ON THE CONCURRENT DESIGN OF ASSEMBLY SEQUENCES AND FIXTURES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Bruce Romney
March 1997

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Jean-Claude Latombe
Dept. of Computer Science
(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Bernard Widrow

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Bruce Wooley

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

In this dissertation, I address the problem of automatically generating both an assembly sequence for a product and a fixture to hold all the intermediate subassemblies, given only a geometric description of the product. The fixture must stabilize every subassembly against gravity and against part-on-part insertion forces, without blocking any of the part motions. Unlike in previous assembly-planning work, the sequence and fixture are generated concurrently rather than sequentially.

I start by presenting an efficient system for the assembly-sequencing problem alone, which I developed jointly with other researchers. This system utilizes geometric-reasoning techniques to generate one or more assembly sequences and to analyze the complexity of the assembly, as measured by various figures of merit.

The introduction of fixturing to the problem improves the realism of the analysis, and allows physically unstable sequences to be eliminated from consideration. The joint problem of designing an assembly fixture concurrently with the sequence raises a number of new issues, which I explore. I present two methods to solve this problem. The first method is fast and successful in most (but not all) cases. The second method, based on the new concept of the insertion-force stability cone, is complete under certain assumptions, and is useful primarily when parts are assumed to be "glued" (or similarly attached) in place upon insertion. Both methods have been implemented for the case of planar assemblies, and experimental results are presented for each.

The long-term vision for this work is a computer-aided design tool which can provide immediate feedback to a product designer about the difficulty of assembling the proposed product, without the need to construct a prototype.

Acknowledgments

STAAT, the assembly-sequencing system described in Chapter 3, was developed jointly by myself, Michael Goldwasser, Cyprien Godard, and G. Ramkumar. It is largely an extension of the work of Randy Wilson [78, 81], who introduced the key concept of the NDBG data structure (described in Section 3.4) and showed how it could be used to reason about disassembly. Later, Goldwasser significantly improved upon this structure, and his improvement is incorporated in the present work.

The research presented in this dissertation has been supported by a National Science Foundation graduate research fellowship, by NSF Grant IRI-9306544, by a grant from the Stanford Integrated Manufacturing Association (SIMA), and by Army ARO MURI Grant DAAH04-96-1-0007.

I wish to thank my advisor, Prof. Jean-Claude Latombe, for his helpful guidance throughout my years at Stanford. As an electrical engineering student daring to do research in artificial intelligence and computational geometry, I have been enlightened by his perspective and wisdom, about how to turn a simple, "ad hoc" idea borne of an engineer's mentality into a truly elegant and interesting piece of work. For this newfound perspective, I thank him.

Thanks also to Randy Wilson and Sandia National Laboratories for the research opportunity which launched my main dissertation research. I entered Sandia not knowing what “avenue” of research to pursue (or even where to find the on-ramp!). I left invigorated with a new idea, and with the realization that I was finally able and ready to make my long-sought contribution to knowledge. Thanks especially to Randy, personally, for all his assistance during that summer.

I also thank my friend and research partner Cyprien Godard, for being a patient sounding board for some of my earlier ideas, both the good ones and the not-so-good ones. Our invigorating conversations often opened up possibilities and perspectives which I had not considered, and I felt, as strongly as with few other students, that working together we were truly greater than the sum of our parts.

Thanks are also due to General Motors and Boeing for their support of my research through several enlightening visits to those companies, and, in the case of General Motors, a summer internship at the General Motors Technical Center. I found it both helpful and interesting to “come down from the ivory tower” and speak to engineers about the practical problems, related to my work, currently being faced by industry.

Finally, I thank my family, who have always supported me and who gently instilled in me the values which continue to guide my life today. And I thank my friends at Stanford, who have made that life so much more enjoyable. Thanks especially to my dear friend Kristiana Kincaid, whose support and friendship have helped me through difficult times, made my joys more complete in joyous times, and brightened the times in between.

Contents

Abstract **v**

Acknowledgments **vii**

1 Introduction **1**

1.1 The Need for Automatic Assembly Planning 1

1.2 Concurrent Engineering 2

1.3 The Assembly Planning Problem 3

1.4 Scope and Place of this Dissertation 7

1.5 Different Approaches for Different Applications 8

1.6 Contributions 9

1.7 Outline 10

2 Previous Work **11**

2.1 Assembly Sequencing 11

2.2 Fixturing 15

2.3 Sequencing and Fixturing Combined 18

3 An Efficient Assembly-Sequencing System **19**

3.1 Preliminaries 19

3.2 Capabilities of STAAT 22

3.2.1 Restrictions 24

3.3 System Overview 26

3.4 The Non-Directional Blocking Graph 28

3.5 Disassembly Sequencing with the NDBG . . . 33
3.5.1 NDBG Traversal . . . 33
3.5.2 Building and Using the DBG . . . 34
3.5.3 Execution and Recursion . . . 36
3.6 Implementation . . . 37
3.7 Experimental Results . . . 37
3.8 Motivation for the Next Step . . . 38

4 The Co-Design Problem **41**
4.1 Problem Statement . . . 41
4.1.1 Insertion Forces . . . 42
4.2 Assumptions . . . 44
4.2.1 The Assembly Model . . . 44
4.2.2 The Fixture Model . . . 48
4.2.3 The Sequence Model . . . 56
4.3 Why Co-Design? . . . 57

5 Key Issues in Sequence/Fixture Co-Design **61**
5.1 Representing the Search Space . . . 61
5.2 Figures of Merit for an Assembly Plan . . . 64
5.3 Assembly Orientation . . . 65
5.4 Grasping Fixels . . . 66
5.5 No Upward Insertions . . . 67
5.6 Removing the Finished Assembly . . . 67

6 A Fast System for Co-Design **69**
6.1 System Overview . . . 69
6.2 The Fixture Continuum . . . 71
6.2.1 Virtual Fixels . . . 72
6.3 The Search Space . . . 73
6.4 Overfixturing . . . 74
6.5 Clearing a Final Removal Path . . . 75

6.6 Enumerating Feasible Actions 76
6.7 Verifying Stability for Actions 78
6.7.1 Basic Approach 78
6.7.2 Efficient Insertion-Force Stability Checking 82
6.8 Choosing an Action 85
6.9 Wrench Space and Convex Hulls 87
6.9.1 Convex Hull Complexity 91
6.9.2 Convex Hull "Shells" 91
6.9.3 Disconnected Assemblies 92
6.10 Experimental Results 93
6.11 Motivation for the Next Step 95

7 A Complete Algorithm for Co-Design 97

7.1 The Extended Translational Freedom Cone 98
7.2 The Insertion-Force Stability Cone 100
7.3 Sequencing for a Known Fixture 106
7.4 Exploring New Fixtures 109
7.4.1 A Simple Method 110
7.4.2 A More Practical Method 111
7.5 Extensions 113
7.5.1 Continuous Fixture Representation 113
7.5.2 Unglued Parts 120
7.5.3 Disconnected Assemblies 122
7.6 Computational Complexity 123
7.7 Experimental Results 127
7.8 Remarks 132

8 Conclusion 133

8.1 To the Next Step and Beyond 134
8.1.1 Future Work: An Overview 134
8.1.2 A Closer Look 136
8.2 Closing Remarks 141

Glossary of Key Terms **142**

Bibliography **149**

List of Tables

3.1 STAAT's running times. 38

7.1 Experimental results for Atlas 3 on various assemblies. 127

List of Figures

1.1 A high-level assembly plan for a Boeing 777 aircraft. 4

2.1 An assembly which is unstable under gravity, even though none of the parts would be unstable moving alone. 17

3.1 An assembly with no monotone two-handed assembly sequence. . . . 20
3.2 A non-linear assembly sequence. 21
3.3 An example of an AND/OR disassembly graph. 22
3.4 Typical display screens for STAAT 23
3.5 AND/OR graphs for the bell assembly. 25
3.6 The 42-part model aircraft engine assembly. 25
3.7 STAAT's basic operating procedure. 27
3.8 A single contact partitions the motion sphere into three regions. . . . 29
3.9 When there are multiple contacts between parts, the regions of local freedom are narrower. 30
3.10 An example of a local translational freedom (LTF) cone. 30
3.11 Adding a third part subdivides the sphere even further. 30
3.12 Within each region of the sphere, the blocking relationship is constant. 31
3.13 The non-directional blocking graph for the simple three-part assembly. 32
3.14 The NDBG sphere for the 42-part engine assembly. 32
3.15 NDBG vertices are not necessarily the directions of maximum *global* freedom 34
3.16 The Reduced Directional Blocking Graph. 35
3.17 Sweeping operations. 36

4.1 An assembly operation which is unstable due to insertion forces. . . . 42
4.2 Insertion forces can occur at any contact point(s), and in any nonnegative combination of magnitudes. 43
4.3 Examples of insertion forces not considered here. 43
4.4 An assembly which should be inverted and re-fixtured. 50
4.5 An assembly on which a nonlinear step should be performed. 51
4.6 Two examples illustrating the difficulties of correctly handling frictional assemblies. 54
4.7 Another assembly whose stability is indeterminate. 54
4.8 The dependence of sequencing on fixturing. 58
4.9 The dependence of fixturing on sequencing. 59
4.10 Even intuitive sequencing heuristics sometimes fail. 60

5.1 A fixture whose validity depends upon the sequence used to produce the subassembly. 62
5.2 An example of a disassembly *tree*, with fixture information included in the nodes. 63
5.3 Representing the search space as multiple graphs. 64
5.4 A structure which cannot be assembled in its current orientation without the use of grasping fixels. 67

6.1 Atlas' basic operating procedure. 70
6.2 An example of a continuous fixture representation. 71
6.3 The creation of virtual fixels. 72
6.4 Sometimes, there may be an infinite number of distinct children. . . . 74
6.5 An example showing the need to fixture a part's extremities. 75
6.6 Atlas' procedure for enumerating feasible actions. 77
6.7 A sweeping operation which shows how to modify the fixture. 78
6.8 An assembly for which not all contact points require explicit IFS checks. 84
6.9 Insertion forces at two "aligned" contact points will exert the same torque. 84

6.10 An insertion in which only four of six contact points will generate distinct torques. 85
6.11 For 3D assemblies, IFS can be verified along convex hull of the projected contact points. 86
6.12 The "tilted blender" example. 88
6.13 Wrench-space representation of the stability condition. 89
6.14 The wrench-space representation of Fig. 6.12b. 90
6.15 The "tilted blender" assembly in its final fixture. 93
6.16 The wrench-space representation of Fig. 6.15. 94
6.17 The A$'$ Discriminator example, in its final fixture. 94
6.18 The "box-lid-cargo" example, in its final fixture. 95
6.19 The fan example, in its final fixture. 96

7.1 An example of an extended translational freedom (ETF) cone. 98
7.2 ETF's are much simpler to compute in the planar case. 99
7.3 Another model of possible insertion forces. 103
7.4 Examples of IFS cones. 105
7.5 Sets of stable, fixture-preserving directions of motion. 107
7.6 The procedure to find all stable, fixture-preserving actions. 107
7.7 The procedure to find a stable disassembly sequence for a given fixture. 108
7.8 An example of a pointless fixture reduction. 110
7.9 Equivalence classes of motions, from the projected fixels. 112
7.10 The procedure to find all stable, fixture-preserving actions while also keeping track of all useful fixture-reducing actions. 114
7.11 The procedure to find a stable disassembly sequence for a given fixture, or, finding none, to suggest new fixtures to explore. 115
7.12 The top-level procedures to perform co-design, for discretized fixtures. 116
7.13 A discrete fixture representation, complemented by additional, continuous-fixture information. 117
7.14 The procedures to perform co-design for a continuous fixture. 119
7.15 A part removal from an already disconnected assembly. 122

7.16 A part removal which breaks the assembly into two disconnected components. 123
7.17 Even a small assembly can have a large assembly graph. 124
7.18 The solution for the tilted-blender example. 128
7.19 The "box-lid-cargo" example, in its final fixture. 129
7.20 The A′ Discriminator example, in its final fixture. 130
7.21 The fan example, in its final fixture. 131
7.22 An example solvable by Atlas 3 but not by Atlas 2. 132

8.1 Fixturing the flat end of a cylinder. 138
8.2 Fixturing the curved side of a planar part. 139
8.3 Sometimes, virtual fixels on curved parts only approximate the fixturing capability of a continuous fixture. 140

Chapter 1

Introduction

1.1 The Need for Automatic Assembly Planning

In a world in which manufactured products are subject to ever more demands from the marketplace and from governmental regulation, computer-aided design systems have become an indispensable tool to make the product design process a manageable one for human engineers.

Early mechanical CAD systems were little more than electronic draftboards and numerical analysis machines, providing some part-interference information but leaving much of the advanced analysis to the designer. Over the years, though, as geometric modeling and related CAD technology have improved, CAD systems have been able to provide increasingly sophisticated feedback to the designer, not only about the product's functionality but also about its manufacturability and serviceability. These advances relieve designers of the need to resolve many mundane details, allowing them to devote more attention instead to higher-level considerations. The software representation of a product is gradually becoming more and more a "virtual prototype", providing many of the same insights, and more, that the construction of a physical mockup provided in the past, but at a fraction of the cost. Still, at present, there remains much room for improvement.

One area in which CAD tools have yet to realize their full potential is the analysis of the assembly process. In many industries, the product development process is characterized by insufficient consideration of the assembly process by design engineers; as a result, difficulties in assembling a product are often not discovered until a mockup or a prototype is constructed. This is especially true when a designer working on one subsystem makes changes which inadvertently affect the assemblability or insertability of another subsystem. It would be helpful if the CAD system used to design a product were able to provide quick feedback regarding the difficulty of assembling it. This feature would shorten the cycle time between the enactment of design changes and the discovery of their effects on assemblability. Such a CAD system would also be useful in such diverse areas as robotic assembly, service and maintenance, the production of "do-it-yourself" instructional manuals and videos, and part retrieval and recycling. Furthermore, by imparting to the system some notion of the costs and requirements of different assembly steps, a manufacturer could use it to optimize the layout of a factory's assembly lines.

It is toward the realization of this vision that the research efforts described in this dissertation are directed.

1.2 Concurrent Engineering

Another practice which can speed up the design cycle is that of concurrent engineering — that is, the solving of design problems concurrently rather than sequentially. When multiple subproblems need to be addressed over the course of the design process, a company which is able to consider them concurrently will be rewarded with a quicker time-to-market than one which solves them one at a time. Examples of these subproblems might include product performance, assembly sequencing, tooling, factory layout, and/or planning for service and maintenance.

However, the interrelationships among these subproblems may be complex and difficult to characterize, and managing the propagation of constraints among them represents an enormous challenge. For example, one cannot design tools without knowing the shapes of the parts; the part shapes, in turn, depend upon a host of other constraints such as the feasibility of assembly, the manufacturing cost, and the ease of servicing. In order to permit a concurrent approach, the interdependent design constraints among the various subproblems need to be explicitly identified, and the relevant information propagated efficiently among the human or automated planners. The difficulty of this task partly accounts for the widespread use of largely sequential design methods in the past.

Computers, with their power to process large amounts of information quickly, offer the possibility of greatly facilitating the use of concurrent design. Now, certainly, much work remains to be done and many of the same fundamental problems still apply, particularly that of characterizing the constraints imposed on one subproblem by the decisions made to solve another. However, if new algorithms were developed which treat several aspects of the design problem as a unified whole, sophisticated CAD tools such as the ones envisioned earlier could disseminate more useful information to designers than would narrower, single-purpose systems. That is, if the feedback to designers described earlier were not only immediate but also comprehensive, the design process would be streamlined enormously.

In order properly to understand the scope and place of the present research in the pursuit of this long-term goal, it is instructive first to examine the problem of assembly planning in greater detail.

1.3 The Assembly Planning Problem

"Assembly planning", generally speaking, refers to the planning of any or all aspects of the assembly process, by engineers and/or an automated system. Like many planning processes, assembly planning is usually a hierarchical process, beginning with a broad picture of the overall assembly plan and gradually including more and more details. Now, clearly, the more complete and realistic the assembly plan, the more easily the manufacturer will be able to anticipate difficulties in assembling a product; but in the early stages of the design process, not all details are available. Still, even in these early stages, some assembly analysis is possible (for example, by treating the parts as free-flying objects in space), and this analysis may provide feedback to guide further development of the product and of the assembly plan.

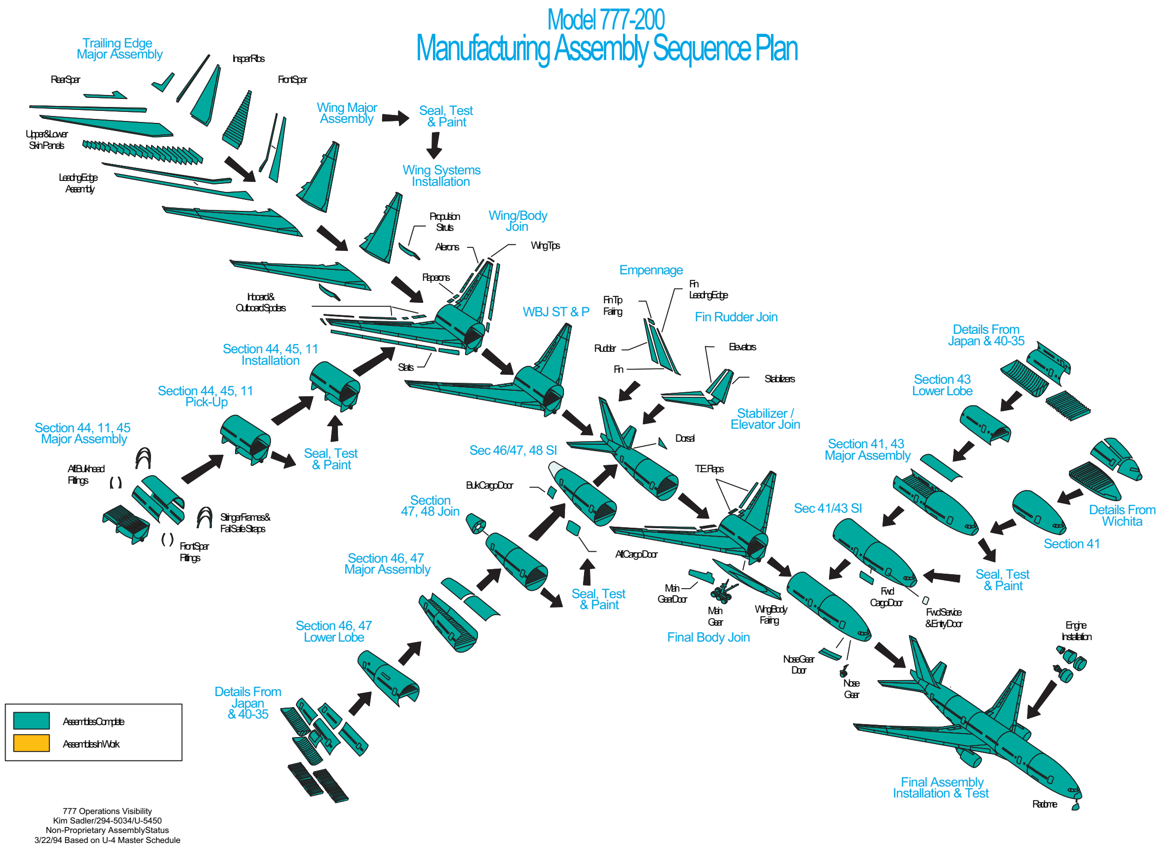


Figure 1.1: An example of a high-level assembly plan. *(Courtesy The Boeing Company.)*

An example of an actual high-level assembly plan, for a Boeing 777-200 aircraft, is shown in Fig. 1.1.

As the assembly plan moves through successive "levels of approximation", more and more aspects of the plan will be mapped out. Among the possible elements to consider in the plan are the following:

- **Sequencing.** Perhaps the most fundamental requirement of a successful assembly process is that the parts not collide with one another. Not surprisingly, then, a great many papers in the automatic assembly-planning literature focus on the particular problem of assembly *sequencing*[1], which is to find the order

[1] Many of the key terms used in this dissertation also appear in a glossary at the end.

and direction of part or subassembly insertions which will assemble the product in a collision-free manner.

Even though sequencing is but one aspect of assembly planning, it is still possible to produce a number of "complexity measures" to evaluate the assembly, based on the sequencing analysis alone. These might include, for example, whether the assembly can be built up one part at a time, whether all the insertions can be performed in the same direction, or the number of parts which must be removed to access and service a given "key part". (These figures of merit are discussed at greater length in Section 3.2.) In an assembly-planning CAD tool, such measures could be calculated and fed back to the product designers for further evaluation.

- **Fixturing.** During the assembly process, something must hold the product in place while parts are inserted, to prevent it from falling down or falling apart. This *fixture* may be a human hand, a flat table, or a collection of pins, clamps, and similar structures. In this last case, the mechanical fixture may be either modular and reconfigurable, or custom-designed for a particular product. Over the course of the assembly sequence, there may, in fact, be many different fixtures used to hold different subassemblies.

  Even before the means of fixturing a product has been chosen, however, an analysis of possible fixturing *locations* on that product can provide valuable information to a team of designers, by ruling out assembly sequences which cannot be stabilized. Physical stability is another fundamental requirement of an assembly plan, so these inferences can be made without committing the designers to a particular workcell configuration or fixturing system.

  A fixturing analysis can also yield additional figures of merit for the product, such as the number of fixturing locations, or the maximum amount of fixturing force required at any point. (See Section 5.2 for more information about figures of merit derived from *both* sequence- and fixture-planning.)

- **Tooling.** Tools are the instruments, such as screwdrivers, riveters, and weld guns, which are necessary to perform the various assembly steps. They may be standard (such as a wrench), or custom-designed for a a particular product (such as the "jigs" which grasp aircraft fuselage panels). Even though the cost of custom tools may be amortized over many manufactured units, the need for and cost of such tools are one possible factor when evaluating a proposed product design. Additionally, the space requirements of the necessary tools may affect the generation of a valid assembly sequence.

More broadly, the manufacturing system used to assemble the product is an integral part of the assembly plan. For example, if robot arms are to be used, then their dexterity and range of motion will have a substantial impact on the validity and/or optimality of other aspects of the plan. Conversely, constraints from the developing assembly plan may challenge or re-affirm the wisdom of choosing a particular manufacturing system.

- **Assembly Functionality.** Non-geometric constraints such as those deriving from the product's intended behavior may affect the assembly plan. For example, auto fuel should not be added until welding is complete; magnetically-sensitive components should not be brought near strong magnetic fields; and delicate instruments should not be subjected to sudden jolts. The engineer or computer system generating the assembly plan should include such considerations in the plan.

- **Factory Layout.** Will all the required equipment fit into a certain space? How will it be arranged? Is it possible and/or desirable to move the production of some subassembly into another facility? Since the answers to questions like these substantially affect the cost of producing a product, an analysis of them would certainly improve the usefulness and realism of an assembly plan.

- **Ergonomics.** Today, even in highly automated industries such as the automobile industry, human assembly workers are essential. Along with the need to employ human workers comes the need to ensure the feasibility and relative comfort of their proposed tasks. Difficult or uncomfortable operations should be avoided (or automated), and easy, convenient operations promoted. A comprehensive assembly plan should include these factors, as well.

This list is by no means all-inclusive; certainly other aspects of the assembly process exist which also have a significant impact on product cost, and therefore merit analysis by an assembly planner.

1.4 Scope and Place of this Dissertation

However, in practice, an automatic system which considers all these aspects of assembly planning is not yet a reality. Nor is it my intent in this dissertation to present such a system. Rather, I address only the first two aspects of the problem, sequencing and fixturing. Other aspects of assembly planning, while certainly important, are left to future research efforts.

Still, it should be noted that the basic principle used here generalizes to other aspects of the problem as well. The rule, simply stated, is that one should never make changes in one aspect of the assembly plan which render the plan *as a whole* infeasible. For example, in the case of a simple assembly sequencer, the system should never plan a part motion which is geometrically infeasible. If a fixture is being generated as well, then the system should also never plan a motion which causes instability under the proposed fixture, nor should it plan a fixture change which destabilizes some subassembly in the sequence. Applications of this principle to other aspects of assembly planning are also not difficult to envision. In the case of tooling, for example, one should never plan a part removal for which the necessary tool doesn't have enough room; this principle was applied by Wilson [79] in his treatment of the combined sequencing and tooling problems.

It is also instructive to note that while this research addresses the problem of fixturing an assembly, it does not address the related problem of grasping the moving parts during insertion operations. Grasping and fixturing are closely-related problems. However, at least for purposes of this dissertation, fixturing is a "passive" process in which the fixture elements resist external forces but do not otherwise react to the presence or absence of an assembly; grasping, on the other hand, is an "active" process in which the gripper may (and often does) dynamically adjust its grip in response to real-time conditions. The design of such "active" elements is beyond the scope of this research.

Assumptions about the precise nature of the sequences and fixtures used in this dissertation are presented in detail in Chapter 4.

1.5 Different Approaches for Different Applications

Even within the context of automatic sequencing and/or fixturing tools, a number of different approaches are possible. For example, some methods described in the literature identify specific assembly features, such as screw holes and slots, and generate sequences or fixtures by applying rules to the feature-based model [24, 67]. Other approaches apply more general geometric reasoning to the overall shapes of the parts, producing assembly plans from physics-based criteria for collision avoidance and/or stability [10, 27, 56, 78, 80]. Still other methods, such as the well-known Boothroyd-Dewhurst design-for-assembly criteria [8], estimate the product's assembly cost without generating an explicit assembly plan. Feature-based methods and "planless" methods have the advantage of being able to incorporate human expertise and understanding directly into decision-making; more general approaches, however, are sometimes able to catch subtle geometric problems — or solutions — that might escape the notice of feature-based systems, and of their human users. The analyses presented in this dissertation take the latter approach, delivering feedback to the designer based on an assembly plan generated through a uniform, systematic analysis of the assembly's geometry.

Another way in which assembly-analysis systems may be classified is by whether they operate "off-line" or "on-line". If the system is intended to analyze a proposed product overnight, while the designers are sleeping, then a far more thorough and exhaustive analysis is possible than if feedback to a product designer is required immediately upon the enactment of a design change. For example, the "off-line" system of Chang and Li [15] sometimes requires hours of CPU time, but in at least one case was able to plan the removal of a complex engine part along a disassembly path previously unknown to the engine's designers. At the other extreme are systems such as STAAT and Atlas, described later in this dissertation. They are viewed as forerunners of an eventual *interactive* CAD tool, one requirement of which is the quick completion of the product analysis, even though that analysis may necessarily be less complete than one would ideally desire.

Still another characteristic of assembly-planning systems is whether they plan for robotic assembly, plan for manual assembly, or make no assumptions at all about the manner of assembly. For example, a robotic planner, even if it does not generate an explicit robot instruction sequence, must at the very least consider the constraints inherent to robotic-assembly operations, such as whether a part insertion is consistent with the robot's degrees of freedom, and whether the operation is tolerant of imperfect geometry. This dissertation, however, does not assume any particular means of assembly, instead focusing on assembly constraints that arise purely from the geometry of the assembly itself.

1.6 Contributions

In this dissertation, I present the following contributions to automated reasoning about assembly planning:

An Efficient System for Geometric Sequencing and Analysis. A software system for geometric assembly sequencing and analysis is presented, which I developed jointly with three other researchers. This system, called STAAT, implements and validates some of the assembly-sequencing concepts and algorithms presented in earlier publications [78, 81]. STAAT is able to generate an assembly sequence efficiently, compute complexity measures based on the set of all possible sequences, and identify the sequence which optimizes a particular complexity measure.

Issues in Sequence/Fixture Co-Design. The problem of generating an assembly sequence and an assembly fixture concurrently, rather than sequentially, is considered.

To my knowledge, no previous research has systematically addressed this question, despite the close interrelationship between the sequencing and fixturing problems. This "co-design" approach raises a host of new issues, and substantially increases the importance of older ones. These issues are described and explored in this dissertation.

A Fast System to Perform Co-Design. A fast algorithm to perform sequence/fixture co-design is presented. This algorithm has been implemented for the case of planar assemblies, and experimental results are presented.

A Complete Algorithm for Co-Design. A method is presented to characterize, in closed form, the complete set of stable actions which will disassemble a subassembly in a given fixture. Under certain conditions, this characterization can be performed in polynomial time. These results give rise to an efficient algorithm to perform sequence/fixture co-design, which is guaranteed to find a solution (of a certain class) when one exists. This algorithm has been implemented as well, and experimental results are presented.

1.7 Outline

The remainder of this dissertation is organized as follows. Chapter 2 presents a review of the relevant past work in sequencing and fixturing. Chapter 3 describes the STAAT system for assembly sequencing and analysis, its algorithms, and its performance.

Chapter 4 then formally poses the problem of sequence/fixture co-design, and discusses at length the assumptions made in this dissertation. Chapter 5 presents some of the key issues raised by the co-design approach, and suggests possible ways to resolve them. Chapter 6 then presents a practical algorithm to perform sequence/fixture co-design, and an implementation of that algorithm. Chapter 7 develops this approach further by establishing the completeness results mentioned earlier, and incorporating them into a new system with performance guarantees. Experimental results for this new system are also presented.

Finally, Chapter 8 proposes some directions of possible future research and concludes. A glossary of the key terms used herein follows.

Chapter 2

Previous Work

2.1 Assembly Sequencing

Many of the early assembly-reasoning systems were interactive ones, querying the user for geometric-reasoning information and generating assembly sequences from the answers. The classic system of Bourjault [9] posed a large number of yes/no questions to the user; De Fazio and Whitney [19] later modified this approach to require a smaller number of questions with more complex answers. Since then, a number of methods and systems have been developed in which the geometric constraints on assembly are supplied as input, either through the CAD database or at run-time by the user [4, 43, 46]. These approaches set aside the key problem of deriving the geometric constraints, focusing instead on the efficient representation and processing of these constraints into assembly sequences. Although they would be unsuitable for a CAD tool which quietly monitors a developing product's assemblability, these methods might well be suitable for other applications; for example, Baldwin *et al.* [4] described an interactive tool allowing a designer to examine and edit assembly sequences manually.

The problem of automatically generating sequences from an assembly's geometry alone is, in its full generality, PSPACE-hard [59, 83]. Even when every operation is constrained to move exactly two subassemblies into their final relative positions, the problem has still been found to be NP-complete in both the two-dimensional [40] and three-dimensional [39] cases. As a result, much of the past and present work in this area focuses on even more restricted variants of the problem.

One common thread that appears in much of the assembly-sequencing literature is the strategy of "assembly by disassembly", in which an assembly sequence is generated by starting with the completed product and working backwards through disassembly steps. Assuming that the parts are rigid and non-toleranced, the disassembly sequence can then be reversed to produce a valid assembly sequence.[1] This strategy is motivated by the observation that a product in its fully-assembled state exhibits far more motion constraints on its parts than in its disassembled state; the assembly-by-disassembly approach, therefore, drastically reduces the range of assembly motions which a sequencer must consider. Other methods do exist, such as that of Chakrabarty and Wolter [13], which builds an assembly plan "bottom up" in terms of the subassembly hierarchy. However, the assembly-by-disassembly strategy is very widely used in the literature, and it is a strategy that will be employed in this dissertation, as well.

A number of systems have been developed to perform assembly sequencing using motions only along the major axes [31, 48, 57, 71] or along pre-specified trajectories [13, 32, 36, 54]. Hoffman [31, 32], in particular, created a system which handles curved surfaces as well as flat ones, and can move parts into intermediate rather than final relative positions when the latter are not immediately reachable (heuristics are used to choose these intermediate positions).

Another simplification used to avoid the NP-completeness of the more general case is to insert only one part at a time [84, 86]. Woo and Dutta [86] created such a system which runs in $O(n \log n)$ average time, where n is the number of faces in contact. To do this they used the notion of a "disassembly tree", showing what parts in the assembly must be removed before what other parts. They also showed how their method could be used to find the optimal sequence to access and remove a given part, for a restricted class of assemblies.

[1] In the larger co-design problem, assembly and disassembly sequences are not actually inverses of one another, because of part-on-part forces which are different in the two cases. However, it is still possible to generate an assembly sequence by working backwards from the completed product. (See Section 4.1.)

Using computational-geometry approaches, Pollack *et al.* [62] wrote an algorithm to separate two polygons by a sequence of translations; and Valade [76] wrote one to separate two three-dimensional polyhedra by multiple translations. Not surprisingly, perhaps, the latter algorithm is quite slow (1 hour of CPU time for the example provided). Arkin *et al.* [2] devised a way to partition polygons into two sets which are separable by a single translation, by identifying the boundary line between them; however, the extension of this method to the three-dimensional case is not apparent.

Wolter [84], Homem de Mello and Sanderson [34], and Lin and Chang [52] each used a basic approach similar to that of the STAAT system of Chapter 3. Directions of possible part removal are suggested by contact information between the parts (or sometimes by the types of parts, *e.g.,* screws). The feasibility of the motions is then determined by "sweeping", or projecting, the parts in the proposed direction.

NDBG-Based Approaches

However, STAAT uses a much more efficient data structure to process the geometric information than do these systems: namely, the *non-directional blocking graph* (NDBG) developed by Wilson [78, 81]. This structure, described in detail in Section 3.4, partitions the continuous space of part motions into a finite and manageable number of equivalence classes with respect to part-part collisions, and thereby represents in compact form all of the blocking relationships among the parts in an assembly, for the given class of part motions. This is a much more efficient scheme than the "generate-and-test" approaches seen elsewhere in the literature [34, 44, 56], which enumerate all possible subassemblies (of which there may be an exponential number), and check each one for removability.

STAAT uses the "local" version of the NDBG, which describes constraints on infinitesimal translations (including an improvement to this structure devised by Goldwasser [65].) A much more powerful version, however, is the "extended" NDBG, which describes the blocking relationships for part translations to infinity. Based on the extended NDBG, Wilson and Latombe [81] presented a polynomial-time algorithm to generate two-handed, monotone assembly sequences of straight-line part insertions in any direction, whenever such sequences exist. They also proposed a number of measures of assembly complexity; a CAD tool such as the one described in Chapter 1 could use such measures to quantify the difficulty of the assembly process.

In [23], Goldberg and Moradi used the NDBG to generate "vertical assembly" plans, consisting of vertical part insertions and assembly re-orientations, with the goal of producing plans which can be executed by a robotic assembly system. Kaufman *et al.* [38] took this approach further and developed an impressive NDBG-based system which is able to take commercial CAD data describing a product, generate an assembly sequence for it, and translate that sequence into actual control code for robots to perform assembly.

Now, for sequences which insert one part at a time, such as those used in Chapters 6 and 7, building the full NDBG is not necessary. Instead, it suffices to consider the *local translational freedom cone* [33] or *extended translational freedom cone* [66] for each part. (These cones are, in fact, the basis elements from which NDBG's are constructed.) Krishnan and Sanderson [44] also use these cones in their disassembly analyses, but under a generate-and-test scheme to remove multi-part subassemblies.

Beyond the Basic NDBG

Since the development of the NDBG, a number of publications have extended this concept to more general situations. Latombe *et al.* [47], for instance, applied the NDBG to the case of toleranced assemblies. They developed the notions of the *strong* and *weak NDBG's*, and presented a complete algorithm to generate an assembly sequence of single-step, infinite translations which successfully assembles the product under any instances of the part dimensions. If no such sequence exists, a second algorithm generates assembly sequences which will work under some instances of the part dimensions.

In another extension, Wilson [79] integrated the NDBG approach with considerations of tooling requirements, resulting in efficient assembly-planning algorithms which ensure the existence of enough free space to apply the appropriate part-joining tools.

Several researchers have applied the larger concept of partitioning the space of part motions to encompass motions other than single-step translations. Guibas *et al.* [25], for example, developed and implemented a complete and efficient method to identify infinitesimal *generalized* motions with which to partition an assembly, by identifying and examining only those directions of generalized motion ($\in \Re^6$) that are relevant to the partitioning problem. Lozano-Pérez and Wilson *et al.* [53, 80] developed the concept of the *interference diagram* (related to the NDBG) to analyze arbitrary classes of part motions, later implementing it for the special cases of single-step translations and infinitesimal generalized motions. Finally, Halperin and Wilson [26, 27] applied the concepts of the NDBG and interference diagram to the problems of partitioning a planar assembly with two translations or with a constant number n of translations, producing algorithms which run in polynomial time with respect to the size of the assembly.

2.2 Fixturing

Part Fixturing

There is a great deal of literature on the fixturing or grasping of a single part, or "workpiece". (Literature surveys on this subject include [28, 29, 73].) Much of this work focuses on establishing complete kinematic closure on a part — that is, the complete immobilization of that part — or testing an existing fixture for this condition. Many authors, such as [3, 7, 10, 58, 72, 74, 87], neglect the stabilizing effects of friction for reasons of simplicity and conservatism, while others [21, 51, 60] do include friction. Although there is no universal agreement on terminology, the condition of an object being completely immobilized by virtue of the fixture's geometry alone, without friction, is often referred to as *form closure*; when an object's motion is prevented by frictional elements applying external forces, the condition is called *force closure*. In [58], Mishra *et al.* derived upper bounds on the number of fixture elements required to hold a part in form closure, and presented linear-time algorithms to generate those elements' positions.

Recently, Rimon and Burdick [64] described another fixturing condition based on second-order geometric effects. As they correctly pointed out, some parts, such as an equilateral triangle grasped at each edge's midpoint, may be able to move infinitesimally, but not by any finite amount. Sudsang *et al.* [72] applied this less restrictive notion of immobility to the design of frictionless fixtures for polyhedra, which use only four fixture elements (as compared to the seven required for ordinary, first-order form closure [45]).

One trend in the design of part fixtures has been the use of modular or reconfigurable fixture elements [3, 10, 67, 69, 77]. Because such elements can be re-used on many different parts, they are often more desirable than expensive custom fixtures, especially for low-volume products. Costs can be even further reduced if fixture is designed for robotic reconfiguration, as in [3, 69].

Notable among the modular-fixturing algorithms is that of Brost and Goldberg [10], which designs a form-closing fixture for a planar part, using three locators and one clamp placed on a regular lattice of possible sites. The algorithm is able to enumerate all possible form-closing fixtures of this type, and is therefore not only complete, but also able to find the globally optimum fixture, according to any metric supplied by the user. Possible figures of merit for fixtures are suggested in [12, 21, 51], and might include the total reaction force required for a fixture to oppose a certain external force, the maximum reaction force by any one element, or the total or maximum reaction forces to *any* possible external force (of a known magnitude).

Nguyen [60] devised a grasp-synthesis algorithm which specifies grasps as independent regions on the part's boundary, such that grasping 'fingers' can be placed anywhere in those regions and always establish form or force closure.

In work that complements these efforts, Whybrew and Ngoi [77] focused on the subsequent problem of translating the set of fixturing locations on the workpiece into an arrangement of actual fixture elements which implement that solution. They devised a software system to do this with an arrangement of modular fixture elements placed on crisscrossed tracks on the baseplate.

In the context of fixturing alone, the establishment of form or force closure is a reasonable goal to pursue, since one may not know *a priori* what external forces to expect. However, fixturing in the context of sequence/fixture co-design is a different problem. In this case, a form-closing fixture is not necessarily desirable, because it may well be too restrictive to allow part removals or insertions. Instead, one merely needs to establish stability against gravity and the part-on-part *insertion forces* (described below in Section 4.1), while still admitting free insertion paths for

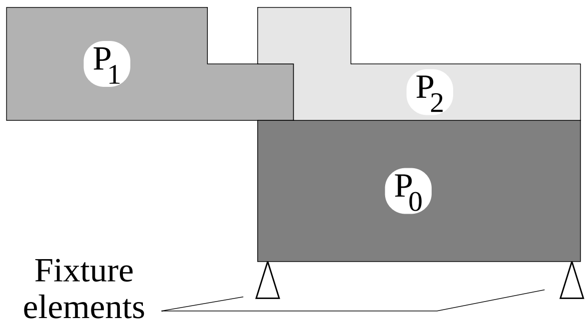


Figure 2.1: An assembly which is unstable under gravity, even though none of the parts would be unstable moving alone.

the parts. Fixturing against known external forces has been studied before [16, 50, 56, 58, 74], especially in the context of fixturing for machining [16, 50]. Lee and Yi [49] even applied the concept of stability against part insertion forces in their analysis of multipart assemblies, although not in the context of a fixturing algorithm.

Assembly Fixturing

The field of assembly fixturing, in which all parts may move independently, is far less represented in the literature than that of part fixturing. Some early work was done by Fahlman [20] for structures of simple blocks. However, Fahlman himself acknowledged some of his stabilization techniques to be "a loose grouping of a very large number of essentially independent tricks".

Some authors [68, 81] have handled assembly fixturing or stability based on the instability of individual parts moving alone. While this is certainly a necessary condition for stability of the assembly as a whole, it is not a sufficient one; for example, the assembly of Fig. 2.1 is unstable under gravity, even though none of the parts would be unstable moving by itself.

The first systematic method to fixture an assembly for true stability was developed by Wolter and Trinkle [85]. Based on the work of Palmer [61] and Trinkle [74], they presented two systems of linear inequalities called the *velocity* and *force formulations*, which are duals of each other in the linear-programming sense. The velocity formulation (also used by Mattikalli *et al.* in [55]) describes the simultaneous motion of the parts so as to lower overall potential energy without violating non-penetration constraints. The force formulation describes the balance of forces and torques between the fixture elements and a known external force (interacting through the part-part contacts). Assembly stability can be verified by solving one of these systems of inequalities with a linear program. Based on this method, Wolter and Trinkle proposed a fixture-synthesis algorithm which scatters fixture elements all over the assembly and then eliminates those that turn out to be unnecessary. This algorithm was the inspiration for the fixturing algorithms presented in Chapters 6 and 7 of this dissertation.

Mattikalli [56] extended this technique to characterize the set of all gravitationally-stable orientations of an assembly, under a given fixture. He also proposed several fixture-synthesis algorithms, including one which builds up a fixture one element at a time, choosing each element based on the orthogonality of its applied force and torque to those of the previously-selected elements. Baraff and Mattikalli [6] showed that finding the smallest set of fixture elements to stabilize an assembly under gravity (or any other given force) is NP-hard.

2.3 Sequencing and Fixturing Combined

As useful as these results are, finding the optimal fixture for a single assembly is not necessarily the most desirable goal from a manufacturing standpoint, if that fixture does not stabilize related subassemblies. Pulling an assembly out of a fixture and placing it in a new one is likely to be expensive, so it would be far better to design a fixture to stabilize the product throughout the assembly process (or at least, for as many stages as possible). However, a review of the literature reveals that no one has yet described a systematic way to do this, let alone generate the sequence as well.

The research presented in this dissertation, then, represents the first attempt to both generate an assembly sequence and fixture that sequence, under a "unified" approach which recognizes and utilizes the interrelationships between fixturing and sequencing.

Chapter 3

An Efficient Assembly-Sequencing System

In this chapter, I present the Stanford Assembly Analysis Tool (STAAT), a software system able to perform automatic assembly-sequence generation and complexity analysis, given only a geometric description of the proposed assembly [65]. STAAT generates sequences with single-step translations, and within the context of such sequences, represents a significant advance in speed and efficiency over previous work. The underlying theoretical framework for STAAT extends to other classes of assembly operations as well, including one-step translations with rotations, n-step translations, and simultaneous independent motions by multiple subassemblies.

3.1 Preliminaries

Before discussing STAAT in detail, it is useful to define the following terms:

- The *number of hands* required to perform a given step in an assembly sequence is defined as the number of subassemblies that are moving with respect to one another. For example, if a single subassembly is being removed from the rest of the assembly, then two hands are required: one for the moving subassembly, and one for the fixed subassembly. If the fixed subassembly is merely resting on a table, then, the table acts as a "hand" in this context.

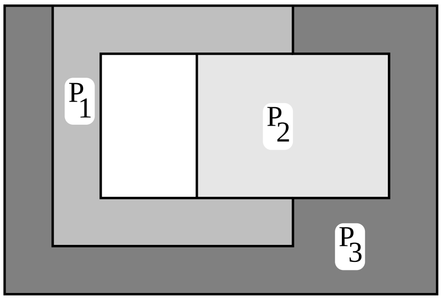


Figure 3.1: An assembly with no monotone two-handed assembly sequence [83].

- A *two-handed* assembly sequence is one in which each step requires only two hands, *i.e.,* no step requires three or more subassemblies to have simultaneous relative motion.

- A *monotone* assembly sequence is one in which each subassembly, once constructed, is final; that is, it is not modified by subsequent operations. For example, the latch assembly in Fig. 3.1 has no monotone two-handed assembly sequence. The only possible two-handed assembly sequence for this structure requires inserting P_2 all the way into P_1, inserting that subassembly into P_3, and then partially removing P_2 again. Because this last step changes the P_1-P_2 subassembly, the sequence is non-monotone.

- A *linear* assembly sequence is one in which each step involves the insertion of a single part into the rest of the assembly. The assembly sequence shown in Fig. 3.2, for example, is *not* linear because the step represented by the long arrow requires the insertion of a two-part subassembly into a three-part subassembly. An assembly for which a linear assembly sequence exists is called a *linear assembly.*

- A *stack* assembly sequence is one in which all the motions occur in a single direction, usually up-and-down. Circuit boards, for example, often have stack assembly sequences. An assembly for which a stack assembly sequence exists is called a *stack assembly.*

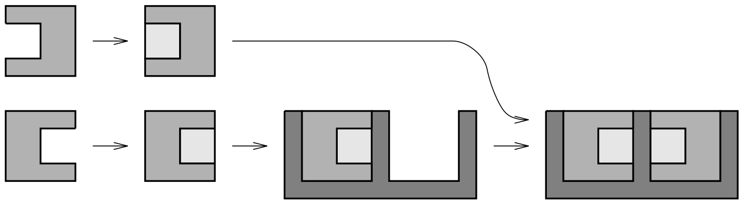


Figure 3.2: A non-linear assembly sequence.

Monotone assembly sequences can be represented using an AND/OR *disassembly graph* [35]. Each node in this graph represents a subassembly, and each AND'ed group of children for that node represents a set of smaller subassemblies into which the given subassembly can be decomposed. When there are multiple ways to break apart a subassembly, there are multiple sets of children OR'd together. For the usual case of two-handed assembly sequencing, then, each node in the AND/OR disassembly graph will have its AND'ed children in pairs. Fig. 3.3 shows an example of such a disassembly graph. The AND/OR structure captures the necessity of recursively disassembling all of the subassemblies resulting from a partitioning in order to completely disassemble the parent assembly.

The motions of the subassemblies can also be classified in several important ways. *Local* motion is motion over an infinitesimal distance. Although such motions are never used in a final assembly sequence, the concept of local motion is important because it gives rise to that of *local freedom*. A subassembly is said to be *locally free* if it can move an infinitesimal distance without colliding with other parts; this is a necessary condition for the subassembly to be removed. The frequent absence or limitation of local freedom is a characteristic which distinguishes assembly sequencing from other types of motion planning, and, consequently, a characteristic commonly exploited by assembly-sequencing algorithms to generate sequences efficiently.

Global or *extended* motion is motion over an arbitrarily long distance. A subassembly is said to be *globally free* if it can be removed to infinity. A *single-step translation* is a type of extended motion in which the subassembly moves to or from infinity in a single, straight line without rotating.

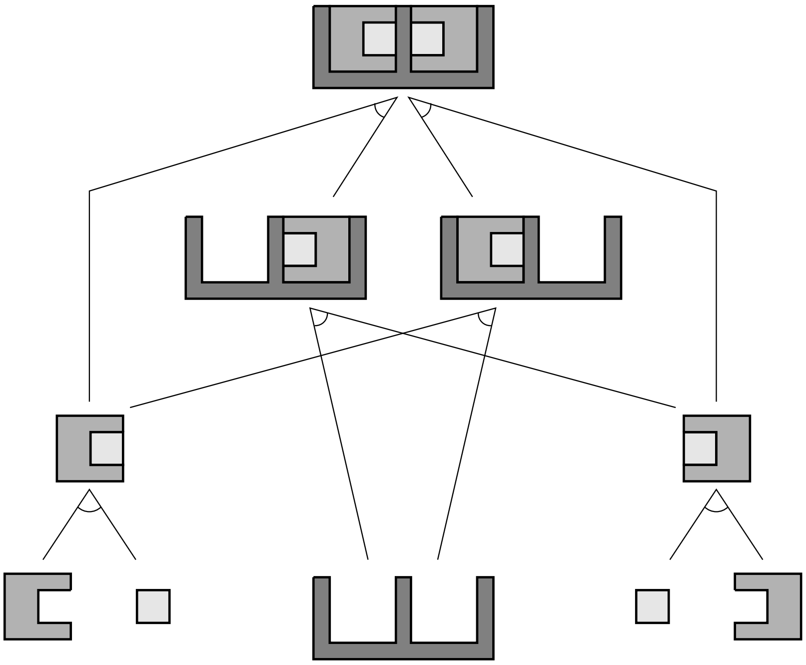


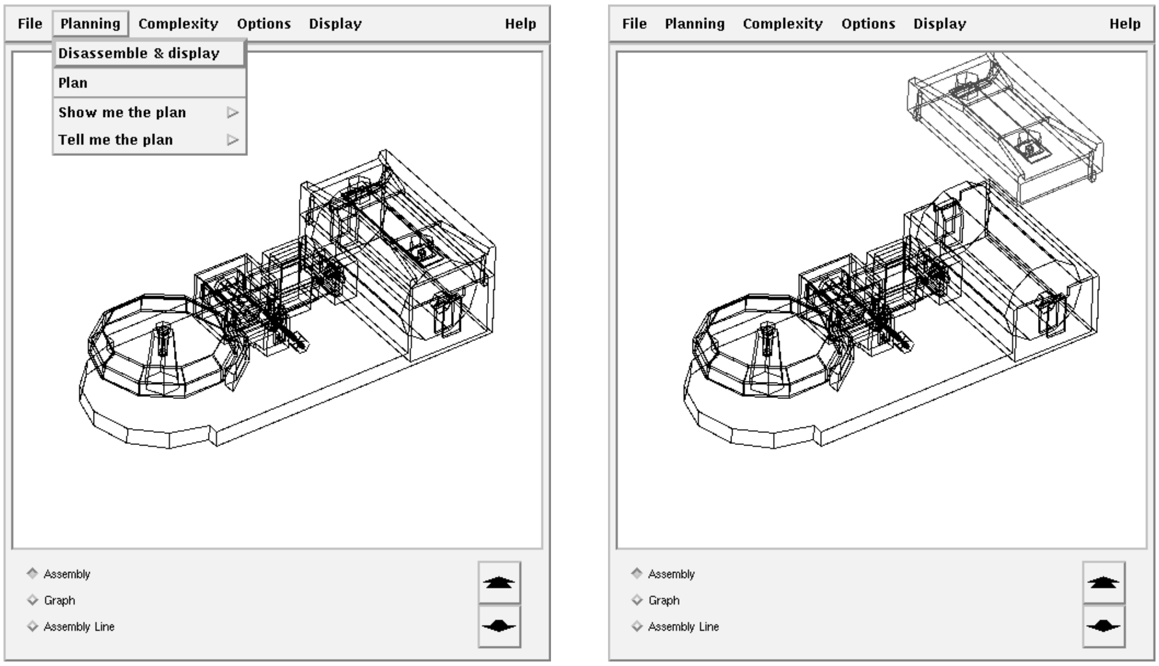
Figure 3.3: An example of an AND/OR disassembly graph.

3.2 Capabilities of STAAT

STAAT is a stand-alone system whose input is a geometric description of the assembly. A typical display screen appears in Figure 3.4a. When instructed to do so, STAAT computes a sequence of steps necessary to disassemble the given assembly; these steps can then be reversed to produce an assembly sequence. STAAT can either illustrate this sequence graphically (see Fig. 3.4b), or do its analysis “quietly” and simply present its conclusions.

In addition to the assembly sequences themselves, STAAT can produce a number of complexity measures for the assembly process. The computation of the assembly sequences can be geared towards optimizing a particular complexity measure. Some examples of these measures are:

- **Linearity.** STAAT can restrict its analysis to linear assembly sequences, and thereby quickly determine if the assembly is a linear one. Linear assemblies have the advantage that they can be built using one long assembly line, in



(a) (b)

Figure 3.4: (a) A typical display screen for STAAT. (b) STAAT illustrating a disassembly sequence.

which each "station" on the assembly line adds a single part. This is helpful because multiple assembly lines may require a great deal of coordination to ensure that completed subassemblies arrive for final assembly at the same rate and at the same time.

- **Stack.** STAAT can also quickly determine if the product is a stack assembly requiring just up-and-down part insertions. If so, then it will not need to be rotated during its construction. This reduces both product cost and production time.

- **Ease of part removal.** STAAT can attempt to find the shortest partial disassembly sequence to extract a given part. This is an indication of how difficult it would be to service that part. For small assemblies, this computation can be done through an exhaustive search of the disassembly graph, producing the absolutely optimal removal sequence; for larger assemblies, more heuristic, non-guaranteed techniques are employed.

- **Ease of fine-motion planning.** Ideally, when a part is inserted, there should be some physical features to align it, but not so many constraints as to make it difficult to insert (for example, in a 3x3 array of blocks, one would usually not insert the middle block last). Using a cost function to quantify these considerations, STAAT can produce assembly sequences which attempt to make component insertion as easy as possible.

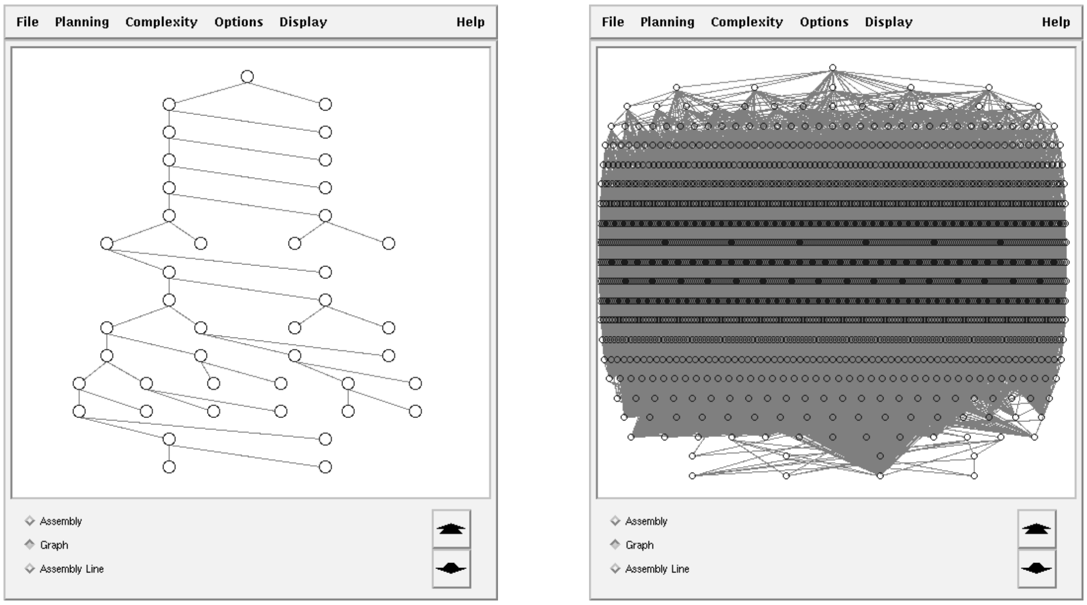
As an option, STAAT is able to restrict its analysis to sequences with connected subassemblies; this is useful because connectedness is often a necessary (but not sufficient) condition for stability.

STAAT can be instructed to compute either a single disassembly sequence or the entire disassembly graph. (See Figs. 3.5a and b.) Computing the full graph is useful for small assemblies because once it has been computed, STAAT can quickly find the absolutely optimal assembly sequence, according to the above or other criteria. With appropriate user-interface facilities, this analysis could even be made interactive, with the user fine-tuning his optimality criteria based on the results of the previous search.

Once an assembly plan has been computed, the system can also produce a schematic diagram of an assembly line one could use to manufacture the product. (See Fig. 3.6.)

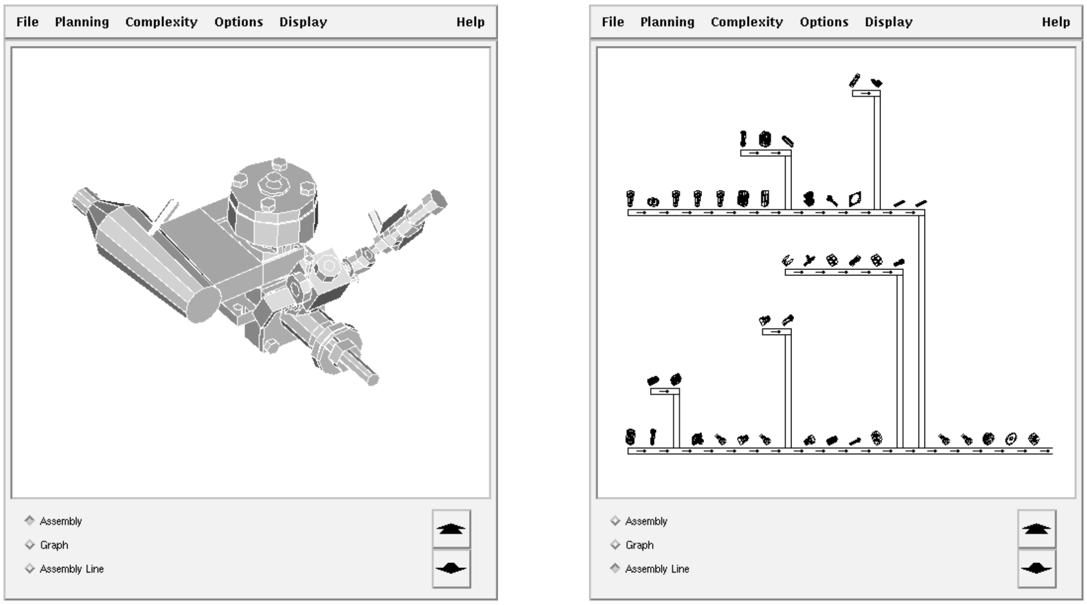
3.2.1 Restrictions

STAAT is limited to assembly sequences which are **two-handed** and **monotone**, and composed of **single-step translations**. The assemblies on which it operates are **polyhedral** and defined by **exact geometry**, *i.e.,* with no tolerances. (Recent results in assembly planning with tolerances are described by Latombe *et al.* in [47].) Additionally, since STAAT's focus is on assembly *sequencing*, the parts are considered **free-flying objects** in space, and no consideration is given to fixturing, stability, or the other assembly-planning concerns described in Section 1.3.



(a) **(b)**

Figure 3.5: AND/OR graphs for the bell assembly of Fig. 3.4a: (a) The graph generated to produce a single assembly sequence; (b) The entire AND/OR graph.



(a) **(b)**

Figure 3.6: (a) A 42-part model aircraft engine assembly; (b) A (non-optimized) set of assembly lines that might be used to manufacture it.

It should be noted, however, that the theoretical foundation for STAAT, embodied in the NDBG data structure, extends far beyond many of these restrictions. (See Section 3.4.)

3.3 System Overview

STAAT's basic operating procedure is illustrated in Fig. 3.7. I shall shortly describe each of its main steps in more detail. However, very briefly, its behavior is as follows:

Find contacts. Take the CAD data for the parts and analyze their geometry to detect contacts. The algorithm used here is a fairly simple one, based on the surface representation of the parts. Admittedly, there may be other algorithms which are more efficient; however, contact detection is not the main focus of STAAT, and I shall not present the details of STAAT's contact-finding procedures here.

Find local translational freedom (LTF) cones. Combine the contact information into an LTF cone for each pair of parts, which gives the directions in which one part is locally free to move with respect to the other.

Build NDBG. Combine the LTF cones into the Non-Directional Blocking Graph, the central data structure describing all the local blocking relationships in the assembly.

Find DBG. For a given direction of motion, compute the Directional Blocking Graph, a directed graph describing what parts are blocking what other parts, for local motions.

Find locally-free subassemblies. Given a direction of motion and a DBG, find a subassembly that is locally free in that direction.

Sweep. In order to examine *global* freedom, project all parts in the given direction. If the "shadow" of a moving part intersects the "shadow" of a stationary part, <u>and</u> the stationary part is in front of the moving part, then there will be a collision; try another motion.

Execute. Once a feasible action has been found, use this action to split the assembly into two subassemblies. Then recurse.

By performing the above steps under different constraints (*e.g.,* only remove one

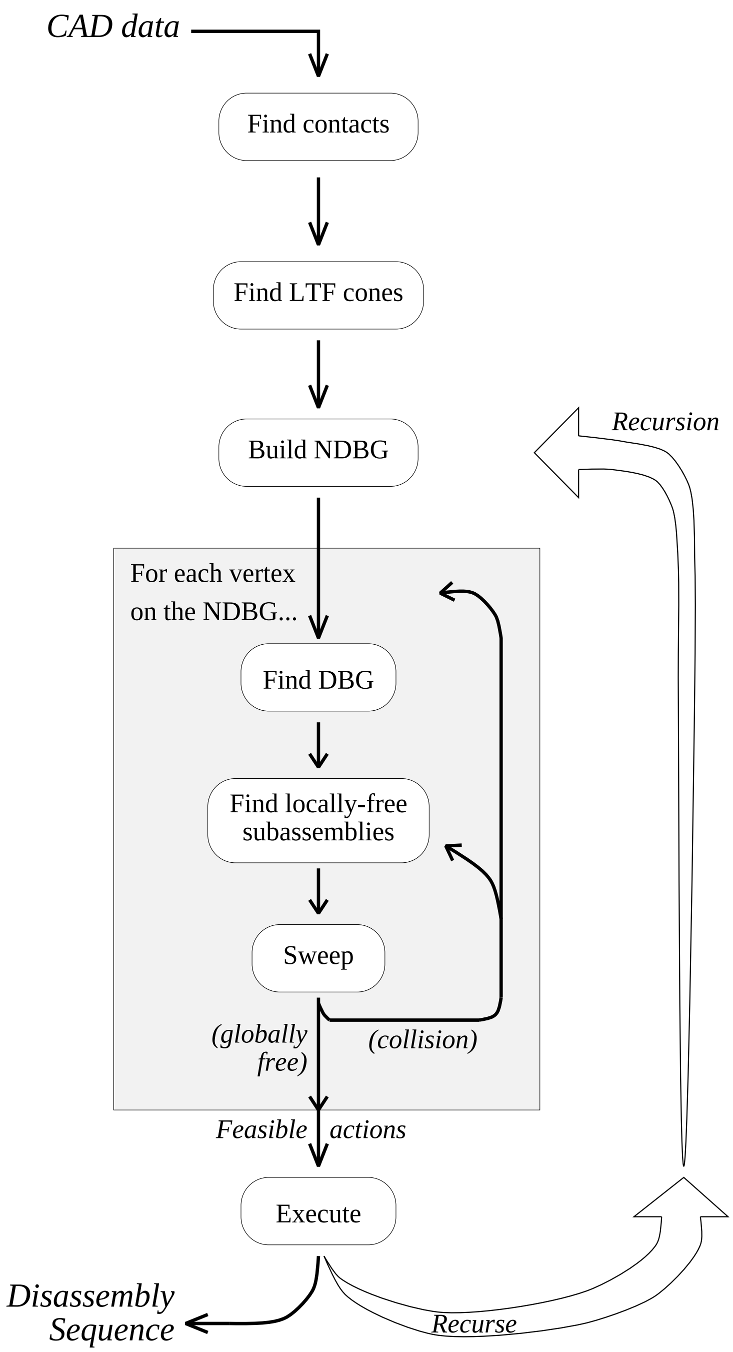


Figure 3.7: STAAT's basic operating procedure for generating assembly/disassembly sequences.

part at a time), one can compute the complexity measures described earlier.

The Non-Directional Blocking Graph is the data structure which is the key to all this analysis. Therefore, it is appropriate that a more detailed discussion of STAAT begin with an explanation of this structure and how it works.

3.4 The Non-Directional Blocking Graph

For many classes of part motion (*e.g.,* infinitesimal translations, translations with rotations, single extended motions, n-step translations, etc.), there exists a *Non-Directional Blocking Graph*, or NDBG, which summarizes the blocking relationships within an assembly for that class of motion [78, 81]. In STAAT, only the NDBG for single-step translations has been implemented, but other authors have recently reported promising results in implementing the extensions to infinitesimal generalized motions [25] and multi-step translations [26, 27]. As a practical matter, the NDBG can be computed reasonably efficiently only for relatively simple classes of motion such as these; however, the theoretical framework for this structure applies far more broadly.

Even within the realm of NDBG's for single-step translations, there are two types of NDBG's available: one for infinitesimal translations and one for extended translations. In this chapter I shall focus primarily on the infinitesimal version (complemented by sweeping operations for global analysis). However, recent progress has been made by STAAT co-author G. Ramkumar on implementing extended NDBG's as well. Because extended NDBG's are guaranteed to yield disassembly solutions when they exist, with polynomial-time algorithms, they would be the preferred means of performing disassembly once fully implemented.

This description of the NDBG, however, shall begin with the version for infinitesimal translations.

Given a part floating in three-dimensional space, the set of directions in which it can translate can be described by a unit sphere, called the *motion sphere.* Now suppose there are two parts, as shown in Fig. 3.8a. The motion sphere is now divided into three regions (Fig. 3.8b):

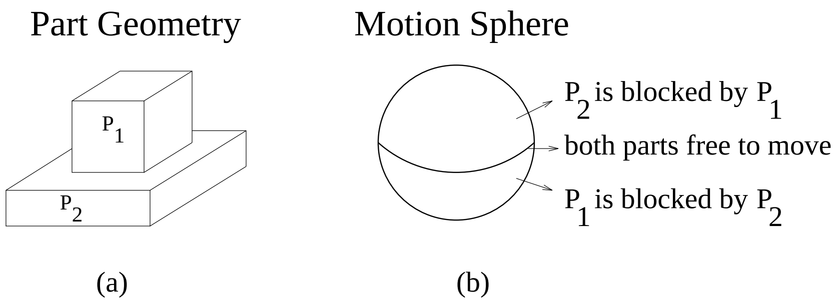


Figure 3.8: In this two-part assembly, the motion sphere is divided into three regions.

1. the top hemisphere, for directions in which P_1 is free to move, but P_2 is blocked by P_1;

2. the bottom hemisphere, for directions in which P_2 is free to move, but P_1 is blocked by P_2; and

3. the "equator" of the sphere, for directions in which both parts are free to move.

If there are multiple contacts between the parts, then the regions of local freedom for each part are narrower, as in Fig. 3.9. The set of directions in which P_1 can move locally without colliding with P_2 is called P_1's *local translational freedom (LTF) cone* with respect to P_2.[1] (See Fig. 3.10.) The LTF cones, as before, partition the motion sphere according to the blocking relationship between the two parts.

An LTF cone can be computed by intersecting all of the contact constraints between the parts. Each contact imposes a half-space restriction on a given part's motion, analogous to Fig. 3.8; therefore, the resulting LTF cone is always a convex region on the sphere. (Note, however, that it may not actually be cone-shaped, but instead take on such convex shapes as a hemisphere, an arc, or an isolated point.)

Continuing with original example, if a third part is added to the arrangement, as in Fig. 3.11, the motion sphere is divided even further. The new partitioning is found by overlaying all LTF cones from all pairs of parts onto the same sphere.

[1] For a more rigorous definition of LTF cones, the reader is referred to Wilson [78]. Wilson, however, did not use LTF cones in his construction of the NDBG, as is being done here.

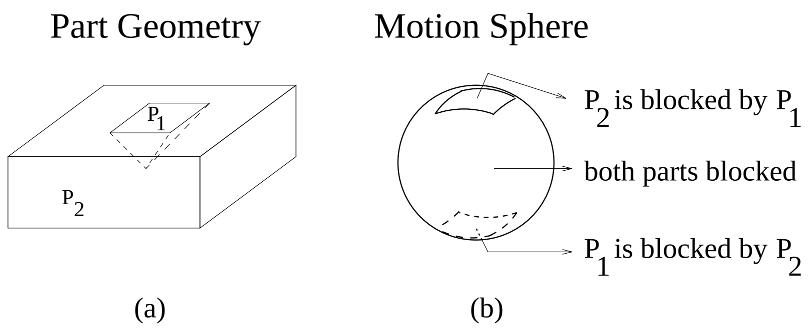


Figure 3.9: When there are multiple contacts between parts, the regions of local freedom are narrower, and described by the parts' LTF cones.

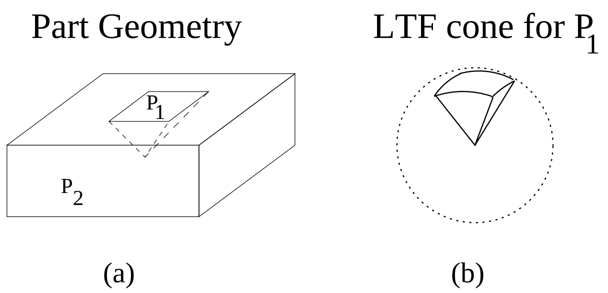


Figure 3.10: An example of a local translational freedom (LTF) cone.

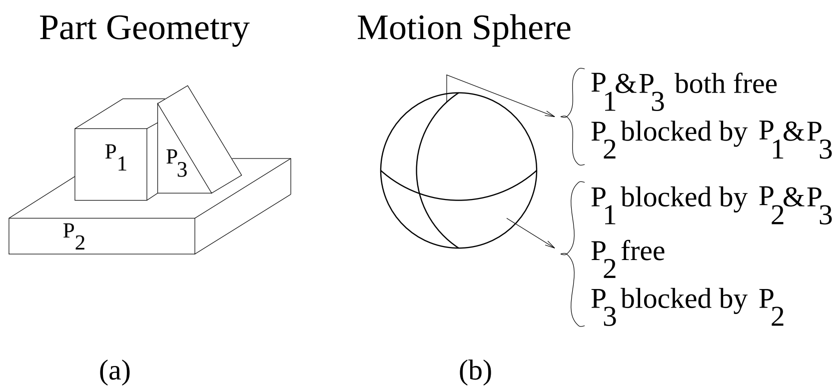


Figure 3.11: Adding a third part subdivides the sphere even further, according to the blocking relationships among the parts.

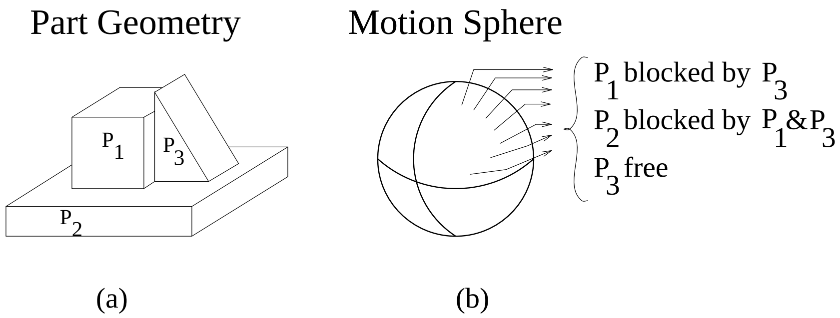


Figure 3.12: Within each region of the sphere, the blocking relationship is constant.

In each region of this sphere, there exists a different blocking relationship among the parts. However, *within* each region, the blocking relationship does not change (for local motions). This is illustrated in Fig. 3.12. And this feature is the key to the usefulness of this arrangement: *By partitioning along the LTF cones, the continuous sphere of possible directions of motion is discretized into a finite number of regions, such that the local blocking relationship within each region is constant.* Furthermore, the number of these regions is polynomial in the number of planes of contact; specifically, k contact planes partition the sphere into no more than $O(k^2)$ regions.

Typically, the blocking relationships are represented using a directed graph (known as a *Directional Blocking Graph*, or DBG), as shown in Fig. 3.13. The sphere and its associated DBG's are together known as the *Non-Directional Blocking Graph.* An example of an actual NDBG partitioning, corresponding to the 42-part engine example of Figure 3.6a, is shown in Figure 3.14.

Now, the preceding discussion was limited to infinitesimal motions; however, as noted above, it is also possible to create an NDBG for extended motions. In this NDBG, the cells on the sphere represent regions with a constant blocking relationship for *extended* motions. This "extended NDBG" can be generated by computing the Minkowski differences of the various part shapes and projecting them onto the sphere [65].

It is also possible to generate the NDBG for generalized motions (translations plus rotations). Since these motions have six degrees of freedom, this NDBG is a five-dimensional sphere in six-dimensional space. It has not been implemented in STAAT,

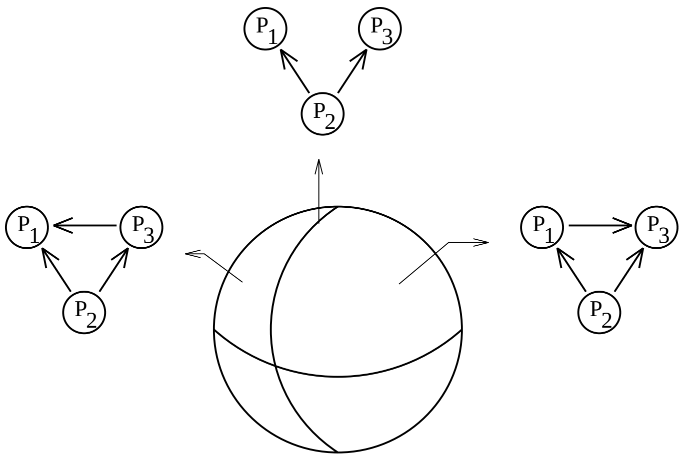


Figure 3.13: The non-directional blocking graph for the assembly shown in Fig. 3.11a.

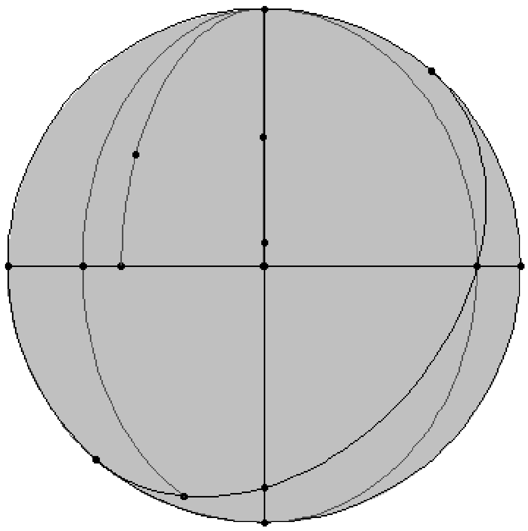


Figure 3.14: The NDBG sphere for the 42-part engine assembly of Fig. 3.6a.

but Guibas *et al.* [25] have recently devised and implemented an algorithm which uses such an NDBG efficiently. This algorithm explores only selected regions of the hypersphere, without sacrificing completeness.

For more details on NDBG's and DBG's, the reader is referred to Wilson [78, 81].

3.5 Disassembly Sequencing with the NDBG

3.5.1 NDBG Traversal

Once the NDBG has been computed, STAAT enters into a loop as it traverses the NDBG looking for directions in which to pull out subassemblies. The primary directions in which it looks are the so-called *NDBG vertices*, which are the locations on the NDBG sphere where two or more arcs intersect. As Wilson [78] showed, it suffices to examine only these directions when looking for locally-removable subassemblies. This is because given two parts P_1 and P_2 in contact along some plane A (as in Fig. 3.8), any motion along A will allow P_1 and P_2 to slide past each other, whereas outside of A, one part will always be blocking the other. Therefore, local freedom is maximized along the contact planes, and by extension, at the NDBG vertices, where multiple contact planes intersect.

Now, it may well be that the directions of maximum *local* freedom are not those of maximum *global* freedom. Consider, for example, the assembly of Fig. 3.15a. Locally, the best directions to move in are $\pm\hat{y}$, since both parts are free to move an infinitesimal distance in those directions. However, as can be seen, separating the parts to infinity in these directions would not be possible, because of protrusions in the larger part which could not possibly have been detected in the contact-based analysis. Hence, when generating extended motions, there may be cases where checking the NDBG vertices for removable subassemblies is not sufficient, and it is necessary also to examine directions in the middle of the *NDBG faces* (the regions on the sphere bounded by the arcs).

Note that even if this is done, one is still not absolutely guaranteed to find a

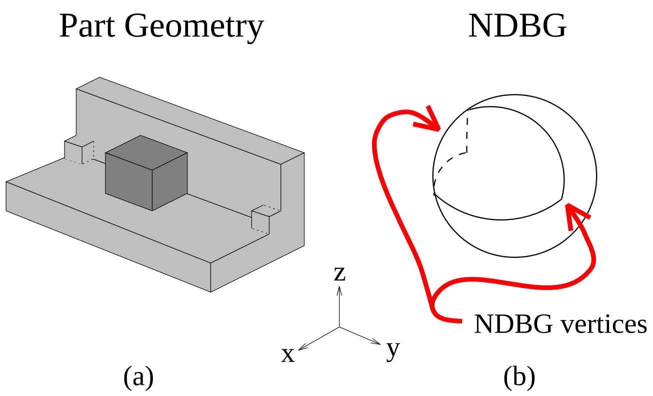


Figure 3.15: Although the NDBG vertices give the directions of maximum local freedom, there may be global constraints which prevent extended motion in those directions.

solution, because for every specific direction checked, there could be some small distant obstacle which blocks motion in that particular direction. The solution to this problem is to use the extended NDBG. However, in practice, using the local NDBG works quite well.

3.5.2 Building and Using the DBG

The first task, once a candidate direction has been identified, is to generate the *Directional Blocking Graph* (DBG) for that direction (see Fig. 3.13). The DBG is a directed graph which explicitly represents what parts are blocking what other parts for local motion in the given direction. If P_1 is blocked by P_2, then there will be an arc from P_1 to P_2 in the DBG. The first DBG constructed is obtained by comparing the proposed direction to the various LTF cones. Thereafter, as STAAT traverses the NDBG, it updates the DBG incrementally to reflect new blocking relationships (a much less expensive process than building the DBG from scratch each time). For example, every time the traversal enters an LTF cone, some "A is blocking B" constraint is relaxed; every time it exits one, such a constraint is re-imposed.

Once STAAT has the DBG, it uses that graph to identify locally-removable subassemblies. The first step in this procedure is to create a condensed version of the

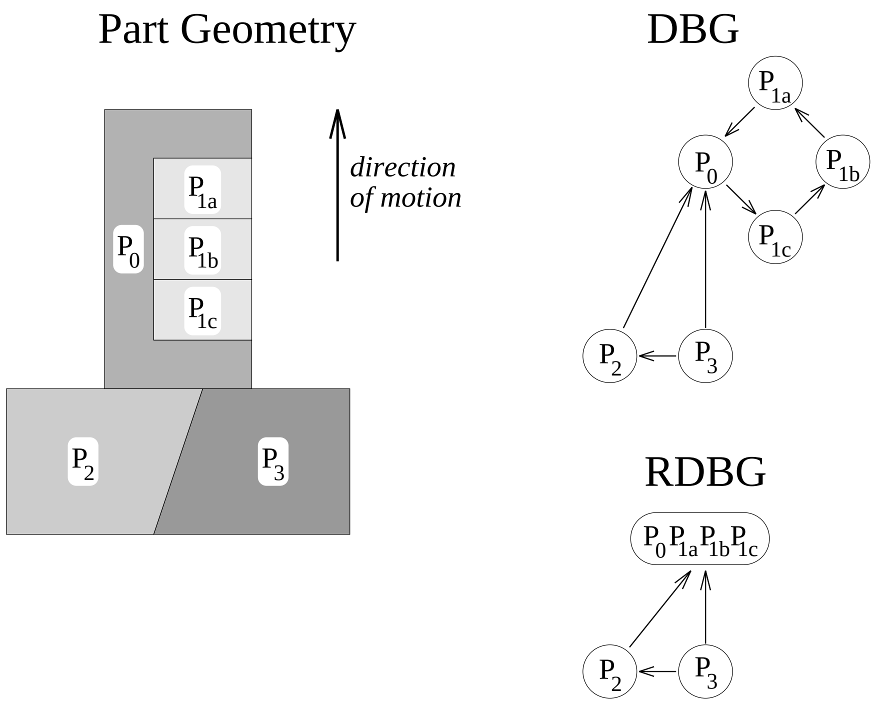


Figure 3.16: The Reduced Directional Blocking Graph, or RDBG, is created by combining each set of strongly-connected DBG nodes into a single RDBG node.

DBG called the *Reduced Directional Blocking Graph*, or RDBG. Each strongly connected component in the DBG is combined into a single RDBG node.[2] This is helpful because strongly-connected DBG nodes correspond to parts which are "interlocked" with respect to the direction of motion, and are therefore inseparable in that direction. For example, in Fig. 3.16, parts P_0 and P_{1a}–P_{1c} are inseparable for vertical motions, so in the RDBG, a single node represents all four parts. (The algorithm used to identify the strongly-connected nodes comes from Aho *et al.* [1].)

The RDBG, which is acyclic, can then be examined to find the locally-removable subassemblies. These correspond to sets of nodes from which there are no outgoing arcs leading to other nodes. For example, in Fig. 3.16, the locally-removable subassemblies are $\{P_0, P_{1a}–P_{1c}\}$ and $\{P_0, P_{1a}–P_{1c}, P_2\}$. A graph-searching algorithm to identify these subassemblies is described by Wilson in [78].

[2] A *strongly connected component* of a directed graph is a maximal subset of nodes such that for any two nodes n_1 and n_2 in this subset, a path connects n_1 to n_2.

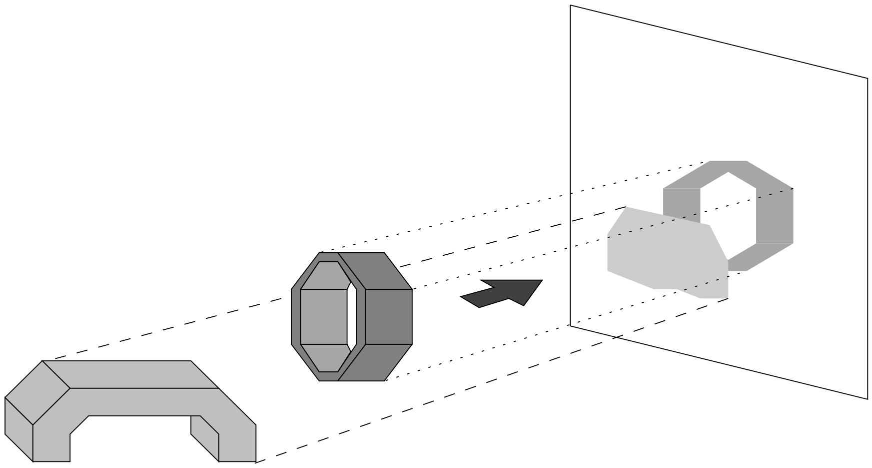


Figure 3.17: By projecting parts in a proposed direction of motion, STAAT can determine if there will be a collision between them. This "global" analysis complements the earlier local analysis, which produced the proposed direction in the first place.

Every locally-free subassembly found in this way is then passed to a *sweeping* module, which determines if the subassembly can be removed to infinity. This is done by "sweeping", or projecting, the subassembly in the given direction (see Fig. 3.17). If the projection of any moving part overlaps the projection of any stationary part, and the stationary part is in front of the moving part, then there will be a collision. In this case, the system goes back and tries again with either a new subassembly or an entirely new candidate direction and DBG.

3.5.3 Execution and Recursion

In this manner, STAAT obtains a list of one or more actions with which to partition the assembly. In the event that there are multiple feasible actions, the choice among them will depend on the overall objective. If the goal is simply to determine, yes or no, whether disassembly is possible, then the choice is arbitrary. However, if, say, the user is trying to extract a given part quickly, then the system might (heuristically) choose the action which removes that part with a minimum of other parts next to it.

Once an action is selected, STAAT uses it to break the assembly up into two subassemblies, and recursively disassemble those as indicated in Fig. 3.7. As the system proceeds with the disassembly, it builds up the AND/OR disassembly graph explicitly in its memory. If, at some point, a subassembly is found which cannot be disassembled, STAAT may be instructed to "back up" the AND/OR graph and try to find another sequence of actions to bypass that particular subassembly. (Normally, this would be ineffective since adding parts back onto an inseparable subassembly only yields another inseparable subassembly. However, if the user has specified that all subassemblies must be connected, then it may well be that the additional parts could render the assembly separable. In any event, this "try-try-again" method is merely a user option.)

Upon completion, the sequence of actions can be reversed to produce an *assembly* sequence. This sequence can be outputted with a graphical demonstration, and/or with assembly instructions in plain English.

3.6 Implementation

STAAT is implemented in C and uses the Motif toolkit in its user interface. It runs on either a DEC Alpha or a DEC 5000 workstation.

The assembly data format, which was developed in-house, is a straightforward boundary representation of a polyhedral assembly. Faces may be non-convex and may contain holes. The data format also supports cylindrical and circular features, for future expansion of STAAT.

A separate stand-alone program can convert IGES 5.1 data files into STAAT's format; this process transforms any curved surfaces into a triangular mesh.

3.7 Experimental Results

Experiments were conducted to measure STAAT's running times using infinitesimal NDBG's plus sweeping. The results are shown in Table 3.1. Two computation times were measured:

- The time to find a successful disassembly sequence; and

Assembly	No. of Parts	STAAT (DEC Alpha) Sequence	STAAT (DEC Alpha) Graph	STAAT (DEC 5000) Sequence	STAAT (DEC 5000) Graph	GRASP [78] (DEC 5000) Sequence	GRASP [78] (DEC 5000) Graph
Bell	17	0.58	5.1	2.0	13	16	522
Bell	22	1.0	31	3.4	82	21	2186
Engine	12	0.8	5.3[3]	3.3	19[3]	30	42
Engine	30	2.4	—	9.4	—	112	—
Engine	42	3.0	—	12	—	189	—

Table 3.1: Running times for the various assemblies, in seconds, excluding time to find contacts.

- The time to compute the entire AND/OR graph for the assembly. For the larger assemblies, this is not feasible in practice because of the large size of the graph.

As can be seen, the running times are significantly better than those reported by Wilson [78], often by a full order of magnitude. Much of this is probably due to the fact that STAAT is implemented in C, and Wilson's GRASP is in Common Lisp; however, STAAT also uses a more efficient version of the NDBG than GRASP, and I and the other authors of STAAT believe that this also contributes significantly to the speedup. The running times shown also compare quite favorably to those reported by Lin and Chang [52] ($\sim$ 2 minutes for a 14-part assembly).

3.8 Motivation for the Next Step

STAAT, then, is a very efficient system for performing assembly sequencing, given only a geometric description of the product. However, in order to realize the goal of a CAD tool which can provide useful feedback about the difficulty of the assembly process, significant steps must be taken to improve upon the realism of the process model. That is, some of the simplifying restrictions of Section 3.2.1 must be relaxed, and/or the scope of the system's analysis expanded.

[3] It should be noted that for the bell assemblies, only connected subassemblies were generated, while for the engine assemblies, STAAT generated *all* possible subassemblies, because the engines have no disassembly sequence using only connected subassemblies.

One of the biggest gaps between the idealized setting of STAAT and the demanding context of industry is the lack of stability considerations by STAAT. An assembly sequence is of little practical use if it is difficult or impossible to execute in a stable manner; consequently, feedback based upon such a sequence would be misleading.

Stability, in turn, depends upon the existence of a stabilizing fixture which does not interfere with the part motions. This observation, and the desire to narrow the "realism gap" in the assembly simulation, have given rise to the development of a joint sequencing/fixturing analysis. It is this analysis which is the subject of the next four chapters of this dissertation.

Chapter 4

The Co-Design Problem

In this chapter, I formally pose the problem of concurrent sequence/fixture generation. I also present and discuss some of the key assumptions made to model this problem and to solve it.

To be sure, many of the assumptions presented in this chapter represent substantial simplifications from reality, and would therefore limit the usefulness of the resulting systems in a practical industrial setting. However, since the research presented herein represents the first attempt to address the sequencing and fixturing problems in a unified fashion, I have sought first and foremost to develop the underlying concepts and techniques for sequence/fixture co-design, even in the simpler cases, leaving it to future research efforts to apply this technology to more realistic cases by releasing the various assumptions. Put another way, I have striven for "depth" of system expertise, sometimes over "breadth" of applicability. However, at various points in this chapter, and later in Chapter 8, I shall also discuss what would be required to eliminate some of these simplifications.

4.1 Problem Statement

The problem to be solved is as follows:

> Given a geometric description of a product, generate (1) a sequence of part insertions which will assemble the product from its parts, and (2)

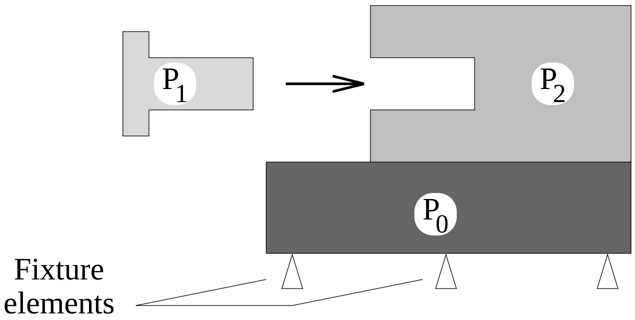


Figure 4.1: An assembly operation which is unstable due to insertion forces.

a fixture to hold the product in place during assembly, such that each intermediate subassembly is stable under gravity and all insertion forces.

4.1.1 Insertion Forces

Insertion forces are part-against-part forces which may arise from part insertions. For example, in Fig. 4.1, either friction or a slight misalignment in P_1's position could cause a force to be imparted on P_2 to the right. To allow for this possibility, one would need one or more additional fixture elements (or *fixels*) to the right of P_2.

To be conservative, it is assumed that these insertion forces could occur at any contact point(s), and in any nonnegative combination of magnitudes. (See Fig. 4.2.) The direction of these forces is presumed to be the same as the direction of insertion. Now, conceivably, oblique-angle forces could also be generated, as could insertion forces not in the final contact region (see Figs. 4.3a and b). These forces are not modeled here, but the co-design methods presented below could easily be adapted to handle them.

Insertion forces could potentially be quite large if the insertions were performed by, say, heavy robotic machinery. In fact, they could potentially overwhelm any stabilizing effects of gravity. To allow for this possibility, insertion-force stability is verified in the presumed absence of gravity. As I shall show later in Chapter 6, if the assembly is stable against each of k unit forces individually (*e.g.,* gravity and $(k-1)$ insertion forces), then it must also be stable against any nonnegative linear combination of them.

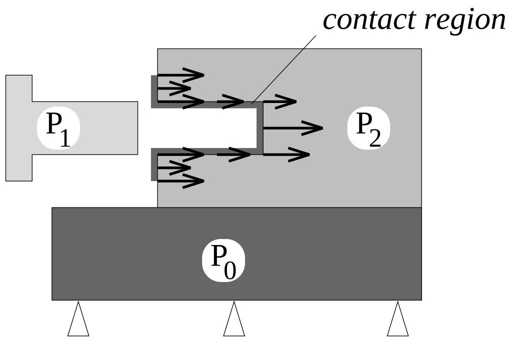


Figure 4.2: Insertion forces can occur at any contact point(s), and in any nonnegative combination of magnitudes.

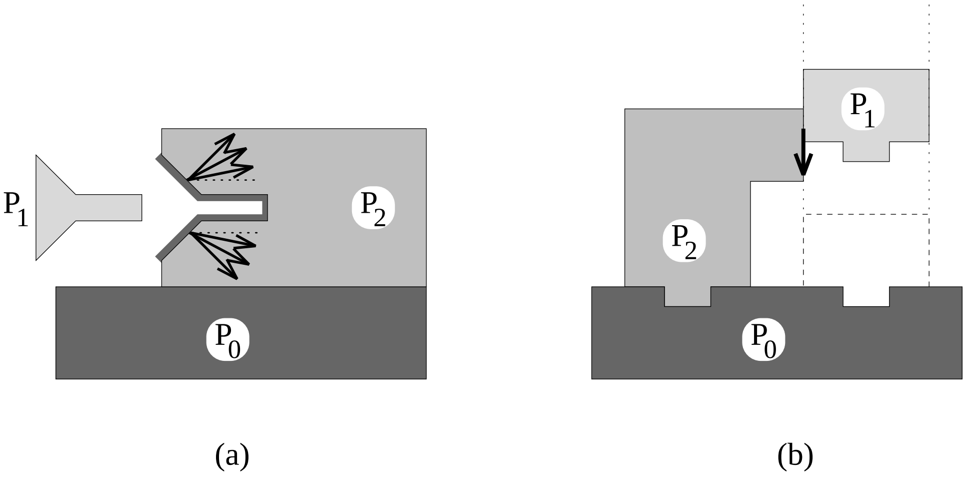


Figure 4.3: Examples of insertion forces not considered here: (a) Insertion forces not in the same direction as the insertion itself; (b) Insertion forces at locations other than the final contact region.

It should be remembered that even though the assembly sequencer generates a *disassembly* sequence (and later reverses it), it is part *insertions* which must be stabilized. So, in Fig. 4.2 for example, the insertion forces act to the right, even though P_1 would be removed to the left. This distinction illustrates an irreversibility in the generated assembly plan. For a given fixture, a sequence may be stable under assembly operations but not under disassembly operations. So in order to compute, say, the shortest disassembly subsequence to remove a key part (as in STAAT), one would need to ensure stability against removal forces rather than insertion forces. (To re-assemble that part, stability against insertion forces would once again be required.)

Insertion-force stability, or IFS, is the condition of being stable against any possible insertion forces. Methods to check IFS efficiently are described later, in Chapter 6.

4.2 Assumptions

4.2.1 The Assembly Model

The assembly is assumed to be **polyhedral**, **rigid**, **frictionless**, and defined by **nominal geometry** (*i.e.,* no tolerances). Sometimes (but not always), parts are assumed to be **glued together** (or similarly attached) upon insertion, so that the entire assembly is rigid and unified for stability-checking purposes.

Polyhedral Assemblies

One of the key lessons I learned at General Motors, where I spent a summer attempting to apply STAAT to GM products, was how critical the ability to handle curved surfaces is to the creation of a useful analysis tool. Polyhedral automobiles simply do not exist, and the same would be true of airplanes, disk drives, engines, and many other major product lines.

However, even for assembly sequencing alone, using NDBG technology, non-polyhedral parts raise substantial difficulties, most notably the generation of LTF cones and the performance of "sweeping" operations on curved-surface parts — or, in the case of extended-NDBG approaches, the generation of Minkowski differences between parts to form the extended NDBG. One solution (and the one I used at GM) is to facet the curved surfaces, *i.e.,* replace them with a triangular mesh. However, in addition to the approximate nature of this model and the vast number of facets usually required, the faceting approach carries a fatal flaw if adjacent parts are faceted "out of phase". For example, in a "peg-in-hole" situation, if the peg is faceted as a square and the hole is faceted as a diamond, then the two polyhedral models will overlap in space, immediately precluding the possibility of a collision-free sequence.

Perhaps a more useful, if not 100% complete, solution would be to expand the underlying technology to handle specific types of curved surfaces, such as cylinders, spheres, and cones. In practice, these will represent a substantial portion of all curved surfaces in an assembly, and so the ability to handle them would significantly increase the usefulness of the assembly planner. Of course, odd-shaped parts such as a car's outer-body parts would still be excluded; but then again, auto body parts are not in dire need of automatic assembly-planning systems to begin with.

From a fixturing point of view, curved parts raise certain fundamental fixturing difficulties. These are explored further in Chapter 8.

Rigid Parts

The most obvious consequence of the rigid-parts assumption is that parts and features which are deliberately non-rigid, such as wires and "snap-fit" connections, are excluded. An analysis of such objects would probably require modeling the manner and extent of allowable deformation. (Recently, Wolter and Kroll [82] have begun examining some of the representational issues for flexible, string-like parts, with a view toward assembly-sequencing applications.) A non-constant assembly geometry would also significantly increase the difficulty of confirming or disconfirming the existence of collision-free motion paths. Therefore, for the sake of simplicity, flexible parts are excluded.

However, the assumption of rigid parts also avoids another, less obvious issue, yet one of great importance to manufacturers: parts which are supposed to be rigid may actually be flexible or deformable. This flexibility, in fact, may have a significant impact on the assembly plan. For example, the large metal parts of an aircraft fuselage tend to "wobble" when not firmly secured. If there were a stability module which could analyze these peculiar dynamics, the incorporation of that module into the fixturing systems presented below would certainly improve the quality of the analysis. Another example of deformable parts occurs in the insertion of rivets into an aircraft wing. During these insertions, the metal in the wing stretches and expands slightly, so that an aluminum wing which is, say, 100 feet long may end up being 100 feet, 1/2 inch after riveting. According to engineers at Boeing, the order in which to insert rivets is an art, not a fully-understood science. This suggests that such considerations are presently unsuitable for analysis by a CAD tool, and that some partial assembly sequences (such as that of the rivets) should be specified directly by the user.

Given the difficulties involved in such analyses, the assumption of rigid parts in these cases seems to be a reasonable course of action.

Frictionless

When parts are not glued together upon insertion, friction between adjacent parts is neglected. Friction between parts and fixture elements is also neglected. This is a conservative assumption, which is discussed in greater detail below in Section 4.2.2, in the context of the fixture model.

Nominal Geometry

In this dissertation, assemblies are analyzed according to their nominal geometries (*i.e.,* without tolerancing).

Tolerancing in assemblies is a subject of enormous interest to manufacturers. In my discussions with industry representatives, the difficulties of this problem and the desire to handle it better have been raised quite frequently. Some companies, such as Sun Microsystems and Boeing, do not yet have systematic, uniform ways to handle tolerances in automated design systems, instead relying on the individual designers and "shop-floor wisdom" to ensure the validity of the proposed products. Aerospatiale in France, in discussions with my colleague Cyprien Godard, has also cited tolerancing problems as one obstacle to the automation of the assembly of their aircraft fuselages. The Aluminum Company of America ("Alcoa"), in Pennsylvania, has expressed interest in these issues, as well.

The effective handling of toleranced assemblies is an extremely complex issue. Tolerance specifications in the individual parts may or may not combine to create large uncertainties in the dimensions of the assembly. Tolerances in the assembly, in turn, must be reflected in the fixture which holds it. (The issue is also related to assembly rigidity — for example, several companies have expressed concern about forcing body parts with off-nominal dimensions into a fixture for welding, only to have them "spring back" when later released, causing internal stress and reduced product lifetime.)

It is precisely this complexity which makes a systematic analysis of toleranced assembly plans so difficult. Latombe *et al.* [47], however, have recently been able to adapt NDBG assembly-sequencing technology to toleranced assemblies. Their method is able to generate assembly sequences, when they exist, which are valid over all allowable instances of part dimensions; or, when no such sequences exist, to generate sequences which will still work for some instances of the dimensions.

Because tolerancing in assembly plans is an issue worthy of a dissertation in itself, I shall not attempt a treatment of the subject here. Indeed, treated as merely a "side issue", any such exploration of the matter would surely be shallow and unsatisfactory. Instead, in this dissertation, I shall focus on sequencing and fixturing based on nominal geometry, leaving to future research efforts the expansion of this technology to toleranced assembly specifications.

Glued Parts

Sometimes, parts shall be treated as glued, welded, screwed, or similarly fastened together upon insertion. (For simplicity, I shall refer to all such attachments simply as "glued parts".) This assumption is based on my tours of actual factories, in which I observed that not all assemblies were held together by their geometry alone.

Now, realistically, not all parts should be glued together this way (for example, moving parts such as a piston in a cylinder would certainly never be glued to one another). Engineers at General Motors have told me that sometimes in an assembly plan there may be several unglued parts inserted, followed by one glued part to hold things together (*e.g.,* as a lid). Therefore, perhaps the ideal solution, rather than a simple yes-or-no assumption about glued parts, would be to have the location of the "glue" (or spot-welds, or other attachments) specified in the input along with the assembly description. However, for now, the analyses in this dissertation shall simply treat all parts and part connections uniformly.

Some might argue that an assumption of glued parts sidesteps the difficulties which make assembly fixturing so much harder than part fixturing. This is certainly a valid commentary. However, the problem at hand, namely, the concurrent design of both an assembly sequence *and* an assembly fixture, poses challenges unseen in either the part-fixturing or assembly-fixturing domains, even with the glued-parts assumption. Specifically, one must design a fixture which stabilizes every step of an assembly sequence without interfering with the motions of that sequence. Therefore, in my opinion, the problem of co-design under the glued-parts assumption lies between part fixturing and co-design under unglued parts, in terms of difficulty and sophistication. Solving this problem will hopefully begin to extend the tractability and solvability of the well-studied part-fixturing problem into the domain of fixtured assembly sequences.

In any event, the glued-parts assumption is simply an option. It is, however, one which brings the complexity of the complete algorithms of Chapter 7 down from exponential to polynomial in the number of parts.

4.2.2 The Fixture Model

The fixture is modeled as a collection of immovable "pins" (or similar elements), which are **passive**, **non-adjustable**, **frictionless**, **firmly secured** and of **negligible size**. Their primary purpose is assumed to be **physical support**. The fixels can only apply **nonnegative force** (*i.e.,* they can "push" but cannot "pull"). Additionally, only **one fixture** is generated for the entire assembly sequence.

Single Fixture

The use of a single fixture to stabilize a sequence is a very desirable feature in an assembly plan. It eliminates the need for re-fixturing, *i.e.,* the transfer of an entire subassembly from one fixture to another. Re-fixturing is in general an expensive operation. The desire to eliminate such steps motivates the single-fixture model.

However, this model has serious drawbacks, most notably that it implies linearity in the assembly sequence. This is because the insertion of a multi-part subassembly would imply the previous use of another fixture to create that subassembly. Furthermore, realistically, not all products can be assembled with a single fixture.

Unfortunately, however, generating a desirable solution with multiple fixtures and a subsequence for each fixture is a much more difficult problem. Even if one seeks to minimize the use of additional fixtures (a reasonable objective for such an assembly plan), the optimal search strategy is not always apparent. Suppose, for example, that in the disassembly analysis, a situation arises where no more linear steps are possible. Should the planner remove a multi-part subassembly, or try re-orienting and re-fixturing the entire remaining assembly? Either option will entail the creation of a new fixture, and the choice between them is by no means clear.

For example, Fig. 4.4a illustrates such a "dead end" situation (assuming unglued parts). In this case, the entire assembly should be flipped and re-fixtured (Fig. 4.4b); this will allow the parts to be removed one at a time. If instead one were to try to resolve the dead end with nonlinear steps, one would find that two such steps were needed (remove P_1-P_2 and remove P_3-P_4), forcing the creation of not one but two additional fixtures. On the other hand, in Fig. 4.5a, a nonlinear step (removing P_1-P_2) is the best solution. If instead one were to try to rotate and re-fixture the assembly, the result would be the configurations of Fig. 4.5b and then Fig. 4.5c. The latter is another dead end due to insertion-force stability requirements; P_2's removal is infeasible here because the corresponding insertion is unstable (see Fig. 4.5d).

Thus, the more general, multi-fixture model would probably require a search over the space of fixtures and linear subsequences to identify a desirable multi-fixture solution.

The problem of generating the single-fixture subsequences themselves, however, is

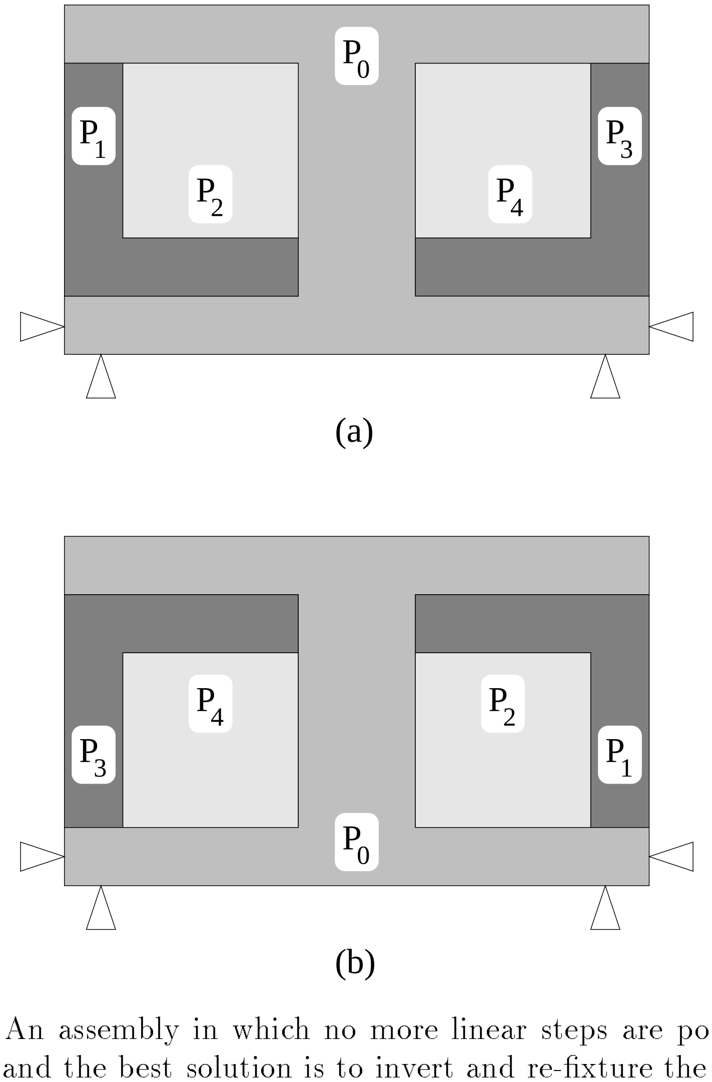


Figure 4.4: (a) An assembly in which no more linear steps are possible (assuming unglued parts), and the best solution is to invert and re-fixture the entire assembly, resulting in (b).

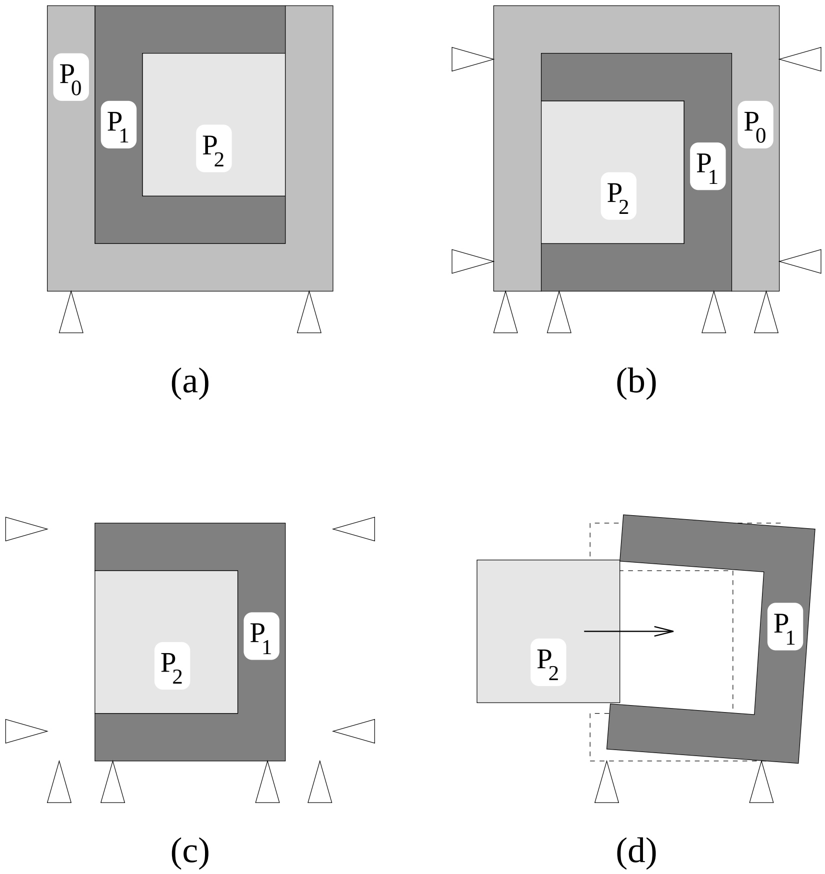


Figure 4.5: (a) An assembly in which no more linear steps are possible, and the best solution is to remove P_1-P_2. If the planner instead tries to invert the assembly and re-fixture, as in (b), the result soon afterwards is (c), which is another dead end due to insertion-force stability requirements. In (c), P_2 cannot be removed because the corresponding insertion would cause insertion-force instability, as shown in (d).

still sufficiently complicated that in this dissertation I shall focus on that task. The solutions presented here, then, could be viewed as potential "building blocks" for a larger, more general co-design system.

Firmly Secured, Negligible-Sized Fixels

The size and shape of the fixels are neglected, but the fixels are assumed to be firmly secured to the outside environment. To be sure, in a realistic assembly environment, the actual shape of each element would pose a significant assembly constraint; and if there were, say, a library of predefined fixture elements available on a CAD system (such as those described in [14, 69]), these constraints could be carefully considered. However, in the present work, I have chosen to focus on positioning only the fixels' points of contact with the assembly, leaving to future efforts the realization of these support points in a practical fixture, presumably under an integrated approach with the techniques presented herein. For now, then, the fixture elements are described only by the positions and orientations of their tips (as in Figs. 4.1–4.5).

However, the assumption of negligible-sized fixels does not bypass the problem of part/fixel collisions. Each fixel's position is defined to be that of its tip, and the presence of that fixel precludes any part from ever "sweeping across" that position. (The avoidance of part/fixel collisions is, in fact, one of the very constraints which make sequence/fixture co-design such an interesting problem.)

Some authors in the literature [55, 70, 85] treat one part in the assembly as immovable, as a means of supporting the rest of the assembly. However, I believe the conditions established here to be a more general approach, because a fixture which holds one part especially securely in place is but one possibility in the realm of producible fixtures. (However, unlike in the "base part" approach, the fixtures generated here allow *every* part to be eventually removed without modifying the fixture, including the base part.) Furthermore, simply assuming one part to be immovable, without guaranteeing the availability of fixturing points, is a simplification not employed here.

Passive, Non-Adjustable Fixture

The fixels are passive; that is, they only apply enough force to repel the external forces applied to them. They are also non-adjustable, *i.e.,* they may not be added or removed during the assembly process. These assumptions are in keeping with the "pins" model of the fixture.

These assumptions do not admit the use of adjustable clamps, which are commonly used in industrial fixtures. Clamps can always be added later to the solutions generated under the "pins" model, in order to hold the assembly more firmly in place. However, by relying on the *pins* to provide stability, one ensures stability regardless of the level of clamping force. (Of course, a pins-only fixture will cost substantially less than a fixture with clamps, due to both the price of the fixture itself and the cost of engaging and disengaging the clamps during the assembly process.)

The assumption that a fixture is non-adjustable is a logical extension of the single-fixture model. However, the techniques developed in this dissertation are adaptable to allow the dynamic addition of fixture elements during the assembly process, if desired; this extension is described in Chapter 8.

Frictionless

Friction between adjacent parts, and between parts and fixture elements, is neglected. (The only exception to this is that the existence of insertion forces is assumed, as described in Section 4.1.1, and friction is one possible cause of these forces.)

This zero-friction assumption is a classical one in fixture design. It represents a conservative approach: if an assembly is stable in the absence of friction, then it must also be stable when friction is included (since friction can only oppose motion). However, the converse is not true. Still, the exclusion of friction implies that assemblies which are stable in the real world may be found unstable in the frictionless world assumed here. One must question, then, the need for this simplification.

Figure 4.6 illustrates the difficulties of analyzing frictional assemblies. In Figs. 4.6a and b, assuming friction is present, is P_0 stable? The only possible answer is "maybe". There do exist fixel forces sufficient to prevent P_0 from falling, but there is no guarantee that such fixel forces will be applied in the real world (or even that the necessary forces would have a magnitude which is reasonable and safe). In fact, if the fixels are passive, then the horizontal fixels in Fig. 4.6 will apply zero force, since there are no horizontal external forces pushing against them, and P_0 will fall. In practice, the

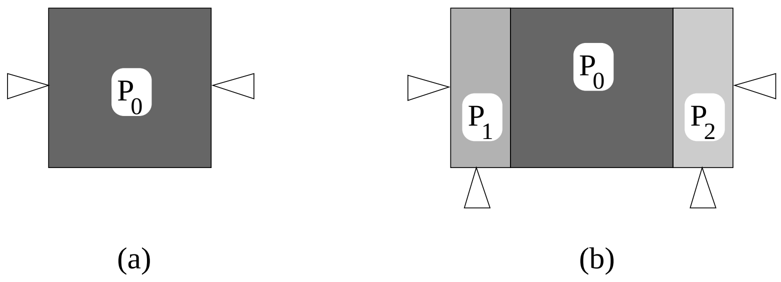


Figure 4.6: Two examples illustrating the difficulties of correctly handling frictional assemblies.

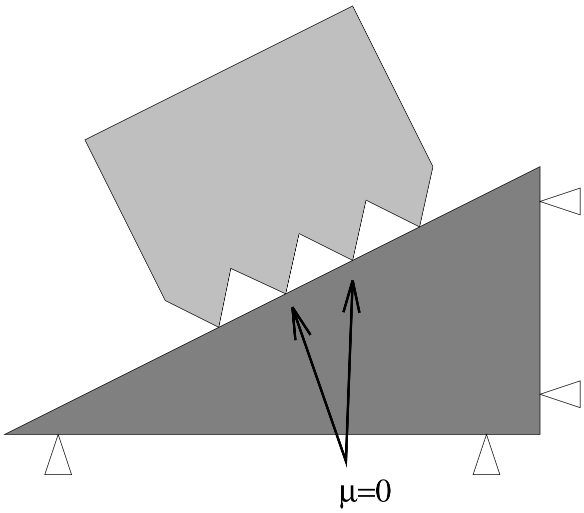


Figure 4.7: Another assembly whose stability is indeterminate [56].

stability of P_0 would probably depend on how the part dimensions varied from nominal specifications: with slightly larger-than-nominal parts, there would be a tight fit and P_0 would be stable, while with slightly smaller-than-nominal parts, there would be a loose fit and P_0 would be unstable. But in a model which does not account for toleranced dimensions, the outcome is unknown.

Another example of indeterminate stability, given by Mattikalli [56], is illustrated in Fig. 4.7. In this example, the two center contacts have zero friction and the two outer contacts have high friction. The stability or instability of the block thus depends upon the distribution of the block's weight over the contacts, which is undetermined.

Mattikalli handles such dilemmas by verifying what he calls "potential stability", which is the mere existence of a set of stabilizing fixel forces and contact forces. All of the examples of Figs. 4.6 and 4.7, then, are potentially stable. With this model, he proposes methods for checking stability and for finding the set of stable orientations. However, I believe the underlying model to be questionable, since it admits configurations such as those in Fig. 4.6 and Fig. 4.7 as stable. Given the inherent uncertainty in these situations, the principle of conservatism in engineering dictates that they be treated as unstable.

However, as Mattikalli points out, efficient methods for determining "guaranteed" stability currently remain elusive. Lee and Cutkosky [50] have considered the grasp of a part by *active* frictional elements (*e.g.,* a clamp), and derived the minimum clamping force necessary to ensure stability against a given external force. However, no methods are known which are able to handle the case of purely passive elements. Therefore, I shall follow the only conservative approach available: to verify stability without the benefit of friction.

Physical Support

It is assumed that the primary purpose of the fixture is to provide physical support to the assembly, as opposed to establishing the parts' precise locations inside tolerance ranges (for indeed, as noted above, toleranced parts are not considered here). The use of fixels to locate parts may in fact motivate a designer to insist on one or more particular fixels, or to choose certain fixturing locations within the allowable ranges prescribed by the co-design system. However, in the absence of toleranced parts, let alone an analysis of tolerance "stack-ups", the "locating" effects of fixels will not be considered by the automatic fixturing algorithms presented below.

Nonnegative Force Only

The stipulation that a fixel may "push" but not "pull" is also consistent with the "pins" model of the fixture. The methods presented here could easily be adapted to handle bidirectional fixels, such as those which operate by air suction or magnetism, by modeling each as a pair of opposing unidirectional fixels. However, these elements are rare in practice, and their use will not be considered further here.

The nonnegative-force assumption and the zero-friction assumption together imply that the set of forces which may be applied by fixture element i, oriented in direction $\vec{d}_i$, is given by

$$\alpha_i \cdot \vec{d}_i \ , \qquad \alpha_i \geq 0 \tag{4.1}$$

4.2.3 The Sequence Model

The assembly sequences generated are **linear**, **monotone**, and composed of **single-step translations**.

Linearity

As noted above in Section 4.2.2, linearity in the assembly sequence is a consequence of the use of a single fixture.

Monotonicity

The assumption of monotonicity reflects the general practice in assembly planning of inserting a part into its final position, and then treating the resulting subassembly as a whole, rather than as k parts whose individual motions must still be planned. Although the assumption of monotonicity does preclude examples such as the latch assembly of Fig. 3.1, it is by and large a reasonable assumption, and one commonly seen in the assembly-planning literature (with a few exceptions [32, 75]).

Single-Step Translations

Parts are inserted along straight-line translational paths from infinity to their final locations in the subassembly. Extensions to rotational and/or multi-step part motions remain a major direction for future research.

The problem with arbitrary insertion paths is not so much the inability to generate them as the inability to generate them quickly. Chang and Li [15] have developed a system which can plan removal paths for complex-shaped parts along arbitrary trajectories, using the technique of *Randomized Path Planning*, or RPP; however, the computations take far too long to provide immediate feedback in a CAD tool. Instead, as suggested in Chapter 1, such a system is better suited to "off-line" applications.

The need to bring the analysis time down to acceptable levels has motivated restrictions such as that of single-step translations, and spurred the development of efficient analysis techniques which operate within those restrictions, such as the NDBG data structure described in Chapter 3. As noted earlier, other researchers in the Stanford CS Robotics Laboratory have recently been able to extend NDBG techniques to the cases of infinitesimal generalized motions [25] and multi-step translations [26, 27]. These algorithms run in polynomial time in the number of parts and contacts. Hoffman [31] has also reported some success in planning multi-step translational insertions. However, none of these algorithms for more general extended trajectories runs (or appears that it will run) in sufficiently short time to be practical in a user-interactive application.

Pending the development of faster algorithms, one modest extension of the single-step translational paradigm *is* possible, however: the use of "screw" motions due to threaded contacts — *i.e.,* translations in some direction $\vec{d}$ combined with rotations about $\vec{d}$. In most examples with threaded contacts, the object being "screwed in" exhibits cylindrical symmetry, and therefore observes the same blocking relationships with a screw motion as with a simple translation (except with the threads themselves). Extending the translational sequencing methods to accommodate screw motions as well is therefore fairly straightforward.

4.3 Why Co-Design?

As noted earlier, an assembly-planning system which can perform both sequencing and fixturing would be more useful than a sequencer alone. However, the motivation for planning these two tasks concurrently, instead of first one and then the other (within the same CAD system), remains to be shown. Indeed, given the absence in the assembly-planning literature of previous concurrent sequencing/fixturing systems, a demonstration of the advantage of the concurrent approach is certainly warranted.

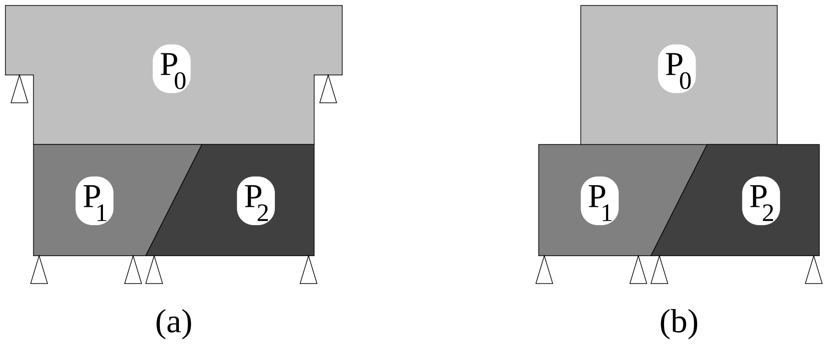


Figure 4.8: The dependence of sequencing on fixturing: (a) In this assembly, a sequencer can remove the parts in any order, and a fixturing module can later stabilize that sequence with fixels under each part, as shown. (Possible horizontal fixels for insertion-force stability are not shown). (b) However, if P_0 cannot be fixtured from below, then the sequencer must remove it first.

In short, sequencing and fixturing should be performed concurrently because of the strong interdependence of these two tasks. A sequence is only valid if there is a fixture which can stabilize it; a fixture is only valid if it does not interfere with the part motions of the sequence. To design one without considering the other is to ignore these dependencies, at the risk of later having to go back and redo the analysis if the proposed solution should be rejected by the other planning module.

For example, consider the simple assembly of Fig. 4.8a. An assembly sequencer, operating alone, could insert these three parts in any order, and if necessary a fixturing module could later stabilize the sequence with fixels placed under each part, as shown. (There might also be horizontal fixels for insertion-force stability, which are not shown.) However, with a modification to P_0, as in Fig. 4.8b, fixturing P_0 with upward-oriented fixels is no longer possible, and a sequencer with no knowledge of fixturing and stability considerations would be unaware that P_0 must now be removed first.

Similarly, a fixturing module cannot operate without a knowledge of the sequencing constraints. Consider Fig. 4.9. If the four interconnected parts are inserted left-to-right, then the circled fixels on the left must be eliminated. But if these parts are inserted right-to-left, then the circled fixels on the right must be eliminated. Furthermore, the choice between these plans might not be arbitrary; for example,

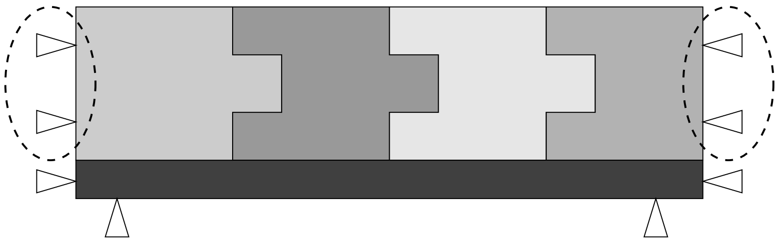


Figure 4.9: The dependence of fixturing on sequencing: The direction of the part insertions will determine which of the circled pairs of fixels will be present.

an additional part off to the right could preclude the right-to-left insertions. Such considerations would be missed by a fixturing module alone.

Now, it might be argued that "smart" sequencers or fixturers can apply rules to get around some of these difficulties. For example, in Fig. 4.8b, a good sequencer would probably disassemble P_0 first anyway, because (1) inserting the parts downward is ergonomically more desirable than inserting them sideways, and/or (2) removing the upper parts before the lower parts is a good rule of thumb when gravitational stability must be considered. However, such heuristics can also work against a planner. Consider, for example, the assembly of Fig. 4.10a. A good sequencer would probably try to remove P_0 first (upwards, say), and then take out P_1 and P_2 in either order. Such a plan would fail when it got to the fixturing module: after the removal of P_0, neither P_1 nor P_2 can be removed, because either operation creates lateral insertion forces which cannot be balanced (see Fig. 4.10b). Note that the right edge of P_2 cannot be fixtured because this would prevent the completed product from being lifted out of the fixture.

Now, certainly there are still remedies to such situations; for example, the product shown in Fig. 4.10a could be rotated and assembled in a different orientation. However, such a rotation would at least warrant a re-evaluation of the assembly sequence (especially if the sequencer used orientation-based heuristics). Alternatively, the assembly in its current orientation could simply be re-sequenced so that parts are inserted from left to right, and the final product is removed from right to left. It suffices to say that, especially in more complex assemblies, several iterations of sequencing, then fixturing, then re-sequencing, then re-fixturing, etc., might be required

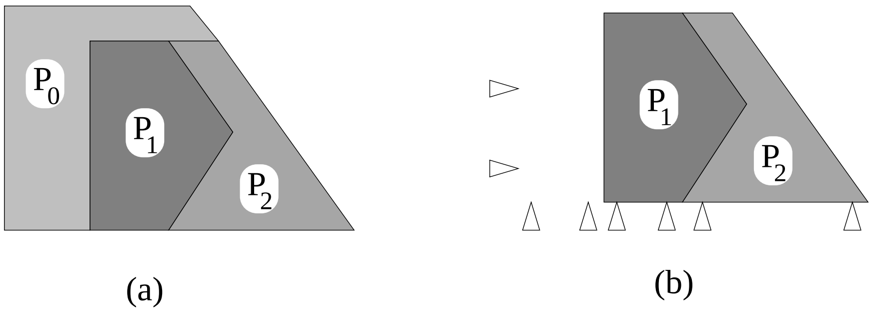


Figure 4.10: Even intuitive sequencing heuristics sometimes fail. (a) A structure for which the normally-preferred first disassembly step (removing P_0) leads to a dead end, (b), due to insertion-force stability requirements.

before a valid assembly plan were produced.

It is this inefficient back-and-forth pattern which motivates the integration of the two planning processes.

Chapter 5

Key Issues in Sequence/Fixture Co-Design

Naturally, performing assembly sequencing and assembly fixturing concurrently requires that one address many of the same questions raised in solving these problems individually. However, the concurrent approach also raises a number of new issues, and substantially increases the significance of others. Some of these issues, such as the need for insertion-force stability, have already been introduced in Chapter 4.

The formulation of an effective co-design system will depend upon a good understanding of these issues and the difficulties they pose. So, setting aside for the moment the question of how exactly sequence/fixture co-design is performed, I now explore some of the more important issues which the discussions thus far have not fully revealed or explored.

5.1 Representing the Search Space

As noted in Chapter 3, the set of all two-handed, monotone disassembly sequences for a product can be represented by a binary AND/OR graph, in which each node represents a subassembly and each AND'ed pair of children represents a decomposition of that subassembly. Now, for linear disassembly sequences, such as those being generated here, every assembly decomposition removes exactly one part. So one child

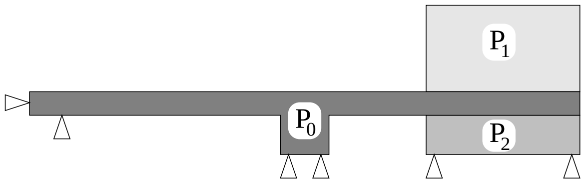


Figure 5.1: A fixture whose validity depends upon the sequence used to produce the subassembly.

out of each AND'ed pair of children will represent an extracted part, and will therefore be a leaf node. For simplicity, one can suppress these nodes, reducing the AND/OR graph to a simple OR graph whose missing leaf nodes are implicit.

The inclusion of fixture information, however, makes the representation of the search space more difficult. It would be incorrect simply to include the various fixtures along with product geometry in the nodes of the graph, because the validity of a fixture depends not just on one subassembly (represented by a node), but on the sequence used to produce it (represented by a path through the graph). For example, the fixture of Fig. 5.1 is valid if the assembly sequence inserts P_2 before P_1, but not if it inserts P_1 before P_2. Additional fixels would be needed to stabilize the latter sequence. Hence, multiple sequence-paths in a disassembly graph could potentially carry conflicting implications for the validity of a fixture stored at a node.

One solution would be to convert the graph into a tree, in which each node has exactly one path to the root, corresponding to a unique disassembly sequence. The candidate fixtures — that is, partial-solution fixtures — could then be safely stored in the nodes; the final fixture solution(s) would be stored in the leaf nodes. This scheme is illustrated in Fig. 5.2.

Another approach is to store the sequence information as part of the fixtures, rather than the other way around. This approach is illustrated in Fig. 5.3. For each fixture, a disassembly graph is set up containing only sequences which are valid for that fixture. The search space then consists of multiple disassembly graphs, some of which correspond to candidate fixtures and some of which correspond to final fixtures.

Of course, there may be other search-space representations in addition to the ones mentioned here. The appropriate representation to use would depend upon the

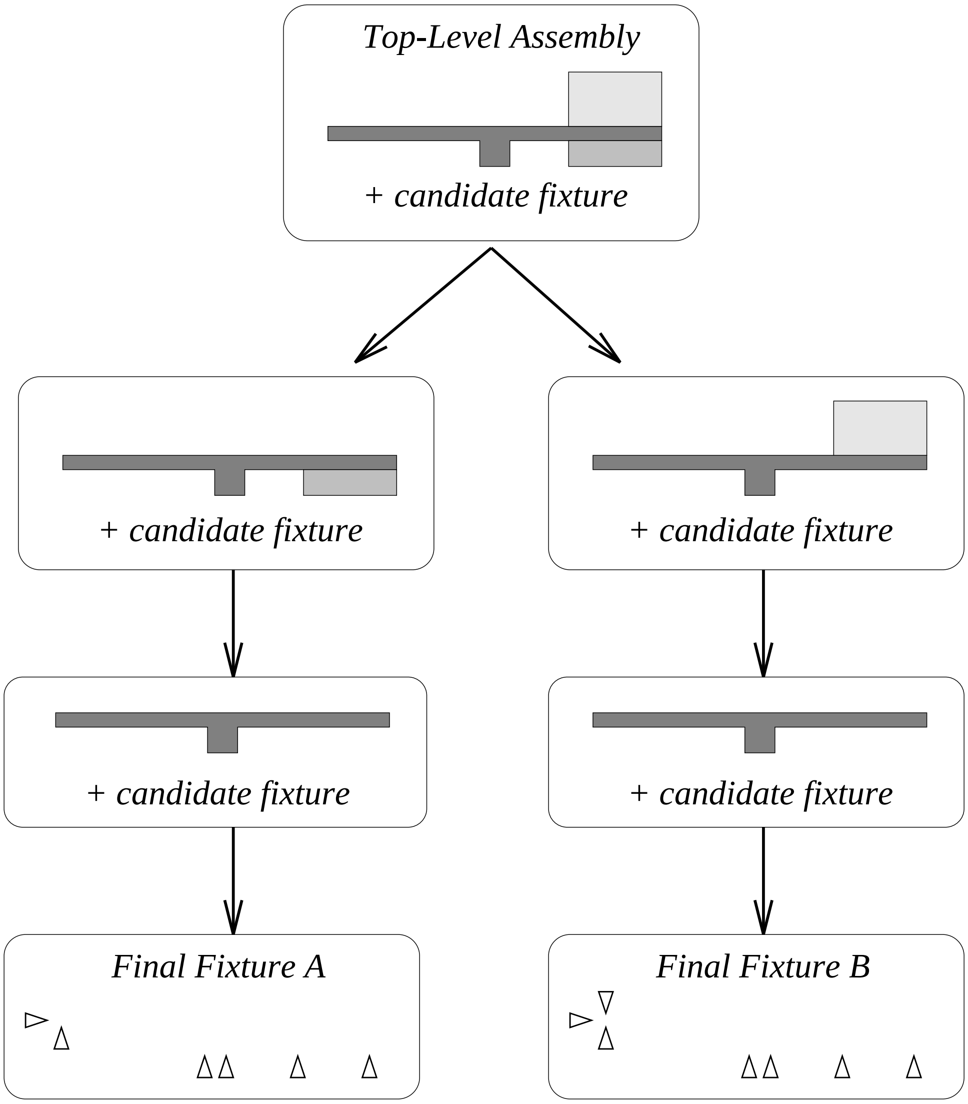


Figure 5.2: An example of a disassembly *tree*, with fixture information included in the nodes.

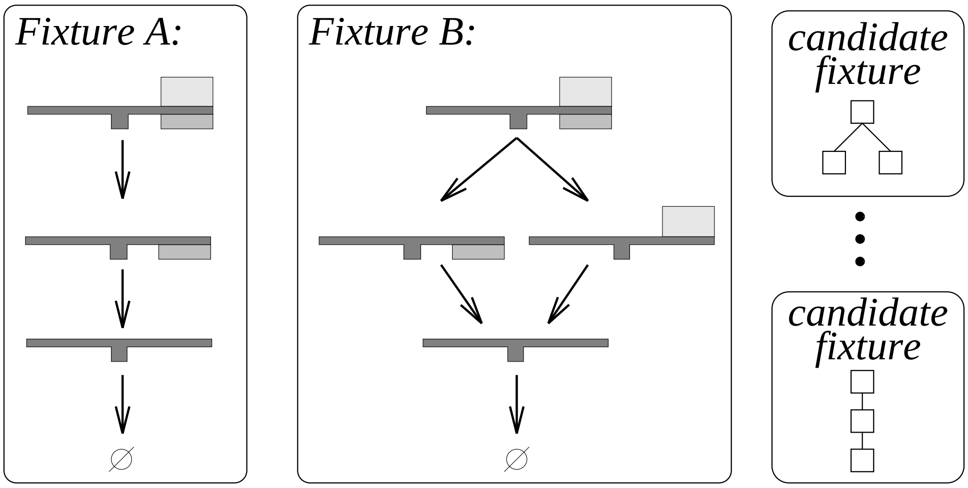


Figure 5.3: The search space may also be represented as multiple graphs, each corresponding to a single fixture.

co-design algorithm.

5.2 Figures of Merit for an Assembly Plan

What makes one assembly plan "better" than another? If the system is to analyze the "best" possible assembly plan for a given product, then it must be given some guidance as to what that means. Previous research in sequencing and fixturing has enumerated a number of complexity measures (see, for example, Section 3.2). However, sequence/fixture co-design offers the possibility for new criteria which encompass both of these aspects of assembly planning.

For example, if multiple fixtures are employed, one measure of assembly cost might be the number of times a product must be re-fixtured during assembly. (The search for a single-fixture solution in this dissertation could, in fact, be viewed as addressing the larger, more general problem, but assuming prohibitively high re-fixturing costs.) Alternatively, one might attempt to estimate the total assembly time, along the lines of Boothroyd and Dewhurst's work [8], taking into account both sequencing and fixturing considerations. Of course, it is difficult to reproduce the judgment of a good manufacturing engineer in these estimates, and so the generation of accurate, quantifiable guidelines would be a challenge in its own right.

The criteria for identifying the best assembly plan would probably depend upon the particulars of the product line and the assembly environment. For example, the best strategy for assembling airplanes would almost certainly be different than the best strategy for, say, toasters, or disk drives, or automobiles. These strategies would reflect differences in such characteristics as production volume, price per unit, part size, and the extent of standardization versus customization in the product line.

The identification of useful figures of merit for an assembly plan remains an open question. For now, because of the difficulty of the combined sequencing/fixturing problem, this dissertation shall focus on ways to find a valid sequence and fixture in the first place, leaving the task of comparing multiple solutions to future efforts.

5.3 Assembly Orientation

Given a product to fixture and assemble, one of the first tasks is to choose an orientation in which to do it. This decision will obviously have a great effect on tests for gravitational stability, so choosing one wisely is important.

In [56], Mattikalli described a way to find the set of all stable orientations for an assembly, by parameterizing the direction of gravity and finding the set of gravity vectors which satisfy the appropriate force-balancing equations. His method is both exact and efficient. However, it analyzes the stability only of a single, known assembly; choosing the best orientation for an entire assembly sequence, without knowing the sequence in advance, is quite a different matter. It might be possible to devise a co-design algorithm (perhaps an expansion of the methods presented below) which operates on a parameterized gravity vector $\vec{g}$, and chooses the best $\vec{g}$ in addition to the best sequence and fixture; the formulation of such an algorithm remains a topic for future research.

Another approach for choosing an orientation is to observe that most desirable orientations have either a bottom face or at least three vertices on the (horizontal) ground; therefore one can identify the planar faces of the assembly's convex hull, and use their inward normals as possible "up" directions. The system could choose among them by means of a heuristic (*e.g.,* fewest number of interlocked parts for removal in the upward direction), or could explore them all. The main drawback of this approach is the potentially large number of discrete orientations thus derived.

For now, the easiest solution would probably be to have the user choose an "up" direction, or a set of potential "up" directions, presumably using his intuition as a designer.

In the remainder of this dissertation, though, I shall not consider the particular means by which an orientation is chosen; instead, I shall assume that the orientation is the one provided in the input, and I shall focus on performing sequencing and fixturing in that orientation.

5.4 Grasping Fixels

In Section 4.2.2, fixels which are removable were explicitly excluded from the domain of fixtures considered by this dissertation. It should be noted, though, that among the possible types of removable fixels are a class I shall call *grasping fixels.* These are temporary fixels which must be removed *during* the part motions, and in practice might represent the grasping of a part by a human or robot hand. For example, consider the assembly of Fig. 5.4. The natural way a human would assemble this structure would be to lay down P_0, then place P_1 and hold it with the left hand while inserting P_2 with the right. The release of P_1 must happen simultaneously with the insertion of P_2. Without the use of grasping fixels in this way, this structure could not be built in the given orientation.

In many cases, the use of grasping fixels could be construed as the use of an additional hand. In Fig. 5.4, for example, the successive insertions of P_1 and P_2 could be considered a single, three-handed operation in which the hands holding P_1 and P_2 move in at different times.

In the interests of simplicity in both the analysis and the generated assembly plans, grasping fixels are not used by the assembly-planning techniques presented herein.

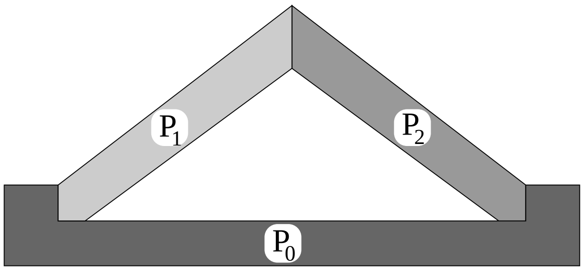


Figure 5.4: A structure which cannot be assembled in its current orientation without the use of grasping fixels.

5.5 No Upward Insertions

From an ergonomic point of view, inserting parts or subassemblies in an upward direction is more difficult than inserting those parts downward. This makes upward motions more time-consuming and expensive for manufacturers than downward motions, and so upward motions are often avoided in practice.

In sequence/fixture co-design, this issue takes on even greater significance, because downward part removals may imply instability, especially if the glued-parts assumption does not apply. For example, if a downward motion does not strike any fixels, then the removed part could simply fall in that direction, so the assembly is unstable. Similarly, if fixels need to be deleted to permit the motion, then deleting them could destabilize the assembly, for the same reason. (Even if parts are glued or fastened in place upon insertion, the operation is still difficult, because during assembly, someone or something will need to hold the part in place while it is being fastened.) These observations provide yet another motivation to restrict the disassembly analysis to part motions with a nonnegative $\hat{z}$ component.

5.6 Removing the Finished Assembly

Naturally, a fixture must not be so constraining as to prevent the product's final removal after it has been assembled. This is not an entirely new consideration; authors such as Asada and By [3] have looked at the question of *part* accessibility before. But for sequence/fixture co-design, the planner must, in effect, allow two (possibly distinct) paths for each part: one for its insertion, and one for its removal with the rest of the assembly.

The assembly-removal constraint is not terribly hard to implement if one knows in advance what the removal path is going to be. In most of this dissertation, I shall assume it will be straight up. Possible methods of choosing alternate (straight-line) removal directions are discussed in Section 6.5.

Chapter 6

A Fast System for Co-Design

In this chapter, I present a method to solve the co-design problem introduced in Chapter 4, of automatically generating both an assembly sequence and a fixture to hold all the intermediate subassemblies. Included in this chapter is a description of the methods used to check stability efficiently, and an exploration of the complex subproblem of choosing among multiple feasible disassembly actions.

The solution has been implemented for the case of planar assemblies, in a system called Atlas[1], and I present experimental results for this system. All of Atlas' techniques, however, extend readily to the three-dimensional case.

6.1 System Overview

Atlas' basic approach is to enclose the final assembly in a completely immobilizing fixture (a process called *overfixturing*), and then to remove parts one at a time, reducing the fixture as necessary. The major steps are shown in Fig. 6.1 and described in greater detail below. During the disassembly process, the requirements imposed by the evolving assembly sequence constrain the evolving fixture, and vice-versa. In general, when Atlas modifies the proposed fixture, it is modifying the *one* fixture which will hold *all* the subassemblies; therefore, not only must it ensure the stability

[1]Atlas is named for a character from Greek mythology, a Titan who as punishment for revolting against Zeus was forced forever to bear the heavens upon his shoulders. Put another way, Atlas was forced to fixture the heavens.

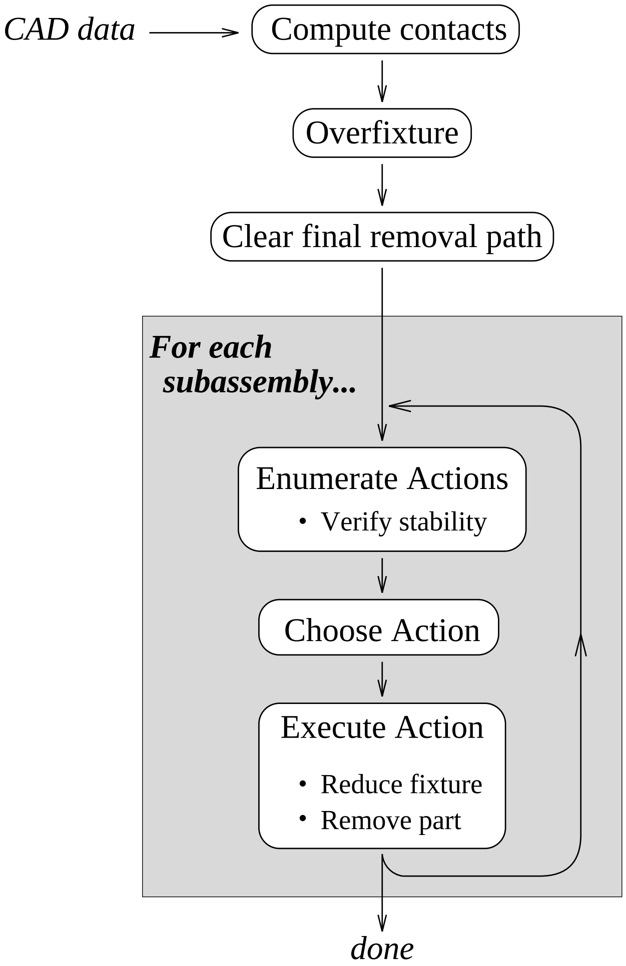


Figure 6.1: Atlas' basic operating procedure.

of the current assembly, but it must also re-verify the stability of all previously-seen superassemblies, under the new fixture.

As mentioned in Chapter 5, a fixture must not be so constraining as to prevent the product's final removal after it has been assembled. So Atlas must perform one additional step after overfixturing, to eliminate all portions of the fixture which would block the removal of the entire assembly. On average this will eliminate roughly half of the fixture elements. This operation is discussed in greater detail in Section 6.5. (Some of the figures in this chapter, however, may neglect the need for such an "escape path", for clarity in illustration.)

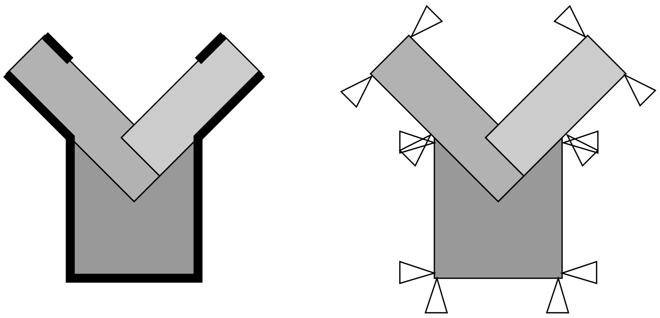


Figure 6.2: A continuous fixture representation (left) describes possible fixturing locations more completely than a discrete fixture (right).

6.2 The Fixture Continuum

Atlas represents the candidate fixture as a *continuum* of possible fixel locations, rather than as a finite set of discrete fixels. (See Fig. 6.2.) Although a discrete representation, such as was used in [56], is certainly valid, the continuous approach is a more complete representation of possible fixturing locations.

For example, typically as the disassembly analysis progresses, certain regions of the assembly boundary will gradually be eliminated as fixturing locations by the sequencing constraints. Now, when a given face is not completely covered by such regions, fixturing that face is still possible. But under a discrete representation, if the prohibited regions happen to encompass each of the spots on the face where discrete fixels belong, then the system might erroneously conclude that no fixturing at all is possible on that face. A continuous-based system, in contrast, would be aware of the remaining fixturing opportunities on the face. Its more complete representation of fixturing locations, in turn, could produce a greater choice of stable disassembly steps.

A continuous fixture representation can also be thought of as an infinite set of differential fixture elements — the only caveat being that finite-sized forces can be applied at any point, unlike in the differential model.

Ultimately, the result of this continuous-based analysis will be an assembly plan which is feasible if all of the possible fixturing locations remaining in the representation are in fact fixtured. The subsequent problem of reducing this solution to one

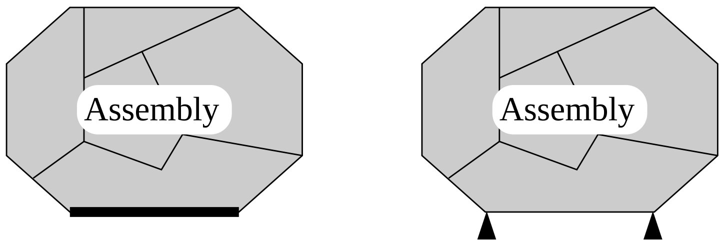


Figure 6.3: To check the stability of a planar assembly, each continuous "fixture segment" (left) is reduced to two virtual fixels (right); together, these virtual fixels can produce any net force and torque that the original segment can produce.

with a small number of fixels (which still stabilize the sequence) is a substantial post-processing step not addressed here. Baraff and Mattikalli [6] showed that finding a minimal fixture to stabilize a single assembly under gravity is NP-hard. Finding the minimal fixture to stabilize a sequence of related subassemblies, under both gravity and insertion forces, seems at least as difficult.

An alternative implementation of a co-design solution would be to use a fixture which is itself continuous; that is, instead of using point-sized fixels at strategic points within the continuum, one employs a fixture whose contact area with the assembly is both finite and substantial. I observed such fixtures in use during a tour of the Boeing 737/757 plant, on thin metal surfaces which might not withstand discrete pressure points.

6.2.1 Virtual Fixels

In order to analyze stability under the continuous representation, one may reduce the fixture continuum to an equivalent, finite set of *virtual fixels*. For planar assemblies, each "fixture segment" is reduced to two virtual fixels at the segment's endpoints, as shown in Fig. 6.3. Any net force and torque producible by the continuous segment can also be produced by these virtual fixels, and vice-versa; this makes virtual fixels useful for checking stability. In the three-dimensional case, the continuous fixture would be composed of two-dimensional "regions" rather than segments, and the virtual fixels would be at the corners of each region's convex hull [30].

The continuous representation in conjunction with virtual fixels resembles somewhat Wolter and Trinkle's use of two fixels at continuously-variable locations along an edge (in the planar case) [85]. However, the fixture continuum has the advantage of allowing segments of fixturable points to be broken up into arbitrarily many disconnected subsegments, as dictated by the evolving sequencing constraints.

6.3 The Search Space

As discussed in Section 5.1, there are several ways to represent the search space for the co-design problem. For example, one approach is to use a disassembly tree with the fixture information included in it; another is to use a collection of single-fixture disassembly graphs.

Atlas' representation is based on the disassembly tree structure of Fig. 5.2 (p. 63). The arcs in this tree represent *actions*, which are of the form

> Delete segments S from the proposed fixture, and remove part P in direction D.

The continuous fixture representation, however, adds new complications. In this scheme, there may actually be an infinite number of distinct actions possible, each resulting in a slightly different fixture. (See Fig. 6.4.) This would imply an infinite number of distinct children for a given node. Now, if one were somehow able to represent this continuum of children (and grandchildren, etc.) in the tree structure, and then exhaustively search the entire resulting tree, one would have a complete solution to the co-design problem; that is, given an assembly, the planner would be guaranteed to find a sequence and fixture for it when such a solution existed.

However, in practice, creating and managing such a tree would be difficult indeed. Equivalence classes of children are difficult to identify because the external forces to which they would respond equivalently are not all known until the sequence has been fully generated. Moreover, whether the fixture representation is discrete or continuous, the size of the full disassembly tree may grow factorially in the number of parts. To address the first problem, Atlas shall discretize the continuum of children,

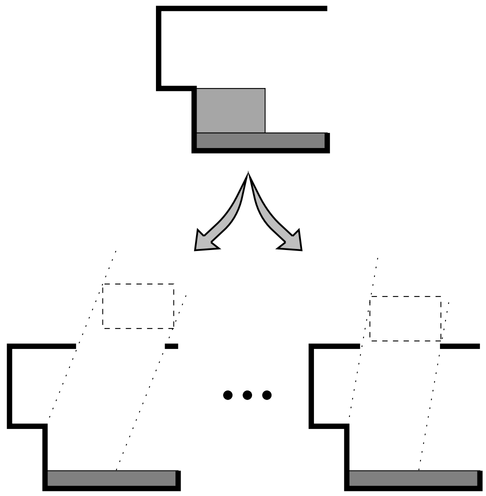


Figure 6.4: Sometimes, there may be an infinite number of distinct children, because the fixture representation is continuous.

by taking the continuum of possible actions and sampling from it a (hopefully) representative set of them. To address the second problem, Atlas shall explore only the most promising child which results; possible figures of merit for this are discussed at length below. To be sure, completeness is sacrificed by these measures, but the resulting problem becomes manageable, and by choosing the child nodes very carefully, failures can be made few and far between.

6.4 Overfixturing

The first main step in the co-design algorithm, after the contacts have been computed, is to overfixture the assembly; that is, to enclose the assembly in a completely immobilizing fixture. This is done by traversing the outer surface of the assembly and creating a continuous fixture region for each exterior face. Interior faces, delimiting closed, empty spaces inside the assembly, are not fixtured since accessing them with a real fixture would plainly be impossible. (This represents an exception, then, to the rule stated earlier about not considering the connection of the fixture points to the outside world.)

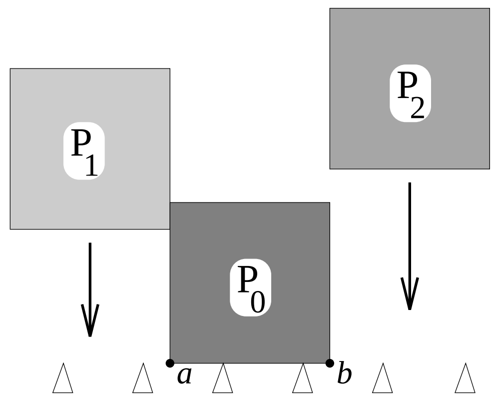


Figure 6.5: In order to stabilize P_0 against insertion forces from P_1 and P_2, fixels must be placed at its extremity points, a and b. If fixels are simply placed at regular intervals, as shown here, this is unlikely to be the case.

Mattikalli [56], in his fixturing algorithms for a single assembly, performed a similar operation with discrete fixture elements. His algorithm establishes a collection of available fixture elements at regular intervals on the outer surface of the assembly. This approach, while good as a starting point, is insufficient because it fails to fixture faces at their extremities, where the fixels can exert the most control over the assembly. When insertion forces are brought to bear, this failure can be pivotal. For example, in Fig. 6.5, the fixture cannot stabilize P_0 against the impending insertion forces unless there are fixels at the extremity points a and b.

This observation further demonstrates the usefulness of the continuous representation described above. Under the "fixture continuum" approach, the output of the overfixturing step is a complete representation of *all* possible fixturing locations on the assembly.

6.5 Clearing a Final Removal Path

The next step is to eliminate the portions of the fixture which block the removal of the completed assembly. The ability to remove the completed assembly is a mandatory and substantial requirement, regardless of the sequence; for this reason, the "escape path" is established early on, and the sequence is generated in light of the reduced fixture which results. Doing this step immediately is also consistent with the concept of working backwards in time (assembly by disassembly): final removal is the last step in an assembly sequence, so it is fitting that its preparation should be one of the first steps in the disassembly analysis.

The best direction in which to perform the removal is an open question. The simplest approach is to remove the assembly straight up. However, for some assemblies, such as automobiles, horizontally might be a more appropriate choice. The means used to remove the assembly (*e.g.,* a robot) might also constrain the choice of directions. No matter what the removal direction, however, if one knows in advance what it will be, then clearing away a path in that direction is a simple matter.

One interesting approach to choosing a direction automatically, suggested by conversations with Boeing engineers, is to empower the user to require fixturing elements at certain locations, and then search for a removal direction which does not violate those constraints. The reasons for such mandatory fixels may derive from process considerations outside the realm of Atlas. For example, if parts are toleranced, the user may place a fixel as a locating pin to firmly establish the position of a certain feature. At first, in the absence of any such constraints, the range of possible removal directions is the entire upper half of the motion sphere; but every "un-deletable" fixel placed by the user substantially reduces this range.

For now, Atlas shall simply remove the assembly straight up. Finding other ways to enumerate logical directions remains a matter for future research. Given a series of prospective directions, either specified by the user or found by some future algorithm, one could conceivably iterate the algorithm presented herein over each of them, and choose the optimal solution from among the results.

6.6 Enumerating Feasible Actions

An action is said to be *feasible* if it involves no part collisions and the fixture modification violates no current or previous stability requirements. Atlas' procedure for enumerating feasible actions is illustrated in Fig. 6.6. Atlas considers each part in turn, and generates its LTF cone with respect to the union of all other parts. (This

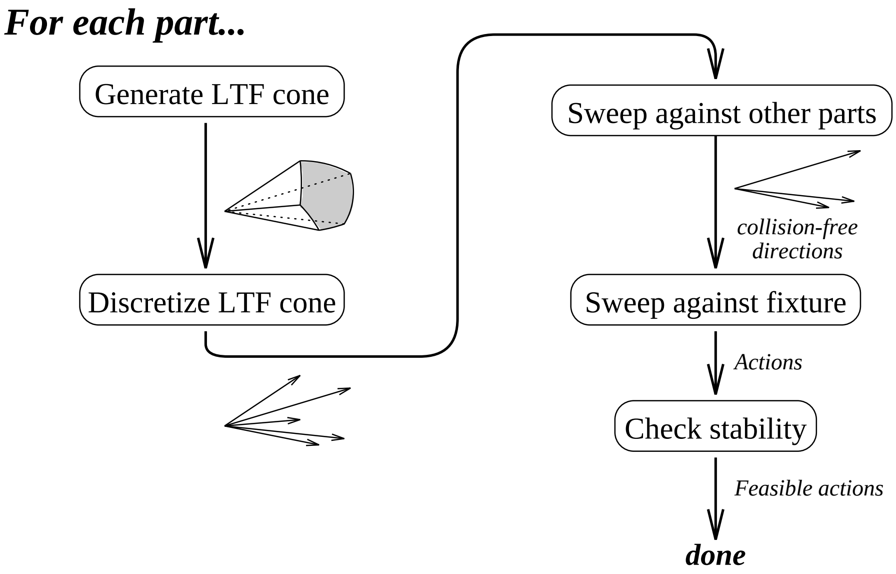


Figure 6.6: Atlas' procedure for enumerating feasible actions.

is equal to the mutual intersection of the LTF cones with respect to the individual parts.) From that cone, Atlas chooses a set of discrete directions, such as the "corners" of the cone, in which to attempt motion. A sweeping operation similar to that of STAAT then eliminates the directions which create part-part collisions.

However, this time (unlike in STAAT), a second sweeping operation is brought to bear, to detect those portions of the fixture that must be deleted in order to permit the part's removal. (See Fig. 6.7.) Finally, the stability of the assembly and all previously-seen superassemblies is verified by means of linear programs (more on this in Section 6.7). The result is a list of feasible actions.

Now, sometimes it is possible to remove a part without reducing the fixture at all. In fact, these *fixture-preserving actions* may be preferred over *fixture-reducing actions*, since the latter may reduce the ability of the fixture to stabilize future subassemblies. In order to try to find fixture-preserving actions when they exist, Atlas performs the preceding analysis a second time, but this time with the fixture treated as unchangeable, like a big, immovable part. That is, in the second pass, the LTF cone includes motion constraints from the fixture as well as from other parts, and the sweeping module rejects any fixture-reducing actions entirely. Because the LTF

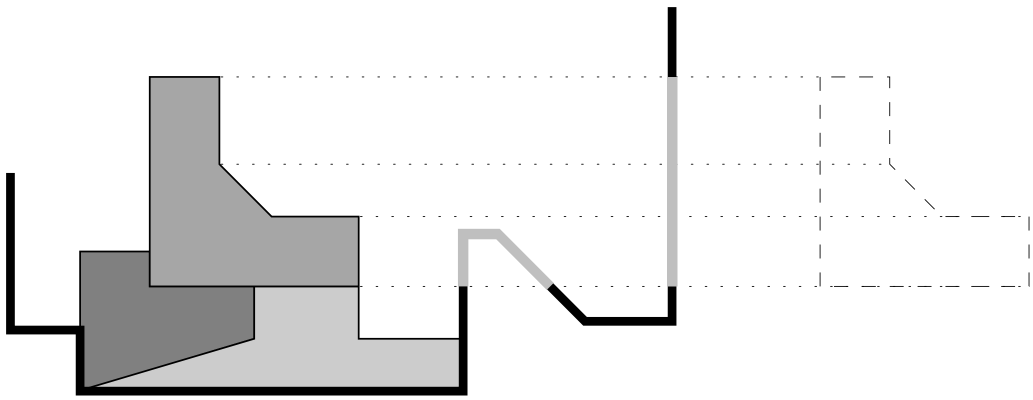


Figure 6.7: Atlas uses a sweeping operation to detect those portions of the fixture which must be deleted in order to permit a part removal.

cone may well be different (more precisely, it may be smaller), it will be discretized into possibly different directions than those considered in the first pass, leading to the creation of new actions. Any actions which then pass the demanding testing process of the sweeping module must be fixture-preserving, and are included in the list of feasible actions for consideration by the next phase of the co-design process.

6.7 Verifying Stability for Actions

Before continuing with a discussion of how Atlas chooses among feasible actions, it is instructive to examine the method by which it verifies assembly stability.

6.7.1 Basic Approach

Atlas checks assembly stability using the method presented by Wolter and Trinkle [85], which revolves around the notion of wrench-balancing. Consider first the case where parts are glued together upon insertion, so that the entire assembly can be treated as a single, rigid unit.

A *wrench*, also called a generalized force, is a force and a torque together:

$$\vec{w} = \begin{pmatrix} \vec{F} \\ \vec{\tau} \end{pmatrix}$$

For a planar assembly, $\vec{w} \in \Re^3$; for a three-dimensional assembly, $\vec{w} \in \Re^6$. Each fixel (or virtual fixel) imparts a wrench on the assembly, as do external forces such as gravity and insertion forces. The net sum of these wrenches must be zero to achieve stability.

Now, since the fixels can only apply nonnegative force, the set of possible wrenches producible by a set of m fixels is:

$$\alpha_0\vec{w}_0 + \alpha_1\vec{w}_1 + \cdots + \alpha_{m-1}\vec{w}_{m-1} \; , \qquad \alpha_i \geq 0$$

where $\vec{w}_i$ is the wrench produced by fixel i when it applies a unit force. So to achieve stability against some external wrench $\vec{W}_{ext}$, there must be some $\{\alpha_i\}$ such that:

$$\alpha_0\vec{w}_0 + \alpha_1\vec{w}_1 + \cdots + \alpha_{m-1}\vec{w}_{m-1} + \vec{W}_{ext} = \vec{0}, \qquad \alpha_i \geq 0 \tag{6.1}$$

or,

$$[\vec{w}_0 \; \vec{w}_1 \; \cdots \; \vec{w}_{m-1}] \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} = -\vec{W}_{ext} \; , \qquad \alpha_i \geq 0 \tag{6.2}$$

Atlas checks stability by seeking a solution $\{\alpha_i\}$ to eqn. (6.2) with a linear program, using the Simplex method [18]. If and only if such a solution exists, the assembly will be stable.

A subtle but important premise here, established in [5, 17], is that if there exists a set of fixel forces which *can* balance out the external wrench, then the external wrench *will* in fact be balanced out (although possibly by a different set of fixel forces). This implication is not true when friction is involved, as was illustrated previously in Fig. 4.6 (p. 54).

In his thesis, Mattikalli [56] developed a useful, similar test for *robust* stability, which ensures that stability is maintained even if $\vec{W}_{ext}$ varies slightly. This avoids, for example, situations where an assembly is gravitationally stable only in a very precise orientation.

For the case where the n parts are not glued together, the notion of wrenches can be generalized, and a similar analysis can be applied. A wrench in this context is a vector of multiple forces and torques, acting on the different parts in the assembly:

$$\vec{w} = \left[\vec{F}_0^T \; \vec{\tau}_0^T \quad \vec{F}_1^T \; \vec{\tau}_1^T \quad \cdots \quad \vec{F}_{n-1}^T \; \vec{\tau}_{n-1}^T\right]^T$$

Here, $\vec{w} \in \Re^{6n}$, with six force/torque components acting on each part. Most of these components will usually be zero (except in the case of the gravity wrench, which affects all parts). With unglued assemblies, wrenches from the interpart contacts must also be included; each contact introduces a nonnegative force between two adjacent parts, equal in magnitude for the two parts but opposite in direction. The resulting wrench equation is

$$[\vec{w}_0 \cdots \vec{w}_{m-1}] \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} + [\vec{c}_0 \cdots \vec{c}_{k-1}] \begin{pmatrix} \gamma_0 \\ \vdots \\ \gamma_{k-1} \end{pmatrix} = -\vec{W}_{ext} \;, \qquad \alpha_i, \gamma_j \geq 0$$

where $\vec{c}_j$ is the wrench imparted on the assembly by the j^{th} contact, for a contact force of unit magnitude; or equivalently,

$$[\vec{w}_0 \cdots \vec{w}_{m-1} \; \vec{c}_0 \cdots \vec{c}_{k-1}] \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{m-1} \\ \gamma_0 \\ \vdots \\ \gamma_{k-1} \end{pmatrix} = -\vec{W}_{ext} \;, \qquad \alpha_i, \gamma_j \geq 0 \tag{6.3}$$

As before, the existence of a solution $\{\alpha_i\}$, $\{\gamma_j\}$ can be determined with a linear program.

For more information on these formulations, the interested reader is referred to [85, 56].

The ability to check stability in the presence of multiple, independent external forces (such as gravity and insertion forces) derives from the following simple theorem.

Theorem 6.1 *If an assembly is stable against an external wrench $\vec{W}_0$, and it is also stable against an external wrench $\vec{W}_1$, then it is stable against any external wrench $u\vec{W}_0 + v\vec{W}_1$, for $u, v \geq 0$.*

Proof. Since equations (6.2), for glued parts, and (6.3), for unglued parts, have the same form, it suffices to consider here only the glued case, governed by eqn. (6.2).

Because the assembly is stable against $\vec{W}_0$ and against $\vec{W}_1$, there must exist some $\{\alpha_i\}$, $\{\beta_i\}$ such that

$$[\vec{w}_0 \cdots \vec{w}_{m-1}] \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} = -\vec{W}_0 , \qquad \alpha_i \geq 0 ,$$

$$[\vec{w}_0 \cdots \vec{w}_{m-1}] \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{m-1} \end{pmatrix} = -\vec{W}_1 , \qquad \beta_i \geq 0$$

Simply taking a weighted sum of these two equations, with coefficients u and v, one obtains

$$[\vec{w}_0 \cdots \vec{w}_{m-1}] \begin{pmatrix} u\alpha_0 + v\beta_0 \\ \vdots \\ u\alpha_{m-1} + v\beta_{m-1} \end{pmatrix} = -(u\vec{W}_0 + v\vec{W}_1) \tag{6.4}$$

Since u, v, α_i and β_i are all nonnegative, each of the coefficients in the column vector must also be nonnegative. Therefore, the stability condition of eqn. (6.2) is satisfied for external wrench $u\vec{W}_0 + v\vec{W}_1$. □

So, by verifying stability against gravity alone and against individual insertion forces alone, the planner automatically establishes stability against an unknown combination of them.

Later in this chapter, the following corollary to Theorem 6.1 shall also prove useful.

Corollary 6.2 *If an assembly, with glued parts, is stable against a unit force $\vec{f}$ applied separately at points $\vec{p}_0, \vec{p}_1 \ldots \vec{p}_{k-1}$, then it is stable against $\vec{f}$ applied at any point in the convex hull of $\{\vec{p}_i\}$.*

For an assembly with unglued parts, the same result holds if $\vec{f}$ is applied to the same part in every instance.

Proof. Let $\vec{CM}$ be the center of mass, either of the assembly (in the glued-parts case) or of the affected part (in the unglued case).

Let $\vec{\tau}_i$ be the torque about $\vec{CM}$ resulting from $\vec{f}$ applied at $\vec{p}_i$.

Let $\vec{p}\,'$ be any point in the convex hull of $\{\vec{p}_i\}$; that is, $\vec{p}\,' = \sum_{i=0}^{k-1} \lambda_i \vec{p}_i$, for $\lambda_i \geq 0$ and $\sum_{i=0}^{k-1} \lambda_i = 1$.

Then the torque at point $\vec{p}\,'$, due to force $\vec{f}$, is

$$
\begin{aligned}
\vec{\tau}\,' &= (\vec{p}\,' - \vec{CM}) \times \vec{f} \\
&= (\vec{p}\,' \times \vec{f}) - (\vec{CM} \times \vec{f}) \\
&= \left(\sum_{i=0}^{k-1} \lambda_i \vec{p}_i \times \vec{f}\right) - (\vec{CM} \times \vec{f}) \\
&= \sum_{i=0}^{k-1} \lambda_i (\vec{p}_i \times \vec{f}) - \sum_{i=0}^{k-1} \lambda_i (\vec{CM} \times \vec{f}) \\
&= \sum_{i=0}^{k-1} \lambda_i (\vec{p}_i - \vec{CM}) \times \vec{f} \\
&= \sum_{i=0}^{k-1} \lambda_i \vec{\tau}_i
\end{aligned}
$$

So in the glued case, the wrench at point $\vec{p}\,'$ is

$$
\begin{aligned}
\vec{w}\,' &= \begin{pmatrix} \vec{f} \\ \vec{\tau}\,' \end{pmatrix} \\
&= \sum_{i=0}^{k-1} \lambda_i \begin{pmatrix} \vec{f} \\ \vec{\tau}_i \end{pmatrix}
\end{aligned}
$$

In the unglued case, the wrench is the same expression in a larger vector padded with zeros.

$\vec{w}\,'$ is thus a nonnegative linear combination of the wrenches at $\{\vec{p}_i\}$. By Theorem 6.1, then, the assembly must be stable against $\vec{w}\,'$. $\square$

6.7.2 Efficient Insertion-Force Stability Checking

As noted earlier, it is assumed that insertion forces could occur at any contact point(s) between the inserted part and the subassembly (see, for example, Fig. 4.2, p. 43).

Because there may be many such contact points, verifying insertion-force stability (IFS) could potentially involve many more stability checks than verifying gravitational stability.

A logical question, then, is whether it is possible to speed up this process without sacrificing either accuracy or generality. The answer is that it is, in several ways.

IFS Caching. Suppose an assembly is known to be stable with respect to some insertion force D acting on part P_0, where the fixels are F and the other parts are $\{P_i\}$. The system can store this stability result in a table, so that if the question should come up again, it can be answered quickly. But the exact same conditions need not be present in order for the previous result to be re-used. For example, since, as noted in Section 4.1.1, the test is being performed in zero gravity, adding parts unaffected by D will not render the assembly unstable. Conversely, a part in $\{P_i\}$ which exhibits no contact or fixel forces in response to D can be removed without destabilizing the assembly. Finally, if only fixels $F' \subset F$ exert a non-zero force in response to D, then only F' are required to ensure stability. Applying these three rules greatly expands the applicability of previous insertion-force stability results.

I call this technique *IFS caching*, and in the examples presented later in this chapter, its use reduced overall execution times by as much as 50-60%.

Convex Hull of Projected Contact Points. In addition to reducing the number of IFS checks which require full-fledged linear programs, one can also reduce the number of checks made in the first place, as follows.

Consider Fig. 6.8, which expands upon Fig. 4.1. The contact region between P_1 and P_2 is represented by point-plane contacts at six distinct points, $\{c_0 \dots c_5\}$.[2] Initially, then, it would seem that at least six different IFS checks are needed to determine the stability of the operation. However, observe that an insertion force at c_1 will exert the same torque as the same insertion force at c_2, because they are aligned with respect to the direction of insertion. This observation is illustrated in Fig. 6.9.

[2]Wilson [78] showed that point-plane contacts can effectively represent any realistic part-part interface in a polyhedral assembly, in terms of reproducing the local freedom constraints. For this reason, the point-plane contact model is used in both STAAT and Atlas.

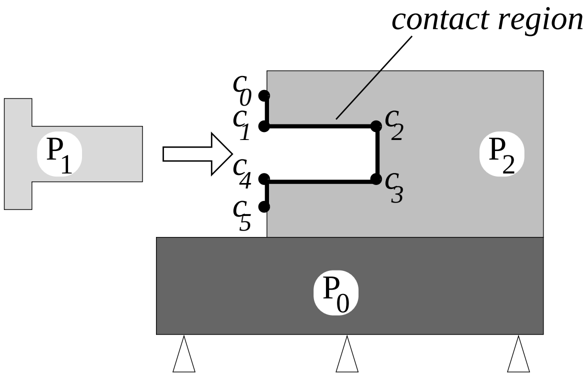


Figure 6.8: There are six (discretized) contact points between P_1 and P_2, but not all of them need to be explicitly checked for IFS.

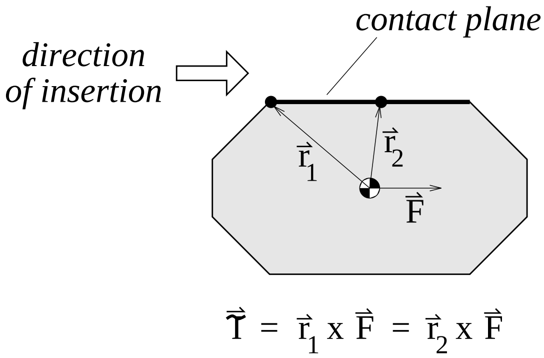


Figure 6.9: Insertion forces at these two contact points will exert the same torque.

Therefore, insertion-force stability at c_1 is equivalent to insertion-force stability at c_2. Similarly, insertion-force stability at c_3 and c_4 are equivalent. It suffices, in fact, to project the contact points in the direction of insertion, and only perform one IFS check for each *projected* contact point (see Fig. 6.10).

The number of IFS checks needed to validate an insertion can be reduced further still, by means of Corollary 6.2. In Fig. 6.10, if the assembly is stable against a unit insertion force anywhere along the pc_0 line, and it is stable against a unit insertion force anywhere along the pc_3 line, then by Corollary 6.2, it must be stable against any insertion force in the convex hull of those lines. In particular, it must be stable against insertion forces along the pc_1 and pc_2 lines. Therefore, in this example only *two* IFS checks are needed: one for pc_0 and one for pc_3.

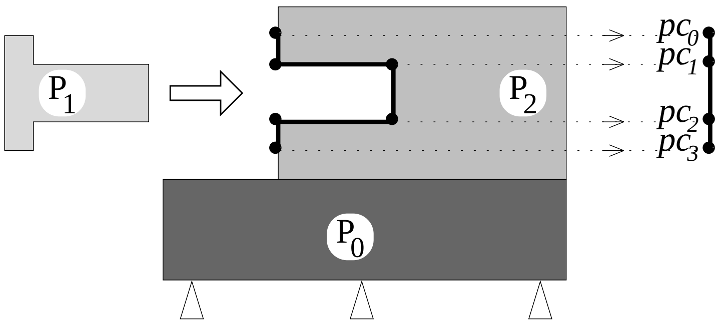


Figure 6.10: Of the six discrete contact points, only four will generate distinct torques.

In the three-dimensional case, as in Fig. 6.11, the projected contact region is two-dimensional. As before, verifying insertion-force stability along ℓ lines parallel to the direction of insertion automatically establishes insertion-force stability at all points inside the convex hull of those lines. Therefore, to ensure insertion-force stability over the entire contact region, it suffices to check IFS at the vertices of the projected convex hull of that region.

Observation 6.3 *To verify insertion-force stability, it suffices to verify IFS at the vertices of the convex hull of the projected contact region.*

Note that in the case of unglued parts, this observation refers to the contact region of each part individually with the moving part — that is, for n stationary parts in contact with the moving one, there will be n projected convex hulls to check.

The optimization represented in Observation 6.3 has not been implemented in the current version of Atlas. In Chapter 7, however, I shall introduce a new, more powerful method to analyze IFS in a complete, closed-form manner, by means of a structure called the *insertion-force stability cone.*

6.8 Choosing an Action

Having enumerated a set of feasible actions, Atlas' next step is choosing among them. This is a crucial decision, because exhaustively exploring all possible assembly sequences may not be practical. In Chapter 3, which considered sequencing but not

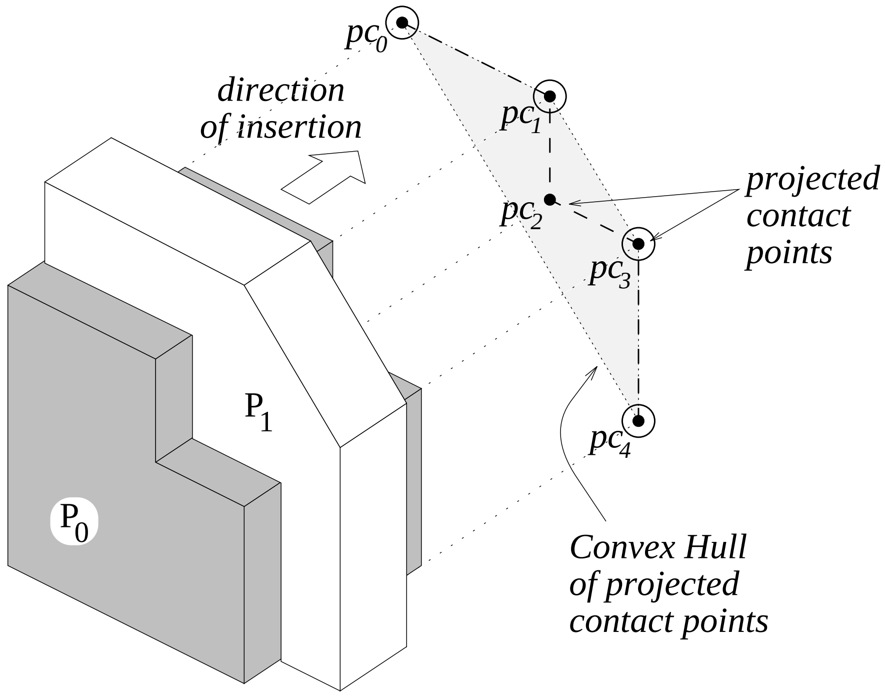


Figure 6.11: For three-dimensional assemblies, IFS can be verified by checking the vertices of the convex hull of the projected contact points.

fixturing, the choice of action was guided by the optimization of an objective function, such as removing a key part as quickly as possible. However, in sequence/fixture co-design, one must first and foremost ensure that the fixture successfully stabilizes the entire assembly sequence. Since this sequence is not fully known until the end, deciding which action will best accomplish that goal is a difficult task indeed.

Nonetheless, several strategies have been found which work well in practice. For example, one approach is to minimize the number of deleted fixels (or deleted virtual fixels) in each step. In the planar case this is essentially equivalent to minimizing the number of deleted fixture segments. The assumption here is that a fixture with more remaining fixels will be more likely to stabilize future, as yet undetermined, subassemblies.

While this method is generally a good one, it can sometimes lead to dead ends in the disassembly process. For example, consider the "tilted blender" example of Fig. 6.12a. If the system deletes as few virtual fixels as possible in each step, it eventually arrives at the subassembly of Fig. 6.12b. No further actions are feasible, primarily because of the need to counteract insertion forces. If one attempts to remove the lid to the left, the leftmost part of the fixture must be deleted, which would leave nothing to counterbalance the insertions of the rightmost parts. Likewise, attempts to remove the jar to the right will leave the fixture unable to counterbalance the insertion of the small "lid cap" from the left.

One can gain additional insight into this problem, and infer a better heuristic for choosing actions, by examining the wrench-space representation of this assembly.

6.9 Wrench Space and Convex Hulls

An assembly is stable if and only if there exists a solution to eqn. (6.2) (for the case of glued parts; the case of unglued parts is similar). That is, stability is achieved if each negated external wrench $-\vec{W}_{ext,j}$ is expressible as a nonnegative linear combination of the fixel wrenches $\vec{w}_i$. Graphically, this implies that each $-\vec{W}_{ext,j}$ must be on or inside the "cone" of $\vec{w}_i$'s, as shown in Fig. 6.13. If one were to project the wrenches onto a unit sphere, as Brost and Mason did in [11], the projected $-\vec{W}_{ext,j}$'s would have to lie

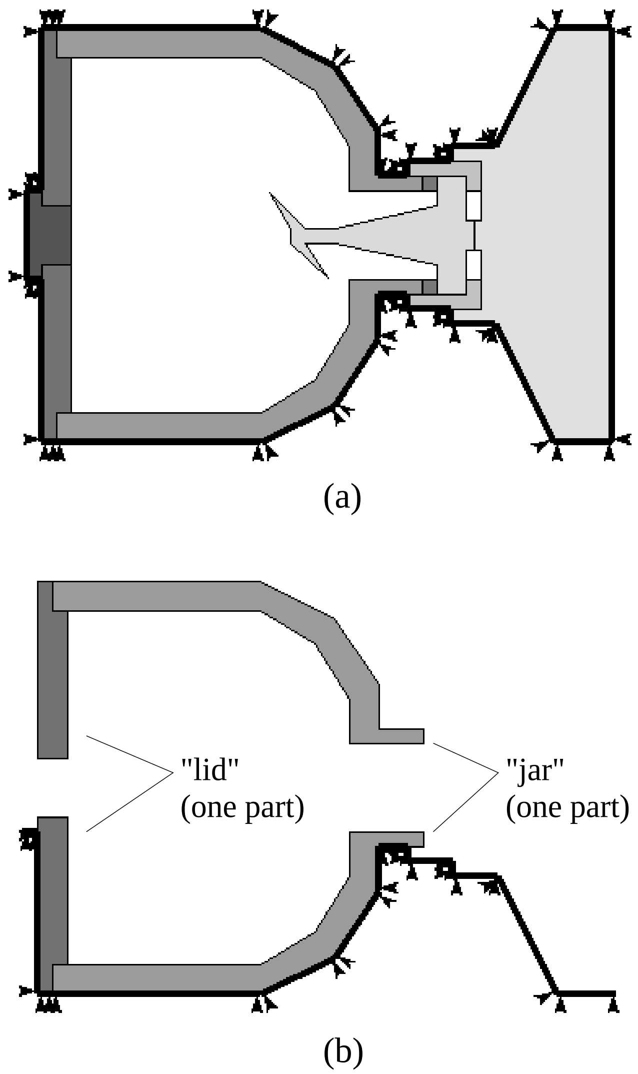


Figure 6.12: The "tilted blender" example: (a) After overfixturing, with virtual fixels shown explicitly. (b) The dead-end situation that results from always deleting as few virtual fixels as possible.

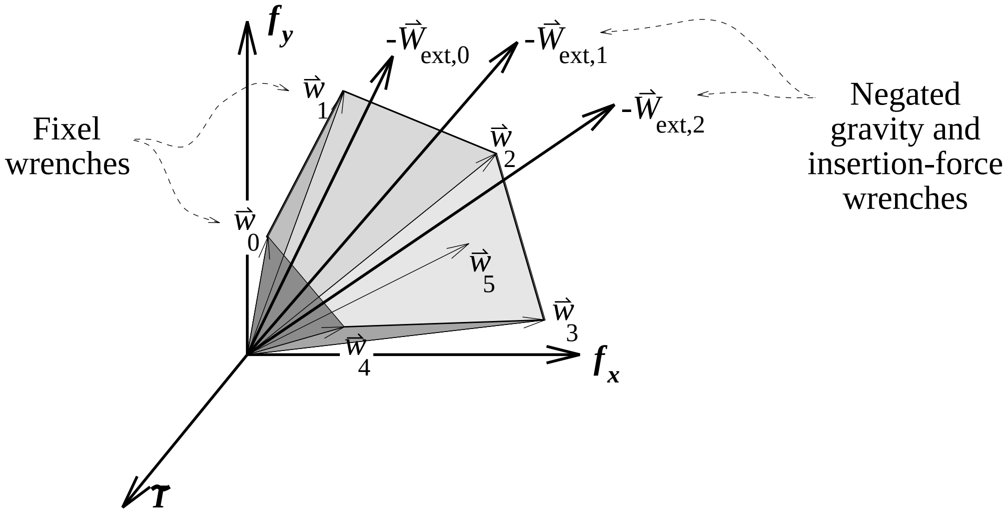


Figure 6.13: Wrench-space representation of the stability condition. Each negated external wrench $-\vec{W}_{ext,j}$ must be contained in the "cone" of fixel wrenches $\vec{w}_i$, in order to achieve stability.

on or inside the projected convex hull of the $\vec{w}_i$'s. Alternatively, the *ordinary* convex hull of the $\vec{w}_i$'s and the origin must contain $-\epsilon\vec{W}_{ext}$ for some suitably small ϵ, $\epsilon > 0$. Note that in practice, this convex hull may not be cone-shaped; instead, it may take a variety of convex forms, such as a flat disc, or a "polyhedral sphere" completely surrounding the origin.

Over the course of the sequence/fixture generation, Atlas will be deleting fixels (and their wrenches) to make way for part removals, and adding new external wrenches corresponding to the gravity and insertion forces on different subassemblies. Throughout this process, it must ensure that all the external wrenches are contained in the fixel-wrench convex hulls in this manner.

An example of an actual wrench-space configuration is shown in Fig. 6.14. This situation corresponds to the "tilted blender" example of Fig. 6.12, under the glued-parts assumption, and in particular to the dead-end case of Fig. 6.12b. As can be seen, the disassembly subsequence leading up to this point has produced a number of gravity and insertion-force wrenches which must be contained.

And here the difficulty of the current predicament can be seen: the external wrenches are so spread apart that maintaining containment of them requires the use

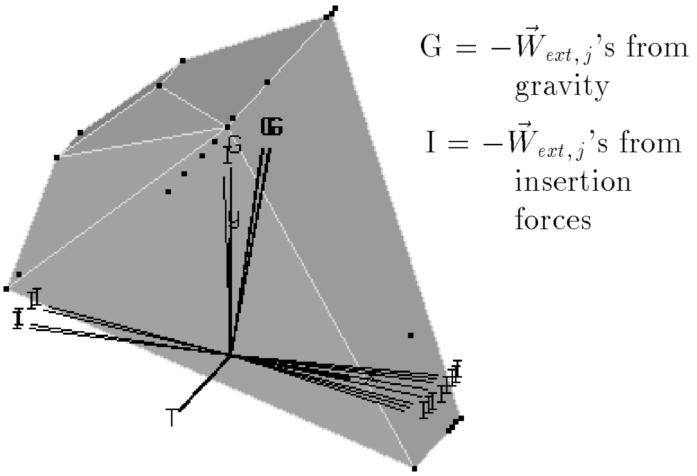


Figure 6.14: The wrench-space representation of Fig. 6.12b.

of most of the remaining fixels. As a result, Atlas does not have sufficient flexibility in deleting fixels. This observation suggests a disassembly strategy which keeps these wrenches more "clustered" together, or seeks redundancy in the spacing of the fixel wrenches, or both.

Clustering among the external wrenches can be measured in a variety of ways. If the parts are glued, the simplest might be to compute the average dot product among the external wrenches (after scaling the torques to bring their magnitudes in line with those of the forces). A more sophisticated approach might analyze the convex hull of the external wrenches, since containment of the wrenches on that convex hull implies containment of all externals.

For the case of unglued parts, the dot-product criterion would have to be modified, since any external wrenches ($\in \Re^{6n}$) affecting different parts would necessarily have a zero dot product. One modification would be first to project these wrenches onto $\Re^3$ or $\Re^6$ before comparing them; in essence this treats the assembly as a single unit for purposes of comparing external wrenches, despite the assembly's many internal degrees of freedom.

Redundancy in the fixel wrenches is much harder to measure, and identifying useful measures of fixture redundancy remains a subject for future research. One possible approach, however, might be to find the minimum number of fixels whose removal would destabilize the assembly in some way.

Atlas selects the action which maximizes clustering, as measured by the dot-product test (using an $\Re^3$ projection in the case of unglued parts). This simple strategy was found to work well on almost all the examples tested.

6.9.1 Convex Hull Complexity

In general, a convex hull of m points in d dimensions has complexity $O(m^{\lfloor d/2 \rfloor})$ [63]. Therefore, if the parts are glued together and act as a rigid body (so d=constant), the complexity is polynomial in the number of fixels m (but constant in the number of parts). If the n parts are not glued, however, the wrench space has dimension $d = 3n$ or $6n$, and the complexity is exponential in the number of parts. Therefore, for larger assemblies, an explicit convex hull computation is only practical under the glued-parts assumption. This limitation may constrain the development of more sophisticated criteria for selecting actions.

Furthermore, a number of rather subtle complications exist to the convex hull structure described thus far. The next two subsections explore some of these complexities.

6.9.2 Convex Hull "Shells"

When parts are removed, the fixels which touched them become *inactive*; that is, they are no longer useful in stabilizing the remaining subassemblies, because they are not in contact with any part. (The fixels which are still in contact with the assembly are the *active* fixels.) For example, in Fig. 6.12b, the fixels on the lower right are inactive. Inactive fixels, however, are still valid for stabilizing the earlier subassemblies. Note that this situation is different from that of deleted fixels, which cannot be considered at all.

In wrench-space terms, inactive fixels may be used to contain external wrenches which affect the earlier subassemblies, but not wrenches affecting the later ones. Therefore, different convex hulls are applicable at different stages of the disassembly process. Since these hulls get progressively smaller as the disassembly proceeds, the overall structure is a series of convex-hull "shells", one inside the next. And the external wrenches do not all correspond to the same shell; the requirement that an external wrench be contained by the convex hull actually means that it must be contained by the particular shell corresponding to the subassembly which that wrench affects. This observation further complicates the assessment of a fixture's future usefulness in stabilizing the assembly sequence.

6.9.3 Disconnected Assemblies

Another complication arises when a subassembly becomes disconnected; that is, when its parts can be grouped into multiple connected components which do not touch each other. In this case, the assumption that the parts are glued together upon contact no longer ensures that the assembly will behave as a single rigid body (for indeed, components which do not touch cannot be glued together).

In this situation, the effect of a fixel on the assembly is properly described by a wrench of dimension $3c$ (for a planar assembly) or $6c$ (for a three-dimensional assembly), where c is the number of connected components. In wrench space, this is similar to an assembly of c unglued parts, but with two key differences. First, the absence of contacts between the components decouples the stability check into c independent subproblems. Second, the previously-generated subassemblies are still described by simple 3- or 6-dimensional wrenches (if parts are glued), and later subassemblies may be, too. So the wrench space is of non-constant dimensionality.

Fortunately, the dot-product "clustering" test can still be made to work. The $\Re^6$ projections of the $\Re^{6c}$ wrenches correctly describe the effect of each wrench on a particular component. And if the pivot point for the torques remains the same, then these projections are entirely comparable to the $\Re^6$ wrenches from the earlier subassemblies. Therefore, maximizing the dot product between the earlier $\Re^6$ external wrenches and the projections of the new $\Re^{6c}$ external wrenches will realize the goal of containing these wrenches with a relatively small number of fixel wrenches.

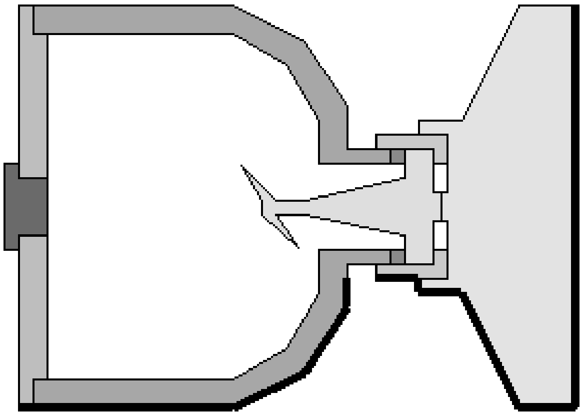


Figure 6.15: The "tilted blender" assembly in its final (non-minimized) fixture, following a successful disassembly. The generated assembly sequence inserts the rightmost part downward, and then all other parts from left to right.

6.10 Experimental Results

Using this dot-product "clustering" strategy, Atlas was able to successfully disassemble the blender example from Fig. 6.12. The final fixture is shown in Fig. 6.15. The corresponding assembly sequence inserts the rightmost part downward, and then all other parts from left to right. The final wrench-space configuration is shown in Fig. 6.16. For the case of glued parts, this solution was generated in 0.05 seconds of CPU time on a DEC Alpha workstation. For the case of unglued parts, the same solution was generated in 0.41 CPU seconds. The higher time reflects the much greater complexity of the stability analysis in the unglued case.

In another example, Atlas was run on an 11-part cross-section of an "A$'$ Discriminator", a safety device being developed at Sandia National Laboratories. It generated the fixture of Fig. 6.17 in 0.16 seconds (glued parts) or 1.08 seconds (unglued parts), along with a sequence that inserts all parts straight down. Again, the same solution was found for both cases. Although this solution seems fairly straightforward, the example is interesting nonetheless because one subassembly is disconnected (consisting specifically of the two largest parts), and therefore the projection approach described in Section 6.9.3 had to be applied.

A third example is shown in Fig. 6.18. The corresponding sequence inserts the box

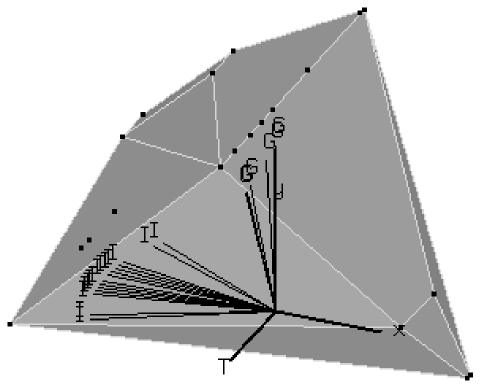


Figure 6.16: The wrench-space representation of Fig. 6.15, showing all the external wrenches from the assembly process.

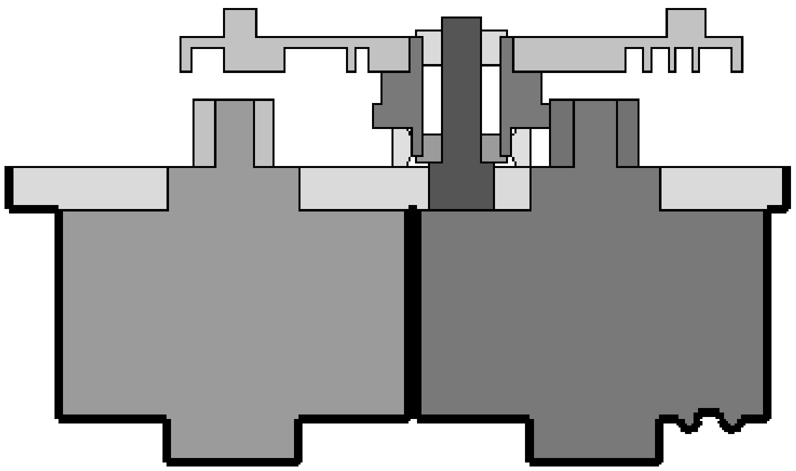


Figure 6.17: The A′ Discriminator example (courtesy Sandia National Laboratories), in its final fixture.

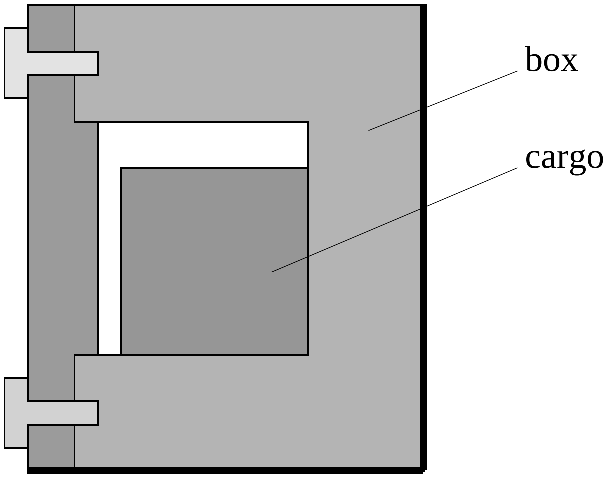


Figure 6.18: The "box-lid-cargo" example, in its final fixture.

straight down and then all other parts from left to right. This solution was generated in 0.02 CPU seconds (glued parts) or 0.09 CPU seconds (unglued parts).

As a final example, Atlas generated a sequence and fixture for the fan assembly of Fig. 6.19 in 0.03 CPU seconds (glued parts) or 0.29 CPU seconds (unglued parts). The fixture is as shown. The sequence inserts the four "thick" parts straight down; then the back of the cage, the fan blades, and the cap from right to left; and finally, the front of the cage diagonally down and to the left. Note that the right-to-left insertions of the thin parts will create insertion forces which can, in fact, be balanced by the fixels around the base, but which may demand very large fixel forces there. An intelligent CAD system might use this analysis to suggest to the designer that the fan be redesigned or assembled in a different orientation.

6.11 Motivation for the Next Step

These results demonstrate Atlas' ability to take a geometric description of a product and very quickly generate both an assembly sequence and a fixture for that product. However, on a fundamental level, the approach remains a "trial and error" one, in several respects.

First of all, the LTF cone completely describes the set of motion directions that are

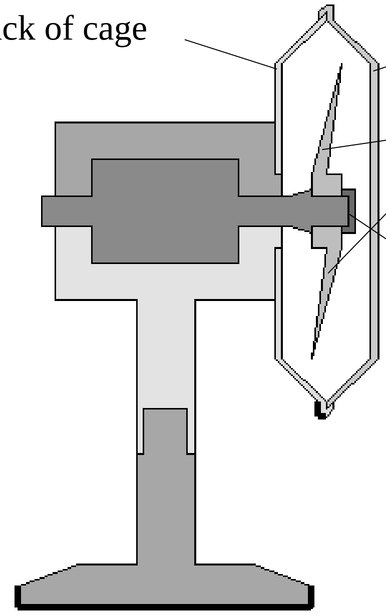


Figure 6.19: The fan example, in its final fixture.

consistent with *local* freedom constraints, but not necessarily consistent with global constraints. As in Chapter 3, it is always possible that a collision-free extended motion exists, but for every discretized direction the system happens to check, there is some distant part blocking the way. In this case, an LTF-based approach will not find the allowable motion.

Second, there is no way of knowing in advance whether any of the proposed actions generated this way will violate insertion-force stability requirements, or whether a stable action even exists. For gravitational stability requirements, there is little problem, because gravitational stability depends only on the subassembly, not on any direction of motion. For an n-part assembly, one could check all n possible subassemblies which would remain after the next step and know with certainty whether a gravitationally-stable subassembly exists (at least in the current fixture). However, insertion-force stability requirements do depend on the direction of part removal; and heretofore no guaranteed method has been described to determine the existence or identity of a "safe" removal direction. Instead, the preceding algorithms have simply searched for one on a "trial and error" basis.

The next chapter shall remedy these shortcomings.

Chapter 7

A Complete Algorithm for Co-Design

In this chapter, I present a technique for sequence/fixture co-design which is complete, in the sense that it is guaranteed to find a solution when one exists under certain conditions. The approach, based on the extended translational freedom (ETF) cone and the new concept of the insertion-force stability (IFS) cone, is most applicable to the case where parts are glued together upon insertion. Therefore, except as noted, it is assumed in the discussions which follow that parts are indeed glued.

I begin by describing the ETF cone and the IFS cone, and then show in Section 7.3 how they can be used to generate the complete set of stable fixture-preserving actions for an assembly. I then expand this approach to encompass fixture-reducing actions as well, first for the case of discrete fixtures and later for continuous fixtures. This co-design approach provides the aforementioned guarantee.

The technique presented here has been implemented in a new version of Atlas, called "Atlas 3". (Atlas 1 was a relatively simple co-design system never presented in a published paper; Atlas 2 was described in the preceding chapter.) Experimental results for Atlas 3 are presented in Section 7.7, followed by some additional remarks.

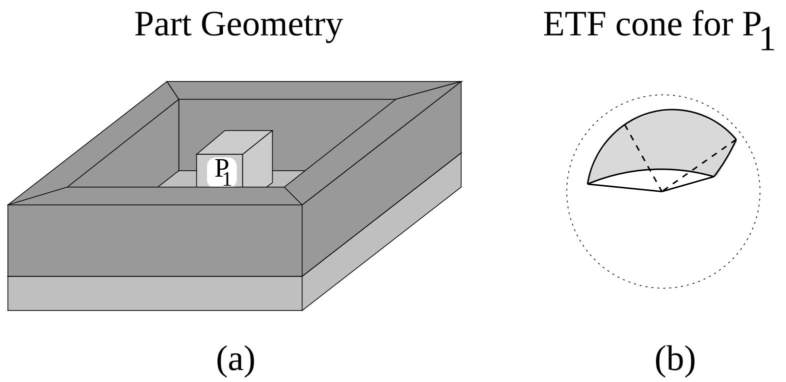


Figure 7.1: An example of an extended translational freedom (ETF) cone.

7.1 The Extended Translational Freedom Cone

The *extended translational freedom cone*, or ETF cone, is the means by which Atlas avoids collisions with distant obstacles in more than a "trial-and-error" manner. The ETF cone is defined as the subset of directions, from the entire sphere of directions, over which a part can translate to infinity without colliding with another part [66]. For example, in Fig. 7.1a, the possible motions of P_1 are described by the ETF cone of Fig. 7.1b. Note that in general, the ETF cone may not be cone-shaped at all; it may be non-convex, empty, degenerate, or composed of several disjoint regions.

In computational geometry terms, the ETF cone of a part P_0 with respect to another part P_1 is equivalent to the unit sphere less the projection onto that sphere of the Minkowski difference of the two parts. Techniques to perform these computations efficiently in the general three-dimensional case are described in [65]. For planar assemblies, however, the generation of ETF cones is much simpler. A planar ETF cone is simply one or more "ranges" of angles of permissible motion, as illustrated in Fig. 7.2. It can be computed by considering, in turn, the constraints imposed by every face of the stationary part P_1 on every face of the moving part P_0; the ETF cone is then the 360° motion circle minus the union of all these constraints. (Conveniently, once it has been found, the ETF cone of P_1 with respect to P_0 is identical except for a 180° rotation.)

The ETF cone of a part P_0 with respect to all other parts specifies *exactly* what

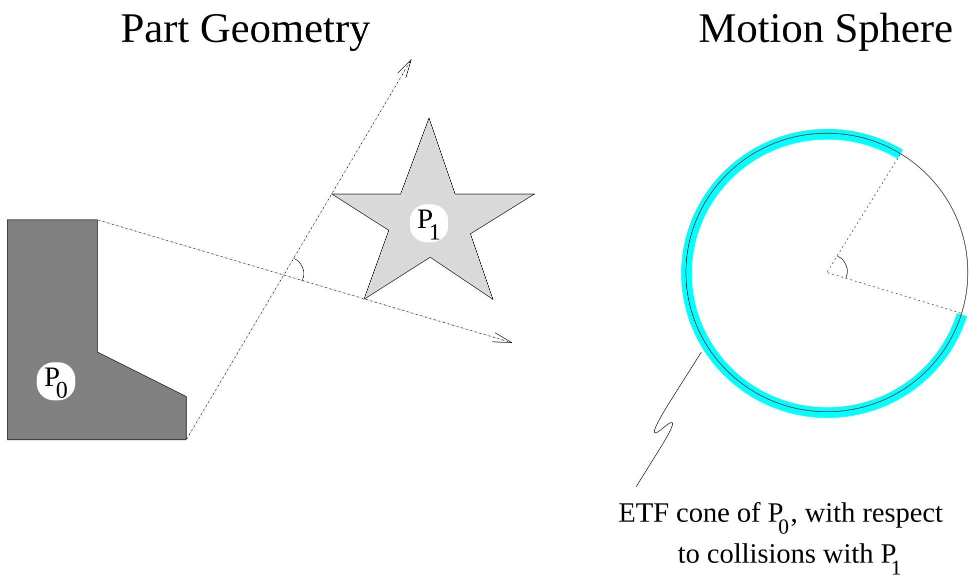


Figure 7.2: ETF's are much simpler to compute in the planar case.

directions of motion for P_0 will be collision-free. Once it has been computed, no part-against-part sweeping operations (as were performed by STAAT and Atlas 2) are necessary.

This structure is the basic building block for the "extended NDBG" described in Section 3.4. Specifically, the extended NDBG sphere is partitioned by overlaying the ETF cones of every part with respect to every other part, as was done with LTF cones for the local NDBG. Each resulting region on the motion sphere exhibits a constant blocking relationship among all parts for extended translations. (For more details on extended NDBG's, the reader is referred to [65].)

Projected Fixtures. In a similar manner, discrete or continuous fixture elements can be projected onto a part's motion sphere, to identify the set of directions that would cause the part to collide with the fixture. The complement of this set (*i.e.*, the set of directions that do not cause part-fixture collisions) can then be intersected with the ETF cones to produce the complete set of directions causing neither part-part nor part-fixture collisions. In other words, this procedure yields the complete set of fixture-preserving actions, or FPA's, for the given part (although without regard to stability).

One more point about this technique is worth mentioning. Because so many ETF-based collision checks are made over the course of a disassembly analysis, the ETF cones can be stored in a table, so that subsequent requests for the same ETF cone(s) can be met much more quickly. Similarly, projections of fixture segments against parts can also be cached for efficiency. These optimizations have been implemented in Atlas 3.

7.2 The Insertion-Force Stability Cone

Let us now explore the method by which insertion-force stability constraints can be handled in a complete manner. Let us suppose, temporarily, that the fixture is constant, *i.e.,* any actions which eliminate portions of the fixture are excluded from consideration.

Each planar contact region in the assembly shall be represented by a finite collection of discrete contact points, located at the vertices of the contact region's convex hull. As established earlier in Corollary 6.2 (p. 81), stability against insertion forces applied at each such point automatically implies stability against the same insertion forces applied anywhere in the contact region.

Now, suppose part P_0 is to be removed from the rest of the assembly. Let the positions of the contact points between P_0 and the remaining subassembly be denoted $\{\vec{c_j}\}$. If the part is removed in direction $\vec{d}$, the corresponding insertion in direction $-\vec{d}$ will induce an insertion-force wrench of

$$\vec{W}_{IF} = \begin{pmatrix} -\vec{d} \\ \vec{c_j} \times (-\vec{d}) \end{pmatrix} \tag{7.1}$$

on the subassembly, for every contact point c_j. The negated insertion-force wrench, then, will be

$$-\vec{W}_{IF} = \begin{pmatrix} \vec{d} \\ \vec{c_j} \times \vec{d} \end{pmatrix} \tag{7.2}$$

To achieve stability, each such $-\vec{W}_{IF}$ must satisfy eqn. (6.2); that is, there must be some nonnegative $\alpha_i \in \Re$ such that

$$[\vec{w}_0 \; \vec{w}_1 \; \cdots \; \vec{w}_{m-1}] \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} = -\vec{W}_{IF}$$

where $\{\vec{w}_i\}$ are the wrenches produced by each fixel or virtual fixel, for a unit fixel force.

However, an equivalent way to express the stability constraint is in terms of the *wrench-space convex hull* (or *WSCH*) of the fixel wrenches. As illustrated in Fig. 6.13 (p. 89), each $-\vec{W}_{IF}$ must be contained in this "cone" of fixel wrenches. (For clarity, let us explicitly declare the WSCH to be bounded only by planes passing through the origin, not by any other planes in the "ordinary" convex hull of $\{\vec{w}_i\}$.[1]) Mathematically, for every plane in the WSCH, with inward normal $\vec{n}_i$, the requirement for stability is that

$$(-\vec{W}_{IF}) \cdot \vec{n}_i \geq 0 \;, \qquad \forall \, i \tag{7.3}$$

Combining eqns. (7.2) and (7.3), the requirement is that

$$\begin{pmatrix} \vec{d} \\ \vec{c_j} \times \vec{d} \end{pmatrix} \cdot \vec{n}_i \geq 0 \;, \qquad \forall \, i, j \tag{7.4}$$

Denote the translational and rotational components of $\vec{n}_i$ as $\vec{n}_{i,tr}$ and $\vec{n}_{i,rot}$, respectively:

$$\vec{n}_i \triangleq \begin{pmatrix} \vec{n}_{i,tr} \\ \vec{n}_{i,rot} \end{pmatrix} \tag{7.5}$$

Then eqn. (7.4) becomes

[1]Note that if the origin were in the interior of this ordinary convex hull, then form closure would be achieved, assuring stability against *any* external wrench [58]. However, this possibility is precluded by the previous creation of a final removal path for the assembly.

$$\begin{pmatrix} \vec{d} \\ \vec{c_j} \times \vec{d} \end{pmatrix} \cdot \begin{pmatrix} \vec{n}_{i,tr} \\ \vec{n}_{i,rot} \end{pmatrix} \;\geq\; 0\,, \qquad \forall\, i,j \tag{7.6}$$

$$\vec{d} \cdot \vec{n}_{i,tr} \;+\; \left(\vec{c_j} \times \vec{d}\right) \cdot \vec{n}_{i,rot} \;\geq\; 0\,, \qquad \forall\, i,j \tag{7.7}$$

Applying the rule $(\vec{A} \times \vec{B}) \cdot \vec{C} = \vec{B} \cdot (\vec{C} \times \vec{A})$,

$$\vec{d} \cdot \vec{n}_{i,tr} \;+\; \vec{d} \cdot (\vec{n}_{i,rot} \times \vec{c_j}) \;\geq\; 0\,, \qquad \forall\, i,j \tag{7.8}$$

$$\boxed{\vec{d} \cdot (\vec{n}_{i,tr} \;+\; \vec{n}_{i,rot} \times \vec{c_j}) \;\geq\; 0\,, \qquad \forall\, i,j} \tag{7.9}$$

Thus, the insertion-force stability requirements have been mapped into constraints directly on the removal direction $\vec{d}$. These are, furthermore, half-space constraints, and the set of directions $\vec{d}$ which satisfy them all is a convex cone of similar form to the LTF cone seen earlier. This structure is called the *insertion-force stability cone*, or *IFS cone*. That is, the IFS cone is the set of directions, computed through eqn. (7.9), over which insertion-force stability is achieved.

Likewise, bounds could also be placed on the removal directions under other models of possible insertion forces. For example, if, as shown in Fig. 7.3, an insertion in direction $-\vec{d}$ could induce insertion forces in a range of directions bounded by $-\vec{d}$ and the inward surface normal $\vec{n}_{surf}$, then the preceding analysis could be supplemented by a simple yes-or-no IFS check in direction $\vec{n}_{surf}$. By similar reasoning as that used for Corollary 6.2, stability against insertion forces in directions $-\vec{d}$ and $\vec{n}_{surf}$ (for the same contact point) can be shown to establish stability against any insertion force between $-\vec{d}$ and $\vec{n}_{surf}$.

For two-dimensional assemblies, a similar result to eqn. (7.9) applies. As before, each negated insertion-force wrench

$$-\vec{W}_{IF} = \begin{pmatrix} \vec{d} \\ \vec{c_j} \times \vec{d} \end{pmatrix} = \begin{pmatrix} \vec{d} \\ c_{j,x}\; d_y - c_{j,y}\; d_x \end{pmatrix} \tag{7.10}$$

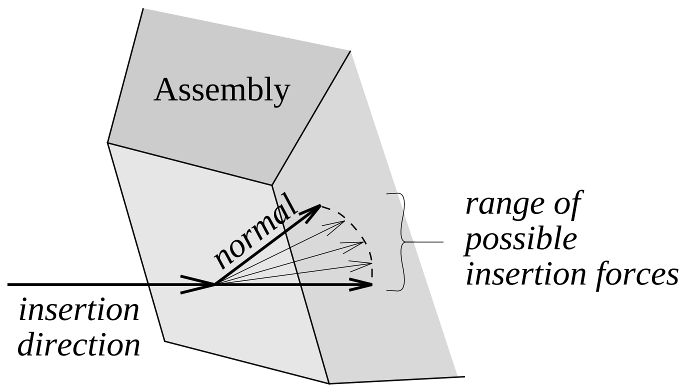


Figure 7.3: Another model of possible insertion forces, which the techniques of this chapter could support with a slight modification.

must satisfy

$$(-\vec{W}_{IF}) \cdot \vec{n}_i \geq 0 \,, \qquad \forall\, i \tag{7.11}$$

in order to be contained in the WSCH. Hence,

$$\begin{pmatrix} \vec{d} \\ c_{j,x}\ d_y - c_{j,y}\ d_x \end{pmatrix} \cdot \vec{n}_i \geq 0 \,, \qquad \forall\, i, j \tag{7.12}$$

This time, let

$$\vec{n}_i \triangleq \begin{pmatrix} n_{i,x} \\ n_{i,y} \\ n_{i,\tau} \end{pmatrix} \tag{7.13}$$

Then eqn. (7.12) becomes

$$\begin{pmatrix} d_x \\ d_y \\ c_{j,x}\ d_y - c_{j,y}\ d_x \end{pmatrix} \cdot \begin{pmatrix} n_{i,x} \\ n_{i,y} \\ n_{i,\tau} \end{pmatrix} \ \geq\ 0 \,, \qquad \forall\, i, j \tag{7.14}$$

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} \cdot \begin{pmatrix} n_{i,x} \\ n_{i,y} \end{pmatrix} + (\ c_{j,x}\ d_y\ n_{i,\tau} - c_{j,y}\ d_x\ n_{i,\tau}\ ) \ \geq\ 0 \,, \qquad \forall\, i, j \tag{7.15}$$

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} \cdot \begin{pmatrix} n_{i,x} - c_{j,y}\ n_{i,\tau} \\ n_{i,y} + c_{j,x}\ n_{i,\tau} \end{pmatrix} \ \geq\ 0 \,, \qquad \forall\, i, j \tag{7.16}$$

$$\boxed{\vec{d} \cdot \begin{pmatrix} n_{i,x} - c_{j,y}\, n_{i,\tau} \\ n_{i,y} + c_{j,x}\, n_{i,\tau} \end{pmatrix} \geq 0 \,, \qquad \forall\, i, j} \tag{7.17}$$

Once again, the insertion-force stability requirements have been mapped into constraints directly on $\vec{d}$.

Now, to be sure, the IFS cone is the intersection of many half-space constraints; the number of them equals the number of faces in the WSCH multiplied by the number of contact points between the part and the subassembly. However, as I shall show later, this quantity is still polynomial in the number of vertices in the assembly.

Several examples of IFS cones are shown in Fig. 7.4. These cones, which were computed from eqn. (7.17), provide stability information which is confirmed by intuition. For example, in Fig. 7.4a, the three fixture segments secure the lower part against insertion forces in any direction in the lower half of the motion circle; therefore, as shown by the IFS cone, the upper part may be removed in any upward direction. In Fig. 7.4b, insertion-force stability is assured only for insertions which move down and/or to the left; the IFS cone reflects the corresponding quarter-circle of possible removal directions. Finally, in Fig. 7.4c, the lower part will be stable as long as the insertion forces are mostly downward, and not too far to the left or right; therefore, the allowable removal directions are in a small cone centered around $+\hat{y}$. An example of the type of instability that arises due to insertions outside the prescribed range is illustrated in Fig. 7.4d.

It should be remembered that in eqns. (7.9) and (7.17), $\vec{c}_j$ refers not to the orientation of a contact, but to its position with respect to the center of mass of the assembly. Therefore, unlike the contact's non-penetration constraints, which constrain the two given parts in opposite directions, the constraints due to eqns. (7.9) and (7.17) treat the two parts alike. The specific parameters are not the same, because the WSCH for the removal of part A would be different from the WSCH for the removal of part B; but fundamentally, the IFS constraints on A and B are "parallel", not "mirror images" as with the non-penetration constraints.

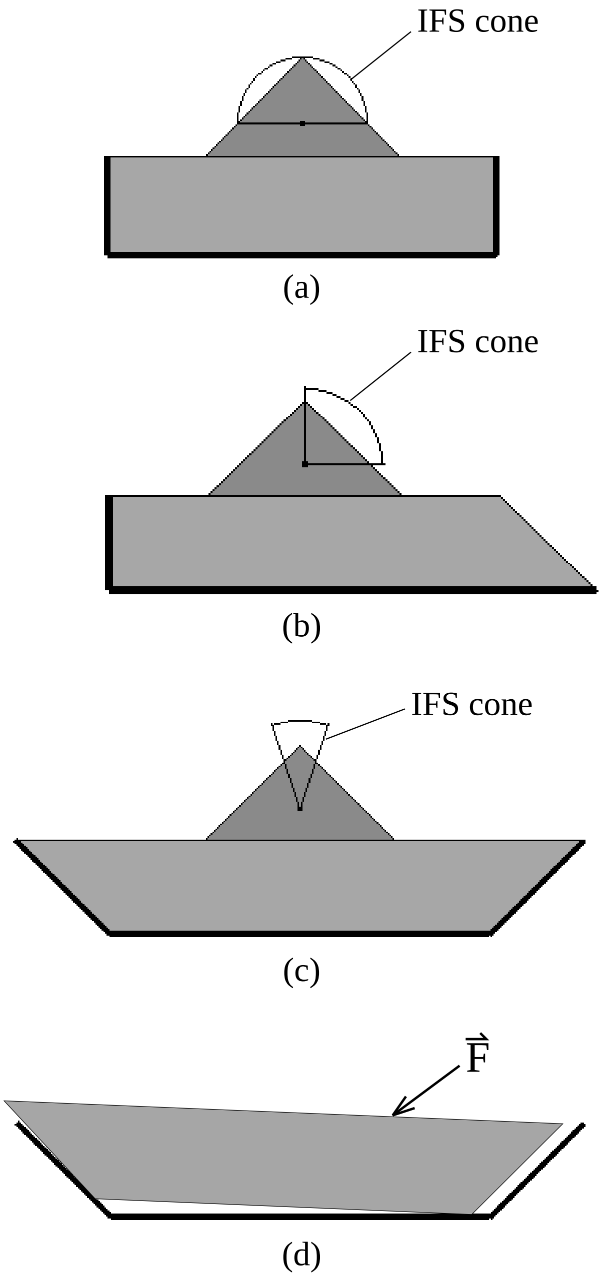


Figure 7.4: (a)–(c). Examples of IFS cones. In each case, the IFS cone shows the set of removal directions for the upper part such that the corresponding insertion would not destabilize the lower part in the given fixture. (d) An example of instability against an insertion force $\vec{F}$, arising from an insertion outside the prescribed range of directions in example (c).

7.3 Sequencing for a Known Fixture

The IFS cone provides a powerful new tool to compute disassembly actions. Given a part P_i to be removed, the mutual intersection of:

- its IFS cone, $\mathcal{I}$,
- its ETF cone with respect to the rest of the assembly, and
- the complement of the projected fixture (on the motion sphere), $\neg\mathcal{F}$

produces the complete set of directions over which P_i can be removed without colliding with either a part or the fixture, nor causing insertion-force instability. Mathematically, this operation can be written as:

$$SETF = \mathcal{I} \cap ETF \tag{7.18}$$

$$\mathcal{D} = SETF \cap (\neg\mathcal{F}) \tag{7.19}$$

The result of the first intersection is the *Stable Extended Translational Freedom cone*, or SETF cone. It is the set of all directions in which the given part could move without colliding with another part, nor causing insertion-force instability under the current fixture. (Computing it as a separate step will lead to optimizations later.)

An example showing two such $\mathcal{D}$'s appears in Fig. 7.5. In this example, each of the smaller parts appears to be removable further to the left than indicated; however, the corresponding part insertions would impose insertion forces on the lower part which could not be counteracted. The IFS cone provides the precise limits on the stable insertion directions.

A simple gravitational stability check on the remaining subassembly verifies the final precondition for part removal, establishing the following result.

Result 7.1 *The set of all stable, fixture-preserving actions for an assembly can be found using the algorithm of Fig. 7.6.*

Note that this method eliminates the need for linear programs to perform IFS checking; they are used now only to ensure gravitational stability. Even that use of linear

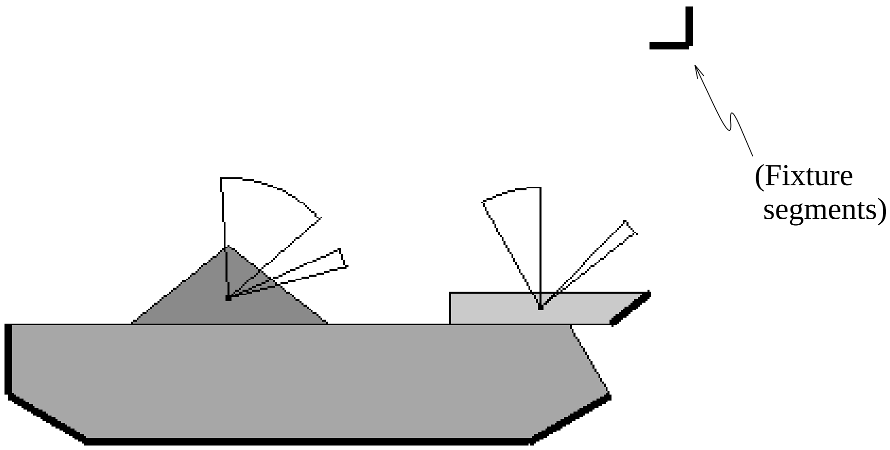


Figure 7.5: An example showing, for each of two parts, the set of stable, fixture-preserving directions of motion for that part.

Procedure *FIND_FPAs*

For each part P_i:

- If the remaining subassembly, $S = A \setminus P_i$, is unstable under gravity, move on to the next part.
- Compute the WSCH of S; then compute the IFS cone of P_i, from the WSCH. Denote that IFS cone as $\mathcal{I}$.
- Compute the ETF cone of P_i with respect to S.
- Let $SETF \leftarrow \mathcal{I} \cap ETF$. If $SETF = \emptyset$, move on to the next part.
- Project the fixture onto P_i's motion sphere; let $(\neg\mathcal{F})$ be the set of directions which do not produce a part/fixture collision.
- Let $\mathcal{D} \leftarrow SETF \cap (\neg\mathcal{F})$.
- If $\mathcal{D} \neq \emptyset$, then append to the running list of actions the removal of P_i in all directions $\mathcal{D}$ (with no fixture reduction).

Figure 7.6: The procedure to find all stable, fixture-preserving actions.

Procedure *FIND_FPA_SEQUENCE*

Given assembly A and fixture F:

- If $A = \emptyset$, return SUCCESS.
- Using procedure *FIND_FPAs*, find the set $\mathcal{Q}$ of all fixture-preserving actions on A.
- For each part P_i whose removal is proposed in $\mathcal{Q}$:
  - If the subassembly $S = A \setminus P_i$ has already been examined, move on to the next part.
  - Otherwise, create a new node in the disassembly graph for S, and recursively call *FIND_FPA_SEQUENCE* on S and F.
  - If the result is SUCCESS, return SUCCESS.
- Return FAILURE.

Figure 7.7: The procedure to find a stable disassembly sequence for a given fixture.

programs is actually unnecessary, because stability against an external wrench can be verified in terms of containment in the WSCH (as in eqn. (7.3) for insertion-force stability). Completely eliminating linear programs from the procedure of Fig. 7.6 would lead to a more predictable running time, at the expense of having to compute the WSCH even for gravitationally-unstable subassemblies.

By recursively applying this algorithm to the remaining subassemblies, a *complete* disassembly graph can be constructed, in the sense that it contains all feasible partitionings of each subassembly. This method will therefore always produce a disassembly sequence of fixture-preserving actions whenever one exists. The algorithm to do this (building only as much of this graph as is needed to find the sequence) is given in Fig. 7.7.

Result 7.2 *For a given fixture, a stable disassembly sequence can be found whenever one exists, using the algorithm of Fig. 7.7.*

At every stage in the sequence, the WSCH will have to be re-computed by *FIND_-FPAs*. This is necessary because every subassembly has a different set of active fixels stabilizing it. (Conceivably, however, an optimization could be made here to update the WSCH incrementally instead of completely re-computing it each time.)

The only exception to this rule concerns the final part to be removed. At that stage, there are no more part-part contacts, so there cannot be any insertion-force instability. The IFS cone is automatically a full sphere (or circle), and the WSCH need not be computed.

7.4 Exploring New Fixtures

The final fixture which will permit a complete disassembly sequence is, of course, not immediately known. To devise a useful co-design system, some means is required of intelligently exploring new fixtures which bring the system closer to an eventual sequence/fixture solution. Fortunately, the procedures described above can be adapted to do just that.

Consider first the case where the fixture consists of discrete elements, rather than the continuous representation described earlier. Section 7.5.1 will later extend the following approach to the continuous model, but the methods are most easily understood for the discrete case.

In view of the powerful "building block" of *FIND_FPA_SEQUENCE*, which solves the sequencing problem for a given fixture in a complete manner, the search-space representation will consist of multiple single-fixture disassembly graphs, as shown in Fig. 5.3 (p. 64).

As in Chapter 6, the overall approach will be to:

1. overfixture the assembly;

2. clear away a final removal path; and

3. reduce the fixture to a well-selected subset of elements which stabilize the sequence without blocking any of the part insertions.

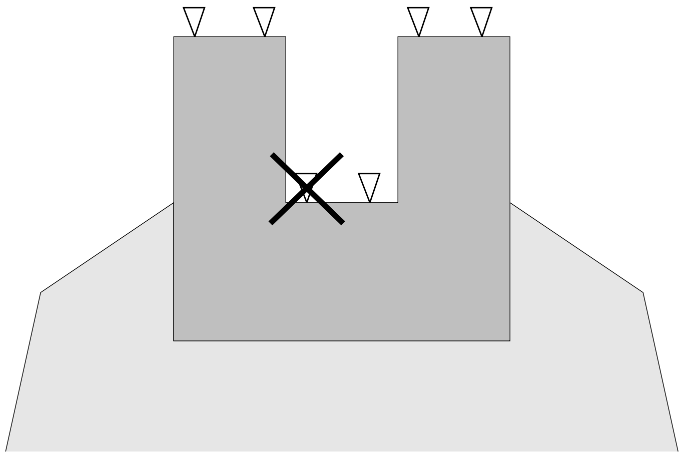


Figure 7.8: An example of a pointless fixture reduction.

For clarity, let the *initial fixture* be defined as the fixture remaining after step #2. Also, let a *subfixture* be formally defined as a fixture whose elements are a subset of those of a given fixture.

The remainder of this section will present a practical method for sequence/fixture co-design, which is complete in the sense that whenever a sequence/fixture solution exists whose fixture is a subfixture of the initial fixture F_0, the algorithm is guaranteed to succeed. (Since the overfixturing process can place arbitrarily many fixels around the assembly, the restriction to subfixtures of F_0 is not terribly constraining.)

7.4.1 A Simple Method

For a discrete fixture, there are only a finite number of subfixtures. So one simple-minded approach would be to search each and every one of them for a feasible sequence; this would surely yield a complete solution in the above sense. However, the number of potential subfixtures, while finite, is enormous: for an initial fixture with k fixels, there are 2^k possible subfixtures. Furthermore, many of the fixture reductions which this method would explore would be utterly pointless; see, for example, Fig. 7.8.

Therefore, a practical algorithm must be selective and explore only "useful" fixture reductions. One characteristic of a "useful" fixture reduction is that it should enable at least one part removal that is not possible under the current fixture. In fact, even if it allows *only* one such part removal, and one which promptly leads to another dead end, the reduction is still useful insofar as it brings the system potentially closer to a solution. Further iterations of this approach, finding each time all "useful" reductions, will eventually produce the final solution when one exists.

7.4.2 A More Practical Method

It is this reasoning which is the basis for Atlas' approach. Given a part which cannot be removed from the assembly under the current fixture, Atlas identifies whether any fixture reduction(s) might enable that part's removal, and if so, adds the resulting fixtures to the list of fixtures to explore.

These fixture reductions are identified through the further use of projected fixels, as follows.

As noted earlier, the position of a discrete fixture element can be projected onto a given part's motion sphere, to identify those directions in which motion by the part would force the elimination of that fixel. However, if instead of excluding such directions from consideration (as was done in Section 7.3), one were to superimpose the projections of all fixels onto the motion sphere, one would obtain a partitioning of the motion sphere into regions, such that the same set of fixels would be deleted for any motion within a region. Put another way, the sphere of directions would be partitioned into equivalence classes, according to the set of deleted fixels. An example of such a partitioning is shown in Fig. 7.9. (In essence, the approach described in Section 7.3 only explored one such class, namely the class requiring no portion of the fixture to be deleted at all.)

Now, some of the fixture reductions found this way may be destabilizing, and still others may correspond to removal directions that produce part-part collisions anyway. However, it is safe to say that no other fixture reduction could possibly permit the removal of the given part from the current subassembly. (More precisely, no fixture reduction which did not encompass one of the given ones would possibly be helpful. It is safe to ignore non-minimal "over-reductions" since the extra fixels can

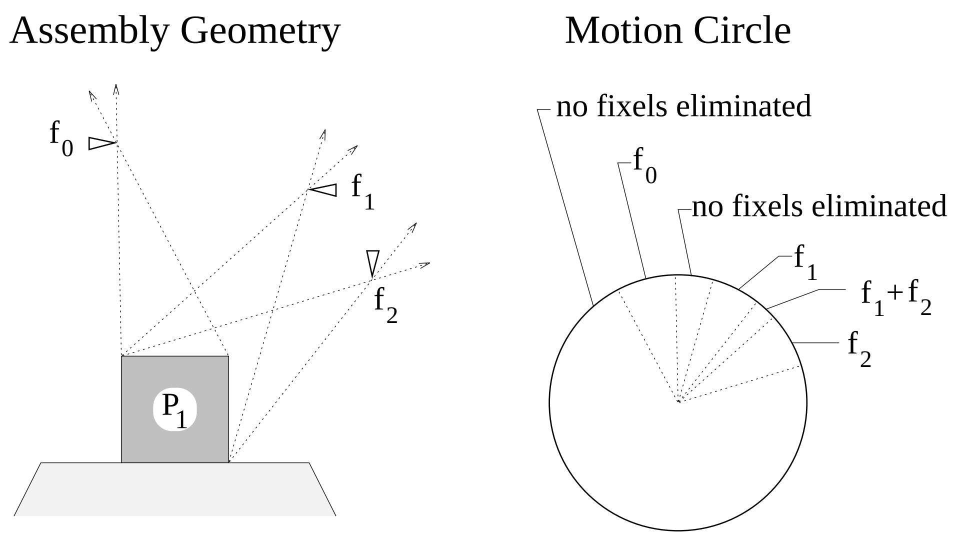


Figure 7.9: A partitioning of the motion circle into equivalence classes, according to the set of deleted fixels.

always be deleted later if necessary.) This method, therefore, enumerates a superset of the minimal, useful, feasible fixture reductions, but a much more manageable one than that used in the simple-minded approach earlier. Furthermore, by intersecting the SETF cone with each equivalence class on the sphere, many unusable fixture reductions can be pruned out.[2]

The algorithm, then, will execute a variant of *FIND_FPAs*/*FIND_FPA_SEQUENCE*, looking for potential fixture-reducing actions (FRA's) in addition to the fixture-preserving actions (FPA's) being sought earlier. In the event that no FPA-only sequence is found, the algorithm will draw upon the set of FRA's to identify new fixtures to explore. For these fixtures it will spin off new disassembly graphs.

In terms of the cone intersections of eqns. (7.18) and (7.19), given by

$$\begin{aligned} SETF &= \mathcal{I} \cap ETF \\ \mathcal{D} &= SETF \cap (\neg \mathcal{F}) \end{aligned}$$

[2]It is true that the SETF cone pertains to the current fixture, not a reduced one; however, fixture reductions can only narrow the SETF cone. Therefore, using the current SETF cone to narrow the range of removal directions is a safe, conservative approach. (It is, incidentally, only in rare and contrived cases that the narrowing of the SETF cone actually makes a difference in partitioning the current subassembly, resulting in some fixture reductions being proposed which are not helpful.)

this approach modifies the $(\neg\mathcal{F})$ term when $\mathcal{D} = \emptyset$ and $SETF \neq \emptyset$. If $SETF = \emptyset$, then the part cannot be stably removed no matter what the fixture, so fixture reductions are irrelevant. Conversely, if $\mathcal{D} \neq \emptyset$, then part P_i is removable by a fixture-preserving action, and reducing the fixture is neither necessary nor helpful.

But if $\mathcal{D} = \emptyset$ and $SETF \neq \emptyset$, then instead of (7.18)–(7.19), the algorithm computes

$$SETF = \mathcal{I} \cap ETF \tag{7.20}$$

$$\mathcal{D}' = SETF \cap \mathcal{R} \tag{7.21}$$

where $\mathcal{R}$ is a region of the partitioned motion sphere, associated with a given fixture reduction. This will produce a set of useful fixture-reducing actions in directions $\mathcal{D}'$. (Because of the fixture reductions, some of these actions may not satisfy stability requirements, but it is actually the fixture reductions themselves which are of interest.)

The algorithm is shown in Figs. 7.10–7.12. With it, the following result is established.

Result 7.3 *Given an assembly, one can find a sequence and a discrete, stabilizing fixture for it, with the guarantee of success whenever a sequence/fixture solution exists whose fixture is a subfixture of the "initial fixture" (defined on p. 110). The algorithm to do this is shown in Figs. 7.10–7.12.*

7.5 Extensions

Let us now consider ways of extending Result 7.3 to a broader domain of conditions, and to the continuous fixture representation described earlier.

7.5.1 Continuous Fixture Representation

The use of a continuous fixture representation immediately raises the problem that there may be an infinite number of distinct fixture reductions which would enable the removal of a given part, as was previously illustrated in Fig. 6.4 (p. 74). Some of these

Procedure *FIND_ACTIONS*

For each part P_i:

- Execute all steps in procedure *FIND_FPAs*.
- If any FPA's were found ($\mathcal{D} \neq \emptyset$), move on to the next part.
- Else partition P_i's motion sphere by projecting all the fixels onto it. For every region $\mathcal{R}$ on the sphere:
  - Let $\mathcal{D}' \leftarrow SETF \cap \mathcal{R}$. If $\mathcal{D}' = \emptyset$, move on to the next region.
  - Append to the running list of FRA's the removal of P_i in all directions $\mathcal{D}'$, with the appropriate fixture reduction.

Return the list of FPA's (for some parts) and the list of FRA's (for other parts).

Figure 7.10: The procedure to find all stable, fixture-preserving actions while also keeping track of all useful fixture-reducing actions.

reductions may be destabilizing, while others may not. A logical question, then, is whether or not the set of stable fixture reductions can be characterized mathematically — or, more to the point, whether the set of stable part-removal directions can be so characterized. Unfortunately, this turns out to be a highly nonlinear problem with no clear solution.

As a result, only a limited guarantee is possible in the continuous case. It can be derived by taking Result 7.3 and gradually "converting" it to the continuous domain, as follows.

Suppose that procedure *DISASSEMBLE* (Fig. 7.12) is run on a particular assembly, and F_0 is the initial fixture (a discrete one) which is computed. Now suppose that the system is modified so that, while executing *DISASSEMBLE* and its subprocedures, it also keeps track, quietly, of *all* regions where fixture elements could be placed, but never takes advantage of that information by placing new fixels. (See Fig. 7.13.) So, at any given stage, the same discrete fixture F_i is used and analyzed as the basis for decisions, but the system knows that fixels at locations $\mathcal{L}_i$ are also ok (*i.e.,* do not

Procedure *FIND_SEQUENCE*

Given assembly A and fixture F:

- If $A = \emptyset$, return SUCCESS.
- Using procedure *FIND_ACTIONS*, find the set $\mathcal{Q}_1$ of all fixture-preserving actions on A, and the set $\mathcal{Q}_2$ of all fixture-reducing actions.
- Generate new fixtures for the actions proposed in $\mathcal{Q}_2$, and append them onto a running list.
- For each part P_i whose removal is proposed in $\mathcal{Q}_1$:
  - If the subassembly $S = A \setminus P_i$ has already been examined under the current fixture, move on to the next part.
  - Otherwise, create a new node in the disassembly graph for S, and recursively call *FIND_SEQUENCE* on S and F.
  - If the result is SUCCESS, return SUCCESS.
- Return FAILURE, but also return the list of proposed new fixtures.

Figure 7.11: The procedure to find a stable disassembly sequence for a given fixture, or, finding none, to suggest new fixtures to explore.

Procedure *DISASM_RECURSIVE*

Given assembly A and fixture F:

- Call procedure *FIND_SEQUENCE* on A and F. If the result is SUCCESS, return SUCCESS.
- Otherwise, let $\mathcal{X}$ be the set of new fixtures found by *FIND_SEQUENCE*. For each fixture F' in $\mathcal{X}$:
  - If F' has been explored before, move on to the next fixture.
  - Start a new disassembly graph for fixture F', and call *DISASM_RECURSIVE* on A and F'. If the result is SUCCESS, return SUCCESS.
- Return FAILURE.

Procedure *DISASSEMBLE*

Given assembly A:

- Overfixture the assembly, and clear away a final removal path. Let F_0 be the resulting fixture.
- Start a disassembly graph for fixture F_0, and call procedure *DISASM_RECURSIVE* on A and F_0.

Figure 7.12: The top-level procedures to perform co-design, for discretized fixtures.

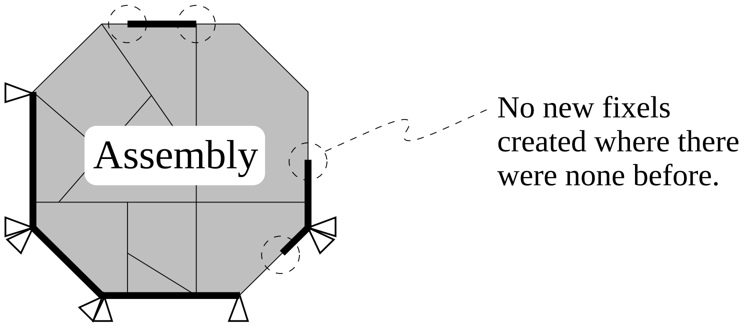


Figure 7.13: A discrete fixture representation, complemented by additional, continuous-fixture information.

block part removals), where $\mathcal{L}_i$ is a region containing all of F_i's fixel locations. (The discrete removal directions used to reduce $\mathcal{L}_i$ are chosen arbitrarily from the ranges of directions $\mathcal{D}'$ in *FIND_ACTIONS*.)

Because the added information does not affect the algorithm, the guarantee contained in Result 7.3 still applies. The only difference is that the output also contains a bit more information about allowable fixturing locations. (For the sake of graph consistency it might, however, be necessary to spin off a new graph for each change in the continuous fixture representation.)

Now suppose the system *can* put in new fixels when reducing a fixture, at locations such as those circled in Fig. 7.13. Such fixels can always be deleted later; and in the meantime, they certainly don't destabilize the assembly. As I shall show below, this added capability will not prevent the algorithm from eventually finding a solution whose fixels are a subset of the initial fixture F_0. And it just might allow the system to find an alternate solution not covered by the guarantee of Result 7.3.

At this point, the proposed system is very similar to the continuous-based system of Chapter 6, with virtual fixels being that chapter's equivalent to the current system's discrete fixels. In fact, only two substantial differences remain between the system of Chapter 6 (Atlas 2), and the system as currently defined. First, in the latter, actions which reduce the continuous representation without eliminating discrete fixels are not treated as fixture-reducing; this is easy to remedy. Second, in Atlas 2, the initial positions of the virtual fixels are predefined, whereas the present system has so far made no assumptions about the initial placement of the discrete fixels. For now, then, let us suppose that the discrete overfixturing process places fixels only at the vertices of the convex hulls of each face, as is the practice for virtual fixels. (This assumption shall be revisited later.)

The resulting system is Atlas 3. Because none of the transformations just described invalidates the guarantee of the previous section, Result 7.4 follows.

Result 7.4 *Given an assembly, one can find a sequence and a continuous, stabilizing fixture for it, with a guarantee of success whenever a sequence/fixture solution exists whose virtual fixels are a subset of those of the initial fixture. The algorithm is as shown in Fig. 7.14.*

It only remains to be shown, then, that the ability to add new fixels to an assembly does not cause the algorithm to fail, given the existence of a solution whose fixels F' are a subset of F_0 (and which, therefore, contain no "added" fixels). This assertion is true because:

1. the addition of fixels never causes F' to be eliminated from the search space of possible fixtures, and

2. the ability to add fixels expands the search space by only a finite amount.

Therefore, if all else fails, the algorithm will eventually find F' in its traversal of the search space.

Statement #1 follows from the fact that if $F' \subset F_0$, then $F' \subset (F_0 \cup \{f_i\})$. Statement #2 follows from the observation that new fixels can be added only when a fixture-reducing action enables a new partitioning; there are only a finite number of possible partitionings, and only a finite number of new fixels to be added with each, so there are only a finite number of new fixels that may ever be added.

Enhanced Overfixturing

As things stand, the condition in Result 7.4 is a rather stringent one. A sequence/ fixture solution may well exist, but it is less likely that one exists whose virtual fixels

Procedure *FIND_ACTIONS_CTS*

For each part P_i:

- Execute all steps in procedure *FIND_FPAs*.
- If any FPA's were found ($\mathcal{D} \neq \emptyset$), move on to the next part.
- Else partition P_i's motion sphere by projecting all the virtual fixels onto it. For every region $\mathcal{R}$ on the sphere:
  - Let $\mathcal{D}' \leftarrow SETF \cap \mathcal{R}$. If $\mathcal{D}' = \emptyset$, move on to the next region.
  - Append to the running list of FRA's the removal of P_i in some direction $d \in \mathcal{D}'$, with the appropriate fixture reduction.

Return the list of FPA's (for some parts) and the list of FRA's (for other parts).

Procedures *FIND_SEQUENCE_CTS,*
DISASM_RECURSIVE_CTS,
DISASSEMBLE_CTS

Identical to their counterparts in Figs. 7.11–7.12, except substituting FIND_ACTIONS_CTS *for* FIND_ACTIONS *(and* FIND_SEQUENCE_CTS *for* FIND_SEQUENCE*, etc.).*

Figure 7.14: The procedures to perform co-design for a continuous fixture.

happen to be a subset of those of the initial fixture. In a three-dimensional assembly, for example, any fixture covering only part of a face may well have virtual fixels outside of the initial "allowed" set. In a two-dimensional assembly, such a fixture *always* has virtual fixels outside of the allowed set.

However, suppose the placement of virtual fixels were to be redefined so that arbitrarily many "superfluous" virtual fixels could be placed on a fixtured face, in addition to those at the vertices of the face's convex hull. For example, they could be added at regular intervals along a grid with an arbitrarily small spacing ϵ, as was done by Mattikalli [56] with discrete fixels. Result 7.4 would still hold, but the applicability of its guarantee would be greatly expanded; that is, the "subfixture" condition which a hypothetical solution would have to meet in order to ensure success would be far easier to satisfy.

I call this modification *enhanced overfixturing*. Just as the placement of many discrete fixels largely circumvents the subfixture condition of Result 7.3, so does enhanced overfixturing circumvent that of Result 7.4. (Indeed, the non-existence of a solution drawn from fixels spaced ϵ apart would imply that in the unlikely event a solution does exist, it must require extremely precise fixel locations and part motions, making it inadvisable anyway.) The continuous-fixture method, meanwhile, still has the advantage of returning an exact set of *all* possible fixturing locations consistent with a sequence. The ability to add new virtual fixels during the analysis enables it to present such fixture solutions even though they may not meet the subfixture condition of Result 7.4.

7.5.2 Unglued Parts

Let us now consider the case where parts are *not* glued together upon insertion. As will be shown below, the IFS cone in this case exists in theory but is impractical to compute.

Given a part P and a direction of removal $\vec{d} \in \Re^3$, each contact $\vec{c_j}$ between P and the remaining subassembly induces an insertion-force wrench of

$$\vec{W}_{IF} = \begin{bmatrix} \vec{0} \\ \vdots \\ \vec{0} \\ \begin{pmatrix} -\vec{d} \\ \vec{c_j} \times (-\vec{d}) \end{pmatrix} \\ \vec{0} \\ \vdots \\ \vec{0} \end{bmatrix}$$

In order to ensure stability, $-\vec{W}_{IF}$ must be contained in the convex hull of the fixel wrenches, which is now a complex structure in $\Re^{6n}$. As before, the requirement is

$$-\vec{W}_{IF} \cdot \vec{n}_i \;\; \geq \;\; 0 \,, \qquad \forall \, i, j \tag{7.22}$$

$$\begin{pmatrix} \vec{d} \\ \vec{c_j} \times \vec{d} \end{pmatrix} \cdot (\vec{n}_i)_k \;\; \geq \;\; 0 \,, \qquad \forall \, i, j \tag{7.23}$$

where $\vec{n}_i$ is the inward normal of a WSCH plane, and $(\vec{n}_i)_k$ is the appropriate $\Re^6$ projection of $\vec{n}_i$. (Specifically, if P_k is the other part involved in contact c_j, then $(\vec{n}_i)_k$ will cover elements $6k$ through $6k + 5$ of $\vec{n}_i$.) Continuing,

$$\vec{d} \cdot (\vec{n}_i)_{k,tr} \; + \; \left[\vec{c_j} \times \vec{d}\right] \cdot (\vec{n}_i)_{k,rot} \;\; \geq \;\; 0 \,, \qquad \forall \, i, j \tag{7.24}$$

$$\vec{d} \cdot (\vec{n}_i)_{k,tr} \; + \; \vec{d} \cdot [\, (\vec{n}_i)_{k,rot} \times \vec{c_j} \,] \;\; \geq \;\; 0 \,, \qquad \forall \, i, j \tag{7.25}$$

The half-space constraint, then, is

$$\boxed{\vec{d} \cdot [\, (\vec{n}_i)_{k,tr} + (\vec{n}_i)_{k,rot} \times \vec{c_j} \,] \geq 0 \,, \qquad \forall \, i, j} \tag{7.26}$$

Hence, the equation defining the IFS cone in the unglued case depends upon $(\vec{n}_i)_k$, an $\Re^6$ projection of the $\Re^{6n}$ normal vector $\vec{n}_i$ of the WSCH. This vector is computable, but would require the construction of the $6n$-dimensional convex hull. Since the complexity of a convex hull is exponential in the number of dimensions,

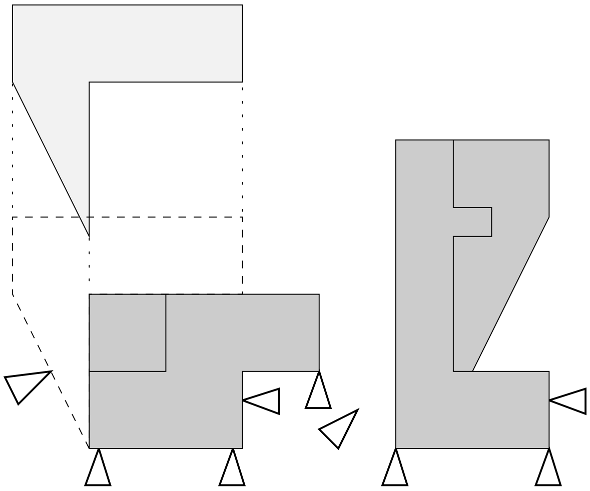


Figure 7.15: When an assembly is already disconnected, the IFS cone is determined by the fixels of only one component.

the algorithm's time and space requirements would be exponential in the number of parts n. Moreover, no projections or other methods are apparent which might produce $(\vec{n}_i)_k$ without this expensive convex hull.

This exponential cost unfortunately makes the construction of the IFS cone inadvisable for the unglued case.

7.5.3 Disconnected Assemblies

Assemblies which merely contain disconnected components (but are otherwise glued) are another matter. If an assembly consists of several components, and a part-removal operation affects only one of these components (see Fig. 7.15), then clearly, the IFS cone can be computed as before, but based only on the fixels for that component.

If, however, the part removal is the very operation which breaks an assembly into multiple components (see Fig. 7.16), then the fixels touching every such component affect the IFS cone. In the assembly sequence, each component, individually, must be stable against that part's insertion forces; therefore, the overall IFS cone is the intersection of the separate IFS cones arising from each component's fixels and contacts.

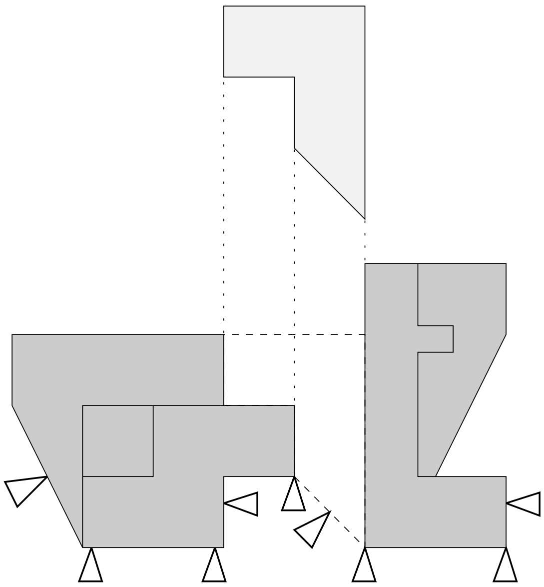


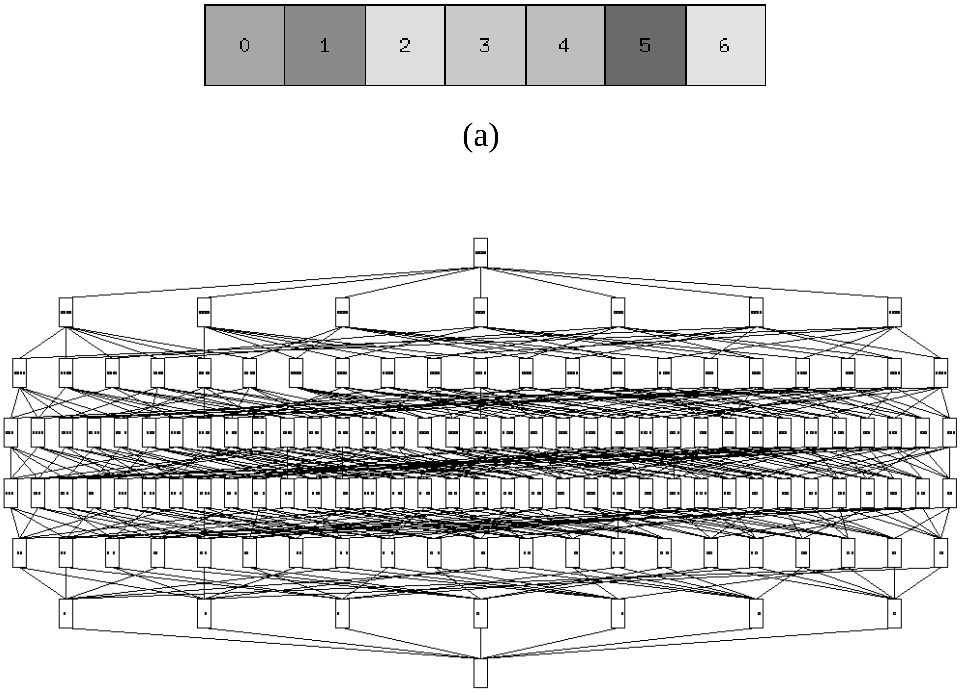
Figure 7.16: When a part removal is the action which breaks an assembly up into disconnected components, the IFS cone is determined by all affected components.

7.6 Computational Complexity

When describing the computational complexity of an assembly-planning algorithm, one must be careful to define precisely what procedure or task is being rated. In general, the number of producible subassemblies for a product may be exponential in the number of parts, so building the full disassembly graph is an exponential-time task (see, for example, Fig. 7.17). A consequence of this is that a thorough sequencing algorithm which tries all possible sequences before acknowledging failure may also be exponential.

Atlas 3 is such a system; and even when it ultimately does find a valid sequence/fixture solution, it will have exhaustively explored the disassembly graphs for the rejected candidate fixtures, at potentially exponential cost. This thoroughness, however, is a reasonable policy, unless some means can be developed to know with certainty that no stable sequence exists for a fixture, without exploring all subassemblies.

Perhaps a more useful benchmark, then, is the computational complexity of stably



(a)

(b)

Figure 7.17: The number of producible subassemblies for a product can be exponential in the number of parts, so even a small assembly can have a relatively large disassembly graph.

partitioning just one subassembly. And in this measure, Atlas 3 has polynomial complexity in the number of vertices in the assembly. To verify this, consider each of the main steps of procedure *FIND_ACTIONS_CTS* (p. 119) and its subprocedure *FIND_FPAs* (p. 107):

1. *Gravitational stability check.*

   As in Chapter 6, the check for gravitational stability is performed by using a linear program to determine the existence of a solution to eqn. (6.2) (p. 79). The number of variables in this linear program is the number of active fixels, which is proportional to the number of vertices v_e on the exterior of the subassembly.[3] Now, Atlas uses the Simplex linear-programming method [18], which has linear running time in practice, although in contrived cases it has been shown to be exponential [42]. However, other methods exist to solve linear programs in worst-case polynomial time, such as the ellipsoid method of Khachiyan [41] and the interior shortcut method of Karmarkar [37]. Galperin [22] has even proposed a linear-time algorithm for "asymmetric" linear programs, of which eqn. (6.2) is an example. The implementation of these algorithms would therefore ensure worst-case linear or polynomial time for the gravitational stability check, with respect to the number of exterior vertices v_e.

2. *Compute WSCH*

   In general, a convex hull of m points in d dimensions has complexity $O(m^{\lfloor d/2 \rfloor})$ and can be constructed in time $O(m^{\lfloor d/2 \rfloor + 1})$ [63]. For the WSCH, the number of points m is equal to the number of active fixels[4], which, again, is proportional to the number of vertices v_e on the assembly's exterior. The number of dimensions d is nominally 6; however, all that is really needed by the algorithm is the set of facets containing the origin. A "smart" algorithm may be able to take advantage of the origin requirement to drop the dimensionality for the construction to

[3]Enhanced overfixturing has not been implemented in Atlas 3. In a system using enhanced overfixturing, the distribution of the extra virtual fixels would also affect the complexity.

[4]Actually, it is the number of active fixels plus one, because the origin must also be included in the convex hull.

5. Certainly the number of normal vectors $\vec{n}_i$ (for the subsequent IFS cone construction) will be based on $d = 5$.

It suffices to note that d is a constant with respect to the complexity of the assembly; therefore the WSCH can be constructed in polynomial time with respect to v_e, and will have a polynomial number of normal vectors with respect to v_e.

3. *Compute IFS cone.*

As noted earlier, the number of half-space constraints in the IFS cone is equal to the number of WSCH normal vectors multiplied by the number of contact points between the part and the subassembly. The former is, again, polynomial in v_e. The latter, when summed over all n parts to be analyzed, is polynomial in the total number of vertices v in the assembly (the exact relationship to v depends on how the discrete contact points are defined). Since $v_e \leq v$, and the mutual intersection operations also do not violate the polynomial bound, it is safe to say that the construction of the IFS cones has no more than polynomial cost in the number of vertices v.

4. *Compute ETF cone.*

Wilson [78] showed that the extended NDBG arrangement can be computed in worst-case time $O(v^4)$. Since this construction is essentially the superimposition of all ETF cones in a single arrangement, the cost of a single ETF cone will naturally be even smaller. (Wilson's algorithm, incidentally, breaks each face into triangles and overlays all the triangle-on-triangle projections, creating a denser arrangement than theoretically necessary, but one which is far easier to compute than the projected convolutions of non-triangulated faces.)

5. *Project the fixels onto the motion sphere, partitioning that sphere.*

The projection of a point-sized fixel onto a motion sphere is equivalent to an ETF cone with a point-sized part. As with the regular ETF cones, an arrangement of such projections can be computed in polynomial time with respect to v.

Assembly	Number of Parts	Running Time	Number of Cand. Fixtures	Atlas 2 Time
Tilted Blender	7	0.65	7	0.05
Box-Lid-Cargo	5	0.12	4	0.02
A' Discrim.	11	0.93	1	0.16
Fan	8	0.30	3	0.03

Table 7.1: Experimental results for Atlas 3 on various assemblies, with Atlas 2 running times for comparison. (Times are in CPU seconds on a DEC Alpha workstation, excluding the time to find contacts.)

These assessments are summarized in the following result.

Result 7.5 *The procedure* FIND_ACTIONS_CTS, *in Fig. 7.14, can be executed in polynomial time with respect to the number of vertices in the assembly.*

7.7 Experimental Results

Atlas 3 implements the algorithm of Fig. 7.14 for planar assemblies, without enhanced overfixturing. Atlas 3, like its predecessor Atlas 2, is written in C and runs on a DEC Alpha workstation. The results for several example assemblies are summarized in Table 7.1, and discussed at greater length below.

The Tilted Blender

Fig. 7.18a shows the “tilted blender” example in the final fixture generated by Atlas 3; this solution is identical to that found by Atlas 2 (see Fig. 6.15, p. 93). As before, the corresponding sequence inserts the rightmost part straight downward, and then all other parts from left to right. This solution was generated in 0.65 seconds of CPU time.

Fig. 7.18b shows the candidate fixtures which were considered and rejected during the course of the analysis. Fixture 0 is the initial fixture. Two fixture-reducing actions were found for this configuration, leading to fixtures 1 and 2. From fixture 2, additional reductions led to fixture 3 and then 4, but fixture 4 is a dead end due to

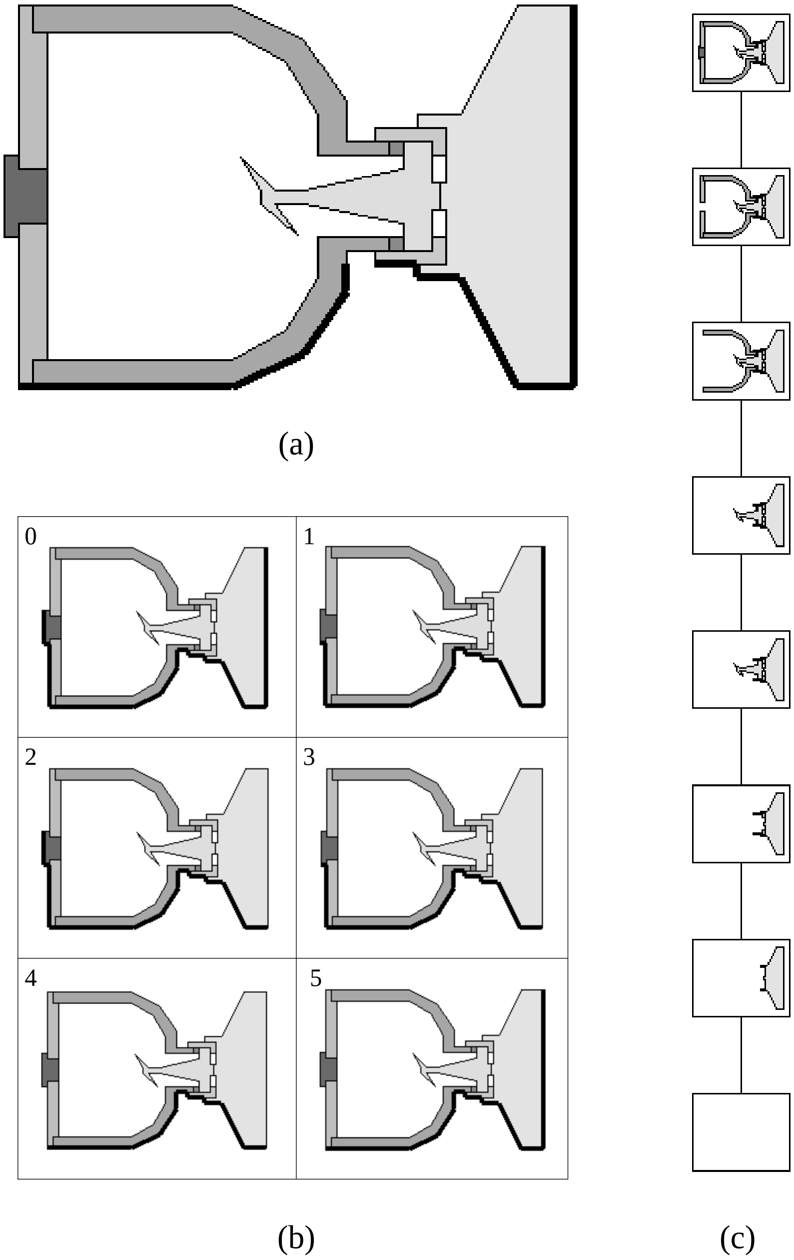


Figure 7.18: (a) The tilted-blender example, in the final fixture found by Atlas 3; (b) The six other candidate fixtures considered and rejected during the analysis; (c) The disassembly graph corresponding to the final fixture.

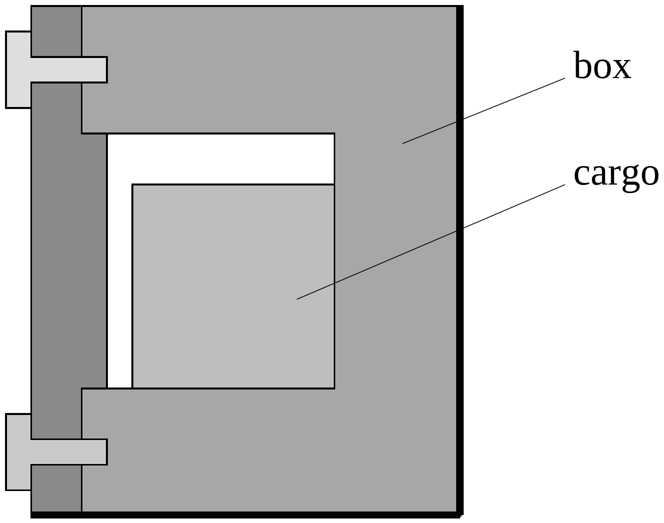


Figure 7.19: The "box-lid-cargo" example, in the final fixture found by Atlas 3. The corresponding sequence inserts the box diagonally down and to the right, then the cargo in a similar fashion, and then the remaining parts horizontally from left to right.

IFS requirements. From fixture 1, fixture 5 was found, and then finally the solution fixture.

Fig. 7.18c shows the disassembly graph corresponding to the solution fixture. Its simple, straight-line structure indicates that the first sequence explored by Atlas in this fixture led to a successful disassembly. (Incidentally, this graph happens also to be the complete disassembly graph for this fixture; the fixture only admits one feasible disassembly action for each subassembly.)

The Box-Lid-Cargo example

The "box-lid-cargo" example of Fig. 7.19 was solved in only 0.12 seconds, because of its relative simplicity. The fixture and the order of part insertions are the same as those found by Atlas 2.

The insertion *directions*, however, are slightly different. The "box" is inserted diagonally down and to the right rather than straight down, and the "cargo" is inserted diagonally rather than horizontally. These differences reflects Atlas 3's policy

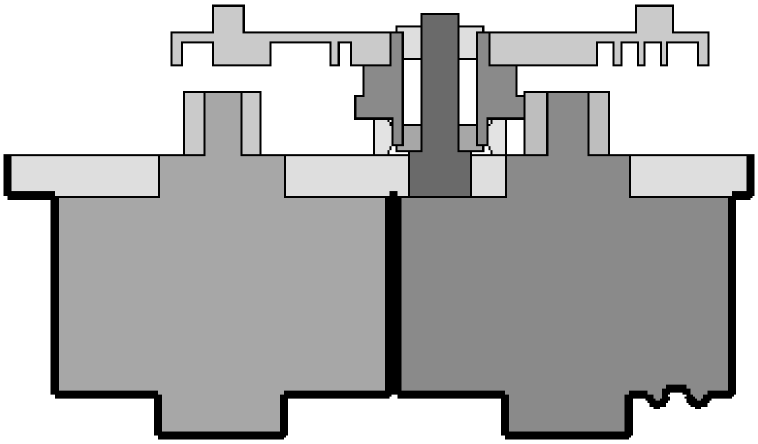


Figure 7.20: The A′ Discriminator example, in the final fixture found by Atlas 3. The corresponding sequence inserts all parts straight downward, except for one small part inserted at a slight angle.

of choosing discrete directions in the middle of a permissible range of directions, rather than at the limits of that range. This policy makes the sequences more robust to real-world variations in the insertion direction. (One enhancement to Atlas might be to ensure the non-degeneracy of these ranges, as an option; this would guarantee robustness at every step.)

The A′ Discriminator

The A′ Discriminator example (Fig. 7.20) was solved in 0.93 seconds. Almost the same solution was found by Atlas 3 as Atlas 2: the insertion of all parts basically straight down into the given fixture. (One small part is inserted at a slight angle, however, for the reason mentioned above.) In this case, the initial fixture turned out also to be the final one.

The Fan

The fan example of Fig. 7.21 was solved in 0.30 seconds. Again, Atlas 3 found almost the identical solution as Atlas 2: to insert the four thick parts straight down into the fixture shown in the diagram; then to insert the back of the cage, the fan blades, and

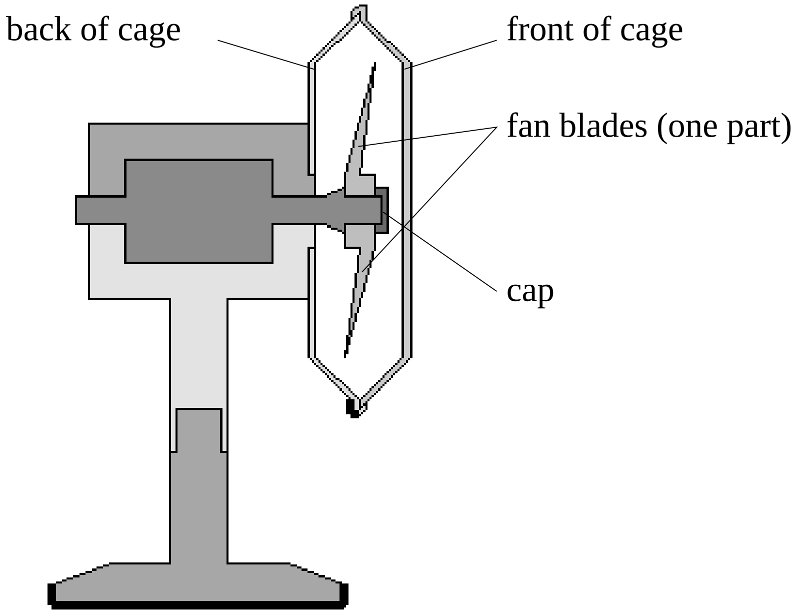


Figure 7.21: The fan example, in the final fixture found by Atlas 3. The corresponding sequence inserts the four thick parts straight downward; then the back half of the cage, the fan blades, and the cap from right to left; then the front half of the cage diagonally down and to the left.

the cap from right to left; then to insert the front of the cage diagonally down and to the left. This solution differs from that of Atlas 2 only by a slight change in the direction of the last insertion.

One Additional Example

One more very simple example is worth mentioning, that of Fig. 7.22. This is an assembly which Atlas 3 was able to solve, but Atlas 2 was not.

The stipulation that the final product be removed straight up eliminates fixture segments from three sides of the assembly. With one remaining segment along the bottom, the only valid insertion/removal direction for part 1 is vertically, due to insertion-force stability constraints. Atlas 3 is perfectly aware of this, by virtue of the SETF cone. However, Atlas 2 examines the part's V-shaped LTF cone and unsuccessfully attempts to remove the part in the two directions bounding that cone. Atlas 2 has no particular reason to consider an upward part removal, and therefore

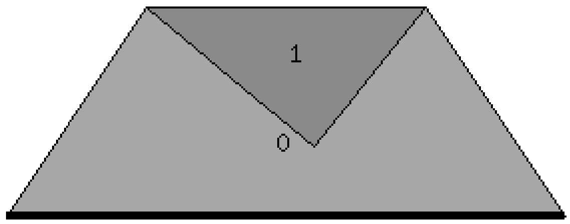


Figure 7.22: An example of an assembly solvable by Atlas 3 but not by Atlas 2. The only stable insertion direction for part 1 is straight down.

has no way of knowing that part 1 is removable. Consequently, Atlas 2 fails on this assembly.

Now, it may be argued that an upward removal/downward insertion is a logical direction to try anyway, for ergonomic reasons. It may also be argued that Atlas 2 can handle this assembly, as long as it uses a different final-removal direction. More complex examples exist, however, which demonstrate Atlas 2's shortcomings using other removal directions, and without any final-removal requirement at all. The basic fallibility of Atlas 2, though, is clear from Fig. 7.22.

7.8 Remarks

Thus, while Atlas 2 is faster than Atlas 3 (as demonstrated in Table 7.1), Atlas 3 is "safer", in that the guarantees presented earlier make Atlas 3 less likely to report failure when a sequence/fixture solution does exist.

Atlas 3 is still remarkably fast, given the complexity of the co-design task. That a system can perform co-design so quickly and still carry certain guarantees of success would seem to bode well for the future of sequence/fixture co-design systems.

Moreover, Atlas 3 achieves the running times shown in spite of the fact that its depth-first search of new candidate fixtures is in an arbitrary order. For example, in Fig. 7.18b, if Atlas had chosen to explore fixture 1 before fixture 2, it would have arrived at the final solution more quickly. It is conceivable, then, that a heuristic might be devised to guide this now-arbitrary search process, and further improve the running times.

Chapter 8

Conclusion

This dissertation has presented a new approach in automatic assembly planning, namely, the generation of assembly sequences and fixtures concurrently rather than sequentially. The joint treatment of these problems by a single planner is called for by the strong interdependence between the two tasks.

Chapter 3 presented a framework for this research by describing STAAT, a system for solving the sequencing problem alone. STAAT is an implementation of the NDBG-based techniques of Randy Wilson [78, 81] in a full-fledged sequencing system, supporting the evaluation of assembly complexity measures and searches for the optimal sequence according to a particular measure. It is, in effect, a "fulfillment" of NDBG technology in a very versatile system.

The fixturing problem was then introduced as a means of making the generated assembly plans more realistic, comprehensive, and useful. The assumptions, restrictions, and parameters of the sequencing and fixturing problems were discussed at length in Chapter 4, and the advantage of a unified co-design system was established.

Following an exploration of some key issues in co-design, the demand for a unified approach was met by the co-design methods of Chapters 6 and 7, and the Atlas systems which implement them in the planar case. The concept of a continuous fixture representation was introduced, as was the important concept of the wrench-space convex hull (WSCH). A fast co-design algorithm was presented in Chapter 6, in which the requirements of the evolving assembly sequence constrain the evolving fixture, and vice-versa. This algorithm indeed generates both an assembly sequence and a stabilizing fixture consistent with that sequence.

The successor algorithm, presented in Chapter 7, addressed some of the shortcomings of the previous system, by using the ETF cone and introducing a new structure, the insertion-force stability cone. These tools together allowed all stable, fixture-preserving actions for an assembly to be identified; the procedure to do this became the basis for a co-design method whose success is guaranteed under certain conditions. This technique is primarily useful when parts are glued together upon insertion. Although the complete approach is apparently somewhat slower than the first one, the planar implementations of both methods still generate sequence/fixture solutions very quickly. Their speed is consistent with the ultimate goal of an interactive CAD tool to evaluate product designs.

8.1 To the Next Step and Beyond

Clearly, in order to realize that goal, much work remains to be done. But even without regard to the specific application which motivated this research, some of the ideas and techniques considered here merit further exploration in their own right — as indeed, research for its own sake often opens the door to applications previously unimagined. The remainder of this chapter, therefore, is devoted to providing insight into some of the avenues along which the research presented herein might be extended, and the issues and potential difficulties that might arise in such an endeavor.

8.1.1 Future Work: An Overview

One important subgoal in the development of an interactive CAD tool to analyze assemblies will be a system which can return not just any sequence/fixture solution, but the "best" one according to some criterion. This objective again raises the question (discussed in Section 5.2) of what makes one assembly plan "better" than another. A system need not presume to answer this question definitively in order to be useful, but may instead provide multiple alternative figures of merit (as in STAAT), which a user may select or even combine. After a figure of merit is established, heuristics would then be needed to identify the partial assembly plans which are most likely to lead to optimal solutions.

Another key direction of future research will be to release many of the assumptions made earlier, and thereby broaden the domain of application of the co-design techniques. These assumptions include:

- rigid parts;
- single-step, translational part motions;
- polyhedral assemblies;
- frictionless surfaces;
- nominal geometry (no tolerances).

Many of the issues that would arise with the relaxation of these assumptions have already been discussed in Chapter 4. However, a possible extension from polyhedral to curved surfaces will be considered further in Section 8.1.2, with the new perspective of wrench space.

Based on my discussions with engineers in industry, the assumption which stands out as particularly "weakening" to the usefulness of Atlas in an industrial setting is that of non-toleranced assemblies. The deviation of parts from nominal geometries is a paramount consideration in the design of assemblies, assembly sequences, and assembly fixtures. Therefore, from the standpoint of system usefulness, the effective handling of toleranced assemblies in the co-design analysis is a very important avenue for further research. An analysis inclusive of tolerances would, moreover, enable the implementation of new practical figures of merit for assembly plans, as well as the proper handling of other related real-world considerations.

In addition to expanding the domain of applicability of co-design techniques, another direction for further research will be to expand the scope of the task being solved. As noted earlier, the problem being addressed in this dissertation is merely one piece of a much larger problem, in several respects. Directions for possible expansion include:

- automatically choosing other final-removal directions besides straight up;
- automatically choosing the optimal orientation for the assembly;
- selecting and placing finite-sized fixels (not just the locations of their tips);
- generating multiple-fixture solutions, possibly with nonlinear assembly operations;
- stabilizing assemblies against a broader range of possible insertion forces;
- incorporating other aspects of assembly planning besides sequencing and fixturing (such as tooling, for example).

The research presented herein, while promising in its own field, would be well-complemented by these additional analyses.

Finally, a full three-dimensional version of Atlas would be useful, as would experiments with more example assemblies (especially realistic ones from industry). These practical advances would help to verify and/or refine the co-design algorithms presented earlier.

8.1.2 A Closer Look

Let us now examine several of these directions for further research in greater detail.

Dynamic Addition of Fixels

One possible way in which the single-fixture assumption might be relaxed without introducing the complications of nonlinear actions (discussed in Section 4.2.2) would be to allow fixels to be added during the execution of the assembly plan. These fixels might, for example, represent adjustable clamps.

The reader will recall that in the Atlas 2 system of Chapter 6, it was necessary, upon each change in the proposed fixture, to go back and re-verify the stability of all previously-seen superassemblies. This was because an assembly which was stable under a fixture F_1 might not be stable under $F_2 \subset F_1$. However, what if this re-verification were simply suppressed? The assembling agent could use F_2 for the smaller subassembly, and then just add fixels to create F_1 for the larger subassembly. Thus, not only is the dynamic addition of fixels during the assembly process supported by the structure of the Atlas 2 algorithm, but it is actually easier and faster to plan than a single-fixture solution.

Modifying Atlas 3 might be a bit trickier, since its search-space representation (consisting of single-fixture disassembly graphs) does not lend itself to this adaptation as ideally as the representation of Atlas 2. Still, it might be possible to support dynamically-added fixels by spinning off new single-fixture graphs which use a subassembly, rather than the top-level assembly, as the root node. The completeness results of Chapter 7, though, would clearly have to be re-examined in this new context.

In the real world, not all fixels are alike, and dynamic addition does imply a certain type of fixel. This extension would therefore require additional information about what types of fixels are available, and what constraints, if any, exist on their placement. It would also increase both the cost of the fixture and the cost of executing the assembly plan, and these costs would have to be weighed against the feasibility and cost of insisting on a single, unchanging fixture.

Curved Surfaces

As noted earlier, the ability to handle curved surfaces is another important feature which a practical CAD tool should have in order to be useful and acceptable to industry.

However, fixturing a curved surface effectively — or even fixturing a flat surface with a curved boundary — is not as straightforward as it might seem. Consider, for example, a cylinder resting on an array of discrete fixture elements, as shown in Fig. 8.1. No matter how many fixels are used, it will never be possible to counteract

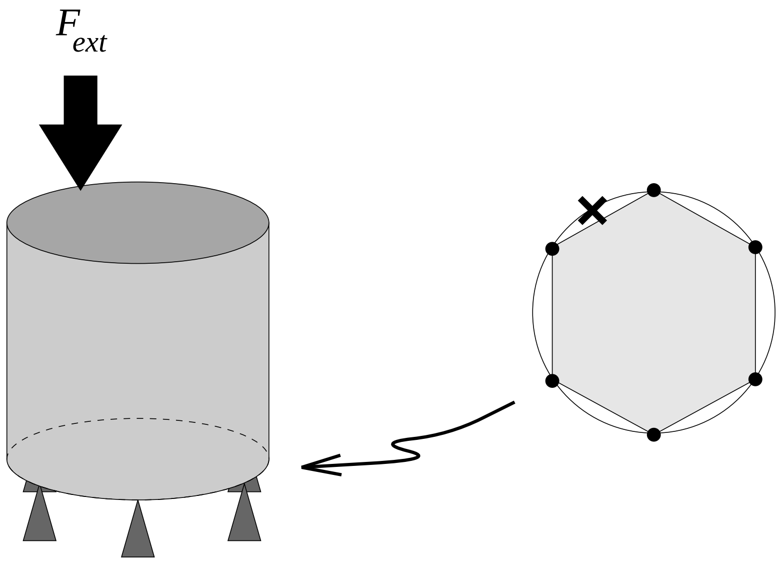
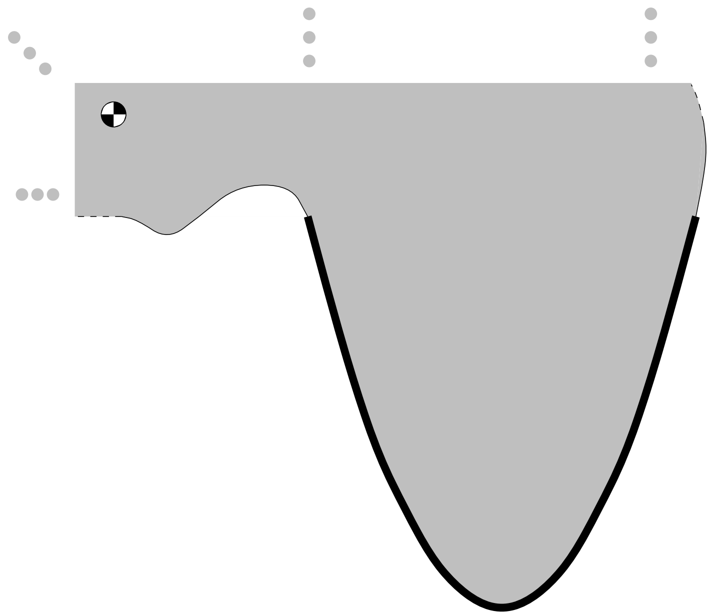


Figure 8.1: No matter how many discrete fixels are placed on the flat end of a cylinder, they can never completely counteract every possible downward force.

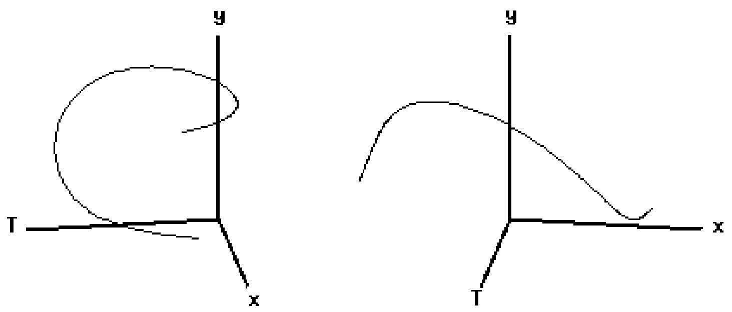
every possible downward force. This is important, because if another cylinder is placed on top of the first one, then downward insertion forces could indeed appear anywhere along the top rim. Similarly, if the fixture supporting the cylinder is continuous (and thus *can* counteract all downward forces), there is no way to represent this fixture with a series of virtual fixels which can be analyzed, and which accurately model its fixturing capability. (A related problem to this is how to represent the possible insertion-force wrenches in such a situation.)

This discretization problem exists even in planar assemblies. Consider the example of Fig. 8.2a, which shows a part fixtured on its curved bottom side using a continuous fixture representation. The set of fixel wrenches generated along this boundary (for unit fixel forces) is plotted in Fig. 8.2b. The set of all realizable wrenches from combinations of fixels would therefore be the open-ended convex hull (*i.e.,* the WSCH) of these points. As illustrated in Fig. 8.3, no finite collection of discrete virtual fixels will be able to duplicate this fixturing capability precisely.

However, it is possible to come close, by placing virtual fixels at well-chosen points along the curved edge. This approach, in effect, "samples" the locus of fixel wrenches.



(a)



(b)

Figure 8.2: (a) A planar part whose curved bottom side is partially fixtured. (b) Two views of the set of possible fixel wrenches generated (for unit fixel forces). No finite collection of virtual fixels will be able to duplicate this fixturing capability precisely.

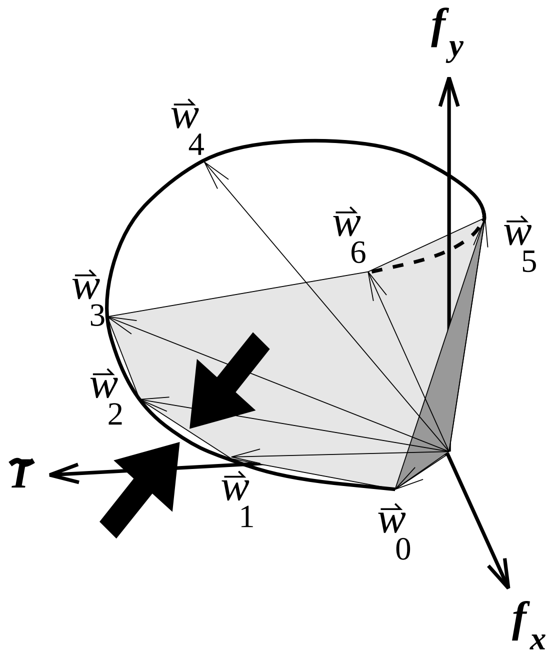


Figure 8.3: For many curved parts, no finite collection of virtual fixels can precisely duplicate the fixturing capability of a continuous fixture.

Furthermore, the error here would be an underestimate of the continuum's fixturing capability, a trend consistent with the principle of conservatism in engineering. A clever sampling algorithm, for example, could place virtual fixels densely in regions corresponding to wrenches on the boundary of the WSCH, and none at all in regions in the interior of the WSCH. It might even be possible to place virtual fixels *adaptively* during the course of the analysis, in response to specific external wrenches.

Nonlinear Actions

The simplest algorithms to perform disassembly sequencing with possibly nonlinear steps are the so-called "generate-and-test" methods [34, 44, 56], which enumerate possible subassemblies, either exhaustively or heuristically, and then try to remove each subassembly. Unfortunately, the number of possible subassemblies is exponential in the number of parts, which makes this approach inadvisable. One of the primary challenges of allowing nonlinear actions is to devise an alternative algorithm which can run in polynomial time and still be complete.

For the sequencing problem alone, this task has been accomplished by means of the extended NDBG, which collects all of the blocking information between pairs of parts and condenses it into a structure which can be processed in polynomial time. For the co-design problem, it would be useful to devise a similar structure, perhaps an adaptation of the NDBG, which includes gravitational and/or insertion-force stability constraints as well.

To handle insertion-force stability, one possibility might be to create some combination of IFS cones, showing the constraints on the removal directions of subassemblies rather than parts. Now, the shape of an IFS cone depends upon the contacts which are broken by the removal, and the fixels which remain to stabilize the subassembly; both of these quantities depend, in turn, on how the assembly is partitioned. Perhaps, then, it might be possible to exploit this dependence in order to identify constraints on both the partitioning and the removal direction.

8.2 Closing Remarks

Just as STAAT's analysis is deeper in sophistication than that of modern CAD tools, yet is narrower in applicability, so are the algorithms behind Atlas deeper yet more focused than those of STAAT. It is, of course, not entirely unexpected that research which addresses new and more difficult problems should initially have a more focused domain than the work which preceded it. It will be interesting to see how the technology presented herein evolves and expands, for clearly, the ideas, methods and systems presented here are not the "final word" in co-design, but rather only the beginning.

Glossary of Key Terms

action. In Chapters 6 and 7, a proposed expansion of the assembly plan, of the form "Delete segments S from the proposed fixture, and remove part P in direction D." *(Section 6.3).*

active fixel. A fixel which is still in contact with the remaining subassembly, and therefore can assist in stabilizing it. *(Section 6.9.2).*

candidate fixture. A fixture representing a partial solution. For example, a candidate fixture may successfully stabilize only the subsequence explored so far. *(Section 5.1).*

co-design. The technique of designing a sequence and a fixture concurrently, rather than first one and then the other.

complexity measure. A figure of merit for an assembly, an assembly sequence, and/or a fixture, which provides some indication of the difficulty or cost of the assembly process. *(Sections 1.3, 5.2).*

continuous. A *continuous* fixture representation is one in which the possible fixturing locations are represented as an infinite but bounded set of points, rather than as a discrete set. (It does not imply continuity in the sense of contiguousness.) *(Section 6.2).*

DBG. (Directional Blocking Graph.) A directed graph showing what parts are blocking what other parts, for motion in a given direction. *(Section 3.5.2).*

disassembly graph. A graph representing the ways in which a product can be disassembled monotonically. For sequences which may be nonlinear (as in Chapter 3), this is an AND/OR graph [35]; for linear sequences (Chapters 4–7), it is a simple OR graph. *(Sections 3.1, 5.1).*

enhanced overfixturing. The practice of overfixturing an assembly with a continuous representation, but with superfluous virtual fixels placed all over the assembly's exterior, so as to increase the applicability of the guarantee in Result 7.4 (p. 118). *(Section 7.5.1).*

ETF cone. (Extended Translational Freedom cone.) The subset of directions, from the entire sphere of directions, over which a part can translate to infinity without colliding with another part [66]. *(Section 7.1).*

feasible. A *feasible* action is one which involves no part-part collisions and whose fixture modification violates no stability requirements. *(Section 6.6).*

fixel. A discrete fixture element, *i.e.,* one of a series of mechanical "fingers" which collectively make up the fixture. *(Section 4.1.1).*

fixture. The structure which holds the product in place during the assembly process. Typically it consists of a number of discrete fixture elements, or *fixels*, which act as mechanical "fingers" to hold the assembly. In this dissertation, fixtures are assumed to be passive, exerting no force other than that necessary to oppose an external force. *(Section 1.3).*

fixture-preserving action. An action which does not involve the deletion of any portion of the fixture. *(Section 6.6).*

fixture-reducing action. An action which does involve the deletion of some portion(s) of the fixture. *(Section 6.6).*

fixturing. The problem of designing a fixture for a product. *(Section 1.3).*

force closure. The condition of a part or assembly being completely immobilized by a frictional grasp applying external forces. *(Section 2.2).*

form closure. The condition of a part or assembly being completely immobilized by a fixture based on the fixture's geometry alone, without including the effects of friction. *(Section 2.2).*

FPA. See *fixture-preserving action.*

FRA. See *fixture-reducing action.*

glued parts. The assumption that parts are glued, welded, screwed, or similarly fastened together upon insertion. *(Section 4.2.1).*

IFS. See *insertion-force stability.*

IFS caching. The technique of storing IFS results in order to answer the same or similar IFS queries later, without the need for a full linear program. *(Section 6.7.2).*

IFS cone. See *insertion-force stability cone.*

inactive fixel. A fixel which is no longer in contact with the subassembly, because the part it fixtured has been removed. Such a fixel cannot, therefore, assist in stabilizing the remaining subassemblies. *(Section 6.9.2).*

initial fixture. The fixture which results after the initial steps of overfixturing the assembly and clearing away a final removal path. *(Section 7.4).*

insertion. The addition of one or more parts to a subassembly to make a larger subassembly. Note that this includes, for example, setting one block on top of another, which might not normally be thought of as an "insertion".

insertion force. A force which a moving part may exert on a stationary part during an insertion, due to, for example, friction or a slight misalignment in the parts' positions. *(Section 4.1.1).*

insertion-force stability. The condition of being stable against any possible insertion forces. *(Section 4.1.1).*

insertion-force stability cone. The set of part-removal directions consistent with insertion-force stability requirements. It is the mutual intersection of half-space IFS constraints, of the form of eqn. (7.9) or, in the two-dimensional case, eqn. (7.17) (assuming parts are glued). *(Section 7.2).*

linear. A *linear* assembly sequence is one in which each step involves the insertion of a single part into the rest of the assembly. An assembly for which a linear assembly sequence exists is called a *linear assembly. (Section 3.1).*

LTF cone. (Local Translational Freedom cone.) The subset of directions, from the entire sphere of directions, over which a given part can translate an infinitesimal distance without colliding [33]. In some instances, as in Chapter 3, the LTF cone refers to collisions with another given part; in other instances, as in Chapter 6, it refers to collisions with the set of all other parts, collectively. *(Section 3.4).*

monotone. A *monotone* assembly sequence is one in which each subassembly, once constructed, is final; that is, its parts do not move with respect to one another in subsequent operations. *(Section 3.1).*

motion circle/sphere. The set of all possible directions of translation of a part or subassembly. It is a circle for planar assemblies and a sphere for three-dimensional assemblies. *(Section 3.4).*

NDBG. (Non-Directional Blocking Graph.) A data structure in which the space of all possible disassembly motions is partitioned into equivalence classes, such that for all motions within a class, the parts exhibit the same blocking relationship. For the case of single-step translations of three-dimensional parts, the NDBG partitions the motion sphere into regions, such that there is a constant blocking relationship within each region [78, 81]. *(Section 3.4).*

overfixture. To surround the assembly with a completely immobilizing fixture. In Chapters 6 and 7, Atlas performs sequence/fixture co-design by first overfixturing the assembly, and later deleting portions of the fixture as appropriate. *(Section 6.4).*

re-fixture. To remove a subassembly from one fixture and transfer it to another one. Sometimes the subassembly will be re-oriented at the same time. In general, re-fixturing is an expensive operation. *(Section 4.2.2).*

sequence. An ordered list of part or subassembly insertions, including their directions, which will assemble the product in a collision-free manner.

sequencing. The problem of finding the order and direction of part or subassembly insertions which will assemble the product in a collision-free manner. *(Section 1.3).*

SETF cone. (Stable Extended Translational Freedom cone.) The set of directions over which a given part can be removed without colliding with another part, nor causing insertion-force instability under the current fixture. It is equal to the intersection of the IFS cone and the ETF cone. *(Section 7.3).*

stack. A *stack* assembly sequence is one in which all the motions occur in a single direction, usually up-and-down. An assembly for which a stack assembly sequence exists is called a *stack assembly. (Section 3.1).*

subfixture. A fixture whose elements are a subset of those of a given fixture. *(Section 7.4).*

two-handed. A *two-handed* assembly sequence is one in which each step requires only two hands, *i.e.,* no step requires three or more subassemblies to have simultaneous relative motion. (A stationary subassembly, in this context, is considered to require a hand.) *(Section 3.1).*

virtual fixel. A hypothetical fixel which, together with others like it, can reproduce the possible wrenches producible by a continuous fixture representation. Reducing the continuous representation to an equivalent, finite set of virtual fixels is a convenient preprocessing step to analyzing stability. *(Section 6.2.1).*

wrench. A force and a torque together in the same vector:

$$\vec{w} = \begin{pmatrix} \vec{F} \\ \vec{\tau} \end{pmatrix}$$

When parts can move independently (*i.e.,* are not glued together), this notion is generalized to a series of forces and torques, acting on the different parts in the assembly:

$$\vec{w} = \left[\vec{F}_0^T \ \vec{\tau}_0^T \quad \vec{F}_1^T \ \vec{\tau}_1^T \quad \cdots \quad \vec{F}_{n-1}^T \ \vec{\tau}_{n-1}^T\right]^T$$

(Section 6.7.1).

WSCH. (Wrench-Space Convex Hull.) The open-ended convex hull of the fixel wrenches, bounded by planes passing through the origin. *(Section 7.2; illustration, Fig. 6.13).*

Bibliography

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*, pages 222–226. Addison-Wesley, Reading, MA, 1985.

[2] E.M. Arkin, R.C. Connelly, and J.S.B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. Fifth Annual Symposium on Computational Geometry*, pages 334–343, Saarbrucken, West Germany, 1989.

[3] H. Asada and A.B. By. Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. *IEEE Journal of Robotics and Automation*, RA-1(2):86–94, 1985.

[4] D.F. Baldwin, T.E. Abell, M.-C.M. Lui, T.L. De Fazio, and D.E. Whitney. An integrated computer aid for generating and evaluating assembly sequences for mechanical products. *IEEE Transactions on Robotics and Automation*, 7(1):78–94, February 1991.

[5] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[6] D. Baraff, R. Mattikalli, and P. Khosla. Minimal fixturing of frictionless assemblies: complexity and algorithms. Technical Report CMU-RI-TR-94-08, Carnegie Mellon University, Pittsburgh, April 1994. To appear in *Algorithmica*, 1997.

[7] J.J. Bausch and K. Youcef-Toumi. Kinematic methods for automated fixture reconfiguration planning. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1396–1401, Cincinnati, OH, 1990.

[8] G. Boothroyd and P. Dewhurst. *Design for Assembly.* Boothroyd and Dewhurst, Inc., Wakefield, RI, 1983.

[9] A. Bourjault. *Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires.* Ph.D. thesis, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, France, 1984.

[10] R.C. Brost and K.Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Transactions on Robotics and Automation*, 12(1):31–46, February 1996.

[11] R.C. Brost and M.T. Mason. Graphical analysis of planar rigid-body dynamics with multiple frictional contacts. In H. Miura and S. Arimoto, editors, *Robotics Research: The Fifth International Symposium*, pages 293–300. MIT Press, Cambridge, MA, 1990.

[12] R.C. Brost and R.R. Peters. Automatic design of 3-d fixtures and assembly pallets. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 495–502, Minneapolis, MN, 1996.

[13] S. Chakrabarty and J. Wolter. A hierarchical approach to assembly planning. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 258–263, San Diego, 1994.

[14] K.C. Chan, B. Benhabib, and M.Q. Dai. A reconfigurable fixturing system for robotic assembly. *Journal of Manufacturing Systems*, 9(3):206–221, 1990.

[15] H. Chang and T.-Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 1012–1019, Nagoya, Japan, 1995.

[16] Y.-C. Chou. A methodology for automatic layout of fixture elements based on machining forces considerations. In P.H. Cohen and S.B. Joshi, editors, *Advances*

in Integrated Product Design and Manufacturing, pages 181–187. ASME, New York, 1990. ASME Production Engineering Division, volume 47.

[17] R. Cottle. On a problem in linear inequalities. *Journal of the London Mathematical Society*, 43:378–384, 1968.

[18] G.B. Dantzig. *Linear programming and extensions.* Princeton University Press, Princeton, NJ, 1963.

[19] T.L. De Fazio and D.E. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Journal of Robotics and Automation*, RA-3(6):640–658, December 1987.

[20] S.E. Fahlman. A planning system for robot construction tasks. *Artificial Intelligence*, 5:1–49, 1974.

[21] C. Ferrari and J. Canny. Planning optimal grasps. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 2290–2295, Nice, France, 1992.

[22] E.A. Galperin. Linear time algorithms for linear programming. *Methods of Operations Research*, 64:3–12, 1991.

[23] K.Y. Goldberg and H. Moradi. Compiling assembly plans into hard automation. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1858–1863, Minneapolis, MN, 1996.

[24] P. Gu and X. Yan. CAD-directed automatic assembly sequence planning. *International Journal of Production Research*, 33(11):3069–3100, November 1995.

[25] L.J. Guibas, D. Halperin, H. Hirukawa, J.-C. Latombe, and R.H. Wilson. A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 2553–2560, Nagoya, Japan, 1995.

[26] D. Halperin and R.H. Wilson. Assembly partitioning with a constant number of translations. Technical Report SAND94-1819, Sandia National Laboratories, Albuquerque, NM, 1994.

[27] D. Halperin and R.H. Wilson. Assembly partitioning along simple paths: the case of multiple translations. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1585–1592, Nagoya, Japan, 1995.

[28] S.K. Hargrove and A. Kusiak. Computer-aided fixture design: a review. *International Journal of Production Research*, 32(4):733–753, 1994.

[29] F.B. Hazen and P.K. Wright. Workholding automation: innovations in analysis, design, and planning. *Manufacturing Review*, 3(4):224–237, December 1990.

[30] H. Hirukawa, T. Matsui, and K. Takase. A general algorithm for derivation and analysis of constraint for motion of polyhedra in contact. In *Proceedings IROS '91. IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. Intelligence for Mechanical Systems*, volume 1, pages 38–43, Osaka, Japan, 1991.

[31] R. Hoffman. Automated assembly planning for B-rep products. In *Proc. IEEE International Conference on Systems Engineering*, pages 391–394, Pittsburgh, 1990.

[32] R. Hoffman. A common sense approach to assembly sequence planning. In L.S. Homem de Mello and S. Lee, editors, *Computer-Aided Mechanical Assembly Planning*, pages 289–313. Kluwer Academic Publishers, Boston, 1991.

[33] L.S. Homem de Mello. *Task Sequence Planning for Robotic Assembly*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 1989.

[34] L.S. Homem de Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 56–61, Scottsdale, AZ, 1989.

[35] L.S. Homem de Mello and A.C. Sanderson. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, April 1990.

[36] Y.F. Huang and C.S.G. Lee. An automatic assembly planning system. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 1594–1599, Cincinnati, OH, 1990.

[37] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[38] S.G. Kaufman, R.H. Wilson, R.E. Jones, T.L. Calton, and A.L. Ames. The Archimedes 2 mechanical assembly planning system. In *Proc. IEEE International Conference on Robotics and Automation*, volume 4, pages 3361–3368, Minneapolis, MN, 1996.

[39] L. Kavraki, J.-C. Latombe, and R.H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229–235, 1993.

[40] L.E. Kavraki and M.N. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 55(3):159–165, 1995.

[41] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979. (Translated from *Doklady Akademii Nauk SSSR*, 224(5):1093–1096, 1979).

[42] V. Klee and G.J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

[43] C.M. Klein and H.-K. Rhee. An assembly sequence generation procedure. *Computers and Industrial Engineering*, 21:63–66, 1991.

[44] S.S. Krishnan and A.C. Sanderson. Path planning algorithms for assembly sequence planning. In *Proc. of the International Symposium on Intelligent Robotics*, pages 428–439, Bangalore, India, 1991. *Proc. of the SPIE* series, volume 1571.

[45] K. Lakshminarayana. Mechanics of form closure. ASME Paper 78-DET-32, 1978.

[46] L. Laperrière and H.A. ElMaraghy. Assembly sequences planning for simultaneous engineering applications. *International Journal of Advanced Manufacturing Technology*, 9(4):231–244, 1994.

[47] J.-C. Latombe, R.H. Wilson, and F. Cazals. Assembly sequencing with toleranced parts. *Computer-Aided Design*, 29(2):159–174, February 1997.

[48] S. Lee and Y.G. Shin. Assembly planning based on geometric reasoning. *Computers and Graphics*, 14(2):237–250, 1990.

[49] S. Lee and C. Yi. Subassembly stability and reorientation. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 521–526, Atlanta, 1993.

[50] S.H. Lee and M.R. Cutkosky. Fixture planning with friction. *Journal of Engineering for Industry, Transactions of the ASME*, 113(3):320–327, August 1991.

[51] Z. Li and S.S. Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal of Robotics and Automation*, 4(1):32–44, 1988.

[52] A.C. Lin and T.-C. Chang. 3D MAPS: Three-dimensional mechanical assembly planning system. *Journal of Manufacturing Systems*, 12(6):437–456, 1993.

[53] T. Lozano-Pérez and R.H. Wilson. Assembly sequencing for arbitrary motions. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 527–532, Atlanta, 1993.

[54] T. Lu, B. Zhang, and P. Jia. Assembly sequence planning based on graph redution. In *Proc. TENCON '93: 1993 IEEE Region 10 Conference on "Computer, Communication, Control and Power Engineering"*, volume 4, pages 119–122, Beijing, October 1993.

[55] R. Mattikalli, P. Khosla, B. Repetto, and D. Baraff. Stability of assemblies. In *IROS '93. Proc. of the 1993 IEEE/RSJ International Conference on Intelligent*

Robots and Systems. Intelligent Robots for Flexibility., volume 1, pages 652–661, Yokohama, Japan, 1993.

[56] R.S. Mattikalli. *Mechanics Based Assembly Planning.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 1994.

[57] J.M. Miller and R.L. Hoffman. Automatic assembly planning with fasteners. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 69–74, Scottsdale, AZ, 1989.

[58] B. Mishra, J.T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(4):541–558, 1987.

[59] B. K. Natarajan. On planning assemblies. In *Proc. of the Fourth Annual Symposium on Computational Geometry*, pages 299–308, Urbana-Champaign, IL, 1988.

[60] V.-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, 1988.

[61] R.S. Palmer. *Computational complexity of motion and stability of polygons.* Ph.D. thesis, Cornell University, Ithaca, NY, 1987.

[62] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete and Computational Geometry*, 3(2):123–136, 1988.

[63] F.P. Preparata and M.I. Shamos. *Computational Geometry: an Introduction*, pages 98, 137–140. Springer-Verlag, New York, 1985.

[64] E. Rimon and J. Burdick. Mobility of bodies in contact. I: A new 2^{nd} order mobility index for multiple-finger grasps. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 2329–2335, San Diego, 1994.

[65] B. Romney, C. Godard, M. Goldwasser, and G. Ramkumar. An efficient system for geometric assembly sequence generation and evaluation. In *Proc. ASME International Computers in Engineering Conference*, pages 699–712, Boston, 1995.

[66] A. Schweikard and R.H. Wilson. Assembly sequences for polyhedra. *Algorithmica*, 13(6):539–552, June 1995.

[67] A. Senthil kumar, A.Y.C. Nee, and S. Prombanpong. Expert fixture-design system for an automated manufacturing environment. *Computer Aided Design*, 24(6):316–326, June 1992.

[68] C.K. Shin, D.S. Hong, and H.S. Cho. Disassemblability analysis for generating robotic assembly sequences. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1284–1289, Nagoya, Japan, 1995.

[69] B. Shirinzadeh. Issues in the design of the reconfigurable fixture modules for robotic assembly. *Journal of Manufacturing Systems*, 12(1):1–14, 1993.

[70] B. Shirinzadeh. CAD-based design and analysis system for reconfigurable fixtures in robotic assembly. *Computing and Control Engineering Journal*, 5(1):41–46, February 1994.

[71] A.K. Subramani and P. Dewhurst. Automatic generation of product disassembly sequences. *Manufacturing Technology CIRP Annals*, 40(1):115–118, 1991.

[72] A. Sudsang, J. Ponce, and N. Srinivasa. Algorithms for constructing immobilizing fixtures and grasps of three-dimensional objects. In Preprints of the Second Workshop on Algorithmic Foundations of Robotics (WAFR '96), edited by J.-P. Lammond and M. Overmars, 1996. Final version to be published by A.K. Peters, Wellesley, MA.

[73] J.C. Trappey and C.R. Liu. A literature survey of fixture-design automation. *International Journal of Advanced Manufacturing Technology*, 5(3):240–255, August 1990.

[74] J.C. Trinkle. On the stability and instantaneous velocity of grasped frictionless objects. *IEEE Transactions on Robotics and Automation*, 8(5):560–572, October 1992.

[75] J. Tsao and J. Wolter. Assembly planning with intermediate states. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 71–76, Atlanta, 1993.

[76] J.-M. Valade. Geometric reasoning and automatic synthesis of assembly trajectory. In *Proc. of '85 International Conference on Advanced Robotics*, pages 43–50, Tokyo, 1985. Japan Industrial Robot Association.

[77] K. Whybrew and B.K.A. Ngoi. Computer aided design of modular fixture assembly. *International Journal of Advanced Manufacturing Technology*, 7(5):267–276, 1992.

[78] R.H. Wilson. *On Geometric Assembly Planning*. Ph.D. thesis, Stanford University, Stanford, CA, 1992. Stanford Technical Report STAN-CS-92-1416.

[79] R.H. Wilson. Geometric reasoning about assembly tools. Technical Report SAND95-2423, Sandia National Laboratories, Albuquerque, NM, 1996.

[80] R.H. Wilson, L. Kavraki, T. Lozano-Pérez, and J.-C. Latombe. Two-handed assembly sequencing. *International Journal of Robotics Research*, 14(4):335–350, 1995.

[81] R.H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.

[82] J. Wolter and E. Kroll. Toward assembly sequence planning with flexible parts. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 528–534, Minneapolis, MN, 1996.

[83] J.D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly*. Ph.D. thesis, University of Michigan, 1988.

[84] J.D. Wolter. On the automatic generation of assembly plans. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 62–68, Scottsdale, AZ, 1989.

[85] J.D. Wolter and J.C. Trinkle. Automatic selection of fixture points for frictionless assemblies. In *Proc. IEEE International Conference on Robotics and Automation*, volume 1, pages 528–534, San Diego, 1994.

[86] T.C. Woo and D. Dutta. Automatic disassembly and total ordering in three dimensions. *Journal of Engineering for Industry*, 113(2):207–213, May 1991.

[87] Y.L. Xiong, D.J. Sanger, and D.R. Kerr. Geometric modelling of boundless grasps. *Robotica*, 11:19–26, 1993.