Planning Multi-Step Error Detection and Recovery Strategies

Bruce Donald*

87-880
November 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

*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 contract N00014-81-K-0494 and in part by the Advanced Research Projects Agency under Office of Naval Research contracts N00014-85-K-0124 and N00014-82-K-0334. The author was funded in part by a NASA fellowship administered by the Jet Propulsion Laboratory.

Planning Multi-Step Error Detection and Recovery Strategies

Bruce R. Donald

4158 Upson Hall
Computer Science Department
Cornell University
Ithaca, NY 14853-7501

Abstract:

Robots must plan and execute tasks in the presence of uncertainty. Uncertainty arises from sensing errors, control errors, and uncertainty in the geometry of the environment. By employing a combined strategy of force and position control, a robot programmer can often guarantee reaching the desired final configuration from all the likely initial configurations. Such motion strategies permit robots to carry out tasks in the presence of significant uncertainty. However, compliant motion strategies are very difficult for humans to specify—for this reason we have been working on the automatic synthesis of motion strategies for robots. In previous work [D], we presented a framework for computing one-step motion strategies that are guaranteed to succeed in the presence of all three kinds of uncertainty. The motion strategies comprise sensor-based gross motions, compliant motions, and simple pushing motions.

However, it is not always possible to find plans that are guaranteed to succeed. For example, if tolerancing errors render an assembly infeasible, the plan executor should stop and signal failure. In such cases the insistence on guaranteed success is too restrictive. For this reason we investigate *Error Detection and Recovery (EDR)* strategies. EDR plans will succeed or fail recognizably: in these more general strategies, there is no possibility that the plan will fail without the executor realizing it. 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.

We describe techniques for planning multi-step EDR strategies in the presence of uncertainty. Multi-step strategies are considerably more difficult to generate, and we introduce three approaches for their synthesis: these are *the Push-forward Algorithm*, *Failure Mode Analysis*, and the *Weak EDR Theory*. We have implemented the theory in the form of a planner, called LIMITED, in the domain of planar assemblies.

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 N00014-85-K-0124 and N00014-82-K-0334. The author was funded in part by a NASA fellowship administered by the Jet Propulsion Laboratory.

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 robot and the environment. In this paper we describe a theory of planning multi-step Error Detection and Recovery (EDR) strategies for compliant motion assemblies. We have implemented the theory in the form of a planner, called LIMITED, in the domain of planar assemblies.

In previous work [D], we addressed two problems. The first is:

(1) The use of active compliance enables robots to accomplish tasks even in the presence of significant sensing and control errors. How can compliant motion strategies be synthesized in the presence of sensing, control, and geometric model error, such that the strategies are guaranteed to succeed so long as the errors lie within the specified bounds? As an example, consider a peg-in-hole assembly with sensing and control uncertainty, with toleranced parts. We wish to synthesize a compliant motion strategy that is guaranteed to suceed so long as the parts lie within the specified tolerances, and the sensing and control errors lie within the specified bounds.

We attacked this problem by introducing additional dimensions to the configuration space; each dimension represented a way in which the parts could parametrically vary. We termed the product space of the motion degrees of freedom and the geometric model variational dimensions "generalized configuration space"and showed how to compute "preimages" [LMT,E] of a geometrical goal in this generalized configuration space. The preimage of a goal is the set of (generalized) configurations from which a particular commanded compliant motion is guaranteed to succeed.

(2) However, it is not always possible to find plans that are guaranteed to suceed. For example, if tolerancing errors render an assembly infeasible, the plan executor should stop and signal failure. In such cases, the insistence on guaranteed success is too restrictive. For this reason we investigated ***Error Detection and Recovery (EDR)*** strategies. EDR plans will succeed or fail recognizably: in these more general plans, there is no possibility that the plan will fail without the executor recognizing it. 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. In [D], we gave a constructive, geometric definition of EDR strategies, and showed how thay can be computed for one-step strategies. (A ***one-step*** compliant motion strategy is a plan in which a force is commanded in one nominal direction (subject to uncertainty), until certain force- and position-sensing data indicate that the motion should be terminated. In the case of an EDR strategy, the run-time executor must also be able to disambiguate the result of the motion as a success (achieving the goal) or failure).

1.1 Examples

1.1.1 Application: Planning Gear Meshing

We must stress that EDR is not limited to problems with model error. There are many applications in which the geometry of the environment is precisely known, but in which guaranteed plans cannot be found, or are very difficult to generate. We now describe such a situation.

An interesting application domain for EDR is gear meshing. It is an example where EDR is applicable even though the shape of the manipulated parts is precisely known. 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 *even though the shape of B is precisely known, we choose to view the uncertainty in B's orientation as a form of model error.* 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. We distinguish between the rotation and non-rotation variants of the problem in order to highlight the additional techniques our planner employs when rotations are introduced.

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.

In an experiment, the EDR theory in the gear domain was applied using the planner, LIMITED, as follows. Consider the problem of meshing two planar gears, under uncertainty as above. Suppose that gear B can rotate passively but has unknown initial orientation, as above. Suppose that A has been gripped by a robot. The initial position of A is uncertain. The robot can impart either pure forces (translations), or pure torques (rotations) to A. The planner can choose the direction of translation or rotation. Can a multi-step strategy of commanded translations and rotations be found to mesh the gears?

LIMITED was able to generate an EDR strategy for this problem. The characteristics of the experiment are:

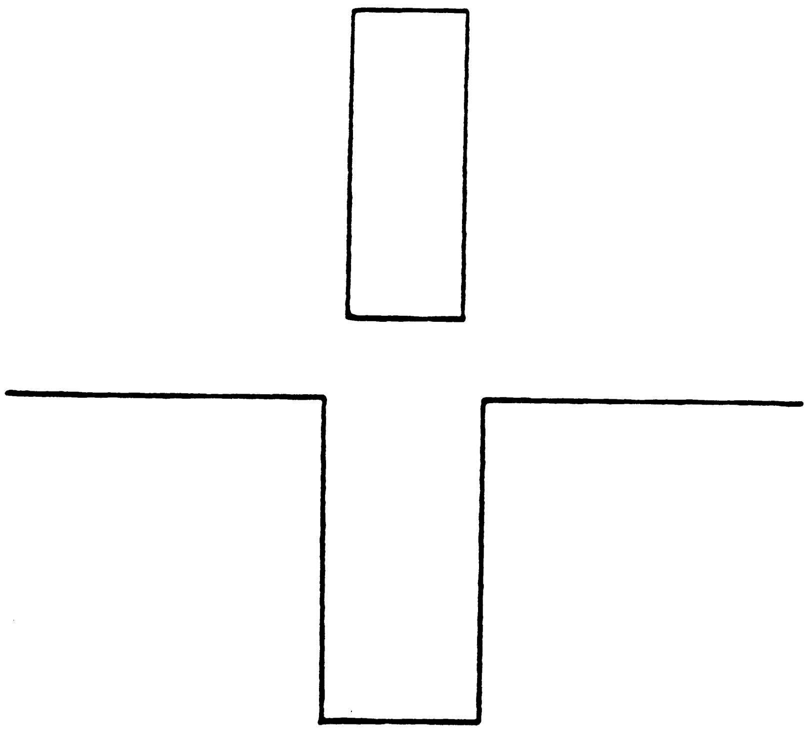

Figure 1: The goal is to insert the peg in the hole. No rotation of the peg is allowed. One can imagine a strategy which attempts to move straight down, but detects contact on the top surfaces of the hole if they occur. If the peg sticks on the top surfaces, the manipulator tries to move to the left 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.

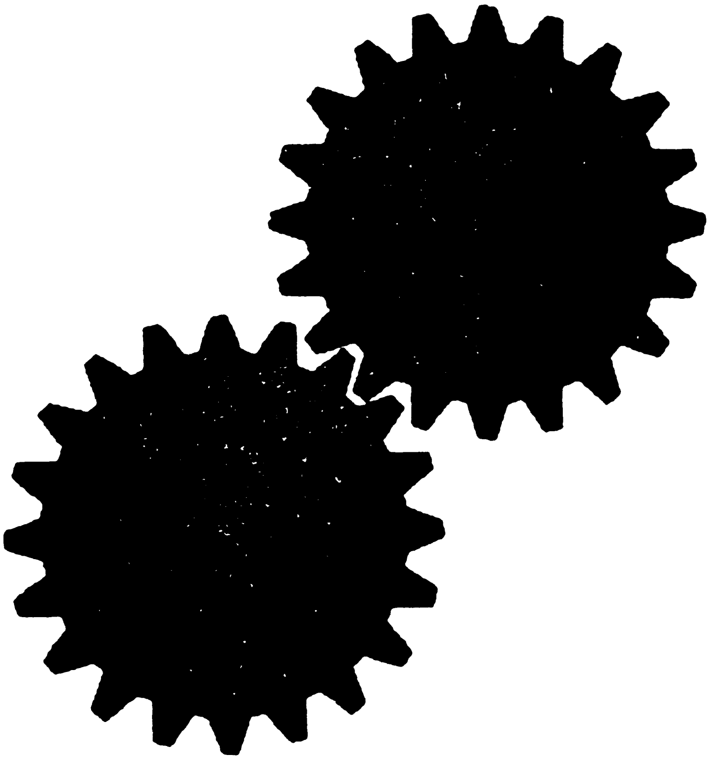

Figure 2: Geometric models of two gear-like planar objects A and B. A is grasped and can translate but not rotate. B can rotate about its center if pushed. The orientation of B is unknown. The task is to generate a motion strategy to mesh the gears.

- There are three degrees of motion freedom (two translational and one rotational) for A.
- There is one degree of rotational model error freedom (the orientation of B).
- It is possible to push B to change its orientation.
- There is sensing and control uncertainty.
- The geometry of the gears is complicated—they have many edges.
- Quasi-static analysis [Mason] is used to model the physics of interaction between the gears.

Thus we have a kind of four-degree of freedom planning problem with uncertainty and pushing. To generate multi-step EDR strategies under pushing, LIMITED employed the EDR theory together with a technique called *failure mode analysis*.

Now, there may exist a guaranteed strategy to mesh the gears. For example, experimental evidence suggests that for involute teeth gears, almost any meshing strategy will succeed. For other gear shapes perhaps some complicated translation while spinning A will always succeed. I don't know if there is such a guaranteed strategy for this case. It seems difficult for a planner to synthesize such guaranteed strategies, or even to verify them, if they exist at all.

A person might try to solve this problem with the following motion strategy:

- *Ram the gears together. See if they mesh.*

Or, somewhat more precisely,

- *Ram A into B. If they mesh, stop. If they jam, signal failure and try again.*

Probabilistically, this is a rather good strategy. It is certainly very simple, and probably easier to generate than a guaranteed strategy. If vision can be used to sense whether A and B are meshed, then it is an EDR strategy with just one step.

Suppose, however, that vision is poor, or that the gears are accessible to the robot gripper, but not to the camera. This means that position sensing will be very inaccurate, and hence may be of no use in determining whether the gears are successfully meshed. This will often be the case in practice. In this case, force sensing must be used to disambiguate the success of the motion (meshing) from failure (jamming in an unmeshed state). If the robot has force sensing, then it might use the following two-step EDR strategy:

- *Ram the gears together. Spin them to see whether they meshed.*

Or, again more precisely,

- *Ram A into B. Next, spin A. If A and B break contact, or if the gears* stick *(don't rotate), then signal failure. Otherwise, signal success.*

This strategy is essentially the one that LIMITED generates. The plan is

Motion 1: *Command a pure translation of A into B.*[1]

Terminate the motion based on force-sensing when sticking occurs (when there is no motion).

Motion 2: *Command a pure rotation of A.*

If breaking contact or sticking occurs, signal failure. Otherwise, signal success.

In this plan, motion (1) does not terminate distinguishably in success (meshed) or failure (jammed). That is, after motion (1) terminates, the plan executive cannot necessarily recognize whether or not the gears are meshed. LIMITED predicts this, and generates motion (2), which disambiguates the result of motion (1). The generation of the second, disambiguating motion involves the use of *failure mode analysis. Breaking contact* and *sticking* are examples of failure modes. The second motion is generated so that from any unmeshed state resulting from motion (1), all possible paths will terminate distinguishably in a failure mode. Failure mode analysis is a robust subtheory of EDR by which LIMITED generates multi-step strategies under pushing.

The theory and implementation behind the generation of motion (1) were discussed extensively in [D]. While we will review these techniques briefly, our concern in this paper is how to generate the multi-step strategy above—or more precisely, how to "extend" motion (1) into a 2-step strategy.

1.1.2 Experiment: Peg-in-Hole with Model Error

This section describes a plan that was generated by LIMITED for a peg-in-hole problem with model error. It gives the flavor of how EDR strategies work. Since pushing motions are not involved here, LIMITED does not use failure mode analysis to solve this problem.

Another peg-in-hole problem is depicted in fig. 3. Again, as in fig. 1, there is uncertainty in the width of the hole; that is, the width is known to lie within some given interval. In addition, there are chamfers on the sides of the hole. The depth of the chamfers is also unknown, but we are given bounds on the depth. Finally, the exact

[1] LIMITED generates the actual force vector.

orientation of the hole is uncertain. The geometry of the hole is input to the planner as a set of parametrically defined polygons. They are defined by a three parameter family, for *width* of the hole, *depth* of the chamfers, and *orientation* of the hole. An associated bounding interval is also input for each parameter. The geometry of the peg is input as a polygon.

In this problem, the width of the hole may be smaller than the width of the peg. Thus there can exist no strategy that is guaranteed to succeed for all geometric uncertainty values. However, assume that the assembly—the hole geometry—is inaccessible to robust vision or position-measuring devices. In particular, the measurement error will typically determine the model error bounds, which in this example are large for the purpose of illustration. *Thus it is not a priori possible to measure the dimensions ahead of time to determine whether or not the assembly is feasible.* Instead, the best we can hope for is an EDR strategy: a strategy that takes some action in the world to attempt the assembly, but whose outcome can be recognizably diagnosed as success or failure by the run-time robot executor.

The peg is allowed to translate in the plane. Its motion is modeled using generalized damper dynamics. This permits sliding on surfaces about the hole. Friction is modeled using Coulomb's law. With these dynamics and *perfect* control, the peg would exhibit straight-line motions in free space, followed by sliding motions in contact, where friction permits. Here, however, there is control uncertainty, which is represented by a cone of velocities. Motions in free space fan out in a kind of "spray." Again, sliding is possible on surfaces, but so is sticking, depending on the effective commanded velocity at a given instant. (In this case, we say sliding is *non-deterministic*). The size of the control uncertainty cone of velocities is an input to the planner. Whether sticking may occur on an edge may be computed by intersecting the friction cone with the *negative* control uncertainty cone.

It is possible to sense the position of the peg and the forces acting on it. This information is only approximate. The error bound on the position sensor readings is input to the planner as the radius of a disc.

LIMITED generates plans using a configuration space representation of the constraints [Lozano-Pérez]. In the plane, one imagines shrinking the moving object to a point, and correspondingly "growing" the obstacles. The point must be navigated through free-space, sliding on surfaces, and so forth, into the hole. Fig. 4 shows configuration spaces for different parametric variations of model error. Notice that when the "real" hole is too small for the peg to fit, then there is simply no hole at all in the corresponding configuration space. Each frame in fig. 4 is called a "slice;" a slice represents a cross-section where the model error parameters are constant. To synthesize an EDR strategy, LIMITED must in some sense consider all such slices. In practice LIMITED works by constructing a finite, although typically large number of slices. We will show how in many cases, only a low polynomial number need be considered. LIMITED begins by considering a small number of slices, and generates a tentative motion strategy. This strategy must pass a test—which we call the EDR test—to be recognized as an EDR strategy. One of the chief goals of this thesis is to derive this test, and to make it formal and algorithmic. Next, LIMITED attempts to "generalize" the strategy by considering successively more slices. The strategy is modified so that it passes the EDR test in all slices. The number of slices considered is the *resolution* of the planning. This approach is called *multi-resolution* planning.

Let's consider an EDR plan that LIMITED computed for this problem. Figs. 5-13 show the plan graphically. Qualitatively, the plan may be described as follows:

- *(1) First, move left and slightly down. The motion will terminate on the left side of the hole, on the left chamfers, or overshoot the hole entirely. Where the motion terminates depends both on the trajectory evolution within the control uncertainty, and on the actual geometry of the hole. The motion may, however, slide down the left edge of the hole all the way into the goal. However, this sliding is non-deterministic, and the motion may stick anywhere along that edge. Since the first motion may terminate arbitrarily close to the goal region,* LIMITED *predicts that the run-time executive system cannot necessarily distinguish whether or not the first motion failed to achieve the goal.*

- *(2) The termination regions from motion (1) are taken as the start regions for a new motion. Next, try to recover by commanding a motion straight down and slightly to the right. This motion may achieve the goal, or may undershoot it, or may overshoot it. The second motion terminates when the peg sticks on a surface. If such a termination surface is outside the goal, it is called a* failure region. LIMITED *calculates that after the second motion, the* failure regions *are distinguishable from the goal regions. Hence after the second motion, the run-time executive can recognize whether or not the plan has failed.*

Finally, since LIMITED is a forward-chaining planner, it is possible to take the failure regions from motion (2) and plan a third recovery motion. Thus, roughly speaking, in the EDR framework, *recovery* actions are planned by forward-chaining from the failure regions of the previous motion. When the failure regions are potentially indistinguishable from the goal (using sensors), then the recovery action must satisfy the formal EDR test when executed from the *union* of the goal and the previous failure regions. For example, when we view motion strategies as "mappings" between subsets of configuration space, then typical "robust" recovery actions are EDR plans in which the goal is a "fixed point."[2] Motion (2) is an example of such a one-step EDR plan.

Figs. 5–13 show the plan in just four different slices, to give a flavor for the plan. The rest of the slices may be found later in the thesis. Fig. 5 shows the configuration spaces of the four slices. The goal region here is shaded black. Note that in one slice, the goal disappears. The initial uncertainty in the position of the peg is represented by

[2]That is, when motion (2) originates in the goal, it also terminates recognizably in the goal.

constraining the reference point (the point to which the peg has been shrunk) to lie in one of the start regions in fig. 6.

Figs. 7–8 represents the *forward projection* of the first motion. This region is the outer envelope of all possible trajectories evolving from the start regions. It is the set of all configurations that are reachable from the start regions, given the commanded velocity and control uncertainty cone.

Fig. 9 shows the termination regions for motion (1). The termination regions outside the goal are not necessarily distinguishable from the goal.

Figs. 10–11 show the forward projection of the second motion.

Fig. 12 shows the termination regions for the second motion.

Fig. 13 shows the size of the position sensing uncertainty ball. The goal and the failure regions in fig. 12 are distinguishable using sensors.

2 A Review of the EDR Theory

In this section we review the EDR theory developed in [D]. This review is necessarily somewhat abbreviated. The reader is cautioned that the account below is somewhat intuitive and informal; we attempt to describe the key points of the EDR theory in the plain style, and omit proofs and mention of certain subtle complications. For a detailed development, please refer to [Donald 86, 87, 88].

2.1 Motivation: Research Issues

The gross motion planning problem with no uncertainty has received a great deal of attention recently. In this problem, the state of the robot may be represented as a point in a configuration space. Thus moving from a start to a goal point may be viewed as finding an arc in free space connecting the two points. Since the robot is assumed to have perfect control and sensing, any such arc may be reliably executed once it is found. In particular, given a candidate arc, it may be tested. That is, motion along the arc may be simulated to see whether it is collision free. For example, an algebraic curve may be intersected with semi-algebraic sets defining the configuration space obstacles. In the presence of uncertainty, however, we cannot simply simulate a motion strategy to verify it. Instead, we need some technique for simulating *all* possible orbits, or evolutions of the robot system, under any possible choice of the uncertain parameters. With sensing and control uncertainty, the state of the robot must be viewed as a subset of the configuration space. Motions, then, can be viewed as mappings between these subsets. Of course there are many such subsets! From this perspective, it is clear that a chief contribution of [LMT] has been to identify and give a constructive definition for a privileged class of subsets, called *preimages*, and show that it is necessary and sufficient to search among this class. This framework appears very promising for planning guaranteed motion strategies

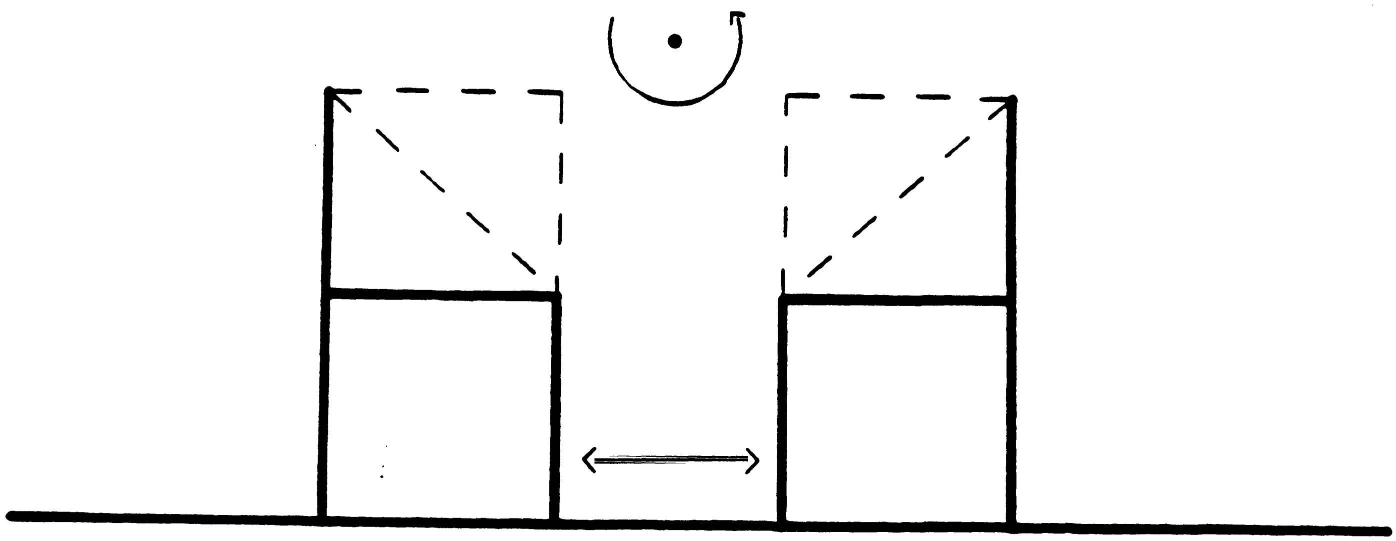

Figure 3: A peg-in-hole environment with model error. The width of the hole (α_1), depth of chamfer (α_2), and orientation of the hole (α_3) are the model parameters. The hole is allowed to close up.

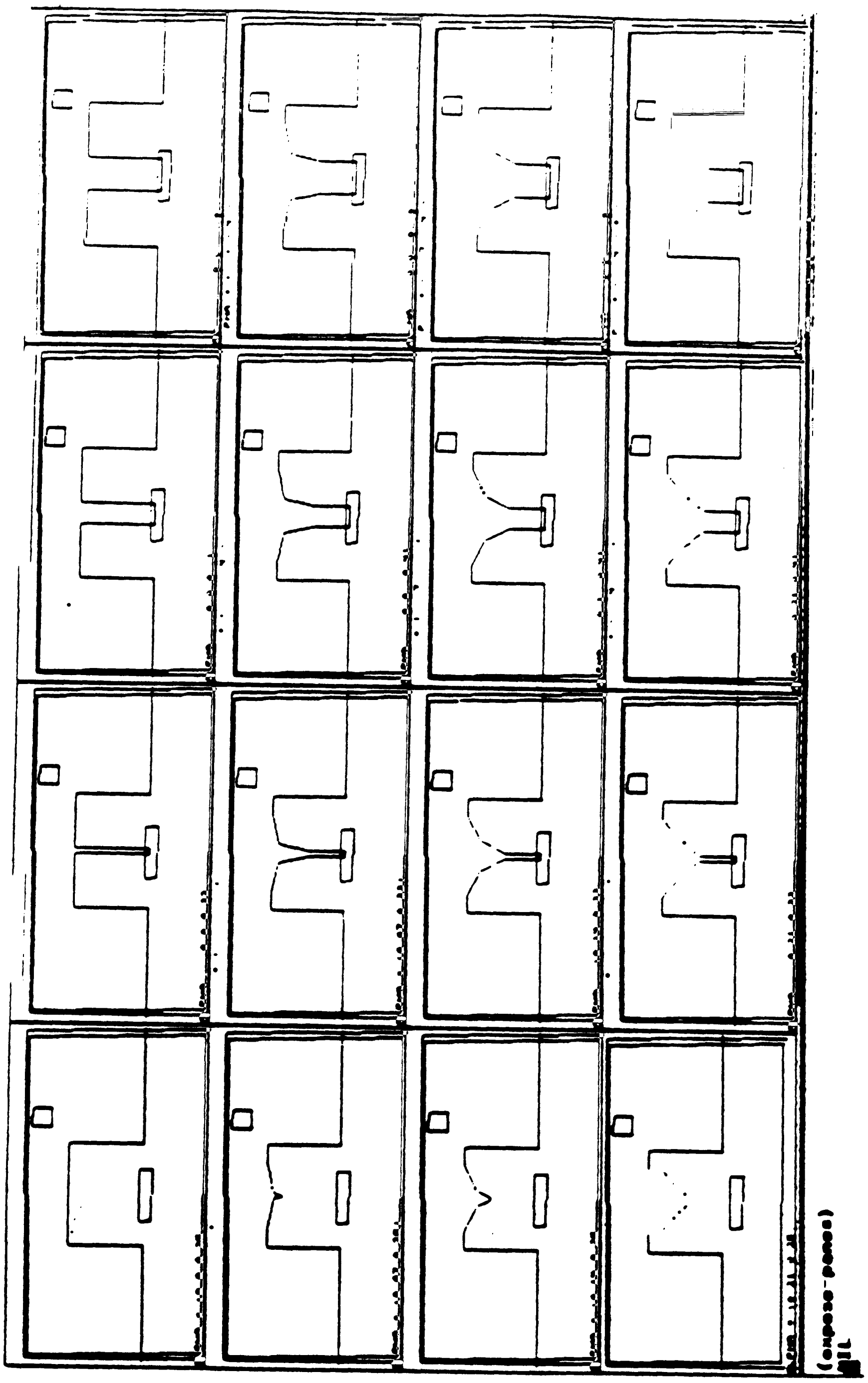

Figure 4: **The configuration space slices for many different parametric model error values. These configuration spaces were generated for the peg-in-hole problem with model error depicted in fig. 3. Fig. 4a shows a few slices taken at constant orientation, whereas in fig. 4b, more slices are shown at various orientations.**

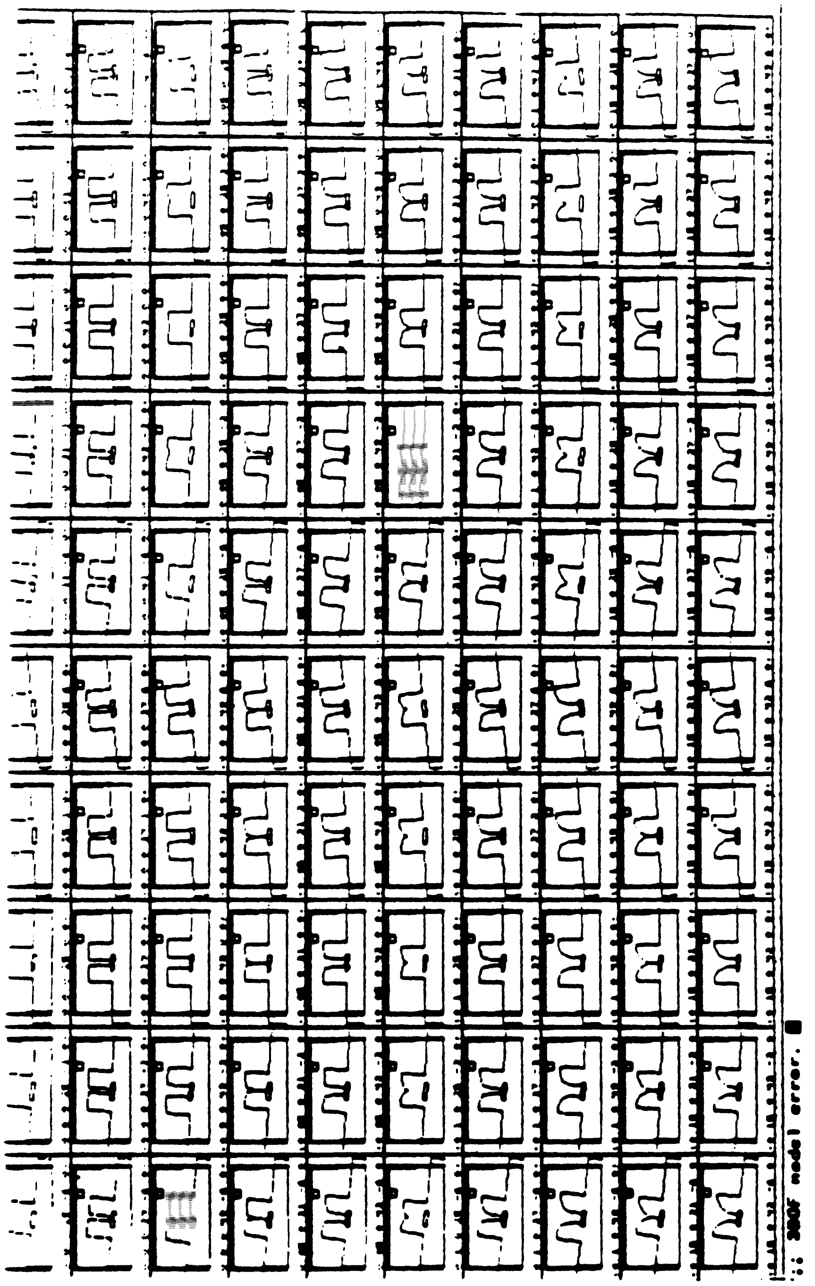

Figure 4b.

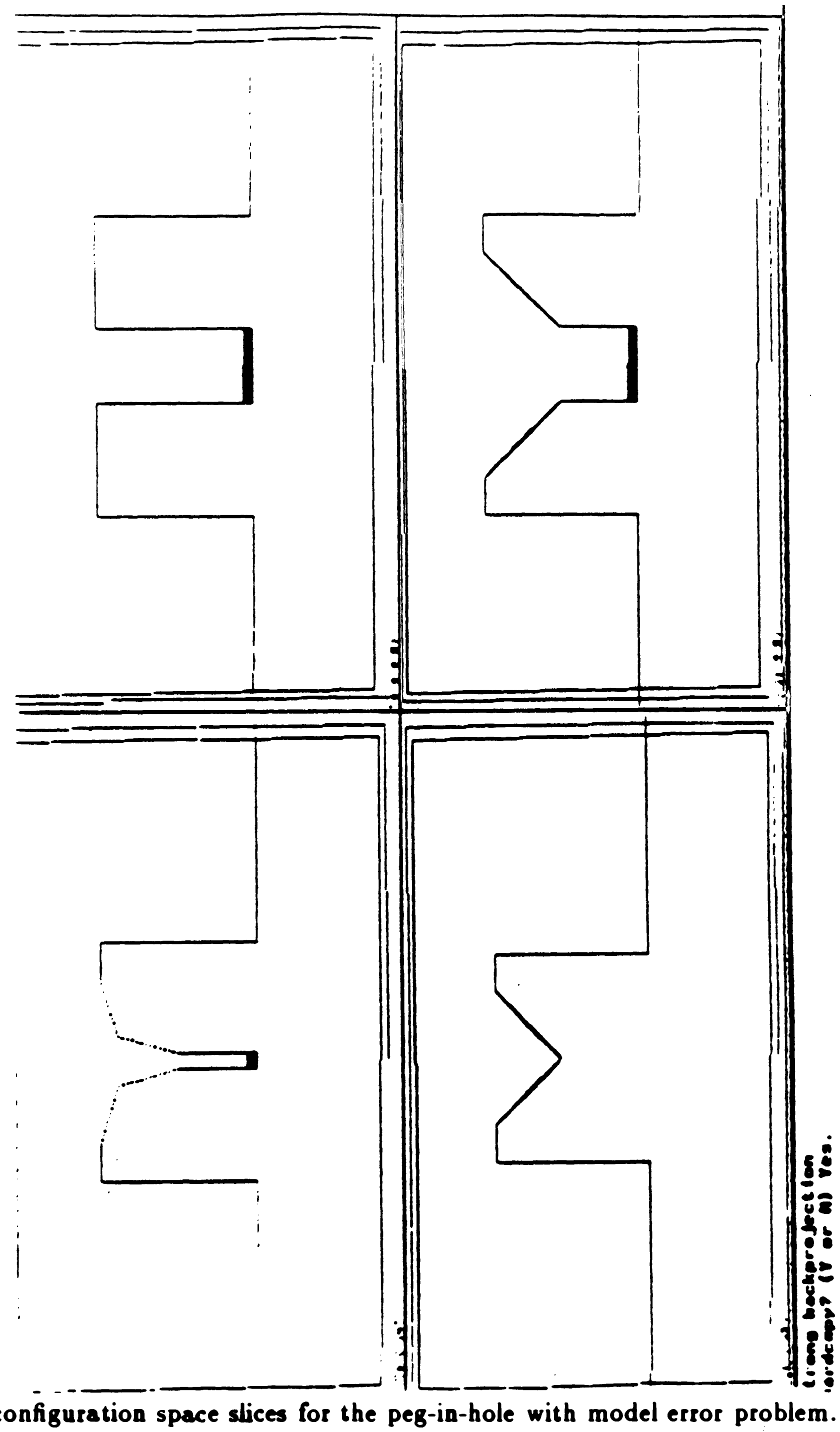

Figure 5: 4 configuration space slices for the peg-in-hole with model error problem. The goal region is shaded black. In one slice, the goal vanishes.

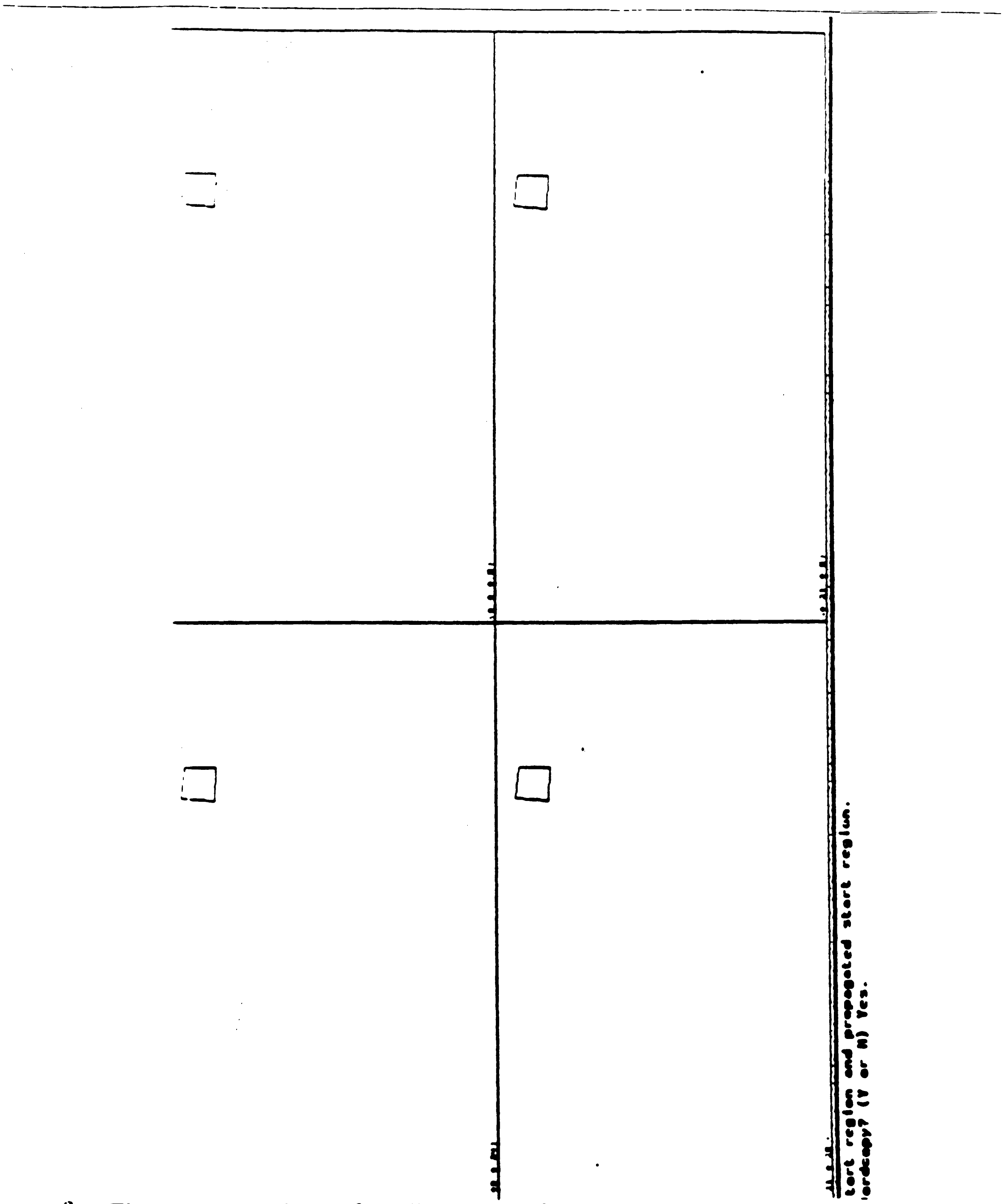

Figure 6: The start region in the four slices. The reference point of the peg is known to start within this region.

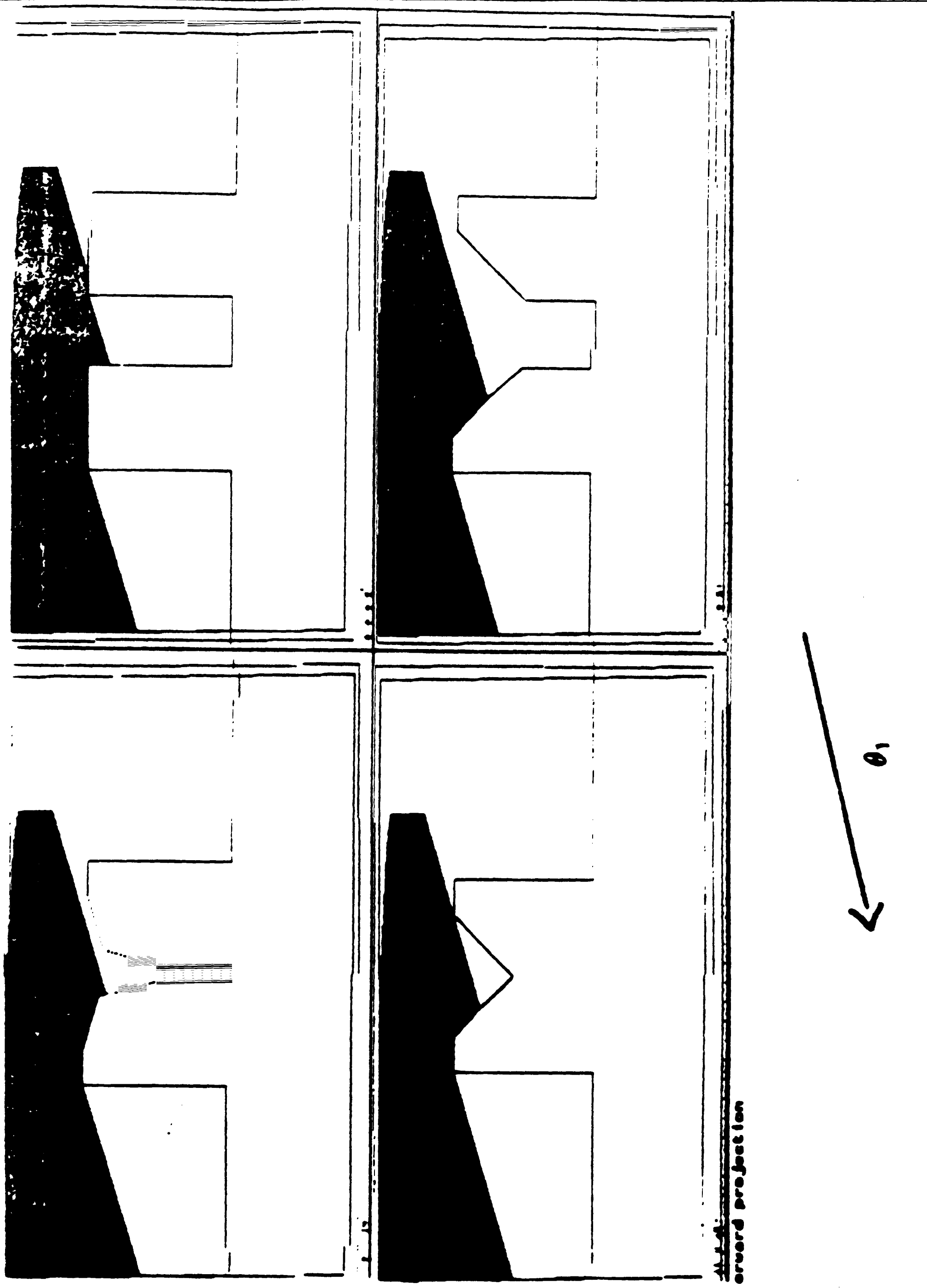

Figure 7: The forward projection of the first motion. This region is the outer envelope of all possible trajectories evolving from the start regions. It is the set of all configurations that are reachable from the start regions, given the commanded velocity and control uncertainty cone.

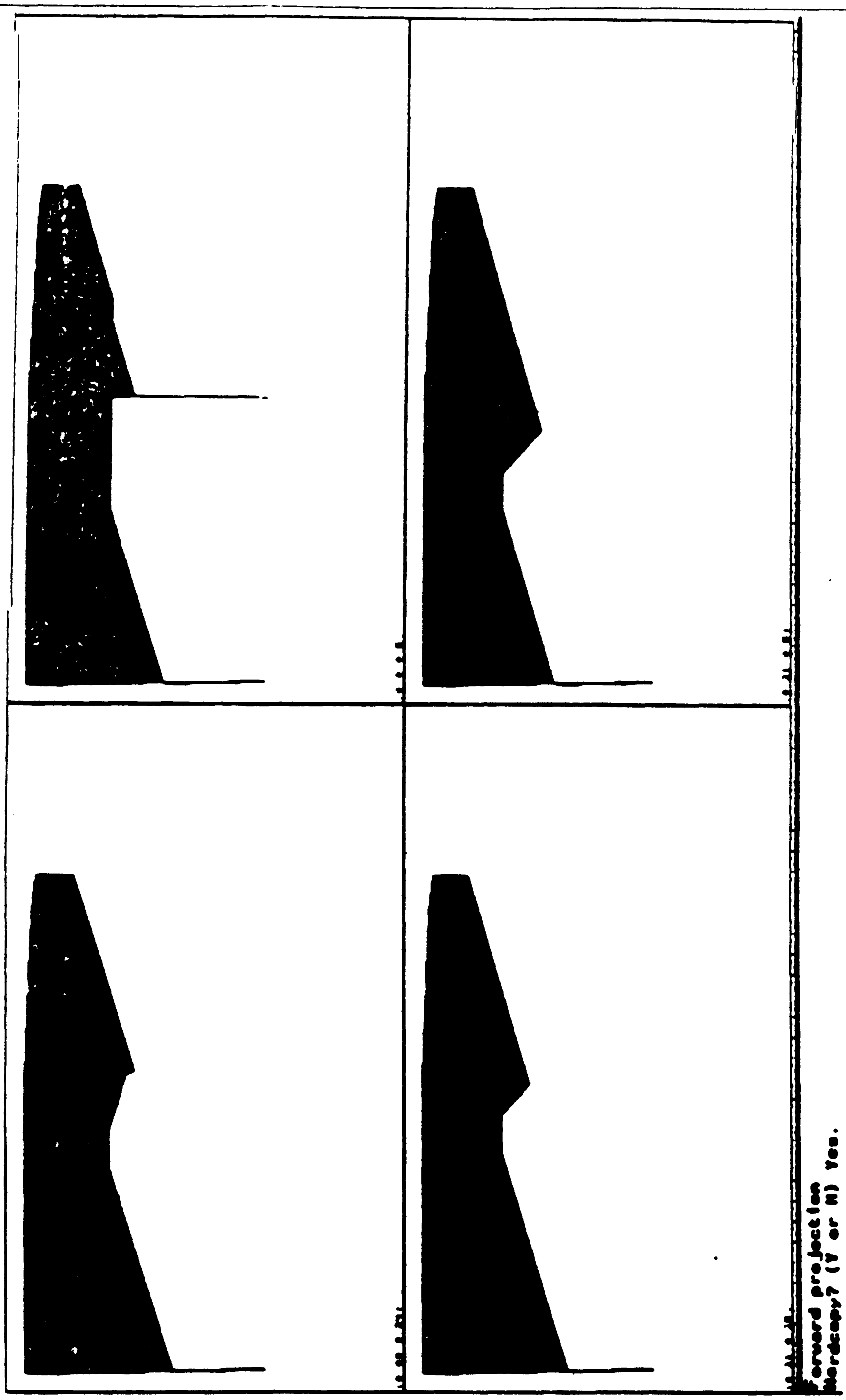

Figure 8: The forward projection of the first motion, shown without the obstacles.

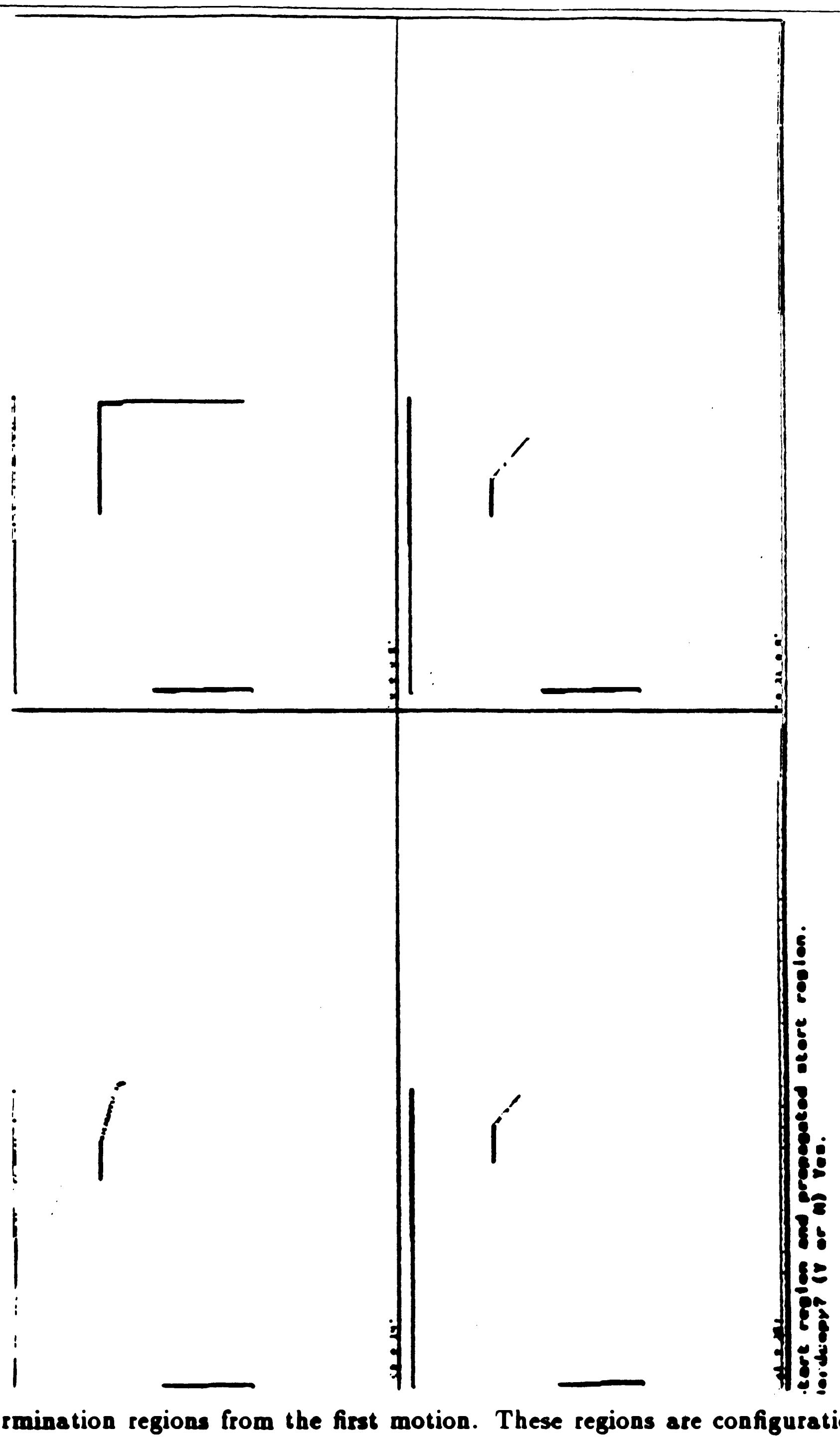

Figure 9: **The termination regions from the first motion. These regions are configurations where the motion finishes.**

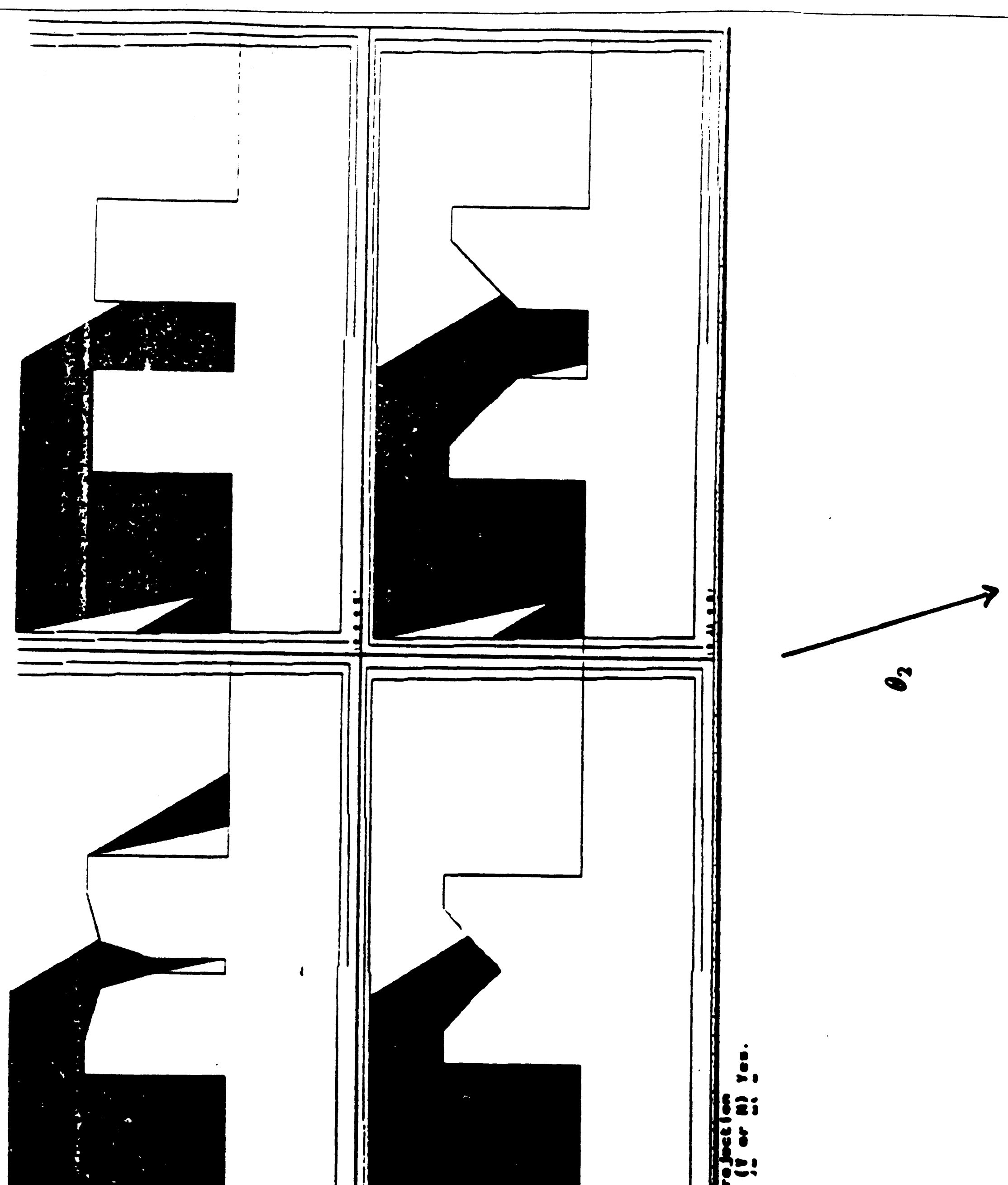

Figure 10: The forward projection of the second motion.

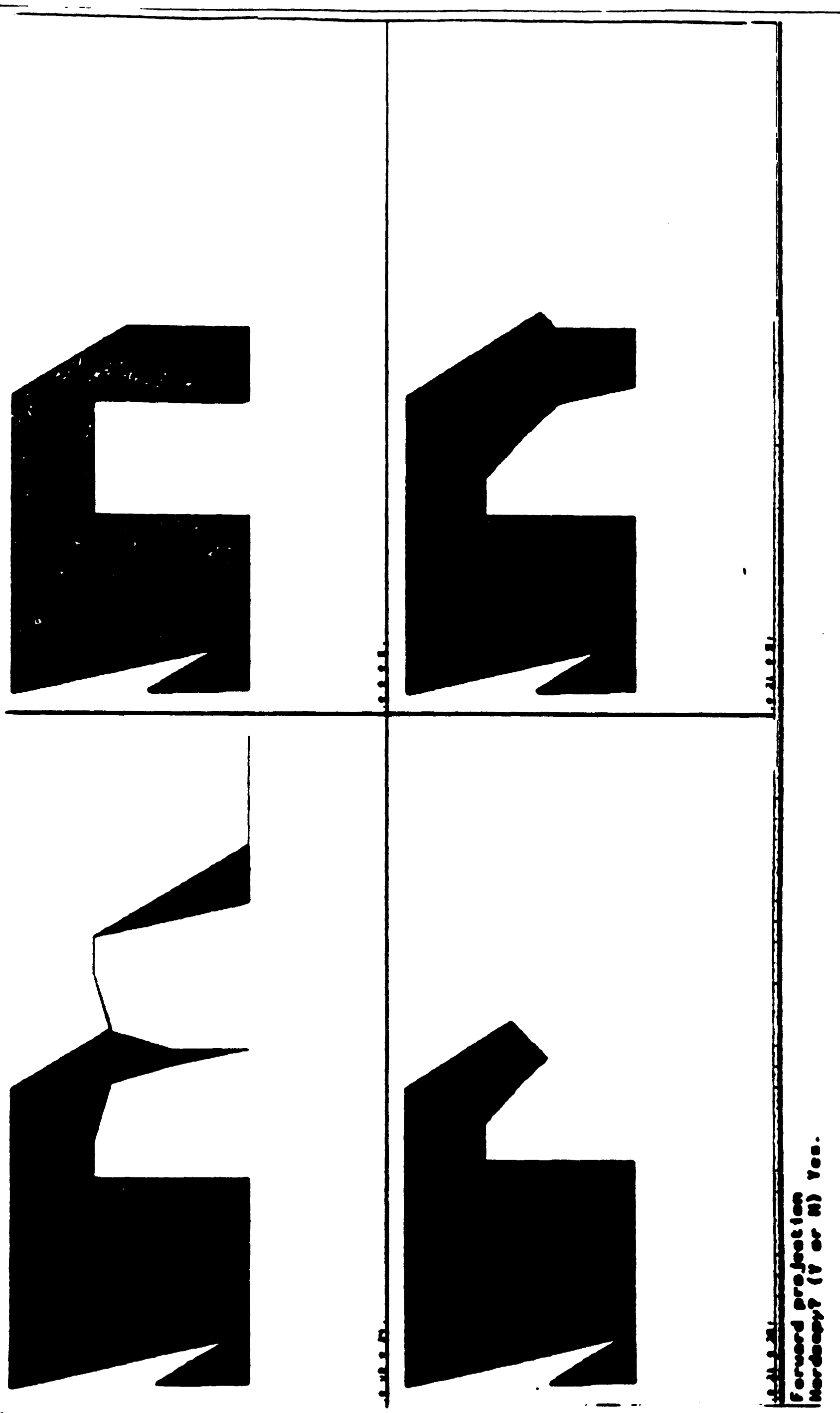

Figure 11: **The forward projection of the second motion, shown without the obstacles.**

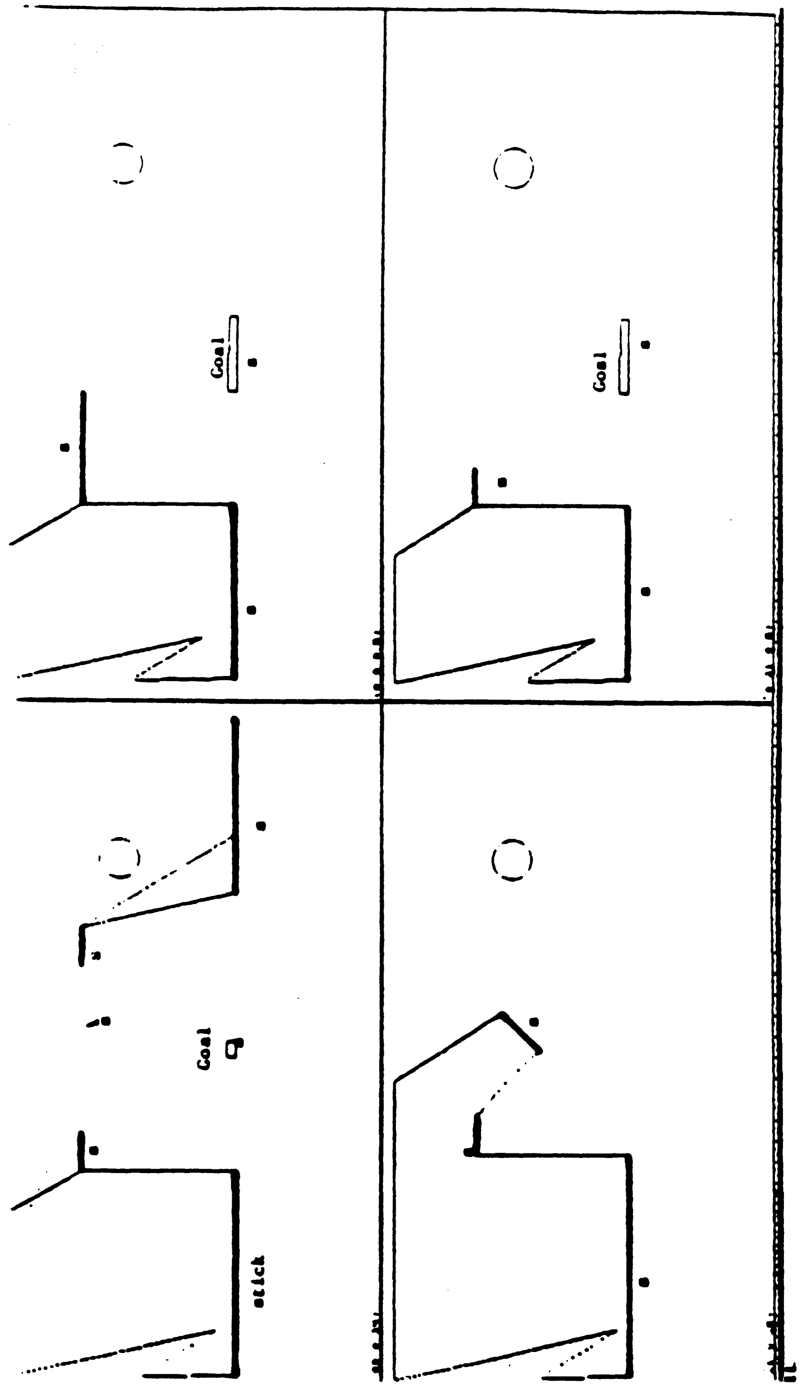

Figure 12: The termination regions for the second motion. These are edges in configuration space where sticking can occur.

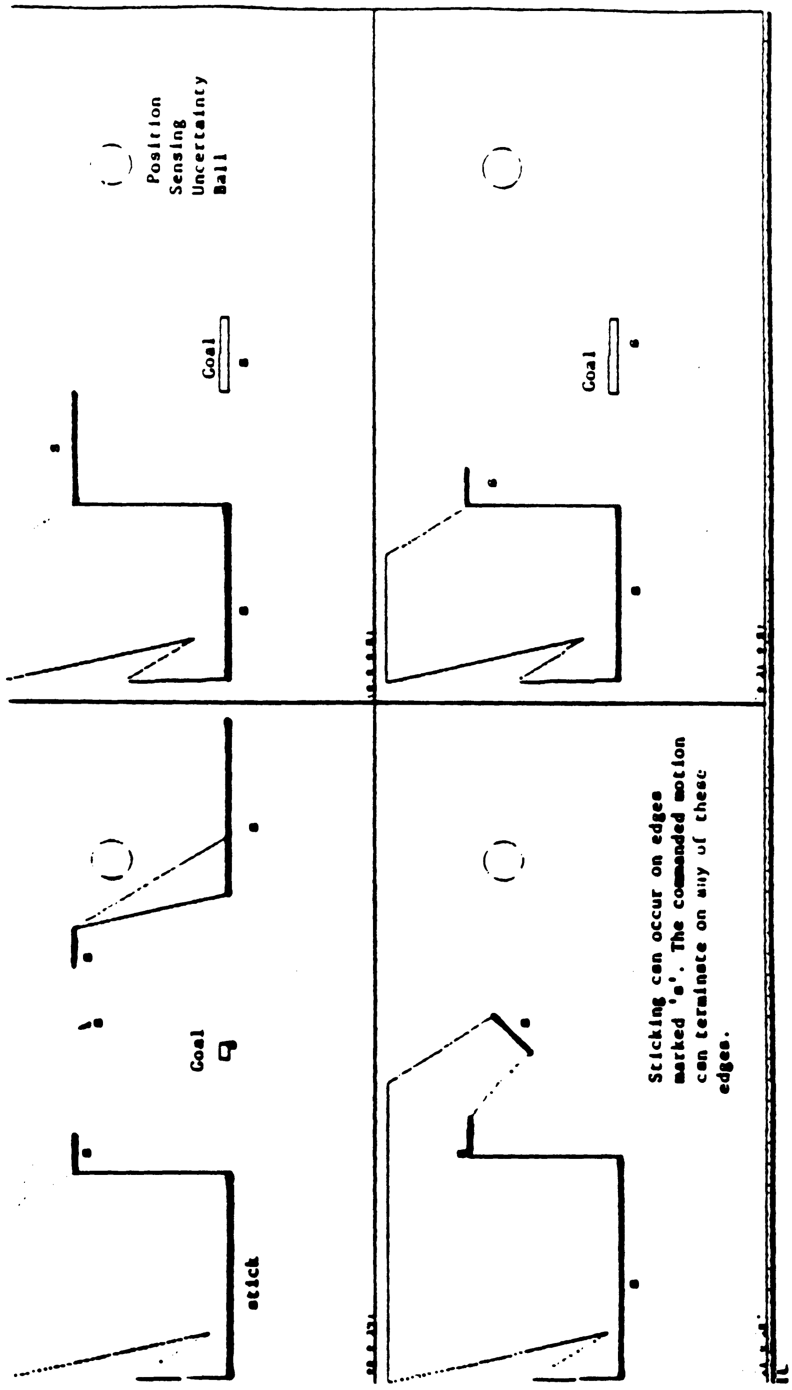

Figure 13: The failure regions and the goal are distinguishable, even given the sensing uncertainty. The disc indicates the magnitude of the position sensing uncertainty.

under sensing and control uncertainty. The [LMT] framework assumes no model error. In [D], we reduced the problem of planning guaranteed strategies with sensing, control, *and geometric model* uncertainty to the problem of computing preimages in a (higher dimensional) generalized configuration space. [3]

This result is useful in as much as there was really no prior theory of planning in an environment whose geometry is not precisely known. However, I do not think that it is the main point the EDR theory. This is because there are certain inadequacies with the planning model. The insistence that strategies be guaranteed to succeed is too restrictive in practice. To see this, observe that guaranteed strategies do not always exist. In the peg-in-hole problem with model error (figs. 3–13) there is no guaranteed strategy for achieving the goal, since the hole may be too small for some model error values. For these values the goal in configuration space does not exist. Because tolerances may cause gross topological changes in configuration space, this problem is particularly prevalent in the presence of model error. More generally, there may be model error values for which the goal may still exist, but it may not be reachable. For example, in a variant of the problem in fig. 3, an obstacle could block the channel to the goal. Then the goal is non-empty, but also not reachable. Finally, and most generally, there may be model error values 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 the peg-in-hole problem with model error, this technique will not work since no plan of any length can succeed when the hole closes up.

This is why we investigate EDR strategies, and, in particular, attempt to formalize EDR planning. The key theoretical issue is: How can we relax the restriction that plans must be guaranteed to succeed, and still retain a theory of planning that is not completely *ad hoc*? We attempt to answer this by giving a constructive definition of EDR strategies. In particular, this approach provides a formal test for verifying whether a given strategy is an EDR strategy. The test is formulated as a decision problem about projection sets in a generalized configuration space which also encodes model error. Roughly speaking, the projection sets represent all possible outcomes of a motion (the *forward projection*), and weakest preconditions for attaining a subgoal (the *preimage*).

Given the formal test for "recognizing" an EDR strategy, we then tested the definition by building a generate-and-test planner. The generator is trivial; the recognizer is an algorithmic embodiment of the formal test. It lies at the heart of this research. A second key component of the planner is a set of techniques for chaining together motions to synthesize multi-step strategies. The planner is a forward-chaining, multi-resolution planner, called LIMITED. LIMITED operates in a restricted domain. Plans found by LIMITED

[3] We use the terms model error and model uncertainty interchangeably.

in experiments are described above.

Finally, let me suggest that a new framework—the EDR framework— for planning with uncertainty may be justified not only by the restrictiveness of the guaranteed-success model, but also by the hardness of the problem. The gross motion planning problem without uncertainty may be viewed, under some very general assumptions, as a decision problem within the theory of real closed fields. This gives a theoretical decision procedure with polynomial running time once the degrees of freedom of the robot system are fixed [SS]. However, no such theoretical algorithm is known for the general compliant motion planning problem with uncertainty. Furthermore, the lower bounds for computing guaranteed strategies even in 3D are dismal: the problem is known to be hard for exponential time [CR]. At this point it is unknown whether EDR planning is more efficient than guaranteed planning. However, there is some experimental evidence leading one to conjecture that certain problems requiring very complicated, exponential-sized guaranteed plans may admit very short EDR plans.

However, the motivation for this work is not complexity-theoretic. Instead, the chief thrust is to show how to compute motion strategies under model error (and sensing and control uncertainty), using a formal and constructive definition of EDR strategies. The first goal was a precise geometric characterization of EDR planning—when one thinks about it, it is in fact somewhat surprising that such a thing should exist at all! But in fact it does, as we shall see. The second goal was to test this characterization by building a planner. Thus it was necessary to devise implementable algorithms to construct the geometric projection sets and decide questions about them. Therefore, this theory and LIMITED contain a mixture of precise combinatorial algorithms and of approximation algorithms. In [Donald, 87] we indicate which algorithms are exact and give combinatorial bounds. We also identify the approximation algorithms, and indicate the goodness of the approximation and whether it is conservative. Much work, or course, remains in developing better algorithms for EDR planning, and in testing out the plans using real robots.

2.2 Representing Model Error

We will review the EDR theory by examining some very simple planning problems with model error. Of course, this does not mean that EDR is limited to situations with model error.

2.2.1 A Simple Example: The Variable-Width Peg-In-Hole

Example (3). Consider fig. 14. 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.

This transformation reduces the planning task for a complicated moving object to navigating a point in configuration space. Consider fig. 15. The configuration point starts out in the region R, which is the position sensing uncertainty ball B_{ep} 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 commanded velocity v_0 is related to the effective velocity v by $f = B(v - v_0)$ where f is the effective force on the robot 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.[4]

The goal in fig. 15 is to move to the region G. Now, with Coulomb friction, sticking occurs on a surface when 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 negative α, B will emerge on the other side of A. In this case, the goal "reappears," and may be reachable again.[5] 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.

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. 15, 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. 15

[4]See [Mason 81] for a detailed description of generalized damper dynamics.

[5]This model is adopted for the purposes of exposition, not for physical plausibility. It is not hard to model the case where the blocks meet but do not fuse.

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. 3, C is again the cartesian plane, and J is a three-dimensional product space. One of the J dimensions is circular, to parameterize the angular variation represented by α_3.

In fig. 16 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. Note that the goal surface vanishes where A and B meet.

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 [Arnold]). 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[6] 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. 15. This is because motions cannot move between universes. In fig. 16, any legal motion in $\mathcal{G}$ is everywhere orthogonal to the J-axis and parallel to the x-y plane.

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. 15 is such a ball. We cannot sense across J: position sensing uncertainty is infinite in the J dimensions.[7] Thus the position sensing uncertainty in $\mathcal{G}$ is the cylinder $B_{ep} \times J$. In figs. 15,16, this simply says that x and y are known to some precision, while α is unknown. The initial position in fig. 15 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. 16.

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.

[6] α_0 is a point in the multi-dimensional space J.

[7] 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.

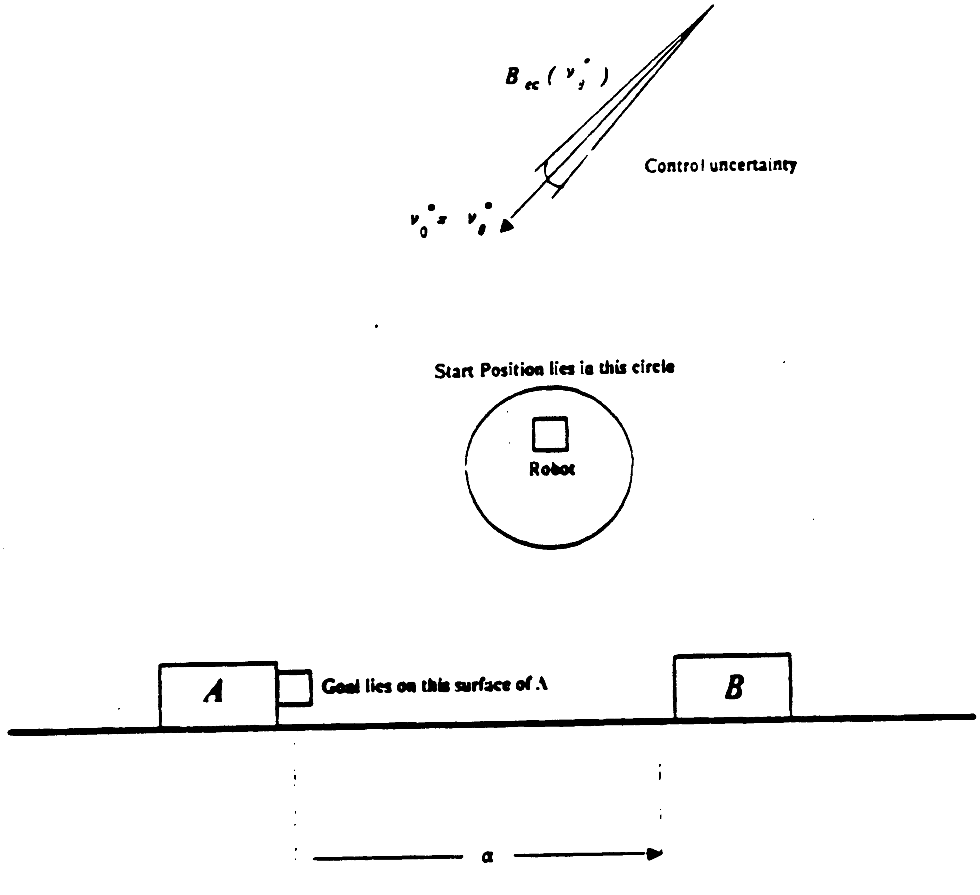

Figure 14: The goal is to bring the robot into contact with the right vertical surface of A. (For example, the "robot" could be a gripper finger). There is position sensing uncertainty, so in the start position the robot is only known to lie within some uncertainty ball. There is also control uncertainty in the commanded velocity to the robot. It is represented as a cone, as shown.

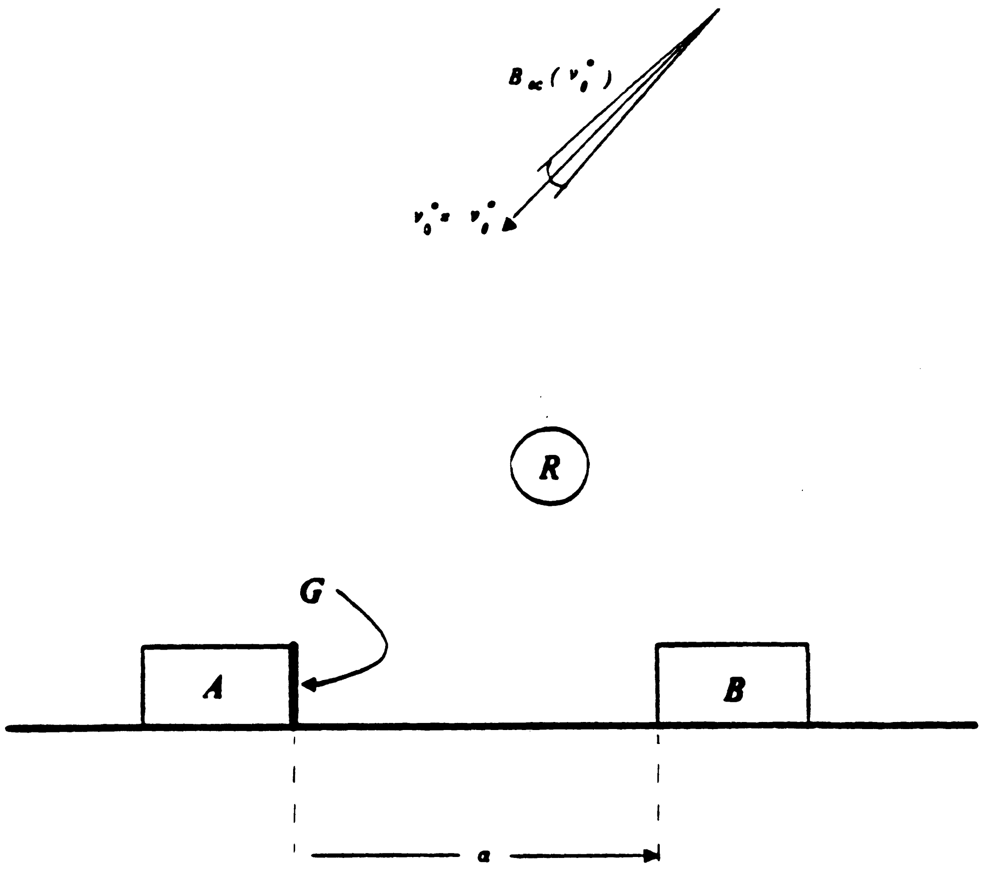

Figure 15: The equivalent problem in configuration space. The blocks A and B, the distance between the blocks a, and the commanded velocity $v_\theta = v_0^*$ with control error cone $B_{ec}(v_0^*)$. The position of A is fixed.

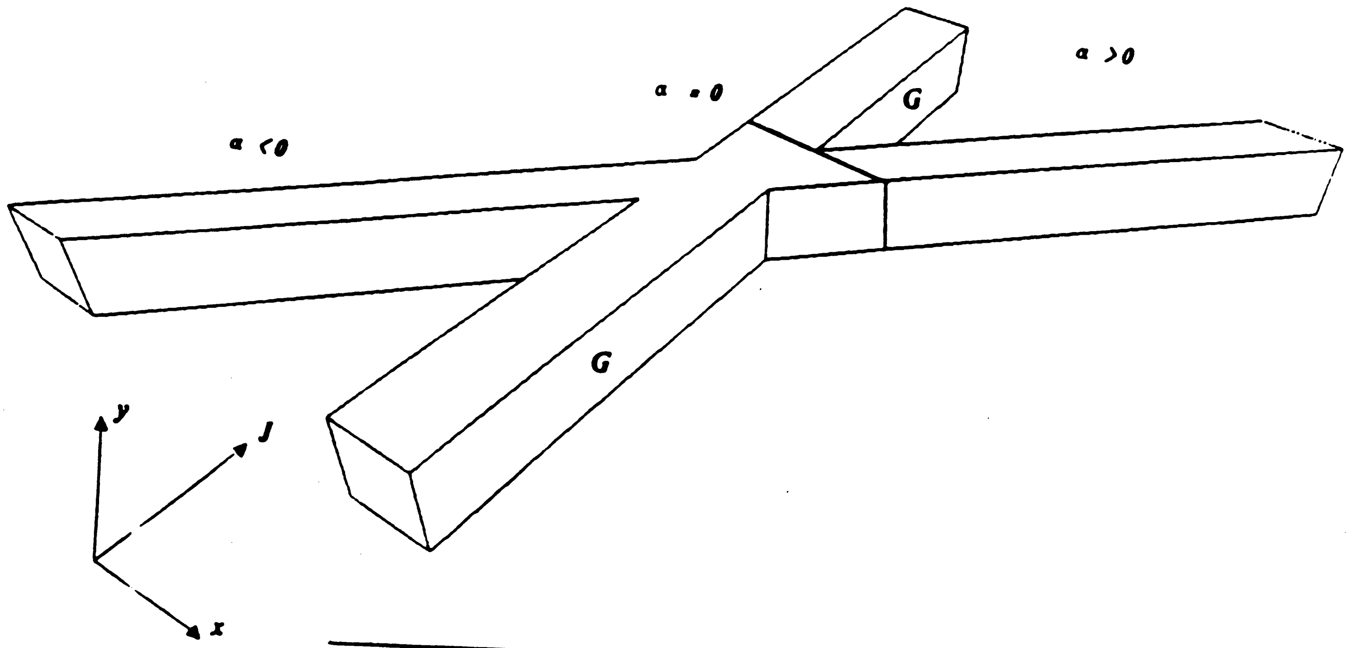

Figure 16: The generalised configuration space obstacles for example (1). The generalised configuration space is three dimensional, having x and y degrees of motion freedom, and an α degree of model error freedom. Legal motions are parallel to the x-y plane, and orthogonal to the J axis.

15. 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-forwardly 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 [Erdmann]) 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 in [Donald 86, 87]. 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 $\mathcal{G}$.

2.2.2 Pushing

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 (3). 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 (3), 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. 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, that is, a motion ***across α-slices of generalized configuration space***. Such a motion is always tangent to a surface in generalized configuration space.

In [D], we 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 can construct motion strategies whose primitives are gross motions, compliant motions, and pushing motions.* This model of pushing is used in the gear-meshing example, where a model error parameter—the orientation of B—can be changed due to pushing.

2.3 Guaranteed Plans

A motion strategy [LMT] is a commanded velocity (such as v_0^* in fig. 15) 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* [LMT]. 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. 17. The entry is recognized by monitoring the position and velocity sensors until the goal is attained. Fig. 17 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.

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. [LMT,E] provide a formal framework for computing preimages where there is sensing and control uncertainty, but no model error. In particular, [Erdmann] 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. 18 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. Furthermore, we assume that the robot has no sense of time (i.e., no clock)—for example, it might be equipped with a contact sensor that only fires once. 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 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. 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

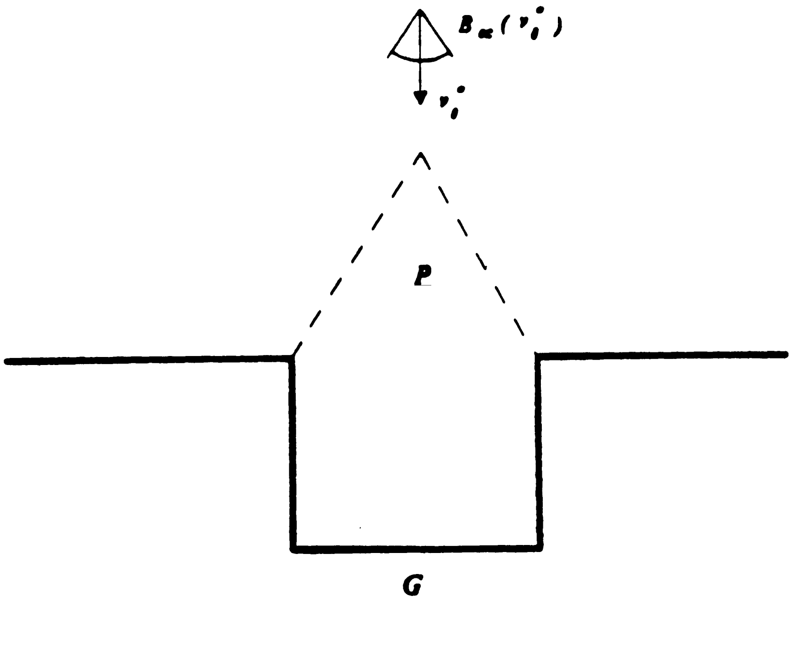

Figure 17: The goal is the region G. Sliding occurs on vertical surfaces, and sticking on horizontal ones. The commanded velocity is v_θ^*, and the control uncertainty is $B_{ec}(v_\theta^*)$. The *preimage* of the G with respect to θ is the region P.

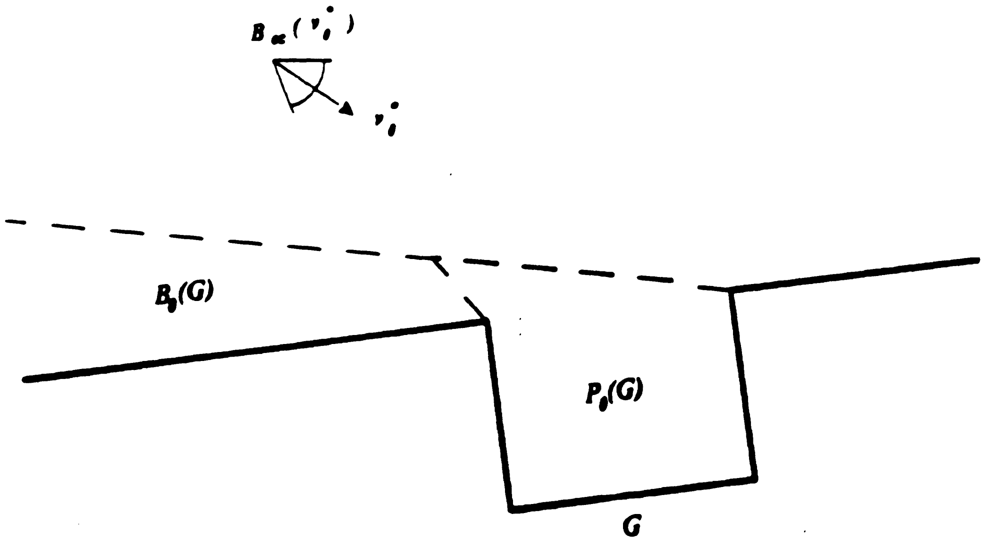

Figure 18: Here, the radius of the position sensing uncertainty ball is twice the width of the hole. Sliding occurs on all surfaces under the control velocities shown. The preimage of the goal under commanded velocity v_θ^* is $P_\theta(G)$. The backprojection $B_\theta(G)$ strictly contains this preimage: 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 recognised.

these uncertainties. The trick here is 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 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 [LMT,E] in configuration space generalize *mutatis mutandis* to $\mathcal{G}$ endowed with this physics; this is proved in [D]. 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. For details, see [Donald 86, 87, 88].

2.4 Error Detection and Recovery

If we were exclusively interested in constructing guaranteed motion strategies in the presence of model error, we would be done defining the framework: 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 [LMT,E] on computing preimages in general configuration spaces.

However, guaranteed strategies do not always exist. In example (3), (figs. 14–16) 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 the peg-in-hole problem with model error (figs. 3-13) 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 a variant of the problem in fig. 3, an obstacle could block the channel to the goal. 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. These are characterized as follows:

- 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. We now state how such EDR strategies may be constructed; for proofs and more detail, see [Donald 86, 87, 88].

Suppose that a planning problem is given with two disjoint geometrical goals, G_1 and G_2. We may insist that the run-time executor be able to terminate the strategy and also be able to disambiguate *which* goal has been reached. In this case, we can construct the *preimage of the distinguishable union of* G_1 *and* G_2, which we write as

$$P_\theta(\{ G_1, G_2 \}).$$

If θ is executed starting in this preimage, then the motion can always be recognizably and distinguishably terminated in either G_1 or G_2.

We can characterize EDR strategies geometrically as follows. Suppose the geometric goal is G. The implicit "meaning" of G is: recognizably achieving G is equivalent to "success." We introduce an "additional" goal-like set H, which is disjoint and distinguishable from G, such that when H is recognizably achieved, then failure of the motion may be signalled. That is, we construct an H such that recognizably achieving H is equivalent to "failure." H is called the *EDR region*. Remarkably, H may be selected such that the EDR axioms are satisfied. In [Donald 86, 87, 88], we derive H as follows. Given a motion θ and a start region R, first we define H using reachability constructs only. Then we test whether H and G are distinguishable using sensors (this is the "formal test" alluded to in the prelude). If so, then by the construction of H, we have

$$R \subset P_\theta(\{ G, H \}).$$

Furthermore, using H, θ is a one-step EDR strategy satisfying the EDR axioms. Here is an idea of what H is like: In fig. 19 the EDR region H is shown (in position space only) for example (3). Consider H as a two-dimensional region in $\mathcal{G}$; just a slice of it is shown in fig. 19. Note that in this example, H only exists in the slices in which G vanishes. Here, given the sensing uncertainty bounds of example (3), the termination predicate can distinguish between G and H based on position sensing, velocity sensing, or elapsed time. Clearly, H satisfies the EDR axioms: the motion is guaranteed to terminate recognizably in G iff the motion began in a universe in which G does not vanish. Otherwise, the motion terminates recognizably in H. In the first case, the termination predicate signals success, in the latter, failure.

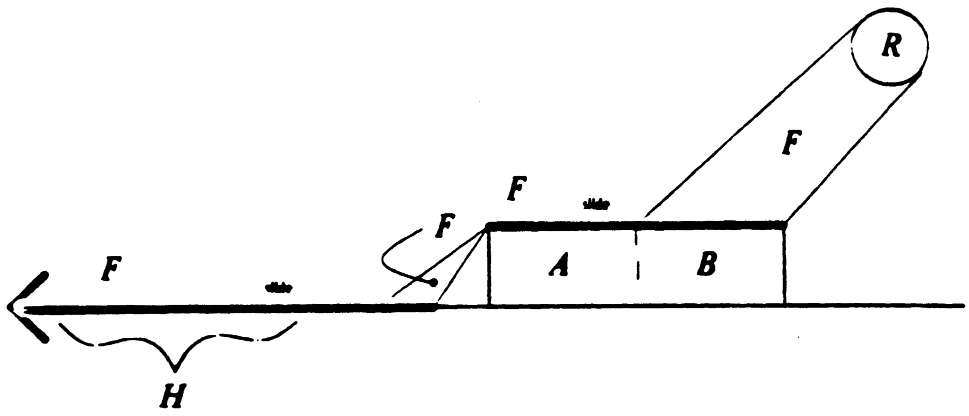

Figure 19: A typical α-slice of the EDR region H, for α small and negative. The goal vanishes in this slice; the dashed line indicates where the goal would be in other slices. Points in H lie within the forward projection (since they are reachable), yet outside the weak-preimage (since the goal is unachievable).

Here is how we construct H. The *forward projection* of a set R under θ is all configurations[8] which are possibly reachable from R under v_θ^* (subject to control uncertainty). It is denoted $F_\theta(R)$. Forward projections only address reachability: the termination predicate is ignored and only the control uncertainty bound and commanded velocity v_θ^* are needed to specify the forward projection. See figs. 8, 9, 11 12.

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 [LMT] (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. 20. We will use the weak preimage to capture the notion of serendipity in the EDR axioms. The idea is that a motion may be terminated in failure as soon as egress from the weak preimage is recognized. The weak preimage is denoted $\widehat{P}_\theta(G)$.

We define H_0 to be the set difference of the forward projection minus the weak preimage:

$$H_0 = F_\theta(R) - \widehat{P}_\theta(G).$$

Clearly, the motion θ can be terminated as a failure whenever H_0 has been reached, since H_0 is outside the weak preimage; hence the goal cannot be attained under θ from there.

We also define H_s to be all regions where sticking is possible in the weak minus strong preimage:

$$H_s = \{\, x \in \widehat{P}_\theta(G) - P_\theta(G) \mid \text{sticking is possible at } x \,\}.$$

See fig. 21. The motion θ should also be terminated as a failure if sticking occurs in H_s. We will decree that the robot has stuck in H_s if its velocity is zero for some duration, or "time-out" period. More precisely, we define H to be a set in phase-space. First, note that H contains all phase-space points (x, v) where x is in H_0. Second, H contains points of the the form $(x, 0)$, where x is in H_s. Thus, viewing phase-space as the tangent bundle to generalized configuration space, H contains the "cylinder" $\pi^{-1}(H_0)$ of velocities over H_0, and the "zero section" $Z(H_s)$ of zero velocities over H_s:

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

This definition of H almost satisfies the EDR axioms—the only tricky point is that we cannot guarantee that after sticking in H_s for a long time, the robot cannot eventually slide into the goal. This may be handled in principle by introducing a time-out period by which the goal must be reached. That is, our definition of H satisfies the EDR axioms if

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

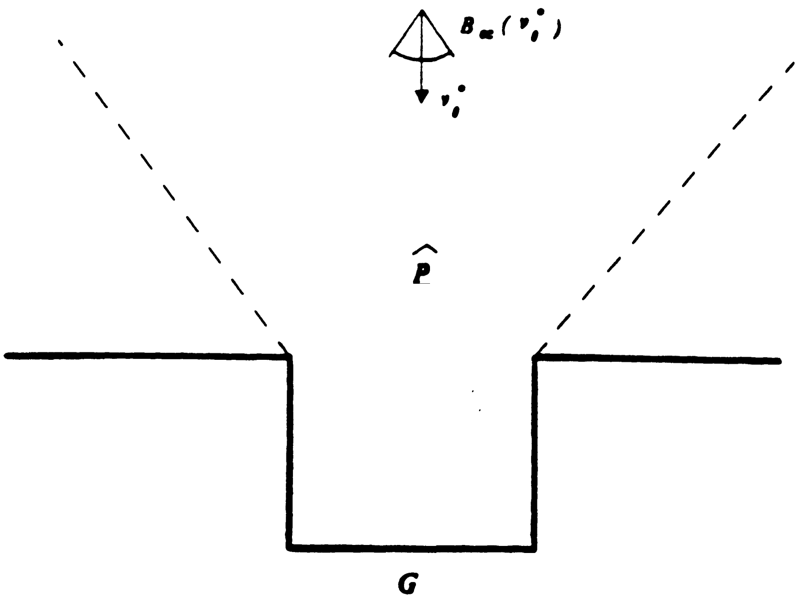

Figure 20: The *weak* preimage of the goal G under v_θ^*. Compare fig. 17.

Figure 21: 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."

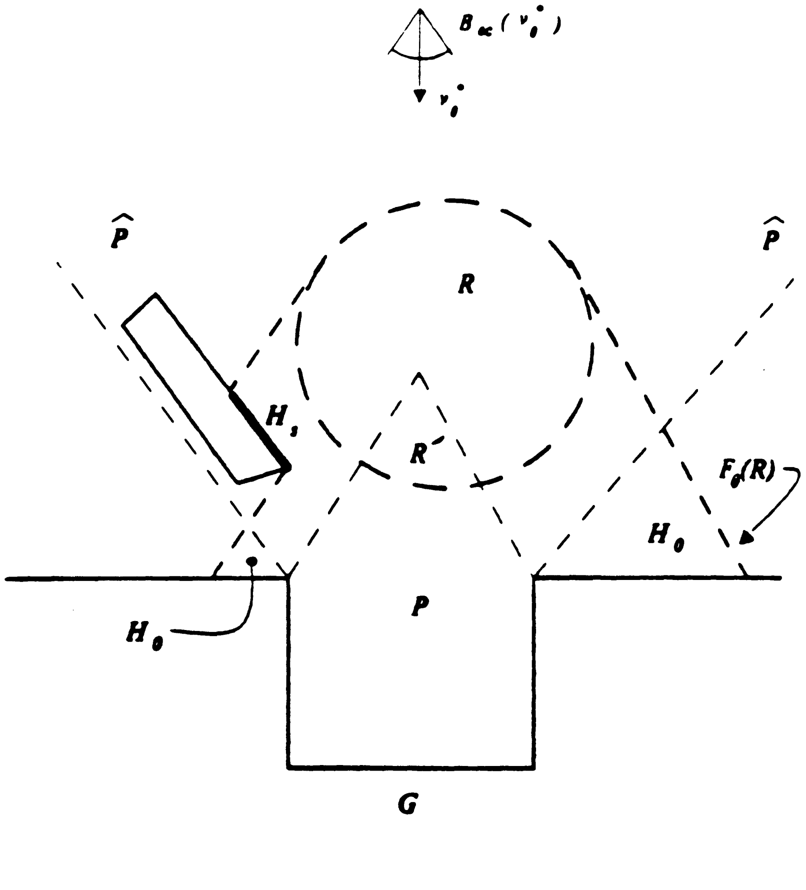

Figure 21. H_0 is not the entire EDR region. Sticking may occur within the weak preimage in H_s. The EDR region must include H_0 for all possible velocities, and H_s for "sticking velocities."

the goal is specified in phase-space-*time* as the product of $\pi^{-1}(G)$ with a compact time interval.

In [D], we showed how to compute H in the domain of planar assemblies with model error—a domain which included both the gear-meshing example (fig. 2) and the peg-in-hole problem with model error (figs. 3–13). We also showed how to compute whether G and H are distinguishable. This is sufficient to generate one-step EDR strategies. These algorithms have been implemented in LIMITED. At a high level, the one-step algorithm is:

Algorithm 1EDR

1. *Generate a commanded velocity v_θ^*.*
2. *Compute the EDR region H for v_θ^*,*
3. *Determine whether the EDR region H and the goal G are distinguishable using sensors. If so, then v_θ^* yields a one-step EDR strategy which recognizably terminates in G or H by monitoring position and force sensors.*
4. *Let* $\text{push}_\theta(G)$ *and* $\text{push}_\theta(H)$ *denote the* sticking push-forwards. *They are the set of obstacle edges within G and H, resp., on which sticking can occur under v_θ^*. Determine whether these regions are distinguishable using sensors. If so, then v_θ^* yields a one-step EDR strategy which recognizably terminates when sticking is detected.*

Here is how LIMITED decides the question, "Are G and H distinguishable using sensors?"

H and G are distinguishable using position sensing alone if their convolutions (Minkowski sums) by the position sensing error ball B_{ep} do not intersect.

Each obstacle edge of H and G has an associated configuration space friction cone. Two edges are distinguishable using force sensing if the convolutions of their friction cones by the force sensing uncertainty B_{ev} have a trivial intersection.[9]

Similarly, the set of possible sensed reaction forces at an obstacle vertex w of G or H may be found by taking the direct sum of the friction cones of the edges cobounding w, and convolving by B_{ev}. Again, a vertex of H and a vertex (or edge) of G are distinguishable using force sensing if their associated cones of sensed reaction forces have a trivial intersection.

The procedure also works for determining the distinguishability of the push-forwards. Note that the procedure is correct for linear edges, where position- and force-sensing are separable because the set of possible reaction forces is constant along an edge. For the general case, see [Donald, 87].

However, for both the gear-meshing and peg-in-hole problem, one-step strategies cannot be generated for the sensing and control error bounds we input. More specifically:

[9] An intersection containing only the zero-vector.

in these cases, H can be computed for any hypothesized one-step motion θ, but LIMITED cannot find a θ for which H is distinguishable from the goal. Intuitively, the reason that H and G are not distinguishable is that (a) since the control uncertainty is "large", therefore H is "large", and (b) since the error bounds on position and force sensing are large, therefore it is not always possible to distinguish between H and G.

What is to be done in this case? In short: multi-step strategies are required. This raises the central issue: how can multi-step EDR strategies be generated? We are interested both in an implementable (and, in LIMITED, implemented) practical approach, and also in developing unifying theory for multi-step EDR planning.

In this paper, we will assume that H can be computed given R and θ, and that we can test whether or not it is distinguishable from G. The reader interested in algorithms for these computations may find details in [Donald 86,87,88]. We now turn our attention to the generation of multi-step EDR strategies, given these tools.

2.5 The Preimage Structure of EDR Regions

Our notation for preimages must be made slightly more formal for the sequel. The key point [LMT,E,Mason] is that knowing where the motion began (that is, knowing R) can be used by the termination predicate to predict what configurations are reachable, and therefore, to effect recognizable termination. This prediction may be accomplished via the forward projection; its use as such was termed ***history*** by Erdmann. Thus, the preimage map P depends not only on θ and G, but also on R. Our notation for preimages must reflect this dependence.

We observed above 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

$$P_{\theta,R}(\{\, G, H \,\}) = R. \qquad (*)$$

We say that the preimage $(*)$ is taken ***with respect to R***. $(*)$ means 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 $P_{\theta,R}(G \cup H)$, 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 $(*)$ because the preimage depends on knowing where the motion started. This is a subtle point, see [LMT,E]. Thus solving preimage equations like $(*)$ for R is like finding the fixed point of a recursive equation. Here, however, we know R, H, and G, so $(*)$ is a constraint which must be true, rather than an equation to solve. Presumably $(*)$ is easier to check than to solve for R; see [LMT,E, D].

2.5.1 The Most General Preimage Equation

We now introduce the most general form of the preimage equation. Suppose $\{ G_\beta \}$ denotes a collection of goals, and $\{ R_\alpha \}$ is a collection of start regions. Recall θ denotes the direction of the commanded motion. Most generally, the preimage equation is

$$P_{\theta,\{ R_\alpha \}}(\{ G_\beta \}) = \{ R_\alpha \}.$$

This says that if the run-time executor knows that the robot is in some particular but arbitrary start region R in the collection $\{ R_\alpha \}$, then if velocity v_θ^* is commanded, then the termination predicate is guaranteed to achieve some goal G in $\{ G_\beta \}$, and, furthermore, it can recognize which goal has been achieved.

3 Multi-Step Strategies

In this section we explore multi-step strategy construction. Now, in principle, having reduced both model error and EDR to essentially "preimage-theoretic" equations, multi-step strategies could be synthesized by solving these preimage equations. While this is proved or at least implicit in previous work [LMT,Mason,E,D], it is far from obvious; furthermore, there are almost no published examples of such strategies. For this reason we begin by presenting a worked-out example of a motion plan using preimages. The motion problem is grasp-centering for a robot gripper in the presence of model error. Both guaranteed and EDR strategies are found by solving the preimage equations.

Preimages are a key underlying tool for the geometric EDR theory, and the [LMT] framework is in some sense a "universal" method for synthesizing multi-step strategies. However, the technique of solving the preimage equations is not computational. For this reason, we introduce a construction called the ***push-forward***. Roughly speaking, the push-forward is that subset of the forward projection where the motion can terminate. Since push-forwards address termination whereas forward projections do not, we may regard them as "dual" to preimages. That is, push-forwards are to forward projections as preimages are to backprojections. Second, the push-forward permits us to develop rather simple algorithms for planning multi-step strategies. These algorithms have been implemented in LIMITED. While the push-forward method for multi-step strategy synthesis is algorithmic, it is less general than the full preimage method (solving the preimage equations). We characterize the loss of power in push-forward algorithms.

In section 1 we presented two EDR plans generated by LIMITED. These were the peg-in-hole insertion strategy with model error, and the gear-meshing plan. Both were two-step plans. We will go into more detail in describing how these plans were generated. The peg-in-hole plan used push-forward techniques. The gear plan used a seemingly unrelated technique called *failure mode analysis*. We describe failure mode analysis and algorithms for computing it.

Next, we will present a view of multi-step strategies which essentially unifies all these techniques. This is called the "weak" EDR theory. The motivation behind this theory is that when a motion terminates ambiguously, a subsequent motion may be synthesized which disambiguates the success or failure of the first. Oddly enough, it is not necessary for either motion individually to satisfy the EDR axioms. However, when taken together, the two-motion plan can often be considered "equivalent" to a one-step EDR strategy.

The weak EDR theory effectively defines some laws of "composition" that permit two single-step plans to be concatenated into a two-step plan satisfying the EDR axioms. Hence it is often possible to construct multi-step plans that are EDR plans "globally" although not "locally". That is, considered as entire plans, they satisfy the EDR axioms; this is the "global" condition. However, "locally" they are not EDR plans, in that no single step is an EDR strategy. The key to pasting together non-EDR plans to make a global EDR strategy lies in defining certain local "niceness" conditions for how plans must mesh. These are called the *linking conditions*.

Starred sections may be skipped if desired at first reading.

4 Planning using Preimages: A Detailed Example

In this section we show how the [LMT] framework can be used to synthesize multi-step strategies. Here are the key points of this section:

- In principle, multi-step plans may be found by solving a family of preimage equations.
- While this was proved by [LMT,Mason,E], it is not obvious how to effect the solution. This example intends to elucidate the process.
- The technique is general enough to plan EDR strategies under model error, once we have cast both the problem of planning with model error and the EDR problem in an essentially "preimage-theoretic" form, as in [D] and section 1.
- However, the technique of solving the preimage equations is not algorithmic.

Furthermore, preimages are a key underlying tool for the geometric EDR theory. It is necessary to make further acquaintance with preimages in order to continue our development of the EDR framework. To that end, this section presents a worked-out example of a motion plan using preimages. The motion problem is grasp-centering for a robot gripper in the presence of model error. The example illustrates the use of the preimage framework to derive a multi-step motion strategy in the presence of model error. The strategy employs time-sensing and force-sensing. This discussion is designed both as a tutorial in solving preimage equations for a motion plan, and as an introduction to the planning of multi-step strategies.

4.1 Example: Planning Grasp-Centering using Preimages

The remainder of this paper builds on the preimage framework to develop the EDR theory. To make the framework more accessible, we provide here a fairly detailed description of a motion planning problem using preimages.[10]

We are now ready to work an example. We solve a particular motion planning problem with model error by solving the preimage equations. This example provides an illustration of planning using preimages. For simplicity, we initially address only the problem of finding a guaranteed strategy. Finding EDR strategies in this domain is discussed afterwards.

Consider the grasp-centering problem shown in fig. 22. The task is to center the robot gripper over the block D. The gripper can translate but not rotate in the plane. In its start position, the gripper is somewhere over D, such that the bottom of the fingers FA and FB are below the top of D. The width of D is unknown, but must be less than the distance between FA and FB. We assume D is fixed (it cannot be accidentally pushed).

Hence we can regard this as a planning problem with model error. C is taken to be the cartesian plane, and J is a bounded interval of the positive reals. Our first question is, what does the generalized configuration space look like? This is easily answered by considering the motion planning problem in fig. 23. The problem is to find a motion strategy for a point robot so that it can achieve a goal exactly halfway between the blocks A and B. The distance α between A and B is unknown and positive. The point robot is known to start between A and B. Again, the point can translate in the plane. The distance α is the model error parameter. It is easy to see that the problems in figs. 22 and 23 are equivalent.

However, we already know what the generalized configuration space for fig. 23 looks like. It was discussed in section 1, and is shown in fig. 16. Hence our example is a planning problem in a familiar generalized configuration space.

Next, we assume that the robot has perfect control, perfect velocity sensing, and a perfectly accurate sense of time. However, it has infinite position sensing error. [11]

Now, since the gripper starts over D with the bottom of the fingers below the top of D, and since the robot has perfect control, it suffices to consider the x axis of C. Since

[10] This problem arose in discussions with Tomás Lozano-Pérez, John Canny, and Mike Erdmann.

[11] This example is easily generalized to non-zero control, time-sensing, and force-sensing error, and finite position-sensing error. This requires giving the goal non-empty interior, however.

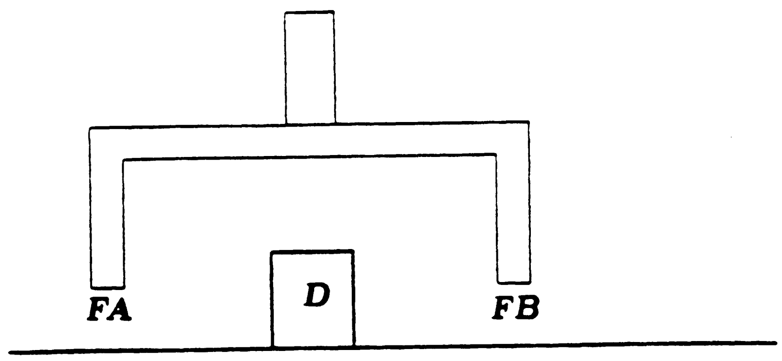

Figure 22: **The grasp centering problem. The width of the block *D* on the table, and the position of the gripper are only known approximately.**

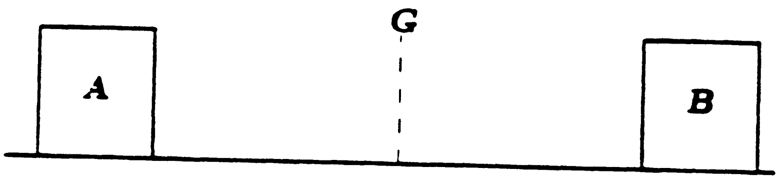

Figure 23: An equivalent problem. A point robot must be navigated halfway between the blocks A and B. The distance between A and B is not known. The robot has force sensing, and a clock. However, it has poor position sensing. We regard C as $\Re^2$ and J as the bounded interval $(0, d]$ for d positive. The generalized configuration space for this problem is the same as in fig. 16, for the positive values in J.

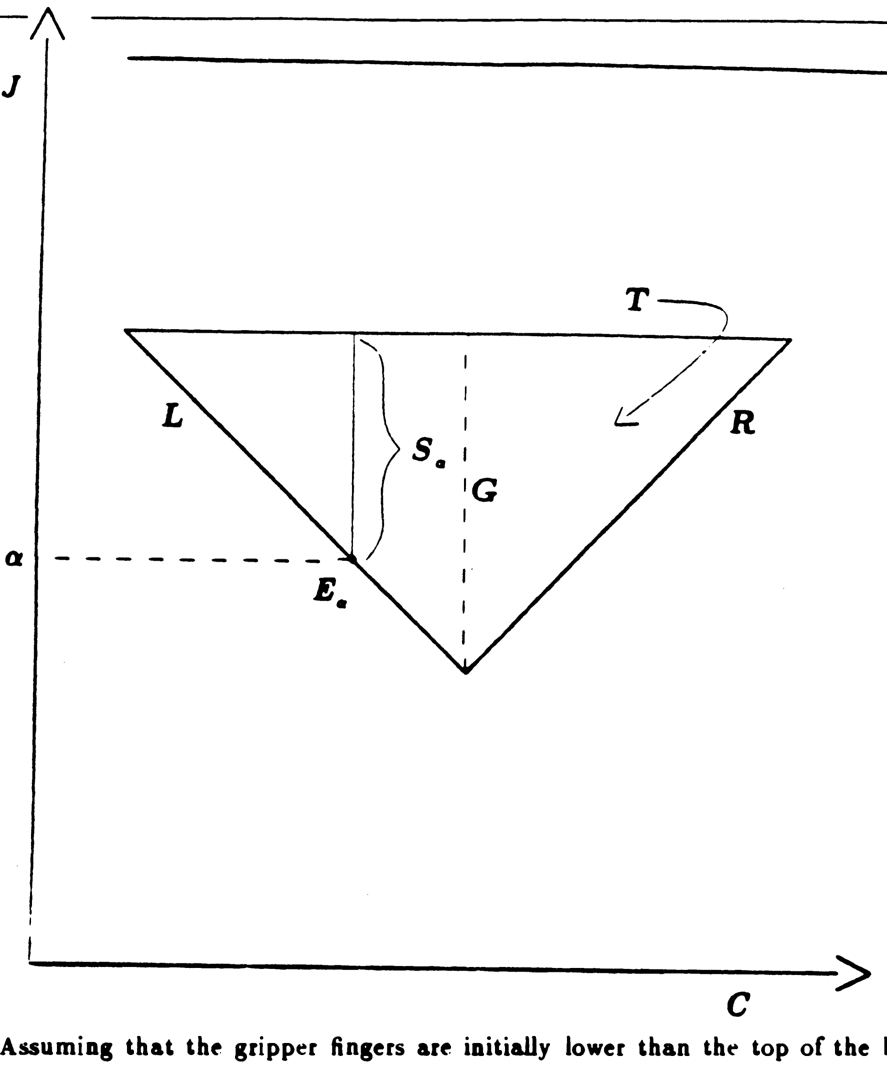

Figure 24: Assuming that the gripper fingers are initially lower than the top of the block D, the y dimension can effectively be ignored. This allows us to examine a cross-section of fig. 16. We treat C as the x axis of motion freedom, yielding a 2D $C \times J$ planning space. L and R are obstacle boundaries in generalised configuration space . The goal is the bisector G between L and R in free-space. The start region T is the triangular region between L and R. E_α is a point on L. S_α is a line in T parallel to G and containing S_α.

---

the y axis can be ignored, we develop our example in the plane, that is, in the generalized configuration space where C and J are both one-dimensional. This 2D generalized configuration space is shown in fig. 24, which is essentially an x-J cross-section of fig. 16, holding y constant with α constrained to be positive. In fig. 24, L and R are left and right obstacle edge boundaries generated by A and B. The goal is the line in free-space bisecting L and R. The start region T is the triangular region in free-space between L and R. (T is the convex hull of L and R).

Now, since motion across J is not permitted, all motions are parallel to the x axis, that is to say, horizontal in fig. 24. There are only two kinds of motions the planner can command. Let $+$ denote a motion to the right, and $-$ a motion to the left. We assume the robot has perfect control over the magnitude as well as the direction of the commanded velocity.

See fig. 24. Now, if α is a point on the J axis, let E_α be the point on the left obstacle edge L with J coordinate α. We will denote the collection of all such points on L by $\{ E_\alpha \}$. Let S_α denote the maximal line segment within T containing E_α and parallel to G. Formally, if E_α has coordinates (x, α), then S_α is the line segment extending from E_α to (x, d) where d is an upper bound on the distance between A and B. We denote the collection of all lines S_α by $\{ S_\alpha \}$.

At this point we are prepared to derive a motion strategy for centering the grasp, that is, for attaining G from T. The strategy has three steps. The termination conditions for the motions involve time- and force-sensing. Here is the motion strategy in qualitative terms:

Strategy Guarantee-Center

1. *Command a motion to the right. Terminate on the right edge R based on force sensing.*

2. *Command a velocity of known magnitude to the left. Terminate when in contact with the left edge L, using force sensing. Measure the elapsed time of the motion. Compute the distance traversed. This gives exact knowledge of where the motion terminated on L. The effect of this step is to measure the distance α between the blocks.*

3. *Move distance $\frac{\alpha}{2}$ to the right, terminating in G based on time sensing.*

We now derive this strategy by solving the preimage equations for the motion planning problem.

First, note that if the run-time executive knows that the robot is inside a *particular* S_α, then G can be reliably achieved by commanding a motion to the right. Since the robot has perfect control and time sensing, the motion can be terminated after moving distance $\frac{a}{2}$, that is, exactly when the line G is achieved. Using the preimage notation, we write this as

$$P_{+,\{S_\alpha\}}(G) = \{ S_\alpha \}. \tag{1}$$

Next, we take the collection $\{ S_\alpha \}$ as a set of subgoals, and try to find a motion that can recognizably attain this collection, and, furthermore, can distinguish which S_α the motion achieves. Consider a leftward motion starting from anywhere on the right edge R. The robot does not know where on R the motion starts, however. To recognizably achieve some S_α, such a motion should move leftward, and terminate when force-sensing indicates that L has been reached. If the termination predicate measures the elapsed time of the motion, and knows the magnitude of the commanded velocity, then it can recognize which point E_α has been reached, and hence which subgoal S_α has been achieved. Writing this down in preimage equations,

$$P_{-,R}(\{ S_\alpha \}) = P_{-,R}(\{ E_\alpha \}) = R. \tag{2}$$

Finally, the right edge R may be achieved from anywhere within the start region T by moving rightward, and terminating when force sensing indicates contact. This is simply

$$P_{+,T}(R) = T. \tag{3}$$

It is instructive to examine the termination conditions for motions (1)–(3). In developing the [LMT] framework for planning guaranteed strategies, [Erdmann] developed an elegant formalization of the question, "Using sensors and history, when can the termination predicate decide that a motion has recognizably entered a goal G_β?" The answer was as follows. Assume that G_β has been lifted into phase-space. Let R be the start region. The forward projection, $F_\theta(R)$ captures the notion of history: it is all positions and velocities that can be reached given that the motion started in R. At a particular instant t in time, let $B_{ep}(t)$ and $B_{ev}(t)$ be the sets of possible positions and velocities. These are the sensing uncertainty balls about a sensed position and velocity in phase space at time t. Thus sensing provides the information that the actual position and velocity must lie within the set $B_{ep}(t) \times B_{ev}(t)$. The forward projection further constrains the actual position and velocity to lie within $F_\theta(R)$. Thus the termination predicate can terminate the motion as having recognizably reached G_β when

$$F_\theta(R) \cap (B_{ep}(t) \times B_{ev}(t)) \subset G_\beta. \tag{$*$}$$

Now, let $F_\theta(R,t)$ denote the ***time-indexed*** or ***instantaneous forward projection*** of R under θ at time t. $F_\theta(R,t)$ denotes the set of positions and velocities that are possibly achievable at elapsed time t, under motion θ, given that the motion started in R. The termination predicate in this case monitors a clock, in addition to position and velocity sensors. In motion (1), only the time-indexed forward projection $F_+(S_\alpha, t)$ is relevant to deciding termination. The motion terminates when $F_+(S_\alpha, t) \subset G$. Motion (3) can be terminated using pure force sensing. It could also be terminated using time, since there exists some t for which $F_+(T, t) = R$. In motion (2), both force sensing and time are required to terminate within a distinguishable E_α. The general form of the termination condition for all three cases is as follows. The termination predicate has the form

$$F_\theta(U, t) \cap (B_{ep}(t) \times B_{ev}(t)) \subset G_\beta$$

for a goal G_β and a start region U. (Assume that all subgoals have been lifted into phase space). In our case, position sensing error is infinite, so $B_{ep}(t)$ is $C \times J$. Let us denote $(C \times J) \times B_{ev}(t)$ by the simpler expression $B_v(t)$. Then the termination conditions for motions (1)–(3) are as follows. For the first motion (3), to terminate, we must have

$$F_+(T, t) \cap B_v(t) \subset R. \tag{4}$$

For the second motion (2) to terminate, we must have

$$F_-(R, t) \cap B_v(t) \subset S_\alpha \tag{5}$$

for some S_α. We think of the termination predicate as "returning" this S_α. Finally, for termination of the last motion (1), we must have

$$F_+(S_\alpha) \cap B_v(t) \subset G, \tag{6}$$

where the S_α in (6) is the same as the one returned by the termination predicate after the second motion as the satisfying assignment for (5).

Finally, note that time is the source of some complexity in this example. This complexity might be removed by employing a distance sensor instead. The output of such a sensor could be modeled as position sensing in J. The sensing action in J would entail measuring the distance between A and B. This relaxes the assumption of no position sensing in the J dimensions, but such modification to the generalized configuration space framework is trivial. With this modification, B_{ep} is simply regarded as a product of a position sensing ball in C and a position sensing set in J.

This concludes the example. We have shown how to derive a multi-step guaranteed motion strategy in the presence of model error. The strategy was derived by solving the preimage equations in generalized configuration space for the motion plan. These preimage equations made the role of time- and force-sensing explicit in deriving conditions for distinguishable termination in a collection of subgoals.

4.1.1 An EDR Strategy for Grasp-Centering

We now generalize the grasp-center example and show how to develop an EDR strategy for this problem.

Assume that the radius of position sensing uncertainty is larger than the diameter of T, but not infinite.[12] Furthermore, assume that α, the distance between A and B, can be zero (but not negative) in the above example. That is, D can be too big to grasp. Hence the hole between A and B can close up, as in fig. 16. Assume that the gripper starts above the height of the block D, in the circular region R in fig. 15. Generalize the discussion of preimages above to describe an EDR strategy using preimages. We will need to consider the y dimension of motion freedom as well, in the 3D generalized configuration space shown in fig. 16, but only the non-negative α in J. Note that EDR is "required" here, since if α can be zero, there exists no guaranteed strategy.

Let us rename the circular start region in fig. 15 to be U, and continue to use R for the right edge in fig. 24. Assume that the x-J slice of generalized configuration space in fig. 24 is taken at $y = 0$, i.e., at the level of the table, and that under the commanded motion v_θ^*, shown in fig. 15, sliding occurs on all horizontal and vertical surfaces. However, clearly sticking will occur under v_θ^* on the concave left edge L between A and the table.

Now, let H be as in fig. 19. Here is the EDR strategy in qualitative terms:

Strategy EDR-Center

E1. *From U, command the motion v_θ^*. Terminate on the left edge L based on sticking, or in H based on time.*

E2. *If H is attained, signal failure. Otherwise, go to step (1) of strategy* Guarantee-Center.

Now, since $H_s = \emptyset$, the preimage equation (3a) for step (E1) simply reduces to

$$P_{\theta,U}(\{ L, H \}) = U. \tag{7}$$

At this point, the remainder of the strategy may be developed in the x-J slice shown in fig. 24. To finish the preimage characterization of the EDR strategy, we must replace eq. (3), which characterizes the first step (1) of strategy *Guarantee-Center*, by

$$P_{+,L}(R) = L. \tag{8}$$

Note that (8) is actually a logical consequence of (3), since L is a subset of T. Analogously, (4) must be changed by replacing T by L. The remainder of the preimage equations (1)–(2) and (5)–(6) remain unchanged.

[12]This assumption is not necessary, but it simplifies our discussion somewhat.

4.2 Solving the Preimage Equations is General but Not Computational

This example shows how multi-step EDR strategies under model error can be generated by solving a family of preimage equations. However, the technique is not an algorithm. We do not claim that such an algorithm could not be developed, but merely that as described above and in [LMT,Mason,E], the method is not (yet) computationally effective.[13] The first reason it is non-computational is that the number of subgoals $\{ E_\alpha \}$ and $\{ S_\alpha \}$ is infinite. The second, and more important reason is that solving the preimage equation is, as stated, a decision problem in second-order set theory. Even if the sets are, say, algebraic, this theory is undecidable. However, there may exist a reformulation of the problem rendering it decidable. Below we describe one such reformulation, using push-forwards, which can be used in effect to solve certain "simple" preimage equations and hence to generate a restricted class of EDR plans.

5 Push-Forwards: A Simple Generalization to n-Step EDR Strategies

The generalized preimage framework [LMT,Mason,E,D] gives a kind of "universal" method for generating multi-step EDR strategies. However, the technique of solving the preimage equations is not algorithmic—it is more like doing a proof by hand. For this reason, we introduce the ***push-forward*** technique for synthesizing multi-step strategies. While considerably less general than solving the full preimage equations, it leads to rather simple multi-step strategy-generation algorithms, which were implemented in LIMITED. The push-forward technique is powerful enough to generate an EDR plan for the peg-in-hole insertion strategy with model error described in section 1. However, it is not general enough to solve all steps of the grasp-centering example discussed above. This gives us a measure of the relative power of push-forward vs. preimage equation techniques.

We first introduce the "Twin Universe" example (4).

5.0.1 The "Twin Universe" Example (4)

Consider fig. 25. 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

[13] However, note that Erdmann's techniques of approximating preimages by backprojections may lead toward a fully-algorithmic method.

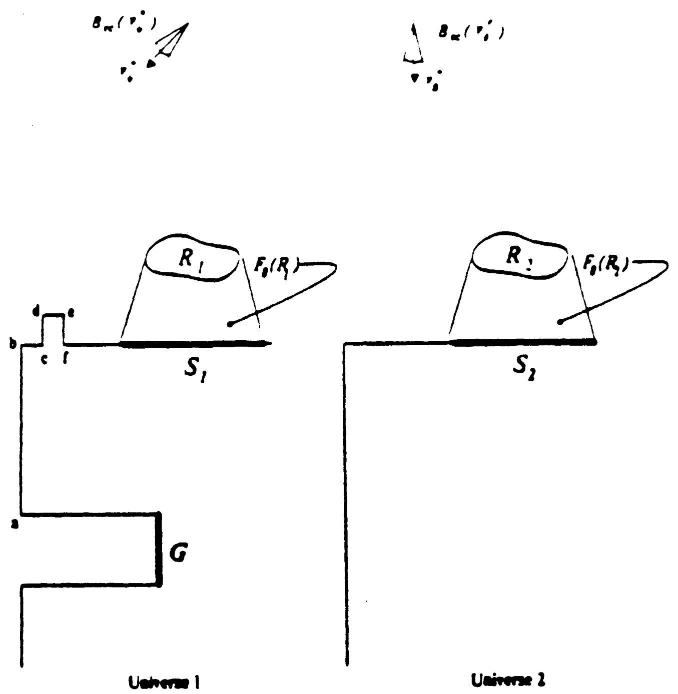

Figure 25: There are two possible universes; the goal G exists in the first but not the second. The start region is $R_1 \cup R_2$. Motion θ is guaranteed to move from R_1 into S_1. Motion ψ is guaranteed to move from S_1 into f. There is an 8-step plan achieving G from R_1. The forward projections of R_1 and R_2 are indistinguishable. There exists no one-step EDR strategy from the motion θ.

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. 25? 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 is then simply the (cylinder over the) forward projection of the "bad" region, $F_\theta(R_2)$. (See fig. 25). 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. 25), and is guaranteed to achieve the vertex subgoal f from start region S_1. We will 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 "southwest" motion ψ, and the preimage of f under ψ.[14] The EDR region for these data is the forward projection of S_2 under ψ (see fig. 26). Presumably this EDR region is (eventually) 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.

5.1 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.[15] A particular type of push-forward is defined formally in [D]; we describe it informally below. In example (4), the push-forward

[14] While S_1 is the preimage of f 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. See secs. 2.5 and 2.4).

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

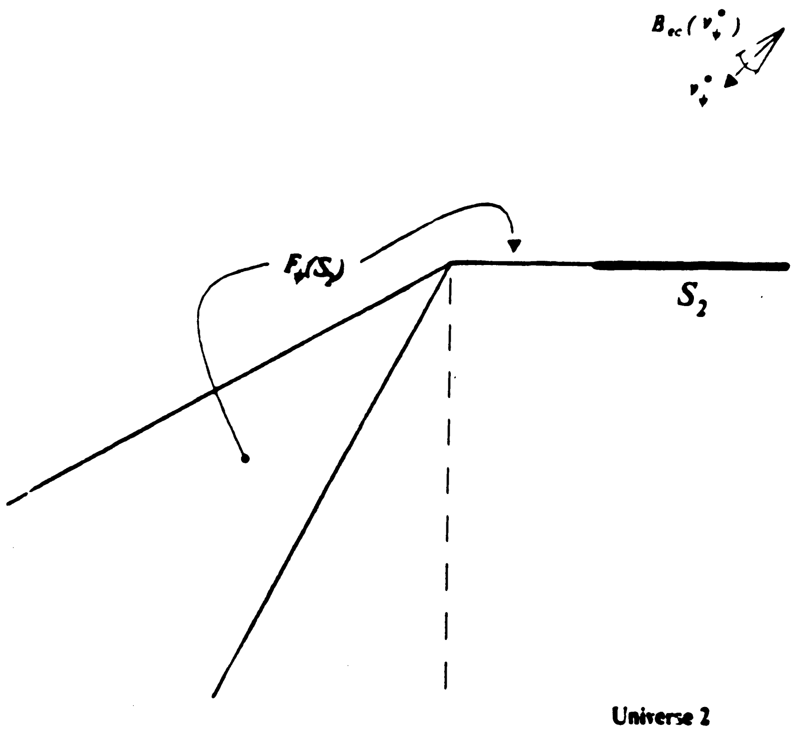

Figure 26: The forward projection under ψ of S_2.

(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 (4), 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.[16] 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.

Actually, this description of the procedure is not quite complete. 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.

5.2 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 that 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

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

in [D]. While in the appendix 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.

5.3 An Approximation to the Push-Forward

We may have to approximate the desired push-forward. We give such an approximation here. 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. 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 (4) 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. Now, if the termination predicate "knew" that in example (4) 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." S_2 is obtained as the set of places outside of S_1 where the lied-to termination predicate can halt.

Even formalizing the construction of this simple push-forward is subtle; details are given in [D]. 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.

5.4 Notation for Push-Forwards

For a motion θ from start region R, we denote the general push-forward by $F_{*\theta}(R)$. That is, $F_{*\theta}(R)$ is that subset of the forward projection $F_\theta(R)$ where the motion θ will terminate when executed from R.

LIMITED employs an approximate push-forward. It is called the ***a priori push-forward based on sticking termination***, and is defined as that subset of the forward projection where sticking is possible. This push-forward is correct as long as all motions use sticking as a termination condition. For a set U we define $\text{push}_\theta(U)$ as all points in U where sticking is possible. Where possible, we will develop the EDR theory for the general push-forward which employs the full power of the termination predicate. In places, however, and when describing LIMITED, we will specialize the push-forward to use sticking termination alone, and thereby set $F_{*\theta}(R) = \text{push}_\theta(F_\theta(R))$.

5.5 Sticking as a Robust Subtheory of EDR

In the abstract EDR theory, one envisions the run-time termination predicate performing whatever computations are necessary to terminate a motion recognizably in G or H. That is, in principle, the planner decides what termination conditions are appropriate for a successful EDR strategy, and encodes them into the motion strategy. Of course, it is also the responsibility of the planner to verify that this encoding will always result in a distinguishable termination. In short, the abstract EDR theory can employ the full power of the [LMT] preimage framework to generate motion strategies.

However, LIMITED employs only certain restricted termination conditions, as we saw above. In particular, sticking is used in most experiments. This restricts the class of strategies LIMITED can generate. The restriction requires some justification, and that is the purpose of this section.

First, recall that in polyhedral environments with a bounding box, sticking termination is sufficient to ensure that all pure translations eventually terminate [Buckley]. In general, in this work we have made the heuristic assumption that motions can eventually be terminated via sticking. Failing this, we also entertain the weaker assumption that if sticking is insufficient, then time can be employed to wait until $G \cup H$ has been achieved before termination.[17]

To analyze the structure of sticking termination, let us introduce the following notation. If the robot recognizably achieves $G \cup H$, this means that the run-time executor can determine that G or H has been achieved, but cannot necessarily tell which of G or H has been entered. If the robot recognizably achieves $\{G, H\}$, then it can further distinguish which of G or H it has reached. $G \cup H$ is called the ***union*** while the set notation is called the ***distinguishable union***.

Throughout this section we assume without loss of generality that the goal G is contained within the forward projection[18] If this is not the case, then intersect them to obtain a new goal.

LIMITED tries to decompose this problem—of ensuring that all trajectories terminate recognizably in $\{G, H\}$—into two subproblems. The first is to ensure that the motion

[17]See sec. 8.3 of [Donald, 87].

[18]This is not a severe restriction; see sec. 7.3 of [Donald, 87].

in fact terminates in $G \cup H$. That is, the problem is to determine that *at least* one of G or H has been achieved, although the robot may not know which. The second problem is to distinguish between G and H, once $G \cup H$ has been achieved.

Note that the first problem requires distinguishing between $G \cup H$ and its complement. Here is the key point:

- The construction of H guarantees tautologously that with sticking termination, $G \cup H$ will be recognizably achieved when the motion terminates. That is, with sticking termination, no motion can terminate outside of $G \cup H$.

This resolves the first subproblem. Thus

With sticking termination, all candidate *one-step EDR strategies eventually terminate recognizably in $G \cup H$ (but not necessarily in $\{G, H\}$). Of these, all* valid *EDR strategies can distinguish between G and H after termination, and hence recognizably terminate in* $\{G, H\}$.

The second subproblem is how to distinguish between G and H once $G \cup H$ has been achieved. Using sensors and history, the termination predicate can decide that a motion has recognizably entered a (phase-space) goal G_β when

$$F_\theta(R) \cap (B_{ep}(t) \times B_{ev}(t)) \subset G_\beta. \tag{$*$}$$

(See sec. 4.1). Now, when is it the case that the termination predicate can distinguish which of G or H has been reached? Exactly when $(*)$ is true for G_β in $\{G, H\}$. However, in our case, sticking termination guarantees that the actual position and velocity lie within $G \cup H$. Furthermore, $G \cup H$ is a subset of the forward projection, and G and H are disjoint by construction. The forward projection provided no further constraint in distinguishing between G and H. Thus history plays no role in the run-time distinguishing actions of the robot executive; history has been pre-encoded into the structure of H. Hence, we can predict that the run-time executor can distinguish which of G or H has been achieved when the planner can predict that G and H are distinguishable using sensors alone. A procedure—albeit not completely general—for deciding this question was described in 2.4.

5.5.1 Generalizations

There are several possible generalizations of these termination techniques. First, it may be possible for the run-time executor to use time to ensure that the motion terminates in $G \cup H$. That is, forward projections may, in principle, be indexed by time. Hence in $(*)$, $F_\theta(R)$ is replaced by $F_\theta(R, t)$, the instantaneous forward projection of R under θ at time t, which is typically much smaller. See sec. 4.1. The termination predicate in this case monitors a clock, in addition to position and velocity sensors. However, in this case, history (by which we mean $F_\theta(R,t)$) could be employed to distinguish G from H, even though the motion had terminated recognizably in $G \cup H$. The reason for this is that the time-indexed forward projection has not been pre-encoded into the structure of H. That is, H was constructed using the *timeless* forward projection, which the union of all time-indexed forward projections. Hence, we can summarize these observations as follows:

- *If a termination predicate without time uses sticking to terminate the motion, then distinguishing G from H is a history-free decision. However, for a termination predicate with a sense of time, the decision is not history-free.*

Thus sticking subtheory does not preclude more general termination techniques based on position, force, and time sensing. However, two computational issues become more difficult. First, sticking termination is a robust method for ensuring termination in $G \cup H$. With time termination, or more general position/force termination criteria, it is more difficult to ensure termination in $G \cup H$—although admittedly these criteria are more powerful. Second, after sticking termination, deciding between G and H is history-free. With more general termination predicates, history can provide extra constraint in distinguishing between G and H.

Finally, note that [Buckley] recognized the value of sticking termination when implementing an [LMT] planner for guaranteed strategies in $\Re^3$. His planner used sticking termination. In particular, he provided certain criteria for guaranteeing that a strategy eventually terminates in sticking. Buckley's criteria amount to ensuring that the environment is finite polyhedral, within a bounding polyhedral box.

5.6 Example: Multi-step EDR Plan for Peg-in-Hole with Model Error

The advantage of the push-forward technique is that it can be made computational. We now give LIMITED's algorithm for generating multi-step strategies using push-forwards, and describe an experiment which used this method.

Recall section 1, figs. 6-16, which described a two-step EDR plan for a peg-in-hole plan with 3 DOF model error. Here is how this multi-step strategy was generated:

Algorithm Multi

1. *First, try to generate a one-step EDR strategy using the algorithm 1EDR in sec. 2.4.*

Suppose this fails. Then:

2. *Generate a commanded velocity v_θ^*, such that the forward projection of the start region intersects the goal in some slice.*

3. *Compute the EDR region H for v_θ^*.*

4. *Compute the sticking push-forward of the motion, $R_1 = \mathrm{push}_\theta(G \cup H)$.*

5. *Using R_1 as the start region, generate a one-step EDR strategy using algorithm 1EDR.*

Of course, in LIMITED the computation is memoized so that the projection and EDR regions computed in step (1) are not recalculated in steps (2) and (3). Obviously, we can extend this algorithm to generate longer strategies which push-forward several times and finally terminate in a single-step EDR strategy.

Now, LIMITED is a multi-resolution planner. The algorithm outlined above generates a multi-step strategy at a single resolution. The *resolution* of planning is simply the set of α values in which slices are taken. A resolution S_1 is *finer* than S_2 if it contains more slices. The multi-resolution outer loop works like this:

M1. *At a coarse resolution, generate a multi-step EDR strategy $\theta_1, \ldots, \theta_n$ using the forward-chaining single-resolution algorithm above.*

M2. *Select a finer resolution. Use the directions $\theta_1, \ldots, \theta_n$ as a suggested strategy and attempt to verify that it is an EDR strategy at the finer resolution.*

M3. *If $\theta_1, \ldots, \theta_n$ is not an EDR strategy at the finer resolution, try to modify it so that it is, by using $\theta_1, \ldots, \theta_n$ as suggested directions, and searching nearby directions at all levels.*

The process terminates when the resolution is finer than some predetermined level.[19] The critical slice method described in [Donald, 87] may be one way to obtain such an *a priori* bound and know that it is sufficient. In LIMITED, however, the bound is a user input because otherwise the number of slices required would be prohibitive.

In the peg-in-hole example there were 3DOF of model error: the *width* of the hole, the *depth* of the chamfers, and the *orientation* of the hole. The resolutions used in planning the two-step strategy were as follows:

R1. Holding orientation fixed, 4 slices of the depth $\times$ width axes.

R2. Holding orientation fixed, 16 slices of the depth $\times$ width axes.

[19] Or, when at some level, no EDR strategy can be found.

R3. Holding orientation fixed, 72 slices of the depth $\times$ width axes.

R4. 100 slices of the depth $\times$ width $\times$ orientation axes.

For detailed output traces of LIMITED, please see [Donald, 87].

5.7 The Loss of Power with Push-Forward Techniques

While push-forwards permit us to develop simple algorithms for generating multi-step strategies, clearly these algorithms are theoretically less powerful than solving the preimage equations in full generality. We now attempt to give an intuitive characterization of the loss of power. In particular, push-forwards are general enough for the peg-in-hole EDR strategy with model error. However, they are not general enough to generate the grasp-centering plan. We now discuss where in the grasp-centering example the push-forward techniques are inadequate. The key point is this: if each commanded motion and termination condition could be non-deterministically "guessed," and a push-forward for each motion and termination condition could be computed, then in the grasp-center example this would suffice to generate a strategy. However, the push-forward algorithms we have developed are not powerful enough to do this.

First, let us derive the push-forwards of each motion in strategies *EDR-Center* and *Guarantee-Center*. Recall that *E1* is the first step of the EDR plan, and motions 1, 2, and 3 are steps in the (subsequent) guaranteed plan. In the third column we note whether or not the push-forward technique is computationally effective for this motion.

Motion	Push – Forward	Computational?
$E1$	$\{L, H\}$	yes
1	R	yes
2	$\{E_\alpha\}$	no
3	G	no

The push-forwards for motions *E1* and *1* can be computed using the algorithms of [Donald, 87] (see sec. 2.4) and algorithm *Multi* above. In motion *E1*, L may be found using sticking termination. H may be found using time, or position and force sensing termination. In motion *1*, R may be found using contact, or sticking termination. However, our algorithms cannot compute the push-forward $\{E_\alpha\}$, which contains an infinite number of components. Furthermore, we have not developed algorithms for computing push-forwards based on time-termination (except for elapsed time termination, of the form "terminate anytime after t seconds"). Thus the push-forward G for the last motion cannot be computed by our algorithms either.

5.7.1 Discussion

Let us pause to review. We first described a fully-general, but non-computational technique for generating multi-step strategies. This method—solving the preimage equations—was applied to the grasp-centering example. Next the push-forward techniques were introduced as a computational, although less powerful approach to the synthesis of multi-step strategies. Push-forward algorithms were described, and we saw how LIMITED used these techniques to generate a two-step plan for the peg-in-hole problem with model error. Finally, we discussed the limitations of the push-forward techniques. We saw that they were not powerful enough to solve the grasp-center problem in its entirety. By describing an experiment where push-forwards suffice, and showing an example where they are insufficiently general, we have tried to give an intuitive but fairly precise characterization for the relative power of push-forwards.

6 Failure Mode Analysis

Push-forward techniques require a precise geometrical characterization of the forward-projection, and algorithms for computing it. The gear-meshing example of sec. 1 is a problem in a four-dimensional generalized configuration space with pushing. Two of the dimensions are rotational: one of these can be commanded, and the other cannot, but the position along this dimension may be changed via pushing. It is difficult to develop good forward projection algorithms in this generalized configuration space, although critical-slice methods are a start. For this reason, a different technique was developed for planning multi-step strategies in this domain. It is applicable for any generalized configuration space with the same degrees of freedom and pushing characteristics (that is, any polygonal shapes in place of the gears). The new technique is called ***failure mode analysis***; we describe it in this section.

Failure-mode analysis is a method for synthesizing multi-step strategies using a kind of "approximate" or "***a priori***" forward projection. At first glance, it may appear unrelated to push-forward or preimage techniques. However, in the next section, on the weak EDR theory, we present a viewpoint which essentially "unifies" the three approaches.

6.1 Introduction to Failure Mode Analysis

Recall the gear-meshing plan LIMITED generated in sec. 1, fig. 4. In particular, we suppose that vision is poor, or that the gears are accessible to the robot gripper, but not to the camera. This means that position sensing will be very inaccurate, and hence may be of no use to determine whether the gears are successfully meshed. This will often be the case in practice. In this case, force sensing must be used to disambiguate the success of the first motion (meshing) from failure (jamming in an unmeshed state). A multi-step strategy is required. While pushing of B can change its orientation, and hence cause motion across J (sec. 2.2.2) our focus here will be on the multi-step strategy generation and not on the physics of pushing.

In the gear-meshing plan, motion θ_2 is used to disambiguate the result of motion θ_1. The technique used is *failure mode analysis*. LIMITED is given a repertory of qualitative failure modes, which comprise sticking and breaking contact. Motion θ_1 can end in a "good" region (meshed) or a "bad" region (jam). LIMITED tried to generate a disambiguating motion as a second step. This motion is required to terminate in a failure mode from all "bad" regions.

Here is how LIMITED generates motion θ_2. Let H be the EDR region for motion θ_1. The planner determines all configurations where motion θ_1 can terminate outside of G. Call this region $\text{push}_{\theta_1}(H)$. $\text{push}_{\theta_1}(H)$ then forms the start region for motion θ_2. LIMITED then uses quasi-static analysis to "prove" that when A is at any configuration in $\text{push}_{\theta_1}(H)$, and a pure rotation of A is commanded, that all possible motions of A result in sticking or breaking contact. Sticking and breaking contact are called *failure modes*; there is a class of EDR plans which can be terminated in failure when sticking or breaking contact are detected. EDR planning with failure modes constitutes a robust subtheory of EDR. It is a subtheory because assuming this kind of failure mode is a restrictive assumption to make planning tractable. It is robust because sticking and breaking contact are easy to recognize, relatively speaking, as failure modes by a run-time robot executor.

From the preimage point of view, failure modes are implemented simply as different classes of termination predicates.

6.2 Specifying the Goal: Functional Descriptions

Recall our discussion of sticking as a termination condition in chapter Sticking had the advantage of ensuring "good" behavior in the EDR region H. In particular, it could be guaranteed that all motions would eventually terminate in $G \cup H$, rendering the distinguishability of G vs. H a history-free decision. However, in order for a sticking termination predicate to generate good EDR plans, it was in fact necessary to ensure that the motion strategy has "good" behavior at the goal as well. In particular, the commanded motion should stick at the goal.

In failure mode analysis, we have a similar situation. The purpose of motion θ_2 is to force all motions starting from $\text{push}_{\theta_1}(H)$ to terminate in sticking or breaking contact. Clearly this is only useful if ***no motion*** from $\text{push}_{\theta_1}(G)$ can *even possibly* terminate in sticking or breaking contact. This is the required "good" behavior at the goal. Thus, in an EDR plan generated by failure mode analysis,

F1. *Under motion θ_2, all motions starting from* $\text{push}_{\theta_1}(H)$ *must terminate in a failure mode.*

F2. *No motion from* $\text{push}_{\theta_1}(G)$ *can possibly terminate in a failure mode.*

F3. *The goal is a fixed-point under motion θ_2.*

LIMITED decides whether or not (F1) is true. However, (F2) is given as input to LIMITED. We will now discuss how (F2) is specified. In the next section we will describe algorithms for computing (F1). (F3) may be decided using forward projections; the actual condition we require is

$$F_{\theta_2}(\text{push}_{\theta_1}(G)) \subset G,$$

which is implied by the fixed-point equation

$$F_{\theta_2}(G) = G, \tag{F3}$$

since of course $\text{push}_{\theta_1}(G)$ is contained in G.

The goal state for gear meshing may be viewed purely geometrically. That is, it may be viewed as a set in generalized configuration space. This view is useful for computing the EDR regions. Alternatively, the goal may be specified through a functional description. For example, we might specify the goal as a difference equation (DE). The intuition behind this difference equation formulation of the goal is, *"In the goal, any finite rotation of A results in an equal and opposite rotation of B."* More precisely, the difference equation specifies:

DE. *Command any non-zero finite rotation $\Delta\alpha_1$ to A. In the goal, this results in a finite rotation of A by $\Delta\alpha_1$ and of B by $-\Delta\alpha_1$.*[20]

This difference equation captures the functional aspects of the gears in their meshed state. Now, it is clear that this equation may be "differentialized." That is, we consider it to be true for all non-zero displacements, no matter how small. If this is the case, then it is clear that breaking contact is in direct contradiction to the truth of the difference equation (DE). This is because if contact is broken, then there exists some finite rotation of A that will not affect the orientation of B. Similarly, sticking contradicts the truth of the difference equation, for if the gears stick, then they are not properly meshed, i.e., we do not obtain equal and opposite rotations.

In LIMITED failure mode analysis, we view the goal state as a combined geometrical and functional specification. In [Donald, 87], we considered three ways of specifying the functional aspects of the goal. The last, which decides questions about goal predicates via the theory of real closed fields, is only of theoretical interest. The second is a heuristic approximation to such an inference engine. The first is a more robust solution with an engineering flavor. It places on the user the burden of ensuring well-behaved qualitative behavior at the goal. We will confine ourselves to the first, and simplest method in this paper:

[20] A and B are the same size. Clearly, this may be generalized to different pitch gears.

6.2.1 Specifying the Functional Aspects of the Goal

Method 1. User input. In this method, it is the responsibility of the user to ensure that (F2) is true. That is, the user must guarantee that failure modes cannot occur at the goal. This, of course, is the easiest method. If the user guarantees that (F2) holds, then it remains only for LIMITED to show (F1).

Some of the greatest and most interesting unsolved problems in geometrical robotics lie in the interaction of functional and geometrical descriptions of goals. In particular, we would like to devise algorithms for computing a geometrical goal region given a functional description—for example, a quantified difference equation—for the desired behavior in the goal state. Conversely, we would like to be able to infer a functional description of the goal from its geometrical aspects. The latter would be useful in automatically generating termination predicates to recognize the goal.

6.3 Approximate Algorithms for Failure Mode Analysis

We now describe algorithms for deciding whether

F1. *Under motion θ_2, all motions starting from* $\text{push}_{\theta_1}(H)$ *must terminate in a failure mode.*

Let us denote $\text{push}_{\theta_1}(H)$ by H_1. These algorithms use time-indexed forward projections to prove that under θ_2, all paths starting in H_1 eventually stick or break contact. The algorithms are approximate, although conservative. That is, if they terminate then (F1) is true. However, they may not terminate if (F1) is false, and they may miss cases where (F1) is true. The accuracy of the algorithm increases as the time steps for the time-indexed forward projections are taken to be finer. In the 4D generalized configuration space for the gears, which is $\Re^2 \times S^1 \times S^1$, these time-steps correspond to the fineness of the slice resolution across the rotational dimensions.

We will first describe a quite general algorithm for deciding (F1). It is applicable wherever we can obtain a computational characterization of time-indexed forward projections. Later, we will give a specialized algorithm in the generalized configuration space for the gears, and show that it is in fact a special case of the general algorithm.

6.3.1 A General Algorithm

The basic idea is to step along in time, simulating the motion, and determine whether or not it breaks contact or sticks. Of course, we must simulate all possible motions, using forward projections.

First we must develop some notation. Recall that for a planar set H_1, ∂H_1 denotes its obstacle edges. Here, we will use it to more generally to denote the obstacle surfaces (as opposed to the free-space surfaces) bounding a set H_1 in generalized configuration space. (In our case H_1, the input to the algorithm, is the push-forward of motion θ_1).

Let x be a point in generalized configuration space. Then $stick_\theta(x)$ is true if sticking is necessary at x under all control velocities $B_{ec}(v_\theta^*)$ consistent with the nominal commanded velocity v_θ^*. Let $stick_\theta(H_1)$ denote all points x in H_1 where $stick_\theta(x)$ holds.

Now, assume some positive minimum modulus bound on the commanded velocity. We use $F_{\theta,\Delta t}(\cdot)$ as the time-indexed forward projection operator (see [Erdmann]). So $F_{\theta,\Delta t}(H_1)$ denotes the set of possible positions the robot can be at at time Δt, having started in H_1 at time $t = 0$.

Now, we are ready to give the general algorithm for deciding (F1):

Algorithm Gen

1. *Let* $F \leftarrow F_{\theta,\Delta t}(H_1)$.
2. *Let* $H_2 \leftarrow \partial F - stick_{\theta_2}(\partial F)$.
3. *When* $H_2 = \emptyset$, *we have proven that all paths from* H_1 *must eventually stick or break contact. Halt.*
4. *Else,* $H_1 \leftarrow H_2$. *Goto (1).*

Note that H_1 is permitted to be in free-space, although given the sticking push-forward it will, in fact, always be on a generalized configuration space boundary. Note that *Gen* is a semi-decision procedure. Clearly, if the algorithm halts, then all paths originating in H_1 eventually break contact or stick. Fig. 27 illustrates the algorithm. Suppose the H_1 region is the edge e. Its forward projection after Δt is the region $U \cup g$. The obstacle edges of the forward projection are e', f, and g. Sticking must occur on f. Hence, H_2 is $e' \cup g$.

We now mention a basic property of forward projections that this algorithm exploits. It is the property that forward projection commutes with union. In particular, if we have

$$H_1 = \overbrace{H_B}^{\text{boundary}} + \overbrace{H_F}^{\text{free-space}} .$$

then

$$F_\theta(H_1) = F_\theta(H_B \cup H_F) = F_\theta(H_B) \cup F_\theta(H_F).$$

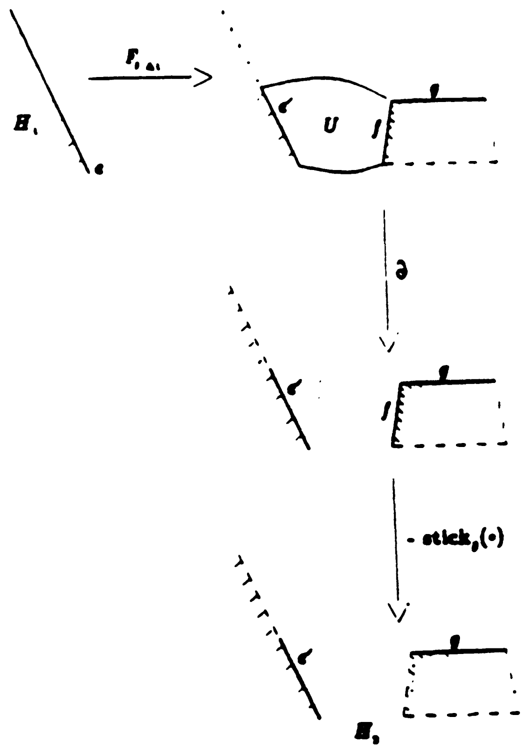

Figure 27: **Illustration of the general algorithm. The start region H_1 is the edge e. Its forward projection after Δt is the region $U \cup g$. The obstacle edges of the forward projection are e', f, and g. Sticking must occur on f. Hence, H_2 is $e' \cup g$.**

This key property permits the algorithm to decompose the failure mode analysis into essentially independent decision problems about the forward projections of the free-space, sliding, and sticking regions in the push-forward.

6.3.2 A Specialized Algorithm

For failure-mode analysis, LIMITED employs an algorithm that is a special case of the general algorithm above. The idea is that when commanding a pure rotation of A, the time-indexed forward projection across slices can be well approximated by the *differential forward projection*. The differential forward projection is a technique for propagating the forward projection across slices, when rotations of A and B are permitted. Recall our notation for motions θ_1 and θ_2. θ_1 is a commanded pure translation of A, and may be viewed as unit vector $v^*_{\theta_1}$ in the plane. θ_2 is a commanded pure rotation of A, and may be viewed as a member of $\{ +d\alpha_1, -d\alpha_1 \}$, for positive and negative commanded rotations.

Differential and Propagated Forward Projections

Pure Translations. Forward projections must be propagated between slices even when a pure translation is commanded, since a pure translation θ_1 can alter the orientation of B, and hence the slice-value, through pushing. Here is how the differential forward projection is constructed for a pure translation θ_1. Let $(x, y, \alpha_1, \alpha_2)$ denote a configuration in the generalized configuration space for the gears, $\Re^2 \times S^1 \times S^1$. (x, y, α_1) denotes a configuration of A. α_2 denotes the configuration of B. Hence, we regard the orientation of B (the "last" S^1 in the product) as J. Now, H_1 is a set in generalized configuration space. Let $H_1|_{\alpha_1,\alpha_2}$ denote a particular x-y slice of H_1 for orientation α_1 of A and α_2 of B.

Motion θ_1 commands a pure translation of A. Now, for each edge in $H_1|_{\alpha_1,\alpha_2}$, LIMITED performs a quasi-static analysis to determine the possible impending motions of A and B. That is, it determines which way(s) A and B can rotate. These directions may be viewed as tangent vectors to the pure rotational dimensions of generalized configuration space. The set of possible directions may be identified with a set of pairs

$$\{ -d\alpha_1, 0, +d\alpha_1 \} \times \{ -d\alpha_2, 0, +d\alpha_2 \} \tag{9}$$

in the tangent space to $(S^1 \times S^1)$. By performing this analysis for all edges, we obtain a set of directions,

$$dF_{\theta_1}(H_1|_{\alpha_1,\alpha_2}),$$

which is called the ***differential forward projection of*** $H_1|_{\alpha_1,\alpha_2}$ ***under*** θ_1. It is assumed that commanding θ_1 from region $H_1|_{\alpha_1,\alpha_2}$ can result in any motion direction in this set.

Suppose (α_1', α_2') is a slice taken in the direction of some tangent vector $\mathbf{v}$ in the differential forward projection. For example, if $\mathbf{v} = (+d\alpha_1, -d\alpha_2)$, then $\alpha_1' = \alpha_1 + \epsilon_1$ and $\alpha_2' = \alpha_2 - \epsilon_2$ for some small positive scalars ϵ_1 and ϵ_2.

Now, the forward projection may be propagated to the adjacent slice (α_1', α_2') as follows. An edge e_i in $H_1|_{\alpha_1,\alpha_2}$ corresponds to the intersection of an algebraic surface V in generalized configuration space with the "plane" $\Re^2 \times \{ (\alpha_1, \alpha_2) \}$. V is followed into (α_1', α_2'), and the forward projection of e_i is taken to be the intersection of V with the "plane" $\Re^2 \times \{ (\alpha_1', \alpha_2') \}$. In this manner, we obtain a set of edges $\{ e_i' \}$ in the new slice. The pure translational forward projection of these edges under θ_1 is then computed within this slice, so the propagated forward projection is $F_{\theta_1}(\{ e_i' \})$. This propagated forward projection is computed at a fixed orientation of A and B. Ideally, the planner should decide whether the sliding characteristics change along V while moving through rotation space. The rotational values which are sliding-critical are discussed in 6.3 of [Donald, 87]. [21] The propagated forward projection increases in accuracy as the slices are taken closer together.

Pure Rotations. Consider the problem of computing forward projections across slices for a commanded pure rotation $\theta_2 \in \{ +d\alpha_1, -d\alpha_1 \}$. For simplicity, we first consider the case where H_1 consists of a single point. Let $\mathbf{x}$ be a point in the plane, and $(\mathbf{x}, \alpha_1, \alpha_2)$ be a configuration where A and B are in contact. Then the differential forward projection of $\mathbf{x}$ under θ_2 will consist of vectors in the set of eq. (9). The differential forward projection has the same structure as in the pure translational case. It may be computed using quasi-static analysis. (see the next subsection below).

Suppose for the sake of development that the differential forward projection consists of *exactly one* direction $\mathbf{v}$, and that (α_1', α_2') is an adjacent slice in that direction, as above. Now we ask, what is the propagated forward projection of $\mathbf{x}$ into the adjacent slice, (α_1', α_2')? Well, it can be one of two things: either it is $\mathbf{x}$, or it is empty. The reason is that x-y position is invariant[22] under θ_2. Thus, an upper bound on the propagated forward projection of $H_1|_{\alpha_1,\alpha_2}$ into an adjacent slice (α_1', α_2') is found by simply "copying"[23] $H_1|_{\alpha_1,\alpha_2}$ into slice (α_1', α_2').

Now, consider the propagated forward projection of $(\mathbf{x}, \alpha_1, \alpha_2)$, under motion θ_2, into slice α_1', α_2'. It is simply the point $(\mathbf{x}, \alpha_1', \alpha_2')$. There are three possible qualitative outcomes:

1. $\mathbf{x}$ is inside a generalized configuration space obstacle in slice (α_1', α_2').

2. $\mathbf{x}$ is in free space in slice (α_1', α_2').

[21] Detecting sliding critical orientation parameters along the algebraic surface V has not been implemented in LIMITED. Thus the propagated forward projection may be larger than it need be.

[22] See below for more on this assumption.

[23] We use the awkward term "copying" instead of "translating", since while the latter is precise mathematically it is confusing robotically.

3. $\mathbf{x}$ is on the boundary of a generalized configuration space obstacle in slice (α_1', α_2').

Obviously, (2) implies that contact has been broken. (1) corresponds to a physically impossible situation. Since the configuration $(\mathbf{x}, \alpha_1', \alpha_2')$ is physically unattainable, this means that the commanded motion θ_2 must result in sticking (no actual motion) before (α_1', α_2') can be reached. Now, if we have either outcome (1) or (2) then we have proven that, under θ_2, any path for the robot starting at $(\mathbf{x}, \alpha_1, \alpha_2)$ must stick (1) or break contact (2).

Suppose, however, we have outcome (3). This outcome is not inconsistent with the negation of (F1). That is, it has not yet been shown that any path from $(\mathbf{x}, \alpha_1, \alpha_2)$ will stick or break contact. In this case, in the new slice (α_1', α_2') we again perform the quasi-static analysis and forward project again into yet another slice. This process continues until either outcomes (1) or (2) are obtained.

More generally, the differential forward projection of $(\mathbf{x}, \alpha_1, \alpha_2)$ could consist of more than one vector. In this case, each must be taken as a forward projection direction, and in each direction we must show that outcomes (1) or (2) eventually occur. That is, the computation above must performed for each direction predicted by the quasi-static analysis, and *all* directions must terminate in sticking or breaking contact.

We have described how the failure mode analysis proceeds when the push-forward H_1 of the first motion θ_1 is simply a point. It remains to generalize the discussion to the case where H_1 is a region in generalized configuration space, represented by slices. We first introduce some notation. If CO denotes the generalized configuration space obstacle for A due to B, then let $CO|_{\alpha_1,\alpha_2}$ denote the x-y slice of CO at orientations (α_1, α_2). As usual, let ∂ denote the obstacle edges of a set. ϵ is the slice resolution parameter. The input to this procedure is a stack Q of x-y-slices of H_1. An entry in Q is a triple, consisting of an x-y slice $H_1|_{\alpha_1,\alpha_2}$, and (α_1, α_2), the orientations at which the slice was computed.

Algorithm Spec

1. *Do until* $Q = \emptyset$:
2. *Pop the triple* $\langle H_1|_{\alpha_1,\alpha_2}, \alpha_1, \alpha_2 \rangle$ *off* Q. *Let* $H_2 \leftarrow H_1|_{\alpha_1,\alpha_2}$.
3. *Let* $dF \leftarrow dF_{\theta_2}(H_2)$.
4. *For each* $\mathbf{v}$ *in* dF *do:*
5. *Let* $(\alpha_1', \alpha_2') \leftarrow \epsilon\mathbf{v} + (\alpha_1, \alpha_2)$.
6. *Compute* $CO|_{\alpha_1',\alpha_2'}$.

7. *Let* $H_3 \leftarrow H_2 \cap \partial CO|_{\alpha_1', \alpha_2'}$.

8. *If* $H_3 \neq \emptyset$, *push the triple* $\langle H_3, \alpha_1', \alpha_2' \rangle$ *onto* Q.

Note that this is a semi-decision procedure. This is the algorithm that is actually implemented in LIMITED. The key step is of course the iteration step (7), which we think of as

$$\text{“}H_2 \leftarrow H_2 \cap \partial CO|_{\alpha_1', \alpha_2'}\text{”}$$

which is repeated "until H_2 is null." $CO|_{\alpha_1', \alpha_2'}$ is computed using the plane sweep union algorithm, as is the intersection.

6.3.3 On the Invariance Assumption

We have assumed that x-y position of A is invariant under a commanded pure rotation θ_2. That is, commanding a pure rotation cannot result in an induced translation. On the other hand, we allow a commanded pure translation of A to induce a rotation of B (but not of A). These assumptions are realistic if, for example, the robot has gripped A by its center shaft, and the manipulator is very stiff in the x-y directions when commanding a pure rotation. In future work, relaxing this asymmetry should be explored.

6.3.4 Quasi-Static Analysis

We now show how the quasi-static analysis is computed. It is quite simple. We view the commanded velocity to A as $\omega = (0, 0, \pm 1)$. When the gears are in contact, this defines a moving constraint in the configuration space of B, which is a one-dimensional space. Given a contact configuration, we compute the moment arm in order to determine the direction of the constraint. The moment arm on B (resp., A) is simply the vector from B's (resp. A's) center of mass to the contact point in real space. The contact point in real space can be recovered from the contact point in configuration space.

Let r_a and r_b denote the moment arms on A and B, resp. Then the instantaneous velocity v_a of the contact point on A, given ω, is $\omega \times r_a$. B's direction of impending motion is given by the sign of the expression

$$r_b \times \pi_2 v_a = r_b \times \pi_2(\omega \times r_a),$$

where π_2 denotes the projection of $\Re^3$ onto $\Re^2$.

We now discuss recovery of the moment arms from the contact configuration. Let COM_A and COM_B denote the centers of mass of A and B. In these experiments, they are simply the centers of the gears. Suppose $(\mathbf{x}, \alpha_1, \alpha_2)$ is a contact configuration. Then it lies on an algebraic surface in the generalized configuration space $\Re^2 \times S^1 \times S^1$. This surface is one of two types [Lozano-Pérez]. Let $-A$ denote the reflection of A about its reference point. A type (A) surface is generated by an edge e_a of $-A$ and a vertex b_j of B. A type (B) surface is generated by a vertex a_i of $-A$ and an edge e_b of B. Each edge-vertex or vertex-edge pair is called the ***generator pair*** of the constraint surface [Donald]. The edges and vertices of $-A$ (resp. B) rotate with α_1 (resp., α_2). An (α_1, α_2)-slice of the surface is found by rotating its generators by (α_1, α_2), and taking their Minkowski sum. Hence the surface may be viewed as a parameterized line-equation, by (α_1, α_2). The table below gives the details for recovering the moment arms from the contact configuration, contact surface in generalized configuration space, and centers of mass. We employ the following notation. For an edge e or a vertex v, $e(\alpha)$ and $v(\alpha)$ respectively denote e and v rotated to orientation α. $\oplus$ denotes ***convolution*** (sometimes known as the *Minkowski sum*). For two sets U and V, $U \oplus V = \{v + u \mid u \in U,\ v \in V\}$.

Type	Surface	Moment arm on B r_b	Moment arm on A r_a
A	$e_a(\alpha_1) \oplus b_j(\alpha_2)$	$b_j(\alpha_2) - COM_B$	$b_j(\alpha_2) - \mathbf{x} - COM_A$
B	$a_i(\alpha_1) \oplus e_b(\alpha_2)$	$\mathbf{x} - a_i(\alpha_1) - COM_B$	$-a_i(\alpha_1) - COM_A$

6.3.5 Stiction

What the *Spec* algorithm does is this: it tries to show that from any slice of H_1, all paths that could possibly evolve from commanding a rotation of A either (1) remain in the first slice, or (2) in some subsequent slice, stick or break contact. We have described how (2) is detected. (1) is a form of stiction; the gears do not turn. Note that (1) is a form of sticking behavior, since no motion occurs. Staying in the same slice means that (α_1, α_2) are fixed, and x and y are fixed ***a priori***. Hence events (1) or (2) satisfy (F1). That is, (1) is also a form of sticking, and can be detected at run-time by the termination predicate.

Now, suppose B sticks but A continues to turn? This type of stiction is also no problem, since it corresponds to a differential motion $(\pm d\alpha_1, 0)$, which can be predicted by the differential forward projection.

6.3.6 Failures Outside the EDR Framework

We will momentarily digress to a practical question. It would appear that for failure mode analysis to work, non-uniform stiction would be required in our physical model of the gears. That is, it would seem that stiction would have to be impossible in the goal, but possible in H_1. This is not the assumption made in the geometrical EDR analysis and implementation. We now show that uniform stiction is in fact not an impediment to failure mode analysis, either.

It is the responsibility of the user, or of some external inference system, to ensure that (F2) holds. Suppose, however, that this inference is incorrect, and that at run-time stiction does, in fact, occur in the goal, and that the gears jam. In this case the run-time executive will signal failure, *even though the geometrical goal has been achieved.* At first glance it appears that this is incorrect. However, when we regard the goal as a combined geometrical *and functional* specification, it is clear that this is actually the correct termination diagnosis. That is, even though the geometrical goal has been achieved, stiction prevents the quantified difference equation (DE) on paths, $goal(\cdot)$, from being satisfied. Since *something* (specifically, stiction) has prevented achievement of the functional goal, it is completely correct for the run-time executive to signal failure in this case. However, note that we regard this as serendipitous failure detection, and not as inherent in the EDR framework.

6.3.7 Generalizations

The specialized algorithm *Spec* may be generalized. The properties it exploits are (1) that certain degrees of freedom in C and J can be held fixed, while others may be commanded, (2) that "slices" of CO can be computed, (3) set intersections can be computed, and (4) differential motion across the non-fixed degrees of freedom can be predicted using quasi-static analysis.

More precisely, the specialized algorithm generalizes to cases where we fix certain degrees of freedom C_f and J_f, command C_c, and permit J_c to vary (through pushing). Hence $\mathcal{G}$ is decomposed into

$$C_f \times C_c \times J_c \times J_f,$$

$B_{ec}(v^*_{\theta_2})$ lies in the tangent space to C_c, and all motion lies in the subspace $C_c \times J_c$. Using quasi-static analysis, we predict the impending motion direction, $\mathbf{v}$ which lies in the tangent space to $C_c \times J_c$. If $\boldsymbol{\alpha}$ is in $C_c \times J_c$, let $H_1|_{\boldsymbol{\alpha}}$ denote a slice of H_1 at $(\boldsymbol{\alpha})$. Thus $C_f \times J_f$ are the dimensions of the slice (like x, y in the gear example). Then we let $\boldsymbol{\alpha}' \leftarrow \boldsymbol{\alpha} + \epsilon\mathbf{v}$. Finally, the iteration step is

$$H_3 \leftarrow H_1|_{\boldsymbol{\alpha}} \cap \partial CO|_{\boldsymbol{\alpha}'}.$$

The rest of the algorithm goes through ***mutatis mutandis***. This generalization is somewhat theoretical, in that in practice the CO-slices, set intersections, and quasi-static analysis may be difficult to compute for higher-dimensional problems.

6.3.8 *Discussion: General vs. Specialized Algorithm for Failure-Mode Analysis

This starred subsection may be skipped at first reading. It contains a detailed proof.

The problem with implementing algorithm *Gen* directly is that arbitrary time-indexed forward projections are difficult to compute. For this reason we introduced a specialized algorithm for the gear planning. While algorithms *Spec* and *Gen* appear quite different, in fact, *Spec* is simply a special case of *Gen*. The motivation behind this viewpoint is to find a uniform framework for characterizing algorithms for failure mode analysis. That is, algorithm *Gen* can be viewed as a high-level computational approach to failure mode analysis, while *Spec* is an implementation of *Gen* in a restricted domain. We now discuss this view of the algorithms.

Recall the definition of $stick_{\theta_2}(\cdot)$. We now define $stick^{\bullet}_{\theta_2}(R)$ to be all points x in R such that any feasible path from x consistent with the control uncertainty $B_{ec}(v^{\bullet}_{\theta_2})$, eventually sticks.

We employ the following topological notions. $\overline{U}$ denotes the closure of a set U. U^c denotes its complement. $i(U)$ denotes its interior. $\overline{U}^c$ denotes the complement of the closure.

Now, consider the following step of the *Spec* algorithm,

7. $H_3 \leftarrow H_2 \cap \partial CO|_{\alpha_1',\alpha_2'}$,

where $H_2 = H_1|_{\alpha_1,\alpha_2}$. This step is equivalent to

$$H_3 \leftarrow H_2 - i(CO|_{\alpha_1',\alpha_2'}) - \overline{CO|_{\alpha_1',\alpha_2'}}^c, \tag{10}$$

where the set difference operator $-$ associates to the left. Now, the set

$$H_2 \cap i(CO|_{\alpha_1',\alpha_2'})$$

corresponds to all configurations $(\mathbf{x}, \alpha_1, \alpha_2)$ in the planar slice (α_1, α_2) such that under θ_2, any path from $(\mathbf{x}, \alpha_1, \alpha_2)$ will stick before reaching (α_1', α_2') if $\mathbf{x}$ is kept fixed. That is, it is configurations such that sticking will occur from $(\mathbf{x}, \alpha_1, \alpha_2)$ *between* (α_1, α_2) and (α_1', α_2').

Below, we argue that the set $H_2 \cap i(CO|_{\alpha_1',\alpha_2'})$ in algorithm *Spec* corresponds in a quite precise fashion to $stick_{\theta_2}(\partial F)$ in algorithm *Gen*. We see this as follows:

The following step of the *Gen* algorithm,[24]

2. $H_3 \leftarrow \partial F - stick_{\theta_2}(\partial F)$.

is equivalent to

$$H_3 \leftarrow F - \overline{CO}^c - stick_{\theta_2}(\partial F). \tag{11}$$

Now, it is possible to modify *Gen* as follows. Let

$$F_2 = F_{\theta_2,\Delta t}(H_1 - stick^{\bullet}_{\theta_2}(H_1)).$$

[24] We have lexicographically substituted H_3 for H_2 throughout algorithm *Gen* to facilitate the comparison with *Spec*.

Then we can replace the assignment (11) by eq. (12) and still have *Gen* be correct:

$$H_3 \leftarrow F_2 - \overline{CO}^c, \tag{12}$$

We wish to compare the step (12) of the thus modified *Gen* with the step of *Spec* given in eq. (10). In essence, we wish to show that eq. (10) is in some sense a "conservative" approximation to eq. (12), and hence conclude that algorithm *Spec* is simply a special case of algorithm *Gen*.

We must introduce some notation to compare eqs. (10) and (12). For a set V in $\Re^2$, we denote the set

$$V \times \{(\alpha_1, \alpha_2)\}$$

by

$$V \times (\alpha_1, \alpha_2).$$

Now, H_1 is a subset of $\mathcal{G}$. A slice of it $H_1|_{\alpha_1,\alpha_2}$ lies in the "plane" $\Re^2 \times (\alpha_1, \alpha_2)$. Let us denote its projection into $\Re^2$ by $\pi_2 H_1|_{\alpha_1,\alpha_2}$. Finally, for an arbitrary set U in generalized configuration space, let $U|_{\alpha_1,\alpha_2}$ denote an (α_1,α_2)-slice of it, that is,

$$U|_{\alpha_1,\alpha_2} = U \cap (\Re^2 \times (\alpha_1, \alpha_2)).$$

Claim: *Eq. (10) is a conservative approximation to eq. (12) in each slice.*
Proof: First, we obviously have

$$CO|_{\alpha_1',\alpha_2'} \subset CO. \tag{13}$$

Next, we need only show that

$$\left(F_{\theta_2,\Delta t}(H_1|_{\alpha_1,\alpha_2})\right)\Big|_{\alpha_1',\alpha_2'} \subset \pi_2(H_1|_{\alpha_1,\alpha_2}) \times (\alpha_1', \alpha_2') \tag{14}$$

and

$$H_1|_{\alpha_1,\alpha_2} \cap \pi_2\left(i(CO|_{\alpha_1',\alpha_2'})\right) \times (\alpha_1, \alpha_2) \subset stick^*_{\theta_2}(H_1|_{\alpha_1,\alpha_2}). \tag{15}$$

Eqs. (14) and (15) are definitional. Now, suppose that configuration $z \in i(CO|_{\alpha_1',\alpha_2'})$. Then clearly $z \notin F|_{\alpha_1',\alpha_2'}$. Hence we have

$$\begin{array}{ccccc}
H_{3,Gen}\Big|_{\alpha_1',\alpha_2'} & = & \underbrace{\left(F_{\theta_2,\Delta t}(H_1 - stick^*_{\theta_2}(H_1))\right)\Big|_{\alpha_1',\alpha_2'}} & - & \overline{CO|_{\alpha_1',\alpha_2'}}^c \\
\cap & & \cap & & \| \\
H_{3,Spec} & = & \overbrace{\left(\pi_2 H_2 \times (\alpha_1', \alpha_2')\right) - i(CO|_{\alpha_1',\alpha_2'})} & - & \overline{CO|_{\alpha_1',\alpha_2'}}^c .
\end{array}$$

□

Note that as a consequence, we may expect that *Spec* is less likely than *Gen* to terminate.

7 Weak EDR Theory, Strategy Equivalence, and the Linking Condition

7.1 Reachability and Recognizability Diagrams

We now introduce a type of diagram which permits notation of reachability and recognizability. These diagrams are a powerful tool for compactly expressing motion strategies. They greatly aid the development of concise and readable proofs.

Suppose we are given a start region R, a goal G, and a motion θ. We construct the EDR region H. Then under sticking termination, all motions from R will terminate in G or H. That is, the push-forward of the motion θ from R is contained in $G \cup H$:

$$\text{push}_\theta(F_\theta(R)) \subset G \cup H. \tag{16}$$

Whenever (16) is true, we write this by the following *reachability diagram*,

$$\begin{tikzcd} & G \\ R \arrow[ur, "\theta"] \arrow[dr, "\theta"'] & \\ & H. \end{tikzcd} \tag{17}$$

Suppose that G and H are distinguishable using sensors. Then θ is an EDR strategy from R, and we have

$$R = P_{\theta,R}(\{\, G, H \,\}). \tag{18}$$

Whenever (18) holds, we write this by the following *recognizability diagram*,

$$\begin{tikzcd} & G \\ R \arrow[ur, Rightarrow, "\theta"] \arrow[dr, Rightarrow, "\theta"'] & \\ & H. \end{tikzcd} \tag{19}$$

The reachability diagram (17) is an equivalent notation for the reachability termination condition (16). The recognizability diagram (19) is equivalent notation for the recognizability termination condition (18). Single arrows ($\longrightarrow$) denote reachability whereas double arrows ($\Longrightarrow$) denote recognizability. If and only if (16) is true, we say that the correspondingly reachability diagram (17) *holds*. If and only if (18) is true, we say that the correspondingly recognizability diagram (19) *holds*. A diagram is said to hold *tautologously* when it is true without additional conditions or suppositions.

The nice thing about sticking termination, as discussed in chapter II, is the following property:

Theorem: *Let R be a start region, θ a motion, and G a goal. Construct the EDR region H for R, θ, and G. Then with sticking termination the reachability diagram (17) holds tautologously.*

Now, in diagrams (17) and (19) we have labeled all the arrows. In the future, when this would clutter the diagrams, we will label only the top arrow and adopt the convention that all arrows aligned below it have the same label.

7.2 More General Push-Forwards

Hence the chief advantage with sticking termination is that (17) is always true. In this chapter, we will generally assume that either sticking termination is employed, or, if more general termination predicates are allowed, then the truth of the reachability diagram (17) can be determined through restrictions on time and history, as described in chap. We now digress briefly, however, to describe how this discussion generalizes for more general termination predicates.

In an appendix, we define a more general push-forward, $F_{*\theta}(R)$, which denotes all configurations at which the motion θ can terminate given more general termination predicates. When more general termination predicates than sticking are considered, then the condition (16) must be replaced by

$$F_{*\theta}(R) \subset G \cup H. \tag{20}$$

When (20) holds, we may then write the equivalent reachability diagram (17).

However, with more general termination conditions, (17) does not hold tautologously. For example, with time-termination and the approximate push-forward described in sec. 5.3, a motion could (*a priori*) terminate without sticking yet within the weak preimage. In such cases, it must be the responsibility of the planner to verify that all motions terminate in $G \cup H$.

The first difference between the sticking push-forward push$(\cdot)$ and the general push-forward $F_*(\cdot)$ is that $F_*(\cdot)$ depends on the start region for the motion, while push$(\cdot)$ does not. That is, $F_*(\cdot)$ depends on history (and possibly time) whereas push$(\cdot)$ does not.

Now, a ***motion sequence*** is a reachability or recognizability diagram of the form:

$$R_n \xrightarrow{\theta_n} R_{n-1} \xrightarrow{\theta_{n-1}} \cdots \xrightarrow{\theta_2} R_1 \xrightarrow{\theta_1} R_0 = G. \tag{21}$$

The second chief difference between the *a priori* sticking push-forward push$(\cdot)$ and the general push-forward $F_{\bullet}(\cdot)$ is that the action of push$(\cdot)$ on a motion sequence (21) is *functorial*, while $F_{\bullet}(\cdot)$ is not. The non-functoriality of $F_{\bullet}(\cdot)$ is a consequence of its history dependence.

7.3 Weak EDR Theory

We now make the following natural refinement of our termination predicate. Suppose the termination predicate is given some *finite* collection of goals $\{ G_\beta \}$ in a distinguishable union. Then the goals $\{ G_\beta \}$ are of course partially ordered by containment. We assume that the termination predicate returns the smallest goal (with respect to containment) if at termination time the actual configuration of the robot is known to lie within two or more goals. (A technical point: if two or more goals overlap, we augment the collection with a new goal which is their intersection).

Now, whenever the reachability diagram (17) holds (which it always does with sticking termination), then we have the following:

$$R = P_{\theta,R}(\{ G, H, G \cup H \}). \tag{22}$$

This is trivial to show; on termination, the termination predicate will return G or H if it can, otherwise it will return $G \cup H$. In particular, it will return G or H in preference to $G \cup H$.

Thus we can write the following recognizability diagram, which is equivalent to (22):

$$\begin{array}{ccc} & \overset{\theta}{\nearrow\!\!\!\!\nearrow} & G \\ R & \overset{\theta}{\Longrightarrow} & H \\ & \underset{\theta}{\searrow\!\!\!\!\searrow} & G \cup H. \end{array} \tag{23}$$

(23) is called the ***Weak EDR Recognizability Diagram for G, H, and θ***. (19) is called the ***Strong EDR Recognizability Diagram***. (17) is called the ***Reachability Diagram***.

Theorem: *Let R be a start region, θ a motion, and G a goal. Construct the EDR region H for R, θ, and G. Then with sticking termination the weak EDR diagram (23) holds tautologously.*

Up to now, in previous chapters, we have described the *strong EDR* theory. This section has introduced the weak EDR theory. It may not appear useful at first glance. However, in the next section we will see that these one-step weak EDR strategies—which are in effect always available—may under certain conditions be chained together to make a multi-step plan very like a strong EDR strategy.

The key idea behind the weak EDR theory is: given a collection of goals $\{G_\beta\}$ (possibly including H), we consider all unions of the subcollections to get some measure of weakest recognizability.

7.4 Strategy Equivalence

A one-step weak EDR strategy is not very interesting. In particular, we can always obtain one! Surprisingly, it is possible to define a way of coupling two weak, one-step EDR strategies together to make a two step strategy which has many of the characteristics of strong EDR. In particular, we will develop a way of making precise the idea that the two weak EDR steps can be combined to make a two-step strategy that is "equivalent" to a one-step strong EDR strategy.

Suppose the commanded motions of the two weak EDR steps are θ_1 and θ_2. The essence of this "equivalence" lies in disambiguating a previous motion's (θ_1's) result without destroying the goal state.

Now, let R be the start region, and G the goal as usual. Assume without loss of generality that G is contained within the forward projection of R under θ_1. Let

$$R_1 = R \cap P_{\theta_1, F_{\theta_1}(R)}(G). \tag{24}$$

Now, we have the recognizability diagrams

$$\underbrace{\begin{array}{ccc} R_1 & \xRightarrow{\theta_1} & G \\ & \nearrow\!\!\!\nearrow & \\ R - R_1 & \Longrightarrow & H \\ & \searrow\!\!\!\searrow & \\ & & H \cup G \end{array}}_{\textbf{recog } \theta_1} \quad \underbrace{\begin{array}{ccc} & & G \\ & \overset{\theta_2}{\nearrow\!\!\!\nearrow} & \\ \text{push}_{\theta_1}(G \cup H) & \Longrightarrow & H' \\ & \searrow\!\!\!\searrow & \\ & & H' \cup G \end{array}}_{\textbf{recog } \theta_2} \tag{25}$$

where H' is the EDR region for motion θ_2.

The question is, how can we link together motions θ_1 and θ_2 into a two-step EDR strategy? The first condition we require of such a two-step strategy is as follows: once θ_1 has reached G, θ_2 should preserve this state and "add" recognizability. That is, G is a "fixed-point" under θ_2. This is given by the following diagram:

Definition: *The* fixed-point diagram *is*

$$\text{push}_{\theta_1}(G) \xRightarrow{\theta_2} G. \tag{26}$$

When the fixed-point diagram (26) holds, (25) admits the following reachability and recognizability diagram:

$$
\begin{array}{ccc}
 & \mathrm{push}_{\theta_1}(G) & \overset{\theta_2}{\Longrightarrow} \quad G \\
 \overset{\theta_1}{\nearrow} & & \\
R & & \\
 \searrow & & \\
 & \mathrm{push}_{\theta_1}(H). &
\end{array}
\tag{27}
$$

It remains to ensure that good EDR behavior occurs when θ_2 is executed from $\mathrm{push}_{\theta_1}(H)$. Now, think of $\theta_1 * \theta_2$ as the composite strategy formed by executing motion θ_1 followed by θ_2. We wish to find additional conditions which, together with (25), will admit both the fixed-point diagram (26) and a strong EDR diagram,

$$
\begin{array}{ccc}
 & & G \\
 & \overset{\theta_1 * \theta_2}{\nearrow\!\!\!\nearrow} & \\
R & & \\
 & \underset{\theta_1 * \theta_2}{\searrow\!\!\!\searrow} & \\
 & & H'',
\end{array}
\tag{28}
$$

for some H'' (see below). Together with the weak EDR diagram (25) (which is tautologously true for sticking termination), the additional conditions below, which we will call the ***linking conditions***, are necessary and sufficient for defining an equivalence between two "linked" weak EDR strategies and a single-step strong EDR strategy, whose recognizability diagram is given by (19), (substituting θ_1 for θ). Henceforth, let $\theta = \theta_1$.

Definition: *If the fixed-point diagram (26) holds and if (25) admits a strong EDR diagram (28) in which*

$$
H'' = \{ H' \},
\tag{29}
$$

then the motion strategy $\theta_1 * \theta_2$ is said to be strongly equivalent *to a strong EDR strategy with recognizability diagram (19).*

An example of such a strategy is the two-step peg-in-hole insertion plan with model error, figs. 4-66.

Definition: *If the fixed-point diagram (26) holds and if (25) admits a strong EDR diagram (28) in which*

$$
H'' = \{ H', H' \cup G \},
\tag{30}
$$

then the motion strategy $\theta_1 * \theta_2$ is said to be weakly equivalent *to a strong EDR strategy with recognizability diagram (19).*

Note that we define (strong or weak) equivalence using (19) with $\theta = \theta_1$, not with $\theta = \theta_1 * \theta_2$. The reason for this is as follows. If $\theta_1 * \theta_2$ satisfies the weak equivalence condition (30) and the fixed-point diagram (26), then after termination, we are assured that the outcome of θ_1 has been completely diagnosed. That is, the run-time executor knows whether or not θ_1 terminated in success or failure. However, it is not necessarily true that the outcome of θ_2 is completely diagnosed. This occurs in the worst case, if $H' \cup G$ is recognizably attained. We discuss this point in some detail below.

The following gives an implicit definition of linking conditions:

Definition: *Let H'' be chosen for either strong or weak equivalence, as in (29) or (30). The* linking conditions *are necessary and sufficient conditions for (25) to admit a fixed-point diagram (26) and a strong EDR diagram (28).*

It remains to show, of course, that linking conditions exist for strong or weak equivalence. We will momentarily postpone the derivation of the linking conditions in order to describe what the linking should effect.

Once "linked," two one-step weak EDR plans should admit the strong EDR diagram (28). The claim is that (28) is in some sense "equivalent" to the strong EDR diagram (19). How is this possible?

(19) indicates that the run-time executor can disambiguate the success or failure of motion θ_1. The same is true of strategy $\theta_1 * \theta_2$ in (28). Here are the possible results of executing $\theta_1 * \theta_2$ when the steps θ_1 and θ_2 are properly "linked:"

1. G is achieved and recognized at termination. In this case, either (i) θ_1 achieved G and the run-time executive may not have recognized it, but θ_2 disambiguated the result while still terminating within G. Alternatively, (ii) θ_1 failed, reaching H, and θ_2 subsequently achieved G from H.

2. H' is achieved and recognized at termination. In this case, θ_1 is *known to have failed*, and the robot is known to be outside G.

(1) and (2) are the only outcomes given strong equivalence. With weak equivalence, a third outcome in also possible:

3. $G \cup H'$ is achieved and recognized at termination. In this case, θ_1 is *known to have failed.*

Thus the key is that θ_2 does not corrupt the goal state; that is, G is a fixed point under θ_2. The desirability of outcomes (1) and (2) are clear. One might ask, what good is weak equivalence? Why would anyone want outcome (3)? The answer is: in one-step strong EDR (19), the run-time executor can (a) disambiguate the result of motion θ_1, and (b) in case of failure, know that the robot is not in the goal. In weak equivalence, we have (a) but not (b). That is, in outcome (3), we have completely diagnosed the result of motion θ_1, although in the process, we may have accidentally moved into the goal. That is, we may indicate failure when we have, in fact, succeeded. However, we will never indicate success unless it is certain. In short, when linked, $\theta_1 * \theta_2$ is "conservative" about declaring success.

7.5 The Linking Conditions

We now derive the linking conditions. Let

$$\begin{aligned} F_{\theta_1} &= F_{\theta_1}(R) \\ R_1 &= R \cap P_{\theta_1, F_{\theta_1}}(G) \\ \mathbf{push}_{\theta_1} &= \mathbf{push}_{\theta_1}(G \cup H) \\ F_{\theta_2} &= F_{\theta_2}(\mathbf{push}_{\theta_1}) \\ R_2 &= \mathbf{push}_{\theta_1} \cap P_{\theta_2, F_{\theta_2}}(G). \end{aligned}$$

The overloading notation for $\mathbf{push}_{\theta_1}$ is symmetric with that for preimages and forward projections: both the map and its image are denoted by the same symbol. The discussion of linking conditions assumes sticking termination. However, the derivation goes through *mutatis mutandis* for more general termination conditions, if we let

$$\mathbf{push}_{\theta_1} = F_{\bullet\theta_1}(R).$$

It remains, however, to extend the linking-conditions for time-indexed forward projections.

We now demonstrate our claim that linking conditions exist.

Definition: *The condition* **(L0)** *is*

$$G \cap \mathbf{push}_{\theta_1} \subset R_2. \tag{$L0$}$$

Here is the motivation behind (L0). (L0) says that whenever motion θ_1 terminates in the goal G, then the state is inside the preimage of G under the next motion θ_2. The intent of (L0) is to admit the fixed-point diagram (26).

Claim: *(L0) implies the fixed-point diagram (26).*
Proof: The preimage equation for (26) is

$$P_{\theta_2, \mathbf{push}_{\theta_1}(G)}(G) = \mathbf{push}_{\theta_1}(G).$$

This preimage is taken with respect to a smaller start region than R_2. □

Note however that the converse is false. (L0) is stronger than the fixed-point diagram (26), since the preimage R_2 is taken with respect to the entire forward projection under θ_2.

Claim: *Linking conditions exist, and, in particular, (L0) is a linking condition.*
Proof: Suppose (L0) holds. This yields the following reachability and recognizability diagram:

$$\underbrace{\begin{array}{ccc} R_1 & \xrightarrow{\theta_1} & G \\ & \nearrow & \\ R - R_1 & \longrightarrow & H \end{array}}_{\text{reachability}} \quad \begin{array}{c} \supset \\ \\ \supset \end{array} \quad \begin{array}{ccc} \overbrace{G \cap \mathrm{push}_{\theta_1} \subset R_2}^{\text{linking condition (L0)}} \\ \\ H \cap \mathrm{push}_{\theta_1} \subset \mathrm{push}_{\theta_1} \end{array} \quad \begin{array}{c} = \\ \\ \supset \end{array} \quad \underbrace{\begin{array}{ccc} R_2 & \xrightarrow{\theta_2} & G \\ & \nearrow & \\ \mathrm{push}_{\theta_1} - R_2 & \longrightarrow & H' \\ & \searrow & \\ & & H' \cup G \end{array}}_{\text{recognizability}} \tag{31}$$

To see that diagram (31) demonstrates weak equivalence, we use a technique like "diagram chasing" (see, eg., [Hungerford]). Assume (L0) holds. Starting from R_1, θ_1 effects a motion reaching G. This motion in fact terminates in $G \cap \mathrm{push}_{\theta_1}$. Since by (L0) $G \cap \mathrm{push}_{\theta_1}$ is within R_2, θ_2 then effects recognizable termination in G.

On the other hand, if the motion begins in $R - R_1$, then θ_1 effects a motion reaching either G or H. If G is reached, then θ_2 will eventually effect recognizable termination in G, by the argument immediately above. If H has been reached, then the motion θ_1 will in fact terminate at some point z in $H \cap \mathrm{push}_{\theta_1}$. Then there are two cases. Case (i): $z \in R_2$. Since the preimage R_2 is constructed with respect to the entire forward projection of push_{θ_1}, motion θ_2 will next effect recognizable termination in G. Case (ii): $z \notin R_2$. In this case, motion θ_2 will effect recognizable termination in one of $\{ G, H, H' \cup G \}$.

We conclude the process by "forgetting" all the intermediate steps, and renaming them to $\theta_1 * \theta_2$. First, observe that the fixed-point diagram (26) holds. Next, to see that (31) admits an EDR diagram (28) in which (30) holds, we remember only the start region R and the "results" G, H', and $H' \cup G$. Diagram chasing shows that these may be joined with recognizability arrows as in (28).

Thus the diagram (31) demonstrates weak equivalence. For strong equivalence, we remove $H' \cup G$ as an outcome of θ_2. Note that the linking condition is not a tautology. However, note that all the other subset relations and the equality in (31) are tautologous.
□

In the future, we will leave similar diagram-chasing arguments to the reader. We may thus conclude that

Theorem: *The linking condition (L0) is a necessary and sufficient condition for weak equivalence of $\theta_1 * \theta_2$ to a one-step strong EDR strategy.*
Proof: The claims above have demonstrated sufficiency. It remains to show (L0) is necessary. Suppose (L0) is false, but (26) still holds. (This is the interesting case, for if (26) does not hold, then equivalence cannot possibly follow). (26) says that when

the motion is known to start within $\text{push}_{\theta_1}(G)$, then it can be guaranteed to terminate recognizably in G. The antecedent is a precondition for success of the motion. After θ_1, however, this precondition may be false: even if θ_1 reaches G, it is only *known* to have reached push_{θ_1}. In particular, (26) says nothing about what happens when θ_2 is executed from H. (L0), on the other hand, says that termination in G can be recognized no matter where θ_2 originates in push_{θ_1}. □

Now, we can derive some equivalent linking conditions that are somewhat simpler in form. Let

$$R_2^* = R_2 \cap G.$$

Definition: *The* linking conditions **(L1)** *and* **(L2)** *are*

$$\begin{array}{rclr} G \cap \text{push}_{\theta_1} & = & R_2^* & (L1) \\ H \cap \text{push}_{\theta_1} & = & \text{push}_{\theta_1} - R_2^* & (L2) \end{array}$$

These linking conditions admit the reachability and recognizability diagram

$$\begin{array}{ccccccc} & & & & \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{\text{linking conditions (L1), (L2)}} & & \\ R_1 & \xrightarrow{\theta_1} & G & \supset & G \cap \text{push}_{\theta_1} \quad = \quad R_2^* & \xRightarrow{\theta_2} & G \\ & \nearrow & & & & \nearrow & \\ R - R_1 & \longrightarrow & H & \supset & H \cap \text{push}_{\theta_1} \quad = \quad \text{push}_{\theta_1} - R_2^* & \Longrightarrow & H' \\ & & & & & \searrow & \\ & & & & & & H' \cup G \\ \multicolumn{3}{c}{\underbrace{\qquad\qquad\qquad\qquad}_{\text{reachability}}} & & & \multicolumn{2}{c}{\underbrace{\qquad\qquad\qquad}_{\text{recognizability}}} \end{array} \tag{32}$$

Comments: Let

$$P_{\theta_2} = P_{\theta_2, F_{\theta_2}(\text{push}_{\theta_1}(G))}(G),$$

so $R_2 = \text{push}_{\theta_1} \cap P_{\theta_2}$. Note that (L2) is not tautologous, for we can have $x \in G$, $x \notin P_{\theta_2}$ if (L1) is false. Therefore $x \in \text{push}_{\theta_1} - R_2^*$ and $x \notin \text{push}_{\theta_1} \cap H$.

Lemma. *The linking conditions (L1) and (L2) are equivalent.*

Proof: (L1) implies (L2). Suppose (L1). Let $x \in H \cap \text{push}_{\theta_1}$. $x \in R_2^*$ implies $x \in G$. Therefore $x \notin H$ is a contradiction. Therefore $x \in \text{push}_{\theta_1} - R_2^*$.

Now let $x \in \text{push}_{\theta_1} - R_2^*$. Therefore $x \notin G \cap \text{push}_{\theta_1} \cap P_{\theta_2}$. Therefore $x \notin G$ or $x \notin P_{\theta_2}$. In the former case, $x \in H$. In the latter, suppose that $x \in \text{push}_{\theta_1}$ and $x \in G$ and $x \notin P_{\theta_2}$. But by (L1), $x \in G \cap \text{push}_{\theta_1}$ implies $x \in P_{\theta_2}$, a contradiction.

(L1) if (L2). Let $x \in G \cap \text{push}_{\theta_1}$. Show $x \in R_2^*$. We need only show that $x \in P_{\theta_2}$. Now, $x \notin P_{\theta_2}$ implies $x \in H \cap \text{push}_{\theta_1}$, a contradiction. Now let $x \in R_2^*$. Therefore, $x \in G \cap \text{push}_{\theta_1}$. □

Lemma: *The linking conditions (L0) and (L1) are equivalent.*
Proof: (L0) implies (L1). Suppose (L0), i.e., $G \cap \text{push}_{\theta_1} \subset R_2$. Show $G \cap \text{push}_{\theta_1} = R_2^\bullet = G \cap R_2$.

Let $x \in G \cap \text{push}_{\theta_1}$. Now, (L0) implies that $x \in R_2$. Therefore $x \in G$ and $x \in R_2$. Hence $x \in R_2^\bullet$.

Let $x \in R_2^\bullet$. Therefore $x \in G \cap R_2$. Hence $x \in G \cap \text{push}_{\theta_1} \cap P_{\theta_2}$, i.e., $x \in G \cap \text{push}_{\theta_1}$.

(L0) if (L1) is trivial. $\square$

Theorem: *The following linking conditions are equivalent:*

$$\begin{array}{rcll} G \cap \text{push}_{\theta_1} & \subset & R_2 & (L0) \\ G \cap \text{push}_{\theta_1} & = & R_2^\bullet & (L1) \\ H \cap \text{push}_{\theta_1} & = & \text{push}_{\theta_1} - R_2^\bullet & (L2) \end{array}$$

7.6 ∗ Beyond the Fixed-Point Restriction

In the discussion above, we have required that the goal was a fixed point under motion θ_2. We now discuss how to relax this restriction. In particular, it is possible to extend the notions of strategy equivalence, and the linking conditions, to the case where a subgoal G_1 is in fact the preimage of the actual, or final goal, G_0, under θ_2. Thus G_1 is no longer the fixed point of θ_2, but rather the *preimage* of G_0. This section is somewhat technical and may be skipped at first reading. We regard relaxing the fixed-point restriction as a digression. The subsequent material may be understood even if this section is omitted, however, the reader may wish to bear in mind that such a generalization does, in fact, exist.

We consider the situation where from R, θ_1 may attain G_0 or G_1, where "$G_1 = P_{\theta_2}(G_0)$." However, G_1 may not be distinguishable from G_0 under θ_1. Thus the three reachability results of θ_1 are G_0, G_1, or H_1, where H_1 is the EDR region for θ_1 when we view the goal as $G_0 \cup G_1$.

To define strategy equivalence in the non-fixed-point case, we first generalize the fixed-point diagram (26) as follows.

Definition: *The* generalized fixed-point diagram *is*

$$\text{push}_{\theta_1}(G_0 \cup G_1) \quad \stackrel{\theta_2}{\Longrightarrow} \quad G_0. \tag{33}$$

Next, we modify the definitions of strategy equivalence and the linking conditions to require that the generalized fixed-point diagram (33) hold in place of the old fixed-point diagram (26). To avoid confusion, we will call (26) the *simple* fixed-point condition.

Now, we let

$$\begin{aligned}
R_1 &= R \cap P_{\theta_1, F_{\theta_1}}(G_0 \cup G_1) \\
\text{push}_{\theta_1} &= F_{*\theta_1}(R) \\
F_{\theta_2} &= F_{\theta_2}(\text{push}_{\theta_1}) \\
P_{\theta_2} &= P_{\theta_2, F_{\theta_2}}(G_0) \\
R_2 &= \text{push}_{\theta_1} \cap P_{\theta_2}.
\end{aligned}$$

Next, define

$$R_2^j = G_j \cap R_2, \qquad (j = 0, 1)$$

It is possible to generalize the definition of R_2^j and the linking conditions to more than two subgoals $\{ G_j \}$. We would do this by writing $(\forall j)$ in place of $(j = 0, 1)$.

We already know one linking condition:

$$P_{\theta_2} \supset G_0 \cup G_1. \qquad (L3)$$

In addition, we can derive the following linking conditions. Recall H_1 is the EDR region for motion θ_1, viewing the goal of θ_1 as $G_0 \cup G_1$.

$$\begin{aligned}
\text{push}_{\theta_1} \cap G_j &= R_2^j & (\forall j) \quad (L1') \\
\text{push}_{\theta_1} \cap H_1 &= \text{push}_{\theta_1} - \textstyle\bigcup_j R_2^j. & (L2')
\end{aligned}$$

Comments: Clearly we have $(L1')$ implies $(L2')$. However I have not been able to prove the converse true. I suspect it is false, since G_0 may intersect G_1, and H_2, the EDR region for θ_2, may intersect G_1, etc.

Finally, note that all ***three*** linking conditions, $(L1', L2', L3)$ are required for the composition $\theta_1 * \theta_2$ to admit an equivalent strong EDR diagram. This points out the chief theoretical advantage of strategy equivalence with the simple fixed-point condition (26). With the ***simple*** fixed-point condition, the linking conditions (L0), (L1) and (L2) were found to be equivalent. With the ***generalized*** fixed point condition (33), not only do the corresponding linking conditions $(L1')$ and $(L2')$ appear to be inequivalent, but we also require the additional independent condition $(L3)$. While it is gratifying that our key concept—composing two weak EDR strategies via linking conditions to admit strategy equivalence—in fact generalized to the non-fixed-point case, the generalization, unfortunately, is correspondingly more complicated.

7.7 What Good is Weak Equivalence?

We now pose the following question. Why is

$$
\begin{tikzcd}
R_1 \arrow[r, Rightarrow, "\theta_1 * \theta_2"] & G \\
R - R_1 \arrow[r, Rightarrow] \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & H' \\
& G \cup H'
\end{tikzcd}
\tag{34}
$$

any better than

$$
\begin{tikzcd}
R_1 \arrow[r, Rightarrow, "\theta_1"] & G \\
R - R_1 \arrow[r, Rightarrow] \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & H \\
& G \cup H
\end{tikzcd}
\tag{35}
$$

?

(35) is simply the weak EDR diagram for motion θ_1. It always holds (given the reachability diagram). (34) is the equivalent recognizability diagram for $\theta_1 * \theta_2$ when a linking condition is satisfied. That is, (34) is obtained through weak equivalence. Why is (34) stronger than (35), and would one prefer (34) to (35)?

Here is our answer. $\text{push}_{\theta_1}(G)$ is a fixed-point of θ_2. Therefore, nothing is "lost" by θ_2. θ_2 serves to disambiguate the result of θ_1, without polluting the state. Second, note that $\theta_1 * \theta_2$ is "conservative" about declaring success. It is as if we used θ_2 to convert the reachability diagram

$$
\begin{tikzcd}
R_1 \arrow[r, "\theta_1"] & G \\
R - R_1 \arrow[r] \arrow[ur] & H
\end{tikzcd}
\tag{36}
$$

into the recognizability diagram

$$
\begin{tikzcd}
R_1 \arrow[r, Rightarrow, "\theta_1 * \theta_2"] & G \text{ "Win"} \\
R - R_1 \arrow[r, Rightarrow] \arrow[ur, Rightarrow] & \text{"Lose, but knowing } \theta_1 \text{ did not achieve } G\text{."}
\end{tikzcd}
\tag{37}
$$

More precisely, the "lose" states are

H' $\approx$ θ_1 did not achieve G, and now the robot is outside of G.

$G \cup H'$ $\approx$ θ_1 did not achieve G, and now we might be in H', but can't guarantee that we're outside of G.

On the other hand (35), achieving $G \cup H$ after θ_1 only tells us that we started in $R - R_1$, and does not tell us the result of motion θ_1.

7.8 Application: Failure Mode Analysis in the Gear Experiment

We now discuss how the failure mode analysis used to generate motion θ_2 in the gear domain may be viewed using the weak EDR theory.

In the gear meshing plan, θ_1 is a pure translation, and θ_2 is a pure rotation. The goal is a fixed point under θ_2. Consider (32). In the gear plan, the reachability arc

$$\begin{array}{ccc} & & G \\ & \overset{\theta_1}{\nearrow} & \\ R - R_1 & & \end{array} \tag{38}$$

is present, but the arc

$$\begin{array}{ccc} & & G \\ & \overset{\theta_2}{\Nearrow} & \\ \text{push}_{\theta_1} - R_2^* & & \end{array} \tag{39}$$

is not. That is, it is possible to serendipitously achieve the goal under translation but not rotation. The linking conditions are satisfied. Now, is the outcome $G \cup H'$ possible? Failure mode analysis yields the answer: No. In this case, $\theta_1 * \theta_2$ is ***strongly*** equivalent to a one-step strong EDR strategy

$$\begin{array}{ccc} R_1 & \overset{\theta_1 * \theta_2}{\Longrightarrow} & G \\ & \Nearrow & \\ R - R_1 & \Longrightarrow & H'. \end{array}$$

The full reachability and recognizability diagram for the gear plan is given by

$$\begin{array}{cccccccccc} R_1 & \overset{\theta_1}{\longrightarrow} & G & \supset & \overbrace{G \cap \text{push}_{\theta_1} \quad = \quad R_2^*}^{\text{linking conditions (L1), (L2)}} & & \overset{\theta_2}{\Longrightarrow} & G \\ & \nearrow & & & & & & \\ R - R_1 & \longrightarrow & H & \supset & H \cap \text{push}_{\theta_1} \quad = \quad \text{push}_{\theta_1} - R_2^* & & \Longrightarrow & H' \\ \multicolumn{3}{c}{\underbrace{\qquad\qquad\qquad\qquad}_{\text{reachability}}} & & & & \multicolumn{2}{c}{\underbrace{\qquad\qquad}_{\text{recognizability}}} \end{array} \tag{40}$$

7.9 Discussion and Review

We now discuss the relationship between push-forward algorithms, failure-mode analysis, and the weak EDR theory. Recall the diagram (32):

$$
\begin{tikzcd}[column sep=small]
R_1 \arrow[r, "\theta_1"] & G & \supset & G \cap \mathrm{push}_{\theta_1} & = & R_2^{*} \arrow[rr, Rightarrow, "a"] & & G \\
R - R_1 \arrow[r] \arrow[ur, dotted] & H & \supset & H \cap \mathrm{push}_{\theta_1} & = & \mathrm{push}_{\theta_1} - R_2^{*} \arrow[urr, Rightarrow, "b"] \arrow[rr, Rightarrow, "c"] \arrow[drr, Rightarrow, "d"'] & & H' \\
 & & & & & & & H' \cup G
\end{tikzcd}
\tag{41}
$$

$\overbrace{\hspace{6em}}^{\text{linking conditions (L1), (L2)}}$ over $G \cap \mathrm{push}_{\theta_1} = R_2^{*}$ and $H \cap \mathrm{push}_{\theta_1} = \mathrm{push}_{\theta_1} - R_2^{*}$; $\underbrace{\hspace{6em}}_{\text{reachability}}$ under $R_1 \xrightarrow{\theta_1} G$, $R - R_1 \longrightarrow H$; $\underbrace{\hspace{6em}}_{\text{recognizability } \theta_2}$ under arrows (a)–(d) to G, H', $H' \cup G$.

(41) is the full reachability and recognizability diagram for weak equivalence. The arrows (a)-(d) all correspond to motion θ_2; we have labeled them so as to be able to refer to them in the discussion.

Failure Mode Analysis. The reachability and recognizability diagram for failure mode analysis (40) is found by deleting arcs (b) and (d) from (41). In LIMITED, arc (a) is essentially a user input[25]. The failure mode analysis algorithms *Spec* and *Gen* decide arc (c). Thus, in sec. 6.2, (c) corresponds to (F1). Failure mode analysis links a weak EDR strategy θ_1 followed by a strong EDR strategy θ_2. (a) warrants that G is a fixed-point under θ_2. (b) ensures that failure is preserved under θ_2: no serendipitous goal achievement from H is possible. Thus such plans are pure ***disambiguation*** strategies.

Push-Forward Algorithms. Plans found by push-forward algorithms such as *Multi* admit a diagram from (41) containing arcs (a), (b), and (c), but not containing (d). The arc (b) (which is shown in detail in eq. (39)) permits serendipitous goal achievement from H under θ_2. The absence of arc (d) yields strong equivalence. Again, push-forward algorithms link a weak EDR strategy followed by a strong one. They differ from failure mode analysis plans in that the arc (b) is permissible, and (a) is not a user input. The peg-in-hole plan with model error (figs. 4–66) is an example of such a plan.

2-Step Weak EDR. A plan admitting the diagram (41) with all four arcs (a)–(d) demonstrates weak equivalence. It is formed by linking together two weak EDR strategies into a 2-Step plan. We have discussed the semantics of such plans above. The key differences between 2-step weak EDR plans and push-forward or failure-mode plans are (1) the existence of arc (d), and (2) the linking of 2 weak (as opposed to a weak and a strong) EDR strategies.

In all cases, note that the linking conditions are required. Thus the linking conditions have somewhat surprisingly turned out to be the underlying characterization for multi-step EDR strategies. That is, since they are necessary and sufficient conditions for

[25] **Although we have discussed methods for inferring (a) computationally, this is really a direction for future work rather than a focus of this research.**

constructing multi-step EDR plans, the linking conditions may, in fact, be taken as the *definition* of multi-step EDR strategies.

Hence in considering LIMITED's techniques for multi-step strategy generation, we find that both failure model analysis and push-forward algorithms are essentially special cases of the Weak EDR theory. This is summarized in the table below:

Method	Arcs in (41)	Strategy Type	Comments
Failure Mode Analysis	a,c	weak∗strong	**Pure Disambiguation.** **(a) is user input, (c) is computed.**
Push-Forwards	a,b,c	weak∗strong	**(b) permits serendipitous goal achievem**
Weak EDR	a,b,c,d	weak∗weak	**2-Step Weak EDR.**

7.9.1 Algebraic Considerations

Let us pause and review the key points in this development. Weak EDR theory, strategy equivalence, and the linking conditions were introduced as a unifying framework for planning multi-step strategies.

1. The linking conditions are necessary and sufficient criteria for admitting the composition of two weak EDR strategies θ_1^w and θ_2^w into a two-step strategy which is weakly equivalent to a one-step strong EDR strategy from θ_1. We may write this as

$$\theta_1^w * \theta_2^w \overset{w}{\simeq} \theta_1^s \tag{42}$$

2. The linking conditions are necessary and sufficient criteria for admitting the composition of a weak EDR strategy θ_1^w and a strong EDR strategy θ_2^s into a two-step strategy which is strongly equivalent to a one-step strong EDR strategy from θ_1. We may write this as

$$\theta_1^w * \theta_2^s \overset{s}{\simeq} \theta_1^s \tag{43}$$

3. The gear plan is a special case of (2). In particular:

4. Failure mode analysis is a special case of satisfying the linking conditions to render a two-step EDR strategy strongly equivalent to a one-step strong EDR strategy.

5. Multi-step strategies may also be planned, by repeatedly pushing forward. This was the gist of algorithm *Multi* in the beginning of this chapter. *Multi* may be viewed as

chaining together weak EDR strategies followed by a strong EDR strategy. *Multi* is also essentially a special case of (2), with the goal fixed-point condition relaxed.[26]

8 Conclusions

The focus of this paper lies in the synthesis of multi-step fine-motion strategies in the presence of uncertainty. We began by reviewing a basic theory of planning. This theory—called the EDR theory—had two chief components. The first is a technique for planning compliant motion strategies in the presence of model error. The second is a precise, geometrical charaterization 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.

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.

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.

The Weak EDR theory introduced a new mathematical tools for studying multi-step strategies—reachability and recognizability diagrams, strong and weak strategy equivalence, linking conditions, and strategy composition. A variety of techniques for planning multi-step EDR strategies were investigated and unveiled as special cases of the Weak EDR theory. In particular, we discussed the design and implementation of push-forward algorithms, and failure-mode analysis. We tested the EDR theory by implementing a planner, LIMITED, and running experiments to have LIMITED automatically synthesize EDR strategies in the domain of planar assemblies.

Much work remains to be done in developing the theory and practice of EDR. Future research can extend the EDR framework—for example, by more sophisticated dynamic models, or by considering probabilistic strategies. In addition, further analysis of the existing framework is required. Towards this end, in [Donald, 87], 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 showed a case where $n = 2$, $k = 1$ and containment in the backprojection could be computed in poly-

[26] Relaxing this restriction was discussed in the section "Beyond the Fixed Point Restriction," above.

nomial time (note for $n = 3$, $k = 0$, this is false [CR]). We also investigated the structure of the non-directional backprojection in the plane. By applying results from computational algebra, it led to a polynomial-time algorithm for computing one-step (guaranteed) strategies, and a roughly singly-exponential algorithm[27] for multi-step strategies.

In addition to structural and algorithmic extensions, a number of other research directions deserve further attention:

- The weak EDR theory, while still in its infancy, has already yielded some interesting results and a fairly clean mathematical framework for studying multi-step strategies. The key idea behind the weak EDR theory is: given a collection of goals $\{ G_\beta \}$ (possibly including H), we consider all unions of the subcollections to get some measure of weakest recognizability. This is perhaps the most exciting theoretical area for future work.

- When rotations and compliant motion are allowed, we do not know of exact algorithms, even in principle, for computing projection sets. For example, the computation of forward projections is not immediately decidable within the theory of real closed fields. This is because the physics of motion are essentially specified "differentially," that is, by a mapping that sends a configuration $x \in \mathcal{G}$ and a commanded motion $\theta \in S^n$ (where $n+1$ is the dimension of C), to a cone $B_c(x, \theta)$ in the tangent space:[28]

$$\begin{array}{ccc} \mathcal{G} \times S^n & \rightarrow & \text{cones in } T\mathcal{G} \\ (x, \theta) & \mapsto & B_c(x, \theta). \end{array}$$

  Thus we have a differential specification of the possible motions $B_c(x, \theta)$ at each point x. The cones at each point specify a parametric family of vector fields—a field of cones to be precise. The integral curves for this family, however, may not be algebraic in general. Good approximate algorithms are needed to construct bounding algebraic envelopes about the image of this family of curves. For example, assuming that an integral curve has a power series, it is possible to construct a recurrence relation for the coefficients of the series. They can be generated deterministically to the accuracy desired. Randy Brost[29] has investigated other numerical techniques for constructing integral curves corresponsing to trajectories in the forward projection. This is an interesting area for future research. In particular, it could be applied to the "full" 4-dimensional gear meshing problem where a commanded pure rotation of the gripped gear could induce translations or rotations of either gear. Such algorithms might also be applied to compute projection sets under different dynamics.

[27] For an environment of n edges, the existence of an r-step strategy can be decided in time $n^{r^{O(1)}}$.
[28] The space of "cones in $T\mathcal{G}$" can be formalized as an appropriate tensor bundle over $\mathcal{G}$.
[29] [Personal Communication].

9 Review of Previous Work

This paper is based on [Donald, 87]. Broadly speaking, previous work falls the following categories: Algorithmic motion planning, Compliant motion planning with uncertainty, Model Error, and Error detection and recovery.

9.1 Algorithmic Motion Planning

In algorithmic motion planning, (also called the piano movers' problem, or the find-path problem) the problem is to find a continuous, collision-free path for a moving object (the robot) amidst completely known polyhedral or semi-algebraic obstacles. It is assumed than once such a path is found, it can be reliably executed by a robot with perfect control and sensing. Many algorithms employ configuration space, [Lozano-Pérez, Arnold, Abraham and Marsden, Udupa]. [Lozano-Pérez and Wesley] proposed the first algorithms for polygonal and polyhedral robots and obstacles without rotations. These results were later extended by [Lozano-Pérez 81, 83] to polyhedral robots which could translate and rotate. [Brooks 83] designed a find-path algorithm based on a generalized-cone representation of free-space. Brooks later extended this method for a revolute-joint robot. [Donald 84,85,87] developed a motion-planning algorithm for a rigid body that could translate and rotate with six degrees of freedom amidst polyhedral obstacles (the so-called "classical" movers' problem). [Lozano-Pérez 85] reported another 6DOF algorithm for 6-link revolute manipulators. [Canny 85] developed an algebraic formulation of the configuration-space constraints, which led to a very clean collision-detection algorithm. All of these algorithms have been implemented.

There are many theoretical results on upper and lower bounds for the find-path problem, see [Yap] for a good survey article. These results begin with [Lozano-Pérez and Wesley], who give the first upper bounds: they give efficient algorithms for planning in 2D and 3D in the absence of rotations. [Reif 79] obtained the first lower bounds, demonstrating the problem to be $\mathcal{PSPACE}$-hard when the number of degrees of freedom are encoded in the input specification of the problem. [Hopcroft, Joseph, and Whitesides] and [Hopcroft, Schwartz, and Sharir] have also given intersesting lower bounds for motion planning. [Schwartz and Sharir] gave a very general theoretical algorithm for motion planning via a reduction to the theory of real closed fields. The algorithm is doubly-exponential in the degrees of freedom, but polynomial in the algebraic and geometric complexity of the input. Over the next five years, there were many papers reporting more efficient special-purpose motion planning algorithms for certain specific cases; see [Yap] for a survey. To date the fastest general algorithm is due to [Canny, 87], who gives a generic motion planning algorithm which is merely singly-exponential in the degrees of freedom. For a motion planning problem of algebraic complexity d, geometric complexity n, and with r degrees of freedom, Canny's algorithm runs in time $(d^{O(r^2)}n^r \log n)$ which is within a log factor of optimal. While none of these theoretical algorithms have been implemented, Canny's is conjectured to be efficient in practice as well.

One might ask whether exact algorithms for motion plannning can ever be utilized after uncertainty in sensing and control are introduced. The answer is a qualified "yes." In particular, the Voronoi diagram has proved to be useful for motion planning among a set of obstacles in configuration space (see [Ó'Dúnlaing and Yap 82; Ó'Dúnlaing, Sharir, and Yap 84; Yap 84], and the textbook of [Schwartz and Yap 86] for an introduction and review of the use of Voronoi diagrams in motion planning). The Voronoi diagram, as usually defined, is a ***strong deformation retract*** of free space so that free space can be continuously deformed onto the diagram. This means that the diagram is complete for path planning, i.e. Searching the original space for paths can be reduced to a search on the diagram. Reducing the dimension of the set to be searched usually reduces the time complexity of the search. Secondly, the diagram leads to robust paths, i.e. paths that are maximally clear of obstacles. Hence Voronoi-based motion planning algorithms are relevant to motion planning with uncertainty. [Canny and Donald] define a "Simplified Voronoi Diagram" which is still complete for motion planning, yet has lower algebraic complexity than the usual Voronoi diagram, which is a considerable advantage in motion planning problems with many degrees of freedom. Furthermore, the Simplified diagram is defined for the 6D configuration space of the "classical" movers' problem. For the 6DOF "classical" polyhedral case, [Canny and Donald] show that motion planning using the Simplified diagram can be done it time $O(n^7 \log n)$.

Many additional robotics issues are discussed in [Paul; Brady *et al.*].

9.2 Compliant Motion Planning with Uncertainty

This section reviews previous work on planning compliant motions which are guaranteed to succeed even when the robot system is subject to sensing and control uncertainty. All of this work assumes perfect geometric models of the robot and obstacles.

Work on compliant motion can be traced to [Inoue, Whitney, Raibert and Craig, Salisbury]. This work in force control attempted to use the geometric constraints to guide the motion. By cleverly exploiting the task geometry, placements far exceeding the accuracy of pure position control can be achieved. [Mason 83] develops spring and damper compliance models, and gives an extensive review of research in compliant motion. [Simunovic, Whitney, Ohwovoriole and Roth, Ohwovoriole, Hill and Roth] have all considered frictional constraints, as well as jamming and wedging conditions. [Erdmann], [Burridge, Rajan and Schwartz] have considered algorithmic techniques for predicting reaction forces in the presence of friction. [Caine] has considered manual techniques for synthesizing compliant motion strategies, generalizing the methods of [Simunovic, Whitney]. [Mason, 82] has developed a way to model pushing and grasping operations in the presence of frictional contact. [Peshkin] has extended this work. [Brost] has further developed techniques for predicting pushing and sliding of manipulated objects to plan squeeze-grasp operations. In addition, Brost is currently investigating the application of

EDR techniques to the squeeze-grasp domain.

Early work on planning in the presence of uncertainty investigated using skeleton strategies. [Lozano-Pérez 76] proposed a task-level planner called LAMA which used geometric simulation to predict the outcomes of plans, and is one of the earliest systems to address EDR planning. [Taylor] used symbolic reasoning to restrict the values of variables in skeleton plans to guarantee success. [Brooks 82] later extended this technique using a symbolic algebra system. [Dufay and Latombe] implemented a system which addresses learning in the domain of robot motion planning with uncertainty.

[LMT] proposed a formal framework for automatically synthesizing fine-motion strategies in the presence of sensing and control uncertainty. Their method is called the *preimage* framework. [Mason, 83] further developed the preimage termination predicates, addressing completeness and correctness of the resulting plans. [Erdmann] continued work on the preimage framework, and demonstrated how to separate the problem into questions of *reachability* and *recognizability*. He also showed how to compute preimages using backprojections, which address reachability alone, and designed and implemented the first algorithms for computing backprojections. [Erdmann and Mason] developed a planner which could perform sensorless manipulation of polygonal objects in a tray. Their planner makes extensive use of a representation of friction in configuration space [Erdmann]. [Buckley] implemented a multi-step planner for planning compliant motions with uncertainty in 3D without rotations. He also developed a variety of new theoretical tools, including a combined spring-damper dynamic model, 3D backprojection and forward projection algorithms, and a finitization technique which makes searching the space of commanded motions more tractable.

[Hopcroft and Wilfong] addressed the problem of planning motions in contact, and proved important structural theorems about the connectivity of the 1-edges of configuration space obstacle manifolds. [Koutsou] has suggested a planning algorithm which plans along 1-edges. Other planning systems for compliant motion have been developed by [Turk], who used backprojections, [Laugier and Theveneau], who use an expert system for geometric reasoning about compliant motion, and [Valade].

Recently, there has been some theoretical work on the complexity of robot motion planning with uncertainty. [Erdmann] showed the problem to be undecidable when the obstacles are encoded as a recursive function on the plane. [Natarajan] has shown the problem to be $\mathcal{PSPACE}$-hard in 3D for finite polyhedral obstacles. [Canny and Reif] have demonstrated that in 3D the problem of synthesizing a multi-step strategy is hard for non-deterministic exponential time; in addition, they proved that verifying a 1-step strategy is $\mathcal{NP}$-hard.

9.3 Model Error

There is relatively little previous work on planning in the presence of model uncertainty. [Requicha] and [Shapiro] address representational questions of how to model part tolerances, and mathematical models for variational families of parts. [Buckley] considers some extensions of his planner to domains with model uncertainty. [Brooks 82] developed a symbolic algebra system which can constrain the variable values in skeleton plans, and introduce sensing and motion steps to reduce these values until the error ranges are small enough for the plan to be guaranteed. Some of the variables in these plans can represent model error—particularly, the position of objects in the workspace—and hence his planner can reason about motion planning in the presence of model uncertainty.

Work on manipulator pushing and sliding [Mason, Peshkin] and squeeze-grasping [Brost] may be viewed as addressing model error where the error parameters are the position and orientation of the manipulated part. The *operation space* of [Brost] is a clever example how to model actions with uncertain effects, and objects with uncertain orientation, in the same space. [Durrant-Whyte] considers how to model geometric uncertainty probabilistically, and how to propagate such information in applications related to motion planning.

[Lumelsky] considers the following problem: suppose that a robot has a 2D configuration space, perfect control and sensing, the obstacles are finite in number, and each obstacle boundary is a homeomorphic image of the circle. Then a collision free-path may be found by tracing around the boundary of any obstacles encountered when moving in a straight line from the start to the goal. At each obstacle boundary encountered, there is a binary choice of which way to go, and the move may be executed with perfect accuracy. Lumelsky also demonstrates complexity bounds under these assumptions, and has considered configuration spaces such as the plane, the sphere, the cylinder, and the 2-torus. While it is not clear how this technique can extend to higher-dimensional configuration spaces, it is useful to compare Lumelsky's approach as an example of how to exploit a useful geometric primitive (wall-following). See also [Koditschek] for extensions to this approach using potential fields. The *potential-field* approach to collision avoidance, as formulated by [Khatib], also can deal with uncertain obstacles, and gross motions around these obstacles can often be synthesized in real time. [Brooks 85] has described a map-making approach for a mobile robot in a highly unstructured environment—i.e., amidst unknown obstacles. His approach allows the robot to aquire information about the position and shape of these obstacles as the robot explores the environment. [Davis] has addressed the mobile robot navigation problem amidst partially unknown obstacles using an approximate map.

There is almost no work on planning compliant motions or assemblies in the presence of model error.

9.4 Error Detection and Recovery

There has been almost no formal analysis of the EDR problem. STRIPS [Fikes and Nilsson] has a run-time executive (PLANEX) which embodied one of the first systems addressing EDR. STRIPS' triangle tables may be viewed as a kind of forward projection. [Ward and McCalla; Hayes] have presented research agendas for error diagnosis and recovery in domain-independent planning. [McDermott] has stressed the importance of EDR in plan execution and sketched an approach based on possible worlds. [Srinivas] described a robot planning system for a Mars rover which could detect certain manipulation errors and recover. [Gini and Gini] have described a view of EDR based on a predetermined list of high-level error types. The domain-independent planning literature [Chapman] is relevant to the history of EDR; for example, the planner of [Wilkins] has an error recovery module in which the executor can detect inconsistencies in the set of logical propositions representing the world state. At this point, an operator can intervene and type in new propositions to disambiguate the state and aid recovery. The robots described by [Brooks 85] have an EDR flavor—they are not required to achieve a particular goal, but merely to attempt it until some other goal takes a higher priority.

The EDR theory in this paper has been presented in [Donald 86a,b; 87]. In [Donald, 88], we describe details of the geometrical characterization of EDR, and of one-step EDR planning. [Brost] is employing some of these EDR techniques in his research on planning squeeze-grasp operations.

References

Certain frequently cited references have been given shorter mnemonics, eg., [LMT].

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

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

3. **Ben-Or M., Kozen D., and Reif J.,** "The Complexity of Elementary Algebra and Geometry", J. Comp. and Sys. Sciences, Vol. 32, (1986), pp. 251-264.

4. **Boyse J. W.,** "Interference Detection Among Solids and Surfaces", Comm ACM, vol 22, No 1 (1979) pp 3-9.

5. **Brady, M. et. al. (eds).** *Robot Motion: Planning and Control.,* Cambridge, Mass.: MIT Press. (1982).

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

7. **Brooks, R. A** *A Robust Layered Control System for a Mobile Robot,* IEEE Journal of Robotics and Automation **RA-2** (1): 14–23. Also MIT A.I. Lab Memo 864 (1985).

8. **Brooks, R.**, "Solving the Find-Path Problem by Good Representation of Free Space", *IEEE Transactions on Systems, Man, and Cyberbetics*, Vol. 13, 1983.

9. **Brooks, R., and T Lozano-Pérez**, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", *Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, August, 1983.

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

11. **Brost, R.**, "Automatic Grasp Planning in the Presence of Uncertainty", *IEEE International Conference on Robotics and Automation*, San Francisco, April, 1986.

12. **Brost, R.** *A State/Action Space Approach to Planning Robot Actions,* Forthcoming Ph.D. Thesis, Computer Science Dept., CMU (to appear).

13. **Buckley, S. J.** *Planning and Teaching Compliant Motion Strategies,* Ph.D. Thesis. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1987. Also MIT-AI-TR-936 (1987).

14. **Burridge, R., Rajan, V. T., and Schwartz, J. T.** *The Peg-In-Hole Problem: Statics and Dynamics of Nearly Rigid Bodies in Frictional Contact,* IEEE ICRA, Raleigh, NC (1983).

15. **Caine, M.**, "Chamferless Assembly of Rectangular Parts in Two and Three Dimensions", S.M. dissertation, MIT Department of Machanical Engineering, June 1985.

16. **Cameron S.**, "A Study of the Clash Detection Problem in Robotics", proc. IEEE conf. on Robotics and Automation, 1985, pp 488-493.

17. **Canny, J.F.** *A New Algebraic method for Robot Motion Planning and Real Geometry,* FOCS (1987).

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

19. **Canny, J.F.** *The Complexity of Robot Motion Planning,* Ph.D. Thesis, MIT Department of Electrical Engineering and Computer Science (1987).

20. **Canny, J.F.** *Computing Roadmaps of Compact Semi-Algebraic Sets,* Intl. Workshop on Geometric Reasoning, Oxford, England, June (1986).

21. **Canny, J.F. and Donald, B. R.** *Simplified Voronoi Diagrams,* Proc. ACM Symposium on Computational Geometry, Waterloo, June (1987).

22. **Canny, J.F. and Donald, B. R.** *Simplified Voronoi Diagrams,* Discrete and Computational Geometry (to appear).

23. [CR] **Canny, J., and J. Reif,** "New Lower Bound Techniques for Robot Motion Planning Problems", *FOCS* (1987).

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

25. **Chistov A. L. and Grigoryev D. Y.,** "Complexity of quantifier elimination in the theory of algebraically closed fields", Lect. Notes Comp. Sci. 176, Springer Verlag, (1984).

26. **Collins G. E.** "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition" Lecture Notes in Computer Science, No. 33, Springer-Verlag, New York, (1975), pp. 135-183.

27. **Cutkosky, M.,** "Grasping and Fine Manipulation for Automated Manufacturing", Ph.D. dissertation, Carnegie-Mellon University, January, 1985.

28. **Davis, E. and McDermott, D.** *Planning and Executing Routes through Uncertain Territory,* Yale University, Dept. of Computer Science (1982).

29. [D1] **Donald, B. R.** *Motion Planning with Six Degrees of Freedom,* MIT AI-TR 791, Artificial Intelligence Lab. (1984).

30. **Donald, B. R.** *On Motion Planning with Six Degrees of Freedom: Solving the Intersection Problems in Configuration Space,* IEEE International Conference on Robotics and Automation, St. Louis, MO (1985).

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

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

33. [D] **Donald, B. R.** *A Theory of Error Detection and Recovery for Robot Motion Planning with Uncertainty,* Intl. Workshop on Geometric Reasoning, Oxford, England (1986b).

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

35. **Donald, B. R.** *Error Detection and Recovery for Robot Motion Planning with Uncertainty,* MIT Artificial Inteligence Laboratory, MIT-AI-TR 982 (1987).

36. **Draper Laboratories**, Fourth Annual Seminar on Robotics and Advanced Assembly Systems, Cambridge, Massachusetts, November, 1983.

37. **Dufay, B., and J. Latombe**, "An Approach to Automatic Robot Programming Based on Inductive Learning", in Brady, M., and R. Paul, *Robotics Research: The First International Symposium*, MIT Press, 1984.

38. **Durrant-Whyte, H.** *Concerning Uncertain Geometry in Robotics,* Intl. Workshop on Geometric Reasoning, Oxford, England, June (1986).

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

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

41. **Erdmann, M., and M. Mason**, "An Exploration of Sensorless Manipulation", *IEEE International Conference on Robotics and Automation*, San Francisco, April, 1986.

42. **Faverjon, B.** *Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator,* Proc. IEEE Intl. Conf. Robotics, Atlanta (March 1984).

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

44. **Fortune, S., Wilfong, G., and Yap, C.** 1986 (April 7-10, San Francisco, California). Coordinated Motion of Two Robot Arms. *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 1216–1223.

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

46. **Grigoryev D. Y.**, "Complexity of Deciding Tarski Algebra" Jour. Symbolic Computation, special issue on decision algorithms for the theory of real closed fields, to appear (1987).

47. **Grossman, D., and R. Taylor**, "Interactive Generation of Object Models with a Manipulator", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 8, No. 9, September, 1978.

48. [Gor] **Gordon, B. B.** *Intersections of Higher-Weight Cycles over Quaternionic Modular Surfaces and Modular Forms of Nebentypus,* Bull. AMS **14** (2), pp. 293-8 (1986).

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

50. **Hopcroft, J. E., Schwartz, J. T., and Sharir, M.** 1984 On the Complexity of Motion Planning for Multiple Independent Objects; *PSPACE*-Hardness of the "Warehouseman's Problem." *International Journal of Robotics Research.* **3**(4):76 88.

51. **Hopcroft J., and Wilfong G.,** "Motion of Objects in Contact," Int. Jour. Robotics Res. vol 4, no. 4, (1986).

52. **Hungerford, T. W.** *Algebra,* Springer-Verlag, New York GTM 73 (1974).

53. **Hogan, N.**, "Impedance Control of Industrial Robots", *Robotics and Computer-Integrated Manufacturing*, Vol. 1, No. 1, 1984.

54. **Inoue, H.**, "Force Feedback in Precise Assembly Tasks", MIT Artificial Intelligence Laboratory, AIM-308, August, 1974.

55. **Khatib, O.** *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,* Int. Jour. Rob. Res. vol. 5, No. 1, pp. 90-99 (1986).

56. **Koditschek, D.**, "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations", Proc. IEEE Intl. Conf. Robotics, Raleigh, March 1987.

57. **Koutsou, A.**, "A Geometric Reasoning System for Moving an Object While Maintaining Contact with Others", *ACM Symposium on Computational Geometry*, Yorktown Heights, N.Y., 1985.

58. **Kozen D., and Yap C.** "Algebraic Cell Decomposition in NC", Proc IEEE symp. FOCS, (1985), pp. 515-521.

59. **Laugier, C.**, "A Program for Automatic Grasping of Objects with a Robot Arm", *Eleventh Symposium of Industrial Robots*, Japan Society of Biomechanisms and Japan Industrial Robot Association, 1981.

60. **Lee, D. T., and Drysdale, R. L.,** "Generalization of Voronoi diagrams in the plane," SIAM J. Comp. (10) (1981) pp. 73-87.

61. **Lieberman, L., and M. Wesley**, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly", *IBM Journal of Research Development*, Vol. 21, No. 4, 1977, pp. 321-333.

62. **Lozano-Pérez, T.**, "The Design of a Mechanical Assembly System", S.M. dissertation, MIT Department of Electrical Engineering and Computer Science, also AI-TR-397, MIT Artificial Intelligence Laboratory, 1976.

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

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

65. **Lozano-Pérez, T.**, "Robot Programming", *IEEE Proceedings*, 1983b.

66. **Lozano-Pérez, T.**, "Motion Planning For Simple Robot Manipulators", *Third International Symposium on Robotics Research*, Paris, October, 1985.

67. **Lozano-Pérez**, "A Simple Motion Planning Algorithm for General Robot Manipulators," in Proceedings of Fifth National Conference for the American Association of Artificial Intelligence, Philadelphia, 1986, pp. 626–631.

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

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

70. **Lumelsky, V. J.** *Continuous Motion Planning in Unknown Environment for a 3D Cartesian Robot Arm. , Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 1569–1574. (April 7–10, San Francisco, Calif.) (1986).

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

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

73. **Mason, M. T.** 1986. Mechanics and Planning of Manipulator Pushing Operations. *International Journal of Robotics Research* **5**(3).

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

75. **Mason, M. T.** 1985 (March 25–28, St. Louis, Missouri). The Mechanics of Manipulation. *Proceedings of the 1985 IEEE Int. Conf. on Robotics and Automation*, pp. 544–548.

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

77. **Natarajan, B. K.** 1986 (Oct. 27–29, Toronto, Ontario). An Algorithmic Approach to the Automated Design of Parts Orienters. *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pp. 132–142.

78. **Natarajan, B. K.** 1986. On Moving and Orienting Objects. Ph.D. Thesis. Ithaca, N.Y.: Cornell University Department of Computer Science.

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

80. **Ó'Dúnlaing, C., Sharir, M., and Yap C.**, "Generalized Voronoi diagrams for moving a ladder: I Topological Analysis," NYU-Courant Institute, Robotics Lab. Tech. report No. 32 (1984)

81. **Ó'Dúnlaing, C., Sharir, M., and Yap C.**, "Generalized Voronoi diagrams for moving a ladder: II Efficient construction of the diagram," NYU-Courant Institute, Robotics Lab. Tech. report No. 33 (1984)

82. **Ó'Dúnlaing C., and Yap C.**, "A retraction method for planning the motion of a disc," J. Algorithms (6) (1985) pp. 104-111

83. **Ohwovoriole, M., and B. Roth**, "A Theory of Parts Mating For Assembly Automation", *Proceedings of the Robot and Man Symposium 81*, Warsaw, Poland, September 1981.

84. **Paul, R.**, *Robot Manipulators*, MIT Press, Cambridge, Massachusetts, 1981.

85. **Peshkin, M.**, "Planning Robotic Manipulation Strategies for Sliding Objects", Ph.D. dissertation, Department of Physics, Carnegie-Mellon University, 1986.

86. **Peshkin, M., and A. Sanderson**, "Reachable Grasps on a Polygon: The Convex Rope Algorithm", *IEEE Journal of Robotics and Automation*, Volume 2, Number 1, March, 1986.

87. **Raibert, M., and J. Craig**, "Hybrid Position/Force Control of Manipulators", *Journal of Dynamic Systems, Measurement, and Control*, No. 102, June, 1981, pp. 126-133.

88. **Reif J.**, "Complexity of the Mover's Problem and Generalizations," Proc. 20th IEEE Symp. FOCS, (1979). Also in "Planning, Geometry and Complexity of Robot Motion", ed. by J. Schwartz, J. Hopcroft and M. Sharir, , Ablex publishing corp. New Jersey, (1987), Ch. 11, pp. 267-281.

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

90. **Salisbury, J.K.**, "Active Stiffness Control of a Manipulator in Cartesian Coordinates", *IEEE Conference on Decision and Control*, Albuquerque, New Mexico, November, 1980.

91. **Salisbury, J.K.**, "Kinematic and Force Analysis of Articulated Hands", Ph.D. dissertation, Stanford University, Department of Mechanical Engineering, 1982.

92. **Segre, A. M., and G. DeJong**, "Explanation-Based Manipulator Learning: Acquisition of Planning Ability Through Observation", *IEEE International Conference on Robotics and Automation*, St. Louis, March, 1985.

93. **Schwartz J., Hopcroft J., and Sharir M.**, "Planning, Geometry and Complexity of Robot Motion Planning", Albex Publishing Co., New Jersey, (1987).

94. **Schwartz J. and Sharir M.**, "On the 'Piano Movers' Problem, II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds," Comp. Sci. Dept., New York University report 41, (1982). Also in "Planning, Geometry and Complexity of Robot Motion", ed. by J. Schwartz, J. Hopcroft and M. Sharir, Ablex publishing corp. New Jersey, (1987), Ch. 5, pp. 154-186.

95. **Schwartz J. and Yap C. K.**, "Advances in Robotics," Lawrence Erlbaum associates, Hillside New Jersey, (1986).

96. **Simunovic, S. N.**, "An Information Approach to Parts Mating", Ph.D. dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, 1979.

97. **Simunovic, S. N.** 1975 (Sept. 22–24, Chicago, Illinois). Force Information in Assembly Processes. *Proceedings 5th International Symposium on Industrial Robots.* Bedford, U.K.: IFS Publications, pp. 415–431.

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

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

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

101. **Tarski A.**, "A Decision Method for Elementary Algebra and Geometry" Univ. of Calif. Press, Berkeley, (1948), second ed. 1951.

102. **Turk, M.**, "A Fine-Motion Planning Algorithm", *SPIE Conference on Intelligent Robots and Computer Vision*, Cambridge, Massachusetts, September, 1985.

103. **Udupa, S.**, "Collision Detection and Avoidance in Computer Controlled Manipulators", Ph.D. dissertation, Department of Electrical Engineering, California Institute of Technology, 1977.

104. **Udupa S.**, "Collision Detection and Avoidance in Computer Controlled Manipulators", Proc. 5th Int. Joint. Conf. on Art. Intell., Mass. Inst. Tech. (1977) pp 737-748.

105. **Valade, J.**, "Automatic Generation of Trajectories for Assembly Tasks", *Sixth European Conference on Artificial Intelligence*, Pisa, Italy, September, 1984.

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

107. **Whitney, D.**, "Force Feedback Control of Manipulator Fine Motions", *Journal of Dynamic Systems, Measurement, and Control*, June, 1977, pp. 91-97.

108. **Whitney, D.**, "Quasi-Static Assembly of Compliantly Supported Rigid Parts", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, March 1982.

109. **Whitney, D.**, "Historical Perspective and State of the Art in Robot Force Control", *IEEE International Conference on Robotics and Automation*, St. Louis, March, 1985.

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

111. **Yap, C.**, "Coordinating the motion of several discs," NYU-Courant Institute, Robotics Lab. No. 16 (1984)

112. **Yap, C.**, "Algorithmic Motion Planning", in *Advances in Robotics: Volume I*, edited by J. Schwartz and C. Yap, Lawrence Erlbaum Associates, 1986.