Automatic Synthesis of Fine-Motion Strategies for Robots

Tomás Lozano-Pérez
Department of Electrical Engineering and Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Matthew T. Mason
Department of Computer Science and Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Russell H. Taylor
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Abstract

Active compliance enables robots to carry out tasks in the presence of significant sensing and control errors. Compliant motions are quite difficult for humans to specify, however. Furthermore, robot programs are quite sensitive to details of geometry and to error characteristics and must, therefore, be constructed anew for each task. These factors motivate the search for automatic synthesis tools for robot programming, especially for compliant motion. This paper describes a formal approach to the synthesis of compliant-motion strategies from geometric descriptions of assembly operations and explicit estimates of errors in sensing and control. A key aspect of the approach is that it provides criteria for correctness of compliant-motion strategies.

This report describes research done in part 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 System Development Foundation, in part by the Office of Naval Research under Office of Naval Research contract N00014-81-K-0494, and in part by the Advanced Research Projects Agency under Office of Naval Research contracts N00014-80-C-0505 and N00014-82-K-0344. R. H. Taylor's work was funded in part by IBM while he was on sabbatical at MIT.

The International Journal of Robotics Research,
Vol. 3, No. 1, Spring 1984,
0278-3649/84/010003-20 $05.00/0,

1. Introduction

The central robot-programming problem is to enable a robot to perform tasks despite its uncertain position relative to external objects. The use of sensing to reduce uncertainty significantly extends the range of possible tasks. Sensor-based robot programs are very difficult to write, however, as there is little theory to serve as a guide. To make matters worse, programs written for one task are seldom, if ever, applicable to other tasks. These two points make the development of an automatic synthesis strategy for sensor-based robot programs a key priority.

In this paper, we propose a formal approach to the automatic synthesis of a class of compliant, fine-motion strategies applicable to assembly tasks. The approach uses geometric descriptions of parts and estimates of measurement and motion errors to produce fine-motion strategies. Although our description of the approach will be in the form of an abstract algorithm, no implementation of this approach exists at present (although implementation is in progress). The formalism provides a structured way of thinking about fine-motion strategies and, therefore, may be helpful to human programmers of such strategies.

1.1. Fine-Motion Strategies

One important source of difficulty in robot programming is that the programmer's model of the environ-

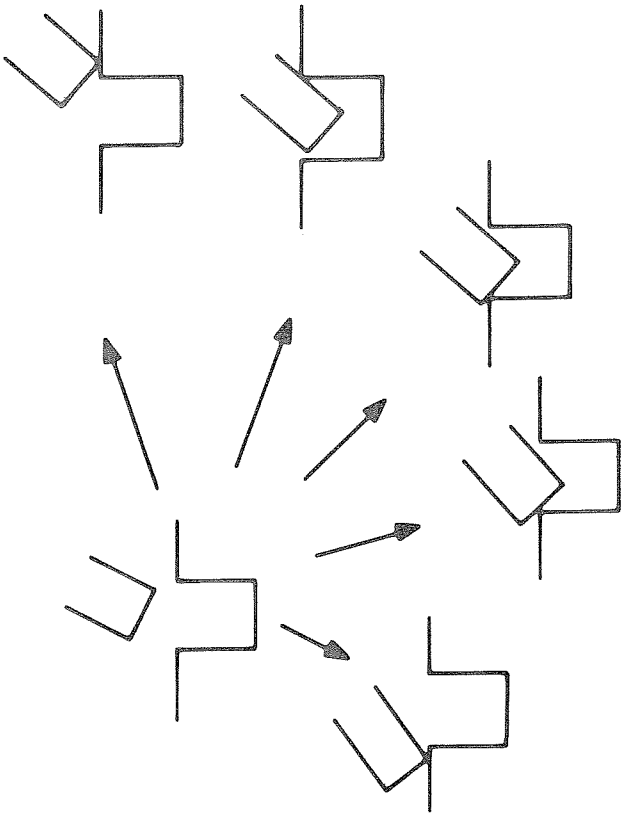


Fig. 1. Some possible configurations for peg-in-hole insertion.

ment is incomplete and inexact as to the shape and location of objects. Vision may be used to determine the approximate shapes and positions of objects, but generally not with sufficient accuracy for assembly by pure position control. Even knowing object shapes and positions sufficiently accurately is not enough. Positioning errors inherently limit the tasks achievable by strict position control. Increasing the mechanical accuracy of robots to levels required for assembly is expensive and ultimately stifling. Instead, one must abandon the paradigm of pure position control for tasks where the allowable motions are tightly constrained by external objects, as they are in mechanical assembly.

The basic method for achieving constrained motion in the presence of position uncertainty is by the use of controlled *compliance* (see Mason's work [1983] for an overview of compliance research). Compliant motion meets external constraints by specifying how the robot's motion should be modified in response to forces generated when constraints are violated. Contact with a surface, for example, can be guaranteed by moving so that a small force normal to the surface is maintained. With this technique, the robot can achieve and retain contact with a surface that may vary significantly in shape and orientation from the programmer's expectations. Generalizations of this principle can be used to accomplish a wide variety of tasks involving constrained motion, for example, inserting a peg in a hole and following a weld seam.

The specification of particular compliant motions to achieve a task requires knowledge of the geometric constraints imposed by the task. Given a description of the constraints, choices can be made for the compliant-motion parameters, for example, motion freedoms to be force controlled and those to be position controlled (Paul and Shimano 1976; Mason 1981; Raibert and Craig 1981), or the center of compliance and axis stiffnesses (Hanafusa and Asada 1977; Salisbury 1980; Whitney 1982). It is common however, for position uncertainty to be large enough so that the programmer cannot unambiguously determine which geometric constraint holds at any instant in time. Figure 1, for example, shows some different initial conditions that can hold in two-dimensional, peg-in-hole insertion. Under these circumstances, the programmer must employ a combined strategy of force and position control that guarantees reaching the desired final configuration from all of the likely initial configurations. We call such a strategy a *fine-motion strategy.*

One of the most widely studied tasks in robotics is the two-dimensional, peg-in-hole task. Detailed analyses have been carried out to determine strategies that guarantee successful insertion once the peg is partly in the hole (McCallion and Wong 1975; Simunovic 1975; Drake 1977; Ohwovoriole and Roth 1981; Whitney 1982). When the initial uncertainty in position is large enough, a strategy must also be devised to ensure that the peg can find the hole (Inoue 1974; McCallion and Wong 1975). We can illustrate a variety of strategies for one task by considering the ways this problem has been addressed.

1. Chamfers. Chamfers on the hole entrance and/or the peg tip increase the range of relative positions where the peg can fall into the hole,

at least partway. This technique is especially effective if the peg support has lateral compliance (Drake 1977; Whitney 1982).

2. Tilting the peg. Tilting the peg slightly also increases the range of relative positions where initial entry into the hole is guaranteed (Inoue 1974). In fact, the geometric effect of tilting the peg is almost identical to that of providing a chamfer (see Section 2.6).
3. Search. The simplest strategy is a search in which the peg slides along the top surface until it falls into the hole. In general, the search has to pick an initial direction of motion and, possibly, back up if the hole is not found.
4. Biased search. A slight modification to the search strategy is to introduce a bias into the initial position of the peg (Inoue 1974). This strategy reduces the chances of initial entry into the hole, but it guarantees that the peg will be to one side of the hole.

In subsequent sections, we will consider such strategies as tilting the peg and biased search. These are simple strategies that employ compliant motion and do not require modifications of task geometry or complicated control structures.

1.2. Previous Work

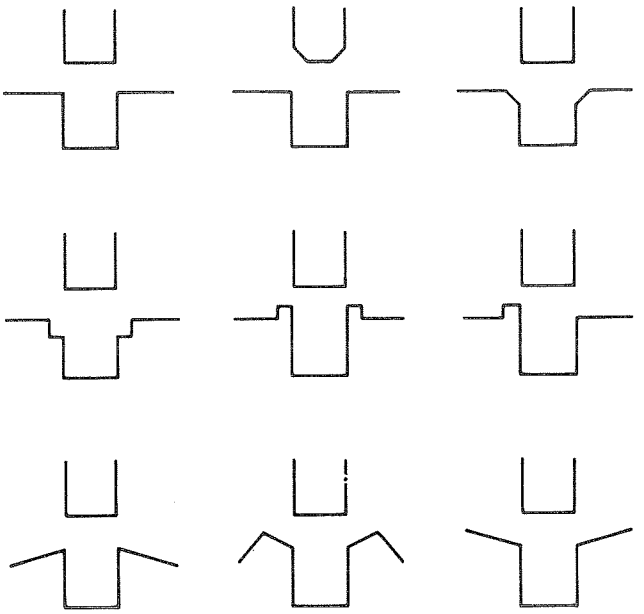
In this paper, we present an approach to the automatic synthesis of a class of fine-motion strategies. We are aware of no previous work with the same goal. There are, however, several bodies of work relevant to this goal. The first of these deals with analyses of geometry and statics of tasks with the aim of developing conditions that successful fine-motion strategies must satisfy. The second is Simunovic's information approach. The third deals with attempts to derive strategies starting from partially specified strategies, known as *skeletons,* or *plans.* The fourth deals with attempts to have the robot "learn" strategies from experience and partial task information.

Quite a few authors have analyzed the peg-in-hole assembly task in detail (Laktionev and Andreev 1966; Andreev and Laktionev 1969; Gusev 1969; McCallion and Wong 1975; Simunovic 1975; Drake 1977; Ohwovoriole, Roth, and Hill 1980; Ohwovoriole and Roth 1981; Whitney 1982). In most of the analyses, the assumption is that the peg is initially partly in the hole, possibly at a chamfer. Two important types of insertion failure have been identified: *jamming* and *wedging.* Jamming is due to misproportioned applied forces; wedging is due to geometric conditions that arise when the parts are slightly deformed. These analyses have led to the formulation of conditions for successful insertion relating forces applied to positions of the peg and hole. As a result, a remote center compliance (RCC) device, a mechanical device (Drake 1975; Whitney 1982), has been built that applies the correct forces in response to small initial errors in positioning. A number of heuristic strategies for peg-in-hole insertion have also been formulated, based on more fragmentary analysis. These heuristic strategies have been used successfully in practice (Inoue 1974; Goto, Takeyasu, and Inoyama 1980).

Mason's (1982) detailed analysis of pushing and grasping operations in the presence of friction also leads to conditions for successful task completion. These conditions provide the basis for synthesis of operations that succeed in the presence of uncertainty (without requiring sensing).

Simunovic (1979) formulated the information approach to fine motion, based on the principle that assembly is purely a relative positioning task. From this premise, he argues that the role of an assembly program is to determine the relative positions of parts during an assembly and to issue position commands to correct errors. He developed an estimation technique to infer, from a series of noisy position measurements and knowledge of the geometry of the parts, the actual relative positions of the parts. One problem with this approach is that it requires a very large amount of on-line computation, although this could be solved with special-purpose electronics. A more fundamental problem is that the approach assumes only position control and a robot capable of making fine incremental motions. This need not be the case for assembly; by exploiting compliant behavior, the robot can perform high-accuracy tasks even with low-accuracy position control, for example, the task of following a surface by maintaining a downward force. Another problem is that Simunovic's estimation technique requires knowing which surfaces are in contact. This limits the method to situations with relatively

Fig. 2. Variations of peg-in-hole insertion require different strategies.



small errors; in more general cases, the identity of the contact surfaces is not known. Our approach is based on a different view of assembly: that the geometric constraints should "guide" the parts to their destination without necessarily having to know exactly where the parts are relative to each other.

One of the earliest explorations in the area of automatic synthesis of fine-motion strategies from *strategy skeletons* was by Taylor (1976). Taylor developed a technique for propagating the effect of errors and uncertainties through a model of a task. These error estimates were used to make decisions for filling in the strategy skeletons. For peg-in-hole insertion, for example, the decision whether to tap the peg against the surface next to the hole was based on whether the error estimate for position normal to the surface exceeded a threshold.

Lozano-Pérez (1976) also proposed a method for selecting the motion parameters in strategy skeletons. Each motion in a skeleton was specified symbolically by the relationship among parts that it was designed to achieve. The expected length of guarded moves and their force-terminating conditions were then computed from the ranges of displacements that achieved this relationship (taking into account uncertainty in position).

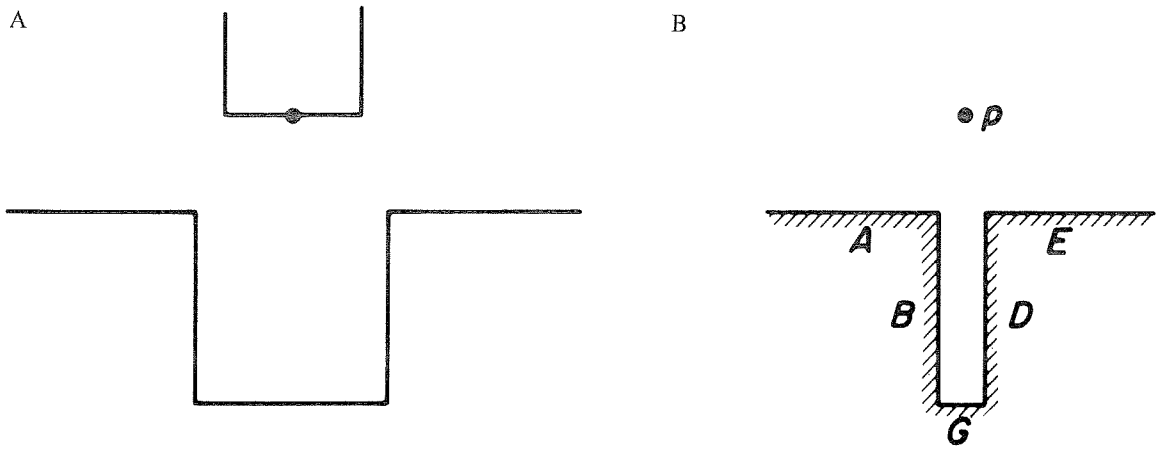
Recently, Brooks (1982) extended Taylor's approach by making more complete use of symbolic constraints in the error computations. The resulting constraints can be used in the "forward" direction to estimate errors for particular oprations. But, importantly, they also may be used in the "backward" direction to constrain the values for plan parameters, such as initial positions of objects, to those that enable the plan to succeed. When no good choice of parameters exists, the system chooses appropriate sensing operations (such as visual location of parts) that reduce the uncertainty enough to guarantee success.

Another line of research has focused on building up programs automatically from attempts by the robot to carry out the operations. Dufay and Latombe (1983) describe how partial local strategies ("rules") for a task can be assembled into a complete program by processing the execution traces of many attempts to carry out the task. The method, however, requires knowing the actual relationship between parts achieved by each motion, for example, which surfaces are in contact. This information can be obtained, in many cases, from careful analysis of the forces and positions, but, in general, the information is ambiguous in the presence of measurement and control errors. Moreover, the rules used by the system are specific to tasks and must be provided by the users.

A related approach to deriving a strategy from "experiments" is based on the theory of stochastic automata (Simons et al. 1982). The goal is to have the robot learn the appropriate control response to measured force vectors during task execution. The method requires a task-dependent evaluation function so as to judge progress toward its goal.

These previous approaches to fine-motion synthesis are based on the assumption that there is a basic repertoire of operations, such as peg-in-hole insertion and block-in-corner placement, whose geometric structure is known a priori. In this view, the task of a synthesis program is to make some predefined set of choices among alternative actions, select the values of some parameters, and, possibly, select the order of operations. *In fact, small changes in the geometry of parts can have significant impact on fine-motion strategies.* The different operations shown in Fig. 2, for example, can all be classified as peg in hole, yet

Fig. 3. Peg-in-hole insertion. A. Original formulation. B. Transformation to an equivalent point (p) problem.



they require substantially different programs to ensure reliable execution. Similarly, differences in expected position errors call for different strategies for the same task.

Our approach is motivated by the belief that the set of possible geometric interactions in a task should directly determine the structure of the fine-motion strategy for the task. Thus, for example, the presence of additional surfaces within the region of possible initial contact typically requires a change in the structure of a strategy. The approach we describe in this paper proceeds directly from geometric descriptions of the parts to a strategy.

2. Overview of the Approach

In this section, we informally outline our approach to fine-motion synthesis using a progression of simple examples. In section 3, we provide a more formal characterization of the approach.

2.1. The Basic Strategy

Consider the simple task of moving the point p from its initial position to any one of the positions in G (see Fig. 3B). This is a simplified problem but not a completely artificial one. It is equivalent to the two-dimensional, peg-in-hole problem in Fig. 3A, where the axes of the peg and hole are constrained to be parallel. The position of p determines the position of the peg. The boundary of the shaded area represents the positions of p where the peg would be in contact with an obstacle. The transformation from Fig. 3A to Fig. 3B corresponds to shrinking the peg to a point and expanding the obstacles accordingly. Note that the sides of the hole have been moved toward each other by half the width of the peg. In this case, the transformation produces an equivalent problem. We postpone a more general discussion of this type of transformation until Section 2.6. The problem of moving rigid objects among other rigid objects can be reformulated as the equivalent problem of moving a point among transformed objects in a higher-dimensional space, called the *configuration space* (Lozano-Pérez 1981; 1983).

The basic step in our synthesis approach is to identify ranges of positions from which p can reach G by a single motion. The directions of such motions can be represented as unit velocity vectors, $\mathbf{v}_i$. For each $\mathbf{v}_i$, we can compute all those positions, P_i, such that a motion along $\mathbf{v}_i$ from that position would reach some point of G (see Fig. 4). We call this range of positions that can reach the goal by a single motion along a specified velocity the *pre-image*[1] of the goal (for that velocity). All we need do to guarantee that p reaches G from any point in any of the P_i is to execute a motion with commaned velocity along $\mathbf{v}_i$.

If no pre-image of G contains the peg's current posi-

1. The rationale for this name stems from viewing motions as mappings from pairs of initial positions and velocities into points along the resulting path.

Fig. 4. Pre-image of the goal for different $\mathbf{v}_i$.

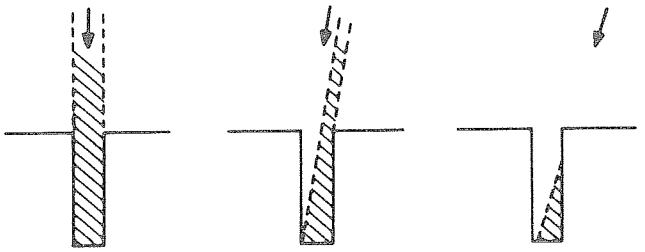
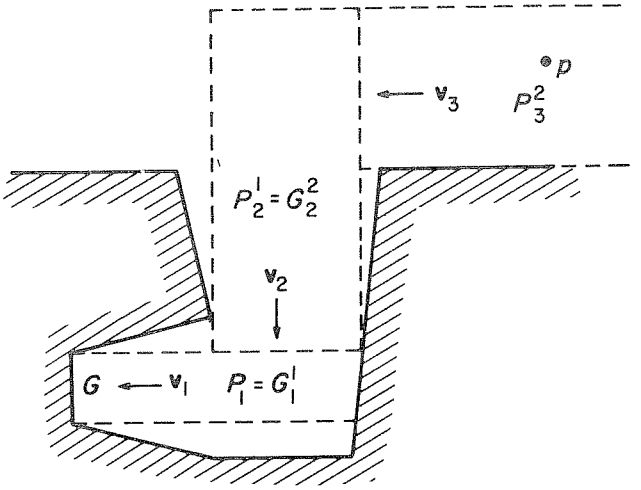


Fig. 5. Backward chaining of pre-images.



tion, then we can apply the same pre-image computation recursively, using each of the existing pre-images as a possible goal. This recursive process is an instance of the problem-solving strategy known as *backward chaining* (Nilsson 1980). Each pre-image of G, P_i, serves to define a new goal set G_i^1 (the superscript indicates the "recursion level"). This process is repeated until some pre-image P_i^k contains the current position of p (see Fig. 5). From this chain of pre-images $P_{i_0}, P_{i_1}^1, \ldots, P_{i_k}^k$, we can construct a motion strategy. The two components of the strategy are a sequence of velocity vectors and a sequence of associated termination predicates. Therefore, the strategies may be construed as a sequence of *guarded motions* (Will and Grossman 1975). Each velocity vector $\mathbf{v}_{i_j}$ defines a motion that moves from anywhere in $P_{i_j}^j$ to $G_{i_{j-1}}^j$. Whenever p reaches one of the goal regions, the velocity command needs to be changed to one appropriate for the new region, for example, from $\mathbf{v}_{i_j}$ to $\mathbf{v}_{i_{j-1}}$. The role of the termination predicates is to detect the arrival of p within a goal region. In the simple case we have been discussing, termination predicates simply test to see whether the position is in the goal region. Termination predicates are much more difficult to construct in the presence of position upcertainty. We will discuss the issue further in Sections 2.2 and 3.

In summary, our basic approach to fine-motion synthesis is to chain backward from the goal toward the current position, characterizing at each step the range of positions that can reach the current goal in one motion, that is, the pre-image of the goal. It remains to be shown how this simple approach is applicable to more realistic assembly problems. To demonstrate its applicability, we will discuss (1) the role played by uncertainty in position and velocity, (2) the introduction of compliant motions, (3) the handling of friction, and (4) the notion that configuration space reduces assembly problems for solids into problems involving a point and surfaces in a higher-dimensional space.

2.2. The Effect of Uncertainty

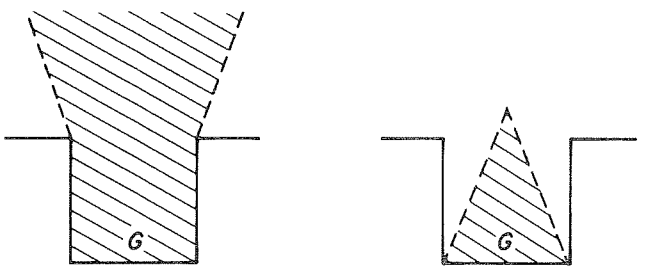
We have assumed thus far that p's position is known exactly at all times and that its direction of motion can be specified exactly. In this section we explore the effects of relaxing these assumptions.

Let us assume that there is error between the actual and the commanded velocity, bounded by ϵ_v. The actual velocity is within a ball of radius ϵ_v in velocity space (the ball of velocities centered on $\mathbf{v}$ is denoted $B(\mathbf{v})$). Therefore, the path of p is constrained to be within a semi-infinite cone centered on the commanded path and whose apex is the initial position. The angle between the actual direction of motion and the commanded direction is constrained to be less than or equal to $\sin^{-1}\epsilon_v$, which will be approximately ϵ_v for small enough ϵ_v.

The synthesis approach above is based on computing the pre-images of goal regions for particular values of commanded velocity. These are locations from which the goal can be reached by a single motion. In the presence of uncertainty in the actual velocity, we define two alternative pre-images (see Fig. 6):

Weak pre-image—locations for which *some* mo-

Fig. 6. Weak and strong pre-images of a goal, with velocity uncertainty.



A. Weak pre-image *B. Strong pre-image*

tion within the range of velocity uncertainty may reach the goal

Strong pre-image—locations for which *all* motions within the range of velocity uncertainty will reach the goal

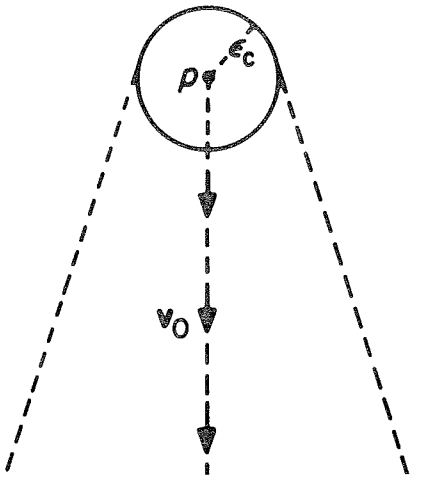
Note that the strong pre-image is a subset of the weak pre-image. In what follows, we will use the term *pre-image* to mean *strong pre-image.*

In addition to uncertainty in the actual velocity along a motion, there is uncertainty in the position of p. One source of position uncertainty is imperfect knowledge of the initial position of the objects in the workspace. Another source of error is inherent limitations in the robot's position sensors. For the sake of simplicity, we will lump these two types of uncertainty into a single upper bound on position uncertainty. This assumption does not affect correctness of any derived motion strategies, but might lead to less efficient strategies. In practice, the two sources of uncertainty should be treated differently.

We assume that the actual position is always within a ball of radius ϵ_c centered at the position observed by the robot. All possible observed positions are within a similar ball centered at the actual position. The ball of possible observed positions centered at a position p is denoted $B(p)$. The range of positions potentially traversed by a motion from an observed position along a commanded velocity is depicted in Fig. 7.

Position uncertainty makes it difficult to define termination predicates for motions. A predicate that simply compares the observed position of p against the boundaries of G^i could terminate a motion prematurely. The actual position of p could be anywhere within a ball of radius ϵ_c from the observed position. In order to *guarantee* success, *all* possible positions of

Fig. 7. Positions reachable by commanded motion, with uncertainty.



p must be within the goal. We can think of this effect of position uncertainty as "shrinking" the goal by ϵ_c for purposes of detecting entry. Shrinking G^i removes from G^i any point at a distance less than or equal to ϵ_c from any point in free space not in G^i. This removes from G^i any point that is ambiguous. In many cases, this means that *no* part of the goal is unambiguously identifiable on the basis of position. We will have to rely on the effects of collisions with surfaces or on history to identify entry into a goal region. This issue is quite subtle; it is the subject of Section 3.1 (also see Section 2.5).

2.3. Compliant Motion

The example above dealt only with position-controlled motions. Due to uncertainty in p's position and velocity relative to the task, this type of motion often leads to empty pre-images. This indicates that position accuracy is not sufficient for the task. We mentioned earlier that the alternative motion regime is compliant motion. We can visualize the effect of compliant motions as producing sliding on the constraint surfaces derived from the obstacles. *Sliding* means that the moving object confines its motions to be tangent to the constraining surface or surfaces (Mason 1981). When not in contact with a surface, the motion is along the commanded velocity (to within the velocity uncertainty).

The *generalized-damping* model (Whitney 1977)

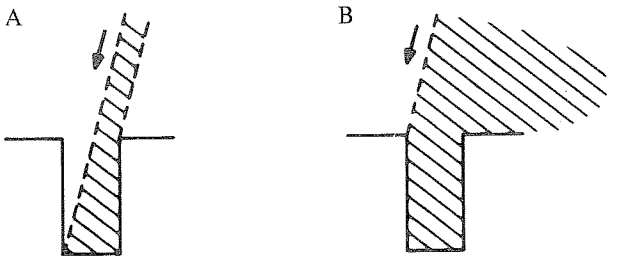


Fig. 8. Pre-images for position control (A) and generalized-damper motions (B).

can be used to implement compliant motions with the properties described above. The desired motion is determined by the relationship

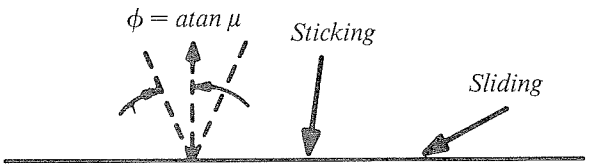
$$\mathbf{f} = \mathbf{B}(\mathbf{v} - \mathbf{v}_0),$$

where $\mathbf{f}$ is the vector of forces acting on the moving object, $\mathbf{v}_0$ is the *nominal velocity vector,* and $\mathbf{v}$ is the actual velocity vector. In what follows, $\mathbf{B}$ is a diagonal matrix. We use $\mathbf{B}$ primarily to relate the units of force to those of velocity. We assume that the control system implements the above equation, and so the behavior of the robot and moving object can be adequately approximated as a damper (for some limited range of operating velocities). The Appendix provides a more detailed treatment of the behavior of a generalized damper.

In practice, because of measurement and implementation errors, there is a difference between the commanded behavior and the actual behavior of the damper. We summarize these differences by introducing the distinction between the actual nominal velocity, $\mathbf{v}_0$, and the *commanded nominal velocity,* denoted $\mathbf{v}_0^*$. Throughout the paper, the asterisk will denote measured or commanded quantities that differ from the actual ones because of the presence of error.

The definition of the pre-image of a goal as the set of positions that can reach the goal with one commanded velocity can be retained for generalized-damper motions. *Under compliant motion, however, the moving object may reach the goal indirectly by sliding on intervening surfaces.* Therefore, compliant motions typically produce larger pre-images than pure positioning motions. The increased pre-image indicates less sensitivity to uncertainty (compare Figs. 8A and 8B).

Fig. 9. Friction cone.



2.4. Friction

A crucial consideration in the analysis and synthesis of fine-motion strategies is the effect of friction. A simple model of friction for planar motion without rotation is as follows. We assume that the objects are of a single material with equal coefficients of static and sliding friction, μ. The reaction force from contact at a point on a surface will lie within a *friction cone* with apex at the point of contact and center line along the surface normal. The angle between the normal and the sides of the cone is the *friction angle,* $\phi = \tan^{-1}\mu$. If the applied force points into the friction cone, that is, if the angle of the force vector to the surface normal is less than ϕ, then no motion will result. If the angle of the force vector to surface normal is greater than ϕ, sliding will result (see Fig. 9).

This model of friction can be extended to include rotations and moments, but the details are beyond the scope of this discussion (see Erdmann 1983). In what follows, we need only assume that some nominal velocity vectors will cause sticking on a surface and others will cause sliding. We assume, furthermore, that the range of nominal velocity vectors that causes sticking for a surface can be conservatively bounded by a cone, which may be wider than the actual cone. The computation of pre-images must take into account the possibility of sticking on a surface. In particular, assuming the motion is generated by a damper (with $\mathbf{B} = b\mathbf{I}$), if the range of nominal velocities for the pre-image contains nominal velocities whose angle to the normal of some surface is less than the friction angle, ϕ, then the motion will stop at that surface (see Fig. 9 and the Appendix).

2.5. Examples

We now have the conceptual tools necessary to synthesize a strategy for the simple example of Fig. 3. In

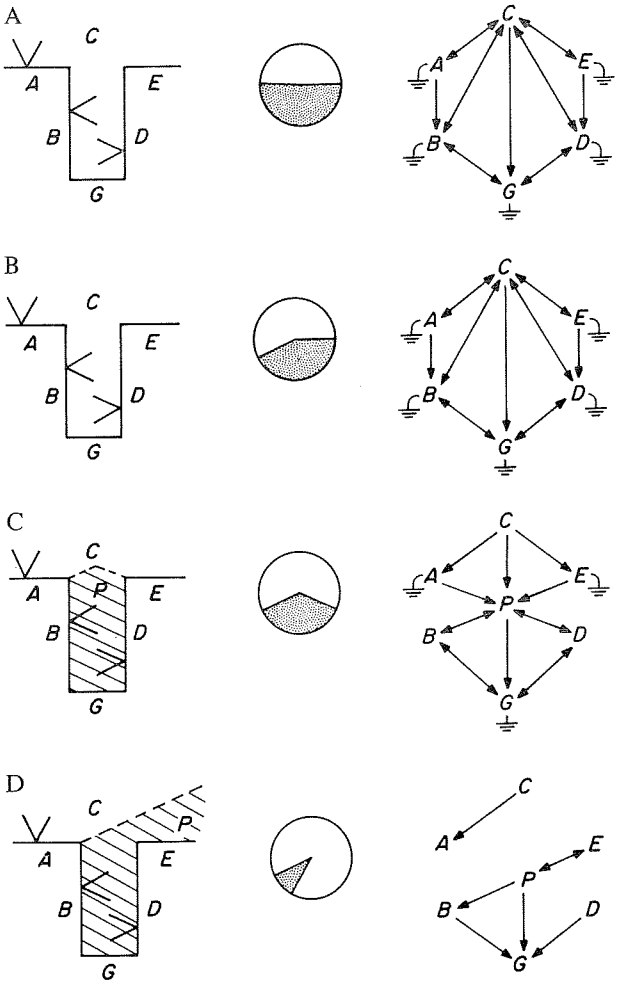


Fig. 10. Single-move, peg-in-hole strategy synthesis.

this section, we illustrate one particular approach to the synthesis of strategies based on pre-images. The method used in this section is a subset of the general approach described in Section 3.

Our goal is to identify some subregion P of the free space C and a command nominal velocity $\mathbf{v}_0$ such that P is the strong pre-image, under $\mathbf{v}_0$, of the goal surface. Equivalently, P must not overlap the weak pre-image of any surface where motion along $\mathbf{v}_0$ may stick or not reach the goal. We will use this constraint to drive a simultaneous search for P and $\mathbf{v}_0$.

The key problem is in discovering $\mathbf{v}_0$. Our approach here is to narrow in on feasible values of $\mathbf{v}_0$ by progressive refinement. We start with the complete range of possible $\mathbf{v}_0$'s and remove from that range any values that can possibly lead to failure (due to sticking or not reaching the goal). At each step of the algorithm, we compute the strong pre-image of the goal for the current *range* of $\mathbf{v}_0$'s. The strong pre-image for a range of command velocities is the intersection of the strong pre-images for each of the velocities. These are the positions guaranteed to reach the goal for *all* the velocities in the range.[2] This is the same definition that we saw in Section 2.2 for the strong pre-image in the presence of velocity error. In fact, as long as the velocity ranges used to compute pre-images are greater than $2\epsilon_v$, we need not concern ourselves further with velocity uncertainty. Once the algorithm has chosen a final velocity range, we can pick a specific velocity from the range such that all velocities within the velocity error fall in the chosen velocity range. Narrower velocity ranges will not yield such a safe velocity.

For now, we will ignore the need for backward chaining and sketch an algorithm for synthesizing single motions. (We will deal with backward chaining presently.) The basic algorithm steps are as follows.

1. Compute P, the strong pre-image of the goal surface, for the current range of commanded velocities. If the current range of velocities is split into disjoint subranges, then steps 1 and 2 should be repeated for each subrange (see Fig. 10).
2. If P includes an uncertainty ball centered at some starting position, then return P and the current velocity range.
3. Otherwise, if P is empty pick x to be a surface (other than a goal) where the robot may stick, that is, such that some velocity in the current velocity range points into the surface's friction cone. If no such surface exists, then notify failure and stop.
4. Remove from the range of commanded velocities any velocity pointing into the friction cone of x.
5. Go to step 1.

We can illustrate the operation of this algorithm on

2. Note that the weak pre-image for a range of velocities is the union of the weak pre-images for velocities in the range, that is, positions that may reach the goal for *some* velocity in the range.

our example as follows. Construct a directed graph with nodes for each of the surfaces in Fig. 3 and one node representing free space (C). A link is directed from node m to node n in the graph if m and n are direct neighbors and m is in the weak pre-image of n for the specified velocity range. That is, there is a link from m to n if some velocity in the current range may cause the robot to move from some point in m (which is not in n) to some point in n (which is allowed to be at the intersection of m and n) without going through points in any other node. In principle, the graph should have nodes representing the vertices; we have left them out for simplicity. This simplification introduces the need for the phrases in parentheses above. See, for example, the link between A and B and B and G (but not vice versa) in Fig. 10A. We will call this the *reachability graph* for that range of commanded velocities. Th reachability graph plays a key role in algorithms for computing the strong pre-image of the goal.

In our example, we start out with a range of commanded velocities including any velocity with a y component less than or equal to zero (we diagram ranges of commanded velocities as sectors of a circle). These are the velocities that will move p from nearby points onto the goal surface G. The reachability graph for this range of velocities is shown in Fig. 10A. In this figure, we have indicated those surfaces where the moving object may stick (using the electrical ground symbol). The (potential) sticking surfaces are those whose friction cones overlap the current velocity range. For simplicity, we assume that the contact on surfaces B and D are point contacts.

Figure 10 illustrates the reachability graph and pre-image of the goal each time step 1 is executed. The surfaces used to constrain the range of commanded velocities (step 3) were chosen in the following order: B, D, A, E. The particular order does not affect the final result in this case. The algorithm terminates at the fourth cycle. In Fig. 10D, we have shown only one of the two velocity ranges (and corresponding P's) that result from discarding velocities that may stick on A or E. The remaining velocity range leads to a pre-image that is the mirror image (about the hole axis) of the one in Fig. 10D. Any commanded velocity within either of these remaining velocity ranges will reach the goal from any position within P. Note that the single motion strategy developed by this approach is a biased search (see Section 1.1). This is a good choice, since we have not included chamfers or rotation in our problem definition.

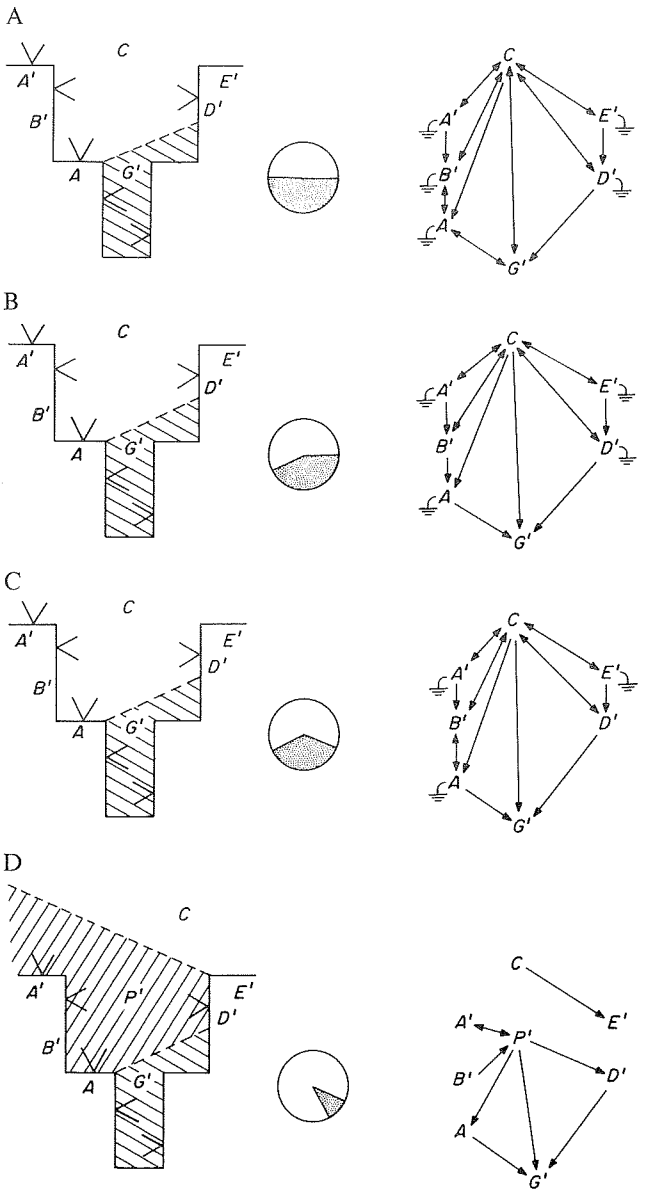
The example in Fig. 10 involves a single motion (because the friction cones of the horizontal and vertical surfaces do not overlap). We did not require the use of backward chaining. After step 2 of the algorithm, we can choose to refine the range of directions or to use the current P as the goal for a recursive call to the same algorithm. In principle, we can follow these two paths nondeterministically. In practice, this requires a search guided by such considerations as the number of motions in the strategy thus far and the size of P.

The example in Fig. 11 involves the use of backward chaining to develop multimove strategies. In this example, we assume that the first four steps proceed essentially as in Fig. 10. The final pre-image of the first example now becomes G^1, the goal for the next recursion level. The method applies as before and generates a new pre-image and velocity range. The strategy, then, consists of choosing some velocity from this range, moving until transition into G^1 is detected, and changing the commanded velocity to one of those from the range obtained in the first example.

We noted earlier that, for each commanded motion in a strategy, it is necessary to define a predicate that indicates that the goal has been reached. In multimove strategies, this condition signals that another motion should be commanded. Three types of basic termination conditions are available.

1. *Position termination.* Terminate if the measured position of p is such that all possible actual positions consistent with the measurement are within the goal region.
2. *Velocity (force) termination.* Terminate if the observed velocity of p is such that all possible actual velocities consistent with the measurement can only occur within the goal region. Note that since motions are generated by a generalized damper, the difference between actual velocity and commanded velocity provides information about reaction forces, for example, contact with a surface.
3. *Time termination.* Terminate if the elapsed

Fig. 11. Multimove, peg-in-hole strategy synthesis.



time is such that all positions consistent with the commanded motion and observed data are within the goal.

Position termination requires that all actual positions consistent with the measured position be within the goal. This is equivalent to the measured position being in the goal after being shrunk by ϵ_c along its boundary with free space. If any of the dimensions of the goal region are less than ϵ_c, then position feedback is not a reliable indicator that the goal has been reached. When the goal is a surface, for example, shrinking will cause the goal region to vanish. In these cases, we must rely on velocity termination, which requires that the observed velocities (e.g., when landing on or leaving from a surface) be unambiguous relative to surfaces that may be confused with the goal due to position-measurement error. Time termination is also useful, when applicable, as it is much simpler to test than position termination.

Velocity termination is the most useful termination condition when there is large position uncertainty. The strategy synthesized from the example in Fig. 11 illustrates this. The first motion required by the strategy can be terminated when the x component of observed velocity is zero, that is, when p strikes one of walls on the right of the hole. The second motion can be terminated when either the x and/or y components of the observed velocity are zero, that is, when p is at the left-bottom corner of the hole.

Figures 10 and 11 illustrate the class of fine-motion strategies we wish to consider. The strategies operate over a wide range of uncertainties without explicitly computing where the parts are relative to each other. The strategies do not keep any explicit history of previous events, although, as we will see later, history is implicit in the strategies.

2.6. Configuration Space

The basic operation in the synthesis method described above is computing the strong pre-image of a goal. To do this, we first transform the input problem, involving a moving object and stationary obstacles, into an equivalent problem, involving a point and transformed obstacles. Transformation has a number of advantages. One is that it enables us to represent the pre-images as areas in the transformed space. The key advantage, however, is that transformation makes the constraints on motion explicit. This is illustrated in Fig. 12, where an upright peg and chamfered hole are shown to lead to transformed obstacles similar (as far as initial entry into the hole is concerned) to those of a chamfered peg and unchamfered hole and to those of a tilted peg

Fig. 12. C-space representations that make motion constraints explicit.

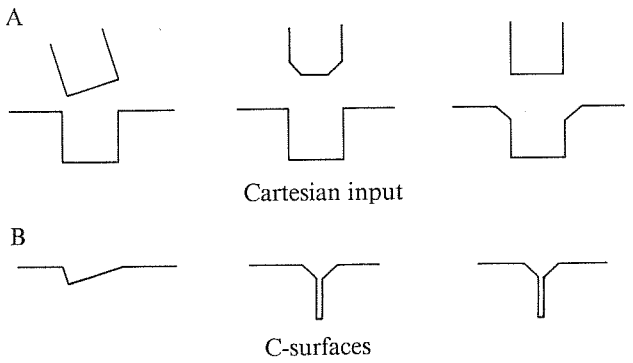
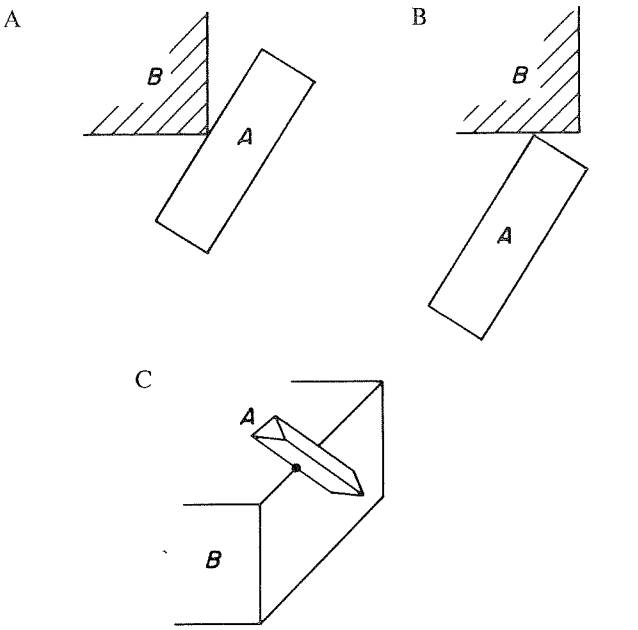


Fig. 13. Geometric conditions giving rise to C-surfaces.



and unchamfered hole. The transformation makes explicit the underlying similarity of motion constraints in these tasks. In fact, the transformation reduces tasks involving "arbitrary" geometric interactions between objects to the interactions possible between a point and a set of surfaces.

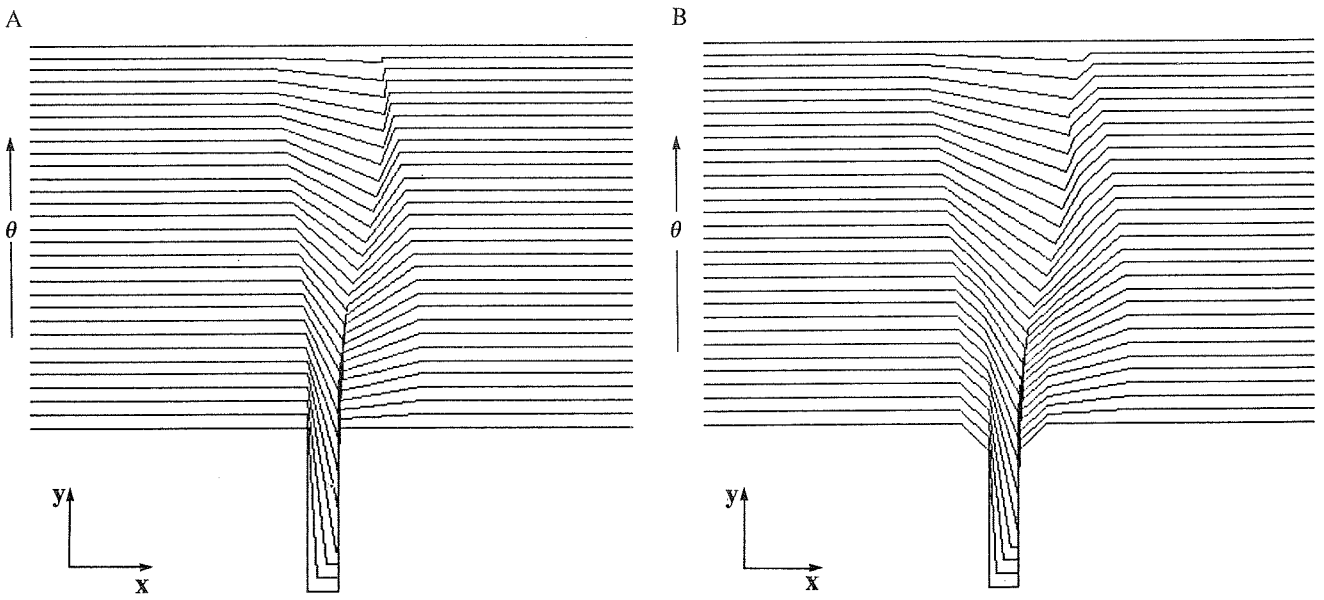
In this paper, we have limited ourselves to two-dimensional translation. It is possible, however, to extend the transformation approach to more general motions by using the *configuration space* (*C-space*) of a task (Arnold 1980; Lozano-Pérez 1981; 1983). A *configuration* of an object is the set of parameters needed to specify completely the positions of all points of the object. The configuration of a rigid, two-dimensional object, for example, can be specified by two displacements and an angle, that of a rigid, three-dimensional object by three displacements and three angles, and that of a robot arm by its joint angles. For concreteness, we will deal exclusively with Cartesian configurations, for example, $(x\ y,\ \theta)$ for objects in the plane, and not joint-angle configurations. The space of all possible configurations for an object is known as the configuration space of that object. An object A is represented as a point in its C-space; the coordinates of that point are the configuration parameters of A.

Stationary obstacles in the environment of a moving object A can be mapped into the configuration space of A. The resulting *C-space obstacles* are those configurations of A that would lead to collisions between A and the obstacles. Configurations on the surface of the C-space obstacle due to B are those where some surface of A is just touching a surface of B. If A and B are both three-dimensional polyhedra, the surfaces of the C-space obstacle for B arise from each of the feasible contacts between vertices, edges, and faces of A and B (see Fig. 13) (Lozano-Pérez 1983). Therefore, each face of a C-space obstacle represents a particular type of geometric constraint on A. A range of positions (and orientations) of A can be represented as a volume in the C-space of A, and a motion of A is a curve in the C-space.

As an illustration of the use of C-space surfaces, consider the familiar two-dimensional, peg-in-hole problem from Fig. 3. We can construct a three-dimensional C-space of (x, y, θ) configurations of the peg. In this space, the hole defines an obstacle (see Fig. 14A). Note that although the resulting surfaces are curved, for each value of θ the (x, y) cross section of the C-space surfaces is polygonal. The surfaces represent one-point contacts, and the edges at the intersections of surfaces represent two-point contacts. Line-line contacts also give rise to edges at the intersections of one-point contact surfaces. Figure 14B shows cross sections for a peg and chamfered hole.

The C-space representation can be extended to more general kinematic situations. In general, motions subject to geometric and kinematic constraints can be defined as collections of equalities and inequalities that must hold among the parameters that determine

Fig. 14. Cross sections of peg-in-hole C-surfaces. A. No chamfer. B. Chamfer.



the configurations of the robot and the objects in the task. These inequalitis represent C-surfaces (Mason 1981). Take the constraint that a robot hand remain in contact with a crank handle as the handle rotates. The constraint relating the position of the hand (x, y), to the position of the crank axis (a constant) and its current angle, α, is a curve (one-dimensional surface) in the configuration space of the task, that is, the (x, y, α) space.

Our goal is to make the detailed analysis of assembly operations algorithmic by casting it in terms of C-surfaces. The purely geometric aspects of the analysis have been exploited in earlier work on obstacle avoidance (Brooks and Lozano-Pérez 1982; Lozano-Pérez 1981; 1983). C-surfaces also share many of the characteristics of "real" surfaces with respect to force analyses. This was exploited by Mason (1981) to synthesize compliant motions. The synthesis approach described here also requires a mechanism for computing the effects of friction. Recent work has resulted in a definition of friction cones for C-surfaces (Erdmann 1983). Work is underway to show that conditions for avoiding jamming for the peg in hole can be restated in terms of the relationship of applied forces to these C-space friction cones.

3. A General Framework

In the previous section, we illustrated an abstract planning algorithm for fine-motion strategies. Although that algorithm is representative of our approach to fine-motion synthesis, it is *not* the most general formulation of the approach. In particular, that algorithm embodies a restrictive assumption on the class of single-motion strategies. It only considers strategies obtained by discarding all velocity vectors that point into the friction cones of some subset of the task surfaces. In some cases, further restrictions of the class of velocity vectors would produce a better strategy. The algorithm of the previous section does not provide a mechanism for further restricting the range of velocities. More significantly, we have not provided a criterion for defining what a "better" strategy might be.

In this section, we will present a more general framework for our approach to fine-motion synthesis. Although the description of this approach takes the form of an algorithm, it is not detailed enough to be considered an effective procedure. Our goal here is to formulate the *correctness conditions* for a class of synthesis algorithms. This framework can be used to

elucidate to what extent particular synthesis methods (for the class of fine-motion strategies we are considering) are "optimal." In particular, we are interested in strategies that make the best possible use of sensory data.

Development of the general framework begins with a description of the form of termination predicates for motions, followed by a discussion of the pre-image definition, and of the necessity of passing multiple subgoals to recursive calls of the planner. The rest of the section consists of a formal description of the framework, expressed as an abstract algorithm, and an extended example of its application to the peg-in-hole problem of Section 2.

3.1. Termination Predicates

Much of the burden of interpreting uncertain information falls on the termination predicate, which must decide when the current goal has been achieved. It is obviously important that termination not be premature; otherwise subsequent motions will proceed on a false assumption. On the other hand, failure to terminate the motion when the goal is demonstrably attained is also bad; the missed opportunity could prevent successful completion of the task. Hence it is important that the termination predicate make the best possible use of the available information.

One restriction is placed on the form of the termination predicates: we will exclude predicates that record sensory data for later use. The decision to terminate the motion must be made based on current sensor readings alone. As we shall see, another mechanism encodes some history, so this constraint is not as debilitating as it may first appear. If later developments suggest that this restriction should be relaxed, the framework can be modified by allowing a state function to be defined along with each predicate.

The form of the termination predicate will be introduced first with the assumption of perfect sensing and control. Consider the situation just after a command has been issued. Given perfect knowledge of the initial position, a perfect controller, and good dynamic models, the planner could predict the subsequent trajectory of the robot. If the position and force sensors were perfect, it would be a simple matter to watch the sensors, or the time, and halt the motion when the robot reached a goal.

To address more realistic problems, we will first relax the assumption of perfect sensing. The planner still knows what trajectory the robot will follow, but the sensing information cannot be taken at face value. It is necessary to construct an interpretation of the sensory data, which will be the set of all positions and velocities consistent with the sensory data and with the trajectory. Once this is accomplished, termination is again simple: if this interpretation of the sensory data is a subset of a goal, the motion is terminated.

The final step is to relax the assumption of perfect control and known initial position. Suppose we have a set of possible initial positions and a set of possible nominal velocities. Each different combination of initial position and nominal velocity will give a different robot trajectory. Without knowing which trajectory is the "real" one, the predicate must terminate the motion within a goal. To see how this is done, imagine that there is a different robot, for each trajectory and that all of the trajectories are being executed simultaneously. For each robot, we can apply the procedure of the previous paragraph: (1) form the set of positions and velocities consistent with sensory data, (2) intersect with the trajectory, and (3) check for inclusion in a goal. If the robots all agree that a goal has been achieved, the motion is terminated. This approach guarantees that for any initial position and nominal velocity consistent with the robot's information, and for any position and velocity consistent with observations, termination will occur only if a goal is attained.

In constructing the termination predicate, it is important to bring all possible information to bear, so that the set of "virtual robots" can be made as small as possible. Thus far, we have concentrated on the information encoded in the sensory data, but there is another important source of information. Formulating a subgoal R and calling the planner recursively have two important effects. First, the robot moves to R. Second, and more to the point, when the recursive call returns and the motion is executed, the planner *knows* that the robot is in R. Consider, for example, that a robot is lightly touching a vertical wall, and suppose that the subgoal R is the wall. Although the robot is in R, this fact might not be apparent to the robot if the

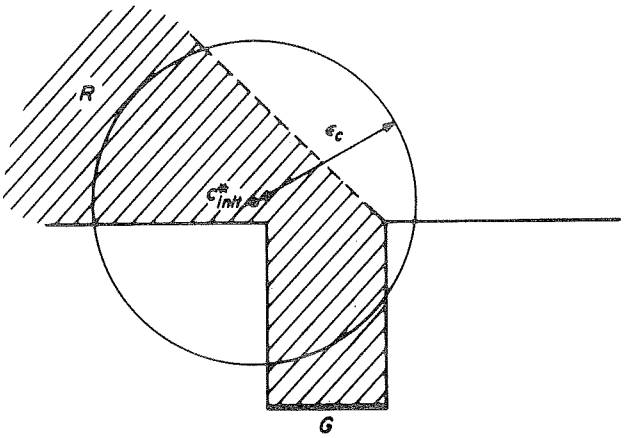


Fig. 15. Some history, represented by the subgoal R, is required to proceed into the hole.

contact force is small and if the position sensors are noisy. Hence the planner is called recursively to "move" the robot to the wall. Presumably, the planner will plan a horizontal motion into the wall. When the motion command is executed, the robot will not move, which the termination predicate will interpret as evidence that the "motion" was successful. When the recursive invocation of the planner returns, it will have accomplished its mission, even though it did not move the robot at all.

Another, more familiar, example illustrates the use of this information to construct the termination predicate. Suppose the planner is applied to the point-in-hole problem, with the position sensor giving a position at the lip of the hole (Fig. 15). Using the position sensor alone, the planner would have to admit the possibility of the robot being positioned anywhere inside the disk centered on the sensed position. To attain the goal with a single motion would be impossible. However, if the accomplished subgoal R is also consulted, the set of possible initial positions is reduced—the robot must be in the intersection of R with the disk. Starting from this smaller set of possible initial positions, with a command nominal velocity down to the right, it is easy to confirm that all the virtual robots will achieve the goal.

Thus the history of the robot, represented by the accomplished subgoal R, must be taken into account to construct the termination predicate. When the set of feasible trajectories is constructed, initial positions outside the subgoal R should be excluded. That the termination predicate is dependent on the accomplished subgoal R is an important observation, which profoundly affects the ultimate form of the planning algorithm.

3.2. Definition of Pre-image

The fundamental element in our approach to planning is the ability to construct a pre-image: a set of points from which the goal can be attained in a single motion. In Section 2, the pre-image depended on the goal G and a range of command nominal velocities. By proceeding more formally in this section, we find that the pre-image need not depend on the command nominal velocities, but that it does depend on the accomplished subgoal R.

Conceptually, we can approach this problem as follows: For goal G, and for every possible observed initial position c^*_{init} and accomplished subgoal R, construct the set $S(c^*_{init}, R, G)$ of all command nominal velocities such that the termination predicate, constructed as in Section 3.1, is guaranteed to terminate the motion. If $S(c^*_{init}, R, G)$ is empty, there is no single motion that can be guaranteed to work for accomplished subgoal R and observed initial position c^*_{init}. If $S(c^*_{init}, R, G)$ is "nonempty," then any element of $S(c^*_{init}, R, G)$ is sufficient to attain the goal. Now if the actual initial position of the manipulator is c_{init}, the observed initial position c^*_{init} could be anywhere in the sphere $B(c_{init})$ centered on c_{init} with radius equal to the tolerance on the position sensor. c_{init} should be in the pre-image if and only if every possible c^*_{init} gives a nonempty $S(c^*_{init}, R, G)$. Hence we can define the pre-image $P_R(G)$ of a goal G:

$$P_R(G) = \{c_{init} \in R | \forall\ c^*_{init} \in B(c_{init}),\ S(c^*_{init}, R, G) \neq \emptyset\}.$$

The subscript R is used as a reminder that the pre-image depends on R. Note also that the definition of $P_R(G)$ excludes points outside R. To have a point in $P_R(G)$ but not in R wouldn't make much sense—such a point would be a good place for the robot to be, provided that it is somewhere else!

3.3. Recursive Calls and Multiple Goals

When the planner is first called, the robot might be anywhere in configuration space C. If the set of strategies guaranteed to attain the goal G from any point in configuration space $S(c^*_{init}, C, G)$ is nonempty, then the planner can choose and execute one of these strategies. If $S(c^*_{init}, C, G)$ is empty, the planner must construct suitable subgoals and initiate a recursive call to the planner to achieve these subgoals.

Clearly, the planner should specify as subgoals only those sets from which it can achieve the goal, otherwise a recursive call will serve no purpose. To characterize these subgoals more precisely, let us look ahead a bit and imagine that the recursive call to the planner has just returned. The recursive call guarantees that the robot's position is now in R. Thus the planner must plan a single motion, from initial position c_{init} in R, which attains the goal G. By construction of the pre-image, such a motion exists only if c_{init} is in the pre-image $P_R(G)$. This observation serves to define suitable subgoals: R should include only those points that are also in $P_R(G)$, that is, $R \subseteq P_R(G)$. Since, by construction, $R \supseteq P_R(G)$, we can restate this observation: R is a suitable subgoal if and only if $R = P_R(G)$.

Thus, any set satisfying the equation $R = P_R(G)$ is a suitable subgoal. In general, a multitude of sets satisfies this equation. For instance, if R satisfies the equation, so does any subset of R. The question is what to do with this multitude of subgoals. Do we pass them to recursive calls one at a time? Needless to say, the branching factor in this search can be rather large. Another issue takes precedence, however. Situations occur in which the planner can be certain to attain one of two goals, but it cannot be known in advance *which* of the two goals it will attain. If the planner were passed either one of the two goals individually, it would fail to find a predicate guaranteed to terminate the motion. With both goals in hand simultaneously, it can plan a motion with confidence that it will ultimulately report which of the goals was attained. Hence we will pass all subgoals to the recursive call. This suggests that the approach be implemented without search, but we are not certain whether such an implementation will be possible

Since the planner will be passed multiple goals rather than a single goal, some adjustment of the notation is required. The set of goals will be written $\{G_\alpha\}$, the set of strategies guaranteed to attain one of the goals for given observed initial position c^*_{init} and accomplished subgoal R will be written $S(c^*_{init}, R, \{G_\alpha\})$, and the pre-image will be written $P_R(\{G_\alpha\})$.

3.4. A Formal Statement of the Framework

3.4.1. Nomenclature

Nomenclature used in the formal statement of the framework is as follows.

c is configuration.
c_{init} is configuration at the beginning of a motion.
v is velocity.
v_0 is nominal velocity.
c^* is observed configuration.
c^*_{init} is observed configuration at the beginning of a motion.
v^* is observed velocity.
v^*_0 is commanded nominal velocity.
t is time
C is C-space, that is, the set of all configurations.
$B(c)$ is the "uncertainty ball' of configurations, that is, the set of all configurations whose distance from c is within the tolerance of the position sensor.
$B(v)$ is the uncertainty ball of velocities.
$B(v_0)$ is the uncertainty ball of nominal velocities.
$\{G_\alpha\}$ is the current goal set. We wish to move the robot to one of the goals and return the identity of the goal.
$p(c^*, v^*, t)$ is the termination predicate. For each goal in $\{G_\alpha\}$ it returns one of the following: BUG, indicating that no possible trajectory is consistent with any interpretation of the sensory data; CONTINUE, indicating that at least one possible trajectory exists, consistent with the sensory data not just at the goal; or WIN, indicating that all possible trajectories consistent with the sensory data are in the goal.
$S(c^*_{init}, R, \{G_\alpha\})$ is the set $\{(v^*_0, p(c^*, v^*, t))|p$ terminates$\}$. By construction of the predicates, guaranteed termination implies guaranteed attainment of a goal. So for a given observed initial configuration and accomplished subgoal R, this gives

the set of all winning strategies, where a strategy comprises a command nominal velocity and a termination predicate.

$P_R(\{G_\alpha\})$ is the pre-image $\{c_{init} \in R | \forall c^*_{init} \in B(c_{init}), S(c^*_{init}, R, \{G_\alpha\}) \neq \emptyset\}$.

$\{R_\beta\}$ is the sets of configurations R such that the pre-image $P_R(\{G_\alpha\})$ includes all of R; that is, $P_R(\{G_\alpha\}) = R$. This is the subgoal set. Satisfaction of an element of this set by a recursive call will allow us to satisfy the current goal set.

R is the subgoal attained by a recursive call to the planner.

MotorCommand (v_0^*): execution of this program statement transmits the commanded nominal velocity to the controller, causing the manipulator to execute the planned generalized-damper strategy.

$D_{i,j}(t)$ is the actual trajectory; it returns (c, v), the actual configuration and velocity at time t, for initial position $c_{init,i}$ and nominal velocity $v_{0,j}$

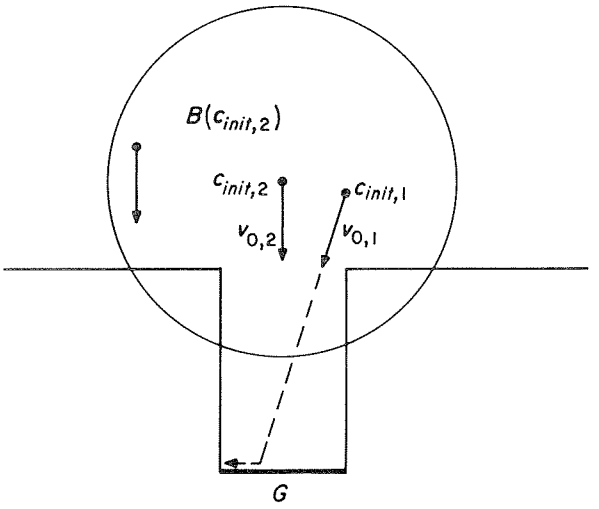
3.4.2. *The Algorithm*

```
Procedure FM({G_α})
  Compute {R_β}
  If C is in {R_β}
    Then R ← C
    Else R ← FM({R_b})
  (v*_0, p) ← choose (S(c*_init, R, {G_α}))
  t ← 0
  MotorCommand (v*_0)
  L  {V_α} ← p(c*, v*, t)
     ForEach α Do If V_α = BUT Then Error
     ForEach α Do If V_α = WIN Then Return(G_α)
     Increment t
     Go L
  End FM
```

3.5. An Example

This section applies the algorithm of Section 3.4.2 to the two-dimensional, peg-in-hole problem (Fig. 16). FM is called with an initial goal set containing the single element G: the bottom of the hole. Let $D_{i,j}(t)$ denote an actual trajectory; it returns (c, v), the actual configuration and velocity at time t, for initial position $c_{init,i}$ and nominal velocity $v_{0,j}$.

Fig. 16. Task illustrating construction of p.



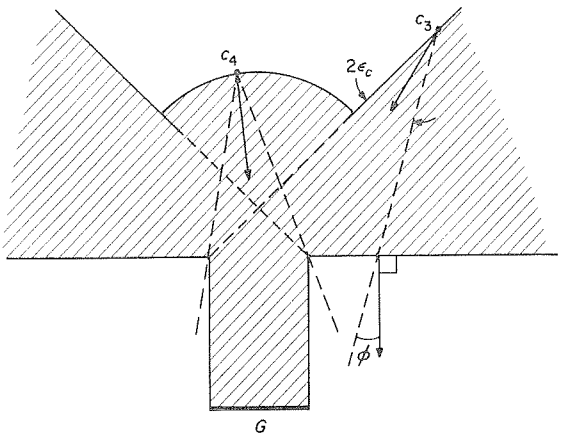
First, we construct an example illustrating construction of the termination predicate $p(c^*, v^*, t)$. Such a predicate must be constructed for each (c^*_{init}, v^*_0, R). Here is the predicate for $(c^*_{init,2}, v^*_{0,2}, R)$, assuming that R includes $B(c^*_{init,2})$.

```
Procedure p_2,2(c*, v*, t)
  Flag ← False
  For all (c_init,i, v_0,j) ∈ B(c*_init,2) × B(v*_0,2)
    (c, v) ← D_i,j(t)
    If (c, v) ∈ B(c*) × B(v*)
      Then If c ∈ G
        Then Flag ← True
        Else Return(CONTINUE)
  If Flag then Return(WIN)
    Else Return(BUG)
```

$S(c^*_{init}, R, G)$ is the set of all (v^*_0, p) that are guaranteed to WIN if executed at a point in $B(c^*_{init}) \cap R$. For our example point, this set is empty. For example, the command nominal velocity $v^*_{0,2}$ is not in $S(c^*_{init}, R, G)$ because trajectories from the left and right edges of $B(c^*_{init})$ will never reach the goal.

The example predicate results in the behavior specified in Sections 3.1 and 3.4, and is therefore "correct." The form of the predicate is not completely satisfactory, however. Goal attainment is tested simply

Fig. 17. Singleton R's.



by testing whether all possible predicted trajectories have attained the goal. This satisfies the requirements of the formalism because of the way control error is modeled—we have assumed that the control error is constant during a trajectory. A more realistic model of error would yield more robust predicates, combining position, velocity, and time information to detect presence at the goal.

The first step in the algorithm is to compute a set of subgoals $\{R_\beta\}$. Recall that each element R of $\{R_\beta\}$ is a set giving a pre-image $P_R(\{G_\alpha\})$ that is equal to R. The simplest way to begin is to construct the sets consisting of a single configuration. Such a set $R = \{c\}$ is valid if and only if it gives a pre-image $P_{\{c\}}(\{G_\alpha\})$ equal to $\{c\}$. Suppose the recursive call reports that the manipulator is at c. Then the question is whether a single motion command can move the manipulator to the goal G. This is possible for all configurations c in the shaded region of Fig. 17. This region is the union of two half planes and a circular disk (Turk 1983). A point in one of the half planes, such as c_3, can move to the hole by selecting a velocity that is guaranteed either to fall to one side of the hole and slide in, or hit the hole directly. A point in the circular region, such as c_4, can move to the hole by selecting a velocity that is guaranteed to hit the hole directly.

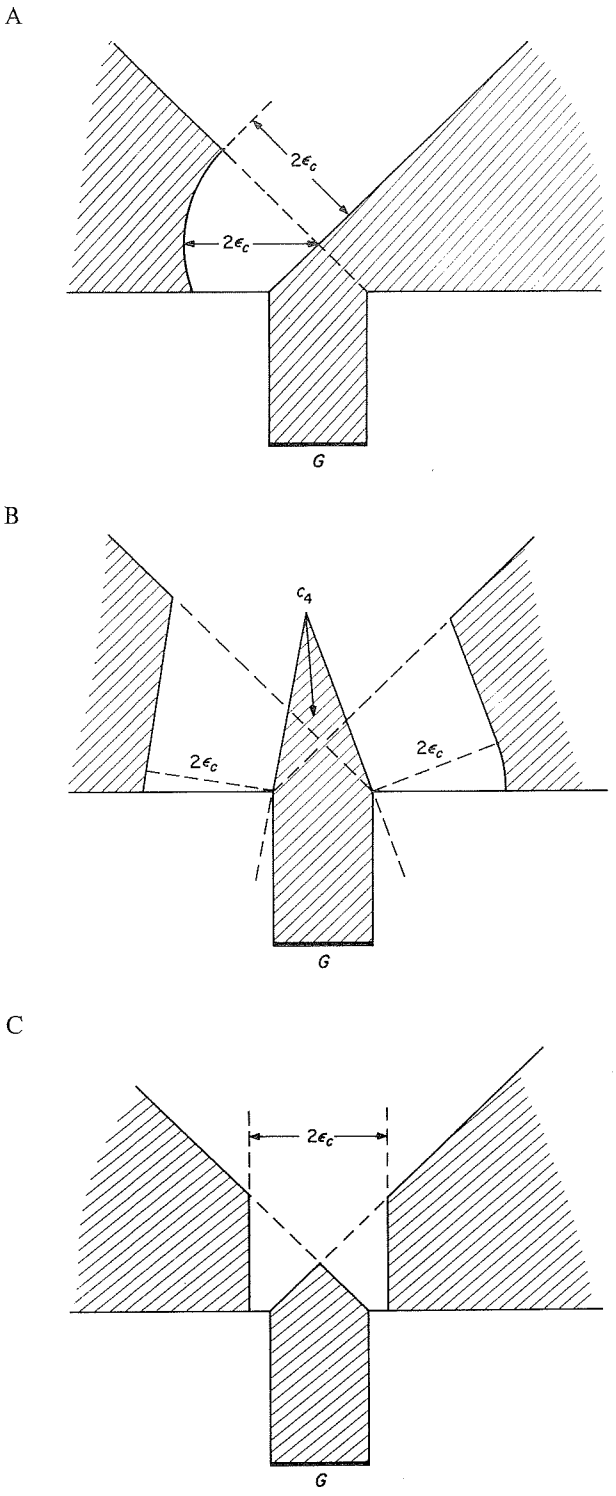
Any set R satisfying $R = P_R(\{G_\alpha\})$ must be a subset of the region indicated in Fig. 17. The shaded region does not itself constitute a good R, however. For instance, as in the earlier example, the planner might be unable to tell whether the robot is to the left or the right of the hole. Hence we must look for subsets R of the shaded region, each of which is equal to the corresponding pre-image $P_R(\{G_\alpha\})$. Three different such subsets are shown in Fig. 18. Consider, for example, the set shown in Fig. 18C. There are three different regions in this set: one region to the left of the hole, one region to the right of the hole, and one region in the hole. We can demonstrate that this R is equal to the corresponding $P_R(\{G_\alpha\})$ as follows. Suppose that a recursive call has reported that the manipulator is in R. We now consult the position sensor. If c^*_{init} is to the right of center, the manipulator cannot be in the left region, and the left-sliding command shown for point c_3 in Fig. 17 will work. Similarly if c^*_{init} is to the left of center, a right-sliding motion will work. This set works because any "incompatible" subsets—the left and right regions in this case—are separated by a distance of at least $2\epsilon_c$.

The sets shown in Fig. 18 are *maximal;* they are not subsets of any other subgoals. Since the subset of any valid subgoal is itself a valid subgoal, it would make sense to pass only the maximal sets to the recursive call. However, situations do occur for which maximal sets do not exist, so we will simply pass all valid subgoals to the recursive call.

Once the subgoal set $\{R_\beta\}$ has been determined, there is a recursive call to FM. As in Section 2.1, we will use a superscript numeral to indicate the "recursion level." Thus we write that the recursive call's goal set is the original call's subgoal set by writing $\{G^1_\alpha\} \leftarrow \{R^0_\beta\}$. Construction of a predicate for a multiple-goal set is a simple variation of the predicate constructed earlier.

```
Procedure p_{1,1}(c*, v*, t)
  Flat ← FALSE
  For All α Win_α ← TRUE
  For All (c_{init,i}, v_{0,j}) ∈ (R ∩ B(c*_{init,1})) × B(v*_{0,1})
    (c, v) ← D_{i,j}(t)
    If (c, v) ∈ B(c*) × B(v*)
      then Flag ← TRUE
      For All α If c ∉ G_α Then Win_α ← FALSE
  If Flag Then If For Some α Win_α = TRUE
    Then Return(WIN)
    Else Return(CONTINUE)
    Else Return(BUG)
```

Fig. 18. Maximal R's.



Now the recursive call must construct subgoals. In this case, the set of all configurations C is a valid subgoal. No further recursion is necessary, because one of the goals can be attained from any configuration whatever by use of a single motion command. The recursive call may immediately choose and execute a motion command. When the predicate terminates the motion, the identity of the subgoal attained is returned to the original call, which then chooses and executes a motion command carrying the manipulator to the bottom of the hole.

4. Conclusion

This paper has presented a formal approach to the synthesis of a class of fine-motion strategies. The approach operates directly from geometric descriptions of the task and explicit bounds on errors in sensing and motion. The basic method is structured around the computation of the pre-image of a goal region, that is, a set of configurations that can reach the goal using a single, compliant motion. We saw that the presence of errors in motion and sensing gives rise to a number of difficult problems in specifying motions and in deciding on termination conditions. Further work is in progress.

Beyond presenting a specific synthesis approach, the paper has attempted to (1) illustrate the usefulness of modeling compliant, fine-motion strategies as generalized damper motions that slide on C-surfaces (corresponding to geometric constraints), and (2) establish correctness conditions for fine-motion programs operating under error in sensing and control. Our approach to these issues provides the foundation for our synthesis method. Moreover, we hope it may be useful to human programmers engaged in fine-motion synthesis.

The approach in this paper is part of an attempt to develop a unified approach to robot-motion planning, spanning-obstacle avoidance (Lozano-Pérez 1981; 1983; Brooks and Lozano-Pérez 1983), compliant motion (Mason 1981), pushing (Mason 1982), grasping (Lozano-Pérez 1981; Mason 1982), and (now) fine-motion strategies. We believe that if sophisticated, sensor-based motion strategies are to be routinely used in robotics, the analysis and synthesis of these strate-

gies cannot (or should not have to) be done by human programmers on a task-by-task basis. Moreover, we are in need of a theoretical basis for the development of the programming and control mechanisms best suited for sensor-based motion. For these reasons, there is a vital need for a unified (preferably mechanizable) approach to analysis and synthesis of robot motion. This paper is a step toward this goal.

Appendix: Compliance via Generalized Damping

Generalized damping is a very simple and flexible mechanism for implementing active compliance[3] (Whitney 1977). The basic approach is to define the desired behavior of the robot by the relation

$$\mathbf{f} = \mathbf{B}(\mathbf{v} - \mathbf{v}_0),$$

where $\mathbf{f}$ is the vector of forces acting on the moving object, $\mathbf{v}_0$ is the nominal velocity vector, and $\mathbf{v}$ is the actual velocity vector. In general, $\mathbf{f}$ is a vector of six Cartesian forces and torques, and $\mathbf{v}$ and $\mathbf{v}_0$ are vectors of six linear and rotational velocities. In our examples here, we limit ourselves to forces and linear displacements in the plane.

Allowing the damping matrix $\mathbf{B}$ to be an arbitrary matrix can produce unusual behavior. One popular example is to relate forces in the $-x$ directions to displacements in the $+y$ direction so that the robot will climb over obstacles. We will, however, limit ourselves to simple damping matrices. In particular, we assume $\mathbf{B}$ to be the a diagonal matrix $b\mathbf{I}$, with $\mathbf{b} > 0$. Note that the damper equation is now simplified to

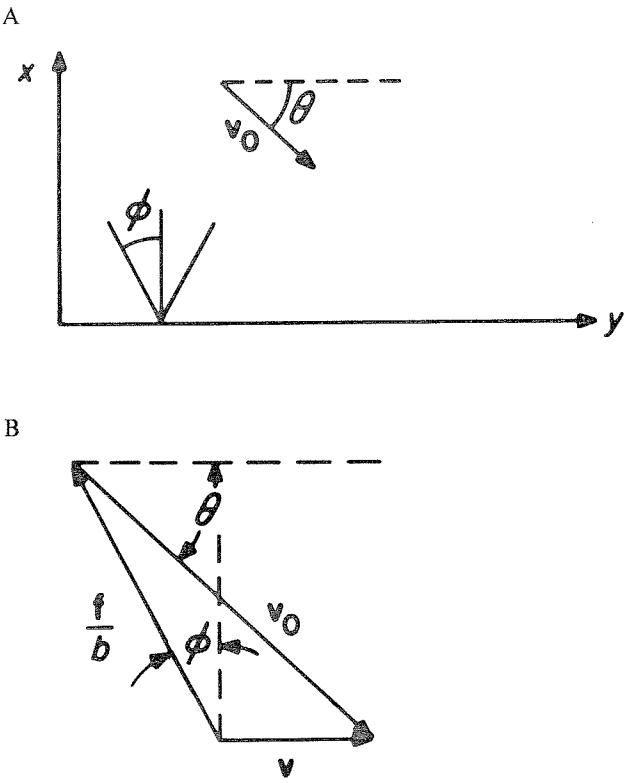
$$\mathbf{f} = b(\mathbf{v} - \mathbf{v}_0)$$

or, alternatively,

$$\mathbf{v} = \mathbf{v}_0 + \frac{1}{b}\mathbf{f}.$$

3. See Mason's paper (1983) for a discussion of generalized damping versus the generalized stiffness and hybrid control approaches to compliant motion.

Fig. 19. Geometry for generalized-damper analysis.



Consider an object controlled by a generalized damper with $\mathbf{v}_0$ (at an angle θ below horizontal) on a rigid surface whose normal points are along the y axis (see Fig. 19A). When the object strikes the surface, the objection can (1) slide to the right, (2) slide to the left, (3) remain motionless. We can use Coulomb's law to determine which of these three possibilities will occur.

First, consider case 1, in which the object slides to the right; the velocity $\mathbf{v}$ is horizontal. Coulomb's law dictates that the contact force $\mathbf{f}$ will make an angle $\phi = \tan^{-1}\mu$ with the surface normal. Using the damper equation in a simple construction in velocity space (Fig. 19B), we see that the nominal velocity angle θ must be less than $\pi/2 - \phi$. Case 2, in which the object slides to the left, is quite similar and yields the constraint that θ must be greater than $\pi/2 + \phi$.

Finally, consider case 3, in which the object sticks (remains motionless). The velocity $\mathbf{v}$ is zero. Coulomb's law gives a constraint on the force

$$\frac{\pi}{2} - \phi \leq \alpha \leq \frac{\pi}{2} + \phi,$$

where α is the angle that the force $\mathbf{f}$ makes with the horizontal. Again, the damper equation implies a corresponding constraint on θ:

$$\frac{\pi}{2} - \phi \leq \theta \leq \frac{\pi}{2} + \phi.$$

The analysis above yields constraints on the nominal velocity $\mathbf{v}_0$, given the motion of the object. We are also interested in the opposite: given the nominal velocity $\mathbf{v}_0$, what will be the resultant motion? In the present analysis, this is easily obtained. If the nominal velocity angle θ is less than $\pi/2 - \phi$, only case 1, sliding to the right, can occur. If the nominal velocity angle θ is greater than $\pi/2 + \phi$, only case 2, sliding to the left, can occur. If the nominal velocity angle θ is in the interior of the friction cone, that is,

$$\frac{\pi}{2} - \phi < \phi < \frac{\pi}{2} + \phi,$$

then only case 3, sticking, can occur. The only ambiguous cases occur when $\theta = \pi/2 + \phi$ or when $\theta = \pi/2 - \phi$. These cases are often referred to as cases of "impending motion."

Thus the class of nominal velocities that gives a desired motion on a given surface is easily characterized in terms of the friction cone of the surface, making the generalized damper an ideal control function for synthesis of fine motions.

Acknowledgments

We would like to thank Rodney Brooks, Steve Buckley, and Mike Erdmann for their helpful comments on earlier drafts.

References

Andreev, G. Y., and Laktionev, N. M. 1969. Contact stress during automatic assembly. *Russian Engineering J.* 49(11):57.

Arnold, V. I. 1980. *Mathematical methods of classical mechanics.* New York, Heidelberg, Berlin: Springer-Verlag.

Brooks, R. A. 1982. Symbolic error analysis and robot planning. *Int. J. Robotics Res.* 1(4):29.

Brooks, R. A., and Lozano-Pérez, T. 1982 (Dec.). A subdivision algorithm in configuration space for findpath with rotation. AI Memo 684. Cambridge, Mass.: Massachusetts Institute of Technology Aritifical Intelligence Laboratory.

Drake, S. H. 1977. Using compliance in lieu of sensory feedback for automatic assembly. Ph.D. thesis, Massachusetts Institute of Technology Department of Mechanical Engineering.

Dufay, B., and Latombe J. C., 1983 (Aug.). An approach to automatic robot programming based on inductive learning. Paper delivered at Int. Symp. Robotics Res. Bretton Woods, N.H.

Erdmann, M. 1983 (Jan.). On a representation of friction in configuration space. Unpublished report. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Goto, T., Takeyasu, K., and Inoyama, T. 1980. Control algorithm for precision insert operation robots. *IEEE Trans. Syst. Man Cybern.* SMC-10(1) 19–25.

Gusev, A. S. 1969. Automatic assembly of cylindrically shaped parts. *Russian Engineering J.* 49(11):53.

Hanafusa, H., and Asada H. 1977 (Tokyo). A robot hand with elastic fingers and its application to assembly process. Paper delivered at IFAC Symp. Information Contr., Problems in Manufacturing Technology. Reprinted in Brady, M. et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Inoue, H. 1974. Force feedback in precise assembly tasks. 1974 (Aug.). AIM-308. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory. Reprinted in Winston, P. H. and Brown, R. H. eds. 1979. *Artificial intelligence: An MIT perspective.* Cambridge, Mass.: MIT Press.

Laktionev, N. M., and Andreev, G. Y., 1966. Automatic assembly of parts. *Russian Engineering J.* 46(8):40.

Lozano-Pérez, T. 1976. The design of a mechanical assembly system. AI Tech. Rept. 397. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory. Reprinted in part in Winston, P. H., and Brown, R. H. eds. 1979. *Artificial intelligence: An MIT perspective.* Cambridge, Mass.: MIT Press.

Lozano-Pérez, T. 1981. Automatic planning of manipulator transfer movements. *IEEE Trans. Syst. Man Cybern.* SMC-11(10):681–689. Reprinted in Brady, M., et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Lozano-Pérez, T. 1983. Spatial planning: A configuration space approach. *IEEE Trans. Comput.* C-32(2):108–120.

McCallion, H., and Wong, P. C. 1975. Some thoughts on the automatic assembly of a peg and a hole. *Industrial Robot* 2(4):141–146.

Mason, M. T. 1981. Compliance and force control for computer controlled manipulators. *IEEE Trans. Syst. Man Cybern.* SMC-11(6):418–432. Reprinted in Brady, M., et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Mason, M. T. 1982. Manipulator grasping and pushing operations. Tech. Rept. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Mason, M. T. 1983. Compliant Motion. Brady, M., et al. eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Nilsson, N. 1980. *Principles of artificial intelligence.* 1980. Palo Alto, Calif.: Tioga.

Ohwovoriole, M. S., and Roth, B. 1981. A theory of parts mating for assembly automation. *Proc. Ro. Man. Sy. —81,* Warsaw, Poland.

Ohwovoriole, M. S., Roth, B., and Hill, J. 1980 (Mar. 5–7, Milan, Italy). On the theory of single and multiple insertions in industrial assemblies. *Proc. 10th Int. Symp. Industrial Robots.* Bedford, U.K.: IFS Publications, pp. 545–558.

Paul, R. P., and Shimano, B. 1976 (July 27–30, W. Lafayette, Ind.). Compliance and control. *Proc. Joint Automatic Control Conf.* New York: American Society of Mechanical Engineers, pp. 694–699. Reprinted in Brady, M., et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Raibert, M. H., and Craig, J. J. 1981 (June). Hybrid position/force control of manipulators. *J. Dyn. Syst. Measurement Contr.* 102:126–133. Reprinted in Brady, M., et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Salisbury, J. K. 1980 (Nov.). Active stiffness control of a manipulator in Cartesian coordinates. Paper delivered at IEEE Conf. Decision and Contr., Albuquerque, N. Mex.

Simons, J., et al. 1982. A self-learning automation with variable resolution for high precision assembly by industrial robots. *IEEE Trans. Automatic Contr.* AC-27(5): 1109–1113.

Simunovic, S. N. 1975 (Sept. 22–24, Chicago). Force information in assembly processes. *Proc. 5th Int. Symp. Industrial Robots.* Bedford, U.K.: IFS Publications, pp. 415–431.

Simunovic, S. N. 1979. An information approach to parts mating. Ph.D. thesis, Massachusetts Institute of Technology Department of Mechanical Engineering.

Taylor, R. H. 1976 (July). The synthesis of manipulator control programs from task-level spcifications. AIM-282. Stanford Univesity Artificial Intelligence Laboratory.

Turk, M. A. 1983 (May). Private communication.

Whitney, D. E. 1977 (June). Force feedback control of manipulator fine motions. *J. Dyn. Syst. Measurement Contr.* 98:91–97.

Whitney, D. E. 1982 (Mar.). Quasi-static assembly of compliantly supported rigid parts. *J. Dyn. Syst. Measurement Contr.* 104:65–77. Reprinted in Brady, M., et al., eds. 1983. *Robot motion.* Cambridge, Mass.: MIT Press.

Will, P. M., and Grossman, D. D. 1975. An experimental system for computer controlled mechanical assembly. *IEEE Trans. Comput.* C-24(9):879–888.