

## Lessons Learned from a Second Generation Assembly Planning System\*

Arlo L. Ames      Terri L. Calton      Rondall E. Jones      Stephen G. Kaufman  
Cathy A. Laguna      Randall H. Wilson

Intelligent Systems and Robotics Center  
Sandia National Laboratories  
Albuquerque, NM 87185-0951

### Abstract

*We briefly describe Archimedes 2, a second generation assembly planning system that both provides a general high-level assembly sequencing capability and, for a smaller class of products, facilitates automatic programming of a robotic workcell to assemble them. Because Archimedes can input CAD data in several standard formats, we have been able to test it on a number of industrial assembly models more complex than any before attempted by automated assembly planning systems. These experiments, and our interaction with industrial manufacturing engineers, have led us to a number of conclusions about the state of assembly planning research, and our own future directions in particular.*

### 1 Introduction

Modern electromechanical products have more parts, of diverse types, designed by geographically separated designers, assembled into less space and designed faster than ever before. Ensuring that these products can be assembled, serviced, and disassembled efficiently and reliably is thus presenting growing demands on designers and manufacturing engineers.

Computer-aided assembly planning techniques promise to address these problems, by providing engineers with software tools to automatically analyze assemblability, systematically explore alternative assembly schemes, and facilitate the design-to-manufacturing transfer. Responding to this need, the last decade has seen an explosion in computer-aided assembly planning research (see, e.g., [11]). Significant advances have been made in both theory and practice, and a great number of experimental systems have been built.

In this paper we present Archimedes 2, an assembly planning system designed to allow preliminary applications of assembly planning in industry, while solidly supporting further research in planning techniques. Archimedes 2 can be considered "second generation" for several reasons [17]:

- It is a descendant of Archimedes, an earlier proof-of-concept assembly planner [31].

- It was consciously designed to take advantage of the lessons and techniques of earlier assembly planning research.
- It is written in C and C++ for efficiency and compatibility with industrial computer systems, and it takes input from industry-standard Computer-Aided Design (CAD) packages.

The system quickly and automatically generates geometrically-valid assembly sequences for a wide range of assemblies. For a more restricted class of products, it determines plans that optimize a given cost function, graphically illustrates those plans with simulated robots, and facilitates the generation of robotic programs to carry out those plans in a robotic workcell.

We have been able to test the system on a number of industrial assemblies, directly using CAD models provided to us by the companies that designed the products. We believe these assemblies are more realistic and more complex than any that previous assembly planners have been tested on, and give us new insights into some of the critical problems facing our work, and assembly planning research in general, in order to be of practical use.

In the next section, we give an overview of the components of the Archimedes 2 system. We use this overview to structure our discussion of prior art, describing for each component previous research from which we have borrowed, or that sheds light on our choices. We then present experimental results of the system applied to industrial assemblies. Finally, we conclude with a discussion of limitations specific to our approaches and system, as well as more basic problems that we believe all assembly planning systems will have to address in the future.

### 2 Archimedes 2

The Archimedes 2 system can be seen as a sequence of modules, each viewing the product at a greater level of detail and supplying more detailed assembly plans and designer feedback than the previous one. At the top is the design module, which captures and represents the geometric, mechanical, and other information about the product required for analysis. The design module only requires design consistency; it does not

---

\*This work was supported by Sandia National Laboratories under DOE contract DE-AC04-94AL85000.

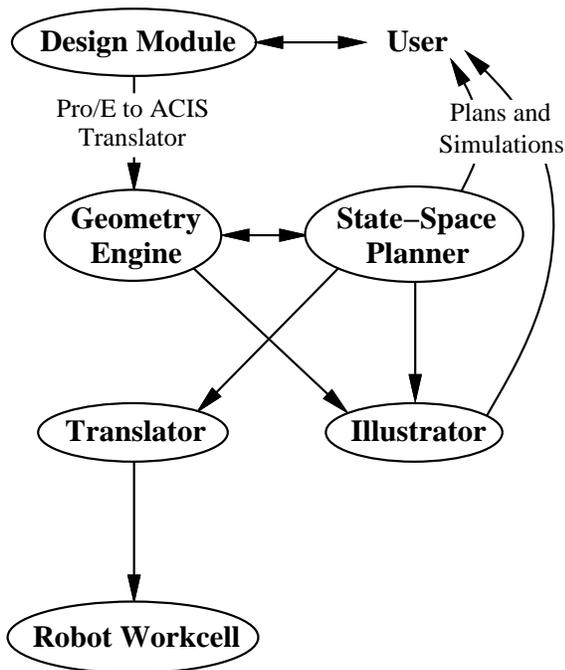


Figure 1: The architecture of Archimedes 2

apply any manufacturing constraints. At the bottom is the robotic workcell; all details must by definition be present in the assembly plans executed there. The architecture of the system is shown in figure 1.

Using industry-standard languages for portability, maintainability, and compatibility with industrial users was a primary focus in writing Archimedes 2. The design module is written in Pro/DEVELOP<sup>®</sup>, and integrated with the Pro/ENGINEER<sup>®</sup> CAD package, while the illustrator is implemented in CimStation using SIL. The rest of the modules are written in C++, using ACIS<sup>®</sup> as a solid modeling kernel and OpenGL for on-line animated output.

Due to space limitations, the various modules must be described at a high level. We refer the reader to more detailed descriptions (ours or others') of the underlying methods where possible.

## 2.1 Design module

Currently under development, the Archimedes design module will allow a user to model the geometric and other aspects of a product necessary for planning. In the past, the auxiliary (non-geometric) information has for the most part been available on final drawings, but not in a computer-accessible form. Built on Pro/ENGINEER<sup>®</sup>, the design module currently supports detailed weld and adhesive specifications, with part and assembly geometry handled by Pro/ENGINEER<sup>®</sup>. Other part attachment information (such as snapfits, interference fits, and threaded attachments) will soon be supported. The design mod-

ule can translate the geometric and auxiliary data into ACIS<sup>®</sup> for input into the downstream Archimedes planners. In some cases, data from other ACIS<sup>®</sup>-compatible CAD systems is used by entering the auxiliary data by hand.

The auxiliary information the design module captures is a standard feature of assembly planning systems. Bourjault's *liaisons* [3] and Ko and Lee's *matching conditions* [18] are early examples, while Homem de Mello's *relational model* [12] is probably the most comprehensive view. Work on CAD standards such as PDES/STEP [15, 22] promises to standardize representations for much of this information.

## 2.2 Geometry engine

The geometry engine accepts ACIS<sup>®</sup> assembly models and auxiliary data, and determines geometrically valid *part-level* assembly sequences. These sequences consider only the geometric blocking constraints between the parts and/or subassemblies to be mated at each step. The geometry engine finds part-part contacts automatically from the CAD data, then constructs a *non-directional blocking graph* of the assembly [35], to quickly identify important directions of motion and subassemblies. Single translations and twisting motions are used to mate parts.

The contact-finding routines employed are relatively straightforward and derived from [33]. The non-directional blocking graph can be seen as a geometrically-complete extension of Wolter's *assembly constraint graph* [36]. A graphics workstation's hardware Z-buffer is used to quickly find collisions between complex faceted models, a method similar to but much faster than ray-casting [10].

## 2.3 State-space planner

This module calls the geometry engine to find geometrically valid part motions, but applies additional constraints in its search for an optimal assembly plan. It only allows vertical assembly, adds subassembly reorientation operations when required, and enforces laser weld site accessibility. The planner performs an A\* search [29] over the space of linear assembly plans (i.e. each operation places a single part in a subassembly) to find the best plan according to a user-specified optimality criterion. The state-space planner also suggests simple assembly fixtures designed using boolean subtraction. The resulting assembly plan can be output to the illustrator or the workcell.

A\* and AO\* search techniques are standard in assembly planners (see for instance [20, 36]). The state-space planner currently implements two typical optimality criteria, minimizing the number of robot tool changes [13] and subassembly reorientations [36].

## 2.4 Illustrator

The illustrator simulates assembly plans in robotic workcells. It takes as input a description of a workcell and fixtures, the CAD model of the assembly, and an assembly plan. The output is a detailed simulation of the robot executing the plan. Educated guesses are made to fill in missing plan details such as gripping points on the parts and part feeding locations. The

resulting plan illustration allows workcell designers as well as assembly designers to visualize the assembly workcell in operation, and could support detailed analyses of cycle times, etc. We are unaware of previous work in this area.

## 2.5 Translator

Finally, the translator compiles assembly plans into V+ code to execute in the Archimedes robotic workcell. Required details such as fixture locations and part gripping offsets are retrieved from a workcell-specific database. Desired relative part locations and approach directions, as well as the sequence of mating operations, reorientations, and welding operations, are derived from the input plan. The workcell program uses a library of intelligent robotic routines to identify, localize, acquire, orient, and mate parts and subassemblies.

The translator uses skeleton strategies for assembly operations, much like those used in AUTOPASS [21] and LAMA [25]. More ambitious systems for generating robotic code automatically for assembly plans and operations include Handey [23] and SPAR [14], which for instance plan regrasping operations and explicitly reason about uncertainty.

## 2.6 Workcell

The Archimedes workcell is centered around an Adept 2 robot. Two robot-mounted cameras provide part recognition and localization. A tool changer allows use of multiple grippers, including suction and parallel-jaw types. A specialized flipping device is used to grasp and reorient parts and subassemblies. A tray for parts presentation and end-of-arm tooling are customized by hand to each product assembled; in addition, modifications to the automatically-designed fixtures are usually required.

## 3 Experiments

Speaking generally, the design module and geometry engine are capable of operating on a large class of electromechanical assemblies, and have been applied to a number of them obtained from various sources. The state-space planner and translator are targeted toward creating detailed linear assembly plans for a more limited set of unidirectional assemblies, and have only been applied to one example so far. The illustrator lies in between: it is capable of illustrating any plans created by the geometry engine or state-space planner, but its detail and realism are greater for unidirectional assemblies.

### 3.1 The pattern wheel

The pattern wheel, shown in figure 2, is a subassembly of a discriminator (see the next example, although a different design). The pattern wheel was a main example handled by the original Archimedes system [31], and as such was our first target. The pattern wheel has 13 parts, assembled unidirectionally, and is fastened together by laser welds.

The pattern wheel was modeled, with detailed welding specifications, in the design module, and translated to ACIS<sup>®</sup> format. The state-space planner then de-

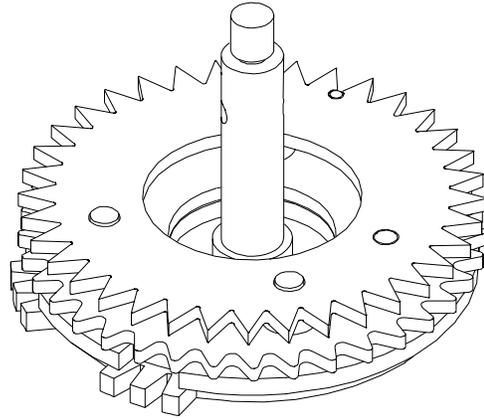


Figure 2: The pattern wheel assembly

termined an assembly plan for the pattern wheel in a total of one minute. Since assembly reorientations and robot tool changes are by far the slowest operations in the workcell, the optimality criterion given to the planner minimizes the number of these operations in the plan. The resulting plan was illustrated in a simulated workcell, and the plan was automatically translated to V+ code. The resulting robot program was executed in the workcell to assemble the pattern wheel. Laser welds were not performed automatically; instead, pre-welded subassemblies were substituted in the workcell when the program specified a laser weld.

To date, the pattern wheel is the only assembly which has exercised all modules of Archimedes 2.

### 3.2 The discriminator

A discriminator is a mechanical safety device designed to prevent accidental operation of a system. The A-PRIMED Project [7] at Sandia chose these devices to demonstrate processes and technologies for agile product design, development, and production. The discriminator in figure 3 has 42 parts described by approximately 12 Mb of Pro/ENGINEER<sup>®</sup> data. To date, only the design module, geometry engine, and illustrator have been applied to the full discriminator. Reading in the models, finding all the contacts between parts, and determining a single geometrically valid part-level assembly sequence for the discriminator takes the geometry engine thirty seconds. This plan uses two non-linear assembly operations (i.e. each operation merges two nontrivial subassemblies) and includes all welding specifications. The resulting sequence can be output directly to the illustrator, which animates it in a model of the A-PRIMED robotic workcell. We are currently attempting to apply the rest of the Archimedes modules to the discriminator.

### 3.3 The Rockwell assembly

This assembly is a circuit board in a case, a relatively simple assembly mechanically, that is currently in low-volume production at Rockwell International. During its assembly, however, a number of tight ac-

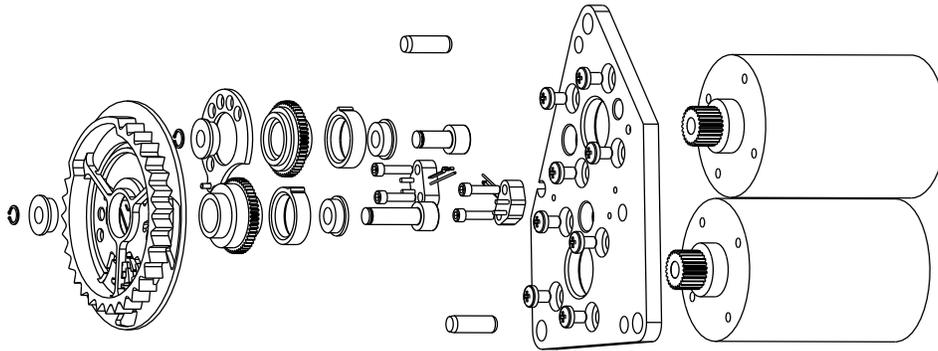


Figure 3: The discriminator

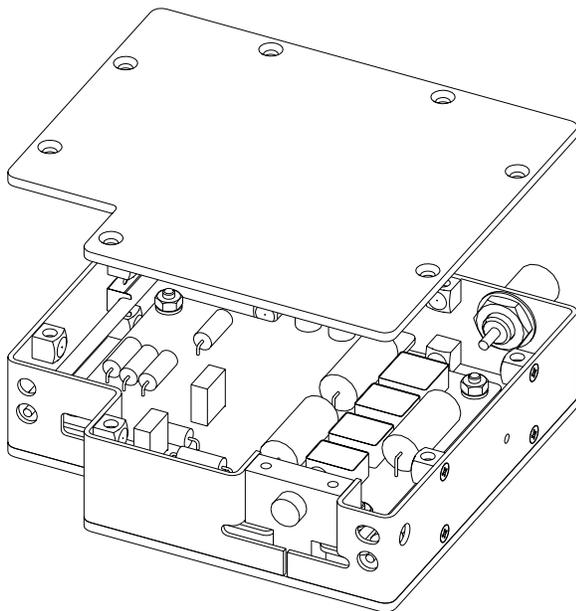


Figure 4: The Rockwell assembly

cessibility questions arise, requiring difficult choices in assembly ordering. The assembly has 78 parts including all fasteners and hardware (the circuit board is modeled as a single part), modeled by about 8 Mb of Pro/ENGINEER<sup>®</sup> data. Only the geometry engine has been applied to the Rockwell assembly; it produces a part-level assembly plan with several non-linear assembly operations in 67 seconds.

### 3.4 Other assemblies

The geometry engine has successfully planned part-level assembly sequences for several other assemblies. These include a subassembly of a fuel pump from Cummins Engine, a seeker assembly modelled in a CAD benchmark at a major missile systems company, and a

Assembly	No. of Parts	ACIS <sup>®</sup> Data (Mb)	Planning Time (sec)
Pattern Wheel	13	0.5	9
Fuel Pump	36	3.6	32
Discriminator	42	4.1	30
Seeker	70	6.1	123
Rockwell	78	3.2	67
Actuator	109	4.5	45

Table 1: Archimedes geometry engine planning times for various assemblies. To calculate ACIS<sup>®</sup> data size, the data for each distinct part is counted only once, regardless of the number of times that part appears in the assembly. Planning times given are to load in the pre-faceted data, identify all contacts in the assembly, and find a single geometrically valid part-level assembly sequence.

locker actuator from NASA/Lockheed. Table 1 summarizes the planning times required for these assemblies. Due to confidentiality agreements with the supplying companies, we cannot show more about these examples.

Interested readers can retrieve color pictures of the above examples, animations of sequences, illustrator output, and video of the robotic workcell assembling the pattern wheel on the World-Wide Web.<sup>1</sup>

## 4 Discussion

Collectively, the Archimedes team has a great deal of experience with automated assembly planning problems and techniques, and with industrial expectations regarding these capabilities. In general, we have found that computer-aided assembly planning methods are quite difficult to apply to real-world problems at the current time. We discuss some of the lessons we have learned below, as well as their implications for future research in assembly planning. While discussed

<sup>1</sup><http://www.sandia.gov/2121/archimedes/archimedes.html>

in terms of limitations in the current Archimedes system, most of these are difficult issues not addressed adequately by any assembly planners to date.

#### 4.1 Missing constraints

Several types of plan details are entered by hand in the current system, including gripper design and grasp planning, fixture design, and motion strategies for part mating. Gripper design, grasp planning, and fixture design are closely related areas, and have been widely studied for holding single parts (see e.g. [4, 6, 16, 27, 30, 32]). However, stability of and holding assemblies has received much less attention [26, 38], and these methods are for the most part not ready to be applied automatically. Similarly, research in automated fine-motion planning [8, 19, 24] has yet to find practical application. But the most important, and unaddressed, question is how to successfully integrate these techniques into the assembly planning process, where constraints flow both ways between the sequence and the tooling and motions used to accomplish it.

#### 4.2 Search

The A\* search algorithm currently employed in the state-space planner quickly becomes impractical when more than a small number of parts are present. We have not decided how to address this problem, but it is clear that optimality will have to be sacrificed, as for instance in [5].

Another limitation with the search strategy in the current system is that planning only proceeds top-down: the system has no recourse when an assembly sequence generated by the state-space planner cannot be translated to V+ due to additional constraints (such as the need to grasp subassemblies) that only appear in the translator. The obvious solution is simply to backtrack and generate another sequence, but this will quickly become computationally impractical. Perhaps better is to integrate these constraints into the initial sequence generation; however, this does not answer the question of how to structure a large system that must include so many types of constraints.

#### 4.3 Interactivity

Archimedes 2 includes little facility for user interaction. All user input to model the assembly and describe constraints is performed up front, in the design module; this input only describes features of the product itself. The system should allow designer-intended assembly plans or plan fragments to be entered, and allow interactive choices of assembly plans later in planning due to manufacturing constraints unknown to the programs. Methods to address these issues exist, but are not perfect. They either require editing a huge set of sequences [1], require constraints to be specified in advance [2, 9], or only apply to certain geometric details [34]. Techniques that allow an engineer to richly and fully participate during the planning process are needed.

#### 4.4 Non-geometric data

Several types of data are not typically captured in CAD models of assemblies, yet are critical to assem-

bly planners. Often the information is called out on the final drawings (like weld sites), or implicit in part IDs (screw threads), but not represented in computer-accessible form. The design module captures some of this information currently, and we are adding more richness to our models. While it is straightforward to add this capability to a modeler, data coming from industrial users typically does not include the information needed for planning. Creating auxiliary data for industrial-size assemblies can be quite time-consuming, and it would be preferable to have the information in the model or derive it automatically (as the geometry engine does for simple contacts).

#### 4.5 Assembly modeling

This is related to the preceding item, but harder to patch. In typical CAD systems today, assemblies are modeled as individual parts, placed in their relative locations. This view of assemblies overconstrains certain parts that are free to move when assembled. However, non-monotone assembly planning would be required to handle these freedoms; very few systems attempt non-monotone planning [10], and it is known to be computationally intractable [36, 28]. This way of modeling assemblies also causes inconsistencies for parts that deform when assembled—interference-fit parts intersect, and springs intersect with other parts—and often for threaded contacts as commonly modeled (cylinders at the outside of the male threads and the inside of the female threads). These inconsistencies in the assembly models cause severe problems for collision-detection algorithms.

This problem of inconsistent data for analysis is not particular to assembly planning; many other types of computer analysis, such as finite element methods and automatic NC program generation, have suffered from incomplete CAD data. In general, CAD systems create data for certain purposes—generating drawings was the first and still the most important purpose, but automated analysis is coming into wider use. When automated assembly planning presents sufficient value, engineers will enter models accurate enough to enable it. Work in PDES/STEP [15, 22] promises to make representations of such information standard, although in practical terms it will be many years before the critical assembly modeling capabilities of STEP are widely available.

#### 4.6 Level of planning detail

Several of the above paragraphs point to a fundamental difficulty in performing research in assembly planning. Fully specified assembly plans must satisfy many constraints and specify many details, ranging from part accessibility, to stability and fixturing, to fine motion plans, to the people or technology targeted to assemble the product. A more aggressive definition would also include factory layout and scheduling, inventory control, and many other factors. In industry, a detail in any of these areas might drive a choice of assembly plan. Hence to be useful, an assembly planning system must somehow enable generation of the “right” plan according to all these constraints.

However, most of these are active areas of research in themselves. Even if techniques existed to evaluate

them, constructing a system that integrated all the constraints on assembly plans would be a huge effort, and it is not at all clear how they should be structured. Hence researchers in assembly planning are placed in the uncomfortable position of trying to integrate many not-fully-understood constraints, and yet produce useful output.

Partly in response to these requirements, much assembly planning research has focused on languages to describe general assembly constraints (see e.g. [1, 13, 37]), often using *precedence constraints*, logical relations specifying partial orders of assembly operations or states. While these languages might allow encoding some constraints, it seems many manufacturing constraints will resist being efficiently expressed in them. This leaves the twin problems of rigorously assessing complex constraints and structuring the system in an efficient manner unresolved. Future research in assembly planning must either address these problems head-on, or work in their shadow.

## 5 Conclusion

We have briefly described Archimedes 2, a second generation assembly planning system that both provides a general high-level assembly sequencing capability and, for a much smaller class of products, facilitates automatic programming of a robotic workcell to assemble them. Because Archimedes can input CAD data in several standard formats, we have been able to test it on a number of industrial assembly models more complex than any before attempted by automated assembly planning systems. These experiments, and our interaction with industrial manufacturing engineers, have led us to a number of conclusions about the state of assembly planning research, and our own directions in particular.

## References

- [1] 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 Trans. on Robotics and Automation*, 7(1):78–94, 1991.
- [2] N. Boneschanscher and C. J. M. Heemskerck. Grouping parts to reduce the complexity of assembly sequence planning. In E. A. Puente and L. Nemes, editors, *Information Control Problems in Manufacturing Technology 1989: Selected Papers from the 6th IFAC/IFIP/IFORS/IMACS Symposium*, pages 233–238. Pergamon Press, 1989.
- [3] A. Bourjault. *Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires*. PhD thesis, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, 1984.
- [4] R. C. Brost and K. Y. Goldberg. A complete algorithm for synthesizing modular fixtures for polygonal parts. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 535–542, 1994.
- [5] S. Chakrabarty and J. Wolter. A hierarchical approach to assembly planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 258–263, 1994.
- [6] M. R. Cutkosky. *Robotic Grasping and Fine Manipulation*. Kluwer Academic Publishers, 1985.
- [7] K. V. Diegert, R. G. Easterling, M. R. Ashby, G. L. Benavides, C. Forsythe, R. E. Jones, D. B. Longcope, and S. W. Parratt. Achieving agility through parameter space qualification. In *Proc. Agile Manuf. Enterprise Forum*, 1995.
- [8] M. A. Erdmann. Using backprojections for fine motion planning with uncertainty. *Intl. J. of Robotics Research*, 5(1):19–45, 1986.
- [9] J. M. Henrioud and A. Bourjault. LEGA: a computer-aided generator of assembly plans. In [11], pages 191–215.
- [10] R. L. Hoffman. Automated assembly in a CSG domain. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 210–215, 1989.
- [11] L. S. Homem de Mello and S. Lee, editors. *Computer-Aided Mechanical Assembly Planning*. Kluwer Academic Publishers, 1991.
- [12] L. S. Homem de Mello and A. C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 7(2):228–240, 1991.
- [13] Y. F. Huang and C. S. G. Lee. Precedence knowledge in feature mating operation assembly planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 216–221, 1989.
- [14] S. A. Hutchinson and A. C. Kak. Extending the classical AI planning paradigm to robotic assembly planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 182–189, 1990.
- [15] ISO. *ISO 10303: Product Data Representation and Exchange*, 1994.
- [16] J. Jones and T. Lozano-Pérez. Planning two-fingered grasps for pick-and-place operations on polyhedra. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 683–688, 1990.
- [17] R. E. Jones and S. G. Kaufman. Automatic assembly planning and its role in agile manufacturing: A sandia perspective. In W. E. Alzheimer, M. Shahinpoor, and A. Bagchi, editors, *Rapid Prototyping: A Paradigm to Agile Manufacturing*. ERI Press, Albuquerque, NM, 1993.
- [18] H. Ko and K. Lee. Automatic assembling procedure generation from mating conditions. *Computer Aided Design*, 19(1):3–10, 1987.
- [19] J.-C. Latombe, A. Laganas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence*, 52:1–47, 1991.

- [20] S. Lee. Backward assembly planning with DFA analysis. In [11], pages 341–381.
- [21] L. I. Lieberman and M. A. Wesley. AUTOPASS: An automatic programming system for computer controlled mechanical assembly. *IBM J. of Research and Development*, 21(4):321–333, 1977.
- [22] T.-H. Liu and G. W. Fischer. Developing feature-based manufacturing applications using PDES/STEP. *Concurrent Engineering: Research and Applications*, 1(1):39–50, 1993.
- [23] T. Lozano-Pérez, J. L. Jones, E. Mazer, P. A. O’Donnell, W. E. L. Grimson, P. Tournassoud, and A. Lanusse. Handey: A task-level robot system. In *Robotics Research: The Fourth Intl. Symposium*, pages 29–36. MIT Press, 1988.
- [24] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *Intl. Journal of Robotics Research*, 3(1):3–24, 1984.
- [25] T. Lozano-Pérez and P. H. Winston. LAMA: A language for automatic mechanical assembly. In *Proc. Intl. Joint Conf. on Artificial Intelligence*, 1977.
- [26] R. Mattikalli, D. Baraff, and P. Khosla. Finding all gravitationally stable orientations of assemblies. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 251–257, 1994.
- [27] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2:541–558, 1987.
- [28] B. K. Natarajan. On planning assemblies. In *Proc. 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.
- [29] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, 1980.
- [30] J. Pertin-Troccaz. Grasping: A state of the art. In Khatib, Craig, and Lozano-Pérez, editors, *The Robotics Review 1*. MIT Press, 1989.
- [31] D. R. Strip and A. A. Maciejewski. Archimedes: An experiment in automating mechanical assembly. In *Proc. 11th ASME Intl. Conf. on Assembly Automation*, 1990.
- [32] A. S. Wallack and J. F. Canny. Planning for modular and hybrid fixtures. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 520–527, 1994.
- [33] R. H. Wilson. *On Geometric Assembly Planning*. PhD thesis, Stanford Univ., March 1992. Stanford Technical Report STAN-CS-92-1416.
- [34] R. H. Wilson. Minimizing user queries in interactive assembly planning. *IEEE Trans. on Robotics and Automation*, 11(2):308–312, 1995.
- [35] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.
- [36] J. D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly*. PhD thesis, Univ. of Michigan, 1988.
- [37] J. D. Wolter, S. Chakrabarty, and J. Tsao. Mating constraint languages for assembly sequence planning. *IEEE Trans. on Robotics and Automation*. To appear.
- [38] J. D. Wolter and J. C. Trinkle. Automatic selection of fixture points for frictionless assemblies. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 528–534, 1994.