respect to θ is the region P.

We envision a back-chaining planner which recursively computes pre-images of a goal region. Successive subgoals are attained by motion strategies. Each motion terminates when all sensor interpretations indicate that the robot must be within the subgoal. [21,13,14] provide a formal framework (LMT) for computing preimages where there is sensing and control uncertainty, but no model error. In particular, [13,14] shows how *backprojections* may be used to approximate preimages. The backprojection of a goal G (with respect to a commanded velocity v_θ^*) consists of those positions guaranteed to enter the goal (under v_θ^*). Recognizability of the entry plays no role. Fig. 10 illustrates the difference between backprojections and preimages. Here the radius of position sensing uncertainty is greater than twice the diameter of the hole. Sliding occurs on all surfaces. The back projection $B_\theta(G)$ strictly contains the preimage $P_\theta(G)$: while all points in the backprojection are guaranteed to reach G, the sensing inaccuracy is so large that the termination predicate cannot tell

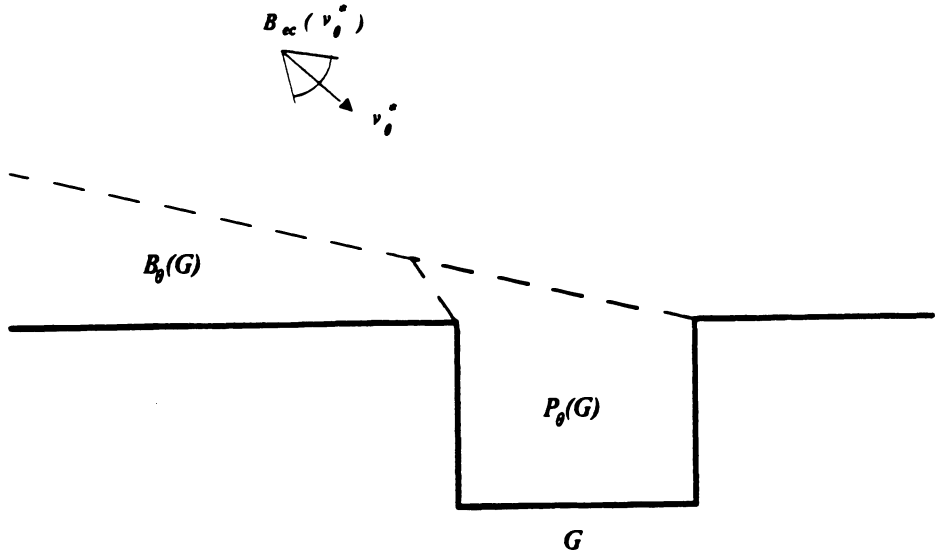

Fig. 10. Here, the radius of the position sensing uncertainty ball is twice the width of the hole. Sliding occurs on all surfaces under the control velocities shown. The preimage of the goal under commanded velocity v_θ^* is $P_\theta(G)$. The backprojection $B_\theta(G)$ strictly contains this preimage: while all points in the backprojection are guaranteed to reach G, the sensing inaccuracy is so large that the termination predicate cannot tell whether the goal or the left horizontal surface has been reached. Only from the preimage can entry into G be recognized.

whether the goal or the left horizontal surface has been reached. Only from the preimage can entry into G be recognized.

Preimages provide a way to construct guaranteed plans for the situation with no model error. Can preimages and backprojections be generalized to situations with model error? The answer is yes. Consider fig. 4,6. The goal in generalized configuration space is the surface G (which has two components). The start region is the cylinder $R \times J$ (where J is $[-d_0, d_0]$). The generalized control and sensing uncertainties in $\mathcal{G}$ are given by the physics axioms above. These uncertainties completely determine how motions in generalized configuration space must behave. We form the backprojection of G under these uncertainties. The backprojection has two components, shown in fig. 11,12. It is a three-dimensional region in $\mathcal{G}$ of all triples (x, y, α) that are guaranteed to reach G under the control uncertainty shown in fig. 4. Equivalently, we can view it as all points in $\mathcal{G}$ guaranteed to reach G under the generalized uncertainties that specify $\mathcal{G}$'s physics. Note that backprojections do not "converge to a point" along the J axis (compare fig. 9). This is because there is perfect control along J, and the commanded velocity along J is zero. This is why in this particular $\mathcal{G}$ there are two disjoint backprojection regions, one from each component of G. Furthermore, recursively-computed backprojections can *never* cover any slice of $\mathcal{G}$ in which the goal vanishes.

The trick here was to view the motion planning problem with n degrees of motion freedom and k degrees of model error freedom as a planning problem in an $(n + k)$-dimensional generalized

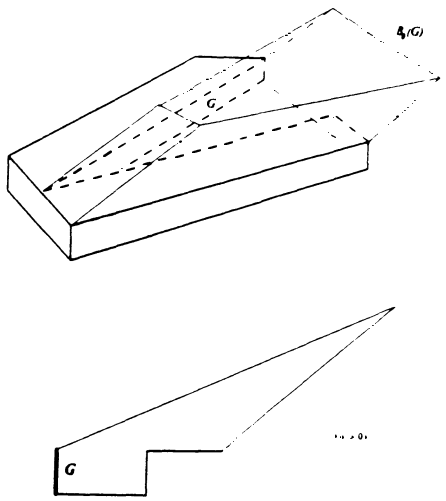

Fig. 11. The backprojection of the goal surface G in generalized configuration space for commanded velocity v_θ^* is denoted $B_\theta(G)$. Here is the backprojection for α positive. A typical α-slice of the backprojection is shown below.

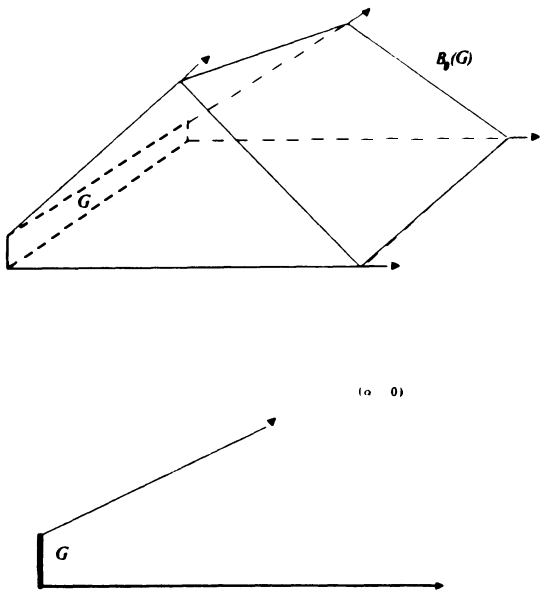

Fig. 12. The backprojection of the other component of G. A typical α-slice for α negative is shown below. The backprojection in $\mathcal{G}$ of the entire goal surface is the union of the backprojections shown in figs. 11,12.

configuration space, endowed with the special physics described above. The physics is characterized precisely by axioms defining certain special sensing and control uncertainties in $\mathcal{G}$. The definitions and results for pre-images and backprojections [21,13,14] in configuration space generalize *mutatis mutandis* to $\mathcal{G}$ endowed with this physics; this is proved in [12]. Thus our framework reduces the problem of constructing guaranteed motion strategies with model error to computing preimages in a somewhat more complicated, and higher-dimensional configuration space.

In this example, because the position of B varies linearly with α, the surfaces in $\mathcal{G}$ are planar and the generalized configuration space obstacles are polyhedral. This means that three-dimensional backprojections of G in $\mathcal{G}$ may be computed in polynomial time [12]. While they have been computed by hand here, note that this reduction gives us an efficient planning algorithm for an important special case.

2. Error Detection and Recovery

If we were exclusively interested in constructing guaranteed motion strategies in the presence of model error, we would be done: having reduced the problem to computing preimages in $\mathcal{G}$, we could now turn to the important and difficult problems of computing and constructing $\mathcal{G}$, and further extend the work of [21,13,14] on computing preimages in general configuration spaces (see [12]).

However, guaranteed strategies do not always exist. In example (1), there is no guaranteed strategy for achieving the goal, since the goal may vanish for some values of α. Because tolerances may cause gross topological changes in configuration space, this problem is particularly prevalent in the presence of model error. In example (2), the goal may also vanish (the hole may close up) for certain regions in J. More generally, there may be values of α for which the goal may still exist, but it may not be reachable. For example, in fig. 5, if we have $\alpha_2 = 180°$ and α_1 positive, then G is non-empty, but also not reachable. Finally, and most generally, there may be values of α for which the goal is reachable but not *recognizably* reachable. In this case we still cannot guarantee plans, since a planner cannot know when they have succeeded.

These problems may occur even in the absence of model error. However, without model error a guaranteed plan is often obtainable by back-chaining and adding more steps to the plan. In the presence of model error this technique frequently fails: in example (1), no chain of recursively-computed preimages can ever cover the start region $R \times J$. The failure is due to the peculiar sensing and control characteristics (1-4) in generalized configuration space .

In response, we will develop Error Detection and Recovery (EDR) strategies.

- An EDR strategy should attain the goal when it is recognizably reachable, and signal failure when it is not.
- It should also permit serendipitous achievement of the goal.
- Furthermore, no motion guaranteed to terminate recognizably in the goal should ever be prematurely terminated as a failure.

• Finally, no motion should be terminated as a failure while there is any chance that it might serendipitously achieve the goal due to fortuitous sensing and control events.

These are called the "EDR Axioms", they will be our guiding principles. Can we construct such strategies? The answer is, basically, yes. Let us construct one for a variant of example (1). We first restrict our attention to the environments where α lies in the interval $[d_1, d_0]$ where d_1 is small and negative.

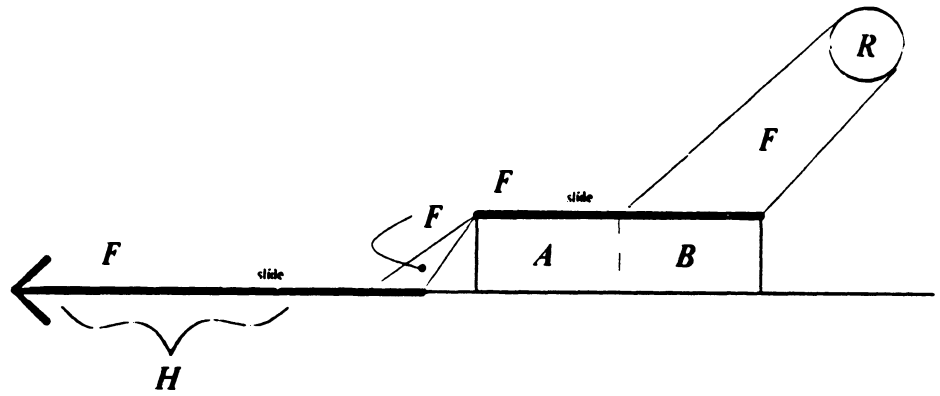

Fig. 13. A typical α-slice of the forward projection of the "bad" region. The forward projection is the region F. α is negative and almost zero. H is an EDR region in the forward projection.

Call the start region $U = R \times J$. The strategy of example (1) commands velocity v_0^* (fig 4). It tries to terminate the motion in G by detecting sticking. Call this strategy θ. We will use θ as a starting point, and try to build an EDR strategy from it. Now, U is divided into a "good" region, from which θ is guaranteed, and a "bad" region, from which it is not. The goal vanishes for the bad region. We wish to *extend* θ to an EDR strategy from all of U.

Let us investigate the result of executing θ from the "bad" region. We employ the forward projection [14]. The *forward projection* of a set V under θ is all configurations[1] which are possibly reachable from V under v_0^* (subject to control uncertainty). It is denoted $F_\theta(V)$. Forward projections only address reachability: the termination predicate is ignored and only the commanded velocity v_0^* is needed to specify the forward projection.

Fig. 13 shows a typical α-slice of the forward projection of the "bad" region. We can now define an EDR strategy as follows. Consider the region H in fig. 13. The termination predicate can distinguish between G and H based on position sensing, velocity sensing, or elapsed time.[2]

[1] Actually, forward projections are in phase-space, so this is the position component of the forward projection.

[2] Given the sensing uncertainties of example (1).

(Consider H as a two-dimensional region in $\mathcal{C}$; just a slice of it is shown in fig. 13). Thus the motion is guaranteed to terminate recognizably in G iff the motion originated in the "good" region of U. Otherwise the motion terminates recognizably in H. In the first case, the termination predicate signals success, in the latter, failure.

Clearly this EDR strategy satisfies the "EDR axioms" above. The problem of constructing EDR strategies may be attacked as follows: We take a strategy θ as data. Next, an *EDR region* H is found. H is introduced as a "bad goal", and a strategy is found which achieves either G or H (subject to the EDR axioms). Finally, we must not only recognize that G or H has been attained, but also know *which* goal has been reached.

Now, think of θ as indexing the "angular direction" of the commanded velocity. By quantifying over all θ, we can in principle define "non-directional" EDR strategies. This problem is similar to constructing non-directional preimages. For now, we restrict our attention to one-step plans. Later, we consider n-step plans.

2.1. Generalizing the Construction

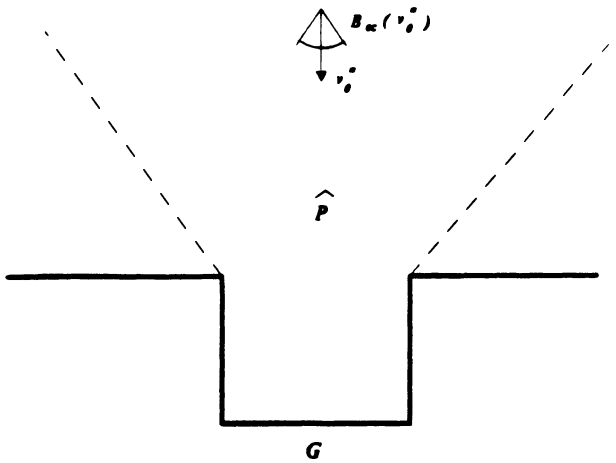

Fig. 14. The *weak* preimage of the goal G under v_θ^*. Compare fig. 9.

We now present an informal account of how the construction of EDR regions and strategies may be generalized. Don't be alarmed if some of our examples are without model error. Since we've reduced the planning problem with model error to planning in a (different) configuration space, it suffices to consider general configuration spaces in this discussion.

So far the preimages we have considered are *strong* preimages, in that *all* possible motions are guaranteed to terminate recognizably in the goal. The *weak* preimage [21] (with respect to a

commanded velocity) is the set of points which could *possibly* enter the goal recognizably, given fortuitous sensing and control events. See fig. 14.

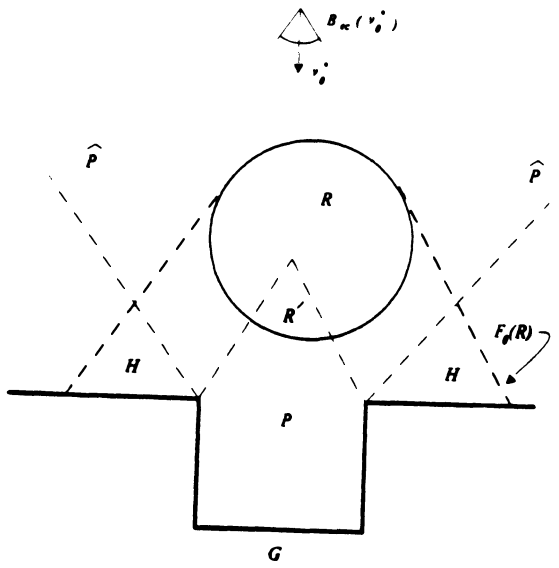

Fig. 15. R is the start region. P is the strong preimage of G. R' is the region in R from which the strategy is guaranteed to reach G recognizably. $\hat{P}$ is the weak preimage. H is the forward projection of R outside the weak preimage. It is the EDR region.

Now consider fig. 15. Sliding occurs on the vertical edges, and sticking on the horizontal ones. The (strong) preimage of the goal G is denoted P. A motion strategy θ with commanded velocity v_θ^* is guaranteed for the region R', but the starting region is the larger[3] R. The weak preimage of G is denoted $\hat{P}$. The forward projection of the "bad" region $R - R'$ is also shown: it is $F_\theta(R - R')$. Using θ as data, how can we construct an EDR strategy that is applicable for all of R? Let us first try taking the EDR region $H = H_0$, where H_0 is the set difference of the forward projection of the "bad" region and the weak preimage:

$$H_0 = F_\theta(R - R') - \hat{P}. \tag{1}$$

If we can distinguish between G and H, then H is a good EDR region, and we have constructed an EDR strategy.

Taking $H = H_0$ as above is not sufficiently general. Consider fig. 16. It is possible for a motion from R to stick forever in the region H_s, which is within the weak preimage. However, a motion through H_s is not guaranteed to stick in H_s: it may eventually slide into the goal. We want sliding motions to pass through H_s unmolested, while the termination predicate should halt sticking motions in H_s.

[3]Note that in general, R and R' need not be cylinders, but can be arbitrary subsets of $\mathcal{C}$.

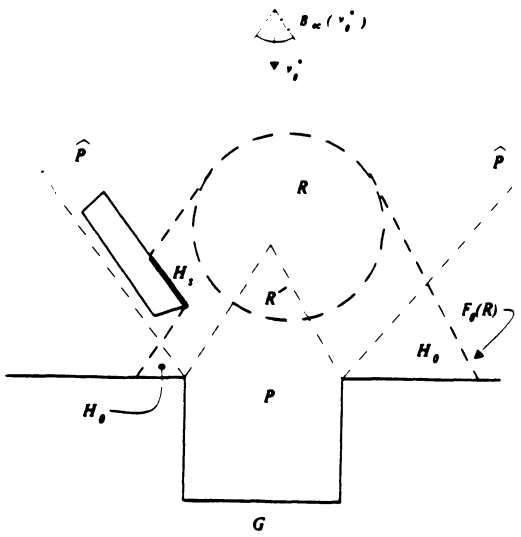

Fig. 16. H_0 in eq. (1) is not the entire EDR region. Sticking may occur within the weak preimage in H_s. The EDR region must include H_0 for all possible velocities, and H_s for "sticking velocities."

The EDR region H region should include H_0. But it should also include H_s, when sticking occurs. In other words, H should include H_0 for *all* velocities, but should only include H_s for *sticking* velocities (that is, zero velocities). To handle this idea we introduce simple velocity goals, as well as position goals. The position and velocity goals are regions in phase space.

A goal in phase space is a region in Position-space × Velocity-space. A phase space goal is attained when the actual position and velocity can be guaranteed to lie in the region. Let us construct the phase-space EDR region $\tilde{H}$. If x is in H_0, then for any velocity v over x, (x, v) must be in $\tilde{H}$. Let $\pi^{-1}(H_0)$ denote all such (x, v) in phase-space.

Now, H_s is the set of all points x in the weak but not strong preimage, such that sticking can occur at x.[4] We wish to distinguish the sticking velocities in H_s. Under generalized damper dynamics, these are essentially the zero velocities. Let $Z(H_s)$ denote the zero velocities over H_s, that is, the set of pairs $(x, 0)$ for x in H_s. This set is in phase space. Then we see that $Z(H_s)$ is also in the phase space EDR region $\tilde{H}$. Thus $\tilde{H}$ is the union of the sticking velocities over H_s, and all velocities over the forward projection outside the weak preimage:

$$\tilde{H} = Z(H_s) \cup \pi^{-1}(H_0). \tag{2}$$

To use $\tilde{H}$ as an EDR region, we must now ensure that $\tilde{H}$ and the cylinder over G are distinguishable goals. In [12], we show that if the strong preimage is known, the definition of (phase space) EDR regions is *constructive up to reachability*. By this we mean that when backprojections,

[4] [14] shows how to compute H_s using configuration space friction cones.

set intersections and differences, and friction cones can be computed, then so can $\tilde{H}$. With $\tilde{H}$ is hand, we add the recognizability constraint to obtain an EDR strategy.

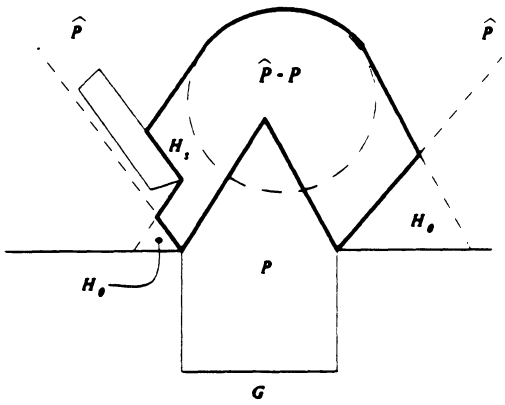

Fig. 17. The weak but not strong preimage $\hat{P} - P$, from fig. 16. Can a motion from R remain in $\hat{P} - P$ forever? One way this may happen is by sticking in H_3. In general, however, there are other ways.

The structure of the "weak but not strong preimage," $\hat{P} - P$ suggests a number of implementation issues. Consider figs. 16,17 once more. Suppose we have a trajectory originating in R, subject to the control uncertainty shown. We do not wish to terminate the motion while it remains in the weak preimage, since fortuitous sensing and control events could still force recognizable termination in G. However, we can terminate the motion as soon as we recognize egress from the weak preimage. This is why the forward projection outside the weak preimage is contained in the EDR region.

As we have seen, however, it is possible for a trajectory to remain within the weak but not strong preimage forever. For example, it can stick in H_3 forever. To handle this case, we introduced phase space EDR goals.

There are other conditions under which a trajectory could stay in $\hat{P} - P$ forever: (a) if the environment is infinite, or $\hat{P} - P$ is unbounded. (b) The trajectory "loops" in $\hat{P} - P$ forever. (a) and (b) are qualitatively different from the case of sticking forever in H_3, because they require motion for infinitely long. In practice this may be handled by terminating the motion in $\hat{P} - P$ after a certain elapsed time. This is called "constructing termination predicates which time-out." In fact, this "solution" works for sticking in H_3 also.

An alternative is to extend our earlier zero-velocity analysis to all of $\hat{P} - P$. That is, we terminate the motion in the weak but not strong preimage when the actual velocity is (close to)

zero. It seems that time-out termination predicates and/or velocity thresholding must be used to solve the looping problem. The issue is subtle and is addressed further in part two.

3. Generalization to n-Step EDR Strategies

3.1. An Example

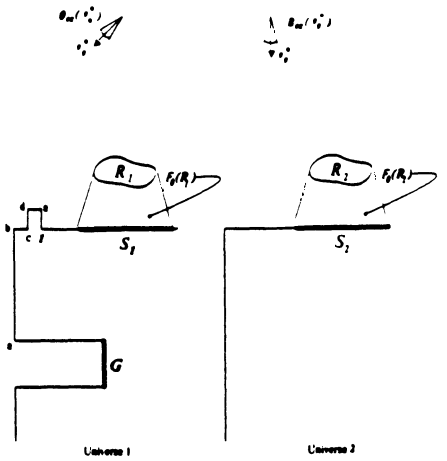

Fig. 18. There are two possible universes; the goal G exists in the first but not the second. The start region is $R_1 \cup R_2$. Motion θ is guaranteed to move from R_1 into S_1. Motion ψ is guaranteed to move from S_1 into f. There is an 8-step plan achieving G from R_1. The forward projections of R_1 and R_2 are indistinguishable. There exists no one-step EDR strategy from the motion θ.

Example (3). So far we have only considered one-step EDR strategies. We now generalize the construction to n-step strategies. Consider fig. 18. Here there are two possible universes, both in the plane, so J is the two element discrete set, $\{1, 2\}$. The start region is the union of R_1 in universe 1, and R_2 in universe 2. The goal exists in universe 1 but not in universe 2. There is no one-step EDR strategy which, from the start region, can guarantee to achieve G or recognize that we are in universe 2. In particular, there is no one-step EDR strategy which can be derived from the motion v_θ^*.

There is an 8-step plan in universe 1 which recognizably achieves G from start region R_1. It is obtained by back-chaining preimages in universe 1. The plan moves from R_1 to the region S_1 under v_θ^*. Then it slides along the top surface to vertex f, and then to the successive vertex subgoals e

through a, and finally into G. We can construct a 2-step EDR strategy, from this plan. First, we execute motion θ from the union of R_1 and R_2. This achieves a motion into S_1 in universe 1, or into S_2 in universe 2. The termination predicate cannot distinguish which has been attained. Suppose the second motion in the 8-step plan is v_ψ^* (see fig. 18), and is guaranteed to achieve the vertex subgoal f from start region S_1. We'll try to construct an EDR strategy out of this second motion. Take as data: the subgoal f, the start region $S_1 \cup S_2$, the horizontal motion ψ, and the preimage of f under ψ. The EDR region for these data is the forward projection of S_2 under ψ (see fig. 19). Presumably this EDR region is distinguishable from f, and so we have constructed an EDR strategy at the second step. After executing the second step, we either terminate the plan as a failure, or proceed to vertex e, and eventually to the goal.

There is a subtle issue of where to terminate the motion within the forward projection of $R_1 \cup R_2$; this "where" is $S_1 \cup S_2$ here, and is called the *push-forward*. Since they address termination, push-forwards are to forward projections as preimages are to backprojections. In part two, they are defined more formally and the n-step EDR construction is given in detail.

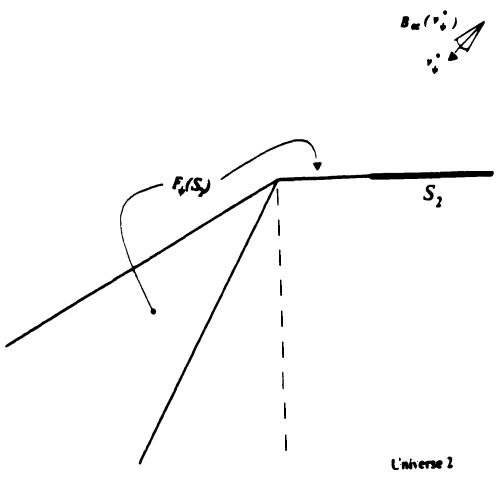

Fig. 19. The forward projection under ψ of S_2.

4. What is "Recovery"?

So far, we have taken a "radical" view with respect to "Recovery." We assume that in planning for error and recovery, one essentially specifies the maximum length plan one is willing to contemplate. The EDR planner considers the class of n-step strategies and tries to formulate a plan which will

achieve the goal given the sensing, control, and model uncertainty.[1] Perhaps such a plan can be guaranteed. If not, then termination in an EDR region would signal failure. This means that there was no guaranteed n-step plan. The recovery action might then be "give up," or "try again, using up the remaining number of steps in the plan," if we are serious in refusing to contemplate plans longer than n steps. As a corollary, the only "Error," then, is "being in the wrong universe," or more accurately, "being in the wrong start region." This viewpoint is a consequence of trying to address EDR and completeness simultaneously. More concretely, suppose we consider some sensory-control-geometric event to be an "error," make a plan to detect it, and a recovery plan in case it is detected. If the plan can be guaranteed, then it can be found using [21]. In this case the "error" is no longer an error, but simply an "event" which triggers a conditional branch of the plan. If the plan cannot be guaranteed, then we have proposed the EDR framework, which allows us to try it anyway. If it fails, however, the only obvious recovery action entails the recursive construction of EDR sub-plans (see below). It is not clear what other kinds of recovery could be attempted without exploiting additional knowledge: the recovery branches have already been tried. As usual the issue is subtle, and deserves further attention.

We give one example which highlights the complexity of the recovery problem. Suppose that we consider the class of 4-step plans. Given a 4-step plan as data, suppose we construct a multi-step EDR strategy which pushes forward on the first motion, and executes an EDR strategy on the the second. After executing the second motion, we have recognizably either achieved the second subgoal, or some EDR region H. If H is achieved, what is the correct recovery action? We could do nothing, and signal failure. Alternatively, we could try to construct a plan (of length less than or equal to two) to achieve the goal. Now, if such a plan exists and can be guaranteed, then the entire EDR analysis was unnecessary, since the LMT framework [21] can (formally) find such plans. However, there might exist a 2-step EDR strategy to (try to) achieve the goal from H. While such a plan could not be guaranteed, it might be worth a try. This suggests that the failure recovery action in an n-step EDR strategy should be to recursively construct *another* EDR strategy to achieve the goal from the EDR region, using no more than the remaining number of steps. If n is 1, the planner should simply signal failure and stop.

5. Implementation and Experiments

In this section we describe experiments with an implemented EDR planner, called `Limited`. We approximate preimages using backprojections (see [13]). `Limited` can compute slice approximations to EDR regions for one-step plans where the generalized configuration space is three-dimensional. The particular generalized configuration space we consider is that of the gear example described in section 1.1. (See fig. 2). In this case, C is the cartesian plane, representing translations of gear

[1] Of course, one could in principle search for strategies of increasing length by quantifying over n. At any one time, however, one would reduce to the case described here and iterate.

A, and J is the 2D rotation group (i.e., a circle), representing orientations of the gear B. The implementation uses slices: by a *slice* we mean an α-slice of generalized configuration space for some α in J. α is the model error parameter, and represents the orientation of B. We have implemented an algorithm which computes slices of the three dimensional EDR regions for both variants of the gear example. In the first, B cannot rotate, so no motion across J is possible. In the second, B can rotate when pushed, so motion across J is possible. In the latter case, backprojections and forward projections must be computed across J, since it is possible to achieve the goal by moving across J (rotating B by pushing and possibly sliding on its surface).

Given a 2D slice of generalized configuration space, `Limited` employs a plane-sweep algorithm for computing unions, intersections, and projections. (By *projections* we mean forward projections, backprojections, and weak-backprojections in that slice). The algorithm uses exact (rational) arithmetic, and computes unions in $O((n+c)\log n)$ time, and projections in $O(n \log n)$ time.[1] The design and implementation of the 2D plane-sweep module is joint work with John Canny; the algorithm is related to [27] (who give a union algorithm) and [13] (who describes an $O(n \log n)$ backprojection algorithm).

To compute projections in the 3D generalized configuration space, `Limited` propagates projections across slices. For example, given a forward projection in a slice, the algorithm finds all obstacle edges and vertices from which it is possible to exert a positive torque on the obstacle (which is gear B in the figures). See fig. 28. Thus by pushing on these edges it is possible to move across slices in the $+\alpha$ direction. Each such edge is a slice of an algebraic ruled surface in generalized configuration space. The vertices are slices of algebraic helicoids. Sliding along the surface of B while causing B to rotate corresponds to following the surface (or helicoid). The surface is traced into the next α-slice, and taken as a start region from which to forward-project in that slice. For example, see figs. 29–30. The propagated forward projection must then be unioned with propagated forward projections from other slices, and with the forward projection of any start regions in that slice. See fig. 33. Weak backprojections are computed analogously.

In order to compute weak backprojections and forward projections, we assume that there can be stiction at the rotation center of B. Thus the ratio of sliding to turning is indeterminate. In general, the computation of strong backprojections under rotation due to pushing will be a second order problem, since it depends on the derivatives of this ratio. We employ a conservative approximation to the strong backprojection (namely, the backprojection in free space alone) to construct the EDR regions. This suffices, since EDR strategies require only the weak backprojection and forward projection (which depend only on the possibility, and not the velocity, of sliding and turning). Thus there is a deep sense in which EDR strategies with model error seem easier to compute than guaranteed strategies, because EDR strategies are "first order." This jibes with the intuition that weak backprojections should be easier to compute than strong backprojections.

Figs. 20–27 show the EDR regions for the gear example when no rotation of B is permitted. Only one slice of $\mathcal{G}$ is shown. In all the figures, the commanded velocity is "towards the center of B", up and to the right.

[1] Where n is the number of vertices in the slice, and c is the number of intersections.

Next, we allow B to rotate. Figs. 31–35 show the EDR regions at four α-slices of the 3D generalized configuration space. In this case motion across J is possible by pushing B, when B rotates. The projections have been propagated across slices and unioned. The results are slices of the 3D EDR regions across J.

The computation of the EDR regions is at the heart of EDR planning. For example, a one-step EDR planner can be implemented as follows. First, we require a module which determines when two regions are distinguishable using sensors and history. This module will be applied to the goal and the EDR regions H to verify the EDR strategy. Such a module could be implemented naively, by "shrinking" the goals and H based on the sensing uncertainties. One can also envision a more complicated algorithm which employs the history available in the forward projection; see [12]. Next, a control structure must be implemented which searches for a commanded velocity yielding an EDR strategy. For example, a generate-and-test technique could compute and verify EDR regions for different commanded velocities. A gross motion planner can be used to generate commanded velocities to try. We have implemented both a one-step and a multi-step EDR planner; see [12]. This is only a first step; much work remains to be done in developing the theory and practice of EDR.

6. Conclusions

This paper offers two contributions to the theory of manipulation. The first is a technique for planning compliant motion strategies in the presence of model error. The second is a precise, geometrical characterization of error detection and recovery (EDR). These led to a constructive definition of EDR plans in the presence of sensing, control, and model error. These more general strategies are applicable in assembly planning where guaranteed plans do not exist, or are difficult to find. We tested the EDR theory by implementing a planner, `Limited` , and running experiments to have `Limited` automatically synthesize EDR strategies.

A number of mathematical tools were developed for the EDR theory. First, we considered compliant motion planning problems with n degrees of motion freedom, and k dimensions of variational geometric model uncertainty. We reduced this planning problem to the problem of computing preimages in an $(n+k)$-dimensional generalized configuration space, which encompasses both the motion and the model degrees of freedom, and encodes the control uncertainty as a kind of non-holonomic constraint. We also showed how pushing motions could be planned using generalized configuration space.

Next, we characterized EDR strategies geometrically via the EDR region H. Determining whether a strategy satisfied the EDR axioms was reduced to a decision problem about forward projections and preimages in generalized configuration space. Making this process formal and algorithmic required a detailed investigation of the geometric and preimage structure of the EDR regions. See the Weak EDR theory of [12] for new mathematical tools for studying multi-step strategies.

In [12], we explored the complexity of EDR planning. We derived bounds both for the implemented planner **Limited** , and for theoretical extensions. While in general it is known that compliant motion planning with uncertainty is intractable, we were able to demonstrate a number of special cases where there exist efficient theoretical algorithms. In particular, we show a case where $n = 2$, $k = 1$ and containment in the backprojection could be computed in polynomial time (note for $n = 3$, $k = 0$, this is false[1]). We also investigate the structure of the non-directional backprojection in the plane. It led to a polynomial-time algorithm for computing one-step (guaranteed) strategies, and a singly-exponential algorithm for multi-step strategies.

Our theory represents what is perhaps the first systematic attack on the problem of error detection and recovery based on geometric and physical reasoning. For more on the structure of constraints in $\mathcal{G}$, a formal description of the EDR construction, push-forwards, and n-step strategies, see part two.

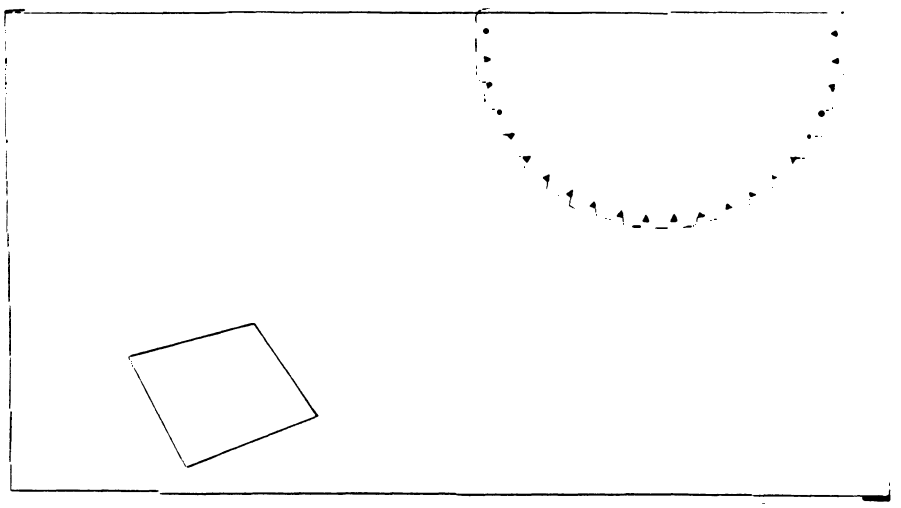

Fig. 20. The configuration space for the gear example (fig. 2) at one α-slice ($\alpha = 0$) of $\mathcal{G}$. The goal region is the "valleys" of the cspace obstacle. The start region is the diamond to the lower left. For figs. 20–27, B is not allowed to rotate, so no motion across J is possible.

PART II

We now undertake a more detailed analysis of the ideas and results described above.

[1][Canny and Reif], Personal communication

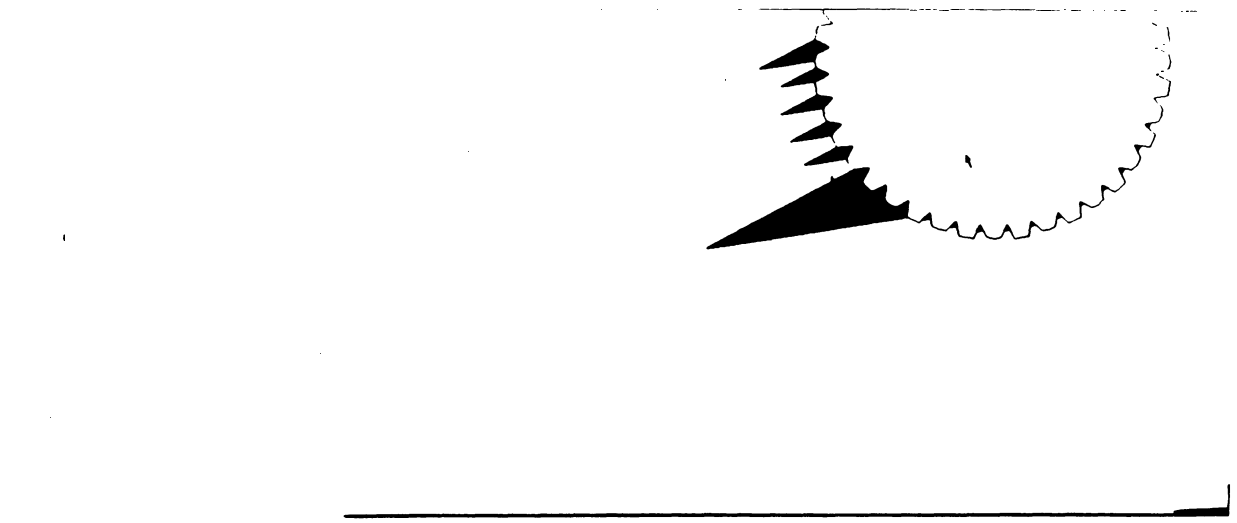

Fig. 21. The strong backprojection in slice $\alpha = 0$ of the goals in fig. 20, assuming that B cannot rotate. In all these experiments, the coefficient of friction is taken to be .25.

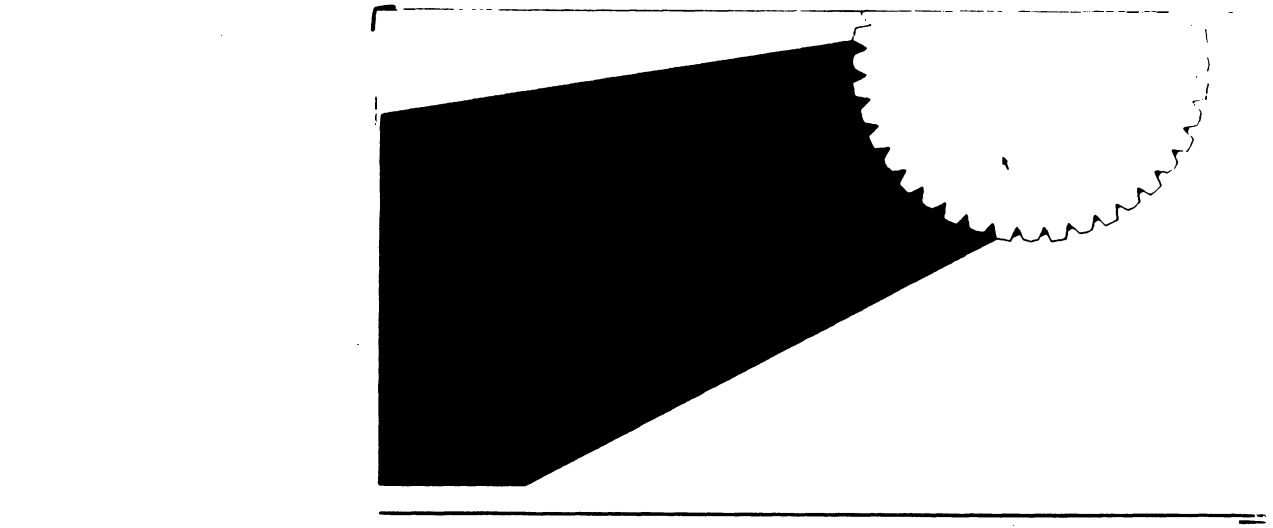

Fig. 22. The weak backprojection of the goals in slice $\alpha = 0$.

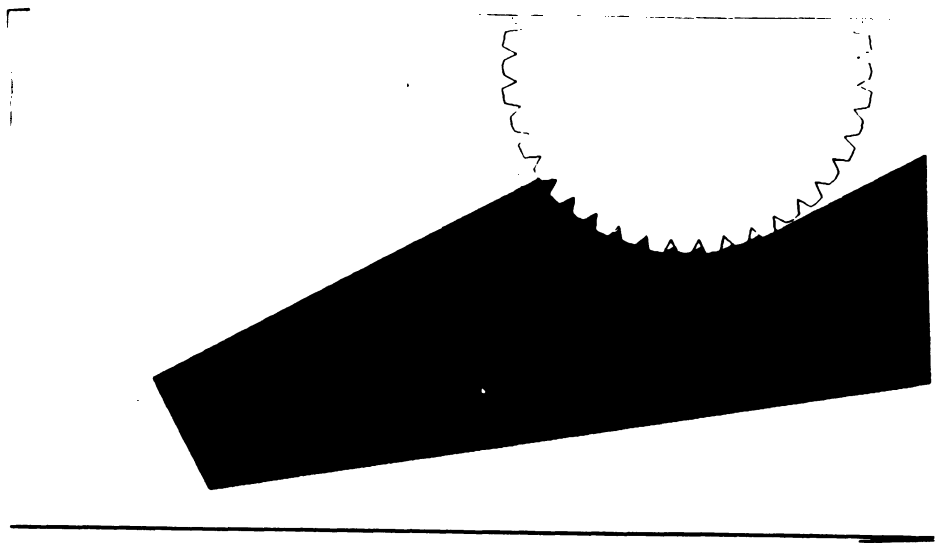

Fig. 23. The forward projection of the start region in slice $\alpha = 0$.

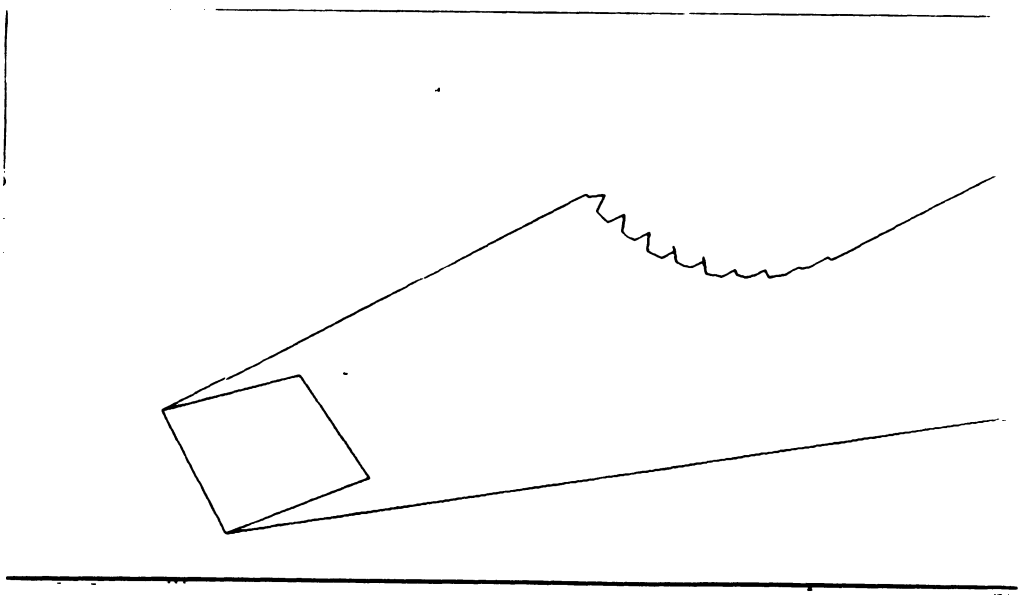

Fig. 24. The forward projection of the start region in slice $\alpha = 0$. Note the degenerate edges due to sliding.

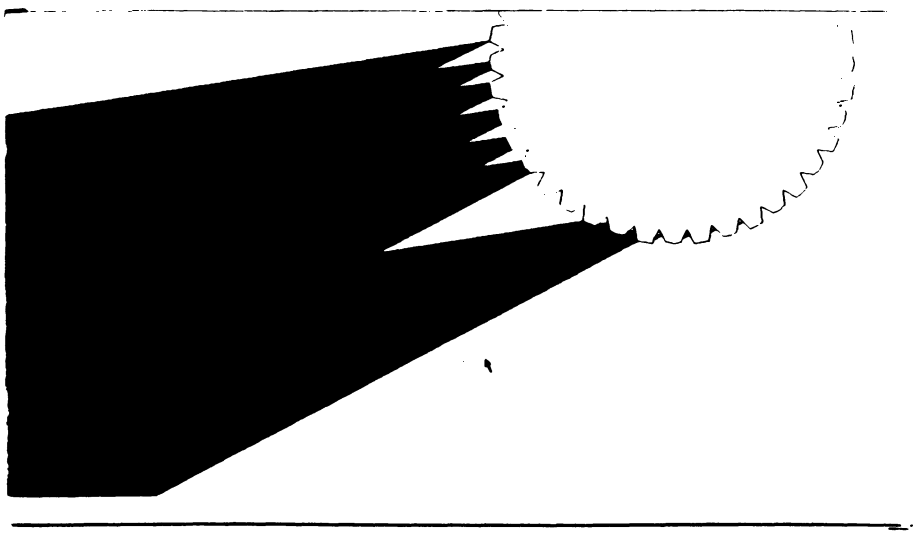

Fig. 25. The weak minus the strong backprojection.

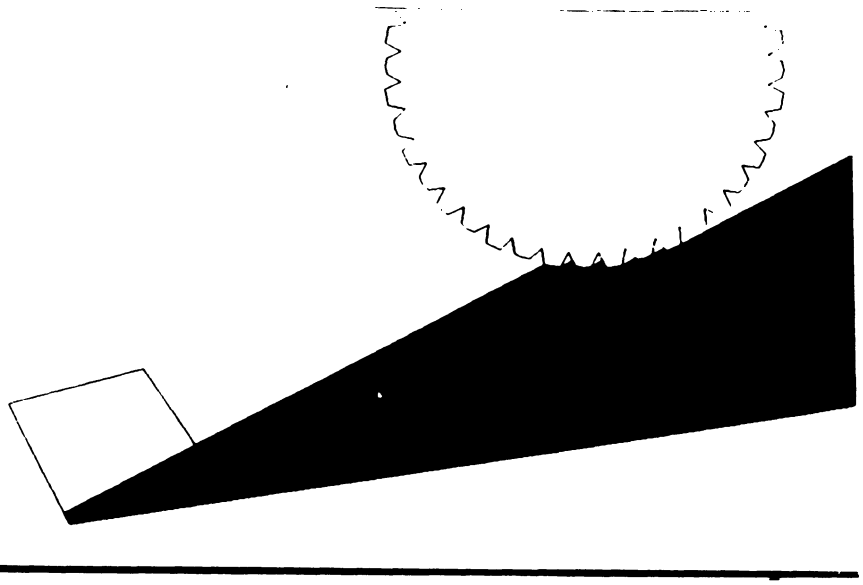

Fig. 26. The H_0 region (the forward projection minus the weak backprojection).

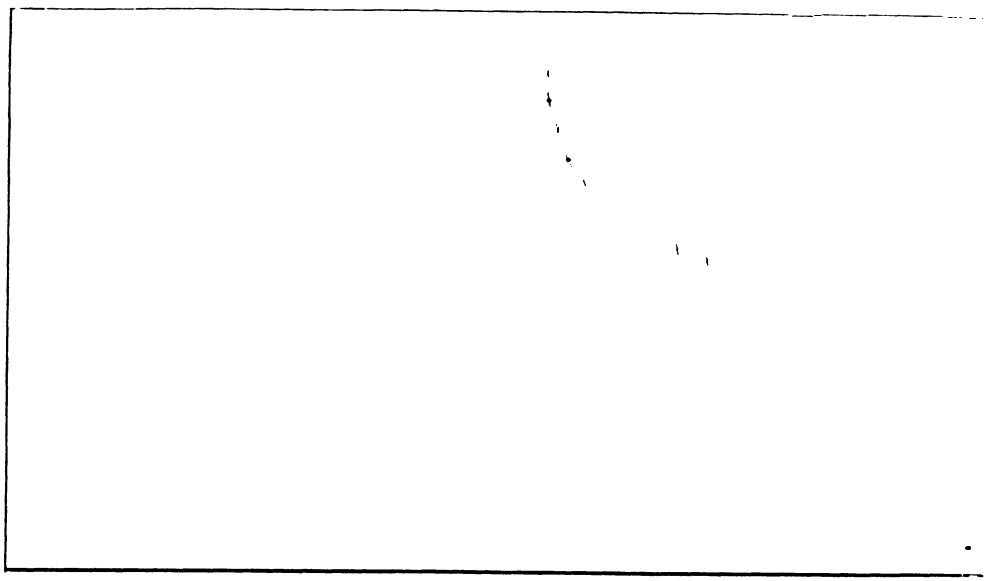

Fig. 27. The H_s region (sticking within the weak but not strong backprojection).

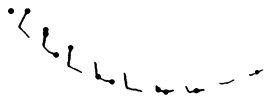

Fig. 28. Now assume that B can rotate when pushed (for figs. 28–35). Here we show the region within the forward projection (fig. 23) from which it it possible to exert positive torque on B. This region is called the *differential forward projection across J in the $+\alpha$ direction.*

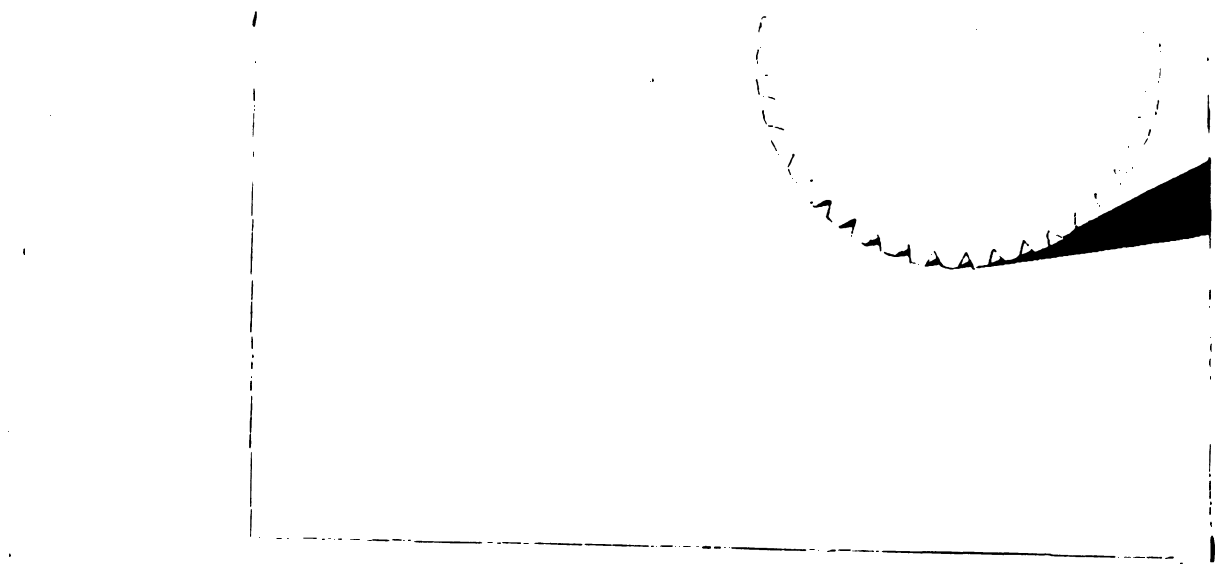

Fig. 29. The differential forward projection is propagated to the next slice in the $+\alpha$ direction. Here we take its forward projection in the next slice.

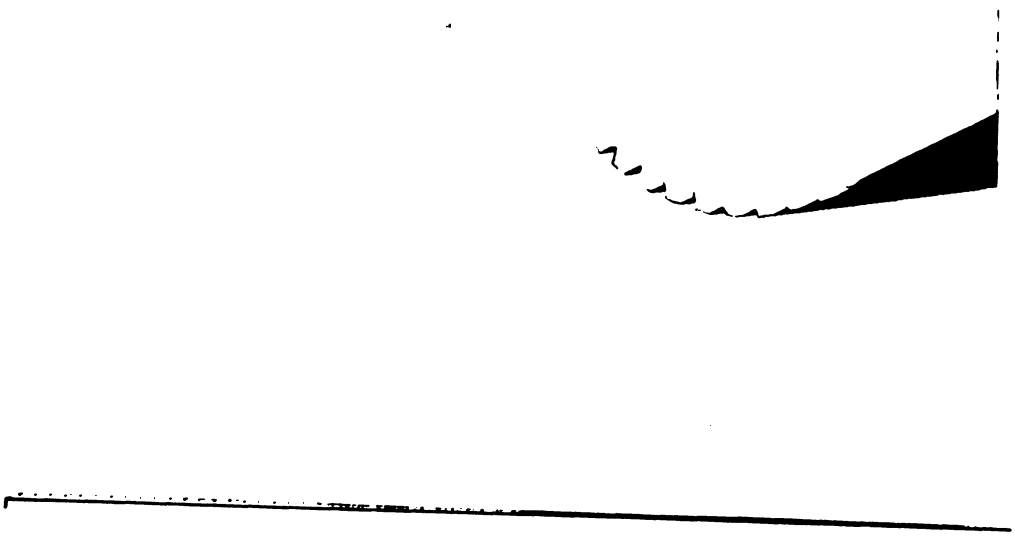

Fig. 30. Another view of fig. 29.

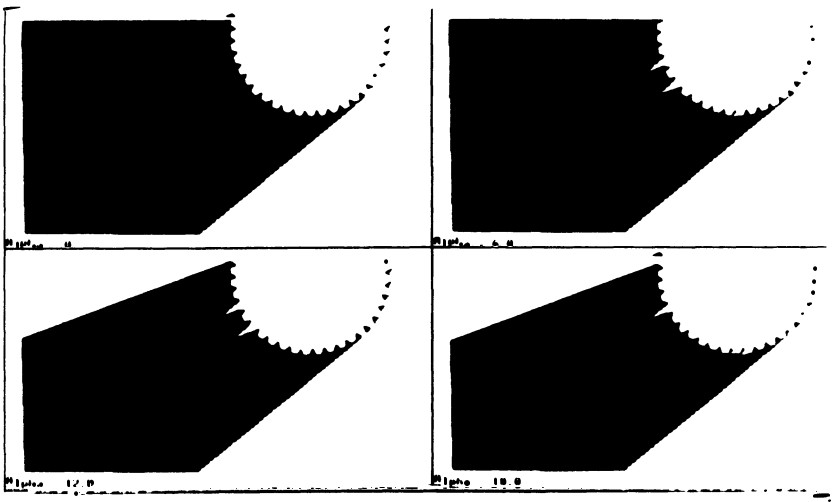

Fig. 31. In the next figures, B is permitted to rotate when pushed. The projection regions are computed across J by the propagation and union algorithm. We show four slices of generalized configuration space, at $\alpha = 0°, 6°, 12°$, and $18°$. The projections take into account possible rotation of B under pushing. Here the weak backprojections across slices are shown. The "spikes" represent regions from which jamming of the gears must occur.

* 7. On the Recognizability of EDR regions

In section (2.1) it was observed that if the termination predicate can distinguish between the goal G and the EDR region H, then H is a good EDR region and an EDR strategy was in hand. Formally, we write this recognizability constraint as[1]

$$P_{R,\theta}(\{G, H\}) = R. \tag{3}$$

This says that the (strong) preimage of the set of goals $\{G, H\}$, with respect to commanded velocity v_θ^*, is all of R. When we have a *set* of goals, the termination predicate must return *which* goal (G or H) has been achieved. This is different from $P_{R,\theta}(G \cup H)$, which means the termination predicate will halt saying "we've terminated in G or H, but I don't know which." The region R appears on both sides of (3) because the preimage depends on knowing where the motion started. This is a subtle point, see [21,13,14]. Thus solving preimage equations like (3) for R is like finding the fixed point of a recursive equation. Here, however, we know R, H, and G, so (3) is a constraint which must be true, rather than an equation to solve. Presumably (3) is easier to check than to

[1] We view $P_{R,\theta}$ as a map. In the informal development we denoted the image of this preimage map by P.

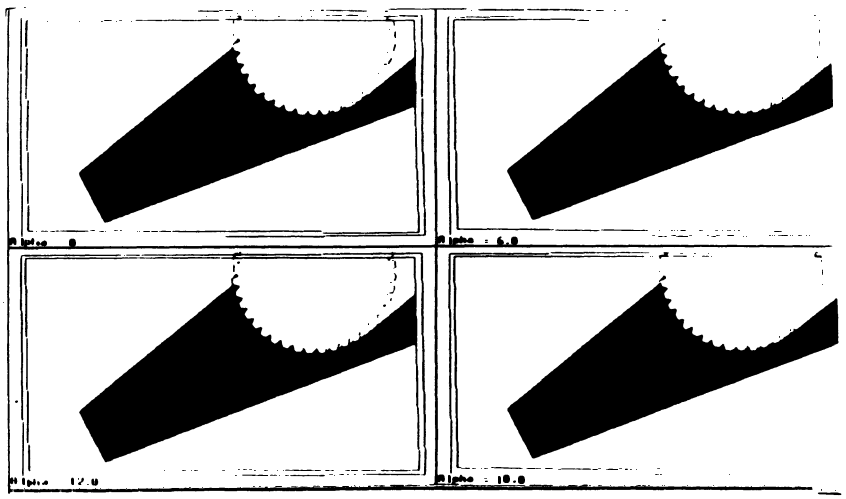

Fig. 32. The forward projections of the start region, propagated and unioned across slices.

solve;[2] see [21,13,14,12].

With this understood, we can now characterize P and R' precisely. This requires specifying the start regions :

$$R' = P_{R',\theta}(G) \tag{4}$$

$$P = P_{F_\theta(R),\theta}(G). \tag{5}$$

$\hat{P}$ is analogously defined by adding "hats" to the P's in (5).

*8. The Structure of Goals in Phase Space

In this section, we examine the structure of phase space goals in some detail.

A goal in phase space is a region in Position-space × Velocity-space. A phase space goal is attained when the actual position and velocity can be guaranteed to lie in the region. We have actually been using phase space goals all along, since the velocity sensors are used to recognize goals. The introduction of arbitrary phase-space goals is problematic, see [14]. Here the goals are sufficiently simple that these dangers are avoided.

[2] i.e., than to solve for R.

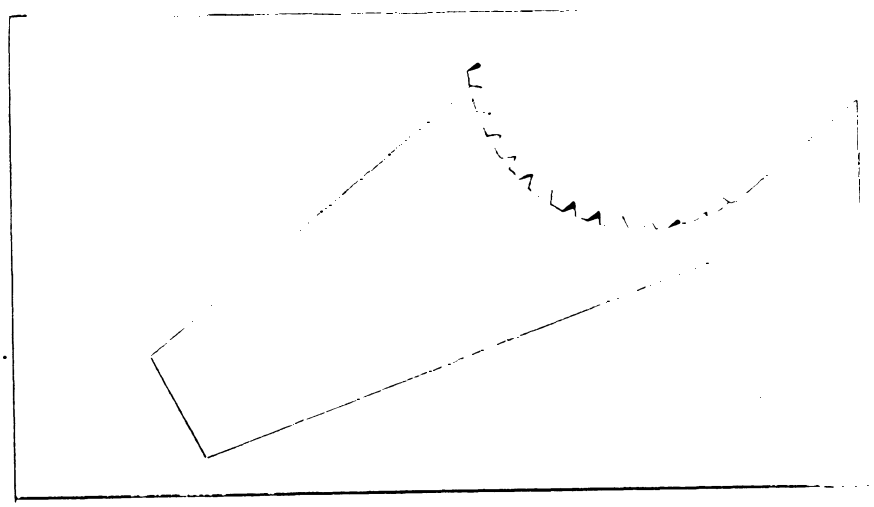

Fig. 33. Detail of the forward projection for $\alpha = 12°$. Note the effect of propagation in the clockwise-most region of the forward projection. This region can only be reached when rotated to from neighboring slice. The shaded region shows the portion of the forward projection which has been propagated by pushing from slice $\alpha = 18°$.

We'll begin with the simpler example. In fig. 36 we proposed a partition of the forward projection F of R into three regions:

Strong preimage, P

Weak but not strong preimage, $\hat{P} - P$

Forward projection outside the weak preimage, $F - \hat{P}$.

Here, the partition was "good" for the purposes of EDR for all velocities, and we could let H be the forward projection outside the weak preimage. We can extend this partition into phase space as shown in fig. 37. There is a natural projection π of Position-space × Velocity-space onto Position-space which sends a pair (x, velocity-at-x) to its position x. Given a region U in position space, we can lift it to phase space to obtain $\pi^{-1}(U)$, the *cylinder* of all velocities over U. A point in $\pi^{-1}(U)$ is (x, v) where x is in U, and v is any velocity at x.[1]

We lift the partition by applying the inverse projection map to obtain a partition of phase space:

Cylinder over strong preimage, $\pi^{-1}(P)$

[1]The cylinders may then intersected with the forward projection of R (in phase space) to obtain more constraint. This may be done by first restricting the domain of π to the forward projection.

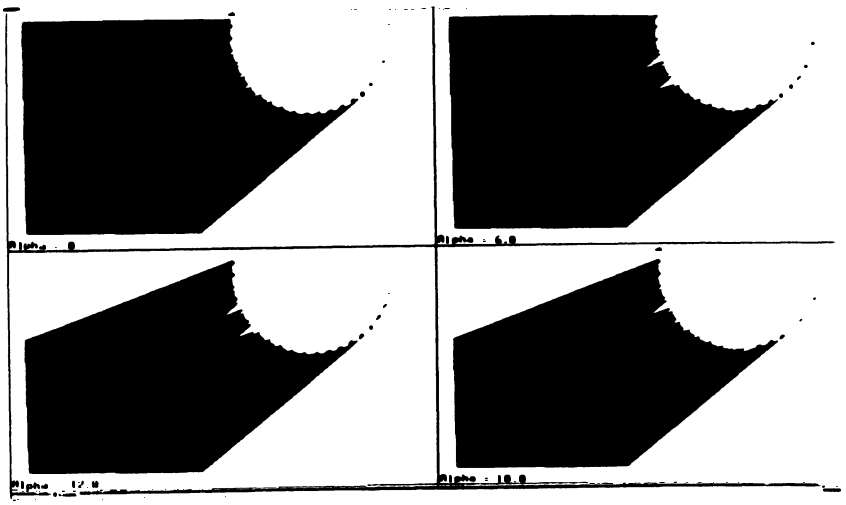

Fig. 34. The weak minus strong backprojections, propagated and unioned across slices.

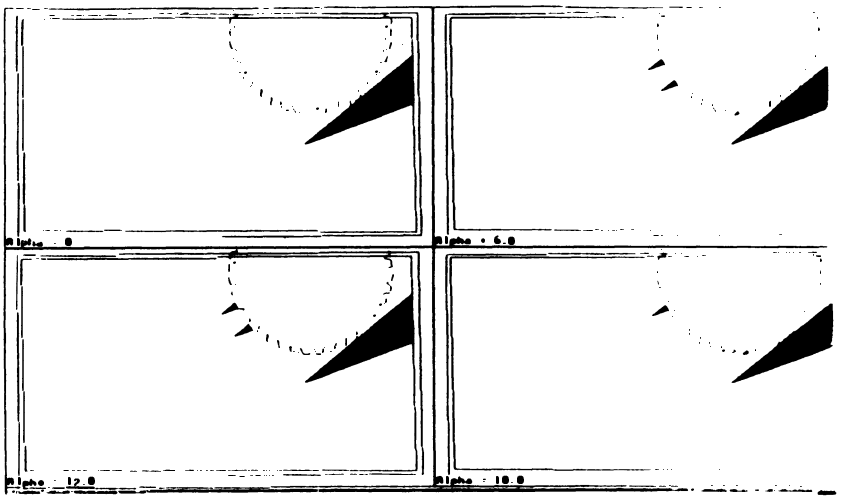

Fig. 35. The H_0 region (forward projection minus weak backprojection) across slices.

Cylinder over weak but not strong preimage, $\pi^{-1}(\hat{P} - P)$

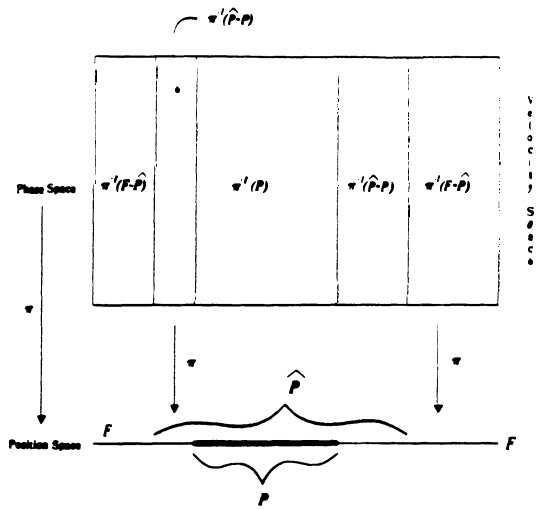

Fig. 36. Position-space is one dimensional. Therefore phase space, which is Position-space × Velocity-space, is 2-dimensional. The velocity "axis" is shown vertically. π projects a position and velocity to the position. We lift the strong preimage P to a cylinder $\pi^{-1}(P)$. We also obtain the cylinders over the weak but not strong preimage $\hat{P} - P$, and over the forward projection outside the weak preimage, $F - \hat{P}$.

Cylinder over forward projection outside the weak preimage, $\pi^{-1}(F - \hat{P})$.

See fig. 36. Now, the cylinder over G and the cylinder over $F - \hat{P}$ are the new goals in phase space. The latter cylinder is the *phase space EDR region* for fig. 15. Both are simply cylinders: all velocities are legal. [2]

Now we must deal with the tricky sticking region H_s in fig. 16. We begin by lifting the partition to phase space again (see fig. 37). Next, we "mark off" regions in the lifted partition to form a phase space EDR region, which we denote $\tilde{H}$. The entire cylinder over $F - \hat{P}$ is clearly in $\tilde{H}$, since its projection (under π) is outside the weak preimage. But the cylinder over H_s is not entirely within $\tilde{H}$: only sticking velocities over H_s are.

Formally, H_s is the set of all points x in the weak but not strong preimage, such that sticking can occur at x. We wish to distinguish the sticking velocities in H_s. Under generalized damper dynamics, these are essentially the zero velocities. Let $Z(H_s)$ denote the zero velocities over H_s, that is, the set of pairs $(x,0)$ for x in H_s. This set is in phase space.[3] Then we see that $Z(H_s)$ is also in the phase space EDR region $\tilde{H}$. Thus $\tilde{H}$ is the union of the sticking velocities over H_s, and all velocities over the forward projection outside the weak preimage:

[2]The weak and strong preimage, and the forward projection are drawn Venn diagrammatically in 1D.

[3]We could also let $Z(H_s)$ be the set of velocities over H_s which are smaller than some threshold.

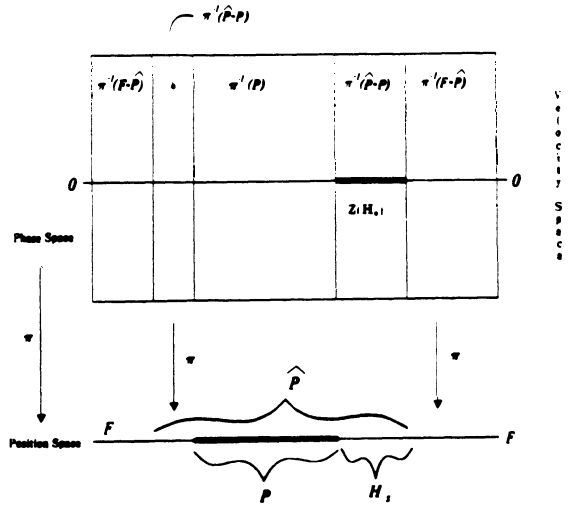

Fig. 37. Compare fig. 36. We have indicated the sticking region H_s in the weak preimage. The zero-velocities $Z(H_s)$ over H_s are in the cylinder over H_s. The EDR region $\tilde{H}$ is the union of $Z(H_s)$ and the shaded cylinders over the forward projection outside the weak preimage $F - \hat{P}$.

$$\tilde{H} = Z(H_s) \cup \pi^{-1}(F - \hat{P}). \tag{6}$$

To use $\tilde{H}$ as an EDR region, we must now ensure that $\tilde{H}$ and the cylinder over G are distinguishable goals. This amounts to allowing goals in phase space—that is, allowing the preimage operator to take simple phase space goals as arguments, and rewriting (3) as

$$P_{R,\theta}(\{\pi^{-1}(G), \tilde{H}\}) = R. \tag{3a}$$

The impact of (3a) is discussed in more detail in [12]. One point is worthy of comment. If the strong preimage is known, the definition of (phase space) EDR regions is *constructive up to reachability.* By this we mean that when backprojections, set intersections and differences, and friction cones can be computed, then so can $\tilde{H}$. With $\tilde{H}$ is hand, we add the recognizability constraint (3a) to obtain an EDR strategy.

*9. More on Weak Preimages

We now examine the structure of the "weak but not strong preimage," $\hat{P} - P$ in more detail. It suggests a number of implementation issues. Consider figs. 16,17 once more. Suppose we have a

trajectory originating in R, subject to the control uncertainty shown. We do not wish to terminate the motion while it remains in the weak preimage, since fortuitous sensing and control events could still force recognizable termination in G. However, we can terminate the motion as soon as we recognize egress from the weak preimage. This is why the forward projection outside the weak preimage is contained in the EDR region.

As we have seen, however, it is possible for a trajectory to remain within the weak but not strong preimage forever. For example, it can stick in H_s forever. To handle this case, we introduced phase space EDR goals.

There are other conditions under which a trajectory could stay in $\hat{P} - P$ forever: (a) if the environment is infinite, or $\hat{P} - P$ is unbounded. (b) The trajectory "loops" in $\hat{P} - P$ forever. (a) and (b) are qualitatively different from the case of sticking forever in H_s, because they require motion for infinitely long. In practice this may be handled by terminating the motion in $\hat{P} - P$ after a certain elapsed time. This is called "constructing termination predicates which time-out." In fact, this "solution" works for sticking in H_s also.

An alternative is to extend our earlier zero-velocity analysis to all of $\hat{P} - P$. That is, we terminate the motion in the weak but not strong preimage when the actual velocity is (close to) zero. Formally this rewrites (6) as

$$\tilde{H} = Z(\hat{P} - P) \cup \pi^{-1}(F - \hat{P}). \tag{6a}$$

Both this and our formal handling of phase space goals for H_s (6) are subject to the "Rolles theorem bug." That is, a trajectory which "reverses direction" will have zero velocity at some point. Hence by (6, 6a) it will be judged to have stuck. This is undesirable. In practice this can be fixed by again requiring the trajectory to stick for some elapsed time. Time-out termination predicates have the following practical justification. We imagine some low-level control mechanism which detects sticking, and after a certain time interval freezes the robot at that configuration and signals termination. Presumably such a mechanism is designed to avoid damage to the robot by detecting excessive torques. It also avoids plans with long delays while the planner waits for the motion to slide again.

The role of time in constructing EDR regions can be formalized by explicitly introducing time into the goal specification. Thus, goals become regions in phase space-time; points in goals have the form (x, v, t), where x is a position, v a velocity, and t a time. Suppose given a goal G in generalized configuration space, we form a phase space-time goal which is the product of $\pi^{-1}(G)$ with a compact time interval. It seems hopeful that the EDR axioms are satisfiable by EDR regions which have the form of a product of (6) with a compact time interval. More study is required.

One also can conceive of alternative models for sticking behavior. H_s is all points in the weak but not strong preimage such that sticking *might* occur there. Note that we cannot guarantee that sticking will occur, since then the point would not be in the weak preimage. We could assume a probabilistic distribution of control velocities in B_{ec}. In this case we could infer that eventually, given an unbounded amount of time, a motion will be commanded which will cause sliding away from any point in H_s at which a trajectory originating in R sticks. In this case, the trajectory

cannot stick forever in H_s. I don't think robot controllers reliably enforce probabilistic distributions of commanded velocities, even if "dithering" control strategies are employed. Even if they could, this model of sticking makes life easier, since it essentially eliminates the possibility of sticking forever in $\hat{P} - P$. We will not make this assumption here. It does not address with the problem of "looping forever" within $\hat{P} - P$ in finite environments. It seems that time-out termination predicates and/or velocity thresholding must be used to solve the looping problem. Both solutions seem inelegant; the issue is subtle and is addressed further in [12].

* 10. Generalization to n-Step EDR Strategies

This section discusses the construction of n-step EDR strategies in more detail.

We first review example (3) (figs. 18,19), highlighting a subtle recognizability issue not emphasized in the prelude. However, this review may be skipped at first reading if you already have example (3) firmly in mind.

* 10.1. A Review of Example (3)

Consider fig. 18. Here there are two possible universes, both in the plane, so J is the two element discrete set, $\{1,2\}$. The start region is the union of R_1 in universe 1, and R_2 in universe 2. The goal exists in universe 1 but not in universe 2. There is no one-step EDR strategy which, from the start region, can guarantee to achieve G or recognize that we are in universe 2. In particular, there is no one-step EDR strategy which can be derived from the motion v_θ^*.

However, there clearly exist multi-step EDR strategies. We will construct one as follows. Recall that to construct one-step EDR strategies, we took as data a goal, a start region R, a commanded motion θ, and the preimage of the goal under θ. Given this data we constructed an EDR region. From the EDR region, we attempted to construct an EDR strategy that achieved the distinguishable union of the goal *or* the EDR region. Now, why does this fail in fig. 18? To answer this question, let us consider what the motion θ was supposed to achieve in universe 1. There is an 8-step plan in universe 1 which recognizably achieves G from start region R_1. It is obtained by back-chaining preimages in universe 1. The plan moves from R_1 to the region S_1 under v_θ^*. Then it slides along the top surface to vertex f. Next it slides to vertex e. It slides to the successive vertex subgoals d through a, and then a horizontal sliding motion achieves the goal G.

The strategy θ is guaranteed to achieve the surface S_1 from start region R_1. Suppose we try to extend it to an EDR strategy with start region the union of R_1 and R_2. The EDR region (6) is then simply the (cylinder over the) forward projection of the "bad" region, $F_\theta(R_2)$. (See fig. 18). There is no way that the termination predicate can distinguish between the forward projection of R_1 and the forward projection of R_2, hence no EDR strategy from θ exists.

We can easily construct a 2-step EDR strategy, however. First, we execute motion θ from the union of R_1 and R_2. This achieves a motion into S_1 in universe 1, or into S_2 in universe 2. The termination predicate cannot distinguish which has been attained. Suppose the second motion in the 8-step plan is v_ψ^* (see fig. 18), and is guaranteed to achieve the vertex subgoal f from start region S_1. We'll try to construct an EDR strategy out of this second motion. Take as data: the subgoal f, the start region $S_1 \cup S_2$, the horizontal motion ψ, and the preimage of f under ψ.[1] The EDR region for these data is the forward projection of S_2 under ψ (see fig. 19). Presumably this EDR region is distinguishable from f, and so we have constructed an EDR strategy at the second step. After executing the second step, we either terminate the motion as a failure, or proceed to vertex e, and eventually to the goal.

*10.2. Generalization: Push-Forwards

Now, let us attempt to capture the salient aspects of the n-step EDR strategy construction. We take as data an n-step plan, with start region R_1. The actual start region is some larger region, say, R. Above, we had R as the union of R_1 and R_2. The first motion in the plan is guaranteed to achieve some subgoal S_1 from R_1. Using this first motion from start region R, we try to construct an EDR region H_1, and a one-step EDR strategy that either achieves S_1 or signals failure by achieving H_1. If this succeeds, we are, of course, done.

Suppose we cannot distinguish between H_1 and S_1. In this case, we want to execute the first motion "anyway," and terminate "somewhere" in the union of S_1 and H_1. The termination predicate cannot be guaranteed to distinguish which goal has been entered.

This "somewhere" is called the *push-forward* of the first motion from R. The push-forward is a function of the commanded motion θ, the actual start region R, the region R_1 from which θ is guaranteed, and the subgoal S_1.[2] A particular example of push-forward is defined formally in [12]. In example (3), the push-forward (under θ) of R_2 is S_2. The push-forward of $R_1 \cup R_2$ is $S_1 \cup S_2$. The push-forward is similar to a forward projection, except that it addresses the issue of termination. In example (3), informally speaking, the push-forward from the region R (under some commanded motion θ) is the result of executing θ from R and seeing what happens. It is defined even when the strategy θ is only guaranteed from some subset (R_1) of R.

Having terminated in the push-forward of R (the union of S_1 and S_2 above), we next try to construct a one-step EDR strategy at the second motion of the n-step plan. The data are: the next subgoal T_1 after S_1 in the plan, the actual start region $S_1 \cup S_2$, the second commanded motion in the plan, and the preimage of T_1 under this motion.[3] This defines a formal procedure for constructing n-step EDR strategies. At each stage we attempt to construct a one-step EDR strategy; if this fails, we push-forward and try again.

---

[1] While S_1 is the preimage under ψ with respect to start region S_1, the preimage with respect to the entire forward projection of $S_1 \cup S_2$ includes the top edge between S_1 and f.

[2] Of course, it also depends on the termination predicate, sensing and control characteristics, etc.

[3] The preimage is with respect to the forward projection of the actual start region $S_1 \cup S_2$.

Actually, this description of the procedure contains a taradiddle. At each step we construct the EDR region as described. However, the one-step strategy we seek must achieve the distinguishable union of the EDR region and *all unattained subgoals* in the plan. That is, the EDR motion must distinguishably terminate in the EDR region, or the next subgoal, or *any* subsequent subgoal. This allows serendipitous skipping of steps in the plan.

By considering different data, that is, quantifying over all motions at each branch point of the n-step strategy, we can in principle consider all n-step strategies and define non-directional EDR strategies. This is at least as difficult as computing n-step non-directional preimages. If we wish to consider plans of different lengths, we must also quantify over all n. Needless to say, the branching factor in the back-chaining search would be quite large.

*10.3. More on the Push-Forward

The problem of defining the push-forward may be stated informally as follows: "Where should the motion be terminated so that later, after some additional number of push-forwards, a one-step EDR strategy may be executed."

Many different push-forwards can be defined. Using the notation above, note the motion is not even guaranteed to terminate when executed from R: it is only guaranteed from R_1. This means that velocity-thresholding and time may be necessary in the termination predicate. There are other difficulties: for example, *a priori* it is not even necessary that entry into the union of the subgoal S_1 and the EDR region H_1 be recognizable. Thus defining the push-forward is equivalent to defining where in $S_1 \cup H_1$ the motion can and should be terminated.

Depending on which push-forward is employed, we may or may not obtain an n-step EDR strategy. It is possible to define constraints on the push-forward that must be satisfied to ensure that a strategy will be found if one exists. These constraints are given in [12]. While we can give equations that the push-forward must satisfy, at this time a constructive definition is not known. This situation is similar to, and possibly harder than the problem of solving the general pre-image equation.

*10.4. An Approximation to the Push-Forward

We may have to approximate the desired push-forward. We give such an approximation here. In general, it does not satisfy the constraints given in [12]. We provide it to show what the push-forwards alluded to above are like. Such approximate push-forwards may prove useful in approximating the desired push-forward. The issue deserves more study; see [12]. Since this approximate push-forward is incomplete, the reader should consider its description here as illustrative of the research problem, and not as an endorsement.

The push-forward employed in example (3) was formed by "executing the strategy anyway, and seeing where it terminated." How do we formalize this idea? Consider the termination predicate as a function of the starting region, the initial sensed position, the commanded velocity, the goal(s),

and the sensor values. The sensor values are changing; the predicate monitors them to determine when the goal has been reached.[4] Now, if the termination predicate "knew" that in example (3) the start region was the union of R_1 and R_2, then the first motion strategy θ could never be terminated: the predicate could never ensure that the subgoal S_1 had been reached. This is simply because S_1 and S_2 are indistinguishable. But if we "lie" to the termination predicate and tell it that the motion really started in R_1, then the predicate will happily terminate the motion in $S_1 \cup S_2$, thinking that S_1 has been achieved. Viewing the termination predicate as a function, this reduces to calling it with the "wrong" arguments, that is, applying it to R_1 instead of $R_1 \cup R_2$. The push-forward we obtain is "where the termination predicate will halt the motion from all of $R_1 \cup R_2$, thinking that the motion originated in R_1."

Even formalizing the construction of this simple push-forward is subtle; details are given in [12]. While this approximate push-forward is incomplete, it does suffice for a wide variety of EDR tasks. The approximate push-forward captures the intuitive notion of "trying the strategy anyway, even if we're not guaranteed to be in the right initial region." It is incomplete because it fails to exploit sufficiently the geometry of the forward projection of the "bad" region. Better push-forwards must be found; this one is merely illustrative of the problems.

* 11. Geometric Reasoning About Pushing in Generalized Configuration Space

*11.1. Example: The Sticking Cone

As an example of how a planner could reason about friction in generalized configuration space, see fig. 38. Here we take the configuration spaces of the robot and of B to be cartesian planes. Assume that we can apply a two dimensional force f_c on the robot, and a two dimensional force f_j at the center of mass of B. (This assumption is for the sake of discussion; in pushing applications, f_j would be zero). The friction cone in generalized configuration space will then be four-dimensional. This is hard to draw; we have selected a fixed, negative normal component for f_j. The 3D force space at the point of contact $\overline{x}$ represents the product of the 2D forces that can be exerted by the robot on the surface of B, with the 1D tangential forces that can be applied at the center of mass of B. An applied force (f_c, f_j) in the cone in fig. 38 represents a combination of forces that causes no motion in $\mathcal{G}$, that is, neither sliding on the surface of B, nor of B on the table. Note that the cone in $\mathcal{G}$ is skewed out of the embedded tangent space to C at x. This is because when a force f_c is applied in the friction cone on the top surface of B, the block B can slide unless an opposing force is exerted tangentially at the center of mass of B.

[4] See [21,13,14] for a thorough discussion of termination predicates.

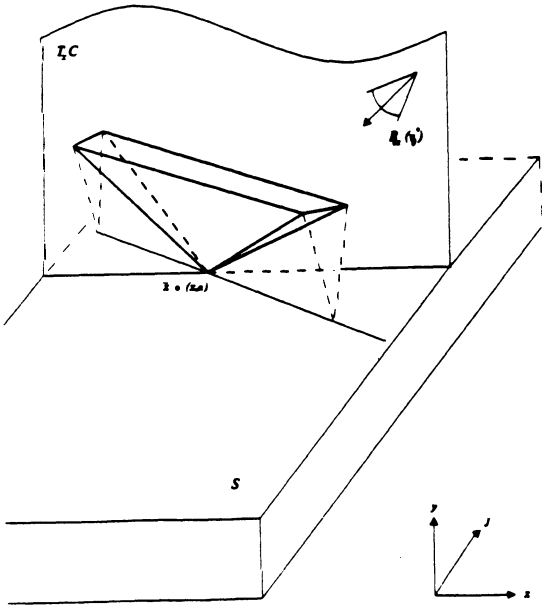

Fig. 38. Assume a fixed, negative normal force at the center of mass of B. The 3D force space at $\overline{x}$ represents the product of the 2D forces f_c that can be exerted by the robot on the surface of B, with the 1D tangential forces f_j that can be exerted at the center of mass of B. An applied force (f_c, f_j) in the cone represents a combination of forces that causes no motion in $\mathcal{G}$, that is, neither sliding on the surface of B, nor of B on the table. Note that the cone in $\mathcal{G}$ is skewed out of the embedded tangent space to C at x. This is because when a force f_c is applied in the friction cone on the top surface of B, the block B can slide unless an opposing force is exerted tangentially at the center of mass of B. By intersecting the sticking cone with the negative velocity cone, we can determine whether sticking is possible on S.

Let us call the cone in fig. 38 the *sticking cone* $\mathcal{K}$. Using the sticking cone, we can now specify a geometrical computation to determine when sticking occurs at $\overline{x}$, assuming generalized damper dynamics: Simply intersect the negative velocity control uncertainty cone $-B_{ec}(v_\theta^*)$ with $\mathcal{K}$. If the intersection is trivial, then sticking cannot occur. If the intersection is non-trivial, then sticking can occur. If the negative velocity cone lies inside $\mathcal{K}$, then sticking must occur.

This shows that the computation to determine whether sticking is possible at a point reduces to simple geometric cone intersection.

Now we return to the pushing application, by restricting the applied force f_j in J to be zero. See fig. 38. Assume it is impossible to apply force at the center of mass of B. Therefore, the velocity cone is two dimensional and lies entirely in the tangent space to C at x; it has no J component. This two-dimensional cone is intersected with the 3D cone $\mathcal{K}$ to determine whether sticking is possible at $\overline{x}$.

Let us emphasize that by insisting that the force f_j applied in J be zero, we obtain a two-

dimensional control uncertainty cone, even though generalized configuration space has four degrees of freedom. Thus, in the model error framework, the generalized control uncertainty can be viewed as a *non-holonomic constraint.*[1] Holonomic constraints are constraints on the degrees of freedom of the moving object(s); non-holonomic constraints are constraints on their differential motions. Holonomic constraints can be captured by surfaces in (generalized) configuration space. To capture non-holonomic constraints geometrically, we must introduce constraints in the phase space. This viewpoint is developed in [12], where we provide a more rigorous account of the construction of normals, friction cones, sticking cones, and the computation of reaction forces in generalized configuration space.

Acknowledgments

I would like to thank Tomas Lozano-Pérez, Michael Erdmann, Rod Brooks, Steve Buckley, John Canny, Eric Grimson, and Margaret Fleck for comments on an earlier draft of this paper, and for valuable discussions on motion planning and uncertainty. I am very grateful to John Canny for all the time he spent hacking on the 2D plane sweep module used in the implementation described here.

References

[1] **Abraham, R. and Marsden, J.** (1978). *Foundations of Mechanics,* Benjamin/Cummings, London.

[2] **Arnold, V. I.** (1978). *Mathematical Methods of Classical Mechanics,* Springer-Verlag, New York.

[3] **Brooks, R. A.** (1982). *Symbolic Error Analysis and Robot Planning,* International Journal of Robotics Research, Vol 1, no. 4, Dec., 29–68.

[4] **Brooks, R. A** (1985). *A Robust Layered Control System for a Mobile Robot,* MIT A.I. Lab Memo 864.

[5] **Brost, R. C.** (1985). *Planning Robot Grasping Motions in the Presence of Uncertainty,* Computer Science Department and the Robotics Institute, Carnegie-Mellon University, CMU-RI-TR-85-12.

[6] **Buckley, S. J.** (1987). *Planning and Teaching Compliant Motion Strategies,* Ph.D. Thesis. MIT, Dept. of EECS, 1987. Also MIT-AI-TR-936.

[7] **Canny, J. F.** (1986). *Collision Detection for Moving Polyhedra,* PAMI-8(2).

[8] **Chapman, D.** (1985). *Planning for Conjunctive Goals,* MIT AI-TR 802.

[9] **Donald, B. R.** (1984). *Motion Planning with Six Degrees of Freedom,* MIT AI-TR 791, Artificial Intelligence Lab..

[10] **Donald, B.R.** (1987). *A Search Algorithm for Motion Planning with Six Degrees of Freedom,* Artificial Intelligence, 31 (3).

[11] **Donald, B. R.** (1986). *Robot Motion Planning with Uncertainty in the Geometric Models of the Robot and Environment: A Formal Framework for Error Detection and Recovery,* IEEE Intl. Conf. on Robotics and Automation, San Francisco.

[1] Michael Erdmann [15] pointed out the relationship between non-holonomic constraints and the physics of $\mathcal{G}$ under pushing.

[12] **Donald, B. R.** (1987b). *Error Detection and Recovery for Robot Motion Planning with Uncertainty,* PhD Thesis; MIT Dept. EECS, Artificial Intelligence Laboratory, MIT-AI-TR 982.

[13] **Erdmann, M.** (1986). *Using Backprojections for Fine Motion Planning with Uncertainty,* IJRR Vol. 5 no. 1.

[14] **Erdmann, M.** (1984). *On Motion Planning With Uncertainty,* MIT AI Lab, MIT-AI-TR 810.

[15] **Erdmann, M.** (1986). *Personal Communication,* .

[16] **Fikes, R. and Nilsson, N.** (1971). *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving,* Artificial Intelligence vol. 2.

[17] **Gini, M. and Gini, G.** (1983). *Towards Automatic Error Recovery in Robot Programs,* IJCAI-83.

[18] **Hayes, P.** (1976). *A Representation for Robot Plans,* 4th IJCAI.

[19] **Lozano–Pérez, T.** (1981). *Automatic Planning of Manipulator Transfer Movements,* IEEE Trans. on Systems, Man and Cybernetics (SMC–11):681–698.

[20] **Lozano–Pérez, T.** (1983a). *Spatial Planning: A Configuration Space Approach,* IEEE Trans. on Computers (C–32):108–120.

[21] **Lozano-Pérez, T., Mason, M. T., and Taylor, R. H.** (1984). *Automatic Synthesis of Fine-Motion Strategies for Robots,* Int. J. of Rob. Res., Vol 3, no. 1.

[22] **Lozano-Pérez, T., and Wesley, M. A.** (1979). *An algorithm for planning collision-free paths among polyhedral obstacles,* Communications of the ACM (22):560-570.

[23] **Mason, M.T.** (1981). *Compliance and force control for computer controlled manipulators,* IEEE Trans. on Systems, Man and Cybernetics (SMC-11):418-432.

[24] **Mason, M.T.** (1982). *Manipulator Grasping and Pushing Operations,* MIT AI Lab, MIT AI-TR-690.

[25] **Mason, M. T.** (1984). *Automatic Planning of Fine Motions: Correctness and Completeness,* 1984 IEEE International Conference on Robotics, Atlanta Ga..

[26] **McDermott, D.** (1982). *A Temporal Logic for Reasoning about Processes and Plans,* Cog. Sci. 6, pp. 101-55.

[27] **Neivergelt, J., and Preparata, F. P.** (1982). *Plane-Sweep Algorithms for Intersecting Geometric Figures,* CACM Vol. 25, no. 10.

[28] **Requicha, A. A.** (1984). *Representation of Tolerances in Solid Modeling: Issues and Alternative Approaches,* Solid Modeling by Computers: From Theory to Applications; Plenum, N. Y..

[29] **Schwartz, J. T. and M. Sharir** (1982). *On the Piano Movers Problem II: General Properties for Computing Topological Properties of Real Algebraic Manifolds,* Department of Computer Science, Courant Institute of Mathematical Sciences, NYU, Report 41, February.

[30] **Shapiro, V.** (1985). *Parametric Modeling and Analysis of Tolerances,* GM Research Lab. Rept. CS-460.

[31] **Srinivas, Sankaran** (1977). *Error Recovery in Robot Systems,* Cal. Tech. Ph.D. Thesis, Computer Science.

[32] **Taylor, R. H.** (1976). *The Synthesis of Manipulator Control Programs from Task-level Specifications,* Stanford Artificial Intelligence Laboratory, AIM-282, July.

[33] **Ward, B. and McCalla, G.** (1983). *Error Detection and Recovery in a Dynamic Planning Environment,* AAAI.

[34] **Wilkins, D. E.** (1984). *Domain-Independent Planning: Representation and Plan Generation,* Artificial Intelligence, Vol. 22 No. 3.