FINE-MOTION PLANNING FOR ROBOTIC ASSEMBLY IN LOCAL CONTACT SPACE

A Dissertation Presented

by

GORDON A. DAKIN

Submitted to the Graduate School of the

University of Massachusetts Amherst in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1994

Department of Computer Science

FINE-MOTION PLANNING FOR ROBOTIC ASSEMBLY IN LOCAL CONTACT SPACE

A Dissertation Presented

by

GORDON A. DAKIN

Approved as to style and content by:

Robin J. Popplestone, Chair

Andrew G. Barto, Member

Roderic A. Grupen, Member

Yanxi Liu, Member

James R. Smith, Member

W. Richards Adrion, Department Chair
Computer Science

ACKNOWLEDGEMENTS

I wish to thank all of the people who contributed to this thesis and provided me with scientific advice and moral support as a graduate student. My advisor Robin Popplestone offered expert guidance in the area of geometric reasoning and spurred my enthusiasm for research in spatial planning. I am grateful to Yanxi Liu for sharing her knowledge of symmetry group theory and assembly planning, and to Rod Grupen for moral support and for encouraging me to instantiate my planning technique on a physical assembly. The remaining thesis readers, Andy Barto and James Smith, supplied valuable comments and drew my attention to topics I would otherwise have overlooked. I also thank Rich Weiss for helping me to clarify my ideas through careful analysis. I enjoyed working with Chandu Ravela and Robin to develop the visual servoing facility, and I am greatful to Kamal Souccar and Vijay Gullapalli for their technical assistance in running the Zebra-Zero robot. I especially enjoyed reminiscing with Steve Smarz about musical oldies. Finally, I thank Elizeth Araujo, Jefferson Coelho, Chris Connolly, Manfred Huber, Ted Schnackertz, and all the other members of the Laboratory for Perceptual Robotics for their support and for helping me to use the labortatory facilities.

ABSTRACT

FINE-MOTION PLANNING FOR ROBOTIC ASSEMBLY IN LOCAL CONTACT SPACE

SEPTEMBER 1994

GORDON A. DAKIN,

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Robin J. Popplestone

This thesis presents a technique for modelling the 6–dimensional contact space of a 2–part assembly and planning compliant motion trajectories within that contact space. A nominal assembly motion plan is supplied *a priori* by a high-level planner, which derives moving part trajectories from kinematic constraints alone, disregarding the effects of uncertainty in the moving part's sensed pose and control velocity. The local contact space surrounding each critical configuration in the nominal trajectory is analyzed and represented as an adjacency graph of contact states, providing a search space for an appropriate sequence of contact state transitions. A heuristic graph search produces a sequence of reliable and recognizable contact state transitions. At a lower level of planning, a piecewise-linear trajectory is synthesized through uncertainty analysis. The resulting fine-motion plan is expressed as a sequence of hybrid position/force control strategy specifications.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Since the mid-seventies, an elusive goal of research in robotic assembly has been the automatic planning of assembly tasks. Ideally, an automatic assembly planning system would be capable of translating a set of goal relationships between the parts of an assembly into a sequence of robot-level commands. Among the several interrelated subproblems that have been addressed in the realm of assembly task planning [94] are *gross motion planning*, whereby collision-free motions are planned in free space, *grasp planning*, in which in which gripper configurations and grasp forces are selected, and *fine-motion planning*, which is the focus of this thesis. Fine-motion planning for assembly generally entails the planning of jam-free sliding motions for a mobile assembly part in contact with a stationary part, toward a goal configuration. In the approach to fine-motion planning presented here and elsewhere [45, 75, 79, 97], multistep fine-motion trajectories are planned within the *contact space* of the assembly. Consisting of poses in $\Re^6$ for which the assembly parts are "just in contact", the contact space is a subset of the assembly's 6–dimensional *configuration space* (or C-space) of translational and rotational pose parameters.

1.1 Motivation

During the past decade, research in fine-motion planning has revealed its inherent complexity. The sensitivity of the moving part's motion to erroneous sensory information, control error, and model inaccuracies has complicated the synthesis of command trajectories and compliances for jam-free assembly operations. In light of the effects of uncertainty, it has become clear that fine-motion plans cannot be chosen from a limited library of manipulator routines [92, 130], but call for a detailed analysis of the contact geometries involved, in order to predict the possible contact events that might occur. The need for a more rigorous treatment of uncertainty gave rise to the *LMT approach* to fine-motion planning, [97], in which multistep fine-motion trajectories were synthesized through recursive backchaining of subgoal regions through C-space. The LMT method's most general algorithms require exponential time in the number of plan-steps [30], however, so the LMT approach has not been implemented outside of very restrictive domains [22, 51]. The LMT approach fails to utilize "obvious" geometric information, such as part symmetries, that can simplify the planning process. In many assembly tasks, particularly those involving small clearances, the set of feasible sliding motions is centered around a single, nominal trajectory in free space which is determined by the symmetries of the assembly parts [87, 89, 90].

The view presented in this thesis is that the fine-motion planning problem can best be solved tractably if it is divided into two phases: (1) motion planning using kinematic constraints alone, without uncertainty considerations, followed by (2) perturbation of the resulting plan around the *critical point* configurations in the nominal trajectory, sites along the trajectory where the set of contacting surfaces changes, and collisions and jamming are likely to occur. Wheareas phase one falls into the domain of high-level assembly motion planning [87, 89, 90], this thesis is concerned with phase two, in which fine-motion trajectories are synthesized within the local contact spaces adjacent to the critical point configurations. The fine-motion planning method presented in this thesis distinguishes itself from other off-line, "two-phase" approaches [4, 17, 63, 92, 111, 120, 130, 137] by the degree to which the planning process has been automated, simplified, and is applicable to a fairly broad range of assembly part geometries.

1.2 Objective

A major goal of this thesis is to simplify the automatic planning of fine-motions, thereby furthering the feasibility of automatic assembly task planning. The simplifications introduced in this thesis are intended to expedite the synthesis of fine-motion trajectories, while retaining much of the rigor found in the more formal techniques [50, 97]. Several means of simplification are employed, from the analysis of contact space to the generation of fine-motion paths. The first simplification is the restriction of the planning domain to the local contact spaces centered around the critical points in the original nominal trajectory, which is provided *a priori* by a high-level assembly planner. Since the topology of an assembly's contact space does not vary between the critical points, all of the contact states of the assembly may be identified by examining these local contact spaces. A second simplification is embodied in the numerical techniques used to analyze contact space through the enumeration of its vertices. While borrowing the combinatorial method employed by Koutsou [75] to generate the vertices of contact space, this thesis employs the rapidly-converging Newton method, together with numerical predicates, to generate and test candidate vertices. Our numerical derivation of vertex positions is simpler and more general than that of Koutsou, who combined spatial relationships using a lookup table, constructed through an exhaustive, casewise derivation of substitution rules. Contact space analysis is further simplified by imposing artificial limits on the extent of contact space, restricting the planning domain to proper halfspaces of $\Re^6$. Although this simplification restricts the selection of traversable contact states, this policy allows the planner to ignore portions of contact space that are "mirror-images" of other, functionally-equivalent portions.

Finally, the planning of fine-motion trajectories through contact space is simplified by requiring the assembly to follow a predictable, determinate sequence of contact states, whose traversability is verified through uncertainty analysis. This simplification admittedly eliminates from consideration any fine-motion plan involving nondeterminate contact state transitions, which can accommodate a greater degree of uncertainty. The more powerful, but apparently intractable, LMT methods [50, 97], are capable of generating such fine-motion plans, so the class of assembly tasks to which they apply is larger than that of the current method. However, our determinacy requirement vastly simplifies the task of uncertainty analysis and reduces the problem of selecting a set of traversable contact states to a graph search of contact space. As shown in chapter 5.2.4, this simplification produces a worst-case, single-exponential time requirement, which is preferable to LMT's double-exponential requirement [28].

1.3 Introductory overview

This thesis presents a "two-phase" approach to fine-motion planning with uncertainty. As in other two-phase methods [4, 17, 63, 92, 111, 120, 130, 137] a nominal mating trajectory is provided *a priori* by a high-level assembly planner, which derives the trajectory from kinematic contraints alone. In the second phase of planning, the nominal trajectory is modified and augmented with compliant control strategies to accommodate the possibility of sensory and control error. Many two-phase approaches to fine-motion planning concentrate on the derivation of a corrective compliance which will allow the assembly to recover from upredictable contact events [63, 111, 120, 122]. The technique presented here focuses instead on modifying the nominal trajectory to restrict the path of the assembly to a predictable sequence of contact states.

The nominal trajectory provided by a high-level planner (for instance, *KA3* [87, 89, 90]) specifies the mating path a *moving part* relative to a *stationary part*, from free space to a goal pose. The high-level planner derives the nominal trajectory geometrically, from solid models whose dimensions do not allow for clearances at the insertion sites. The *critical points* in the nominal trajectory are then identified by the high-level planner, indicating the "trouble spots" where jamming and collisions are most likely to take place (see Figure 1). The critical points are the moving part locations where the set of contacting surface features changes along the nominal trajectory, when clearance-free solid models are involved. Nonzero insertion clearances, specified by the designer of the assembly parts, are introduced to the models just prior to the second phase of planning, which is the focus of this thesis.

Given a pair of solid models with specified insertion clearances, together with a set of critical point configurations, the goal of the fine-motion planner is to formulate an alternative command trajectory, in order to safely (without jamming or undesirable collisions) maneuver the moving part through the local contact space associated with each critical point. Figure 2, for instance, shows a command trajectory planned within a local contact space of a rectangular peg-in-hole assembly. The trajectory passes through a specified

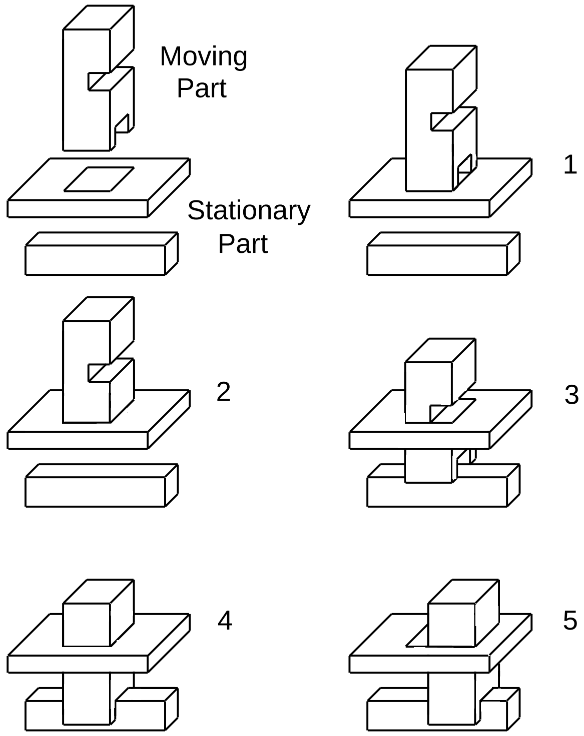


Figure 1: Critical points in a nominal trajectory.

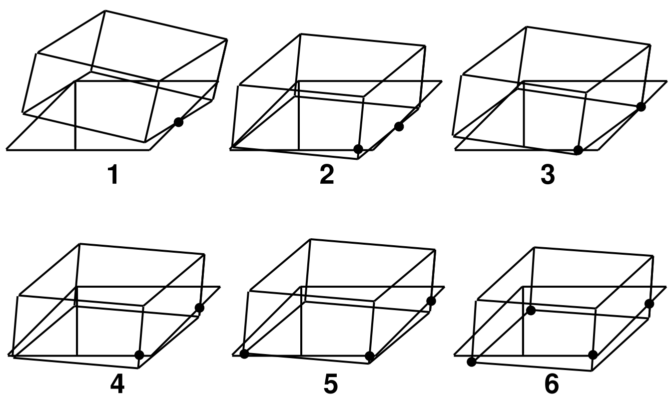


Figure 2: A path through a local contact space.

sequence of contact states within the local contact space surrounding the assembly's most prominent critical point. Between the critical points in an assembly, the trajectory can typically be confined to a 1-D or 2-D contact state, effectively minimizing uncertainty in the moving part's motion.

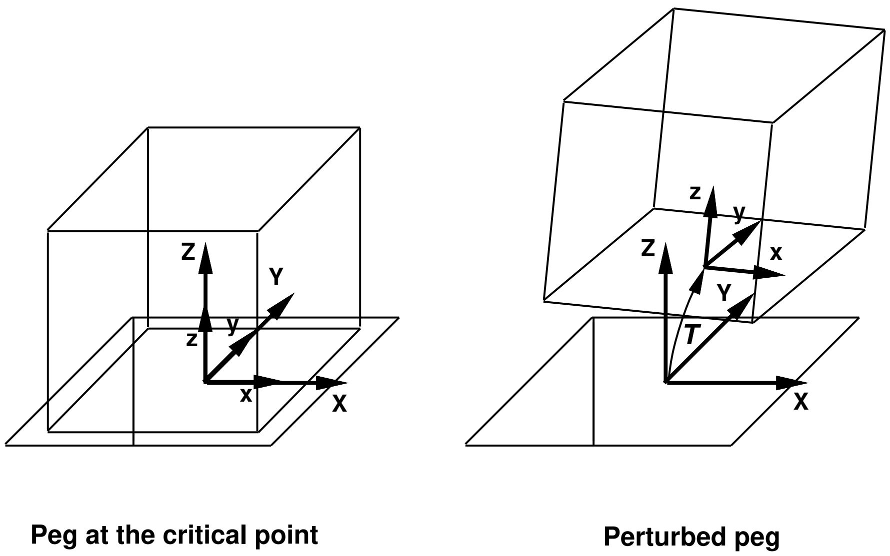


Figure 3: Poses in local contact space.

Each critical point pose serves as a reference frame for moving part displacements in the vicinity of the critical point. In Figure 3, for example, the moving part pose is expressed as a transformation T relative to the critical point frame. A moving part pose in $\Re^6$ represents its 3 translational, and 3 rotational, degrees of freedom (*dofs*). The critical point configuration corresponds to the origin of local configuration space, in which moving part locations are represented as 6-element pose vectors. Each local, 6-dimensional configuration space is partially surrounded by a local contact space, providing the planning domain in which to synthesize fine motions. The fine-motion planner searches each local contact space to form the most reliably-traversable sequence of contact state transitions through which to safely guide the assembly past the critical point. A fine-motion trajectory is also generated in contact space, to replace the original nominal trajectory.

To support the search for an optimal sequence of contact states within each critical point's local contact space, each contact space is constructed as an adjacency graph of contact states, containing all of the possible sequences of contact states in a fine-motion plan. The arcs in the adjacency graph represent contact state transitions involving the establishment or separation of a single contact at a time. Once a local contact space has been constructed in the form of an adjacency graph of contact states, fine-motion planning proceeds at two levels: the higher-level selection of a reliably-traversable sequence of contact states, and the lower-level synthesis of an optimal curvilinear trajectory through the chosen sequence of contact states.

The output of the fine-motion planner consists of a sequence of contact states in each local contact space, plus trajectories for traversing each sequence of states. Nominal force signatures are also calculated for each contact state, to facilitate the switching of compliant control strategies from one contact state to the next. By confining the assembly path to a predetermined sequence of contact states whose geometries can be analyzed off-line, it is possible to prespecify a hybrid control strategy for each separate contact state. The C-surface tangent space of a contact state determines which C-space axes must be position-controlled; and the remaining axes are force-controlled. The fine-motion trajectory synthesized for each contact state gives rise to the velocity commands for that state's hybrid control strategy.

1.4 Review of Related Literature

1.4.1 Robotic assembly planning

The automatic planning and specification of robotic assembly procedures has been studied at several hierarchical levels of detail. The number of levels and their definitions varies among authors. Homem de Mello [66] adopted the following four levels: (1) the assembly planning level, where a sequence of assembly tasks, each involving the mating of two subassemblies, is planned, (2) the task planning level, where a mating task is elaborated into a sequence of operations, such as fixturing or grasping a subassembly, (3) the operation programming level, in which a sequence of manipulator motions is specified to accomplish an operation (eg, planning gross- or fine-motion trajectories), and (4) the manipulator motion programming level, at which the trajectories and forces specified for an operation are translated into a robot control program. Barbera [6] described a hierarchical robot control architecture with levels analogous to those described above, plus an additional, top level, where multiple workstations are coordinated.

Lozano-Pérez [13] decomposed task planning into three subproblems: (1) grasp planning, in which gripper trajectories, grasp sites and contact forces are planned to produce a grasp that is stable (resistant to slip) and functional (supports exertion of the required forces), (2) gross motion planning, whereby collision-free motions are planned in free space, and (3) fine-motion planning, in which jam-free, compliant sliding motions are synthesized. Each of these subproblems has become an area of active research, summarized in later sections. Lozano-Pérez and Brooks [95] augmented these subproblems with three additional ones: (4) parts feeding, which deals with the manner in which each assembly part is introduced into the workspace, (5) layout, in which the location of every operation is specified, ideally minimizing the amount of time required to complete the assembly, and (6) fixturing, involving the selection of jigs and fixtures to maintain subassembly pose and stability in the presence of forces. In general, the solutions to these subproblems are interrelated.

1.4.2 The planning of assembly sequences

At the assembly-planning level, a sequence of assembly tasks is chosen to achieve a desired goal state in the configuration of the assembly. Each sequential subassembly connection must not preclude the establishment of subsequent connections. Bourjault [11] introduced the *liaison graph* to represent the set of connections established between the parts of an assembly. By querying a human expert about possible establishment sequences, an *establishment condition* predicate was derived for each connection in an assembly. Once the establishment condition was obtained for each connection, the feasible task sequences were enumerated. Defazio and Whitney [41] improved Bourjault's technique by making the acquisition of establishment conditions more efficient. The resulting set of feasible task sequences was represented as a lattice of liaison graphs, whose traversal from top to bottom yielded a feasible sequence of assembly tasks.

Homem de Mello [66] adopted an AND/OR graph representation of assembly. Each node in the graph represented a subassembly, with a disjuction of subtrees representing alternative decompositions into consituent subassemblies. A solution tree extracted from the AND/OR graph described a partially-ordered sequence of assembly tasks. Liu's assembly planner $\mathcal{KA}3$ [87, 89, 90] derived partially-ordered assembly tasks at a finer level of detail, concerning motions between assembly configurations at which the set of contacting surface features changes.

In addition to considering possible preclusions of subsequent connections, the evaluation of a feasible ordering of assembly tasks may also be based on mobility analysis [66, 82, 87, 89, 90] stability and weight of the subassemblies [82], availability of resources in the assembly cell [68], and the parallelizability of assembly tasks [67]. Moreover, in the *strategic product design* [138] view of designing and manufacturing a product, the assembly sequence should be selected during the functional design of the product, to find that sequence which will optimally support the "design for assembly" [10] approach to product design.

1.4.3 Task-level languages and planners

Early research in robotic assembly focused on the development of task-level programming languages. The objective was to enable the user to specify an assembly task in terms of the desired spatial relationships between the parts. A task planner would then translate the desired relationships into the required robot configurations or motion commands. The majority of task planning languages and systems developed to date have focused on either task specification, i.e., calculation of the end-effector positions required to achieve desired spatial relationships between the assembly parts, or one or two of the subproblems that comprise task-level planning, such as gross-motion planning or grasp planning. The most comprehensive exception is TWAIN [95], a proposed task-planning architecture. TWAIN performs constraint propagation to solve the interdependent subproblems of grasp planning, gross motion planning, fine-motion planning, parts feeding, layout, and fixturing. The most fully-implemented task planner is Handey [96], a robot system capable of executing commands of the form "MOVE object TO destination" within a fixed, polyhedral environment. The system locates a part within a pile of objects and plans a grasp and the necessary gross motions to acquire and transport the part to the specified location.

The RAPT system, created by Popplestone and his colleagues [114], is a task-specification system which translates symbolic spatial relationships, such as "Part-A AGAINST Part-B" into assembly part and end-effector locations. In Ambler and Popplestone [1], a graph was constructed to represent the spatial relationships between a set of bodies, and each cycle in the graph yielded a matrix equation in assembly part location variables. The matrix equations were then expanded into scalar equations and matched with standard equality forms whose solutions were available by formula. Latombe and Mazer [77] developed the robot programming language LM, and later LM-GEO [104], which were similar in essence to RAPT, but relied on vector equations to describe the spatial relationships. A later version of RAPT [35] recognized frequently occurring pairs of relationships that could be replaced with single relationships, eventually constraining the body to a fixed pose. Compositions or cycles of two or more relations in the graph were replaced, if possible, by a single relation representing the composition or intersection of the original relations.

Popplestone [113] later introduced symmetry group theory to the inference of body locations in robotic task planning. Spatial relationship were described in terms of cosets of continuous and discrete symmetry groups. The symmetry group formulation of relationships helps to mitigate the combinatorics arising from multiple relationships. Thomas and Torras [131] combined

Hervé's group-theoretic formulation of continuous groups with RAPT, using a lookup table to perform group compositions and intersections. Liu [87] and Popplestone, *et al.* [115] developed an efficient method of performing symmetry group compositions and intersections, by operating on the groups' *characteristic invariants*, geometric entities such as points and lines, by which the translational and rotational components of a symmetry group were more concisely represented. Liu formalized this approach in [85, 87], proving a one-to-one correspondence between *TR subgroups*[1] and their characteristic invariants. Hence, the intersection of two assembly features' subgroups was indicated by the group intersection's characteristic invariant, which was readily obtained by performing simple geometric computations on the invariants of two subgroups in question [88].

Liu's assembly planner $\mathcal{KA}3$ [87] is incorporated into the the fine-motion planning paradigm of chapter 2.7, where $\mathcal{KA}3$ [87] serves as the "high-level" planner which produces a nominal assembly plan from kinematic constraints. $\mathcal{KA}3$ is unique among assembly planners with respect to its input, which only describes the assembly parts to be mated, not the choice of contacting surface features. This minimal assembly task specification is characterized by $\mathcal{KA}3$ as a constraint satisfaction problem. In the initial stage of planning, feasible sets of mating features are enumerated [86], and the assembly configuration corresponding to each set of surface contacts is inferred by intersecting the TR subgroups of the contacts. Kinematic and spatial legality conditions are then checked, to produce feasible assembly configurations. In the latter stage of planning, a disassembly analysis is performed for each feasible configuration, yielding a *partially-ordered assembly motion set* (POAMS), represented as

[1] A TR subgroup is a subgroup of the Euclidean group whose rotational and translational components can be "separated" into subgroups of sequential motions.

a directed graph of contact states. Every path in the POAMS from free space to the goal configuration constitutes a candidate sequence of assembly motions.

Taylor [130] extended RAPT's treatment of spatial relationships to non-contact relationships, represented by location variable inequalities. He designed a *task checker* capable of forward-propagating numerical range limits on the unconstrained location variables to obtain bounds on the relative positions of the bodies. The resulting uncertainty estimates were used in the selection and iterative parameterization of *task skeletons*, manipulator-level commands such as "grasp", "move to", and so on. LAMA [92] also verified the choice of task skeleton parameters by forward-propagating uncertainty bounds. Generalizing the latter approaches to task-checking, Brooks [17] represented relationship constraints symbolically, so that inequalities expressing the conditions for a successful a mating task could be back-propagated to derive initial conditions for the plan variables. To perform the constraint propagation symbolically, Brooks built a constraint manipulation system [14] based on the $SUP-INF$ method of Bledsoe [7], whereby the extreme values of an unknown variable were recursively expressed in terms of subvariables' exteme values. This method of constraint propagation was also employed in TWAIN [95], to parameterize the task skeletons selected for the interwoven subproblems in an assembly task.

1.4.4 Gross motion planning

Most approaches to gross motion planning fall into one of three categories: (1) hypothesize and test techniques, (2) evolutionary path planning, (3) decomposition methods, and (4) Voronoi diagrams. The hypothesize and test techniques and evolutionary methods rely on local geometric information, so they are generally faster but less reliable than the global methods. In gross motion planning, collision-free trajectories are usually planned within the *configuration space* (C-space) of the robot, representing the robot's location as a point in the parameter space of its translational and rotational *dofs*. For a manipulator, the C-space axes are defined with respect to the joint variables, while for rigid objects such as assembly parts, the C-space axes are defined with respect to the object's 3 translational and 3 rotational *dofs*.

In the *hypothesize and test* approach, a planner first proposes a path, then iteratively checks for collisions between the robot and the environment and modifies the path until collisions can no longer occur. This technique was introduced by Pieper [112] to iteratively modify the trajectory of a 6-*dof* revolute manipulator. Pieper, as well as Myers [106], used a collision predicate to detect intersections between the robot's path and obstacles in the workspace, plus heuristics (e.g., "raise joint upward") to modify the trajectory whenever a collision was predicted. Methods used for detecting collisions have also included (1) static interference detection [105], (2) sweeping out volumes along the robot's trajectory [12], and (3) detecting interference in 4−dimensional space-time [25]. Due to the local nature of the overlap information used in the hypothesize-and-test approach, major modifications are not possible, so longer paths than necessary are generated.

The hypothesize and test approach to motion planning evolved into the *flexible trajectory* approach, whereby a trajectory was iteratively modified to increase the robot's distance to the obstacles. The binary collision predicate was replaced by a real-valued cost function incorporating the robot's proximity to the obstacles in the environment, as well as path length. The initial trajectory was modified while descending the gradient of the cost function, until an optimal path was produced. The techniques of Luh and Lin [99] and Gilbert and Johnson [55] relied on distance calculations that returned zero for all overlapping configurations. Since the gradient was identically zero for such points in the trajectory, these methods could only be applied to the improvement of robot paths shown in advance to be free of collisions. To accommodate overlapping configurations in flexible trajectory planning, Buckley and Cameron [21, 26] used the *minimum translational distance* required to bring the manipulator "just in contact" with an obstacle.

Due in part to the need for rapid, repetitive distance calculations in the flexible trajectory methods, algorithms for computing the shortest distance between two bodies have been investigated separately (see [56] for a review). Chin and Wang [31] showed that the shortest distance between two polygons with a total of M vertices could be computed in $O(logM)$ time. Dobkin and Kirkpatrick [44] presented an $O(M)$ algorithm for polyhedra. Gilbert, Johnson, and Keerthi [56] implemented an algorithm based on mathematical programming techniques, which could calculate the shortest distance between convex polytopes in $\Re^n$ in $O(M)$ time. Noting that in flexible trajectory planning, the closest facets on two polyhedra are usually the same from one iterative step to the next, Lin and Canny [84] used precomputed applicability constraints for verifying the closest pair of features in constant time.

Somewhat related to the flexible trajectory techniques in their reliance on proximity functions, the *evolutionary path planning* methods trace out trajectories between obstacles, rather than modifying existing paths. Khatib [74] placed *attractor* goals amidts *repellant* obstacles to form a potential field whose steepest descent from a starting pose traced out a collision-free path to a goal configuration. Artificial forces generated by the obstacles and goals were incorporated into the real-time control of the manipulator. Hogan [65] incorporated the real-time employment of potential functions into an *impedance control* approach to manipulation, relating sensed and artificial forces to the end-effector's position, velocity, and acceleration. Impedance specifications provide the most general models of manipulation.

The evolutionary path planning approach is only successful when the problem is convex with respect to the initial conditions. Otherwise, the process of tracing out a trajectory becomes "stuck" at local minima. Rimon and Koditschek [118] developed *navigation functions* for path-planning in topologically simple C-spaces called *sphere worlds*, which were devoid of local minima. They later extended their technique to include a large class of geometrically complex C-spaces [119], by transforming semi-algebraic obstacle descriptions into the sphere-world domain. Connolly *et al.* [33] showed that by imposing Laplace's equation as a constraint on the potential function, the resulting *harnonic function* is free of local minima. They ultilized Dirichlet-style boundary conditions, assigning high potentials to obstacles and low potentials to goals, producing motions that were normal to the obstacle and goal surfaces. Noting that the use of Dirichlet-style boundary conditions can generate near-zero potential gradients, which are difficult to compute with precision, Tarassenko and Blake [129] opted for Neumann-style boundary conditions, which have fewer precision problems but lead to the undesirable "grazing" of C-space obstacles. Connolly and Grupen [34] combined the desirable qualities of the Dirichlet and Neumann-style boundary conditions by superposing the two potential function solutions.

In the decomposition approach to gross motion planning, a robot's C-space is subdivided into distinct, local cells, typically represented in an adjacency graph. The graph is then searched to find a connected sequence of cells from a starting configuration to a goal configuration. Udupa [134] first applied this technique to plan paths for the "boom" of the Stanford arm in its spherical configuration space. Lozano-Pérez [91, 93] applied this technique to the entire manipulator. The extent of each translational joint was represented by polyhedral free space cells, which formed slices of free space associated with the discretized angles of a single rotational joint. Faverjon [53], and later Lozano-Pérez [98], further refined this technique to accommodate manipulators with multiple rotary joints. Free space was represented implicitly by hierarchical quantization of all the joint angles, avoiding explicit C-surface descriptions.

Schwarz and Sharir [123, 124] employed the *cylindrical algebraic decomposition* of Collins [32] to solve the *generalized piano movers problem*, using quite general robots whose kinematics could be described algebraically. The algorithm is of theoretical, but not of practical, importance, since its running time is double-exponential in the number of degrees of freedom in the robot's configuration. Canny [27] produced a single-exponential time *roadmap algorithm* to solve the generalized movers problem, partitioning C-space with the coarsest possible *stratification*. His algorithm produced *freeways*, constituting the "backbones" of channels in free space. Though its single-exponential time complexity represented a significant improvement over the Collins decomposition, the algorithm was generally regarded as intractable.

In order to reduce the running time required by the roadmap algorithm, Canny and Lin [29] supplemented its global, algebraic calculations with a local, potential field planner. Using the distance calculation technique of [84] the local planner efficiently computed the roadmap's freeways as curves of maximum clearance from C-space obstacles. In an earlier attempt to mitigate the complexity of C-space subdivision planning, Donald [45] used heuristics to explore selectively the explicit 5-dimensional C-surfaces bounding C-space. Point navigation operators chose sliding directions to bring the robot configuration closer to the goal. Koutsou's planner [75] generated sliding motions, as well, through the 1-*skeleton* of contact space edges connecting the 0-dimensional vertices of contact space. Vertex points and edge curve equations were derived by combining spatial relationships, using an extension of the RAPT system [114]. To manage the combinatorics involved in computing contact space vertices from all the 6-tuples of 5-dimensional surfaces, Koutsou recommended imposing limits on the extent of various C-space dimensions.

Voronoi diagrams have been employed to partition real space or C-space into cells based on proximity to obstacles in the environment. A path planned on a Voronoi diagram has maximal clearance from the closest obstacles. This technique was applied to path planning with polygonal obstacles by Drysdale [49], to produce a *generalized Voronoi graph*. Brooks [16] approximated generalized Voronoi graphs with generalized cones through in free space, each consisting of a linear spine together with width functions to describe the clearance on either side. Brooks [15] later applied this technique to the planning of pick-and-place operations for a PUMA robot with 6 rotary joints. Donald [46] extended the use of generalized Voronoi graphs to motion planning in 6−dimensional C-space.

1.4.5 Visual recognition and localization of assembly parts

The use of computer vision to identify and localize assembly parts greatly increases the autonomy of a robotic assembly system. The identification of an assembly part typically involves matching one or more of the part's surface features to corresponding features in a geometric model. Matching of surface features is complicated by occlusion and defects in the assembly parts.

Most techniques for visually identifying an assembly part are based on the recognition of either a single *extended feature* on a part's surface, or numerous, small features. Ballard [5] employed a generalized Hough transform to recognize an object's distinguishing extended feature, consisting of large segments of the surface boundary. In contrast, Bolles and Cain [9] iteratively examined local features, chosen off-line for their disambiguating qualities, until a single object model remained consistent with the observations. Grimson and Lozano-Pérez [58] recognized polyhedral parts using sparse range data consisting of pairs of surface points and normals. The examination of each successive data pair reduced the number of consistent correspondences between data pairs and hypothesized model surfaces.

The pose of an assembly part is usually estimated by comparing the locations of its observed features with corresponding features in the geometric model. Gordon and Seering [57] used a planar light-striping technique to obtain piecewise-linear segments of range data from the surfaces of polyhedral objects. The pose of an object was determined by exploiting knowledge of the light-stripe's plane and the surface planes in the object model. Bolle and Cooper [8] assigned Gaussian distributions to range data points to characterize the uncertainty with which a surface point varied along the surface normal. A maximum-likelihood estimate of each surface feature's pose, and in turn, the pose of the assembly part, were then calculated. Thompson and Mundy [131] estimated the pose of known polyhedral objects without first establishing feature correspondences. For each *vertex pair* (two vertices and two adjacent edges) examined in the scene, every possible correspondence with a pair of model vertices gave rise to an object pose, which was entered into a 6-dimensional histogram. The greatest peak in the histogram then revealed the approximate pose.

The uncertainty accompanying an assembly part's estimated pose may be characterized as a point set surrounding the pose in 6-dimensional C-space, or by a multivariate probability distribution. In ACRONYM [14, 18], Brooks represented uncertainty point set boundaries symbolically, propagating feature uncertainty limits "downward" to predict bounds on an object feature's position in the scene, and "upward" to constrain the pose of the object. In the probabilistic realm, Durrant-Whyte devised a methodology for merging and propagating Gaussian distributions arising from erroneous sense data, while accounting for the known, invariant relationships between the features. Pose estimates and uncertainties arising from disparate and partial sense data were characterized as expected values and information matrices, to be merged through Kalman filtering.

1.4.6 Fine-motion planning

In general, assembly operations cannot be performed solely as position-controlled motions in free space, since uncertainty in the position and geometry of the assembly parts precludes collision-free mating. However, the final goal configuration of an assembly can usually be attained by executing a compliant sliding motion, whose underlying control strategy is carefully chosen to prevent assembly failure in the presence of position or control error. The planning of jam-free sliding motions toward a goal assembly configuration is referred to as *fine-motion planning* for assembly. Fine-motion planning typically relies on the compliant control of a *moving part* relative to a *stationary part*, whereby the forces exerted through the moving part's coordinate frame are made to relate to its position [120, 137] velocity [63, 97, 136], or acceleration [4, 72].

1.4.6.1 The two-phase approach

Early research in fine-motion planning involved the selection and parameterization of task skeletons [17, 92, 130] (see chapter 1.4.3), generic robot programs for basic assembly operations such as fitting a peg into a hole or packing a block into the corner of a box. Following Inoue's success in employing heuristics to select an appropriate control strategy for the peg-in-hole task [70], it was hoped that any assembly task could be matched to a task skeleton chosen from a small library of generic operations. It soon became apparent, however, that small changes in the dimensions of the parts or the degree of position uncertainty can drastically affect the fine-motion strategy required [97], so fine-motion planning research involving task skeletons has diminished.

In the "two-phase" approach to fine-motion planning, a nominal trajectory is first computed in free space, specifying the motion of the moving part relative to the stationary part, without considering the effects of error in the position, velocity, or models. The uncertainty in the moving part's motion, and the danger of jamming or deviating away from the goal, is accommodated in a second phase of planning. Uncertainty can be addressed in the second phase through the use of fixed compliances, logical switching of compliant control laws, and/or modification of the nominal plan. Task skeletons fall into the latter category, since parameters are iteratively adjusted after a plan checker propagates uncertainty bounds along kinematic chains [130] or through sequential steps in a plan [92] to reveal possible failures in the assembly operation. Caine's manual approach to fine-motion planning [23] featured the iterative modification of plan design parameters, which included goal configurations and applied forces.

The most straightforward use of compliance is to allow the moving part's surface to conform to the surface of the stationary part when the parts are unaligned. For a peg-in-hole operation involving the insertion of a sprinkler stem into a sprinkler head, Salisbury [120] chose high stiffnesses along the stem's axis of motion and low lateral stiffnesses. The low stiffnesses permitted the tapered stem to center itself, correcting any misalignments due to random positioning errors. In a somewhat more sophisticated use of compliance, off-diagonal terms in the compliance matrix are chosen to effect more drastic, corrective motions than simple surface alignments [136]. The derivation of damping control laws to recover from erroneous contact state transitions has been investigated recently [63, 111, 122].

Finally, uncertainty accommodation in the "two-phase" approach to fine-motion planning can involve the logical switching of control strategies, where deviations of the moving part from the nominal trajectory trigger error recovery strategies or patch-plans. These contingency plans may be computed on-line [42, 140] or off-line [3, 48]. The former is termed *replanning.* Whenever the assembly halts in an assembly configuration other than the goal, a force command is computed on-line to generate a corrective motion bringing the part back to the nominal trajectory. In Dufay and Latombe's off-line approach [48], the nominal, *ground plan* is iteratively amended with *patch plans* to recover from failure states encountered on-line.

1.4.6.2 Manual methods

The approaches to motion planning with uncertainty known as the "manual methods" [76] were developed for deriving applied force constraints to prevent jamming and wedging during peg-in-hole insertions. The first peg-in-hole force analyses were performed in 2-D, by Simunovic [125] and Whitney [137]. Assuming quasistatic conditions, they established inequality constraints on the forces and torques that may be exerted through the peg's coordinate frame without causing jamming or wedging. Whitney also derived constraints on the peg's artificial stiffness, which were incorporated into the design of the Remote Center of Compliance, a passively compliant peg support used for high-speed insertions [139]. Rajan, Burridge, and Schwarz [117], performed a more systematic analysis of the 2-dimensional peg-in-hole problem, partitioning applied wrench space into regions associated with the types of motion they would generate, which included breaking of contact, sliding with acceleration, and sticking.

Analysis of the peg-in-hole problem was generalized by Ohwovoriole to multiple insertions in 3 dimensions [109, 110]. Caine [23] and Sturges [128] extended Whitney's manual derivation of applied force constraints for jamming avoidance to the 3-dimensional rectangular peg-in-hole task. Caine [23] developed designer tools for manually constructing a fine-motion trajectory and deriving inequality constraints in 6-dimensional applied wrench space. Additional wrench constraints were derived to prevent the separation of existing contacts. Such constraints are useful for restricting the peg to traverse a preselected sequence of contact states. As demonstrated in [24], it is often necessary to restrict the set of contact states that might be encountered, if one wishes to obtain a single applied force satisfying constraints for jamming avoidance in all the anticipated contact states.

1.4.6.3 Natural and artificial constraints

The motion of an assembly part and the forces it encounters are constrained by the geometry of the assembly and the task to be performed. Mason [102] referred to the motion and force constraints arising from part geometries and rigidities as *natural constraints*, and those required by the task as *artificial constraints*. He observed that the normals and tangents of a C-surface in configuration space indicate the directions along which these constraints occur: the natural constraints for traversing a C-surface dictate zero velocity along every C-surface normal, and zero force along every C-surface tangent. The complementary, artificial constraints require zero force along every C-surface normal, plus a desired velocity within the tangent space. Since the constraints on the velocity and force are expressed in orthogonal spaces, they suitably specify a *hybrid control strategy* [116], whereby position (and velocity) control is executed along one set of end-effector axes, and force control is executed along an orthogonal set of axes.

Lee and Hou [80] devised an algorithm for selecting the *C-frame* in which the natural and artificial constraints of a task can be most simply expressed. By choosing C-frame axes that are parallel to the contact surfaces (whenever possible), every C-frame axis will be purely position-controlled or force-controlled. De Schutter and Van Brussel [43] developed a language for specifying a hybrid control strategy in terms of the C-frame, the natural and artificial constraints, and the C-frame's "feedworward" velocity. Their technique was generalized by Bruyenincks and DeSchutter [20] to twist-wrench hybrid control, providing a means for specifying and tracking the axes in 6-space which are to be position-controlled or force-controlled. Strip [127] derived hybrid control strategies manually for a polyhedral peg-in-hole assembly, claiming that the control strategies apply to a wide range of peg-in-hole geometries.

1.4.6.4 Fine-motion planning in contact space

A few researchers in fine-motion planning, the author included, have adopted contact space as the motion planning domain in which to construct fine-motion trajectories. Since each additional contact established between the moving part and stationary part reduces the mobility of the moving part, the traversal of lower-dimensional contact space facets is accompanied by a lower degree of uncertainty than motions in higher-dimensional facets of free space. The planning of explicit trajectories in contact space contrasts with the corrective compliance [63, 111, 122] or error recover methods [3, 42, 48, 140], which treat contact formations as unavoidable but correctable deviations from a nominal trajectory in free space.

Applying the C-space decomposition method of path planning to the planning of 2-dimensional objects in contact, Hopcroft and Wilfong [64] decomposed contact space into cells comprised of the path-connected components of contact space regions defined by sets of contact relationships. The resulting decomposition described the basic topology of contact space, reducing the motion-planning problem to a combinatorial search of a contact state graph. Koutsou [75] generalized this decomposition to 3-dimensional objects and devised a finer decomposition that produced a connected 1-skeleton of contact space. Consisting of the edges and vertices of contact space, the 1-skeleton provided a simple search space for planning a motion between two assembly configurations. The assembly configurations associated with the vertices were found by combining symbolic spatial relationships in the manner of RAPT [114], and the solid modeller *Robmod* [25] was used to discard vertices associated with spatial interference.

Laugier and Theveneau [79] built a system for constructing and searching a contact state graph, with arcs representing compliant motions. The motions, which were restricted to pure translations, were chosen heuristically while backward-chaining from the goal state. Heuristics included a preference for directions perpendicular or parallel to a given surface, and an avoidance of contact states where sticking might occur. After the graph of contact space trajectories was constructed, alternative trajectories were tested until one was produced that could be traversed in the presence of uncertainty. Valade [135] built a similar system for planning translational compliant motions, focusing on concavities in the parts.

1.4.6.5 The LMT approach

In the seminal paper of what has become known as the *LMT approach* to motion planning with uncertainty, Lozano-Pérez, Mason and Taylor [97] introduced a technique in which the synthesis of nominal motion commands was based on the degree of uncertainty in sensing and control. This approach contrasted with previous methods, in which uncertainty was not addressed until after a nominal motion was calculated. Motion planning in the LMT approach takes place in the configuration space of the robot. A damper control model is assumed, so a surface may traversed by choosing a command velocity that points toward the surface, but not into its friction cone.

In the simplest form of LMT planning, a starting region and a goal region are represented in configuration space, and a command velocity is chosen that optimizes the extent of the goal's *directional pre-image*, the region from which the robot is guaranteed to recognizably reach the goal region via the given command velocity. A *termination predicate* signals when the robot is guaranteed to have reached the goal region, in spite of uncertainty in the robot's sensed position and velocity. Once the goal's pre-image is calculated, it becomes a subgoal region, and the process recurses until the starting region is enveloped by the recursive pre-images of subgoals. The resulting sequence of command velocities constitutes the fine-motion plan.

More generally, the LMT approach employed a *nondirectional pre-image*, the region from which *some* command velocity existed (which the robot would determine on-line) with which to recognizably reach the goal. Planning with the larger, nondirectional pre-images is more powerful than with directional pre-images, since fewer steps are necessary during the recursive preimaging of subgoals. The most general form of LMT makes use of an even larger nondirectional pre-image, consisting of regions from which one of a *set* of goals may be reached. Mason [101] showed that this more complex pre-image is computable by combining simple, directional pre-images. Erdmann's work on configuration space friction cones [50] elaborated the process of calculating pre-images that preclude the possibility of sticking on a surface of configuration space. By avoiding velocities that pointed into a reachable surface's friction cone, the robot was guaranteed not to stick.

Canny [28] proved that the LMT method of compliant motion planning was computable in double-exponential time in the number of plan-steps. He showed that the problem of finding an n-step fine-motion plan from a starting region to a goal region could be reduced to a problem in the theory of reals involving $O(n)$ quantifiers, which could always be solved in $O(2^{2^n})$ time. Moreover, Canny and Reif [30] proved that the LMT compliant motion planning problem was exponential time hard. Owing to the large time-complexity of the general LMT method, implementations have only been attempted in restricted domains. Erdmann presented an algorithm that operates in a 2-dimensional, polygonal configuration space [51]. The algorithm relied on *backprojections*, which approximate preimages by requiring that recognizability be based on current position sensing alone.

Buckley [22] implemented an LMT-type planner that synthesized motions for polygonal and polyhedral objects moving via pure translations. Motion terminations were planned by allowing the robot to stick after reaching a goal state. Position commands for traveling from a starting state to a goal state were computed via set difference operations, subtracting the set of commands that *might* bring the robot to a nongoal state from the set of commands that *might* bring the robot to the goal state. Buckley's work is similar to that of Erdmann and Mason [52], who studied exclusively planar motions, but allowed rotations. Configuration space was divided into *atoms*, sets of points that shared a common set of possible reaction forces. Robot states were defined as sets of atoms that may be confused due to uncertainty in sensing or control.

Donald [47] is the only author who has included geometric model tolerancing as a source of uncertainty in the LMT approach. C-space is augmented with additional dimensions for size and shape parameters varying in known tolerance zones. Since variation in geometric tolerances may alter the gross topology of C-space, it is not always possible to construct a plan which is guaranteed to reach a goal region. Noting this, Donald introduced the notion of an *error detection and recovery strategy* (EDR), consisting of a motion plan that is guaranteed to succeed if the goal is attainable, or to signal failure if the goal cannot be reached. Each step in an EDR is constructed by modifying a motion strategy computed using nominal object descriptions. Since the dimensionality of generalized C-space grows with each variable shape parameter, this treatment of model uncertainty is computationally prohibitive when dealing with a large number of model tolerances.

1.4.6.6 Learning methods

Various learning techniques have been applied to the robotic acquisition of fine-motion skills. Dufay and Latombe [48] devised an inductive learning scheme in which responses to specific instances of assembly failure were generalized to form situation-action pairs. Patch-plans associating specific motion commands to specific sensor readings were created in response to assembly failures, and the patch-plans were then generalized to associate motion classes with continuous ranges of sensor values. Lee and Kim [81] incorporated inductive learning into EARSA, an expert system capable of learning fine-motions. Positive and negative examples of situation-action pairs were supplied by a human and then generalized by EARSA as hypothetical rules. EARSA then generated its own test examples to refine the hypotheses.

The learning of fine-motion control strategies has also been accomplished using neural networks. In the supervised learning paradigm, Asada [2] constructed a multi-layer neural network for acquiring nonlinear compliant control laws from samples of ideal force-velocity pairs. The first layers in the network recognized the contact state of the assembly, and the remaining layers computed the mapping of forces to velocities appropriate for the current state. In the realm of associative learning, Gullapalli [60, 61] devised a stochastic reinforcement scheme to learn nonlinear compliances. Velocity responses to input positions and forces were perturbed randomly, to aid in the exploration and evaluation of alternative compliance mappings. Using a separate network to model the scalar reinforcement signal, the expected reinforcement determined the variance with which exploratory responses were stochastically generated.

1.5 Relating this thesis to other methods

The author's fine-motion planning technique falls into the "two-phase" category of fine-motion planning methods. A nominal trajectory is first computed without considering the effects of position, velocity, or model error, and the resulting uncertainty in the moving part's trajectory is accommodated in a second phase of planning. Whereas several approaches accommodate uncertainty in the second phase via corrective compliance, using a single control strategy designed to correct deviations of the moving part from the nominal path [4, 63, 72, 120, 136], the present approach *modifies* the nominal trajectory with fine motions planned within local contact spaces surrounding critical points in the the nominal trajectory.

As in Koutsou [75], the topology of contact space is decomposed by first enumerating its vertices, and then listing the higher-dimensional contact states adjacent to the vertices. In Koutsou's method, the enumerative process ends after generating the 0$-$ and 1$-$dimensional states, while the author's technique produces all of the contact states up to the 5$-$dimensional primitive contacts. Whereas Koutsou's planner finds vertex points by combining 6$-$tuples of spatial relationships in the manner of RAPT, the current technique invokes the multivariable Newton method to solve for the vertex points. And while Koutou's planner relies on a solid modeller to tell if a candidate vertex lies in the forbidden zone, the present method invokes a numerical predicate to determine whether the vertex occupies the forbidden zone's point set in C-space.

As described in chapter 2.7, the fine motion trajectory planned within each local contact space is synthesized through iterative optimization, maximizing its distance to contact states outside of the selected set of states. The calculation of optimal trajectory segments is somewhat related to the *flexible trajectory* technique of gross motion planning [21, 55], whereby an arbitrary trajectory is first plotted, then optimized through a gradient descent procedure. In the current approach, a small number of goal points, which define a piecewise-linear trajectory through contact space, are modified during each optimization. As in Lin and Canny [84], repetitive distance calculations are facilitated by checking applicability conditions for nearest facets and "walking" to the appropriate facet when a condition is not met.

The present method's iterative modification of an existing trajectory, as well as its forward-propagation of uncertainty regions, distinguishes it from the LMT methods [50, 97], which employ backprojections or pre-images of uncertainty regions. The LMT approach involves the construction of complex uncertainty regions that may span several contact states at once. In order to side-step the overwhelming complexity of the LMT approach, the current method restricts the fine-motion trajectory to a single, well-chosen sequence of contact states. As in the approach of Caine and his colleagues [23, 24], a curvilinear command trajectory is constructed to reliably transport the assembly through the determinate sequence of contact states. Caine's preference for a determinate sequence of states simplified the manual task of designing a single force control strategy satisfying each state's jamming constraints. However, the current preference for a determinate sequence of states is motivated by the need to mitigate the complexity of automatically synthesizing fine motion trajectories while performing uncertainty analysis in every state traversed. Moreover, since the contact space facets to be traversed are known prior to the uncertainty analysis, there is no advantage in using back-propagation over forward-propagation, which is simpler to implement than the former.

CHAPTER 2

CONTACT SPACE ANALYSIS

In the approach to fine-motion planning presented in this thesis, a high-level assembly planner first provides a nominal trajectory derived from geometric constraints alone, using solid models whose dimensions do not yet allow for clearances at the insertion sites. Prior to the introduction of clearances, the moving part is confined to a restrictive contact space which permits little mobility at the sites of insertion. In a cylindrical peg-in-hole task, for example, the cylindrical peg is confined to a two-dimensional contact space facet as it traverses the hole interior, permitting only two degrees of freedom (*dofs*) of mobility. The insertion clearances are supplied by the designer of the parts, who specifies the degree to which each model face recedes in the direction of its inward normal, and the amount by which to decrease each cylindrical surface's radius.

After insertion clearances are introduced to the assembly part models, the moving part inhabits a 6-dimensional configuration space. This *free space* region is partially bounded by a contact space whose facets range in dimensionality from 0 to 5, depending on the assembly in question. Figure 4 illustrates the analogy in 2 dimensions, where the introduction of clearances gives rise to 3 *dofs* in the mobility of the moving part during the insertion. Assuming the moving part's nominal trajectory remains fixed while the faces of the part models recede inward along their surface normals the same nominal trajectory now defines a path through free space to the goal configuration. In Figure 4, for example, the rectangular peg travels through free space for the

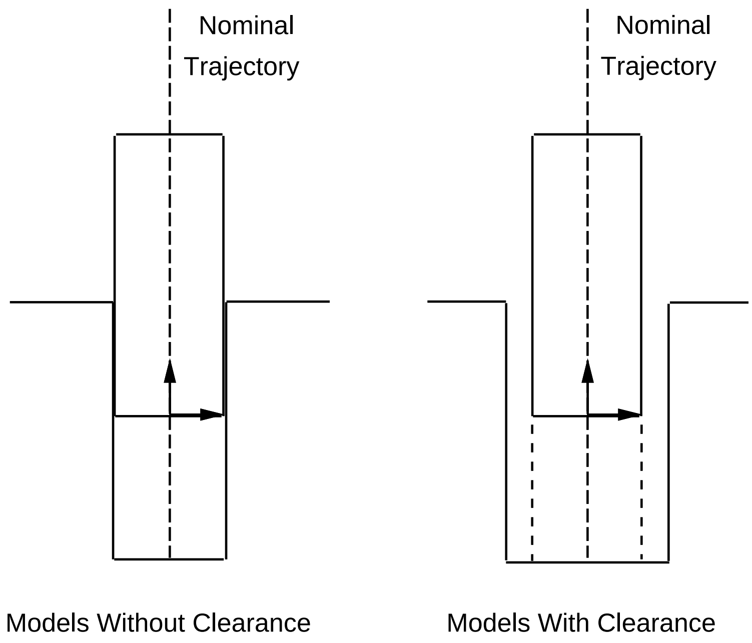


Figure 4: Introducing clearances to the part models.

entire motion until it reaches the final goal configuration.

In addition to supplying a nominal trajectory for the moving part, the high-level planner also identifies the *critical points* in the nominal trajectory, indicating the configurations where the set of contacting surface features changes as the clearance-free assembly traverses the nominal path. As shown in Figure 5, the critical points then indicate the sites where jamming and collisions are likely to occur during the assembly of the modified parts.

2.1 Perturbing the nominal trajectory

If the nominal trajectory were employed as the command trajectory supplied to the robot system, uncertainty in in the moving part's trajectory arising from sensory and control error and model inaccuracies would give rise to an unpredictable series of contact events. In fact, the central location

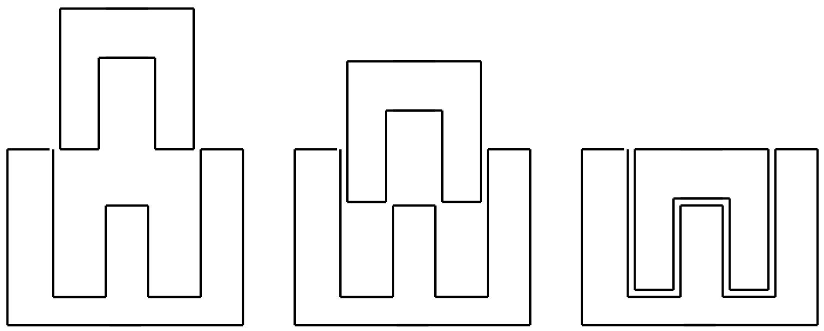


Figure 5: Critical points in the nominal trajectory.

and aligned attitude of the moving part generally *maximizes* the number of indeterminate contact states that might be encountered as the moving part deviates from the command trajectory. In order to reduce the non-determinacy in the assembly's behavior and to maneuver the moving part safely past each critical point region without sticking, the nominal trajectory is perturbed during the fine-motion planning process, tilting and displacing the moving part to constrain its motion to a predictable sequence of contact states. Since the critical point regions are the most problematic sites, fine-motion planning takes place within the local contact spaces surrounding the critical points. In between the critical points, the assembly motion can usually be confined to the traversal of a lower-dimensional contact state shared by the local contact spaces of the surrounding critical points.

2.2 Local contact spaces

To support the search for a determinate sequence of contact state transitions through which to guide the moving part safely and predictably past each critical point, an adjacency graph of contact states is constructed to represent the local contact space surrounding each critical point. The analysis and representation of a local contact space is the subject of the current section. As described in the following section, a heuristic search of each contact state graph yields a sequence of contact states judged to be traversable in the presence of uncertainty.

Within each local C-space, the moving part's pose is characterized as a 6-element point vector $\mathbf{X}$ in $\Re^6$, consisting of a translational perturbation $\mathbf{dx} = [dx\, dy\, dz]^T$ and a rotational perturbation $\delta\mathbf{x} = [\delta x\, \delta y\, \delta z]^T$ of the moving part away from the C-space origin (the critical point configuration). The translational component $\mathbf{dx}$ represents the homogeneous transformation

$$trans(dx, dy, dz) = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

and the rotational component $\delta\mathbf{x}$ denotes *roll-pitch-yaw* angles of successive rotations about the fixed, global z, y, and x axes:

$$RPY(\delta x, \delta y, \delta z) = rotz(\delta z)\, roty(\delta y)\, rotx(\delta x) \qquad (2)$$

$$= \begin{pmatrix} C_z & -S_z & 0 & 0 \\ S_z & C_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C_y & 0 & S_y & 0 \\ 0 & 1 & 0 & 0 \\ -S_y & 0 & C_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_x & -S_x & 0 \\ 0 & S_x & C_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where the S_i and C_i denote the sine and cosine of δx_i.

2.3 Primitive contacts

The local 6-dimensional configuration space surrounding a critical point is bounded by 5-dimensional C-surfaces, which correspond to the *primitive contacts* (PCs) that can occur near the critical point. For polyhedral objects,

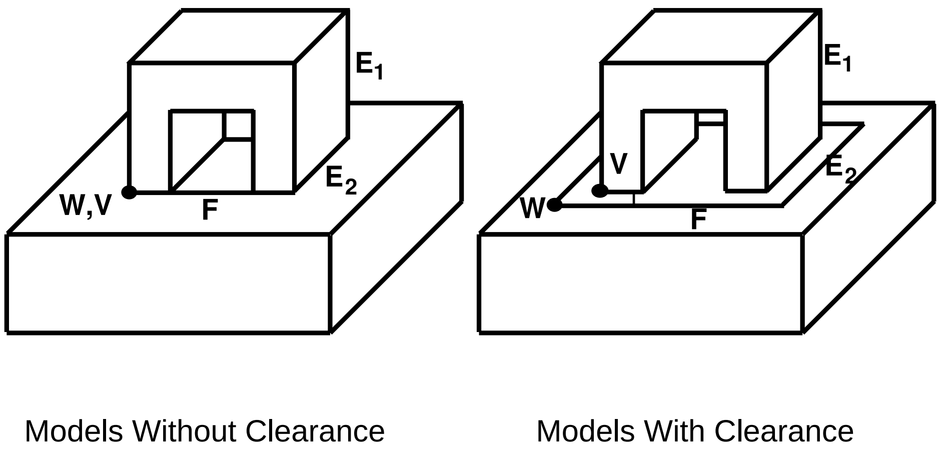


Figure 6: Enumeration of primitive contacts.

each PC is a single-point contact involving either a convex vertex and a face, or two convex edges. While this thesis focuses primarily on polyhedral assemblies, the formulae for C-surface functions and tangent hyperplanes in sections 2.3.2 and 2.3.4 apply to cylindrical surface features as well, using support functions for eight types of PCs in Appendix A. All of the contact states in contact space arise from combinations of one or more primitive contacts.

2.3.1 Enumeration of primitive contacts

As shown in Figure 6, the PCs that can occur in the presence of insertion clearances are just the PCs that occur simultaneously between the two parts prior to the introduction of clearances to their models. Moving part vertex V, for example, lies on the border of stationary part face F in the clearance-free assembly, indicating the presence of the corresponding C-surfaces in the contact space of the models that include clearances. The existence of a primitive contact between edges E_1 and E_2 is similarly inferred by noting

```
/* Enumerate Vertex-Plane Contacts */
for V in MovingPartVertices do
   for F in StationaryPartFaces do
      if V lies on F's surface
         then AddPC(V, F)
      endif
   endfor
endfor
/* Enumerate Plane-Vertex Contacts */
for F in MovingPartVertices do
   for V in StationaryPartFaces do
      if V lies on F's surface
         then AddPC(F, V)
      endif
   endfor
endfor
/* Enumerate Line-Line Contacts */
for E1 in MovingPartEdges do
   for E2 in StationaryPartEdges do
      if E1 intersects E2
         then AddPC(E1, E2)
      endif
   endfor
endfor
```

Figure 7: Algorithm for enumerating PCs.

that their line segments meet in the clearance-free models.

The process of enumerating primitive contacts is summarized in the algorithm of Figure 7, which operates on the clearance-free models, calling function AddPC(f_1, f_2) for each PC detected between a pair of features f_1, f_2.

2.3.2 Primitive contact functions

The 5-dimensional C-surface of primitive contact PC_i is characterized by a function $f_i : \Re^6 \rightarrow \Re$. A moving part pose $\mathbf{X}$ lies on the C-surface of PC_i if $f_i(\mathbf{X}) = 0$. Each C-surface function f_i is derived from the geometries of the assembly part features, using models that have been modified to incorporate designer-specified clearances. The geometry of a vertex-plane contact is illustrated in Figure 8. Vector $\mathbf{r}$ represents the position of vertex V in the moving part frame, and point $\mathbf{s}$ represents the closest point on stationary part face F to vertex V. Clearance dc in Figure 8 shows the separation between vertex V and face F when the moving part is positioned at the critical point configuration. The geometries of plane-vertex and line-line contacts are shown in Figures 9 and 10, respectively.

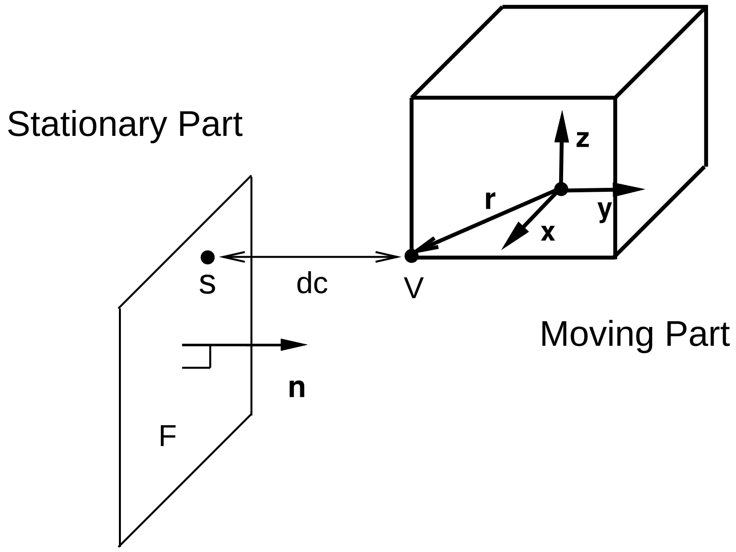


Figure 8: A vertex-plane contact.

The C-surface equation of a PC is derived by asserting that a point on the moving part feature and a point $\mathbf{s}$ on the stationary part feature form equal dot products with the PC's *contact normal* $\mathbf{n}$:

$$(\mathbf{dx} \ + RPY\,(\delta\mathbf{x})\,\mathbf{r}) \ \cdot \ \mathbf{n} \ = \ \mathbf{s} \ \cdot \ \mathbf{n} \tag{3}$$

For vertex-plane contacts, $\mathbf{n}$ is just the outward normal of the stationary face. For plane-vertex contacts, $\mathbf{n} = -RPY(\delta\mathbf{x})\ \mathbf{n}$, while for line-line contacts, $\mathbf{n}$

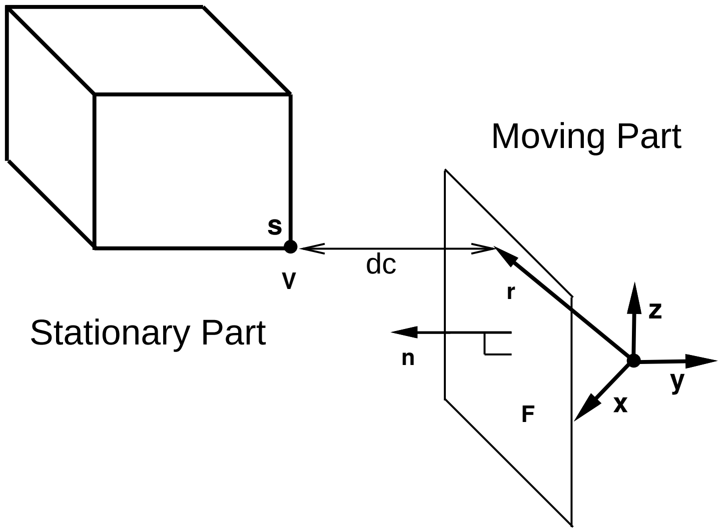


Figure 9: A plane-vertex contact.

is parallel to $\pm RPY(\delta\mathbf{x})\ \mathbf{v} \times \mathbf{w}$. Substituting Equation 2 for $RPY\ (\delta\mathbf{x})$, we obtain the C-surface function

$$\begin{aligned} f(\mathbf{X}) &= dx\, n_x \ + \ dy\, n_y \ + \ dz\, n_z \ + \\ &\quad I\, B \ + J\, D \ + I\, E \ - J\, F \ + H\, n_z \ - \\ &\quad \mathbf{s} \cdot \mathbf{n} \\ &= 0 \end{aligned} \tag{4}$$

where

$$\begin{aligned} I &= C_y r_x + S_x S_y r_y + C_x S_y r_z \\ B &= C_z n_x \\ J &= S_x r_z - C_x r_y \\ D &= S_z n_x \\ E &= S_z n_y \\ F &= C_z n_y \\ H &= S_x C_y r_y + C_x C_y r_z - S_y r_x \end{aligned} \tag{5}$$

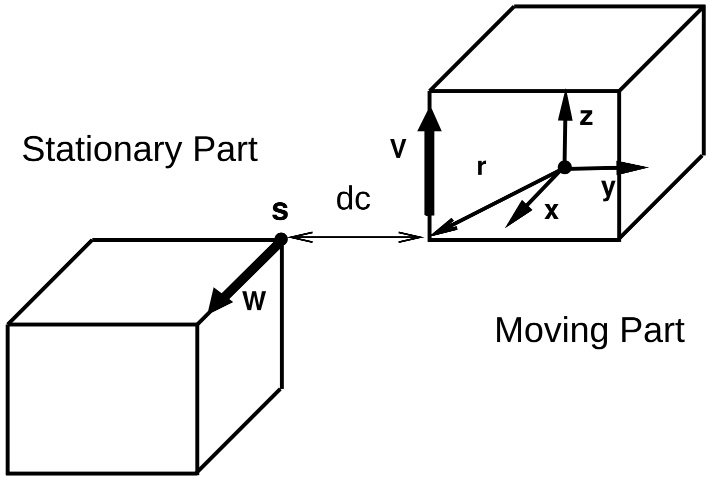


Figure 10: A line-line contact.

Equation 4 describes a 5–dimensional C-surface that locally divides $\Re^6$ into halfspaces corresponding to "free space" and the *forbidden zone*, the subset of $\Re^6$ where the assembly parts overlap. Adopting the convention that $\mathbf{n}$ points away from the stationary part feature, the poses $\mathbf{X}$ on the "free space" side of the C-surface correspond to $f_i(\mathbf{X}) > 0$. The poses for which $f_i(\mathbf{X}) < 0$ lie on the side of the forbidden zone.

2.3.3 Computing C-surface functions

The C-surface function formulated by Equations 4 and 5 is parameterized by contact normal $\mathbf{n}$, moving part contact point $\mathbf{r}$, which is defined in the moving part frame, and stationary part contact point $\mathbf{s}$, which is defined in the global frame[2]. Generally speaking, $\mathbf{n}$, $\mathbf{r}$, and $\mathbf{s}$ vary with the pose $\mathbf{X}$, so a more exact C-surface formula would express the elements of these vectors in terms of the pose variables, in place of the constants in Equations 4

[2]Note that $\mathbf{r}$ and $\mathbf{s}$ do not always represent the same point, since $f(\mathbf{X})$ is defined for poses $\mathbf{X}$ associated with contact, separation or overlap.

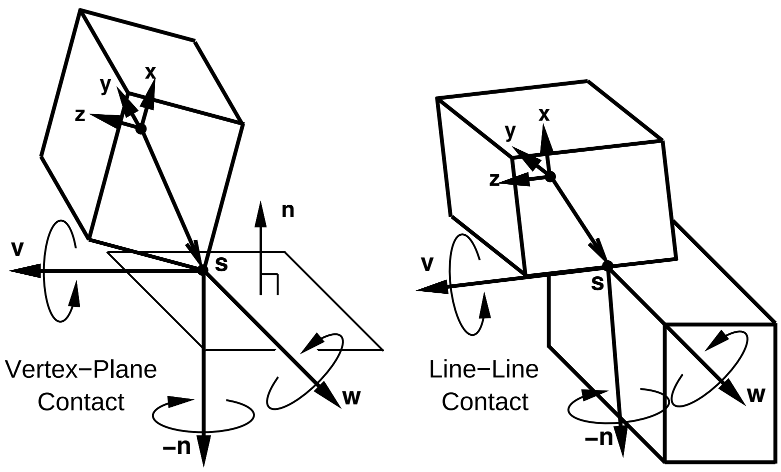


Figure 11: Primitive contact mobility.

and 5. In practice, however, the C-surface functions are evaluated efficiently by first computing the contact descriptors **n**, **r**, and **s**, and then treating those descriptors as constants in Equations 4 and 5. Appendix A contains procedures for computing the contact points and normals for eight types of PCs, including the three types of PCs associated with polyhedral assemblies.

When **n**, **r**, and **s** are held constant in Equations 4 and 5, the C-surface described by the formula is that of a vertex-plane contact. For other types of contacts, contact point **r** simulates a vertex by remaining fixed in the moving part frame; the contact normal **n** and scalar $\mathbf{s} \cdot \mathbf{n}$ in Equation 4 describe an imaginary plane. Our treatment of all classes of PCs as vertex-plane contacts is valid with regard to the evaluation of PC functions, since $f(\mathbf{X})$ is correct as long it is evaluated at the same pose for which which contact descriptors **n**, **r**, and **s** are obtained. The treatment of different classes of PCs as vertex-plane contacts is also valid for the purpose of deriving C-surface tangent hyperplanes, since the vector space of differential twists away from **X** is identical for all classes of PCs. Shown in Figure 11, for example, the mobility associated with a vertex-plane contact has 5 *dofs*: two translational *dofs* along planar tangents **v** and **w**, and three rotational *dofs* about contact point **s**. Similarly, a line-line contact has two translational *dofs* along edge vectors **v** and **w** and three rotational *dofs* about contact point **s**. As with the vertex-plane contact, **v** and **w** span a mobility plane perpendicular to the contact normal **n**. Provided with identical contact descriptors **n**, **r**, and **s**, the two contacts would be equivalent with respect to mobility, sharing the same C-surface tangent hyperplane in 6–space.

2.3.4 Primitive contact C-surface tangents

The outward normal to PC_i's C-surface tangent, and a formula for the PC's hyperplanar tangent surface, can be expressed in terms of the pose $\mathbf{X} \in \Re^6$. The C-surface tangent's outward normal $\mathbf{N} \in \Re^6$ is obtained by differentiating Equation 4 with respect to the 6 components of the pose and then normalizing the resulting vector:

$$\bigtriangledown f = \begin{pmatrix} n_x \\ n_y \\ n_z \\ AB + CD + AE - CF + Gn_z \\ HB + HE - In_z \\ JB - ID + JE + IF \end{pmatrix} \tag{6}$$

and

$$\mathbf{N} = \frac{\bigtriangledown f}{\mid \bigtriangledown f \mid} \tag{7}$$

where

$$\begin{aligned} A &= C_x S_y r_y - S_x S_y r_z \\ C &= C_x r_z + S_x r_y \\ G &= C_x C_y r_y - S_x C_y r_z \end{aligned}$$

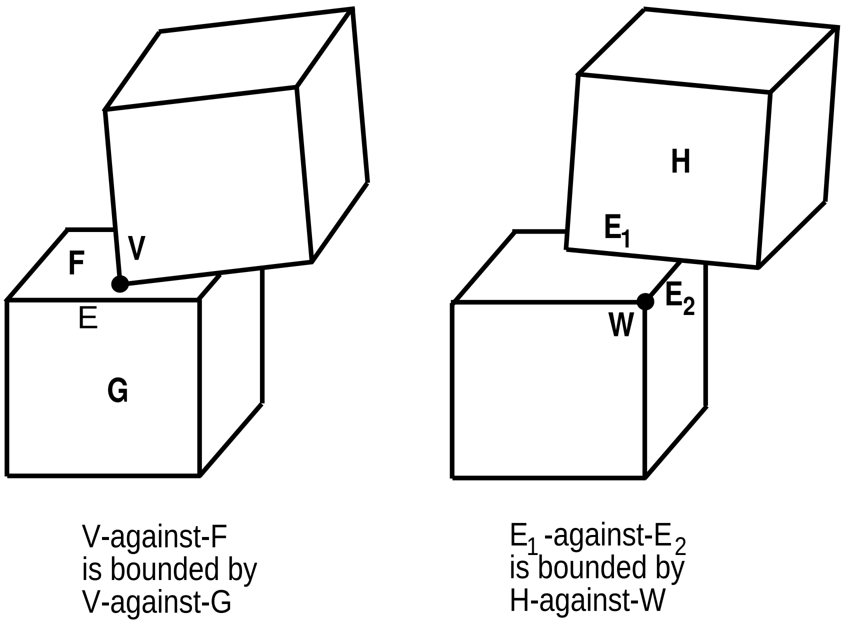


Figure 12: Primitive contact bounds.

$$(8)$$

Taking $f_i(\mathbf{X})$ as PC_i's approximate distance to $\mathbf{X}$, the surface tangent is approximated by the hyperplane

$$\mathbf{N}_i \cdot \mathbf{X} \; = \; D_i \tag{9}$$

where

$$D_i = \mathbf{X} \cdot \mathbf{N}_i - f_i(\mathbf{X}) \tag{10}$$

2.3.5 Primitive contact bounds

The 5-dimensional contact space facet associated with a PC may have borders corresponding to the finite extent of the assembly part surface features. In Figure 12, the contact involving vertex V and face F is bounded by the convex edge E shared by F and its adjacent face G. The poses $\mathbf{X} \in \Re^6$

associated with the PC formed by V and F are confined to the halfspace $g(\mathbf{X}) < 0$, where g is just the C-surface function for the contact between V and G. The V-against-G contact is called a *bounding contact* of the V-against-F contact. In general, a primitive contact PC_i has some n_i bounding contacts PC_{ij}. Assuming the part faces of all vertex-plane and plane-vertex contacts are convex, the half-spaces $g_j(\mathbf{X}) < 0$ of a PC's bounding contacts intersect to form a *quasi-convex* region in $\Re^6$. This will be discussed in more detail in section 2.3.6. If a part face is concave, it is divided into convex faces prior to PC enumeration.

```
/* Enumerate Vertex-Plane Contacts and Their Bounding Contacts */
for V in MovingPartVertices do
   for F in StationaryPartFaces do
      if V lies on F's surface
         then
            AddPC(V, F) → C
            for E in EdgesOfFace(F) do
               if edge E is convex and V lies on E
                  then
                     OtherFaceOfEdge(E, F) → G
                     AddBound(V, G, C)
               endif
            endfor
      endif
   endfor
endfor
```

Figure 13: Algorithm to enumerate vertex-plane PCs and bounding PCs.

The algorithm for enumerating PCs introduced in figure 7 is modified here to generate the contact bounds of each PC, as well. Figures 13, 14, and 15 show separate algorithms for enumerating vertex-plane, plane-vertex, and line-line PCs, respectively, along with their bounding contacts. For simplicity, only bounding contacts that form C-space concavities are enumerated, excluding those that form C-space convexities. The justification for this pol-

```
/* Enumerate Plane-Vertex Contacts and Their Bounding Contacts */
for F in MovingPartVertices do
   for V in StationaryPartFaces do
      if V lies on F's surface
         then
            AddPC(F, V) → C
            for E in EdgesOfFace(F) do
               if E is convex and V lies on E
                  then
                     OtherFaceOfEdge(E, F) → G
                     AddBound(G, V, C)
               endif
            endfor
      endif
   endfor
endfor
```

Figure 14: Algorithm to enumerate plane-vertex PCs and bounding PCs.

```
/* Enumerate Line-Line Contacts and Their Bounding Contacts */
for E1 in MovingPartEdges do
   for E2 in StationaryPartEdges do
      if E1 intersects E2
         then
            AddPC(E1, E2) → C
            for V in VerticesOfEdge(E1) do
               if V is convex and V lies on edge E2
                  then
                     Let F be the stationary part face adjacent to E2
                     whose normal is most nearly perpendicular to E1
                     AddBound(V, F, C)
               endif
            endfor
            for V in VerticesOfEdge(E2) do
               if V is convex and V lies on edge E1
                  then
                     Let F be the stationary part face adjacent to E1
                     whose normal is most nearly perpendicular to E2
                     AddBound(V, F, C)
               endif
            endfor
      endif
   endfor
endfor
```

Figure 15: Algorithm to enumerate line-line PCs and bounding PCs.

icy is that bounding contacts forming C-space convexities would implicitly represent the forbidden zone, which will be represented separately in chapter 2.3.6. In Figure 16, for example, the V-against-F PC forms a C-space convexity where it meets V-against-G, so the former is not considered a bounding

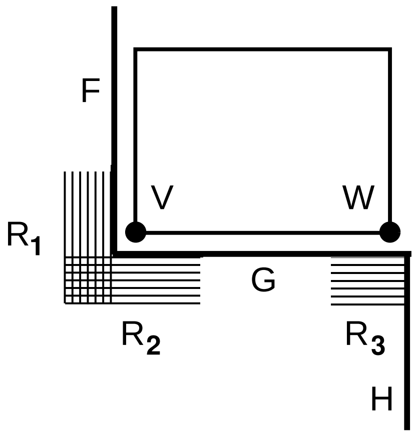


Figure 16: Regions of overlap for vertex-plane contacts.

contact of the latter. However, the W-against-H PC forms a C-space concavity where it meets W-against-G, so the former is declared a bounding contact of the latter. This concavity requirement implies that the bounding edge associated with a vertex-plane or plane-vertex contact's bounding PC be convex, and that the vertex associated with a line-line contact's bounding PC must be convex. The algorithms in figures 13 through 15 make use of a function AddBound(f_1, f_2, C), which states that primitive contact C is bounded by the PC formed by features f_1 and f_2. Function OtherFaceOfEdge(E, F) returns the face adjacent to E that is not F, and function VerticesOfEdge(E) returns a list of vertices adjacent to E.

2.3.6 Primitive contact point sets

Each PC's facet in contact space, as well as the forbidden zone of overlapping configurations behind the facet, are represented by point sets expressed in terms of the C-surface functions $f_i(\mathbf{X})$. The *forbidden zone* F_i of PC_i is the region of $\Re^6$ behind PC_i's C-surface and its bounding contacts:

$$F_i \;=\; \{\mathbf{X} \in \Re^6 \mid f_i(\mathbf{X}) \;<\; 0,\; f_{ij}(\mathbf{X}) \;<\; 0, j = 1 \ldots n_i\} \qquad (11)$$

The entire forbidden zone $\mathcal{F}$ surrounding a critical point's local C-space is formed by the union of all the PCs' forbidden zones:

$$\mathcal{F} \;=\; \cup_i F_i \qquad (12)$$

In Figure 16, for example, a cross-section of an assembly is shown in which primitive contacts can form between vertex V and face F, vertex V and face G, and vertex W and face G. Lying beneath face F, 3-space region R_1 corresponds to the contact space forbidden zone behind the V-against-F C-surface, while region R_2 corresponds to the forbidden zone behind the V-against-G C-surface. Note that R_1 and R_2 overlap, as do their corresponding forbidden zones in $\Re^6$, contributing to the point set union (12). As discussed in chapter 2.3.5, the V-against-F PC is not a bounding contact of the V-against-G PC, and vice-versa, since the two PCs form a C-space convexity.

The point set S_i in $\Re^6$ associated with a primitive contact PC_i consists of the portion of the contact's C-surface that lies behind its border PCs' C-surfaces but not in the forbidden zone:

$$S_i \;=\; \{\mathbf{X} \in \Re^6 \mid f_i(\mathbf{X}) \;=\; 0,\; f_{ij}(\mathbf{X}) \;< 0, j = 1 \ldots n_i, \mathbf{X} \notin \mathcal{F}\} \qquad (13)$$

Since the forbidden zone is generally concave, a PC_i's point set S_i is usually concave as well and cannot be represented by a single intersection of halfspaces. For the purpose of efficiently measuring the distance between a trajectory segment and a PC's point set, it is therefore preferable to represent PCs as convex sets, even if they overlap the forbidden zone. When calculating distances between trajectory segments and "obstacle" PCs in chapter 3.3.2, a PC's closest point may then lie in the forbidden zone of a closer, intervening obstacle PC, but the closer PC's presence will overshadow the importance of the former obstacle PC.

Ignoring the presence of the forbidden zone, PC_i and its n_i bounding contacts PC_{ij} define a *quasi-convex* point set, which may have slight concavities due to the small curvatures of its defining C-surfaces, but is devoid of gross concavities arising from polyhedral part convexities:

$$S_i' = \{\mathbf{X} \in \Re^6 \mid f_i(\mathbf{X}) = 0,\ f_{ij}(\mathbf{X}) < 0, j = 1 \ldots n_i\} \tag{14}$$

The linearization of S_i' about pose $\mathbf{X}$ yields a convex polytope in $\Re^6$. If $n_i = 0$, the polytope is just the infinite C-surface tangent of PC_i, constituting a single facet. If $n_i = 1$, then the C-surface tangent of PC_i, and the 4–dimensional affine space formed where it intersects its single bounding contact, form a 2–faceted polytope. The case of $n_i = 2$ yields a polytope of 3 or 4 facets, consisting of PC_i's C-surface tangent, the two 4–dimensional affine spaces formed where it intersects the two bounding contacts, and possibly a 3–dimensional affine space, formed if all three C-surface tangents intersect. n_i is never greater than 2, which is the number of bounding contacts associated with vertex-plane and plane-vertex PCs involving vertex-vertex contacts. As described in chapter 2.5.2, affine space representations of S_i''s facets are readily computed from the hyperplanar tangents of PC_i and its bounding contacts PC_{ij}.

2.3.7 Point set predicates

For notational purposes, it is useful to define predicates for the point sets described above. The forbidden zone point set of Equation (12) gives rise to the following predicate, which is easily implemented using the C-surface functions $f_i(\mathbf{X})$ formulated in Equation 4:

$$P_{\mathcal{F}}(\mathbf{X}) \iff \mathbf{X} \in \mathcal{F} \iff \exists\, i \;\; \mathbf{X} \in \mathbf{F}_i \tag{15}$$

The PC point set defined in Equation 13, and the quasi-convex point set of Equation 14 are associated with the following predicates:

$$P_{S_i}(\mathbf{X}) \iff \mathbf{X} \in S_i \tag{16}$$

$$P_{S_i'}(\mathbf{X}) \iff \mathbf{X} \in S_i' \tag{17}$$

2.4 Contact space vertices

The first step in constructing the local contact space surrounding the critical point is the enumeration of its vertices, i.e, the 0-dimensional facets of contact space. The vertices are obtained by computing all of the legal moving part configurations determined by 6-tuples of primitive contacts. This combinatorial method was employed by Koutsou [75], who combined the spatial relationships of PC 6–tuples symbolically to yield each vertex configuration. In the current technique, however, the vertex configurations are calculated numerically, using the multivariable Newton method.

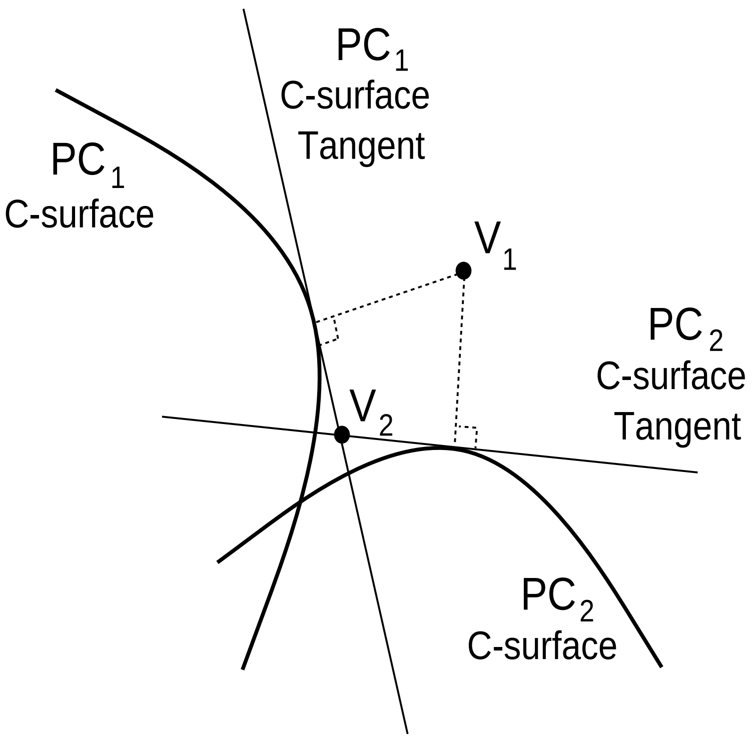


Figure 17: Vertex calculation through iterative refinement.

2.4.1 Calculating contact space vertex configurations

The point of intersection $\mathbf{V} \in \Re^6$ associated with the C-surfaces of 6 primitive contacts $PC_1, \ldots, PC_6$ is just the common root of the C-surface equations $f_i(\mathbf{V}) = 0, \ i = 1..6$, as formulated in Equation 4. Given an initial estimate of $\mathbf{V}$, the multivariable Newton method iteratively refines the estimate, converging rapidly to its true position. For low-clearance assemblies, the vertices of contact space are never more than a perturbation away from the origin, so the origin itself provides a suitable first approximation.

As Figure 17 shows in 2 dimensions, the initial estimate $\mathbf{V}_1$ of the vertex point where two C-surfaces intersect gives rise to two C-surface tangents, which intersect at an improved vertex estimate $\mathbf{V}_2$. An initial estimate (e.g., the origin) of the vertex is thus iteratively refined until its measurable modification ceases. For 3-dimensional assemblies, the C-surface tangent equation for each of six PCs is obtained by parameterizing Equations 6...10 with the successively refined poses $\mathbf{V}_i$. Each refined vertex approximation $\mathbf{V}_i$ is then obtained by simultaneous solution of the 6 resulting equations.

2.4.2 Identifying nonspurious vertices

If the C-surfaces of 6 primitive contacts $PC_1, \ldots, PC_6$ intersect at a single point $\mathbf{V}$ in $\Re^6$, then $\mathbf{V}$ represents a contact space vertex only if it lies in the point set S_i of each of its associated PC_is. This condition could be verified by testing whether $\mathbf{V}$ satisfies predicate $P_{S_i}(\mathbf{V})$ (see chapter 2.3.7), for each PC_i, which excludes the forbidden zone. Computationally, however, it is more efficient to determine whether $\mathbf{V}$ satisfies the quasi-convex point set predicate $P_{S_i^!}(\mathbf{V})$ for each PC_i, which does not exclude the forbidden zone, and then check that $\mathbf{V}$ does not satisfy the forbidden zone predicate $P_{\mathcal{F}}(\mathbf{V})$.

Figure 18 shows the analogy in 2-dimensional contact space. Each candidate vertex is formed by intersecting the C-surfaces of a pair of PCs. V_1 is found to be a legitimate vertex, since it lies within the point sets of its constituent contacts PC_2 and PC_3. V_2, on the other hand, lies in the forbidden zone, and V_3 lies in the point set of only one of its PCs, so V_2 and V_3 are discarded as spurious vertices.

2.4.3 Complexity of vertex enumeration

As demonstrated in chapter 5.2.2,the generation of local contact space vertices described above requires $\mathcal{O}(n^6)$ computation time in the number of

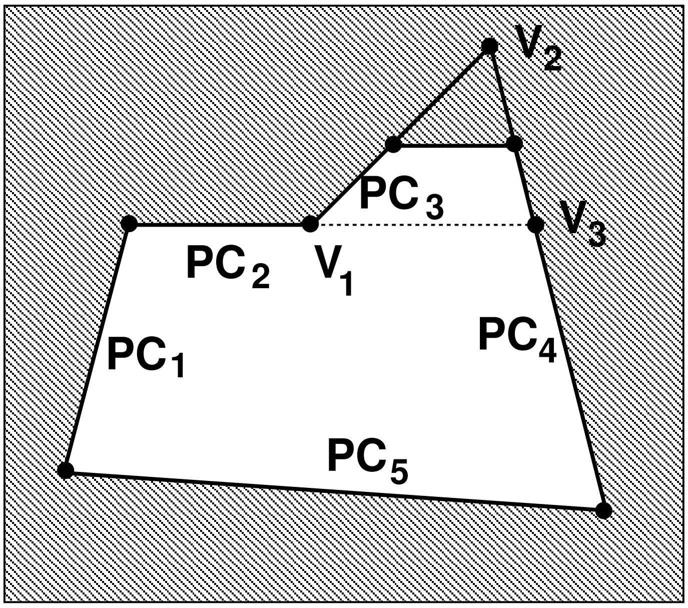


Figure 18: Identifying nonspurious contact space vertices.

PCs. For complex assemblies involving dozens of PCs, the enumeration of contact space vertices can become impractical, due to the enormous computation time required. To mitigate the time requirements associated with large numbers of PCs, Koutsou [75] has proposed restricting the dimensions of contact space, thereby limiting the number of PCs and the computation time required to generate the contact space vertices.

We have indeed found it convenient, and often necessary, to impose *artificial C-space limits* upon the rotational component of contact space. The C-space limits we employ are rotational constraints which enforce strictly positive or negative tilts about various rotational axes. The relevant axes are just the moving part edges associated with, or adjacent to, the PCs in question. In Figure 19, for instance, the vertex-plane PC between vertex V and face F cannot be established if the moving part is rotated positively

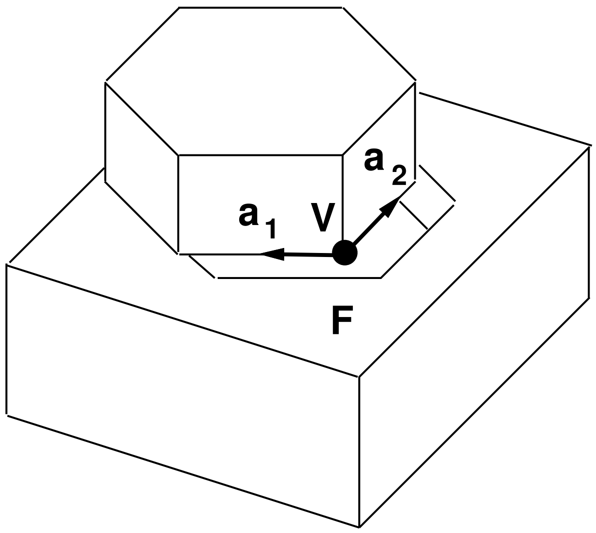


Figure 19: A PC's axes of rotational constraint.

about axis $\mathbf{a}_1$, or negatively about axis $\mathbf{a}_2$, of its neighboring edges. This preclusion of the vertex-plane PC arises from the interfering line-line PC(s) that would form instead. Hence, by artificially limiting the rotational component of C-space to strictly positive or strictly negative rotations about various edge axes, a number of PCs may be eliminated from consideration during the construction of contact space.

To enumerate the vertices of the square peg-in-hole assembly's contact space, strict rotational inequality constraints are imposed about the x, y, and z axes. In practice, these constraints reduce the number of PCs from 32 to 9, with a corresponding reduction in the number of PC 6–tuples from 906192 to 84. For a symmetric assembly such as the square peg-in-hole assembly, this restriction of the moving part's orientation to a single octant of $\delta x \times \delta y \times \delta z$–space eliminates a large number of mirror-image "duplicates" from the resulting set of contact states. On the down side, the use of strict inequalities such as $\delta x < 0$, effectively eliminates certain contact states in which the peg is aligned with the rim of the hole, e.g., with 0-rotation about the x, y, or z-axis. The use of such artificial C-space limits therefore renders several contact states unavailable for fine-motion planning.

2.4.4 Vertex enumeration with virtual PCs

The elimination of contact states that stems from the use of artificial C-space limits can produce a contact space devoid of vertices. Since the contact space analysis technique presented in the current section relies on the generation of contact space vertices, from which all higher-dimensional contact states are enumerated, the abbreviated set of PCs must be augmented manually with a few "virtual PCs". When contructing a local contact space for the square peg-in-hole assembly, for example, three virtual PCs are temporarily added to the 9 "filtered" PCs to represent imaginary PCs with C-surface equations $\delta x = -\epsilon$, $\delta y = -\epsilon$, and $\delta z = -\epsilon$, where $0 < \epsilon \ll 1$. Contact states involving virtual PCs are then sifted out of the resulting adjacency graph. For the square peg-in-hole assembly with $\frac{1}{32}$ inch clearances, the 9 PCs from the restricted contact space, augmented with the 3 virtual PCs, combine to form 924 6–tuples of PCs. Among the 924 candidate vertices, 22 are nonspurious.

2.5 Contact space facets

The local contact space adjacent to a critical point configuration is comprised of 0- to 5-dimensional facets, each corresponding to one or more PCs which can form simultaneously. The facet associated with a set of primitive contacts $PC_1, \ldots, PC_n$ is denoted by

$$facet(PC_1, \ldots, PC_n)$$

Assuming the extent of a local contact space is small, as dictated by artificial limits imposed along its axes, each of its facets will be fairly "flat" and can be approximated by an affine space (see chapter 2.5.2). Moreover, local contact space is approximately hyperpolyhedral, and one may conceptualize the trajectory planning domain as the higher-dimensional analogy to a polyhedral environment in which piecewise-linear curves are to be synthesized.

2.5.1 PCs of a contact space facet

Every contact state in a local contact space arises from the PCs that can develop between features of the moving part and stationary part. The PCs form various combinations of point, line, and surface contacts. Barring the presence of line and surface contacts and some rare, singular configurations, the m-dimensional facets of contact space arise from $(6 - m)$-tuples of PCs. Contact states involving line and surface contacts, however, may include additional, "redundant" PCs, whose presence does not reduce the dimensionality of the corresponding contact space facet. Moreover, a single facet may be comprised of distinct regions, each corresponding to a different set of PCs. The planar contact in Figure 20, for example, has three distinct contact states, S_1, S_2, and S_3, each involving at least three PCs, restricting the pose to the same 3-dimensional facet in contact space. The three contact states correspond to distinct regions of the facet, whose surface can be represented qualitatively by any three of the PCs shown.

Since the set of PCs associated with a contact space facet ϕ may differ

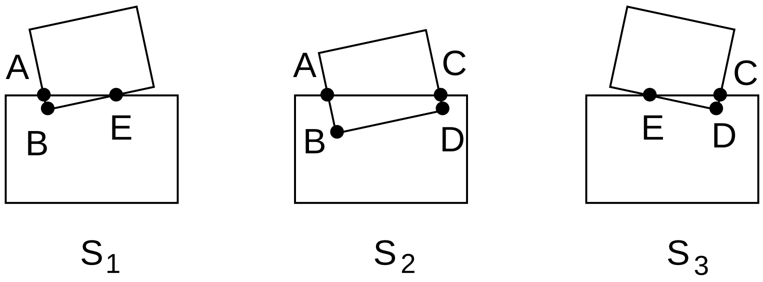


Figure 20: Multiple C-states of a single C-space facet.

from the various sets of PCs comprising its contact states, it is useful to denote by $facetPCs(\phi)$ the set containing all of the PCs of the facet's contact states. The set $supportPCs(\phi)$, on the other hand, denotes a nonunique, "minimal" set of PCs whose 5-dimensional C-surfaces intersect to form facet ϕ. That is, the C-surface normals associated with $supportPCs(\phi)$ are independent in $\Re^6$ throughout ϕ. The set of PCs associated with a contact state S is denoted by $cstatePCs(S)$. With regard to the planar contact of Figure 20, for instance,

$$\begin{array}{rcl} facetPCs(\phi_0) & = & \{A, B, C, D, E\} \\ supportPCs(\phi_0) & = & \{A, B, D\} \\ cstatePCs(S_1) & = & \{A, B, E\} \\ cstatePCs(S_2) & = & \{A, B, C, D\} \\ cstatePCs(S_2) & = & \{E, C, D\} \end{array} \tag{18}$$

where ϕ_0 denotes the facet in contact space.

2.5.2 Linearization of obstacle PC facets

A contact space facet ϕ defined by $supportPCs(\phi) = \{PC_1 \ldots PC_n\}$ can be approximated as a $(6-n)$-dimensional affine space. This linear approximation facilitates the measurement of the distance separating an "obstacle" PC and a linear trajectory segment in chapter 4.1, Given a pose

$\mathbf{X} \in \Re^6$ about which to linearize contact space, the facet is approximated by an $(6-n)$-dimensional affine space, denoted by $ASpace_{\mathbf{X}}(PC_1 \ldots PC_n)$. This affine space, whose descriptors will include a point $\mathbf{P}$ and a vector space basis $\mathbf{V}_1 \ldots \mathbf{V}_{6-n}$ in $\Re^6$, is computed as follows.

For $\mathbf{P}$, choose the pose corresponding to any contact space vertex adjacent to S (i.e., a 0-dimensional contact state whose 6 associated PCs contain $PC_1 \ldots PC_n$). Let $\mathbf{N}_1 \ldots \mathbf{N}_n$ be C-surface tangent normals for $PC_1 \ldots PC_n$, as computed in (6)...(8). The vector space basis $\mathbf{V}_1 \ldots \mathbf{V}_{6-n} \perp \mathbf{N}_1 \ldots \mathbf{N}_n$ is readily computed by applying the Gram-Schmidt procedure to the ordered spanning set $\mathbf{N}_1 \ldots \mathbf{N}_n, \mathbf{U}_1 \ldots \mathbf{U}_{6-n}$, where $\mathbf{U}_1 \ldots \mathbf{U}_{6-n}$ are $6-n$ standard axes in $\Re^6$, each independent of $\mathbf{N}_1 \ldots \mathbf{N}_n$. The Gram-Schmidt procedure effectively transforms the standard axes to produce the required basis.

2.6 Adjacency graph of contact states

An adjacency graph of contact states is constructed by enumerating subtuples of the contact space vertices' 6-tuples of PCs, giving rise to the multidimensional facets and contact states of contact space. Arcs are placed in the graph to represent the spatial adjacency of contact state point sets in $\Re^6$. As shown in Figure 21, the contact state adjacency graph provides a search structure for exploring the multidimensional contact states that constitute the local contact space. The following sections describe procedures for constructing contact state adjacency graphs.

2.6.1 Adjacency graphs: one C-state per facet

Once the vertices of a local contact space have been identified and located, the higher-dimensional contact states can be enumerated and placed in an

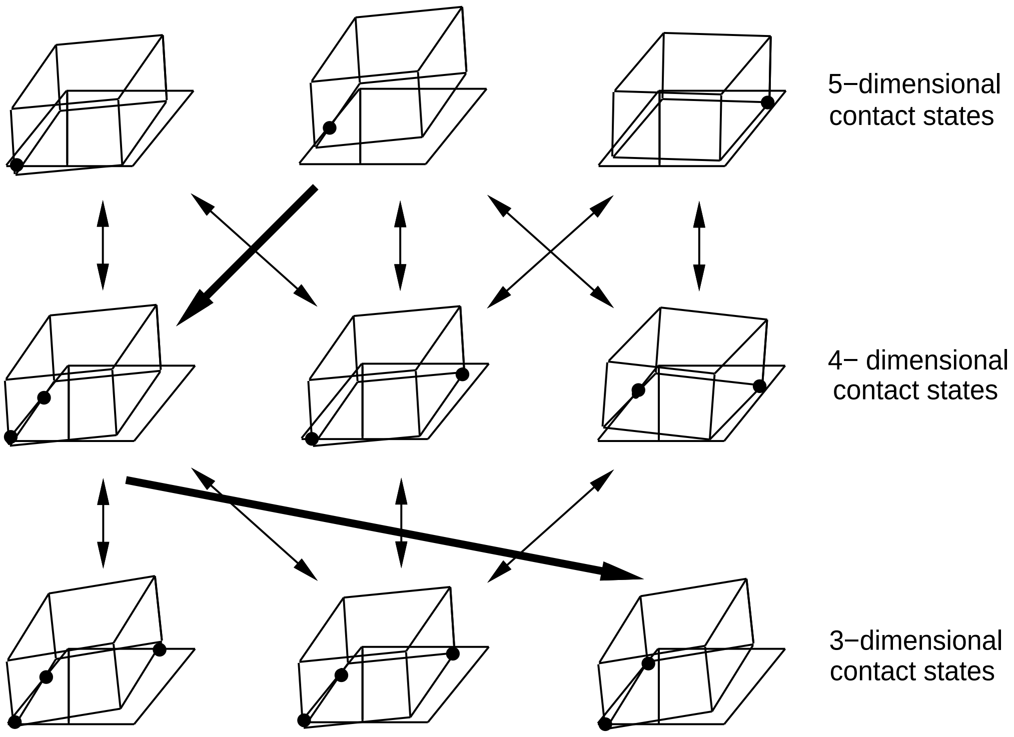


Figure 21: Searching an adjacency graph of contact states.

adjacency graph. For some assemblies, such as the rectangular peg-in-hole assembly, the imposition of artificial C-space limits (see chapter 2.4.3) leads to a relatively simple adjacency graph construction. Through the use of appropriate rotational constraints, such as $\delta x < 0, \delta y < 0, \delta z < 0$ for the square peg-in-hole assembly, the line and surface contacts of contact space may be eliminated. The remaining contact states are comprised of "pure" point contact combinations. In such a contact space, the formation or separation of each PC is generally associated with a change in contact space facet dimensionality, so there is normally a one-to-one correspondence between contact space facets and contact states. This leads to a relatively simple procedure for constructing a contact state adjancency graph.

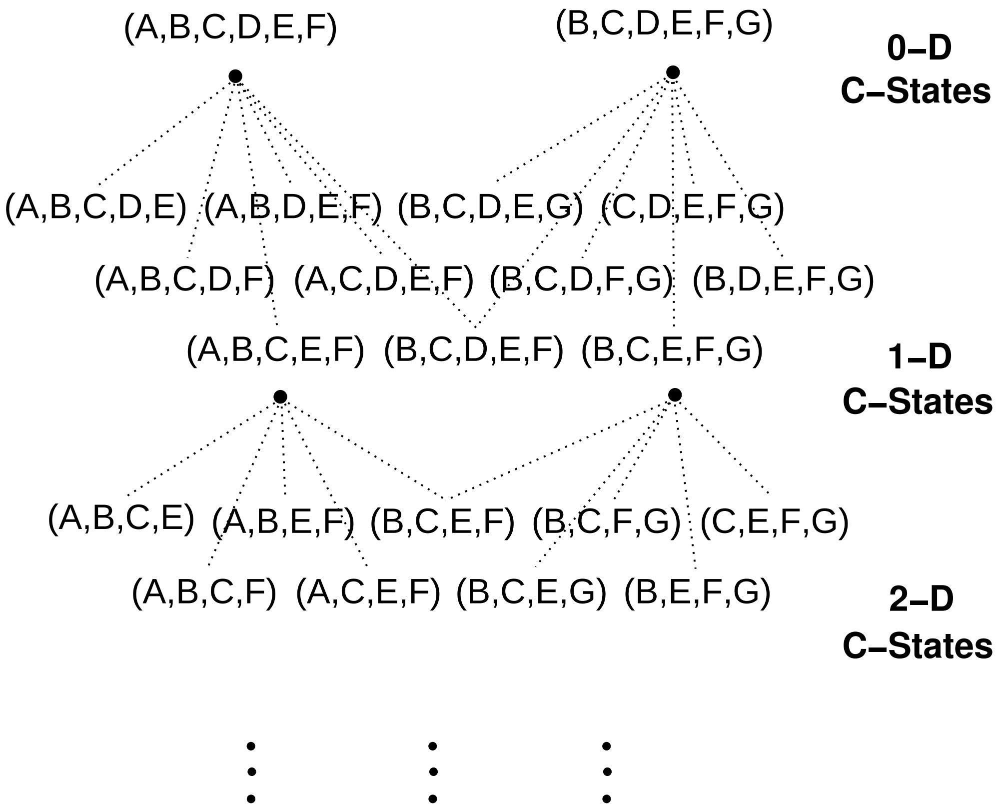


Figure 22: Generating a contact state adjacency graph.

In the absence of line or surface contacts, an adjacency graph of contact states is constructed as follows. As shown in in Figure 22, the 1-dimensional edge states of contact space are generated by enumerating the 5-tuples of PCs contained in the vertices' 6-tuples. The 4-tuples of the edges' PCs give rise to the 2-dimensional contact states, and so on, until the 5-dimensional primitive contact states have been generated. Each new and distinct contact state thus enumerated is placed in an adjacency graph together with arcs denoting whether a contact state's set of PCs is a subset of an adjacent "child" state's set of PCs.

2.6.2 Adjacency graphs: multiple C-states per facet

If line or surface contacts are permitted to form between the surface features of the assembly parts, the construction of the contact state adjacency graph is more complex. As discussed in section 2.5.1, the presence of line and surface contacts gives rise to contact space facets which correspond to more than one contact state (see Figure 20). Consequently, the enumeration of facets and contact states must be performed separately. Additionally, the dimensionality of a contact state is no longer determined by the number of PCs that define the state, since the C-surface normals of the PCs may not be independent. As mentioned in chapter 2.5.1, an i-dimensional facet of contact space is represented by a $(6 - i)$-element list, $support(\phi)$ of "independent" PCs, and an m-element list $facetPCs(\phi)$ of some $m \geq 6 - i$ PCs. The latter collectively represents all of the contact states associated with the facet.

The algorithm in Figure 23 constructs an adjacency graph of contact states that allows more than one contact state per facet. A second algorithm, shown in figure 24, groups together contact states according to the facets they occupy. These algorithms construct adjacency graphs such as the one in Figure 25, which shows the adjacency graph for an imaginary assembly, consisting of four facets, with each facet comprised of two, three or four contact states. Figure 23's algorithm requires input in the form of a list $facets(0)$, which contains the contact space vertices, enumerated as in chapter 2.4. The presence of line and surface contacts can result in coincident vertices in $\Re^6$, which must be "merged" prior to the construction of the adjacency graph. Following the fusion of redundant vertices, every vertex ϕ is assumed to have a defining 6-tuple of PCs, represented by list $support(\phi)$, and a list $facetPCs(\phi)$ of 6 or more associated PCs. The presence of more

```
/* Procedure for Generating a Contact State Adjacency Graph */
for i = 0 to 4 do
/* Enumerate the (i+1)-dim facets of contact space */
   Initialize empty list facets(i+1)
   for each i-dim facet φ in facets(i) do
      Initialize empty list children(φ)
      for each (5-i)-tuple PC_1 ... PC_{5-i} in facetPCs(φ) do
         if the C-surface normals of PC_1 ... PC_{5-i} are independent in ℜ^6
            then
               Create facet φ'; [PC_1 ... PC_{5-i}] → facetPCs(φ'), support(φ')
               if EqualFacet(φ', φ'') for some facet φ'' in facets(i+1)
                  then
                     Merge(facetPCs(φ'), facetPCs(φ'')) → facetPCs(φ')
                     Add φ to parentss(φ'');   Add φ'' to children(φ)
                  else
                     Add φ' to facets(i+1)
                     Initialize list parentss(φ') to [ φ ];   Add φ' to children(φ)
               endif
         endif
      endfor
   endfor
   /* Enumerate the (i+1)-dim contact states of contact space */
   for each facet φ in facets(i+1) do
      Initialize empty list cstates(φ)
      for each m-tuple PC_1 ... PC_m (m ≥ 5-i)  in facetPCs(φ) do
         if Feasible1(φ, PC_1 ... PC_m) and
            VSpace(PC_1 ... PC_m) = VSpace(support(φ))
            then
               Create C-state S; [PC_1 ... PC_m] → cstatePCs(S)
               Add S to cstates(φ)
               for each facet φ' in parents(φ) do
                  for each contact state S' in cstates(φ') do
                     if cstatesPCs(S) ⊂ cstatePCs(S')
                        then
                           Add S' to parents(S);   Add S to children(S')
                     endif
                  endfor
               endfor
         endif
      endfor
   endfor
endfor
```

Figure 23: Algorithm to generate a contact state adjacency graph.

```
/* Set up adjacency links between C-states within the same facet */
for i = 0 to 4 do
/* Process all the i-dim facets */
   for each i-dim facet φ in facets(i) do
      for each contact states S in cstates(φ) do
         Initialize empty list sibs(S)
      for each pair of contact states S1, S2 in cstates(φ) do
         /* See if a legal pose exists which lies in both contact states */
         if Feasible2(φ, S1, S2)
            then Add S2 to sibs(S1);  Add S1 to sibs(S2)
         endif
      endfor
   endfor
endfor
```

Figure 24: Algorithm for grouping C-states according to their facets.

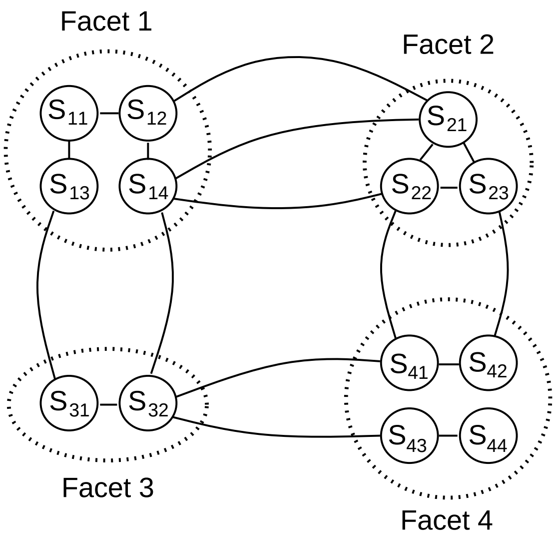


Figure 25: Adjacency graph of contact states and facets.

than 6 PCs in $facetPCs(\phi)$ results from combining the 6-tuples of coincident vertices.

Beginning with the 0-dimensional facets, the contact space facets of successively higher dimensions are enumerated. As in the process described in chapter 2.6.1, the contact space facets are generated by examining subtuples of the PCs associated with the existing facets. In the presence of line and surface contacts, however, distinct PC subtuples may define the same contact space facets, so procedure EqualFacet, which is described in Appendix B, must check for facet equivalence. Facets defined by different PC tuples are added to the graph only if they are distinct, and PCs associated with identical facets are merged. Adjacency relationships for each facet ϕ are represented as "parent" and "child" lists $parents(\phi)$ and $children(\phi)$, containing facets which correspond to removing or restoring a single degree of freedom.

After the facets of dimension i have been enumerated and stored in list $facets(i)$, the multiple contact states which make up each facet ϕ are generated and placed in list $cstates(\phi)$. Procedure $Feasible1$ (see Appendix C) is used to determine which subtuples of ϕ's associated PCs can coexist on their own, without establishing additional contacts that cannot be avoided. The adjacency relationships of a contact state S which correspond to the removal or restoration of a single degree of freedom are represented by lists $parents(S)$ and $children(S)$. In the algorithm of Figure 24, adjacency relationships are established which link each contact state S to its adjacent contact states within the same facet. These adjacent states are are stored in list $sibs(S)$. Procedure $Feasible2$ (see Appendix C) judges this accessibility by deciding if two contact states S_1 and S_2 ever share the same pose in $\Re^6$.

2.7 Partitioning C-space into quasi-convex regions

The local C-space surrounding a critical point configuration is typically concave, due to the convexities of the assembly parts, as well as the small curvatures in the 5-dimensional C-surfaces that border the local C-space. The following procedure partitions C-space into quasi-convex regions that are free of gross concavities arising from polyhedral part convexities.

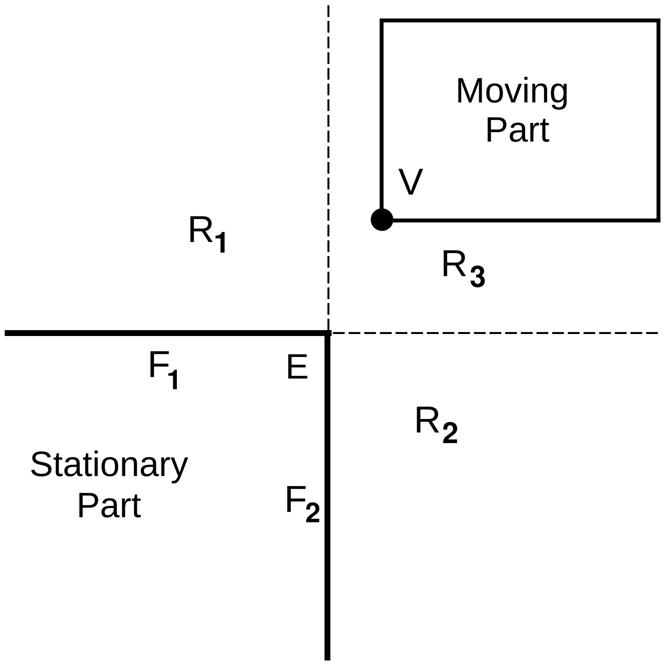


Figure 26: Partitioning free space at a vertex-edge contact.

For polyhedral assemblies, the gross concavities of C-space correspond to contacts that can form between convex vertices and convex edges. Each convex vertex-edge (or edge-vertex) contact restricts the vertex in question to the concave region of $\Re^3$ above one or both of the faces adjacent to the edge. As shown in Figure 26, for example, vertex V occupies the concave region composed of three convex regions R_1, R_2, and R_3, respectively situated above face F_1, face F_2, and faces F_1 and F_2. Confining V to one of these convex regions in $\Re^3$ effectively confines the assembly pose to a quasi-convex region in $\Re^6$. The C-space regions corresponding to R_1, R_2, and R_3 are readily expressed as point sets

$$
\begin{array}{lcl}
C_1 & = & \{\mathbf{X} \in \Re^6 \mid f_1(\mathbf{X}) \geq 0,\ f_2(\mathbf{X}) < 0\} \\
C_2 & = & \{\mathbf{X} \in \Re^6 \mid f_2(\mathbf{X}) \geq 0,\ f_1(\mathbf{X}) < 0\} \\
C_3 & = & \{\mathbf{X} \in \Re^6 \mid f_1(\mathbf{X}) \geq 0,\ f_2(\mathbf{X}) \geq 0\}
\end{array}
\tag{19}
$$

where $f_1(\mathbf{X})$ and $f_2(\mathbf{X})$ are the C-surface functions for V-against-F_1 and V-against-F_2, resp.

A local C-space is decomposed into quasi-convex regions by restricting the vertex involved in every vertex-edge or edge-vertex contact to one of the three real-space regions R_1, R_2, or R_3 associated with the contact. If the local contact space contains some n vertex-edge or edge-vertex contacts, every quasi-convex point set in the decomposed C-space is represented by an n-tuple

$$
(i_1, \ldots, i_n) \quad i_j \in 1 \ldots 3,\ j \in 1 \ldots n \tag{20}
$$

where i_j indicates which convex region of $\Re^3$ must contain the vertex associated with the jth convex vertex-edge or edge-vertex contact. While there are some 3^n such tuples, the use of artificial C-space limits (see chapter 2.4.3) can greatly reduce the number of quasi-convex regions generated by this partitioning technique.

CHAPTER 3

FINE-MOTION PLANNING

The method of fine-motion planning presented in this thesis assumes the existence of a nominal trajectory, provided *a priori* by a high-level assembly planner. The role of the fine-motion planner is to perturb the nominal trajectory in such a manner as to restrict the path of the moving part to a deterministic sequence of contact states in the presence of positioning error. Since the topology of contact space does not vary between critical points, fine-motion planning is restricted to the local contact spaces surrounding the critical points in the original nominal trajectory. To travel from one critical point to the next, the assembly can normally be confined to a lower-dimensional contact state shared by the two critical points' local contact spaces. Each local contact space is represented as an adjacency graph of contact states, whose construction is described in chapter 2.6. In the author's fine-motion planning technique, a heuristic search of each local contact space yields the most traversable sequence of contact states, within which a piecewise-linear command trajectory is synthesized. The entire nominal trajectory supplied by the high-level assembly planner is thus replaced by a perturbed nominal trajectory, whose synthesis is performed within a few local contact spaces.

Fine-motion planning in a local contact space takes place at two hierarchical levels: the higher-level selection of a reliably-traversable sequence of contact states, and the lower-level formation of a piecewise-linear command trajectory through the chosen states. At the higher level, a heuristic search of the contact state adjacency graph yields a traversable sequence of contact states. At the lower level, a perturbed nominal trajectory is synthesized by optimizing the trajectory's clearance with respect to extraneous contact states. The latter process of trajectory refinement is explained in chapter 4. After the perturbed nominal trajectory is synthesized in local, 6-dimensional contact space, its constituent trajectory segments are finally expressed as a sequence of hybrid control strategies.

As explained in chapter 3.3.2, trajectory planning may be performed with or without a detailed uncertainty analysis. In the more rudimentary approach, uncertainty is simply modelled in terms of a constant ratio between translational and rotational uncertainty limits. In the more sophisticated treatment of uncertainty, a more complex model of uncertainty must be provided, in the form of an initial pose uncertainty ellipsoid and a velocity error cone. Both uncertainty models ignore the presence uncertainty in the assembly part models, whose dimensions are assumed fixed, and whose surfaces are regarded as smooth and flawless. The author has previously incorporated model error into the uncertainty analysis that accompanies trajectory planning [37], following Donald's use of additional C-space dimensions for representing uncertain model paramters [47]. In the author's experience, the computational cost of performing geometric analyses in an increasingly higher-dimensional space is intractable. In the current approach, therefore, the impact of unmodelled uncertainties in the assembly part dimensions and surface texture is assumed to be negligible or otherwise incorporated into the velocity uncertainty model.

3.1 Heuristic search of local contact space

The sequence of contact states to traverse within a local contact space is chosen through a best-first search of the contact state adjacency graph. The relative shortcomings associated with each branch of the search tree are summarized in the accumulated cost of the branch, which is based on the danger of deviating into an undesired contact state and the difficulty in distinguishing between the contact states according to their force signatures. The best-first search of a local contact state adjacency graph originates from a set of starting states and terminates at one of a number of goal states. If the critical point in question is the first critical point encountered in the original nominal trajectory, then it represents a transition from free space. In that case, the starting states are just the PCs most accessible from free space. Otherwise, the starting states are the contact states shared by the previous critical point's local contact state adjacency graph, where they serve as goal states. Similarly, the goal states of the heuristic search are just the contact states shared by the subsequent critical point's local contact state adjacency graph, where they serve as initial states.

3.1.1 Contact state transitions

The heuristic search of contact space explores alternative sequences of contact state transitions, each representing the formation or separation of a single primitive contact. Each transition involves a starting state, a traversed state, and a goal state. For transitions involving the formation of a contact, the dimension of the traversed state is one greater than the dimension of the goal state. For transitions involving a contact separation, the dimension of the starting state is one less than the dimension of the traversed state. In the

3–dimensional contact space of Figure 27, for instance, the transition from pose $\mathbf{P}_0$ to pose $\mathbf{P}_1$ extends from a 3-dimensional starting state, traversing the same 3-dimensional state to reach a 2-dimensional goal state. This transition represents the formation of a new contact. The transition from pose $\mathbf{P}_2$ to pose $\mathbf{P}_3$ extends from a 2-dimensional starting state, traversing an adjacent 3-dimensional state to a different 2-dimensional goal state. This transition represents the separation of one contact, followed by the formation of another.

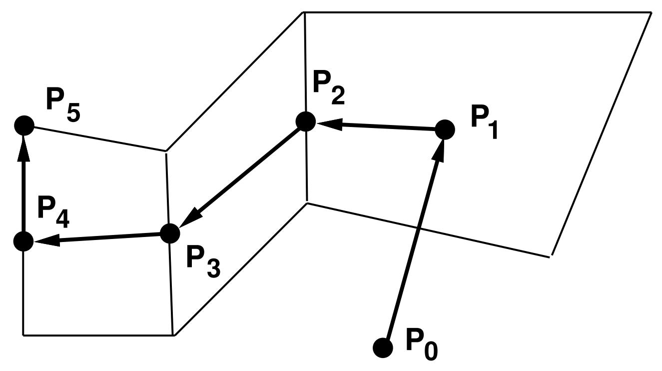


Figure 27: Transitions terminating on sensed force.

Each contact state transition shown in Figure 27 terminates when the force and/or torque exerted by the stationary part against the moving part changes abruptly. A sharp increase or decrease in the force characteristic of the established or separating contact wrench signals the completion of each fine motion. These termination conditions are important for transitions spanning small clearances, where position-servoed terminations are apt to be less precise than the clearance width. In addition to force-terminating transitions, Figure 28 contains position-terminating transitions to goal states of higher dimension than the starting state. These inherently less-precise transitions terminate when the assembly is position-servoed to the goal pose.

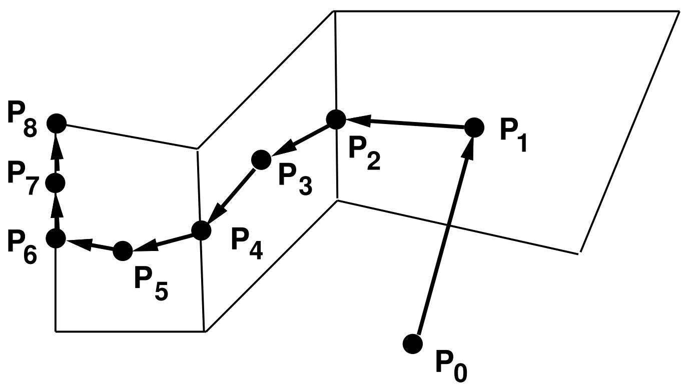


Figure 28: Transitions terminating on sensed position or force.

It is important to note that the control strategy termination conditions described here are *not* reference forces for the control strategies themselves. Rather, they specify the sensory events that signal when to switch from one control strategy to the next. As will be discussed in chapter 3.2, each control stratagy specification includes a reference force in $\Re^6$, consisting of a 6–element bias wrench whose purpose is to maintain contact throughout the motion step. The termination of a motion step is accomplished by monitoring the value of the moving part's wrench in the direction of the wrench signature (see chapter 3.1.2) of the soon-to-be established (or separated) contact.

Following the best-first search scenario, the most promising branch of the search tree is always extended next, "expanding" the current search node, i.e., contact state, by visiting a subset of its adjacent states. For a heuristic search involving the general transition sequences of Figure 28, the "next" contact state to be visited in the search tree may be any state adjacent to the current one, minus any previously visited states. An exception to this rule concerns *transient* contact states, which may be encountered but not actually traversed. A transient contact state contains at least one contact between a convex vertex and a convex edge, In Figure 29, for example, the transient contact state shown has two vertex-plane PCs involving the same vertex V in contact with both faces adjacent to edge E. The resulting contact between vertex V and edge E is encountered for an instant as the square peg slides into the hole. The transient state can serve as a starting state or a goal state, but not the traversed state, for a transition.

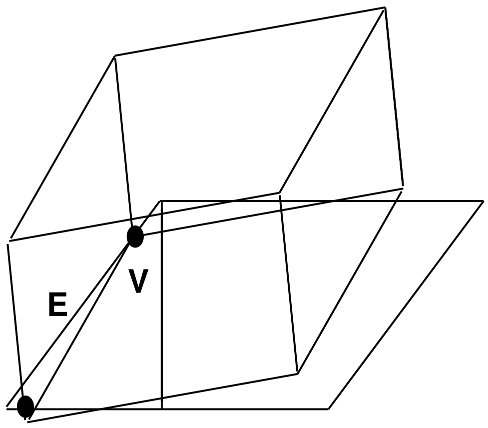


Figure 29: A transient contact state.

3.1.2 Contact wrench signatures

The *moving part wrench* **W** is a 6-element vector describing the current force and torque exerted by the stationary part through the moving part frame. Assuming the robot employed in the assembly operation is equipped with a $6-dof$ force sensor in its wrist, the sensed forces and torques can be readily transformed to the moving part frame, where they provide an estimate of the moving part wrench. The completion of a contact state transition can be detected by sensing an abrupt change in the moving part wrench,

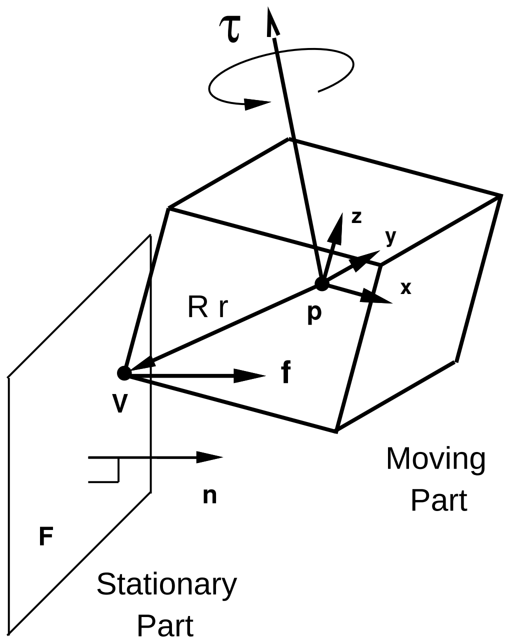


Figure 30: Contact force and torque.

corresponding to the the formation or separation of a primitive contact. In the presence of sensor noise and hard-to-model sliding forces, however, it may be difficult to distinguish the characteristic wrench of an anticipated contact from wrench displacements due to jamming.

The sensed wrench impulse associated with the formation or separation of primitive contact PC_i is nominally parallel to the contact's *wrench signature* $\mathbf{W}_i$, a wrench characteristic of the particular contact and assembly pose. As shown in Figure 30, the characteristic wrench of a PC is determined by the geometry of the contact in question, together with the expected orientation of the moving part. For the vertex-plane contact shown, the characteristic wrench consists of a unit force $\mathbf{f}$, parallel to the nonvarying contact normal $\mathbf{n}$, plus a torque τ. The torque arises from the unit force and the position

$\mathbf{r}$ of the vertex in the model, after rotating $\mathbf{r}$ to the current orientation: $\tau = R\mathbf{r} \times \mathbf{n}$. If required, $\mathbf{f}$ and τ may be transformed back to moving part coordinates by computing $R^{-1}\mathbf{f}$ and $R^{-1}\tau$. In the absence of sensory noise and sliding forces, a PC's wrench signature is parallel or antiparallel to the sensed moving part wrench displacement associated with PC's formation or separation, respectively.

3.1.3 Contact space search heuristics

In order to compare alternative contact state transitions during the best-first search of contact space, the transitions are evaluated according to the following criteria: (1) the danger that the assembly will deviate from the selected path and encounter an *obstacle* PC outside of the sequence, (2) the difficulty in distinguishing between the force signatures of the chosen contact states from likely *error states* outside of the sequence, and (3) the difficulty in distinguishing between sequential contact states according to their force signatures. Criterion (2) is unnecessary in principle, assuming that the fine-motion planning technique is sound and based on valid models of model geometries and motion uncertainties, but it is recommended in practice, to accommodate any unmodelled uncertainties.

To evaluate the cost of a contact state transition during the best-first search, an optimal trajectory segment T is first computed, transporting the assembly from its configuration in the current contact state to an optimal goal pose in the new state. As described in chapter 4, segment T is computed by minimizing its proximity to "obstacle" PCs outside of the pair of contact states involved in the transition. After T is optimized, the contact state transition is evaluated by calculating its cost with respect to the three aforementioned criteria:

$$cost(\mathbf{T}) = w_P cost_P(\mathbf{T}) + w_G cost_G(\mathbf{T}) + w_E cost_E(\mathbf{T}) + cost(prev(\mathbf{T})) \quad (21)$$

where $cost_P(\mathbf{T})$ represents the trajectory's proximity to "obstacle" PCs, $cost_G(\mathbf{T})$ measures the difficulty in distinguishing between traversed state wrenches and the goal state's wrench signature, $cost_E(\mathbf{T})$ measures the difficulty in distinguishing between traversed or goal state wrenches and obstacle states' wrench signatures, and prev($\mathbf{T}$) is the preceding segment in the trajectory. Weights w_P, w_G, w_E represent the relative importance of the three sub-costs. $cost_P$ is described in detail in sections 4.1 through 4.4, while $cost_G(\mathbf{T})$ and $cost_E(\mathbf{T})$ are shown in chapter 4.5.

3.2 Hybrid control strategies

Since the geometry of each contact state to be traversed by the assembly will be known in advance, the fine-motion planner can prescribe a *task-consistent* compliance for the traversal of each sequential state. Given a basis for the tangent space of the traversed facet, and a complementary basis for the normal space, the two sets of axes in $\Re^6$ may be position-controlled and force-controlled separately. Each segment of the planned trajectory is therefore expressed as a hybrid control strategy specification. Each specification includes (1) a set of position-controlled axes in $\Re^6$, i.e., a basis for the traversed state's affine space, (2) a complementary set of force-controlled axes, (3) a velocity for the moving part frame in $\Re^6$, as determined by the start and goal point of the trajectory segment, and (4) the termination condition for the control strategy. In the terminology of chapter 2.5.2, the affine space approximation of a traversed state S with n primitive contacts $PC_1, \ldots, PC_n$ is just $ASpace_{\mathbf{G}}(PC_1, \ldots, PC_n)$, which represents the linearization of S about transition goal pose $\mathbf{G}$. A basis for the orthogonal vector space provides the set of force-controlled axes. For transitions involving the formation of a new contact, termination is signalled by an impulse in the moving part wrench in the direction of the new contact's wrench signature. For transitions terminating within the traversed state, there is no contact formation to signal the assembly's arrival at the goal state, so the transition terminates after the moving part has been servoed to the goal pose, via position-sensing.

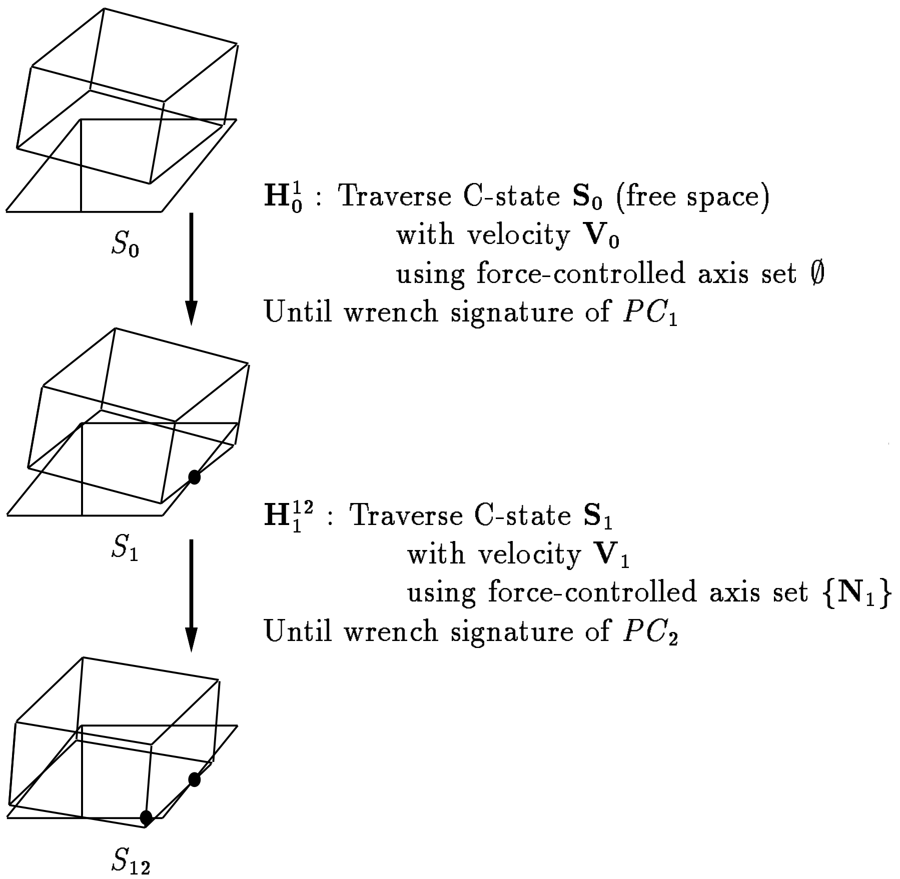


Figure 31: Specifying hybrid control strategies.

Figure 31, for example, depicts two contact state transitions, along with hybrid control strategies for accomplishing each transition. The two control strategy specifications $\mathbf{H}_0^1$ and $\mathbf{H}_1^{12}$ prescribe control behaviors for the transitions from state S_0 (free space) to state S_1, and from S_1 to S_{12}, respectively. The first transition passes through free space, where no axes are force-controlled, and the second transition traverses the 1-point contact state of PC_1, whose contact normal $\mathbf{N}_1$ provides the single force-controlled axis of 6-dimensional C-space. $\mathbf{H}_0^1$ and $\mathbf{H}_1^{12}$ terminate on the wrench signatures of the newly formed contacts — PC_1 and PC_2, respectively.

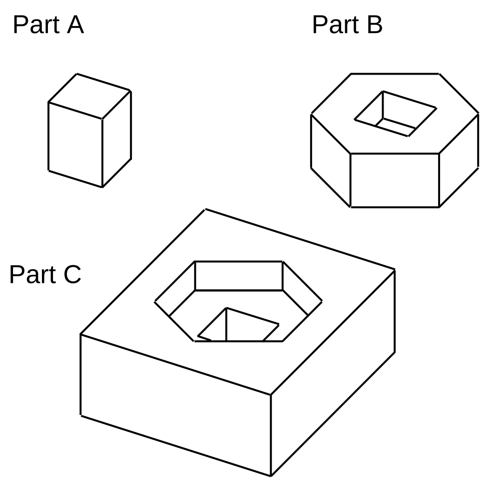


Figure 32: A 3-Part Assembly.

3.3 An illustrative example

In this section, the fine-motion planning technique is demonstrated by way of example, as it applies to the 3-part assembly of Figure 32. This assembly involves square and hexagonal peg-in-hole operations. The high-level assembly planner *KA3* [87, 89, 90] first computes a goal assembly configuration, as shown in Figure 33, satisfying a set of user-specified goal relationships. The goal configuration is derived by intersecting the **TR** symmetry groups associated with the desired kinematic constraints (see [85]). *KA3* then per-

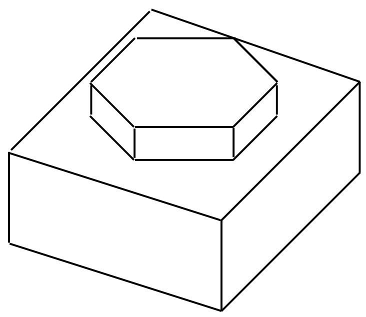


Figure 33: The goal configuration.

forms a disassembly motion analysis to derive the possible assembly motion sequences, one of which appears in 34. This ordering defines three motions, delimited by the five critical points of Figure 34: (1) a motion from $\mathbf{CP}_1$ to $\mathbf{CP}_2$, (2) a motion from $\mathbf{CP}_3$ to $\mathbf{CP}_4$, and (3) a motion from $\mathbf{CP}_4$ to the final configuration, $\mathbf{CP}_5$. The single degree of freedom of mobility associated with each motion gives rise to three nominal trajectories, derived without regard to insertion clearance or uncertainty in the moving part's motion. These nominal trajectories will now be refined by the fine-motion planner to produce more robust, perturbed nominal trajectories.

Nominal trajectory refinement begins with the analysis of the local contact spaces surrounding the critical point configurations shown in Figure 34. Each local contact space is represented by an adjacency graph of contact states, corresponding to the 0- to 5-dimensional facets of the contact space. The adjacency graphs, whose construction is described in chapter 2, provide search spaces for heuristically selecting traversable sequences of contact states. Just prior to this planning stage, insertion clearances are added to assembly part models, which previously fit together without "slop", as in Figures 33 and 34.

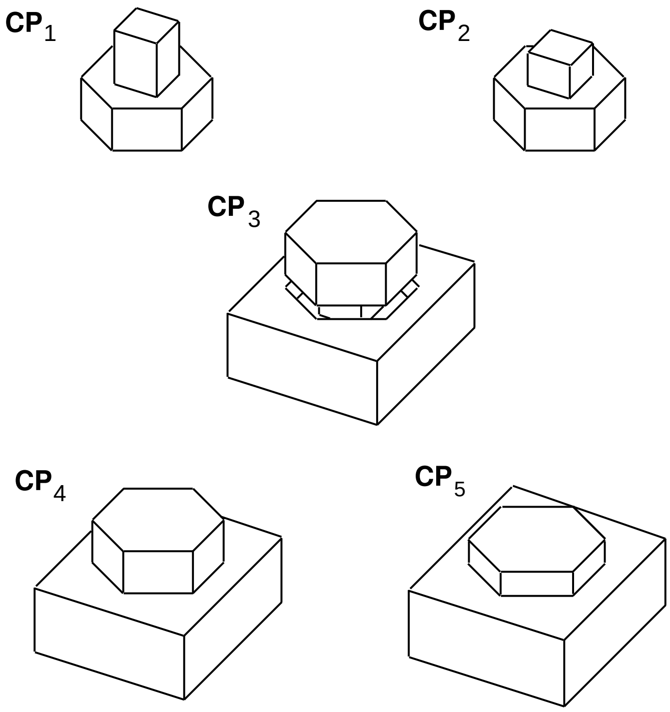


Figure 34: Critical Points for the FAC.

3.3.1 Local contact space analysis

A small portion of the contact state adjacency graph associated with $\mathbf{CP_3}$ is shown in Figure 35. The top level contains PCs that can form between a convex vertex and a face, or between two convex edges. These contact states correspond to 5-dimensional C-surfaces of the local contact space. The contact states associated with lower-dimensional facets arise from combinations of PCs. Arcs in the graph denote state adjacencies, indicating that a state can be reached by forming or breaking a single primitive contact.

Starting from the PCs most accessible from free space, a best-first search of $\mathbf{CP_3}$'s adjacency graph is performed to explore alternative routes through contact space. Candidate sequences of contact states are compared to yield

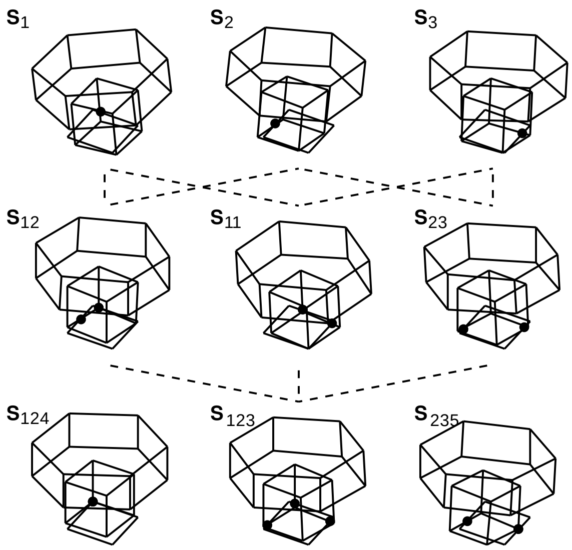


Figure 35: Contact state adjacency graph for $\mathbf{CP_3}$.

the most traversable route, i.e., the one the assembly is most able to follow successfully in the presence of uncertainty. The heuristic search of the local contact space associated with $\mathbf{CP_3}$ can terminate at any 1-dimensional state shared with the local contact space of $\mathbf{CP}_4$. Such a contact state provides an ideal goal state, since it can be traversed to reach the local contact space of $\mathbf{CP}_4$ without lateral velocity error. Shown in the first frame of Figure 36, the shared state then provides a starting state for the heuristic search of $\mathbf{CP}_4$'s local contact space. In the last frame of Figure 36, the candidate sequence again terminates at a contact state shared by the contact space surrounding the next critical point, which in this case is the final configuration of Figure 33. To minimize uncertainty, the planner selects a contact state sequence involving lower-dimensional states, prefering a path through the *1-skeleton* of 0- and 1-dimensional states (see [75]).

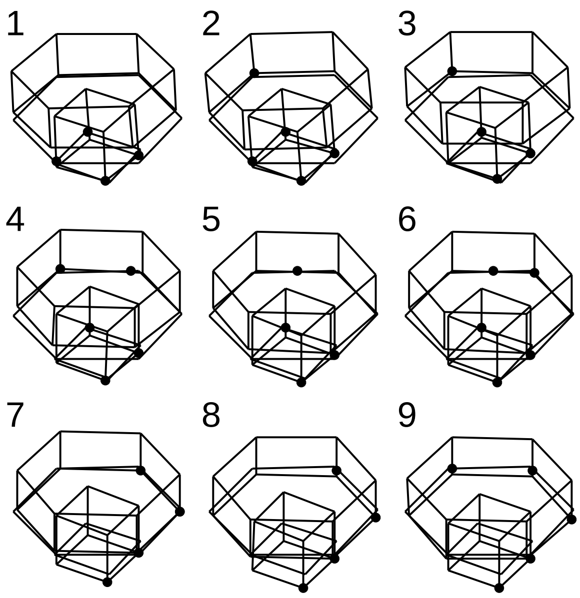


Figure 36: Contact state sequence for $\mathbf{CP}_4$.

3.3.2 Fine-motion trajectory synthesis

The higher-level planning of contact state sequences takes place in parallel with the lower-level synthesis of command trajectories to traverse the candidate sequences. Each trajectory plotted through contact space is optimized to reduce the likelihood that the assembly will deviate from the given sequence in the presence of sensing and control error. Once optimized, the estimated safety with which a command trajectory can steer the assembly through the proposed contact state sequence provides, in turn, a heuristic for selecting or rejecting the contact sequence in question.

Given a candidate sequence of contact states, a command trajectory is first initialized by placing an arbitrary goal pose in each contact state, defining a piecewise-linear trajectory in local contact space. As described in chapter 4, the goal poses are then iteratively perturbed along their C-surface facets to increase the resulting trajectory's distance to "obstacle" contacts

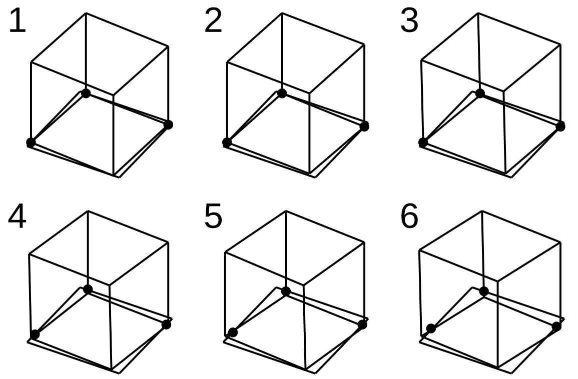


Figure 37: Goal pose refinement.

outside the contact state sequence. Figure 37 traces the iterative refinement of the goal pose for the contact state S_{123} of Figure 35 (*sans* hexagonal Part B). Consisting of three *line-line* contacts, the 3-dimensional state S_{123} has "obstacles" which include nearby vertex-plane contacts. The refinement process incrementally increases the amount of uncertainty that can be tolerated without allowing the possible formation of one of the undesired vertex-plane contacts. The fine-motion plans thus generated are finally expressed as hybrid control strategy specifications, as described in chapter 3.2.

CHAPTER 4

TRAJECTORY PLANNING

This chapter addresses the formation of a piecewise-linear command trajectory through 6-dimensional contact space. The trajectory is generated within a specified sequence of contact states in local contact space. Trajectory synthesis begins with the placement of arbitrary goal poses in the specified contact states. The goal poses define an arbitrary, piecewise-linear trajectory through contact space. The trajectory is then refined by iteratively perturbing each goal pose to increase the trajectory's clearance with respect to adjacent "obstacle PCs" (see Figure 38). Trajectory optimization can be performed with or without a rigorous uncertainty analysis involving ellipsoidal uncertainty regions.

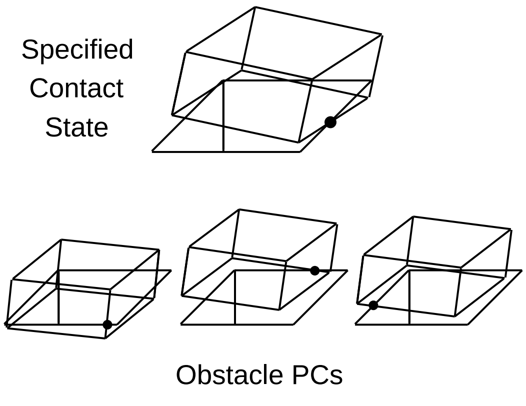


Figure 38: Obstacle PCs adjacent to a specified contact state.

4.1 Obstacle PCs

A trajectory segment is optimized by increasing its distance from the *obstacle PCs* associated with the contact state transition. The obstacle PCs are just the PCs associated with any contact state adjacent to the starting state, traversed state, or goal state, which do not also belong to the PCs of those three contact states. Figure 38 shows the obstacle PCs associated with a transition from free space to a line-line contact. Maximizing the command trajectory's clearance with respect to the obstacle PCs reduces the risk that the assembly will deviate into contact states outside of the specified path through contact space. The obstacle PCs involved in synthesizing a trajectory segment may be enumerated by examining the contact state adjacency graph: first list the PCs associated with the children (adjacent, lower-dimensional contact states) of the starting, traversed, and goal states, and then subtract the PCs of the latter states.

4.1.1 Obstacle PC facets

As discussed in chapter 2.3.6, every primitive contact PC_i in contact space has an associated convex point set S_i', consisting of PC_i's 5-dimensional C-surface, bordered by some n_i bounding PCs. The linearization of S_i' about a fixed assembly pose produces a convex polytope in $\Re^6$, consisting of a number of hyperplanar facets. PC_i's own C-surface produces a 5–dimensional facet, while each of PC_i's bounding PCs forms a 4-dimensional facet where it intersects PC_i's C-surface. S_i' may also have a 3-dimensional facet, but such facets are ignored in this chapter, for simplicity. Hence, an obstacle PC_i can be characterized by $n_i + 1$ facets, one for PC_i's 5–dimensional C-surface and one 4–dimensional facet for each bounding contact:

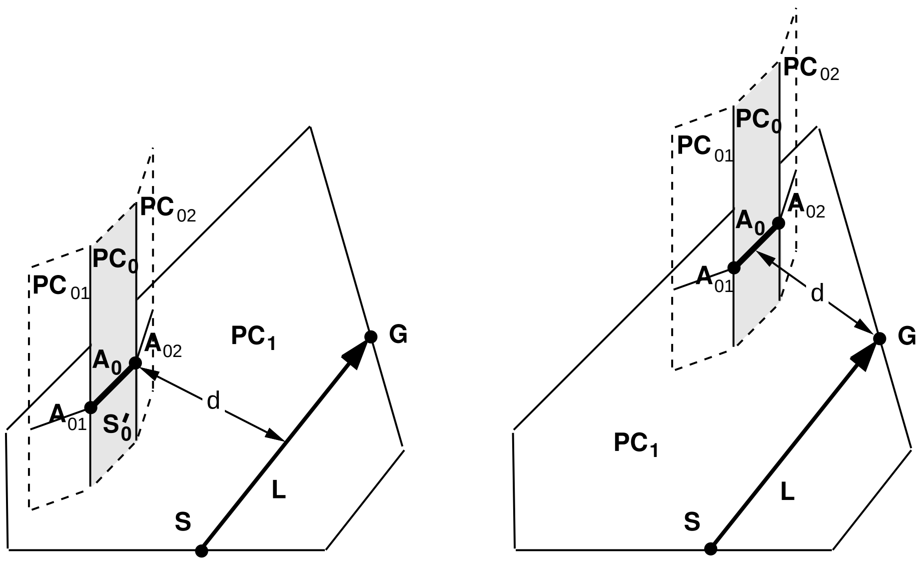
$$
\begin{array}{l}
facet(PC_i) \\
facet(PC_i, PC_{i1}) \\
\quad\quad\vdots \\
facet(PC_i, PC_{in_i})
\end{array}
\tag{22}
$$

As explained in the next section, however, the above facets are not the obstacle PC facets involved in the measurement of the trajectory's distance to the obstacle PC. In order to measure trajectory clearance associated with the traversal of a contact state, lower dimensional facets are required.

4.1.2 Trajectory clearance

The trajectory segment associated with a contact state transition is denoted by a triple $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{G})$. Segment $\mathbf{T}$ extends from a starting pose $\mathbf{S}$ in the starting state of the transition, along a line segment $\mathbf{L}$ in the traversed state, to a goal pose $\mathbf{G}$ in the goal state. The starting pose $\mathbf{S}$ is typically the goal pose of the previous transition. The distance from a trajectory segment $\mathbf{T}$ to an obstacle PC_0 is computed as the shortest distance from a trajectory component $\mathbf{L}$, $\mathbf{S}$, or $\mathbf{G}$ to PC_0's point set S_0', *as measured along the contact space facet of the traversed state.* The latter condition assumes that the uncertain trajectory is at least confined to the traversed state's C-surface, i.e., the established contacts will not separate.

Because the trajectory's distance to an obstacle PC's convex point set is measured along the surface of the traversed state, the relevant portion of the convex point set is its intersection with the traversed state's C-surface in contact space. When traversing a contact state formed by some m primitive contacts $PC_1 \ldots PC_m$, the relevant subfacets of an obstacle PC_0's $n_0 + 1$ facets are the following:



Measuring an obstacle PC's distance to the trajectory segment interior

Measuring an obstacle PC's distance to a trajectory segment endpoint

Figure 39: Measuring a trajectory segment's distance to obstacle PCs.

$$\begin{array}{l} facet(PC_0, PC_1, \ldots, PC_m) \\ facet(PC_0, PC_{01}, PC_1, \ldots, PC_m) \\ \qquad\qquad\vdots \\ facet(PC_0, PC_{0n_0}, PC_1, \ldots, PC_m) \end{array} \tag{23}$$

Assuming the m primitive contacts of the traverse state have independent normals in $\Re^6$, the first of the above subfacets is has dimensionality $(5-m)$, while the remainder are $(4-m)$-dimensional facets.

To illustrate the above discussion of subfacets, Figure 39 depicts an obstacle PC's convex point set, and its intersection with a traversed contact state, shown schematically in 3 dimensions. Obstacle PC_0's convex point set has one 5-dimensional (shaded) facet associated with PC_0 itself, plus two bordering 4-dimensional facets (shown as vertical lines) arising from its bounding contacts PC_{01} and PC_{02}. In this example, the traversed contact state is a lone C-surface, so $m = 1$. Since the uncertain trajectory is confined to the traversed state's C-surface, the relevant obstacles are in fact the (darkened) subfacets that intersect the traversed contact state of PC_1. In the example shown in Figure 39, these subfacets are $facet(PC_0, PC_1), facet(PC_0, PC_{01}, PC_1)$, and $facet(PC_0, PC_{02}, PC_1)$. The trajectory's shortest distance d to the obstacle PC's point set extends from the interior **L** of trajectory segment, or one of its endpoints **S** or **G**, to the affine space approximation of one of these three subfacets. More generally, the shortest measured distance extends from **L**, **S** or **G** to the affine space approximation of one of the $n_0 + 1$ subfacets of expression 23.

Since the facets of a local contact space are nearly flat, the linearization of each obstacle PC facet about the current goal pose produces a suitable affine space approximation which facilitates the rapid, repetitive distance calculations performed during trajectory refinement. Chapter 2.5.2 describes how to compute an affine space approximation of a contact space facet. The affine spaces for the obstacle facets in Figure 39 are thus obtained:

$$\begin{aligned} A_0 &= ASpace_{\mathbf{G}}(PC_0, PC_1) \\ A_{01} &= ASpace_{\mathbf{G}}(PC_0, PC_{01}, PC_1) \\ A_{02} &= ASpace_{\mathbf{G}}(PC_0, PC_{02}, PC_1) \end{aligned} \tag{23}$$

4.2 Scaling C-space

The trajectory refinement process described in this chapter takes on two forms. In its most basic form, the placement of piecewise-linear trajectory segments is based on maximizing the segment's clearance with regard to obstacle PCs. In the more rigorous form of the refinement process, clearance is maximized with respect to the forward-projection volumes surrounding the uncertain trajectory segments. As described in chapter 4.6, the procedure for constructing forward-projections requires models of the uncertainty in the initial pose and velocity of the moving part.

In its simpler form, however, the trajectory refinement procedure only requires the ratio σ_r between the expected degrees of translational uncertainty and rotational uncertainty. To treat translational and rotational distance measurements equitably, a linear transformation $T : \Re^6 \rightarrow \Re^6$ is applied to contact space, "stretching" its rotational component by a factor of σ_r. Derived in equations 6...10, the descriptors $\mathbf{N}_i, D_i$ of each PC_i's hyperplane are modified and represented in the scaled contact space. The distance calculations described below are then performed in the scaled contact space, wherein a unit of translational clearance with respect to an obstacle represents the same margin of safety as a unit of rotational clearance.

A linear scaling of 6-dimensional C-space may be specified by a 6-element scaling vector $\vec{\sigma}$ and accomplished with a transformation $T : \Re^6 \rightarrow \Re^6$ which maps each point $\mathbf{X} \in \Re^6$ to $\mathbf{X}' \in \Re^6$, where $\mathbf{X}'_i = \vec{\sigma}_i \mathbf{X}_i,\ i = 1 \ldots 6$. The transformation T maps a halfspace $\{\mathbf{X} \in \Re^6 \,|\, \mathbf{N} \cdot \mathbf{X} \leq D\}$ to a new halfspace $\{\mathbf{X} \in \Re^6 \,|\, \mathbf{N}' \cdot \mathbf{X} \leq D'\}$ whose descriptors $\mathbf{N}'$ and D' are computed as follows. Let

$$\mathbf{N}^* \in \Re^6, \quad \mathbf{N}^*_i = \frac{\mathbf{N}_i}{\vec{\sigma}_i} \ (i = 1 \ldots 6) \tag{25}$$

Then

$$\mathbf{N}' = \frac{\mathbf{N}^*}{|\mathbf{N}^*|}\,, \ D' = D\mathbf{N} \cdot \mathbf{N}' \tag{26}$$

To achieve a uniform expansion of C-space along its three rotational axes by a factor of σ_r, use scaling vector $\vec{\sigma} = [1\ 1\ 1\ \sigma_r\ \sigma_r\ \sigma_r]^T$. Alternatively, a linear scaling of 6-dimensional C-space may be specified by $n \leq 6$ scaling factors $\sigma_1 \ldots \sigma_n$ and n orthonormal axes $\mathbf{U}_1 \ldots \mathbf{U}_n$ of 6-space. We first augment the n scaling factors with $\sigma_{n+1} = \cdots = \sigma_6 = 1$ and the n axes with $6-n$ complementary axes $\mathbf{U}_{n+1} \ldots \mathbf{U}_6$, so that $\mathbf{U}_1 \ldots \mathbf{U}_6$ form an orthonormal basis for $\Re^6$. Then transformation $T : \Re^6 \rightarrow \Re^6$ is defined

$$\mathbf{X}' = \sum_{i=1}^{6} \sigma_i \left(\mathbf{X} \cdot \mathbf{U}_i\right) \mathbf{U}_i \tag{27}$$

Halfspaces are transformed as previously, except the normal $\mathbf{N}^*$ of equation 25 is now computed as follows.

$$\mathbf{N}^* = \sum_{i=1}^{6} \frac{\mathbf{N} \cdot \mathbf{U}_i}{\sigma_i} \mathbf{U}_i \tag{28}$$

4.3 Measuring trajectory clearance

Procedures are now outlined for measuring the shortest distance from a trajectory segment to an obstacle PC's convex point set S', through an m-dimensional contact state. The measured distance extends from a trajectory component $\mathbf{L}$, $\mathbf{S}$ or $\mathbf{G}$ to the closest affine space approximation of the subfacets of Equation (23). Provided that the C-surface normals are indepedent, an affine space of $facet(PC_0, PC_1, \ldots, PC_m)$, such as A_0 in Figure 39, has dimensionality $(5-m)$, while the affine space of each subfacet $facet(PC_0, PC_{0i}, PC_1, \ldots, PC_m)$, such as A_{01} or A_{02} in Figure 39, is $(4-m)$-dimensional.

The specific trajectory component and subfacet of PC_0 involved in the shortest distance measurement depends on the closest points' satisfaction of various proximity conditions. In Figure 39, for example, the closest obstacle facet to the trajectory segment is A_{01} or A_{02}, provided the measurement extends to a point on segment interior $\mathbf{L}$ between $\mathbf{S}$ and $\mathbf{G}$. Otherwise, the

```
/* Compute the shortest distance between affine spaces A_1 and A_2. */
dist(A_1, A_2) → d

/* Compute the shortest distance between affine spaces A_1 and A_2,
    as well as the closest point P_1 on A_1 to A_2. */
dist1(A_1, A_2) → d, P_1

/* Compute the shortest distance between affine spaces A_1 and A_2,
    as well as the closest point P_2 on A_2 to A_1. */
dist2(A_1, A_2) → d, P_2
```

Figure 40: Subroutines for computing affine space distance.

closest trajectory component is **S** or **G**. The closest obstacle facet to **G** is A_0, provided the measurement extends to a point on A_0 between A_{01} and A_{02}. Otherwise, the closest facet is A_{01} or A_{02}.

4.3.1 Subroutines for computing distance

Subroutines are defined in figure 40 for computing trajectory segment clearances measurements. The reader should assume that these subroutines collectively perform the procedure described in Appendix D for computing the distance separating two affine spaces A_1 and A_2, the closest point $\mathbf{P}_1$ on A_1 to A_2, and the closest point $\mathbf{P}_2$ on A_2 to A_1.

Figure 41 illustrates the calculation of the shortest distance from the trajectory segment to a specified $(4-m)$-dimensional facet of an obstacle PC. The obstacle PC, linearized for the purpose of distance measurement, is represented by $(5-m)$-dimensional facet A_0 and $(4-m)$-dimensional facets A_{01} and A_{02} (where A_0 meets A_1 and A_2). When computing the trajectory segment's distance to affine space A_{01}, the closest point $\mathbf{P}_1$ on the segment line ℓ is first calculated. Noting that $\mathbf{P}_1$ lies beyond endpoint $\mathbf{G}$, however, it is apparent that $\mathbf{G}$ is in fact the segment's closest point to A_{01}. When

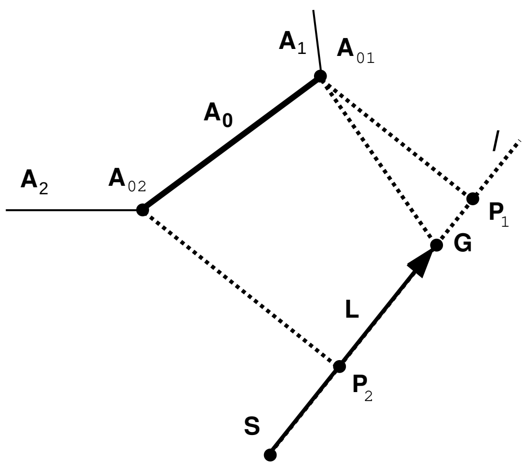


Figure 41: A segment's unsigned distance to a bounding subfacet.

computing the segment's distance to A_{02}, on the other hand, the closest point $\mathbf{P}_2$ on segment line ℓ is found to lie between the endpoints $\mathbf{S}$ and $\mathbf{G}$, so $\mathbf{P}_1$ represents the closest segment point.

To systematize the preceding analysis with regard to Figure 41, the procedure of Figure 42 states that the measured distance from trajectory segment interior $\mathbf{L}$ to a $(4-m)$-dimensional affine space A_{0i} is only defined when the closest point on segment line ℓ to A_{0i} lies between the segment endpoints.

/* Compute the distance d from the trajectory segment interior
  L to the $(4-m)$-dimensional affine space A_{0i}. */
define $dist^{\dagger}(\mathbf{L}, A_{0i}) \rightarrow d$
  $dist1(\ell, A_{0i}) \rightarrow d, \mathbf{P}_1$
  if $\mathbf{P}_1$ lies between $\mathbf{S}$ and $\mathbf{G}$
    then return d
  else return ∞
  endif
enddefine

Figure 42: Calculating a segment interior's distance to a facet.

The measurement of the shortest distance from a trajectory segment endpoint to the convex point set of an obstacle PC is depicted in Figure 43. After

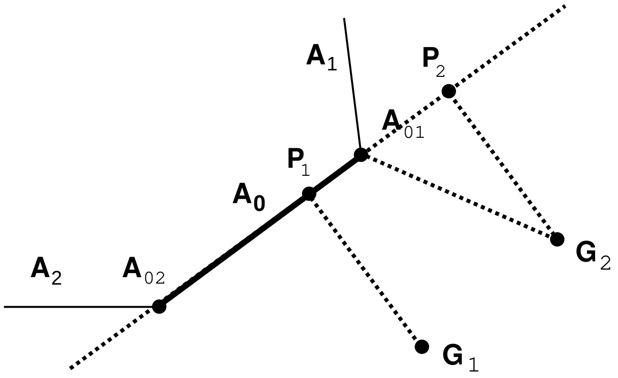


Figure 43: A goal point's unsigned distance to an obstacle PC.

the closest point $\mathbf{P}_1$ on affine space A_0 to goal pose $\mathbf{G}_1$ is computed, it is found to satisfy all of the bounding PC constraints, since $\mathbf{P}_1$ lies between A_{01} and A_{02}). Hence, $\mathbf{P}_1$ is indeed the obstacle PC's closest point. On the other hand, the closest point $\mathbf{P}_2$ on A_0 to the goal pose $\mathbf{G}_2$ fails to satisfy the bounding PC constraint corresponding to the affine space A_{01}, so the obstacle PC's closest point must lie on A_{01}.

The latter discussion is embodied in procedure *dist** (Figure 44), which measures the distance from a trajectory segment endpoint to the $(5-m)$-dimensional affine space A_0. As with *dist*†, the distance is undefined when the closest point on A_0 fails to satify some bounding contact's constraint.

```
/* Compute the distance d from a trajectory segment endpoint
     P to the affine space A_0 of PC_0's convex point set. */
define dist*(P, A_0) → d
   dist2(P, A_0) → d, P_2
   if P_2 satisfies each bounding PC_0i constraint
      then return d
   else return ∞
   endif
enddefine
```

Figure 44: Calculating a segment endpoint's distance to a facet.

4.3.2 Negation of distance measurements

Procedures $dist^{\dagger}$ and $dist^{\star}$ of Figures 42 and 44 represent a means for computing the distances of separation between the affine spaces comprising a trajectory segment (i.e., its line and endpoints) and the linearized facets of an obstacle PC. Assuming the trajectory segment lies outside of the forbidden zone, the positive values returned by $dist^{\dagger}$ and $dist^{\star}$ are valid. To accommodate configurations in which the trajectory segment passes through the forbidden zone, however, these procedures must be modified to return negative distances which signify the extent of the overlap in C-space. The trajectory optimization process described in chapter 4.4 will then perturb the trajectory segment in the appropriate direction for overlapping, as well as nonoverlapping conditions. The conditions for detecting overlap are discussed here briefly.

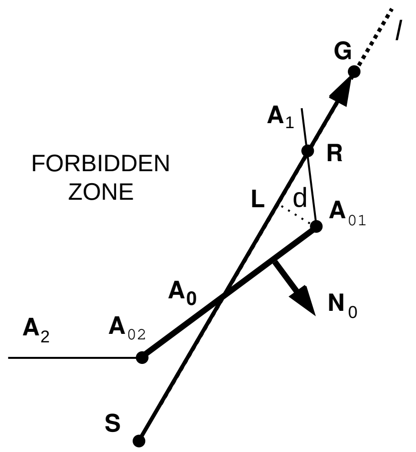


Figure 45: A trajectory segment's signed distance to a bounding subfacet.

In Figure 45, a trajectory segment overlaps the forbidden zone by passing "behind" the $(4 - m)$-dimensional facet A_{01} of an obstacle PC. Segment interior $\mathbf{L}$'s distance from A_{01} represents the extent of overlap, so the distance returned by procedure $dist^\dagger$ should be negative. The $(4-m)$-dimensional facet A_{01} is just the intersection of $(5-m)$-dimensional facets A_0 and A_1, where A_0 represents the obstacle contact PC_0, while A_1 represents PC_0's bounding contact PC_1. The overlap can be expressed algorithmically by noting that trajectory line ℓ intersects A_1 at a point $\mathbf{R}$ which lies "behind" A_{01} with respect to A_0's outward normal $\mathbf{N}_0$. Procedure $dist^\dagger$ is therefore augmented with the computation shown in Figure 46, whereby any noninfinite distance d is negated in the presence of overlap.

/* Negate distance d from L to A_{i0} if overlap is present */
$\ell \cap A_i \rightarrow \mathbf{R}$
Let $\mathbf{P}_2$ be a point on A_{0i}
if $\mathbf{R} \cdot \mathbf{N}_0 < \mathbf{P}_2 \cdot \mathbf{N}_0$
    then $-d \rightarrow d$
endif

Figure 46: Negating a segment interior's clearance.

The trajectory segment in Figure 47 also overlaps the forbidden zone, with its endpoint $\mathbf{G}$ lying "behind" the $(5-m)$-dimensional facet A_0 of the obstacle PC.

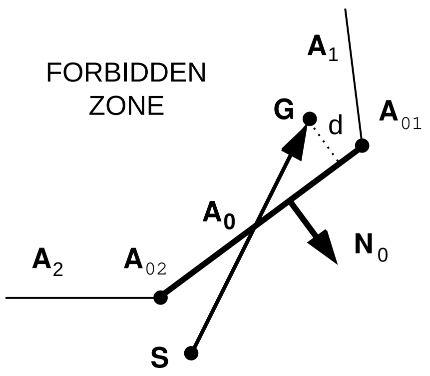


Figure 47: A goal point's signed distance to an obstacle PC.

$\mathbf{G}$'s distance d to A_0 represents the extent of overlap, so d must be negated in procedure $dist^\star$, as shown in Figure 48.

```
/* Negate distance d from P to A_0 if overlap is present */
Let P_1 be a point on A_0
if P · N_0 < P_1 · N_0
    then −d → d
endif
```

Figure 48: Negating a segment endpoint's clearance.

4.3.3 Computing the minimum distance

The minimum distance between a trajectory segment $\mathbf{T}$ and the convex point set of an obstacle contact PC_0 is now defined, using the distance functions defined above.

$$Dist(\mathbf{T}, PC_0) = \min_{i \in 1 \ldots n_0, \mathbf{P} \in \mathbf{S}, \mathbf{G}} (dist^{\dagger}(\mathbf{L}, A_{0i}), dist^{\star}(\mathbf{P}, A_0), dist(\mathbf{P}, A_{0i})) \tag{29}$$

In an alternative approach, only the obstacle clearance of the trajectory segment's goal pose is measured, ignoring that of the segment's interior. This is accomplished by dropping the $dist^{\dagger}(\mathbf{L}, A_{0i})$ term from Equation (29). Although the simpler clearance measurement is less rigorous, the same goal pose can serve as the endpoint of more than one trajectory segment, since its refinement does not depend on the segment's starting pose. Consequently, the number of pose refinements performed is greatly reduced.

4.4 Refining a trajectory

A trajectory segment's proximity to each obstacle PC influences the estimated likelihood that the desired contact state transition will fail, as embodied in the following cost function:

$$cost_P(\mathbf{T}) = \sum_k \frac{1}{C - Dist(\mathbf{T}, PC_k)} \tag{30}$$

The proximity function in the denominator of equation 30 returns a negative distance when the trajectory segment overlaps the obstacle PC. Asymtote C determines the overlap distance at which the cost becomes infinite. Note that the closest obstacles are the most significant in equation 30, so they receive the "most attention" during trajectory refinement.

4.4.1 Iterative optimization of a trajectory segment

To optimize a single trajectory segment, the method of steepest descent is employed to perturb the goal pose along the goal state's surface until a minimum trajectory cost is obtained, as computed in equation 30. The linearization of obstacle PC facets described in chapter 2.5.2, which yields the affine space approximations used in chapter 4.3, is performed by linearizing each obstacle PC about the current goal pose. Because local contact space is nearly linear to begin with, each linearization remains valid for several goal pose perturbations. To calculate the direction in which to perturb the goal pose, the cost function's gradient $\bigtriangledown_{cost}$ is computed numerically in the goal state's vector space. Since $\bigtriangledown_{cost}$ represents the direction in which $cost(\mathbf{T})$ increases most rapidly, the goal pose is perturbed a fixed distance in the opposite direction. This process repeats iteratively until the magnitude of $\bigtriangledown_{cost}$ falls below a threshold, indicating the presence of a minimum. Contact space is generally not convex, so the steepest descent approach to trajectory refinement can fail due to local minima. Observed failures due to false minima have been relatively infrequent, however.

4.4.2 Optimizing a multiple-segment trajectory

The fine-motion trajectory can be optimized one segment at a time, or several segments may be optimized in parallel. In the simplest approach, each new trajectory segment is optimized after it is created during the the heuristic search of contact space. When the search tree extends into a new contact state, an arbitrary goal pose is placed in the new goal state and then iteratively optimized to produce a command trajectory for the new contact state transition. Once the goal pose is refined through iterative optimzation, it serves as a fixed starting pose for any trajectory segment extending from the current contact state to an adjacent state.

In a more general approach, the n most recently created segments in a trajectory are refined in parallel. This approach calls for simultaneous perturbations of the n most recently formed goal poses, using a gradient $\bigtriangledown_{cost}$ defined in the combined vector spaces of the last n goal states. The simpler technique described above is just the case where $n = 1$. Regardless of n, the combined, optimized costs of the all the trajectory segments making up the entire candidate trajectory contribute, in part, to the heuristic evaluation of the contact state sequence in which the trajectory is defined. Employing this technique for $n > 1$ is more computationally expensive than for $n = 1$, since the gradient descent search space is of greater dimension, and trajectory segments are subject to repeated optimizations. On the other hand, the more general approach enables the fine-motion planner to consider the effect of a goal pose placement on subsequent segments in the trajectory.

4.4.3 Local minima and C-space concavities

The trajectory refinement process described in chapter 4.4 may not always yield segments that are globally optimal with respect to proximity function $cost_P$. The goal pose computed via gradient descent may in fact lie at local minimum, owing to the concavities of C-space. In Figure 49, for example, a 2-dimensional C-space contains two convex regions A and B. Equipotential curves within the C-space represent proximity to the repellant C-surfaces. The concavity of C-space is reproduced by the equipotential curves, which mimic the shape of C-space near its outlying border and form islands at the centers of the convex regions A and B. Moreover, the duplicity of proximity function minima arises from the concavity of C-space. Although the global minimum lies in A, a goal pose initially placed in region B may be unable to migrate to the global minimum while descending the gradient of $cost_P$.

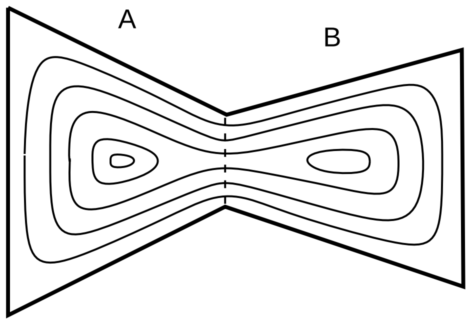


Figure 49: Local minima associated with C-space concavities.

An obvious solution to the dilemma posed by local minima would be to employ the procedure described in chapter 2.7 to partition C-space into quasi-convex regions. With C-space thus partitioned, trajectory segment optimization would take place within each individual quasi-convex region C_i, using artifical repellant constraints to restrict the goal pose to C_i. After performing trajectory segment optimization for each region C_i, the fine-motion planner would choose the trajectory segment associated with the lowest local minimum encountered. Unfortunately, the use of artificial repellant constraints compromises the accuracy of the resulting minima. If the true minimum lies on or near the hyperplane of one or more repellant constraints, the goal pose will be unable to approach it. In general, the addition of artificial constraints alters the terrain of the potential fields traversed by the migrating goal pose, modifying the global minimum itself. Further investigation will be required to determine the means by which to derive the true minimum reliably and effiently.

4.5 Evaluating a trajectory

Once a trajectory segment T has been optimized with respect to obstacle PC clearance, the proximity function $cost_P(\mathbf{T})$ of equation 30 provides the first of four criteria by which to evaluate the contact state transition, as discussed at the start of chapter 3.1. The overall evaluation of T is performed by the cost function $cost(\mathbf{T})$ of equation 21, which, in addition to $cost_P(\mathbf{T})$, tallies sub-costs $cost_G(\mathbf{T})$ and $cost_E(\mathbf{T})$, regarding the distiguishibility of PC wrench signatures encountered along the trajectory segment.

$cost_G(\mathbf{T})$ measures the difficulty with which the wrench signature of the goal state may be differentiated from the wrenches encountered in the traversed state. This is an important criterion for selecting a contact state transition, since recognition of the goal state's wrench signature is necessary for switching hybrid control strategies between trajectory segments. The traversed contact state has some n associated PCs, with characteristic wrenches

$\mathbf{W}_1 \ldots \mathbf{W}_n$. Ignoring frictional forces[3] and sensor error, the moving part wrench encountered within the traversed state will be a linear combination of $\mathbf{W}_1 \ldots \mathbf{W}_n$. For a transition involving the formation of a new primitive contact with characteristic wrench $\mathbf{W}_{n+1}$, the degree to which the new contact might be confused with the existing contacts in the presence of sensor error is estimated by projecting $\mathbf{W}_{n+1}$ into the vector space of $\mathbf{W}_1 \ldots \mathbf{W}_n$:

$$\mathbf{W}'_{n+1} = proj(\mathbf{W}_{n+1}, span(\mathbf{W}_1 \ldots \mathbf{W}_n)) \tag{31}$$

The magnitude of $\mathbf{W}_{n+1}$'s projection then serves to indicate the difficulty in distinguishing the goal state's wrench from the wrenches of the traversed state:

$$cost_G(T) = \left|\mathbf{W}'_{n+1}\right| \tag{32}$$

The difficulty in recognizing that the assembly has entered an error state is measured by function $cost_E(\mathbf{T})$. Given an obstacle contact PC_k with characteristic wrench $\mathbf{W}_k$, the degree to which PC_k might be confused with the traversed state's PCs or the goal PC is estimated by projecting $\mathbf{W}_k$ into the vector space of $\mathbf{W}_1 \ldots \mathbf{W}_{n+1}$:

$$\mathbf{W}'_k = proj(\mathbf{W}_k, span(\mathbf{W}_1 \ldots \mathbf{W}_{n+1})) \tag{33}$$

As in proximity function 30, priority is given to contacts closest to the trajectory segment, because they represent the most probable error states:

$$cost_E(\mathbf{T}) = \sum_k \frac{|\mathbf{W}'_k|}{C - Dist(\mathbf{T}, PC_k)} \tag{34}$$

[3] A more complete analysis, incorporating frictional forces at the contact points, is omitted here for brevity

4.6 Uncertainty analysis

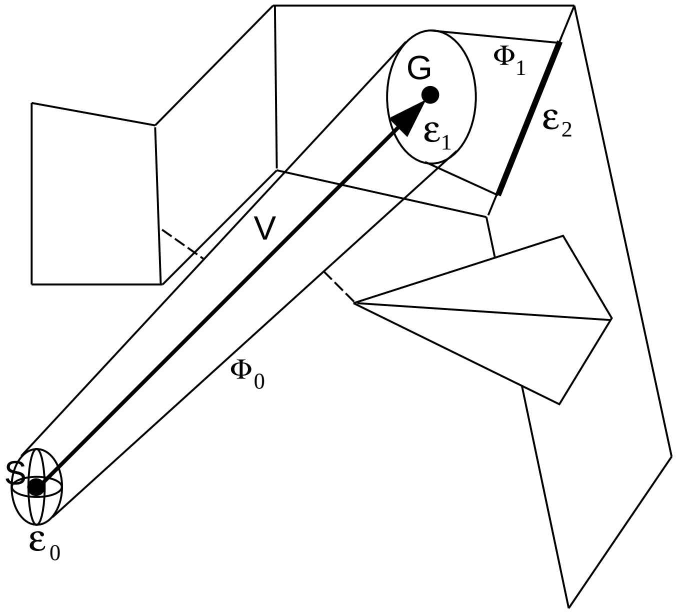


Figure 50: Forward-projections of ellipsoidal uncertainty regions.

In this section, the trajectory refinement process is augmented with a more explicit representation of uncertainty. Pose uncertainty is characterized as a hyperellipsoid ϵ (hereafter called an "ellipsoid") surrounding each succeessive command pose in the fine-motion trajectory (see Figure 50). In the presence of velocity uncertainty, an uncertainty ellipsoid sweeps out a conic, forward-projection volume Φ. The dimensionality of the projection volume is that of the traversed contact state, or in the case of free space traversals, a six-dimensional volume. To incorporate uncertainty analysis, the trajectory refinement process is modified to maximize the clearance of Φ, rather than the trajectory segment, with respect to obstacle PCs.

4.6.1 Forward-projections versus backward-projections

The LMT approach to fine-motion planning with uncertainty [50, 97] is based on the recursive backchaining of subgoal regions from a final goal region, involving the construction of successive pre-images or "backprojections" away from the goal. This geometric uncertainty analysis reveals the multitude of contact states, and regions therein, from which the current subgoal can be reached reliably and recognizably. In the worst case, however, the LMT approach requires double-exponential time in the number of plan-steps [28]. In practice, a great deal of the computational intractability may arise from the need to construct (or deal with in some manner) pre-images that span a multitude of contact states. The topology of a pre-image in contact space often mirrors the complex topology of contact space itself, since the borders of a contact space facet, i.e., its neighboring, lower-dimensional facets, are projected to form the borders of the pre-image. The inherent complexity of constructing multi-state pre-images motivated the author's policy of restricting the sequence of contact states traversed by the assembly to a predetermined, determinate sequence.

While backpropagation is suitable for planning which allows a nondeterministic sequence of contact states, the advantage of pre-image backchaining is nullified when one introduces the requirement of deterministic contact state transitions. Furthermore, LMT's use of nondirectional pre-images, based on the assumption that the robot will determine the appropriate velocity online from (e.g.) position-sensing, is unsuitable for our planning domain of local contact spaces and narrow insertion clearances. Since the uncertainty associated with the pose information derived from sensory data is apt to be of greater magnitude than simple knowledge of the contact state's identity, use of the LMT method would be confined to directional pre-images. In such a scenario, the problem of deriving maximal pre-images reduces to the optimal selection of velocities (or equivalently, goal poses), similar to the trajectory planning technique described in this thesis, but employing topologically complex pre-images rather than forward-projections. Moreover, the method of uncertainty analysis described in the following sections relies on the forward-projection of hyperellipsoidal uncertainty regions, whose simple topology varies only with the dimensionality of the traversed contact state.

4.6.2 Representing uncertainty

The uncertainty region surrounding a trajectory point $\mathbf{S}$ is characterized as an ellipsoidal point set residing in the affine space of the trajectory point's contact state. The point set is bounded by an ellipsoid centered at $\mathbf{S}$, with a surface equation expressed in a coordinate frame whose axes $\mathbf{C}_1 \ldots \mathbf{C}_n$ form a basis for the contact state's vector space. The surface equation $\mathbf{X}^T A \mathbf{X} = 1$ expresses the n-dimensional ellipsoid in the coordinate frame of the contact state. The symmetric, $n \times n$ matrix A may be diagonalized (or expressed *a priori* as a diagonal matrix):

$$A = BDB^T \tag{35}$$

where the columns of B are the principle axes of the ellipsoid (the eigenvectors of A), and diagonal matrix D holds the n eigenvalues $D_{ii} = 1/r_i^2$, which correspond to the principle axis radii r_i. In Figure 51, for example, an uncertainty ellipse ϵ lies in the affine space of a 2-dimensional contact state, with basis vectors $\mathbf{C}_1$ and $\mathbf{C}_2$. Centered at trajectory point $\mathbf{S}$, the ellipse has two principle axes $\mathbf{B}_1, \mathbf{B}_2$, which are defined in $\mathbf{C}_1, \mathbf{C}_2$-space.

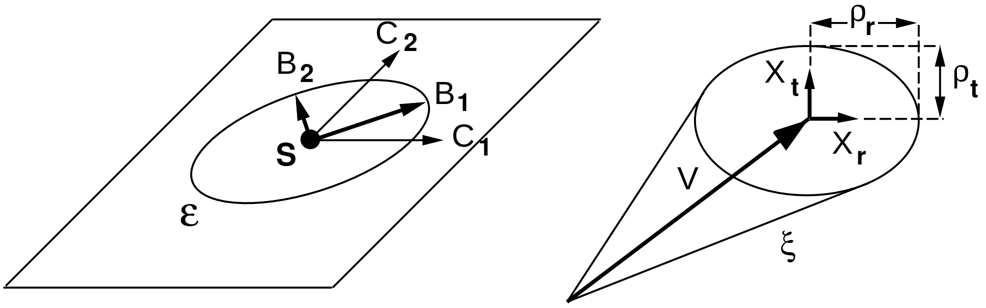


Figure 51: Uncertainty ellipse and velocity cone.

Velocity uncertainty is characterized as a velocity cone ξ whose ellipsoidal base bounds the maximum lateral displacement along any direction $\mathbf{U}$ perpendicular to the unit command velocity $\mathbf{V}$. As shown in Figure 51, the rate of increase in the uncertainty along translational axes $\mathbf{X}_t$ is assumed to be a constant ρ_t, while the rate of increase along rotational axes $\mathbf{X}_r$ is denoted ρ_r. The rate of increase in uncertainty along a general perpendicular $\mathbf{U} \perp \mathbf{V}$ is provided as a function $\nu(\mathbf{U}, \mathbf{V})$, which returns the maximum displacement along $\mathbf{U}$ per unit command displacement along $\mathbf{V}$. The ellipsoidal bases for the velocity cones of all unit command velocities in C-space are represented *en masse* as a single, 6-dimensional ellipsoid:

$$\frac{1}{\rho_t^2}X_1^2 + \frac{1}{\rho_t^2}X_2^2 + \frac{1}{\rho_t^2}X_3^2 + \frac{1}{\rho_r^2}X_4^2 + \frac{1}{\rho_r^2}X_5^2 + \frac{1}{\rho_r^2}X_6^2 \;=\; 1 \tag{36}$$

The velocity cone base for a given command velocity $\mathbf{V}$ is the 5-dimensional ellipsoid where the ellipsoid 36 intersects the 5-dimensional vector space orthogonal to $\mathbf{V}$. For a general lateral displacement direction $\mathbf{U} \perp \mathbf{V}$, the rate of uncertainty is calculated by determining the point on the ellipsoid of equation 36 corresponding to direction $\mathbf{U}$. As explained in Appendix E, the point in question is just $\lambda_{\mathbf{u}}\mathbf{U}$, where

$$\lambda_{\mathbf{u}} = \frac{1}{\sqrt{\frac{u_1^2}{\rho_t^2} + \frac{u_2^2}{\rho_t^2} + \frac{u_3^2}{\rho_t^2} + \frac{u_4^2}{\rho_r^2} + \frac{u_5^2}{\rho_r^2} + \frac{u_6^2}{\rho_r^2}}} \tag{37}$$

Finally, we assume that a unit of rotational velocity gives rise to σ_r times the (translational and rotational) uncertainty arising from a unit of translational uncertainty. Accordingly, the command velocity is "stretched" along the rotational dimensions of C-space by a factor of σ_r, yielding a vector $\mathbf{V}'$ whose length reflects the degree of uncertainty associated with $\mathbf{V}$:

$$\mathbf{V}' = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ \sigma_r V_4 \\ \sigma_r V_5 \\ \sigma_r V_6 \end{pmatrix} \tag{38}$$

In sum, the rate of increase in uncertainty along $\mathbf{U}$ associated with a command velocity is estimated to be

$$\nu(\mathbf{U}, \mathbf{V}) = \lambda_{\mathbf{U}} |\mathbf{V}'| \tag{39}$$

where $\lambda_{\mathbf{U}}$ and $\mathbf{V}'$ are calculated as in equations 37 and 38.

4.6.3 Computing forward-projection cross-sections

Assuming the velocity cone angle is small, a conic forward-projection may be approximated by a cylindrical volume whose ellipsoidal radii remain the same throughout the trajectory segment. As shown in Figure 52, an uncertainty volume ϵ_0 sweeps out a cylindrical forward-projection Φ_0', which approximates the expanding Φ_0 in Figure 50. The uniformity of Φ_0' along the command velocity, as well as lateral uniformity achieved through scaling in chapter 4.2, simplifies the task of measuring the forward-projection's

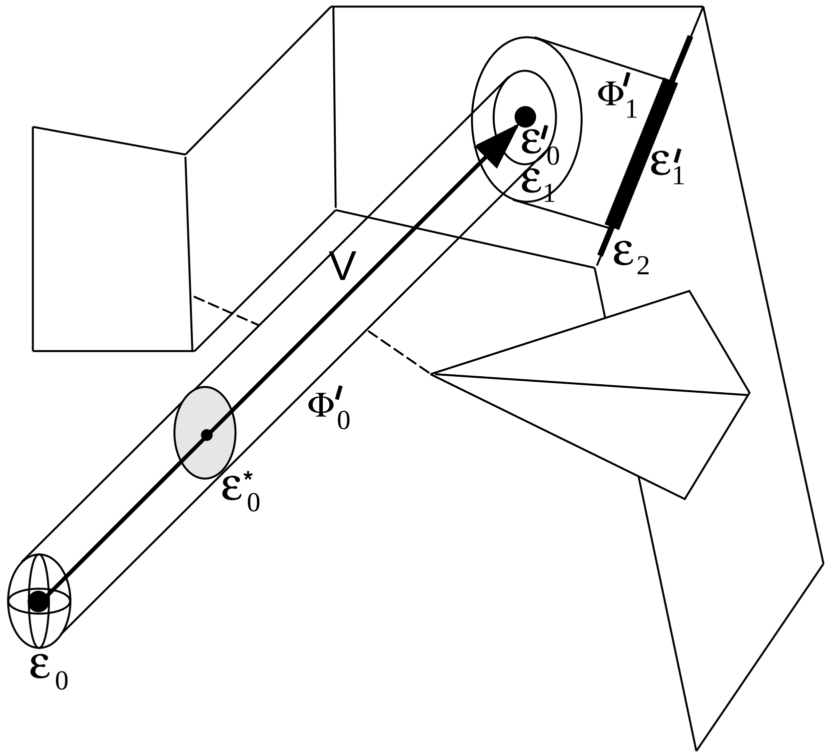


Figure 52: Cylindrical approximations of conic projections.

proximity to obstacles. In Figure 52, the lateral expansion due to velocity uncertainty is finally acknowledged when recovering the dimensions of the ellipsoid projection ϵ_1.

The cylindrical approximation of a forward-projection is represented by its cross-section ϵ^*, which sweeps out the cylindrical volume Φ' along the command velocity. The cylindrical volume Φ_0' in Figure 52, for example, is swept out by cross-section ϵ_0^*,not to be confused with ϵ_0's nonperpendicular projection ϵ_0' onto the goal state. An ellipsoidal cross-section ϵ^* is computed as follows.

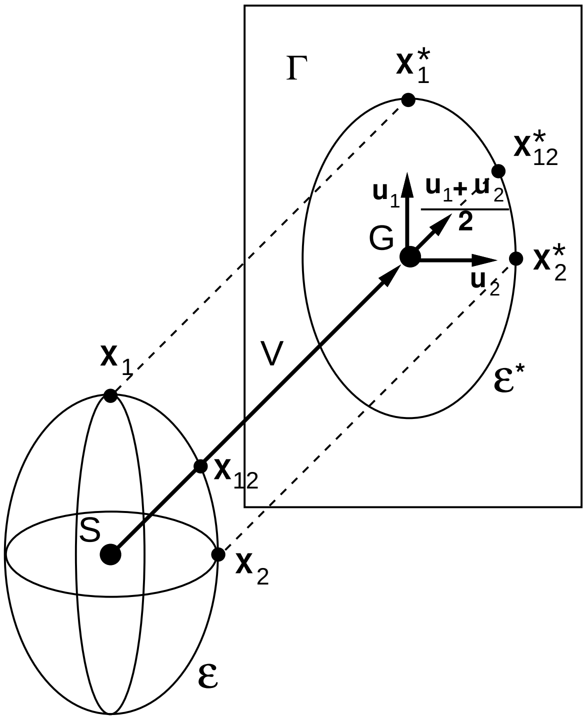


Figure 53: Computing the cross-section of a forward-projection.

We consider the case of an $(n+1)$-dimensional ellipsoid ϵ, projected through an $(n+1)$-dimensional contact state, along command velocity $\mathbf{V}$. As shown in Figure 53, the cross-section is obtained by projecting the ellipsoid onto an affine space Γ which is perpendicular to the command velocity. Γ is defined by the goal point $\mathbf{G}$ and a basis $\mathbf{U}_1 \ldots \mathbf{U}_n$ for the subspace of the traversed state's vector space which is orthogonal to $\mathbf{V}$. The projection onto Γ is accomplished by projecting surface points of ϵ onto Γ and recovering the ellipse determined by the projected points. Defined in $\mathbf{U}_1 \ldots \mathbf{U}_n$-space, the projected points $\mathbf{X}_i^*$ are expressed as displacements relative to the goal point, which serves as ϵ^*'s centroid. Recovery of ϵ''s surface equation $\mathbf{X}^T A^* \mathbf{X} = 1$ will require $n + \binom{n}{2}$ coefficients in its symmetric descriptor matrix A^*, so we project surface points corresponding to lateral directions $\mathbf{U}_1, \ldots, \mathbf{U}_n, (\mathbf{U}_i + \mathbf{U}_j)/2$, $(i, j \in 1 \ldots n, i \neq j)$. In the example of Figure 53, where $n = 2$, the directions are just $\mathbf{U}_1$, $\mathbf{U}_2$, and $(\mathbf{U}_1 + \mathbf{U}_1)/2$. The surface points $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_{12}$, of ϵ corresponding to these three directions may be computed as shown in Appendix E. The projected points $\mathbf{X}_1^*, \mathbf{X}_2^*, \mathbf{X}_{12}^*$ parameterize three equations in the three projected ellipsoid coefficients:

$$\begin{array}{rcl} \mathbf{X}_1^{*T} A^* \mathbf{X}_1^* & = & 1 \\ \mathbf{X}_2^{*T} A^* \mathbf{X}_2^* & = & 1 \\ {\mathbf{X}_{12}^*}^T A^* \mathbf{X}_{12}^* & = & 1 \end{array} \tag{40}$$

In general n-space, the solution of $n + \binom{n}{2}$ equations in as many unknowns yields the coefficients in an $n \times n$ matrix A^*. Finally, A^* is diagonalized to obtain the principle axes $\mathbf{B}_i^*$ and corresponding radii (expressed as eigenvalues $1/r_i^{*2}$) of the ellipsoidal cross-section defined in $\mathbf{U}_1 \ldots \mathbf{U}_n$-space: $A^* = B^* D^* B^{*T}$. As required in chapter 4.6.4 we define a $6 \times n$ matrix C^* whose columns hold $\mathbf{U}_1 \ldots \mathbf{U}_n$, so that points defined in $\mathbf{U}_1 \ldots \mathbf{U}_n$-space, such as $\mathbf{X}$ satisfying ellipsoid equation $\mathbf{X}^T A^* \mathbf{X} = 1$, may be transformed to global point $C^* \mathbf{X} \in \Re^6$.

4.6.4 Trajectory refinement

Trajectory refinement with ellipsoidal uncertainty regions proceeds as described in chapter 4.4, except that the forward-projection's clearance, instead of the trajectory's, is iteratively maximized. To facilitate the distance measurements, a linear transformation $T' : \Re^6 \to \Re^6$ is applied to contact space to make the ellipsoidal cross-section ϵ^* of the cylindrical forward projection Φ' spherical (see Figure 54). To achieve uniformity of ϵ^*'s radii $r_1 \ldots r_n$, we stretch contact space along each of ϵ^*'s major axes, making the scaled radii match the largest radius r_i. Specifically, contact space is scaled along the jth

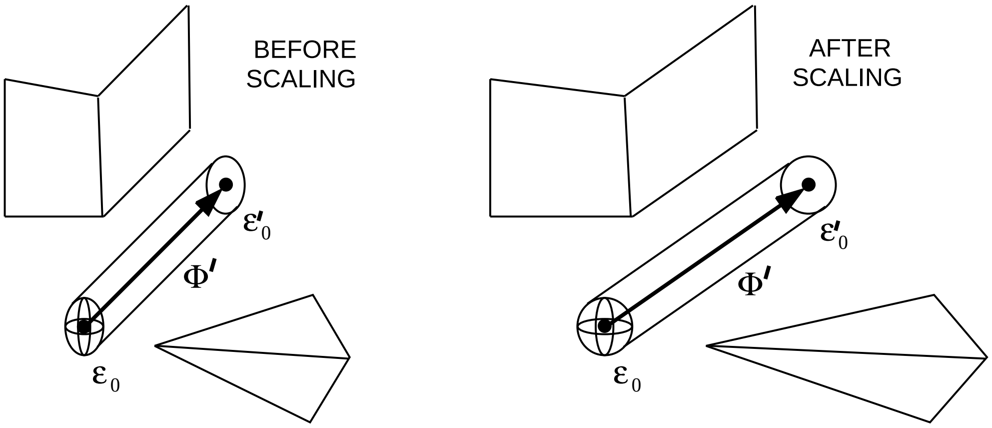


Figure 54: Scaling contact space.

major axis by a factor of r_i/r_j, for $j = 1 \ldots n$. Since each major axis $\mathbf{B}_j^*$ is defined in $\mathbf{U}_1 \ldots \mathbf{U}_n$-space, it must first be transformed to the global frame. Contact space may be scaled along the global axes $C^* B_j^*$ by the scaling factor r_i/r_j, as described in chapter 4.2 Once the forward-projection's cross-section is uniformly spherical with radius r_i, its distance to an obstacle PC is just $d - r_i$, where d is just the trajectory segment's distance to the obstacle PC, as calculated in chapter 4.3.3). For distance measurements to the goal point $\mathbf{G}$, the proximity vector $\mathbf{W}$ is generally nonperpendicular to the velocity $\mathbf{V}$, so the the measurement is modified as $d - r_i/(1 - \mathbf{V} \cdot \mathbf{W})$.

4.6.5 Recovering successive uncertainty ellipsoids

The uncertainty ellipsoid surrounding each trajectory point must be projected onto the affine space of the subsequent contact state, where it will, in turn, influence the refinement of the subsequent trajectory segment. As shown in Figure 55, however, a conic forward-projection onto an affine space A yields a projected region whose centroid $\mathbf{G}'$ does not coincide with the projected image $\mathbf{G}$ of the cone's apex $\mathbf{S}$. In order to retain the goal point $\mathbf{G}$ of a trajectory as the centroid of the projected region and avoid a more complicated recovery of uncertainty ellipsoids, we approximate the ellipsoidal projection as follows. The procedure used in chapter 4.6.3 to compute a cylindrical forward-projection's cross-section is modified here to project the cross-section onto the affine space of the goal state. After recovering the resulting ellipsoid, the radii along its principle axes are increased to represent the lateral expansion arising from velocity uncertainty.

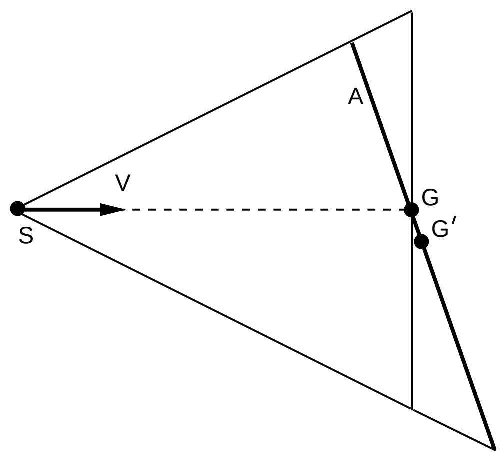


Figure 55: Migration of a conic projection's centroid.

As in chapter 4.6.3, we consider the projection of an $(n+1)$-dimensional ellipsoid ϵ through an $(n+1)$-dimensional contact state. As illustrated in Figure 56, for example, an ellipsoid ϵ_0 surrounding the starting point $\mathbf{S}$ of a trajectory segment is projected onto the affine space of the next contact state in the trajectory, yielding an ellipsoid ϵ_0'. The affine space of the goal state is defined by the goal point $\mathbf{G}$ and a vector space whose basis is contained in the columns of a $6 \times n$ matrix C'. The same surface points $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_{12}$, of ϵ are projected as in chapter 4.6.3 Their projections $\mathbf{X}_1', \mathbf{X}_2', \mathbf{X}_{12}'$ are expressed in

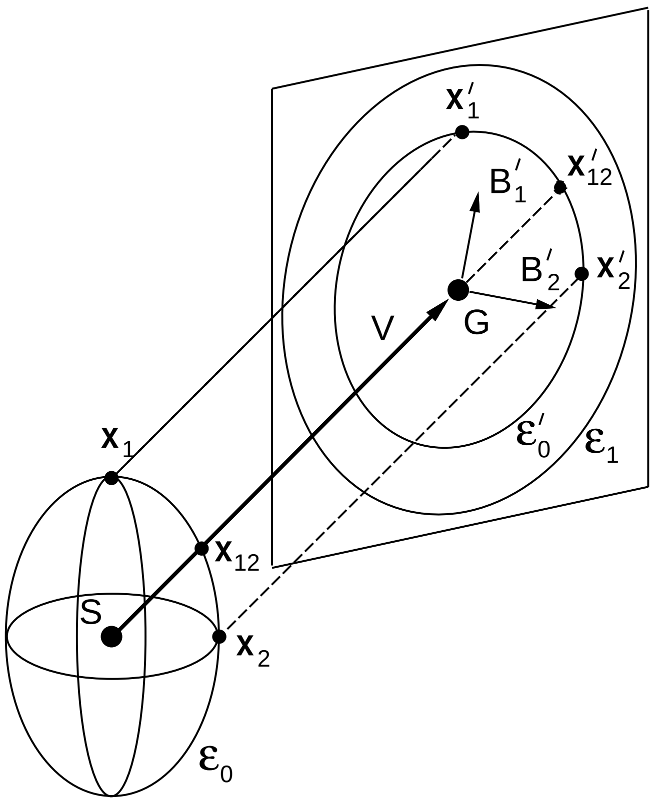


Figure 56: Recovering the projected uncertainty region.

C''s column-space as displacements relative to the goal point. The projected points parameterize three equations in the three coefficients of the ellipse ϵ_0':

$$\begin{array}{rcl} \mathbf{X}_1'^T A' \mathbf{X}_1' & = & 1 \\ \mathbf{X}_2'^T A' \mathbf{X}_2' & = & 1 \\ \mathbf{X}_{12}'^T A' \mathbf{X}_{12}' & = & 1 \end{array} \tag{41}$$

The diagonalization of $A' = B'D'B'^T$ yields principle axes $\mathbf{B}_1'$ and $\mathbf{B}_2'$, whose radii are extracted from eigenvalues $1/{r_1'}^2$ and $1/{r_2'}^2$. As in chapter 4.6.3, this procedure generalizes to the case of an n-dimensional goal state, where $n + \binom{n}{2}$ goal projected points are required to determine the $n + \binom{n}{2}$ coefficients of symmetric matrix A'.

Finally, as shown in Figure 56, the effects of velocity uncertainty are represented by enlarging the projected uncertainty region along its major axes. The lateral deviation vector $\mathbf{U}_i' \perp \mathbf{V}$ corresponding to major axis $\mathbf{B}_i$ is obtained as

$$\mathbf{U}_i' \;=\; C\,\mathbf{B}_i' \;-\; (C\,\mathbf{B}_i' \cdot \mathbf{V})\,\mathbf{V} \tag{42}$$

and the velocity cone expands laterally along $\mathbf{U}_i'$ at a rate of $\nu(\mathbf{U}_i', \mathbf{V})$. The total lateral expansion along $\mathbf{U}_i'$ throughout the trajectory segment is therefore

$$v_i \;=\; \nu(\mathbf{U}_i', \mathbf{V}) \cdot |\mathbf{G} - \mathbf{S}| \tag{43}$$

and the radius r_i along principle axis $\mathbf{B}_i'$ is accordingly increased by

$$\Delta r_i \;=\; \frac{v_i}{1 - |C\,\mathbf{B}_i' \cdot \mathbf{V}|} \tag{44}$$

Note that the typical nonperpendicularity of the major axis and the command velocity is accommodated in equation 44, allowing the projected ellipsoid to elongate along nonorthogonal projection planes.

CHAPTER 5

IMPLEMENTATION AND ANALYSIS

The fine-motion planning procedures described in this thesis were implemented in Pop-11, Prolog, and C. Fine-motion plans generated by the planner for a square-peg-in-hole assembly were executed by a Zebra-Zero robot. This chapter summarizes the software implementation and robotic assembly experiments. A complexity analysis of the planning algorithms is also presented, together with the actual run-times associated with the various phases of the planning process.

5.1 Software implementation

The procedures described in chapters 2, 3, and 4 for constructing a contact space adjacency graph, and heuristically searching the graph to yield a fine-motion plan, were implemented in POPLOG, a multilanguage AI programming environment running on a Sun-4 workstation. High-level, symbolic computations are performed in Prolog and Pop-11, while low-level, numerical computations are executed in C. A Prolog interface to the geometric solid modeller ACIS parses CSG descriptions of assembly part models, which are translated into ACIS input format. Boundary representations of the models provided by ACIS are then reconstructed in Pop-11 and represented as adjacency graphs of surface features. The analysis of, and planning within, the local contact space corresponding to any specified critical point configuration is driven by Pop-11 programs, which invoke externally loaded C-routines to perform the computationally intense number crunching procedures. Fine-motion planning within a local contact space is performed using the basic trajectory refinement process described in chapters 4.1 through 4.5. The more rigorous uncertainty analysis of chapter 4.6 is only partly implemented.

5.1.1 Modeller interface

The interface to the solid modeller ACIS enables the user to define solids in terms of Boolean combinations of primitive bodies, such as blocks and cylinders, as well as user-defined solids. The Boolean syntax is similar to the syntax used by the modeller Robmod [25]. The following Prolog statement, for example, defines a square peg and a square hole, expressing the latter as a set difference involving a user-defined peg:

```
:- acis_set_solid(sqpeg,block(10,10,60)),
   acis_set_solid(sqhole,(block(60,60,60) \ (peg)).
```

After parsing the Boolean expression representing a body, the modeller interface converts the CSG description from Robmod syntax to ACIS format. The boundary representation computed by ACIS is accessed from Pop-11 by invoking procedure MakeSolid( ). This procedure constructs an adjacency graph of faces, edges, and vertices from the adjacency data provided by ACIS.

5.1.2 Pop-11 software

Once the Pop-11 representation of a two-part assembly is obtained, the construction of local contact spaces and the planning of fine-motions within them is accomplished by invoking high-level Pop-11 procedures. Those procedures, together with their supporting subroutines, are described below. The user may select various combinations of artificial C-space limits in order to eliminate different sets of primitive contacts from consideration during the construction of a local contact space. As described in chapter 2.4.3, it is generally useful to impose artificial limits to reduce the computation time required to enumerate the contact space vertices. On the other hand, the lack of contact space vertices in a local contact space may dictate the need for user-supplied, artificial C-surfaces (see chapter 2.4.4), whose presence will give rise to the "virtual" contact space vertices from which the "real" facets of contact space are generated. The user also influences the heuristic search of contact space, by supplying the sets of starting states and goal states. In principle, the starting and goal states should be among the lower-dimensional contact states found in the local contact space adjacent to the local contact space in question, or the primitive contact states directly accessible from free space. The most relevant, high-level procedures are outlined here:

- **SetCLimits**(CLimitList) establishes artificial limits upon the rotational component of contact space. A set of rotational axes is supplied in *CLimitList*. Each axis **V** represents a constraint imposing strictly negative rotation about **V**.

- **PrimitiveContacts**(MPart,SPart,CriticalPoint) returns a list of *contact* records, each containing a pair of features from moving part and stationary part boundary representations *MPart* and *SPart*. Each *contact* record represents the primitive contact that can form when the moving part is near location *CriticalPoint*, while subject to the pose constraints in SetCLimits.

- **VirtualPrimitiveContacts**(VPCList) augments the global list of *contact* records with a set of “virtual” PCs, represented in *VPCList* as hyperplanes in 6–space.

- **MakeVertices**(PCList) returns a list of *cstate* records representing the 0–dimensional contact states formed from 6–tuples of the *contact* records in *PCList*.

- **MakeCSpace**(VList) returns an adjacency graph of *cstate* records, representing the multidimensional C-states formed from subtuples of the contacts associated with each contact space vertex in *VList*.

- **PurifyCSpace**(AGraph) removes all the “virtual” C-states from adjacency graph *AGraph* and returns the leaner graph with updated adjacencies.

- **MakeMotion**(PrevMotion,TState,GState) returns a *motion* record *Motion*, representing a contact state transition from the goal state of motion *PrevMotion*, through traversed state *TState*, and ending in goal state *GState*. A linear trajectory segment is defined in 6–space by endpoints *StartPose(Motion)* and *GoalPose(Motion)*. *StartPose(Motion)* receives the value of *GoalPose(PrevMotion)*, and *GoalPose(Motion)* is arbitrary.

- **CostMotion**(Motion) evaluates the heuristic cost of contact state transition *Motion* according to criteria which include the distance from the trajectory segment defined by *StartPose(Motion)* and *GoalPose(Motion)* to the “obstacle” PCs adjacent to the traversed states and goal states of *Motion*.

- **OptimizeMotion**(Motion) minimizes *CostMotion(Motion)* in a gradient descent process which perturbs *GoalPose(Motion)* along the surface of *Motion*'s goal state. When an apparent minimum is reached, the perturbation length is iteratively divided by scalar EpsRefineFactor until descent is again possible at a finer scale, or until a desired precision is attained. In the absence of apparent minima, the perturbation length is increased by the same factor.
- **BestFirstSearch**(AGraph,SList,GList) finds a least-cost path through adjacency graph *AGraph*, from a state in *SList* to a state in *GList*. Search paths are extended via *MakeMotion*, and *OptimizeMotion* refines the trajectory segments of candidate motions prior to their evaluation by *CostMotion*.

5.1.3 Affine space computations

The Pop-11 routines described above manipulate affine spaces defined in $\Re^3$ and $\Re^6$. To support the numerical treatment of assembly part features in $\Re^3$ and contact space facets in $\Re^6$, a facility was developed in C for treating general affine spaces in $\Re^n$. The facility supports several levels of computation, from vector and matrix arithmetic at the lowest levels, to vector space and affine space operations at the highest levels. Each affine space in $\Re^n$ is characterized as a point plus a vector space, which are represented by a vector and a matrix, respectively. The matrix holds an orthonormal basis for the vector space. Orthonormal bases are calculated through the Gram-Schmidt procedure. Affine space operations include *intersection*, which returns the intersection of two affine spaces; *join*, which returns the smallest affine space containing two given affine spaces; and *distance*, returning the shortest distance between two affine spaces, as described in Appendix D. Distance measurements are employed throughout the trajectory refinement process of chapter 3.3.2, whereby a linear trajectory segment's clearance with respect to the linearized facets of obstacle PCs is iteratively maximized.

Situated hierarchically above the affine space facility and below the Pop-11 command level is a package of C routines devoted to primitive contact computations. Serving as an intermediary between the Pop-11 software and the lower-level number crunching routines, this package is primarily concerned with numerical computations in 6–dimensional contact space. The routines provided in this package include operations for evaluating PC functions (equation 4), constructing C-surface tangents (equations 6 ... 10), evaluating PC point set predicates (chapter 2.3.7), computing the point where 6 PC surfaces intersect in $\Re^6$ (chapter 2.4.1), approximating obstacle PC facets as affine spaces (chapter 2.5.2), and computing the distance between a linear trajectory segment and a linearized obstacle PC facet (chapters 4.1.2 and 4.3.1).

5.2 Complexity analysis

The time-complexity of our fine-motion planning method is now analyzed, in order to compare its efficiency to that of other approaches. Worst-case time-requirements are derived for enumerating the PCs adjacent to a critical point, constructing a local contact space surrounding a critical point, and planning a fine-motion trajectory within a local contact space. The latter analysis concerns the heuristic search of a contact state adjacency graph and the iterative optimization of a trajectory segment's obstacle clearance.

5.2.1 Primitive contact enumeration

For the purpose of analyzing the computation time required to enumerate the PCs and bounding contacts associated with a critical point, let f, e, and v denote the maximum numbers of faces, edges, and vertices on the two assembly part models. Additionally, let e_f denote the maximum number of edges bordering a model face. In the procedure for enumerating PCs presented in chapter 2.3.5, the vertex-plane contacts are identified by examining each distinct vertex-face pair (V,F) in the respective models. Further, the identification of bounding contacts entails examining every edge of F. Hence, the time requirement for enumerating vertex-plane PCs and bounding contacts is $\mathcal{O}(vfe_f)$. Enumeration of plane-vertex PCs and bounding contacts is similarly $\mathcal{O}(vfe_f)$. Line-line contacts are identified by examining each distinct pair (E_1,E_2) of edges in the respective models. The bounding contacts are enumerated by examining the vertices of E_1 and E_2, of which there is a constant number (4), so the time requirement for finding the line-line PCs and bounding contacts is $\mathcal{O}(e^2)$. In summary, the computation time required for enumerating all PCs and bounding contacts around a critical point is $\mathcal{O}(vfe_f + e^2)$. A similar analysis reveals that the number of PCs is $\mathcal{O}(vf + e^2)$, though this worst case is atypically large.

5.2.2 Contact space construction

The construction of a contact state adjacency graph involves enumerating the vertices (0–dimensional facets) of contact space, which in turn give rise to all the higher-dimensional contact states. As described in chapter 2.4, the enumeration of contact space vertices is performed by generating and testing all 6-tuples of PCs and virtual PCs. Given a total of n PCs and virtual PCs, there are some

$$\binom{n}{6} = \frac{n!}{(n-6)!\,6!} \tag{45}$$

6-tuples, numbering somewhat less than n^6. Vertex generation therefore requires $\mathcal{O}(n^6)$ computation time.

Let N denote the number of contact states in a local contact space. It can be shown that

$$\binom{n}{i} < \binom{n}{6} \tag{46}$$

where

$$n \geq 10, \quad 0 \leq i \leq 5$$

Noting that

$$N \leq \sum_{i=0}^{5} \binom{n}{i} \tag{47}$$

it follows that $N \sim \mathcal{O}(n^6)$. The procedure described in chapter 2.6.1 for constructing an adjacency graph generates the contact states by enumerating all of the subtuples of PCs associated with successively higher-dimensional contact states' PC tuples. Since each subtuple may be "spawned" by up to n parent tuples, this procedure processes no more than $nN \sim \mathcal{O}(n^7)$ PC tuples. Each PC tuple is placed in the graph as a contact state, unless it is already present in the graph. To facilitate the storage and lookup of existing contact states, a 6–level tree is maintained, representing contact states as sequences of PC identifiers. Since no more than 6 levels of search are necessary to locate the pointer to an existing contact state in the adjacency graph, the retrieval of an existing contact state is performed in constant time. The process of adjacency graph construction in chapter 2.6.1 is therefore computationally comparable to the enumeration of contact space vertices, requiring $\mathcal{O}(n^7)$ and $\mathcal{O}(n^6)$ time, respectively. In practice, however, the discarding of spurious contact space vertices described in chapter 2.4.2 yields an adjacency graph whose size N is typically quite small compared to the number of PC 6–tuples enumerated for the generation of contact space vertices. The time complexity of practical importance is therefore that of vertex enumeration, which requires $\mathcal{O}(n^6)$ time in the number of PCs.

The algorithm for adjacency graph construction described in chapter 2.6.2 is somewhat more complex. The first phase of the algorithm, in which some N' contact space facets are enumerated, is analogous to the algorithm of chapter 2.6.1, which enumerates N contact states. The upper bound on N in equation 47 holds true for N', so $N' \sim \mathcal{O}(n^6)$. Whereas the algorithm of chapter 2.6.1 checks for the equivalence of contact states in constant time by comparing their PC tuples, the algorithm of chapter 2.6.2 tests for spatial equality of facets in constant time, using procedure EqualFacet. Moreover, each facet may be "spawned" as a PC subtuple of up to n parent tuples, so phase one has an overall time requirement of $\mathcal{O}(n^7)$.

In the second phase, each facet is partitioned into distinct contact states, using procedure Feasible to test each m–tuple of PCs associated with each facet of dimension $i+1$, where $m \geq 5-i$. Let r be the maximum number of "redundant" PCs associated with a facet, so that $5-i \leq m \leq 5-i+r$. The case of $i=0$ concerns the 1–dimensional edges of contact space, where tuples of size $5 \leq m \leq 5+r$ are selected from a set of $5+r$ PCs. There are $\mathcal{O}(r^5)$ such tuples, so each 1–dimensional facet requires $\mathcal{O}(r^5)$ feasibility tests. Each 2–dimensional facet similarly requires $\mathcal{O}(r^4)$ feasibility tests, fewer than the number required by a 1-dimensional facet. The higher-dimensional facets generally involve fewer feasibility tests than the 1-dimensional facets, so we may conclude that each facet generated in phase 1 requires $\mathcal{O}(r^5)$ processing time for the enumeration of its constituent contact states. Finally, the establishment of adjacency arcs between sibling states is comparatively trivial, requiring the enumeration of all pairs of contact states within each facet. In conclusion, the time requirement of the more complex adjacency graph construction is $\mathcal{O}(n^7 r^5)$.

5.2.3 Optimization of trajectory goal poses

In the trajectory refinement procedure of chapter 4.4, the goal pose of a trajectory segment is perturbed along the surface of the segment's goal facet, iteratively increasing the segment's clearance with respect to obstacle PCs. The segment's proximity to the obstacle PCs is embodied in function $cost_P$ (see equation (30)), whose minimization is achieved through the process of gradient descent. In the implementation described in chapter 5.1.2, the length of the perturbations varies, decreasing exponentially in the presence of apparent minima until a desired precision $\epsilon_{r_{min}}$ is attained. In the absence of apparent minima, the perturbation length increases exponentially, hastening the migration of the goal pose across the goal facet's surface. If the extent of the local contact space is bounded in all directions by distance d_{max}, then the number of calls to $cost_P$ is

$$\mathcal{O}(log\frac{d_{max}}{\epsilon_{r_{min}}}) \tag{48}$$

Measurement of the trajectory's distance to each obstacle PC is performed by function $Dist$ of equation (29), which contains a small, bounded number of distance and nearest point calculations. Appendix D describes a procedure which calculates the distance between two affine spaces, and their closest points, by solving a 6×6 system of equations. Such a computation is performed in constant time. Moreover, equation $cost_P$ calls function $Dist$ no more than k_0 times, where k_0 is the maximum number of obstacle PCs, so $cost_P$ is computed in $\mathcal{O}(k_0)$ time. Hence, from equation 48, a trajectory segment is optimized in time

$$\mathcal{O}(k_0 \, log \frac{d_{max}}{\epsilon_{r_{min}}}) \tag{49}$$

5.2.4 Heuristic search of contact space

The best-first search for a path through a contact state adjacency graph generates a search tree with a branching factor bounded by k, the maximum number of contact states adjacent to any given state. When viewed as an exhaustive search, the generation of a fine-motion path consisting of p steps requires an exponential computation time of $\mathcal{O}(k^p)$. In practice, of course, this worst case time is quite rare, since the search is a heuristic one.

The heuristic search evaluation function *cost* of equation (21) in chapter 3.1.3 is comprised of a weighted sum of sub-costs representing several criteria by which to rate a candidate path in the search tree. The only sub-cost that is computationally nontrivial is $cost_P$, whose time requirement appears above, in equation (49). If $cost_P$ were evaluated at each step of an exhaustive search, the computation time would be

$$\mathcal{O}(k^p \, k_0 \, log \frac{d_{max}}{\epsilon_{r_{min}}}) \tag{50}$$

or noting that $k_0 \leq k$,

$$\mathcal{O}(k^{p+1} \, log \frac{d_{max}}{\epsilon_{r_{min}}}) \tag{51}$$

Hence, a fine-motion trajectory may be planned within a local contact space in exponential time in the number of plan-steps p. Since p is seldom large, typically less a dozen, even an exhaustive search would not be intractable. Moreover, the search is constrained to follow the best current path, and the distance (number of nodes) to the goal state is never increased, so the worst-case time is quite rare.

Chapter 4.3.3 discusses an alternative approach to measuring trajectory clearance, which is less rigorous but results in a significantly more efficient heuristic search. In this simpler, but less rigorous approach, the clearance of the trajectory segment's goal pose, but not of the segment's interior, is iteratively optimized. In such a scheme, function $cost_P$ is only optimized once for each contact state, though each contact state encountered during the heuristic search is usually shared by numerous paths in the search tree. Since the bulk of the computation time is spent performing pose optimizations which only need to be performed once per contact state in the search tree, the time requirement for this alternative fine-motion planning technique would be linear in the number of contact states, and hence, polynomial in the number of PCs. Noting the time requirement for trajectory segment optimization in Equation (49), the reduced search time becomes

$$\mathcal{O}(k_0\, n^6\, log \frac{d_{max}}{\epsilon_{r_{min}}}) \tag{52}$$

5.2.5 Complexity summary

The complexity analysis above reveals that the construction of a local contact space requires polynomial time in the number of PCs. Moverover, there are $\mathcal{O}(vf\, +\, e^2)$ PCs, so the construction time is polynomial in the number of PCs. Fine-motion planning within an adjacency graph representation of a local contact space generally takes exponential time in the number of plan-steps, i.e., the number of contact states in the resulting plan. Since the number of plan-steps is seldom large, e.g., less than a dozen, even an exhaustive search of contact space is fairly tractable. The complexity of this technique compares favorably with the double-exponential complexity of the LMT approach [97].

5.3 Software run-time experiments

The fine-motion planner described in chapter 5.1 has been tested with respect to various assembly paradigms, including a square-peg-in-hole assembly, and a hexagonal peg-in-hole assembly. Fine-motion planning run-time results for these two assemblies are presented here, along with the associated run-times. Both of these assemblies have two critical points, one aligning the base of the peg with the hole rim, and the other placing the peg entirely in the hole. The former critical point is the only one of interest, and its associated contact space was constructed by the fine-motion planner.

Up to three artificial C-space limits were employed with regard to the square-peg-in-hole assembly, enforcing strictly negative rotations about axes corresponding to the three edge directions on the peg's surface. Up to four limits were imposed for the hexagonal-peg-in-hole assembly, one for each edge direction on the peg's surface. As one may observe in the run-time results of chapter 5.3.1, the reduced number of PC 6–tuples associated with the use of artificial C-limits illustrates the advantage in employing such constraints. The run-time results for fine-motion planning in chapter 5.3.2 were all obtained using the maximum number of C-space limits, primarily because the procedures for optimizing trajectory goal poses were not implemented to accommodate poses that place contacting edges parallel to one another.

5.3.1 Contact space construction

The CPU time requirements for constructing local contact spaces are shown here for the two assemblies, using two different insertion clearances and varying numbers of artificial C-space limits. In all instances, three virtual PCs were included with the existing PCs, forming hyperplanes in 6–space associated with differential rotations of -0.001 radians about the x, y, and z axes. As explained in chapter 2.4.4, the addition of virtual PCs is necessary prior to the enumeration of contact space vertices, if the contact space in question is actually devoid of vertices. The vertex totals shown below therefore include virtual vertices, whose 6–tuples of PCs include virtual PCs. After the contact state adjacency graph is generated from the virtual vertices, the planner removes all virtual contact states involving virtual PCs.

Table 1. Square-peg-in-hole with 0.01 mm. clearance.

		Vertex Enumeration			Graph Construction	
C-Limits	PCs	6–Tuples	Vertices	Seconds	C-States	Seconds
3	12	924	51	5	637	1
2	18	18564	261	134	2793	8
1	27	296010	1956	2015	18365	274

Table 2. Square-peg-in-hole with 0.0793 mm. clearance.

		Vertex Enumeration			Graph Construction	
C-Limits	PCs	6–Tuples	Vertices	Seconds	C-States	Seconds
3	12	924	31	5	525	1
2	18	18564	198	122	1939	6
1	27	296010	1477	1986	12287	228

Table 3. Hexagonal-peg-in-hole with 0.01 mm. clearance.

		Vertex Enumeration			Graph Construction	
C-Limits	PCs	6–Tuples	Vertices	Seconds	C-States	Seconds
4	17	12376	105	69	1497	3
3	27	296010	755	1804	12792	271

Table 4. Hexagonal-peg-in-hole with 0.0793 mm. clearance.

		Vertex Enumeration			Graph Construction	
C-Limits	PCs	6–Tuples	Vertices	Seconds	C-States	Seconds
4	17	12376	130	74	1877	3
3	27	296010	1204	1832	17835	384

5.3.2 Fine-motion planning

The CPU time requirements for synthesizing a fine-motion trajectory in a local contact space are shown here for the square- and hexagonal-peg-in-hole assemblies. The contact spaces were constructed using 3 and 4 C-space limits respectively, with two different insertion clearances. Run-time results were tabulated for three different heuristic searches of contact space, using three different starting states and always the same goal state. The alternative starting states were PC states directly accessible from free space, and the goal state was a two-dimensional contact state, shared by the other critical point's local contact space. Several best-first search run-times are included in the tables below, along with the number of contact state visitations (nodes in the search tree), the number of distinct contact states visited, and the number of segments (contact state traversals) in the resulting trajectory.

Trajectory refinement was performed using the specified ratio $\sigma_r = 50$ mm/rad between the expected rotational and translational uncertainty. This relationship reflects the observed behavior of the Zebra-Zero robot with which the fine-motion plans were tested. As described in chapter 4.2, trajectory refinement was carried out in a contact space whose dimensions were stretched along the rotational axes by a factor of $\sigma_r = 50$. Goal pose optimization was performed to a specified precision of $\epsilon_{r_{min}} = 0.001$ within the stretched C-space, yielding a precision of 0.001 millimeters along translational axes and 0.00002 radians along rotational axes. Using these refinement parameters, the goal pose optimization times averaged around 0.5 seconds apiece.

Table 5. Square-peg-in-hole with 0.01 mm. clearance.

Start State	C-State Visits	C-States Visited	Segments	Seconds
S_3	9	8	5	4
S_7	21	16	6	10
S_{15}	22	19	6	10

Table 6. Square-peg-in-hole with 0.0793 mm. clearance.

Start State	C-State Visits	C-States Visited	Segments	Seconds
S_3	8	8	5	4
S_7	45	30	6	21
S_{15}	21	16	6	9

Table 7. Hexagonal-peg-in-hole with 0.01 mm. clearance.

Start State	C-State Visits	C-States Visited	Segments	Seconds
S_{10}	33	25	7	18
S_{14}	36	32	7	19
S_{16}	27	25	7	15

Table 8. Hexagonal-peg-in-hole with 0.0793 mm. clearance.

Start State	C-State Visits	C-States Visited	Segments	Seconds
S_{10}	24	22	7	13
S_{14}	28	26	7	15
S_{16}	21	20	7	12

5.4 Numerical precision

We have observed that the numerical calculation of contact space vertices via the multivariable Newton method is subject to failure with respect to vertices that lie close to singularities in contact space. Since the technique of chapter 2.4.1 derives vertex points in 6–space by solving 6×6 systems of equations, the numerical precision of the calculated point is compromised when the system is nearly singular. In other words, the position of a vertex formed where 6 PC surfaces meet cannot be computed precisely when the 6 C-surface normals are barely independent. Consequently, a legitimate vertex may appear to lie in the forbidden zone, or outside of its PCs' point sets. Conversely, a spurious vertex may appear to be legitimate. The omission of legitimate vertices and inclusion of spurious vertices can lead to the construction of an adjacency graph with extra or missing contact states. Unfortunately, the occurence of contact space vertices near singularities is not rare. The numerical procedure for computing vertex points therefore needs to be enhanced to handle nearly-singular configurations with greater precision.

5.5 Robotic assembly testbed

Fine-motion plans derived for the square-peg-in-hole assembly were tested using a Zebra-Zero robot driven by an IBM P.C., with a peg-in-hole apparatus designed to provide variable insertion clearances. The P.C. contained a small library of robot commands for executing motions and sensor readings, and these commands were invoked by a Pop-11 program running on a Sun-4 workstation. Since the P.C.'s library did not include hybrid twist/wrench control commands or a facility for specifying stiffnesses in 6 dimensions, the individual fine-motion segments were executed in position control mode. Forces and torques obtained by the robot's wrist-mounted force sensor were processed to indicate termination conditions. A relatively large insertion clearance of $\frac{1}{32}$ inch (0.0793 mm.) was used for all fine-motion experiments, so the typical hazards of jamming and wedging were avoided, even in the absence of explicit force control. The worn state of the robot gears contributed to very significant backlash error, giving rise to peg translational errors of up to two millimeters. After partially compensating for gear backlash, the ratio between expected translational and rotational uncertainties associated with peg movements was estimated to be $\sigma_r = 50$ mm/rad, providing the C-space stretching factor required for fine-motion planning.

A visual servoing system was developed in Pop-11 and C to facilitate the initial positioning of the peg at the starting pose of the fine-motion trajectory. A pair of CDC cameras provided two 512×512 gray-scale images, from which an edge-tracking facility extracted the line parameters of specified peg edges. A pose Jacobian was formed by the visual servo program to map translational and rotational peg displacements to edge parameter displacements. Observed edge parameter errors were then mapped to corrective peg displacements by the method of least squares. The visual servo's precision was adequate for positioning the peg at its specified approach pose in free space, as well as goal poses within the fine-motion trajectory.

The fine-motion planner's interface to the Zebra-Zero was designed so that each segment of a fine-motion trajectory could be executed incrementally, moving through a succession of interpolated goal poses. Each fine-motion segment's incremental execution terminated upon arrival at the goal state, where the formation or separation of a single PC was detected by comparing the sensed forces and torques with the PC's characteristic wrench.

A number of peg-in-hole insertions were executed using the testbed thus described. Fine-motion plans were computed around the first critical point in the square-peg-in-hole assembly, with each trajectory terminating at a two-dimensional goal state which placed the peg's base inside the rim of the hole. From that two-dimensional state, the insertion could be completed through a single downward motion. All of the fine-motion plans were generated within an artificially-constrained contact space, constructed in 6 seconds of CPU time, and each fine-motion trajectory was calculated in no more than 21 seconds of CPU time. The trajectories included the three trajectories, originating from three different starting states, that are documented in Table 6 of chapter 5.3.2.

In an initial set of experiments, the visual servo system was employed simply to position the peg at its initial pose in free space, i.e., the starting pose of the first motion segment. For approximately one-half of such experimental insertions, the peg was successfully servoed through the specified sequence of contact states. In the remaining half, the peg encountered contact states outside of the designated sequence. The unsuccessful runs were apparently due to lateral deviations arising from backlash error. To improve positioning accuracy during the fine-motion sequence, a second set of experiments was performed in which the visual servo guided the peg through the first two fine-motion segments, servoing the peg to the interpolated poses along each segment. Visually-servoed peg displacements and peg displacements for achieving commanded bias forces were executed along orthogonal axes. With the improved positioning accuracy provided by the visual servo, roughly ninety percent of the insertions were accomplished successfully.

CHAPTER 6

EVALUATION AND CONCLUSION

This thesis has presented a new approach to fine-motion planning in contact space, whereby fine-motion trajectories may be derived efficiently within a simplified motion planning domain. Owing to a simplified construction of contact space and efficient optimization of trajectory segments, this approach has a worst-case computational time requirement that is only single-exponential in the (typically small) number of motion steps. The complexity of this approach compares favorably with that of the more rigorous, LMT method [97], which requires double-exponential time [28] in its most general form and whose implementation has been limited to 2–dimensional [51], and nonrotational, 3–dimensional assemblies [22]. As reported in chapter 5.3, the computational tractability of the technique is underscored by its ability to generate fine-motion trajectories for a square or hexagonal peg-in-hole assembly in less than 30 seconds of CPU time. Moreover, the fair success of the robotic assembly experiments in the presence of very significant gear backlash error supports the validity of this fine-motion planning technique.

6.1 Discussion

The tractability of this fine-motion planning method arises from various simplifications which are employed in the analysis of contact space and the planning of fine-motion paths. An essential simplification is the restriction of the planning domain to local contact spaces centered around the critical points in the original nominal trajectory. Since the topology of contact space does not vary between critical points, all of the relevant contact states may be identified through a limited number of local contact space analyses. The local nature of the resulting planning domains and the artificial restrictions on their extent simplifies, in turn, the process of constructing an adjacency graph of contact states. Our reliance upon the multivariable Newton method for generating contact space vertices is made possible by the small curvatures of the local C-spaces' facets, and use of artificial C-space constraints limits the number of PCs, thereby reducing the (polynomial) time required for contact space vertex enumeration.

Another significant planning simplification adopted in this thesis is the restriction of the assembly path to a determinate sequence of contact states. This policy effectively reduces path planning to a graph search of the contact state adjacency graph, resulting in the single-exponential time complexity of equation 51. The restricted extent of local contact space, and the policy that each fine-motion step must constitute an entire contact state transition, leads to a fairly small number of motion steps. The worst-case, single-exponential time requirement for path planning is therefore quite tractable, especially in light of the rarity of worst-case heuristic searches. Indeed, the runtimes reported in chapter 5.3 support the claim that this technique is practical with respect to CPU time utilization.

The robotic assembly experiments involving the square peg-in-hole apparatus were inconclusive but promising. Owing to the significant degree of backlash error displayed by our Zebra-Zero robot, the fine-motion plans were not reliably executed, though the use of the visual servo to guide the assembly through the first two contact state transitions improved the success rate to approximately ninety percent. The achievement of jam-free sliding motions, which were successfully generated in spite of our reliance on the robot's pure position control mode, supports the policy of motion planning along the C-surfaces of the assembly. Insertion experiments involving smaller clearances will be required, however, before the utility of task-consistent compliance can be fully evaluated with respect to truly challenging assembly tasks. Given the severity of backlash exibited by our Zebra-Zero, most of our attempts to execute insertions with tighter clearances have been unsuccessful.

The fine-motion planning method described in this thesis has been shown to be practical and algorithmically efficient, as evidenced by its documented implementation in chapter 5.1, and the complexity analysis and runtimes of chapters 5.2 and chapter 5.3. There remain a number of unresolved issues, however, concerning the correctness and completeness of contact state adjacency graphs, and the potential fallibility of task-corrective compliance. These concerns are summarized here:

- *Elimination of contact states via artificial C-space limits.* The construction of a contact state adjacency graph takes polynomial time in the number of PCs, so it is convenient to eliminate a number of PCs from consideration by imposing rotational limits on C-space. As a consequence of this practice, contact states associated with nonpositive (or nonnegative) tilt about various rotational axes are not present in the resulting adjacency graph. It is unclear whether the elimination of such states may sometimes handicap the heurustic selection of a reliably-traversable sequence of contact states.
- *The selection of virtual PCs.* Prior to constructing a contact state adjacency graph for a local contact space that is devoid

of vertices, a number of virtual PCs (see 2.4.4) must be added to the list of existing PCs, producing the vertices required for adjacency graph construction. As yet, there is no general procedure for generating the virtual PCs necessary to enumerate all the contact states in the (artificially constrained) local contact space.

- *Computing contact space vertices close to singularities.* The numerical calculation of contact space vertices via the multivariable Newton method can fail for vertices whose 6×6 system of equations is nearly singular (see chapter 5.4). The vertex position is calculated with imprecision, producing poses that are mistakenly judged to lie in (or not in) the forbidden zone. It will require further investigation to discover the most appropriate remedy, e.g., increasing the numerical precision or adopting a better numerical method.

- *Relying upon task-consistent compliance to prevent jamming.* The use of a hybrid wrench/twist control as an effective means of preventing jamming and wedging has yet to be validated experimentally for assemblies with small insertion clearances. If task-consistent compliance does not in itself prevent kinematic and wrench conditions associated with jamming and wedging from arising, then these conditions must be derived explicitly, as in Caine [23] or Whitney [137].

- *Calculating goal poses in the presence of local minima.* The trajectory refinement process, which minimizes the proximity function $cost_P$, may fail to yield trajectory segments that are globally optimal, due to the concavities of C-space. Although the procedure of chapter 2.7 may be invoked to partition C-space into quasi-convex regions, wherein local searches may be performed to attain the global minimum, further research will be required to determine an efficient and reliable means.

6.2 Future extensions

The demonstrated efficiency of the fine-motion planning technique described in this thesis arises from several simplifying restrictions and assumptions, including (1) the policy that the fine-motion plan should specify a *determinate* sequence of contact states through contact space, and (2) a hybrid control model is assumed, whereby a task-consistent compliance may be derived in a straightforward manner from the contact geometry and motion trajectory. While restricting the assembly to a determinate path in the contact state adjacency graph effectively reduces path-planning to a graph search, the technique could accommodate a greater amount of uncertainty if the assembly were allowed to deviate from the specified path and then recover through appropriate corrective actions. And while the relatively simple derivation of task-consistent compliances reduces fine-motion planning to spatial planning in contact space, an assembly operation would be more robust with respect to C-surface curvatures and unmodelled uncertainties if a more sophisticated, nonlinear compliance could be learned on-line. These scenarios are embodied in the following suggested offshoots of the current method.

6.2.1 Patch planning in contact space

In the *patch planning* approach to fine-motion planning, deviations of the moving part from the nominal trajectory trigger error recovery strategies, which are computed off-line [3, 48], or on-line [42, 140]. After identifying the contact state encountered, the real-time system executes the appropriate corrective action, which is derived from the contact geometry of the error state. Most patch-planning methods use the nominal path in free space, derived from kinematic constraints, as the default trajectory to be executed when not recovering from an error state. A more powerful patch planning approach would employ a more reliable default trajectory through contact space, i.e., a fine-motion trajectory whose calculation is described in thesis. By choosing a contact state path according to the criteria described in chapter 3.1, the fine-motion planner would effectively minimize the need for error-recovery. Ironically, the nominal trajectories in free space typically used by patch-planning investigators *maximize* the nondeterminacy of the assembly state, since the centering and alignment of the moving part with respect to the stationary part places the assembly pose equidistant from all contact states whose point sets in 6–space contain the origin of C-space. The fine-motion planning technique of this thesis generates a sequence of contact states whose associated error states are few in number and may be readily differentiated according to their wrench signatures. Since the author's fine-motion planner already provides (perturbed) nominal trajectories, together with the anticipated error states and corresponding wrench signatures, the only additional research required to incorporate patch-planning would concern the derivation of corrective trajectories to transport the assembly from identified error states to desired goal states.

Augmenting the present fine-motion planning approach with a patch-planning capability would yield a more powerful planner, capable of handling more geometrically complex and challenging assemblies, and/or tolerating a greater degree of uncertainty. Still requiring single-exponential CPU time, the augmented planner could handle more general assembly tasks than both the current planner and existing patch-planning techniques. This relationship among planning methods is illustrated in Figure 57, which shows a partial ordering of fine-motion planning techniques with respect to the degree of difficulty they can accommodate in an assembly task. LMT remains the most rigorous, general method, but it is unfortunately intractable for 3–dimensional assemblies with rotations. Corrective compliance [63, 111, 120, 122] is the most expedient technique, apparently requiring linear CPU time in the number of anticipated contact events (PCs), but severely limited in generality. Whereas a corrective compliance method produces a fixed, nonvarying compliant behavior, the present technique generates a sequence of such behaviors, thereby accommodating more geometrically complex assemblies.

6.2.2 On-line acquisition of compliant control laws

Another suggested offshoot of the current technique would involve merging the current, off-line planning method with the on-line learning of robust, nonlinear compliances. Concerned primarily with the spatial aspects of fine-motion planning, the current planner's modelling of contact space topology would complement the force-related capabilities of an on-line learning system based on neural networks. The fine-motion planner could be merged with the on-line, associative reinforcement technique of Gullapali [60, 61], which has

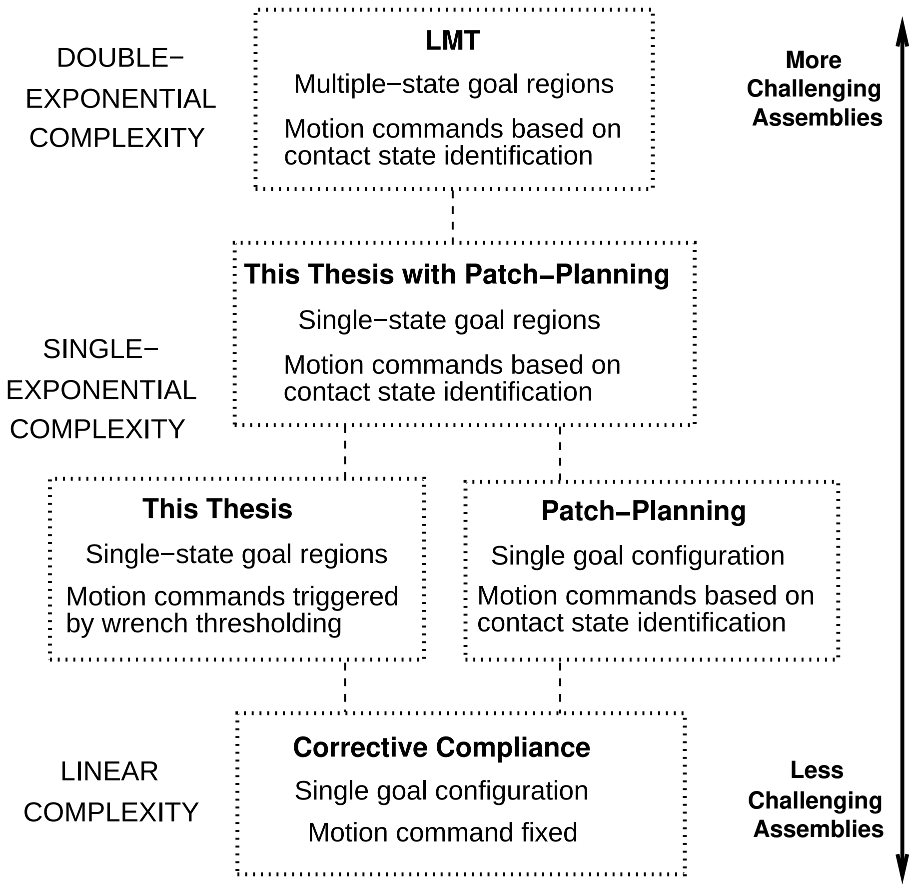


Figure 57: Generality versus efficiency in f.m. planning methods.

demonstrated jamming-avoidance skills that are difficult to replicate off-line. In Gullapali's stochastic reinforcement technique, the learning system maximizes a scalar response which quantifies the success or failure of an insertion, as well as the avoidance of excessive contact force. In the suggested fusion of methods, the scalar reinforcement signal would indicate success or failure in traversing the desired sequence of contact states without excessive force. Accommodating the nonlinearities of contact space while avoiding jamming in the presence of noise and uncertainty, the learning system would thus augment the command trajectory with a nonlinear compliant behavior.

6.3 Conclusion

A new approach to fine-motion planning was presented, whereby a constrained analysis of contact space produces a simplified motion planning domain in which fine-motion trajectories are efficiently derived. By reducing fine-motion planning to a graph search of contact space, the method achieves a worst-case time requirement that is single-exponential in the number of motion steps, which is typically small. This complexity compares favorably with that of more formal methods, which are more rigorous but harder to implement and computationally intractable. The practicality of the current method was demonstrated by implementing a fine-motion planner in Pop-11 and C. Reflecting the technique's tractability, the planner has generated fine-motion plans for square and hexagonal peg-in-hole assemblies in less than 30 seconds of CPU time.

Unresolved issues remaining to be addressed include numerical difficulties in calculating contact space vertices near contact space singularities, and the selection of the necessary virtual PCs required to generate an adequate set of contact space vertices. Robotic assembly experiments, performed without the benefit of hybrid wrench/twist control, were inconclusive, due in part to the significant amount of backlash error present. Nevertheless, square peg-in-hole operations involving a fairly large, $\frac{1}{32}$ inch clearance were fairly successful using pure position control and force-terminated motions.

APPENDIX A

CONTACT POINTS AND NORMALS

Procedures are presented here for computing the contact points $\mathbf{r}$ and $\mathbf{s}$, and the contact normal $\mathbf{n}$, for eight types of PCs. As required by the C-surface formula equations (4)...(10), the moving part contact point $\mathbf{r}$ is expressed in the moving part frame, while the stationary part contact point $\mathbf{s}$ and the contact normal $\mathbf{n}$ are expressed in the global frame. The first three PC types are for polyhedral features exclusively, while the remainder involve cylindrical faces and circular edges, as well.

A.1 Vertex-plane contacts

A vertex V on the moving part has model position $\mathbf{r}$, and planar face F on the stationary part has globally-defined outward normal $\mathbf{n}$, surface point $\mathbf{q}$, and orthonormal planar basis vectors $\mathbf{b}_1$,$\mathbf{b}_2$. $\mathbf{r}$ and $\mathbf{n}$ provide the moving part's contact point and the contact normal. To compute contact point $\mathbf{s}$ on the stationary part:

/* Transform vertex V to the global frame */
$\mathbf{dx} + RPY(\delta\mathbf{x})\,\mathbf{r} \rightarrow \mathbf{r}'$
/* Project vertex V onto face F */
$((\mathbf{r}' - \mathbf{q}) \cdot \mathbf{b}_1)\,\mathbf{b}_1 + ((\mathbf{r}' - \mathbf{q}) \cdot \mathbf{b}_2)\,\mathbf{b}_2 \rightarrow \mathbf{s}$

A.2 Plane-vertex contacts

A face F on the moving part has outward normal $\mathbf{n}_f$ and surface point $\mathbf{q}$ in model coordinates, and stationary part vertex V has global position $\mathbf{s}$, which provides the moving part's contact point. To compute contact point $\mathbf{r}$ on the moving part and contact normal $\mathbf{n}$:

/* Transform face F's normal to the global frame */
$-RPY(\delta\mathbf{x})\,\mathbf{n}_f \rightarrow \mathbf{n}$
/* Project vertex V onto face F */
$\mathbf{s} - ((\mathbf{s} - \mathbf{q} \cdot \mathbf{n})\,\mathbf{n} \rightarrow \mathbf{r}$
/* Transform moving part contact point to the model frame */
$(RPY(\delta\mathbf{x}))^T\,(\mathbf{r} - \mathbf{dx}) \rightarrow \mathbf{r}$

A.3 Line-line contacts

An edge E_1 on the moving part has point $\mathbf{q}_m$ and direction $\mathbf{v}$ in model coordinates, and stationary part edge E_2 has global point $\mathbf{q}_s$ and direction $\mathbf{w}$. Let $\mathbf{n}_1$ and $\mathbf{n}_2$ be the outward normals of the two faces adjacent to E_2. To compute contact points $\mathbf{r}$ and $\mathbf{s}$ on the two parts, and contact normal $\mathbf{n}$:

/* Transform edge E_1's point and direction vector to the global frame */
$RPY(\delta\mathbf{x})\,\mathbf{q}_m + \mathbf{dx} \rightarrow \mathbf{q}_m'$
$RPY(\delta\mathbf{x})\,\mathbf{v} \rightarrow \mathbf{v}'$
/* Obtain closest points between moving part and stationary part edges */
$NearestPoints(line(\mathbf{q}_m, \mathbf{v}), line(\mathbf{q}_s, \mathbf{w})) \rightarrow \mathbf{r}, \mathbf{s}$
/* Transform moving part contact point to the model frame */
$(RPY(\delta\mathbf{x}))^T\,(\mathbf{r} - \mathbf{dx}) \rightarrow \mathbf{r}$
/* Obtain contact normal */
$(\mathbf{v}' \times \mathbf{w}) / |\mathbf{v}' \times \mathbf{w}| \rightarrow \mathbf{n}$
/* Reverse contact normal if it points into the stationary part surface */
if $\mathbf{n} \cdot (\mathbf{n}_1 + \mathbf{n}_2) < 0$
  then $-\mathbf{n} \rightarrow \mathbf{n}$
endif

A.4 Vertex-cylinder contacts

A vertex V on the moving part has model position $\mathbf{r}$, and cylindrical face F on the stationary part has globally-defined axis point $\mathbf{q}$, axis direction $\mathbf{a}$, and radius α. Orthonornal basis vectors $\mathbf{b}_1, \mathbf{b}_2$ span the vector space orthognal to $\mathbf{a}$, and $\mathbf{r}$ provides the moving part's contact point. To compute the contact point $\mathbf{s}$ on the stationary part and contact normal $\mathbf{n}$:

/* Transform vertex V to the global frame */
$\mathbf{dx} + RPY(\delta\mathbf{x})\,\mathbf{r} \rightarrow \mathbf{r}'$
/* Project vertex V onto the vector space orthogonal to $\mathbf{a}$ */
$((\mathbf{r}' - \mathbf{q}) \cdot \mathbf{b}_1)\,\mathbf{b}_1 + ((\mathbf{r}' - \mathbf{q}) \cdot \mathbf{b}_2)\,\mathbf{b}_2 \rightarrow \mathbf{b}$
/* Obtain contact normal */
$-\mathbf{b}/\,|\mathbf{b}| \rightarrow \mathbf{n}$
/* Obtain closest point on cylindrical face to vertex */
$\mathbf{q} + (\mathbf{r}' \cdot \mathbf{a}) - \alpha\,\mathbf{n} \rightarrow \mathbf{s}$

A.5 Circle-cylinder contacts

A circular edge E on the moving part has centroid $\mathbf{p}$, radius α_1, and orthonormal basis $\mathbf{a}_1, \mathbf{a}_2$ for its circle's planar vector space, all in model coordinates. Cylindrical face F on the stationary part has globally-defined axis point $\mathbf{q}$, axis direction $\mathbf{a}$, and radius α. Orthonornal basis vectors $\mathbf{b}_1, \mathbf{b}_2$ span the vector space orthognal to $\mathbf{a}$. To compute contact points $\mathbf{r}$ and $\mathbf{s}$ on the two parts, and contact normal $\mathbf{n}$:

/* Transform circular edge E to the global frame */
$RPY(\delta\mathbf{x})\,\mathbf{p} + \mathbf{dx} \rightarrow \mathbf{p}$ $\quad RPY(\delta\mathbf{x})\,\mathbf{a}_1 \rightarrow \mathbf{a}_1$ $\quad RPY(\delta\mathbf{x})\,\mathbf{a}_2 \rightarrow \mathbf{a}_2$
/* Create a function to compute the distance of the circular edge
point corresponding to angle θ from the cylindrical axis of F
define $f(\theta)$
/* Generate circular edge point corresponding to θ */
$\mathbf{p} + \alpha_1\,cos\theta\mathbf{a}_1 + \alpha_2\,cos\theta\mathbf{a}_2 \rightarrow \mathbf{r}$
/* Project circular edge point into plane perpendicular to cyl face */
$((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_1)\,\mathbf{b}_1 + ((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_2)\,\mathbf{b}_2 \rightarrow \mathbf{b}$
return $|\mathbf{b}|$
enddefine
/* Obtain "best" contact points on E and F */
Maximize $f(\theta)$ and retain corresponding $\mathbf{r}$, $\mathbf{b}$
/* Obtain contact normal */
$-\mathbf{b}/\,|\mathbf{b}| \rightarrow \mathbf{n}$
/* Obtain contact point on statinoary part */
$\mathbf{q} + (\mathbf{r} - \mathbf{q}) \cdot \mathbf{a} + \alpha_2\mathbf{n} \rightarrow \mathbf{s}$

/* Transform moving part contact point to the model frame */
$(RPY(\delta\mathbf{x}))^T(\mathbf{r} - \mathbf{dx}) \rightarrow \mathbf{r}$

A.6 Circle-plane contacts

A circular edge E on the moving part has centroid $\mathbf{p}$, radius α_1, and orthonormal basis $\mathbf{a}_1, \mathbf{a}_2$ for its circle's planar vector space, all in model coordinates. Planar face F on the stationary part has globally-defined outward normal $\mathbf{n}$, surface point $\mathbf{q}$, and orthonormal planar basis vectors $\mathbf{b}_1$,$\mathbf{b}_2$. To compute contact points $\mathbf{r}$ and $\mathbf{s}$ on the two parts, and contact normal $\mathbf{n}$:

/* Transform circular edge E to the global frame */
$RPY(\delta\mathbf{x})\,\mathbf{p} + \mathbf{dx} \rightarrow \mathbf{p} \quad RPY(\delta\mathbf{x})\,\mathbf{a}_1 \rightarrow \mathbf{a}_1 \quad RPY(\delta\mathbf{x})\,\mathbf{a}_2 \rightarrow \mathbf{a}_2$
/* Create a function that returns the $\mathbf{n}$-ordinate of the circular edge
point corresponding to angle θ
define $f(\theta)$
/* Generate circular edge point corresponding to θ */
$\mathbf{p} + \alpha_1\, cos\theta\mathbf{a}_1 + \alpha_2\, cos\theta\mathbf{a}_2 \rightarrow \mathbf{r}$
return $\mathbf{n} \cdot \mathbf{r}$
enddefine
/* Obtain "best" contact points on E and F */
Minimize $f(\theta)$ and retain corresponding $\mathbf{r}$
/* Project circular edge point onto planar face */
$((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_1)\,\mathbf{b}_1 + ((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_2)\,\mathbf{b}_2 \rightarrow \mathbf{s}$
/* Transform moving part contact point to the model frame */
$(RPY(\delta\mathbf{x}))^T(\mathbf{r} - \mathbf{dx}) \rightarrow \mathbf{r}$

A.7 Circle-line contacts

A circular edge E_1 on the moving part has centroid $\mathbf{p}$, radius α, and orthonormal basis $\mathbf{a}_1, \mathbf{a}_2$ for its circle's planar vector space, all in model coordinates. Stationary part linear edge E_2 has global point $\mathbf{q}$ and direction $\mathbf{w}$. The planar faces adjacent to E_2 have normals $\mathbf{n}_1$ and $\mathbf{n}_2$. To compute contact points $\mathbf{r}$ and $\mathbf{s}$ on the two parts, and contact normal $\mathbf{n}$:

```
/* Transform circular edge E_1 to the global frame */
RPY(δx) p + dx → p    RPY(δx) a_1 → a_1    RPY(δx) a_2 → a_2
/* Create a function to compute distance between the circular edge point
   corresponding to angle θ and the linear edge */
define f(θ)
   /* Generate circular edge point corresponding to θ */
   p + α_1 cosθ a_1 + α_2 sinθ a_2 → r
   q + ((r − q) · w) w → s
   return |r − s|
enddefine
/* Obtain "best" contact points on E_1 and E_2 */
Minimize f(θ) and retain corresponding r, s
/* Compute circular edge tangent at r */
−sinθ a_1 + cosθ a_2 → v
/* Obtain contact normal */
(v × w) / |v × w| → n
/* Reverse contact normal if it points into the stationary part surface */
if n · (n_1 + n_2) < 0
   then −n → n
endif
/* Transform moving part contact point to the model frame */
(RPY(δx))^T (r − dx) → r
```

A.8 Circle-circle contacts

A circular edge E on the moving part has centroid $\mathbf{p}$, radius α_1, and orthonormal basis $\mathbf{a}_1, \mathbf{a}_2$ for its circle's planar vector space, all in model coordinates. Stationary part circular edge E_2 has global centroid $\mathbf{q}$, radius α_2, and orthonormal basis $\mathbf{a}_1, \mathbf{a}_2$ for its circle's planar vector space. The planar face adjacent to E_2 has normal $\mathbf{n}_1$. To compute contact points $\mathbf{r}$ and $\mathbf{s}$ on the two parts, and contact normal $\mathbf{n}$:

```
/* Transform circular edge E_1 to the global frame */
RPY(δx) p + dx → p    RPY(δx) a_1 → a_1    RPY(δx) a_2 → a_2
/* Create a function to compute distance between the point on E_1
   corresponding to angle θ, and E_2 */
define f(θ)
   /* Generate circular edge point corresponding to θ */
   p + α_1 cosθ a_1 + α_2 sinθ a_2 → r
```

/* Project E_1 point into plane of E_2 */
$((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_1)\,\mathbf{b}_1 + ((\mathbf{r} - \mathbf{q}) \cdot \mathbf{b}_2)\,\mathbf{b}_2 \rightarrow \mathbf{b}$
$\mathbf{q} + \alpha_2\,\mathbf{b} \rightarrow \mathbf{s}$
return $|\mathbf{r} - \mathbf{s}|$
enddefine
/* Obtain "best" contact points on E_1 and E_2 */
Minimize $f(\theta)$ and retain corresponding $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{b}$
/* Compute circular edge E_1's tangent at $\mathbf{r}$ */
$-sin\theta\,\mathbf{a}_1 + cos\theta\,\mathbf{a}_2 \rightarrow \mathbf{v}$
/* Compute circular edge E_2's tangent at $\mathbf{s}$ */
$-(\mathbf{b} \cdot \mathbf{b}_2)\mathbf{b}_1 + (\mathbf{b} \cdot \mathbf{b}_1)\mathbf{b}_2 \rightarrow \mathbf{w}$
/* Obtain contact normal */
$(\mathbf{v} \times \mathbf{w}) \,/\, |\mathbf{v} \times \mathbf{w}| \rightarrow \mathbf{n}$
/* Reverse contact normal if it points into the stationary part surface */
if $\mathbf{n} \cdot (\mathbf{n}_1 - \mathbf{b}) < 0$
then $-\mathbf{n} \rightarrow \mathbf{n}$
endif
/* Transform moving part contact point to the model frame */
$(RPY(\delta\mathbf{x}))^T\,(\mathbf{r} - \mathbf{dx}) \rightarrow \mathbf{r}$

APPENDIX B

EQUIVALENT CONTACT SPACE FACETS

To test for the equivalence of two facets in contact space, we first determine if their tangent hypersurfaces are equal, and then check for coincident surface points on their actual surfaces. Although this technique is somewhat *ad hoc* (since only a finite number of coincident surface points are tested), the probability that a pair of unequal surfaces will be judged to be equal is fairly small. Surface points are computed by the following procedure, which perturbs a facet surface point $\mathbf{X}$ in the direction of a specified surface tangent basis vector, returning the resulting surface point:

```
/* Perturb surface point X of facet φ in the direction of the
    ith vector in basis U_1, ... U_n */
define PerturbFacetPoint(U_1, ... U_n, PC_1, ..., PC_{6-n}, X, i) → Y
    Set up n 5-dim hyperplanes U_j · Y = D_j, where:
        D_j = U_j · Y + ε when j = i, D_j = U_j · Y otherwise
    Solve for the point Y of intersection between the n hyperplanes and
        the 6 − n C-surfaces of PC_1, ..., PC_{6-n}, via the multivariable
        Newton method, as described in chapter 2.4.1
enddefine
```

Procedure *EqualFacets* tests for equality between two facets by first verifying the equality of the two surface tangents at a fixed surface point, and then checking for coincident surface points. Each test point is defined by perturbing the fixed surface point in the direction of a surface tangent basis vector:

```
/* Determine whether primitive contacts PC_1 ... PC_m define a distinct
    contact state within facet φ */
define EqualFacets(φ_1, φ_2) → equal
    Let X be the pose of a contact space vertex adjacent to φ
    if ASpace_X(support(φ_1)) = ASpace_X(support(φ_2))
        then
            Let U_1, ... U_n be an orthnml. basis for ASpace_X(support(φ_1))
            True → equal
            for i = 1 to n do
                PerturbFacetPoint(U_1, ... U_n, support(φ_1), X, i) → Y_1
                PerturbFacetPoint(U_1, ... U_n, support(φ_2), X, i) → Y_2
                if Y_1 ≠ Y_2
                    then
                        False → equal
                        Exit loop
                endif
            endfor
        else
            False → equal
    endif
enddefine
```

APPENDIX C

EQUIVALENT CONTACT STATES

Here we consider how to determine whether a set of primitive contacts $PC_1 \ldots PC_m$ form a distinct contact state within a facet ϕ. $PC_1 \ldots PC_m$ are known to be a subset of $facetPCs(\phi)$, defining the same (infinite) facet as ϕ. It remains to be determined whether contacts $PC_1 \ldots PC_m$ can coexist without the necessary formation of additional contacts in $facetPCs(\phi)$. With regard to the planar contact in Figure 20 of section 2.5.1, for example, it is apparent that PCs A, B, and D cannot coexist without forming contact C. Procedure $Feasible1$, defined below, performs an optimization of assembly pose $\mathbf{X}$, maximizing its distance from all contacts outside of $PC_1 \ldots PC_m$, while restricting $\mathbf{X}$ to the convex point sets of $PC_1 \ldots PC_m$. $PC_1 \ldots PC_m$ can coexist as a distinct contact state iff $\mathbf{X}$'s optimized distance to each "outside" contact is greater than zero. Supporting the optimization of pose $\mathbf{X}$, equation 29 is first modified to yield the distance from $\mathbf{X}$ to an obstacle PC's point set:

$$Dist'(\mathbf{X}, PC_0) \;=\; \min_{i \in 1 \ldots n_0} (dist^{\star}(\mathbf{X}, A_0), dist(\mathbf{X}, A_{0i})) \tag{53}$$

Equation 30 is similarly modified, to utilize function $Dist'$:

$$cost'_P(\mathbf{X}) \;=\; \sum_{k \in opcset} \frac{1}{C - Dist'(\mathbf{X}, PC_k)} \tag{54}$$

Procedure $Feasible1$ decides if there exists a pose $\mathbf{X}$ which lies in the point sets of m primitive contacts $PC_1 \ldots PC_m$, and does not lie in the point set of any other PC. Restriction of the pose to the PCs' bounded point sets is accomplished by establishing artificial C-space limits (see chapter 2.4.3), which serve as halfspace constraints corresponding to the PCs' bounding contacts.

```
/* Determine whether primitive contacts PC_1 ... PC_m define a distinct
    contact state within facet φ */
define Feasible1(φ, PC_1 ... PC_m)
    Let bpcset be the set of bounding contacts associated with
       PC_1 ... PC_m, minus the PCs in facetPCs(φ)
    Let opcset be all PCs of the local C-space, minus PC_1 ... PC_m
       and minus the PCs in bpcset
    Construct an artificial C-space limit for each PC in bpcset
    Minimize cost'_P(X) varying X along φ within the stated C-limits
    if Dist'(X, PC_k) > 0 for all k in opcset
          then return True
          else return False
    endif
enddefine
```

Procedure $Feasible2$ decides if there exists a pose $\mathbf{X}$ shared by the point sets of two contact states. Facet ϕ is supplied as the surface over which $\mathbf{X}$ varies.

```
/* Determine whether contact states S_1 and S_2 intersect in 6-space */
define Feasible2(φ, S_1, S_2)
    Let PC_1 ... PC_m be the combined PCs of states S_1 and S_2
    return Feasible1(φ, PC_1 ... PC_m)
enddefine
```

APPENDIX D

AFFINE SPACE PROXIMITY

Given a pair of affine spaces A_1, A_2 in $\Re^6$ with respective point and basis descriptors $\mathbf{P}_1$, $\mathbf{U}_1 \ldots \mathbf{U}_{m_1}$ and $\mathbf{P}_2$, $\mathbf{V}_1 \ldots \mathbf{V}_{m_2}$, calculate the the closest point on A_1 to A_2 and the closest point on A_2 to A_1, as well as the shortest distance from A_1 to A_2, as follows. Let $\mathbf{U}_1 \ldots \mathbf{U}_{m_1}, \mathbf{U}_{m+1} \ldots \mathbf{U}_m$ be an orthonormal basis for $span\{\mathbf{U}_1 \ldots \mathbf{U}_{m_1}, \mathbf{V}_1 \ldots \mathbf{V}_{m_2}\}$. Compute the proximity vector $\mathbf{W}$ and the distance of separation d:

$$\mathbf{R} = \mathbf{P}_2 - \mathbf{P}_1 \tag{55}$$

$$\mathbf{W}^* = \mathbf{R} - \sum_{i=1}^{m} (\mathbf{R} \cdot \mathbf{U}_i)\ \mathbf{U}_i \tag{56}$$

$$\mathbf{W} = \frac{\mathbf{W}^*}{|\mathbf{W}^*|} \tag{57}$$

$$d = \mathbf{R} \cdot \mathbf{W} \tag{58}$$

Now let $\mathbf{U}_1 \ldots \mathbf{U}_{m_1}, \mathbf{V}_1 \ldots \mathbf{V}_{m_2'}$ be a basis for $span\{\mathbf{U}_1 \ldots \mathbf{U}_{m_1}, \mathbf{V}_1 \ldots \mathbf{V}_{m_2}\}$, with $m_2' \leq m_2$. Then

$$\mathbf{P}_1 + \sum_{i=1}^{m_1} \lambda_{1i} \mathbf{U}_i + d\mathbf{W} \ = \ \mathbf{P}_2 + \sum_{i=1}^{m_2'} \lambda_{2i} \mathbf{V}_i \tag{59}$$

for some $\lambda_{11} \ldots \lambda_{1m_1}, \lambda_{21} \ldots \lambda_{2m_2'}$. The term $\mathbf{P}_1 + \sum_{i=1}^{m_1} \lambda_{1i} \mathbf{U}_i$ is the closest point on A_1 to A_2, and the term $\mathbf{P}_2 + \sum_{i=1}^{m_2'} \lambda_{2i} \mathbf{V}_i$ is the closest point on A_2 to A_1. (26) is a consistent system of 6 equations in $m_1 + m_2' \leq 6$ unknowns, and its solution parameterizes the closest points on the two affine spaces.

APPENDIX E

CONTACT POINTS AND NORMALS

Given an ellipsoid whose principle axes correspond to the standard axes of n-space:

$$\sum_{i=1}^{6} D_i X_i^2 = 1 \tag{60}$$

the point $\mathbf{X}$ on the ellipsoid's surface that corresponds to the direction $\mathbf{U}$ is $\lambda_{\mathbf{U}}\mathbf{U}$ where

$$\lambda_{\mathbf{u}} = \frac{1}{\sqrt{\sum_{i=1}^{6} D_i U_i^2}} \tag{61}$$

References

REFERENCES

[1] Ambler, A.P., Popplestone, R.J. (1975), "Inferring the positions of bodies from spatial relationships", *Artificial Intelligence*, Vol. 6, No. 2, pp. 157-174.

[2] Asada, H. (1990), "Teaching and learning of compliance using neural nets: representation and generation of nonlinear compliance", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.1237-1244.

[3] Asada, H., Hirai, S. (1989), "Towards a symbolic-level force feedback: recognition of assembly process states", *Proceedings of the 5th International Symposium of Robotics Research, Tokyo, 1989.*

[4] Asada, H., Kakumoto, Y. (1988), "The dynamic RCC hand for high-speed assembly", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.120-125.

[5] Ballard, D.H. (1981), "Generalizing the Hough transform to detect arbitrary shapes", *Pattern Recognition*, Vol 13, No, 2, pp. 111-122.

[6] Barbera, A.J., *An Architecture for a Robot Hierarchical Control System*, National Bureau of Standards, Special Publication 500-23, U.S. Government Printing Office, Washington, D.C.

[7] Bledsoe, W.W. (1975), "A new method for proving certain Presburger formulas", *IJCAI at Tibilsi, Georgia, U.S.S.R.*

[8] Bolle, R.M., Cooper, D.B. (1986), "On optimally combining pieces of information with application to estimating 3-D complex-object position from range data", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 5, pp. 619-638.

[9] Bolles, R.C., Cain, R.A. (1982), "Recognizing and locating partially visible objects: the local-feature-focus method", *International Journal of Robotics Research*, Vol. 1, No. 3, pp. 57-82.

[10] Boothroyd, G., Poli, C, Murch, L. (1982), *Automated Assembly*, New York: Marcel Dekker, pp. 280-283.

[11] Bourjault, A. (1984), *Contribution a une Approche Méthodologique de L'Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires*, Thèse d'Etat, Université de Franche-Comté, Besançon, France.

[12] Boyse, J.W. (1979), "Interference detection among solids and surfaces", *Communications of the ACM*, Vol. 22, No. 1, pp. 3-9.

[13] Brady, M., Hollerbach, J.M., Johnson, T.L., Lozano-Pérez, T., Mason, T. (1982), *Robot Motion: Planning and Control*, Cambridge, Massachusetts: MIT Press., Chapter 6.

[14] Brooks, R.A. (1983), "Model-based three-dimensional interpretations of two-dimensional images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 5, No. 2, pp. 140-150.

[15] Brooks, R.A. (1983), "Planning collision-free motions for pick-and-place operations", *International Journal of Robotics Research*, Vol. 2, No. 4, pp. 19-44.

[16] Brooks, R.A. (1983), "Solving the find-path problem by good representation of free space", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, pp. 190-197.

[17] Brooks, R.A. (1982), "Symbolic error analysis and robot planning", *Int. Journal of Robotics Research*, Vol. 1, No. 4 pp. 29-68.

[18] Brooks, R.A. (1981), "Symbolic reasoning among 3-D models and 3-D images", *Artificial Intelligence*, Vol. 17, pp. 285-348.

[19] Brost, R.C. (1986), "Automatic grasp planning in the presence of uncertainty", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1575-1581.

[20] Bruynincks, H., DeSchutter, J. (1992), "A systematic derivation of on-line motion constraint identification equations for model-based compliant motions", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1513-1518.

[21] Buckley, C.E. (1985), "A proximity metric for continuum path planning", *9th International Joint Conf. on Artificial Intelligence*, pp. 1096-1102.

[22] Buckley, S,J. (1987), "Planning and teaching compliant motion strategies", PhD Thesis, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass. Also M.I.T. A.I. Lab T.R. 936.

[23] Caine, M.E. (1985), "Chamferless assembly of rectangular parts in two and three dimensions", S.M. Thesis, Dept. of M.E., MIT.

[24] Caine, M.E., Lozano-Pérez, T., Seering, W.P. (1989), "Assembly strategies for chamferless parts", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 472-477.

[25] Cameron, S,A. (1984), "Modelling solids in motion", PhD Thesis, University of Edinburgh.

[26] Cameron, S.A., Culley, R.K. (1986), "Determining the minimum translational distance between two convex polyhedra", *International Conference on Robotics and Automation*, pp. 591-596.

[27] Canny, J.F. (1987), "The complexity of robot motion planning", PhD Dissertation, Massachusetts Institute of Technology, Cambridge, Mass.

[28] Canny, J.F. (1989), "On computability of fine motion plans", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 177-182.

[29] Canny, J.F., Lin, M.C. (1990), "An opportunistic global path planner", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1554-1539.

[30] Canny, J.F., Reif, J. (1987), "New lower-bound techniques for robot motion planning problems", *Proceedings of the IEEE Symposium on the Foundations of Computer Science.*

[31] Chin, F., Wang, C.A. (1983), "Optimal algorithms for the intersection and minimum distance problems between planar polygons", *IEEE Transactions on Computers*, Vol. C-32, pp. 1203-1209.

[32] Collins, G.E. (1975), "Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in *Second GI Conference on Automata Theory and Formal Languages*, *Lecture Notes in Computer Science*, Vo. 33, pp. 134-183, Springer-Verlag, Berlin.

[33] Connolly, C.I., Burns, J.B., Weiss, R. (1990), "Path planning using Laplace's Equation", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2102-6.

[34] Connolly, C.I., Grupen (1993), "The applications of harmonic functions to robotics", *Journal of Robotic Systems*, Vol. 10, No. 7, pp. 931-46.

[35] Corner, D.F., Ambler, A.P., Popplestone, R.J. (1983), "Reasoning about the spatial relationships derived from a RAPT program for describing assembly by robot", in *Proc. of IJCAI 1983*, pp. 842-844.

[36] Cutkosky, M.R., Wright, P.K. (1986), "Modelling manufacturing grips and correlations with the design of a robot hand", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1533-1539.

[37] Dakin, G., Popplestone, R.J. (1991), "Augmenting a nominal assembly motion plan with a compliant behavior", *Proceedings of the AAAI Conference*, Vol. 1, pp. 653-658.

[38] Dakin, G., Popplestone, R.J. (1992), "Simplified fine-motion planning in generalized contact space", *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 281-6.

[39] Dakin, G., Popplestone, R.J. (1993), "Contact space analysis for narrow-clearance assemblies", *Proc. IEEE International Symposium on Intelligent Control*, pp. 542-7.

[40] Dakin, G., Popplestone, R.J. (1993), "Fine-motion planning in the contact space of narrow-clearance assemblies", *Proc. IASTED International Conference on Robotics and Manufacturing*, pp. 206-8 and T.R. 93-60, Cmp. Sci., U.Mass., Amherst MA.

[41] DeFazio, T.L., Whitney, D.E. (1987), "Simplified generation of all assembly sequences", *IEEE Journal of Robotics and automation*, Vol. RA-3, No. 6, pp.640-658.

[42] Desai, R.S. (1987), "On Fine Motion in Mechanical Assembly in Presence of Uncertainty", PhD Dissertation, Dept. of Mechanical Engineering., Univ. of Michigan.

[43] De Schutter, J., Van Brussel, H. (1988), "Compliant robot motion I. A formalism for specifying compliant motion tasks", *International Journal of Robotics Research*, Vol. 7, No. 4, pp. 3 - 16.

[44] Dobkin, D.P., Kirkpatrick, D.G. (1985), "A linear algorithm for determining the separation of convex polyhedra", *Journal of Algorithms*, Vol. 6, pp. 381-392.

[45] Donald, B.R. (1984), "Motion planning with six degrees of freedom", Technical Report AI-TR-791, A.I. Laboratory, MIT.

[46] Donald, B.R. (1985), "On motion planning with six degrees of freedom: solving the interaction problems in configuration space", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 536-541.

[47] Donald, B.R. (1986), "Robot motion planning with uncertainty in the geometric models of the robot and environment: a formal framework for error detection and recovery", *Proceedings of the IEEE Int. Conference on Robotics and Automation*, pp. 1588-1593.

[48] Dufay, B., Latombe, J.C. (1984), "An approach to automatic robot programming based on inductive learning", *International Journal of Robotics Research*, Vol. 3, No. 4, pp. 3-20.

[49] Drysdale, R.L, "Generalized voronois diagrams and geometric searching", Dept. of Computer Science, Stanford Univ.

[50] Erdmann, M. (1984), "On motion planning with uncertainty", Tech. Report AI-TR-810, Artificial Intelligence Laboratory, MIT.

[51] Erdmann, M. (1985), Using backprojections for fine motion planning with uncertainty", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 549-554.

[52] Erdmann, M., Mason, M. (1986), "An exploration of sensorless manipulation", *IEEE International Conference on Robotics and Automation*, pp. 1569-1574.

[53] Faverjon, B. (1984), "Obstacle avoidance using an octree in the configuration space of the manipulator", *Proceedings of the IEEE International Conference on Robotics, Atlanta, Ga.*, pp. 504-510.

[54] Ferrari, C., Canny, J. (1992), "Planning optimal grasps" *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2290-2295.

[55] Gilbert, E.G., Johnson, D.W. (1985), "Distance functions and their application to robot path planning in the presence of obstacles", *IEEE Journal of Robotics and Automation*, Vol. RA-1, pp. 21-30.

[56] Gilbert, E.G., Johnson, D.W., Keerthi (1988), "A fast procedure for computing the distance between complex objects in 3-dimensional space", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, pp. 193-203.

[57] Gordon, S.J., Seering, W.P. (1997), "Locating polyhedral features from sparse light-stripe data", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 801-806.

[58] Grimson, W.E.L., Lozano-Pérez, T. (1984), "Model-based recognition and localization from sparse range data or tactile data", *International Journal of Robotics Research*, Vol. 3, No. 3, pp. 3-35.

[59] Grupen, R.A., (1988), "General purpose grasping and manipulation with multifingered robot hands", PhD thesis, University of Utah, Merrill Engineering Building, Salt Lake City, UT 84112.

[60] Gullapalli, V. (1990), "A stochastic reinforcement learning algorithm for learning real-valued functions", *Neural Networks*, Vol. 3, pp. 671-692.

[61] Gullapalli, V., Grupen, R.A., Barto, A.G. (1992), "Learning reactive admittance control", *Proceedings of the IEEE Conference on Robotics and Automation*, Vol. 2, pp. 1475-1480.

[62] Hervé, J.M., "Analyse structurelle des mécanismes par groupe des déplacements", *Mechanism and Machine Theory*, Vol 13, No. 4, 1978.

[63] Hirai, S., Iwata, K. (1992), "A model-based generation of damping control law for part-mating", submitted to the IEEE International Conference on Intelligent Robots and Systems.

[64] Hirzinger, G., Landzettel, K. (1985), "Sensory feedback structures for robots with supervised learning", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.627-635.

[65] Hogan, N. (1985), "Impedance control: an approach to manipulation", *ASME Journal of Dynamics, Systems, Measurement, and Control*, Vol 107: pp. 1-7.

[66] Homem de Mello, L.S., (1989), "Task sequence planning for robotic assembly", PhD Dissertation, ECE Dept., Carnegie Mellon Univ., Pittsbirgh PA.

[67] Homem de Mello, L.S., Sanderson, A.C. (1990), "Evaluation and selection of assembly plans", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.1588-1593.

[68] Huang, Y.F., Lee, C.S.G. (1990), "An automatic assembly planning system", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1594-1599.

[69] Iberall, A.R. (1987), "A neural model of human prehension", PhD Thesis, U. Mass., Amherst MA.

[70] Inoue, H. (1974), "Force feedback in precise assembly tasks", MIT Artificial Intelligence Laboratory, AIM-308.

[71] Jacobsen, S.C., Iverson, E.K., Knutti, D.F., Johnson, R.T., Biggers, K.B. (1986), "Design of the Utah/MIT Dextrous Hand", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1520-1532.

[72] Kazerooni, H. (1988), "Direct-drive active compliant end effector (active RCC)", *International Journal of Robotics Research*, Vol. 4, No. 3, pp. 324-333.

[73] Kerr, J., Roth, B. (1986), "Analysis of multifingered hands", *International Journal of Robotics Research*, Vol. 4, No. 4, pp. 3-17.

[74] Khatib, O. (1985), "Real-time obstacle avoidance for manipulators and mobile robots", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 500-505.

[75] Koutsou, A. (1986), "Parts mating by moving objects in contact", PhD Thesis, Dept. of A.I., Edinburgh University.

[76] Latombe, J.C. (1989), "Motion planning with uncertainty: on the preimage backchaining approach", in *The Robotics Review*, (eds) O. Khatib, J. Craig, T. Lozano-Pérez, Cambridge, Mass.: MIT Press.

[77] Latombe, J.C., Mazer, E. (1981), "LM: a high-level programming language for controlling manipulator robots", *Proeedings of the 11th International Symposium on Industrial Robots*, Tokyo, Japan.

[78] Laugier, C. (1989), "Planning fine motion stategies by reasoning in contact space", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.653-659.

[79] Laugier, C., Theveneau, P. (1986), "Planning sensor-based motions for part-mating using geometric reasoning techniques", *European Conference on Artificial Intelligence*, Brighton.

[80] Lee, C.S.G., Hou, E.S.H. (1988), "Automatic generation and synthesis of C-frames for mechanical parts in an insertion task", *Journal of Robotics and Automation*, Vol. 4, No. 3, pp. 287 -293.

[81] Lee, S., Kim, M.H. (1989), "Learning expert systems for robot fine-motion control", *Proceedings of the IEEE International Symposium on Intelligent Control 1988*, Arlington, VA.

[82] Lee, S., Shin, Y.G. (1990), "Assembly planning based on subassembly extraction", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1606-1611.

[83] Li, Z., Hsu, P., Sastry, S. (1989), "Grasping and coordinated manipulation by a multifingered robot gripper", *International Journal of Robotics Research*, Vol. 8, No. 4, pp. 33-50.

[84] Lin, M.C., Canny, J.F. (1991), "A fast algorithm for incremental distance calculation", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1008-1014.

[85] Liu, Y. (1993) "A geometric approach for denoting and intersecting TR subgroups of the Euclidean group", Tech Report 93-44, Department of Computer Science, University of Massachusetts, Amherst MA.

[86] Liu, Y. (1989), "Planning for assembly from solid models", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 222-227.

[87] Liu, Y. (1990), "Symmetry groups in robotic assembly planning", PhD Dissertation, COINS Dept., U.Mass., Amherst MA 01003.

[88] Liu, Y. and Popplestone, R. (1994), "A Group Theoretic Formalization of Surface Contact", *International Journal of Robotics Research*, Vol. 13, No. 2, pp. 148. The MIT Press.

[89] Liu, Y. and Popplestone, R.J. (1990), "Symmetry Constraint Inference in Assembly Planning", *Proc. Eighth National Conference on A.I.*, pp. 1038-44.

[90] Liu, Y. Popplestone, R.J. (1991), "Symmetry groups in analysis of assembly kinematics", *Proc. IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 572-7.

[91] Lozano-Pérez, T. (1981), "Automatic planning of manipulator transfer movements", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 10, pp. 681-697.

[92] Lozano-Pérez, T. (1976), "The design of a mechanical assembly system", Technical Report AI TR 397, AI. Lab, MIT.

[93] Lozano-Pérez, T. (1983), "Spatial planning: a configuration-space approach", *IEEE Transactions on Computers*, Vol. C-32, No. 2., pp. 108-120.

[94] Lozano-Pérez, T. (1982), "Task planning", in Brady, M., Hollerbach, J.M., Johnson, T.L., Lozano-Pérez, T., Mason, T. (eds), *Robot Motion: Planning and Control*, Cambridge, Massachusetts: MIT Press., pp.473-498.

[95] Lozano-Pérez, T., Brooks, R. (1985), "An approach to automatic robot programming", *AI Memo 842*, MIT, Cambridge, Mass.

[96] Lozano-Pérez, T., Jones, J.L., Mazer, E., O'Donnel, P.A., Grimson, E.L. (1987), "Handey: a robot system that recognizes, plans and manipulates", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 843-849.

[97] Lozano-Pérez, T., Mason, M., Taylor, R.H. (1984), "Automatic synthesis of fine-motion strategies for robots", *International Journal of Robotics Research*, Vol. 3, No. 1, pp. 3-24.

[98] Lozano-Pérez, T. Grimson, W.E.L. (1987), "Finding cylinders in range data", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1806-1809.

[99] Luh, J.Y.S., Lin, C.S. (1981), "Optimum path planning for mechanical manipulators", *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 102, pp. 142-151.

[100] Lyons, D. (1985), "A simple set of grasps for a dextrous hand", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 588-594.

[101] Mason, M. (1983), "Automatic planning of fine motions: correctness and completeness", Computer Science Dept. and Robotics Institute, Carnegie Mellon Univ.

[102] Mason, M. (1981), "Compliance and force control for computer controlled manipulators", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11, No. 6, pp. 418-432.

[103] Mason, M. (1982), "Manipulator grasping and pushing operations", Technical Report AI-TR-690, MIT Artificial Intelligence Lab.

[104] Mazer, E. (1984), "LM-GEO geometric programming of assembly

[105] Meyer, J. (1981), "An emulation system for programmable assembly robots", *IBM Journal of Research and Development*, Vol. 25, No. 6.

[106] Myers, J. (1981), "A supervisory collision-avoidance system for robot controllers", Master's Thesis, Carnegie-Mellon University.

[107] Nguyen, T.N., Stephanou, H.E. (1990), "A topological algorithm for continuous grasp planning", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 670-675.

[108] Nguyen, V.D, (1989), "Constructing stable grasps", *International Journal of Robotics Research*, Vol. 8, No. 1, pp.26-37.

[109] Ohwovoriole, M.S. (1980), "An extension of screw theory and its applications to the automation of industrial assemblies", Stanford Artificial Intelligence Lab Memo AIM-338.

[110] Ohwovoriole, M.S., Hill, J.W., Roth, B.(1980), "On the theory of single and multiple insertions in industrial assemlies", *Proceedings of the 10th International Symposium on Industrial Robots*, Milan, Italy, March 1980, pp. 545-558.

[111] Peshkin, M.A. (1990), "Programmed compliance for error corrective assembly", *IEEE Transactions on Robotics and Automation*, Vol 6, No. 4, pp. 473-482.

[112] Pieper, D.L (1968), "The kinematics of manipulators under computer control", PhD thesis, A.I. Lab., Stanford, Univ..

[113] Popplestone, R.J., "Group theory and robotics", in Brady, M., Paul, R. (eds), *Robotics Research, The First International Symposium*, Cambridge, MA: MIT Press.

[114] Popplestone, R.J., Ambler, A.P., Bellos, I. (1980), "An interpreter for a language describing assemblies", Artificial Intelligence, Vol. 14, No. 1, pp. 79-107.

[115] Popplestone, R., Weiss, R., Liu, Y. (1988), "Using characteristic invariants to infer new spatial relationships from old", *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, Vol. 2, pp. 1106-1112.

[116] Raibert, M.H., Craig, J.J. (1981), "Hybrid position/force control of manipulators", Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control, Vol. 102, pp. 126-133.

[117] Rajan, V.T., Burridge, R., Schwartz, J.T. (1987), "Dynamics of a rigid body in frictional contact with rigid walls", *Proceedings of the IEEE Int. Conference on Robotics and Automation*, pp. 671-677.

[118] Rimon, E., Koditschek, D.E. (1990), "Exact robot navigation in geometrically complicated but topologically simple spaces", *Proceedings of the IEEE Conference on Robotics and Automation*, Vol. 3, pp. 1937-42.

[119] Rimon, E., Koditschek, D.E. (1988), "Exact robot navigation using cost functions: the case of spherical boundaries in E^n", *Proceedings of the IEEE Conference on Robotics and Automation*, Vol. 3, pp. 1791-6. with boundary",

[120] Salisbury, J.K. (1980), "Active stiffness control of a manipulator in Cartesian coordinates", *Proceedings of the 19th IEEE Conference on Decision and Control*, pp. 95-100.

[121] Salisbury, J.K., Craig, J.J. (1982), "Articulated hands: force control and kinematics", *International Journal of Robotics Research*, Vol. 1, No. 1.

[122] Schimmels, J.M., Peshkin, M.A. (1992), "Admittance matrix design for force-guided assembly", *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 213-227.

[123] Schwartz, J.T., Sharir, M. (1983), "On the piano movers problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal bariers", *Communications on Pure and Applied Mathematics*, Vol. 36, pp. 345-398.

[124] Schwartz, J.T., Sharir, M. (1983), "On the piano movers problem: II. General techniques for computing topological properties of real algebraic manifolds", *Advances in Applied Mathematics*, Vol. 4, pp. 298-351.

[125] Shimoga, K.B., Goldenberg, A.A. (1992), "Constructing multifingered grasps to achieve admittance center", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2296-2301.

[126] Simunovic, S. (1975), "Force information in assembly processes", *Proc. of the 5th Int. Sym. on Industrial Robotics*, pp. 415-427.

[127] Strip, D.R. (1988), "Insertions using geometric analysis and hybrid force-position control: method and analysis", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.1744-51.

[128] Sturges, R.H. (1988), "A three-dimensional assembly task, quantification with application to machine dexterity", *International Journal of Robotics Research*, Vol. 7, No. 4, pp. 34-78.

[129] Tarassenko, L. Blake, A. (1991), "Analogue computation of collision-free paths", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 540-545.

[130] Taylor, R.H. (1976), "A synthesis of manipulator control programs from task-level specifications", Stanford Univ., A.I. Lab., Memo AIM-282.

[131] Thomas, F., Torras, C. (1988), "A group-theoretic approach to the computation of symbolic part relations", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 6.

[132] Thompson, D.W., Mundy, J.L. (1987), "Three-dimensional model matching from an Unconstrained Viewpoint", *IEEE International Conference on Robotics and Automation*, pp. 208-220.

[133] Trinckle, J.C., Abel, J.M., Paul, R.P. (1987), "Enveloping, frictionless, planar grasping", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 246-251.

[134] Udupa, S.M. (1977), "Collision detection and avoidance in computer controlle manipuators", *Proceedings of the 5th International Conference on Artificial Intelligence*, MIT, Cambridge Mass.

[135] Valade, J. (1984), "Automatic generation of trajectories for assembly tasks", *Sixth European Conference on A.I.*, Pisa, Italy.

[136] Whitney, D.E. (1985), "Historical perspective and state of the art in robot force control", *IEEE International Conference on Robotics and Automation*, pp. 262-268.

[137] Whitney, D.E. (1982), "Quasi-static assembly of compliantly supported rigid parts", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, pp. 64-77.

[138] Whitney, D.E., De Fazio, T.L., Gustavson, R.E, Graves, S.C., Abell, T., Cooprider, C., Pappu, S. (1989), "Tools for strategic product design", *Preprints, Proceedings of the NSF Engineering Design Research Conference*, University of Massachusetts, Amherst, Mass., pp. 581-595.

[139] Whitney, D.E., Nevins, J.L., and CSDL staff (1978), "What is the remote center compliance and what can it do?", CSDL Report P-728, presented at the 9th International Symposium on Industrial Robots, Washington, D.C.

[140] Xiao, J., Volz, R. (1988), "Design and motion constraints of part-mating planning in the presence of uncertainties", *Proceedings of the IEEE Int. Conference on Robotics and Automation*, pp.1260-1268.