SANDIA REPORT

SAND99-2809
Unlimited Release
Printed January 2000

Human Assisted Assembly Processes

Terri L. Calton and Ralph R. Peters

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (703) 605-6000
Web site: http://www.ntis.gov/ordering.htm

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

SAND99-2809
Unlimited Release
Printed January 2000

Human Assisted Assembly Processes

Terri L. Calton and Ralph R. Peters
Intelligent Systems and Robotics Center
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1008

Abstract

Automatic assembly sequencing and visualization tools are valuable in determining the best assembly sequences, but without Human Factors and Figure Models (HFFMs) it is difficult to evaluate or visualize human interaction. In industry, accelerating technological advances and shorter market windows have forced companies to turn to an agile manufacturing paradigm. This trend has promoted computerized automation of product design and manufacturing processes, such as automated assembly planning. However, all automated assembly planning software tools assume that the individual components fly into their assembled configuration and generate what appear to be a perfectly valid operations, but in reality the operations cannot physically be carried out by a human. Similarly, human figure modeling algorithms may indicate that assembly operations are not feasible and consequently force design modifications; however, if they had the capability to quickly generate alternative assembly sequences, they might have identified a feasible solution. To solve this problem, HFFMs must be integrated with automated assembly planning to allow engineers to verify that assembly operations are possible and to see ways to make the designs even better.

Factories will very likely put humans and robots together in cooperative environments to meet the demands for customized products, for purposes including robotic and automated assembly. For robots to work harmoniously within an integrated environment with humans the robots must have cooperative operational skills. For example, in a human only environment, humans may tolerate collisions with one another if they did not cause much pain. This level of tolerance may or may not apply to robot-human environments. Humans expect that robots will be able to operate and navigate in their environments without collisions or interference. The ability to accomplish this is linked to the sensing capabilities available. Current work in the field of cooperative automation has shown the effectiveness of humans and machines directly interacting to perform tasks. To continue to advance this area of robotics, effective means need to be developed to allow natural ways for people to communicate and cooperate with robots just as they do with one another.

Intentionally Left Blank.

Preface

This manuscript serves as the technical report required by Sandia National Laboratories Laboratory Directed Research and Development (LDRD) Program for the project entitled "Ergonomics in Life Cycle Assembly Processes". The LDRD Program financially supported the entire project and this manuscript summarizes the technical achievements made throughout the project's two-year history. The work conducted under the purview of the LDRD focused primarily on automating analysis and planning techniques for human assisted assembly processes. However, to facilitate solutions to additional needs from within the Sensing and Intelligent Controls Investment Area, additional goals were added during the second year of the LDRD. Of particular interest was the need to more tightly couple the enabling of human skills in cooperative automation with automated assembly analysis. It was hoped that such a coupling would provide improved intelligent controls for creating cooperative automated assistants for a human's logistical capabilities and safety in hazardous operating environments.

In alignment with the goals outlined in the re-direct of the LDRD, efforts were combined with equal amounts from three separate LDRDs ("Enabling Human Skills with Cooperative Automation", "Adaptive 3D Sensing", and "Volumetric Video Motion Sensing for Unobtrusive Human-Computer Interactions"). The major technological achievement under this project's purview was the development of vision algorithms to recognize (static) human gestures for use in robot communication and cooperation.

In support of the project's original objectives, Sandia's Automated Assembly Analysis System, Archimedes 3.0©, was used as the foundation for integrating algorithms that automatically generate assembly and disassembly human assisted assembly processes. These results, combined with the results produced under the purview of two other LDRDs projects, "Automatic Planning for Life Cycle Assembly Processes" (a three-year LDRD (FY97 - FY99)) and "Analysis of Very Large Assemblies", (a two-year LDRD (FY98 - FY99)) constitute the features used to upgrade Archimedes 3.0© to Archimedes 4.0.

Archimedes 3.0© was copyrighted in 1998. The copyright for Archimedes 4.0 is in progress. The fundamental planning concepts underlying Archimedes 3.0© is the manufacturing plan selection criteria (constraints) framework. A patent release for this framework combined with the planning algorithms is expected to be issued early in the calendar year of 2000.

To better assist the reader of this manuscript, Appendices which are devoted to the manufacturing constraints framework and planning algorithms of Archimedes 3.0© have been included. The depth of coverage in the appendices provides the necessary background for the

first two chapters, which focus on assembly planning algorithms for human-assisted assembly processes.

Acknowledgements

The authors wish to thank the following people (all Sandians) for their many contributions to this project: Bob Anderson, Russell Brown, Jeff Carlson, Jill Fahrenholtz, Scott Gladwell, Yong Hwang (no longer with Sandia), Dan Schmitt, Dan Small, Jeff Trinkle, and Peter Watterberg. Bob, Russell, Jill, Scott, Yong, Jeff Trinkle, and Peter helped develop the framework for geometric reasoning about human figures and factors in assembly processes. Dan Schmitt, Jeff Carlson and Dan Small are largely responsible for the gesture recognition work.

October 1999

T.L. Calton

R.R. Peters

Contents

Figures

Intentionally Left Blank.

Executive Summary

Incorporating human figure models into automated assembly sequence planners requires addressing the following issues. Initially, a robust geometric and kinematic model of a human must be realized for accurate interference checking and reachability analysis. Secondly, models of a human hand grasping tools need to be defined. Thirdly, human factors need to be incorporated into the human model to evaluate ergonomic aspects of assembly operations. Fourthly, geometric-reasoning algorithms need to be developed to either determine feasible remove-and-replace operations for humans or suggest a design change. Lastly, an efficient human motion planner must be developed to achieve collision-free paths for the assembly operations. Implementations of these capabilities and optimizations of component designs and human motions need to accurately reflect real-world situations, but at the same time introduce approximations to make the computation tractable, (i.e., polynomial time). A general framework for realizing these issues has been developed by integrating commercially available human figure modeling software packages with Sandia's Automated Assembly Analysis System, known as Archimedes 3.0©. Chapter 1 outlines the framework for integrating the geometry-based assembly planning algorithms of Archimedes 3.0© with two commercially available human figure modeling software packages, Envision's Deneb/ERGO™ module and Jack® by Transom™, Inc. What differentiates this methodology from other approaches is the capability to automatically analyze complete assembly and disassembly by representing and reasoning about human related geometric issues and human factors for a variety of hand tools used in assembly operations. What is lacking in this framework is the ability to quickly generate collision-free motion paths and hand-grasps for the human. Future work should be aimed at integrating automatic motion planning and automatic grasp planning algorithms into the framework.

Chapter 2 presents an integrated solution for combining automated fixture design tools and automated assembly planning. This initiative focused on integrating Sandia's Fixture Design Software Package, HoldFast™, with Sandia's Automated Assembly Analysis Software Package, Archimedes 3.0©. The integration resulted in a two-phased approach. In the first phase, HoldFast™ was upgraded to use ACIS® 3D objects. Specific tasks that were required for the ACIS® upgrade included converting the HoldFast™ geometry from 2-1/2-D to 3D and developing interface capabilities between HoldFast™ and ACIS®. HoldFast™ previously used 2-1/2D geometry to interrogate an object's geometry. Those functions were rewritten in order to be compatible with the ACIS® 3D representation. Also, since HoldFast™ had initially been written in LISP and ACIS® is written in C++, interface routines had to be developed to allow a seamless transition between HoldFast™ and ACIS®. When the conversion was complete, the actual integration of the two software packages was made. The integration method starts with three-dimensional CAD descriptions of an assembly whose assembly tasks require a fixture to hold the

base part. CAD descriptions of that part are used by the fixture planning algorithms to generate geometric representations of a fixture pallet for use in the assembly planning software. (The resulting fixture is designed to rigidly constrain and locate the part, obey various task constraints, is robust to part shape variations, is easy to load and is economical to produce.) The geometric descriptions generated by the fixture planner and additional assembly components are then loaded into the assembly system for analysis. Using the assembly analysis constraint framework, the fixture is first declared as a base component. The assembly analysis system then tests for feasibility of tool use during the pallet assembly and assembly of the components. The assembly planner not only generates assembly plans for the assembly, but is used to generate assembly plans for the fixture as well. The combined algorithms guarantee the fixture will hold the base part without interfering with any of the assembly operations. What is lacking in this framework is the ability for the assembly planner to automatically provide feedback to the fixture planner. Future work should be aimed at providing a more complete integration of the planners.

People communicate with each other mainly through voice and body language. Natural interaction consists of two major parts, natural language understanding and synthesis and visual observation of the users body pose and gestures. One effective means for a human to communicate with a machine is to use body gestures similar to those used when communicating with another person. These gestures include such things as pointing to indicate a destination. The most important features of the human body that provide information about the intentions and desires of a user are the arms/hands and the head. Computer vision, using low cost devices, makes it possible for a human to use body motions or any convenient object-as digital input devices. Chapter 3 describes methods that were implemented for extracting static pointing gesture information provided by a human from Sandia's 3D Video Motion Detection System for use in cooperative automation environments. It further describes the static pointing gesture interpretation algorithms used for robot-assisted operations. The static pointing gestures were used to indicate predefined portions to which a robot should move, and perform the appropriate task at that destination. By pointing to those destinations, the operator commands the robot to move to that destination and perform the task described at that position. Future work in this area should encompass the development of an instruction language for robots using dynamic hand and arm gesture and the development of robust and real-time recognition and interpretation algorithms.

Human Assisted Assembly Processes

Intentionally Left Blank.

1

A Framework for Geometric Reasoning about Human Figures and Factors in Assembly Processes

Automatic assembly sequencing and visualization tools are valuable in determining the best assembly sequences, but without Human Factors and Figure Models (HFFMs) it is difficult to evaluate or visualize human interaction. In industry, accelerating technological advances and shorter market windows have forced companies to turn to an agile manufacturing paradigm. This trend has promoted computerized automation of product design and manufacturing processes, such as automated assembly planning. However, all automated assembly planning software tools assume that the individual components fly into their assembled configuration and generate what appear to be perfectly valid operations, but in reality some operations cannot physically be carried out by a human. For example, the use of a ratchet may be reasoned feasible for an assembly operation; however, when a hand is placed on the tool the operation is no longer feasible, perhaps because of inaccessibility, insufficient strength, or human interference with assembly components. Similarly, human figure modeling algorithms may indicate that assembly operations are not feasible and consequently force design modifications; however, if they had the capability to quickly generate alternative assembly sequences, they might have identified a feasible solution.

To solve this problem, HFFMs must be integrated with automated assembly planning. This allows engineers to quickly verify that assembly operations are possible and to see ways to make the designs even better.

This chapter presents a framework for integrating geometry-based assembly planning algorithms with commercially available human figure modeling software packages. Experimental results to selected applications along with lessons learned are presented.

1.1 Introduction

All automated assembly planning software tools assume that the individual components fly into their assembly positions; in reality they have to be moved with human hands or other mechanical devices. These assembly planning software tools are valuable for determining the best assembly and/or disassembly sequences/plans, but without Human Factors and Figure Models (HFFMs), it is difficult to evaluate or visualize human interaction. Automated assembly planning software tools that include HFFMs allow engineers to quickly verify that assembly and disassembly operations are possible and to see ways to make a design even better.

Incorporating human figure models into automated assembly sequence planners requires addressing the following issues. Initially, a robust geometric and kinematic model of a human must be realized for accurate interference checking and reachability analysis. Secondly, models of a human hand grasping tools need to be defined. Thirdly, human factors need to be incorporated into the human model to evaluate ergonomic aspects of assembly operations. Fourthly, geometric-reasoning algorithms need to be developed to either determine feasible remove-and-replace operations for humans or suggest a design change. Lastly, an efficient human motion planner must be developed to achieve collision-free paths for the assembly operations. Implementations of these capabilities and optimizations of component designs and human motions need to accurately reflect real-world situations, but at the same time introduce approximations to make the computation tractable, (i.e., polynomial time). A general framework for realizing these issues has been developed by integrating commercially available human figure modeling software packages with Sandia's automated assembly analysis and planning software tool, known as Archimedes 3.0©. Appendices A and B outline the assembly planning system. What differentiates this methodology from other approaches is the capability to automatically analyze complete assembly and disassembly by representing and reasoning about human related geometric issues and human factors for a variety of hand tools used in assembly operations.

This chapter outlines a framework for integrating the geometry-based assembly planning algorithms of Archimedes 3.0© with two commercially available human figure modeling software packages, Envision's Deneb/ERGO™ module [5] and Jack® by Transom™, Inc. [7]. Section 1.2 describes the process used to select the human modeling software packages. Section 1.3 provides a high level abstraction of the Archimedes 3.0© Automated Assembly Analysis Software Tool. Critical for successful integration of HFFMs and assembly processes is the ability to automatically reason hand/tool geometry for the tools used in the operations. Section 1.4 gives a brief description of the Archimedes 3.0© tools constraints framework and how it has been extended to incorporate human factors. In Section 1.5, the integration platforms are described. Section 1.6 describes the experiments conducted in each of the Deneb/ERGO™ and Transom™ Jack®

integration phases with Archimedes 3.0©. Finally, Section 1.7 concludes the manuscript with lessons learned and suggests areas for future work.

1.2 Commercially Available Software

There are a small number of human modeling software packages commercially available that consider human factors for manufacturing processes, all with varying degrees of sophistication. A 6-month survey was conducted to identify the best commercially available human figure modeling software packages as probable candidates to integrate with the Archimedes 3.0© assembly planning algorithms. The criteria used to select the human modeling packages emphasized the maturation of geometric and kinematic models, motion planning capabilities, and human factors capabilities (e.g., accessibility, visibility, and strength).

The Logistics Research Division of the Armstrong Laboratory contracted with Raytheon (formerly Hughes) Missile Systems Company to develop a 3D-simulation maintenance analysis tool. This tool, DEPTH, [6], (Depth Evaluation for Personnel, Training Human-factors) uses articulated 3D human figure models provided by the Jack® software package developed at the Center for Human Modeling and Simulation at the University of Pennsylvania (available through Transom™ Inc.) to simulate maintenance operations and to report information on human factors. Transom™ Jack® provides a 3D interactive environment for controlling articulated figures. It features a detailed human model and includes realistic behavioral controls, anthropometric scaling, task animation and evaluation systems, view analysis, automatic reach and grasp, collision detection and avoidance, and many other useful tools for a wide range of applications, [7]. However, its motion planning capability is limited and it fails to find a solution in a cluttered space.

McDonnell Douglas is working on a system that enables electronic simulation/demonstration of assembly, and maintenance operations early in the design process using 3D animated human manikins, [10]. This system is a menu-driven, interactive computer program used to define design requirements and aid in design evaluation but is lacking in human factors analysis.

Human Centered Design is developing SAFEWORK©, the Human Modeling Software for Advanced Ergonomic Design. SAFEWORK© is a software tool that creates virtual humans of various percentiles to study fit and accessibility in a workstation, [11]. The system exhibits some of the same features as Jack®, but tends to focus more on human factors and not necessarily mechanical assembly.

Deneb has developed a product called Deneb/ERGO™. Deneb/ERGO™ is an accurate, physics-based human motion analysis tool which can be used to design safe working environments accommodating a wide rang of workers. With the Deneb/ERGO™ module, human factors can be evaluated and both the designer and the end-user can resolve ergonomic issues like Design for Assembly (DFA) early in the design process. Users can prototype and analyze human motion, including human joint reach and motion, reach and accessibility, and visual perspective. Deneb/ERGO™ also includes advanced techniques for determining safe postures, lifting and

energy expenditure evaluation, [5]. However, like Transom™ Jack®, the Deneb/ERGO™ motion planning capability is limited and it too fails to find a solution in a cluttered space.

These and other systems tend to bias either human factors or mechanical assembly but not both and none are capable of automated assembly reasoning and sequencing. Even though the Deneb/ERGO™ and Transom™ Jack® packages both exhibited limitations in their motion path planning capabilities as well as in their grasp planning capabilities, they appeared to be the most mature and showed the best potential for automatically reasoning human factors in an assembly environment.

1.3 Assembly Planning Overview

Because of its powerful geometric reasoning algorithms and assembly constraint framework, Sandia's Automated Assembly Analysis System, Archimedes 3.0© [3,4,8], was selected as the basis for the development of a framework for realizing HFFMs in assembly processes. Archimedes 3.0© is a constraint-based, interactive assembly planning software system used to plan, optimize, simulate, and document sequences of assembly. Archimedes 3.0© is a constraint-based interactive assembly planning software tool used to plan, optimize, simulate, visualize, and document sequences of assembly. Given a CAD model of the product, the program automatically finds part-to-part contacts, generates collision-free insertion motions, and chooses assembly order. Disassembly operations are generated using the Non-Directional Blocking Graph approach discussed in [13]. A graphics workstation's hardware Z-buffer is used to quickly find collisions between complex facetted models. The search space implemented in the Archimedes 3.0© System is an AND/OR graph of subassembly states and the operations used to construct them from smaller subassemblies. The strategy is designed to generate a first plan as quickly as possible, like a depth-first search, but to avoid being caught by bad early decisions as a depth-first search would. This is critical to achieve the desired view-constrain-re-plan cycle of interaction. During each pass of the search algorithm, a single assembly sequence is generated, making random choices of operations to construct each subassembly. The first time any subassembly is visited, only a single operation is generated to construct it, and the known subassemblies of that operation are then visited. Bounds on quality measures for each subassembly and operation are stored and propagated in the AND/OR graph as they are generated. This allows useless search paths to be identified and pruned, and an optimal plan to be identified when it becomes available. The same algorithm functions as an any-time algorithm to optimize the assembly sequence when the user requests.

In an Archimedes 3.0© application session, the engineer specifies a quality metric in terms of application-specific costs for standard assembly process steps, such as part insertion, fastening, and subassembly inversion. Combined with an engineer's knowledge of application-specific assembly process requirements, Archimedes 3.0©allows systematic exploration of the space of possible assembly sequences. The engineer uses a simple graphical interface to place constraints on the valid assembly sequences, such as defining subassemblies, requiring that certain parts be placed consecutively with or before other parts, declaring preferred directions, etc. Archimedes 3.0© is implemented in C++ using the ACIS® solid modeling kernel and Tcl/Tk™ for the graphical interface. Animation and user interface routines use OpenGL™ and X Windows™. The reader is

referred to Appendices A and B for a broader overview the Archimedes 3.0© System and assembly planning constraint framework.

1.4 Tools Constraint Framework

Constraints on assembly plans vary depending on product, assembly facility, assembly volume, and come from a wide variety of choices. Design requirements, part and tool accessibility, assembly line and workcell layout, requirements of special operations, and even supplier relationships can drive the choice of a feasible or preferred assembly sequence. Computer-aided assembly planning systems promise to help product and assembly system designers to manage this complexity and choose good assembly sequences.

Two types of constraints on assembly plans are utilized by the Archimedes 3.0© System. They are classified as strategic constraints and tactical constraints. Strategic constraints apply to the entire assembly and its plan, while tactical constraints only apply to certain subsets of the parts. The constraint framework provides a library of constraint types from which a user can instantiate constraints on the assembly plan. This framework provides the underlying mechanics to optimization algorithms.

Planning for assembly requires reasoning about various tools used by humans, robots, or other automation to manipulate, attach, and test parts and subassemblies. Constraints on assembly plans deriving from the need to use various tools in assembly or disassembly are called tool constraints. A framework to represent and reason about geometric accessibility issues for a variety of such assembly tools is implemented in Archimedes 3.0© [12]. This framework is extremely important to the integration with human figures.

Tool constraints are implemented as just another type of process constraint. Spatial constraints on applying tools are characterized by certain volumes in the assembly. The tool-volume is the spatial extent of the tool itself. The use-volume of a tool is the space swept out by the tool as it is applied to a part. Archimedes 3.0© uses predefined swept volumes for tools. Several different use-volumes exist for the same physical tool, corresponding to different ways of moving it or using it to accomplish a task. For planning purposes, each different use-volume for a physical tool is treated as a distinct canonical tool.

Archimedes 3.0© uses a straightforward instance of the FINDPLACE problem [9] to find a collision-free placement of use-volume; however, this method does not include the space required by the human hand and/or arm wielding the tool or the volume swept by moving the tool to the application point. To overcome the former, the tools framework was extended to accommodate for hand held tools by predefining discrete handgrips for a subset of the hand tools in the Archimedes 3.0© tool library. For each of the defined handgrips, upper bounds on the space required to apply it were established; such volume represents sufficient conditions for tool applications. This "sufficient" volume includes the space swept out by the tool plus the hand that wields it, once the tool has been placed on the assembly, applied, and removed. To overcome the latter, new motion path planning algorithms were implemented within the Deneb environment, while existing algorithms embedded the Jack® software were implemented. The pros and cons to each approach will be discussed in the following sections.

1.5 The Integration

Both the Deneb/ERGO™ and the Transom™ Jack® human modeling software products require virtually the same input data to realize human factors in the assembly operations automatically generated by Archimedes 3.0©: assembly and component geometry information and assembly sequence steps with mating trajectories. To integrate Archimedes 3.0© with both the Deneb/ERGO™ and Transom™ Jack® software packages, two basic steps were required. First, the Archimedes 3.0© assembly planner was used first to automatically analyze the feasibility of assembly and then to automatically generate an assembly sequence for selected assemblies requiring the use of hand tools to put them together. A textual script capturing this assembly sequence, part-mating trajectories, and tool-and-hand application trajectories was generated. Independent output routines were added to the Archimedes 3.0© output module to realize the subtle differences between the two input formats required by Deneb/ERGO™ and Transom™ Jack®. Second, the ACIS® geometry files used in Archimedes 3.0©, including assembly components, tools and respective use-volumes, and handgrips on those tools with their respective tool-and-hand use-volumes, were converted to the IGRIP® files. Both Deneb/ERGO™ and Transom™ Jack® accept IGRIP® format as input.

1.6 Experiments

To validate the integration steps, an assembly requiring the use of hand tools was selected. Figure 1.1 shows an assembly fixture kit, a housing part manufactured by casting a near-net-shape part, and electrical components to be placed into the housing. Before the housing was fabricated or machined-finished, CAD designs of it, the fixture design, and additional assembly components were loaded into the Archimedes 3.0© System for analysis. Using the constraint framework, the fixture was declared as a base component. This meant that the fixture pallet components were defined as the set of parts to be assembled first and would then remain stationary throughout the assembly analysis. A socket wrench and phillips-screwdriver, both with pre-selected handgrips, were tested for feasibility of use during the pallet assembly and assembly of the housing and electrical components. The Archimedes 3.0© System identified four potential design flaws in the housing that prohibited the use of the screwdriver, but not the hand, for attaching the cover plate. Once the design was corrected, an assembly plan was generated and all of the information listed in Section 5 was exported to the Deneb/ERGO™ and the Transom™ Jack® software packages.

Figure 1.1. Transom™ Jack® is performing an assembly operation generated in Archimedes 3.0©.

As pointed out in the opening remarks of this manuscript, all assembly planners, including Archimedes 3.0©, assume parts and tools just "fly" into their assembled configurations once the mating trajectories are determined. For this reason, the human models were placed at a workstation table with the tools and unassembled components placed to one side. The placement locations provide the starting positions of the paths, while the mating trajectories generated in Archimedes 3.0© define the tail of the paths. Each assembly instruction requiring the use of a tool was carried out by the human figures by "re-attaching" the hand (with the predetermined grips for the tools) to the models. For the assembly operations not requiring the use of a hand tool, manually generated instructions had to be provided to both of the human models on how to grasp the components to avoid collision during component placement.

To facilitate Deneb/ERGO™'s limited collision avoidance capabilities (i.e., lots of tweaking on joints is required), a 7-DOF inverse kinematic numerical solution and a straight-line motion planner were implemented to compute the arm joint angles that are necessary to move the tool-and-hand along a desired path. These algorithms were selected from Sandia's Modular Architecture for Robotics Tele-operation, SMART [1], software packages. The selection of algorithms resulted from the need to simulate very natural human motions and intuitive human factors. The algorithms, originally designed for use with redundant robotic manipulators, ensure minimal energy paths subject to joint gravity constraints and can be altered to ensure non-fatiguing poses and optimization of linear velocity constraints can be correlated to human-related stress calculations. With these algorithms and the remaining modules in SMART it was relatively easy to take the series points generated by Archimedes 3.0© combined with additional manually-generated path-defining points and teleoperate the arm with a given tool grip to move through these points.

With Transom™ Jack®, a less rigorous approach was taken. With the realization of Jack®'s limited motion path planning capabilities in a cluttered environment, collision-free paths were generated by specifying a series of points between the pick and place of the components. Using Jack®'s own inverse kinematic algorithms, the points along the paths were tweaked until realistic motions were realized.

During simulation of assembly by both human models additional human factors, such as reachability, stress, fatigue, etc., were obtained: both the Deneb/ERGO™ and Transom™ Jack® software packages have sophisticated algorithms for calculating human factors for ergonomic issues. However, these calculations were not directly coupled with the assembly planning geometric reasoning algorithms. A request to modify one of these factors meant that Archimedes 3.0© had to be run several times before realizing the desired effect. Each run meant new paths had to be created in both Deneb/ERGO™ and Transom™ Jack® software. This was a long and tedious process. Future work is planned to tightly couple the human figure modeling software packages with Archimedes 3.0©.

1.7 Conclusions and Future Work

A general framework for geometric reasoning about human figures and factors in assembly process has been demonstrated by the successful integration of the Deneb/ERGO™ and Transom™ Jack® software packages with the automatic assembly planning algorithms of Archimedes 3.0©. What differentiates this methodology from other approaches is the capability to automatically analyze complete assembly and disassembly by representing and reasoning human related geometric issues and human factors for a variety of hand tools used in assembly operations.

What is lacking in this framework is the ability to quickly generate collision-free motion paths and hand-grasps for the human. The Archimedes 3.0© system has been extended to perform interference checks between the parts and tool-and-hand use-volumes and integrated with both of the Deneb/ERGO™ and Transom™ Jack® human modeling software packages. An inverse kinematics routine was integrated with Archimedes 3.0© and the Deneb/ERGO™ product, but it became a bottleneck when modifying assembly sequences; collision-free paths had to be manually generated. Similarly, the existing motion and path planning algorithms in the Transom™ Jack® product required manual intervention for collision-free planning paths. Sandia is currently extending[1] the framework presented here to include its geometric reasoning algorithms in its motion planning software, known as SANDROS[2], to automatically generate collision-free paths for the human arm. Because the human arm is modeled with seven or more degrees of freedom, the motion planning of the arm requires a search in a seven or more dimensional space. An efficient search algorithm needs to be developed to find a feasible arm motion. The existing SANDROS planner searches for a collision-free motion in a robot's joint space [2]; however, humans tend to plan their arm and hand motions in the world space. The SANDROS will be modified to perform its search in the world space so that planned motions resemble human motions more closely.

Automated grasp planning capabilities also needs to developed for the predefined tool grips. This would overcome the problem of having to establish individual hand and hand-and-tool geometric models. A more useful and broader area of research would encompass automatic grasp planning algorithms for the human hand to grasp any object.

An essential component missing in both Deneb/ERGO™ and Jack® is reconfigurable kinematics. Reconfigurable kinematics is the ability to automatically reconfigure human motion in order to overcome task constraints, (e.g., knowing when to transition from a "move from shoulder" operation to a "move from waist" operation or knowing when to change a grasp on a given tool).

---

[1] Sandia has contracted with Lockheed Martin Tactical Aircraft Systems in Fort Worth, Texas under the Lockheed Martin Shared Vision Program to automate the motions of the humans within the Archimedes 3.0© assembly planning framework.

[2] Sandia has previously integrated the SANDROS motion planning algorithms coupled with the C-Space Toolkit [14], a Sandia developed software toolkit used to make fast distance queries, with Deneb's Telegrip® software package [15].

References

1. R. J. Anderson. SMART: A Modular Architecture for Robotics and Teleoperation. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 416-421, 1993.
2. P.C. Chen and Y.K. Hwang. Sandros: A motion planner with performance proportional to task difficult. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 2346-2353, 1992.
3. T.L. Calton. Advancing design-for-assembly: the next generation in assembly planning. In *Proc. of 3rd Intl. Symposium on Assembly and Task Planning*, 1999.
4. T.L. Calton and R. R. Peters. A practical approach for integrating automatically designed fixtures with automated assembly planning. In *Proc. 3rd Intl. Conf. on Engineering and Automation*, 1999.
5. Deneb/ERGO. Located at http://www.deneb.com/.
6. J.D. Ianni. Crew Chief: Present and Future. In *Human Centered Technology for Maintainability Workshop Proceedings*, pp. 32-26, 1991.
7. Jack®, the human modeling and simulation system. Transom™ Technologies, Inc., located at http://130.91.6.84/~hms/jack.html.
8. R.E. Jones, R.H. Wilson, and T.L. Calton. On constraints in assembly planning. In *IEEE Intl. Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 849-863, 1998.
9. T. Lozano-Perez. Spatial planning: A configuration space approach. In *IEEE Transactions on Computers, C-32:108-120, 1983.*
10. McDonnell Douglas Human Modeling System. Located at http://www.boeing.com/assocproducts/hms/index.html.
11. SAFEWORK. Located at http//www.safework.com.
12. R. H. Wilson. Geometric reasoning about assembly tools. *SAND95-2423*, 1997.
13. R. H. Wilson and J. –C. Latombe. Geometric reasoning about mechanical assembly. In *Artif. Intell*, vol. 71, no. 2, pp. 371-396, 1994.
14. P.G. Xavier. Fast Swept-volume distance for robust collision detection. *In Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1162-1169, 1997.
15. P. Watterberg, P. Xavier, Y. Hwang. Path Planning for Everyday Robotics with SANDROS. *In Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1170-1175, 1997.

2

A Practical Approach for Integrating Automatically Designed Fixtures with Automated Assembly Planning

2.1 Introduction

Product assembly problems vary widely; here the focus is on assemblies that are characterized by a single base part to which a number of smaller parts and subassemblies are attached. This method starts with three-dimensional CAD descriptions of an assembly whose assembly tasks require a fixture to hold the base part. It then combines algorithms that automatically design assembly pallets to hold the base part with algorithms that automatically generate assembly sequences. The designed fixtures rigidly constrain and locate the part, obey task constraints, are robust to part shape variations, are easy to load, and are economical to produce. The algorithm is guaranteed to find the global optimum solution that satisfies these and other pragmatic conditions. The assembly planner consists of four main elements: a user interface, a constraint system, a search engine, and an animation module. The planner expresses all constraints at a sequencing level, specifying orders and conditions on part mating operations in a number of ways. Fast replanning enables an interactive plan-view-constrain-replan cycle that aids in constraint discovery

and documentation. The combined algorithms guarantee that the fixture will hold the base part without interfering with any of the assembly operations. This chapter provides an overview of the planners, the integration approach, and the results of the integrated algorithms applied to several practical manufacturing problems. For these problems initial high-quality fixture designs and assembly sequences are generated in a matter of minutes with global optimum solutions identified in just over an hour. The next section introduces a commercially available fixture kit that is used as the basis for the fixture planning algorithms. Section 2.3 introduces an experimental assembly used to illustrate the integration of the planners. Sections 2.4 and 2.5 provide overviews of the planners while Section 2.6 describes the integration process. Finally, Section 2.7 concludes the manuscript and discusses future work in this area.

2.2 The Fixture Kit

Figure 2.1 shows an example of a hole-based modular fixture kit. This fixture kit contains a base plate, side locators, support pads, and side and top clamps required by the fixture planning algorithms. This particular kit is comprised of commercial elements available from Qu-Co, Inc. [6]. The base plate has an alternating grid of smooth and threaded holes, spaced 0.75 inches apart; the lower half of each smooth hole is threaded, so that clamps may be attached to any hole, while side locators may only be attached to alternating holes. All fixture designs returned by the algorithms are comprised of a few basic fixturing elements. These include round lateral locators, a side clamp, cylindrical support pads with a flat or self-aligning top, and swing-arm top clamps with a flat or self-aligning contact pad.

Locating and clamping elements in this class are widely available, and are often employed in fixtures based on the 3-2-1 location scheme [4]. The algorithm to design fixtures that hold the work piece in kinematic form closure uses these elements; that is, part motion is only possible through deformation of either the part or the fixture. The side locators, support pads, and clamps all have associated spacers that allow them to be set at various heights.

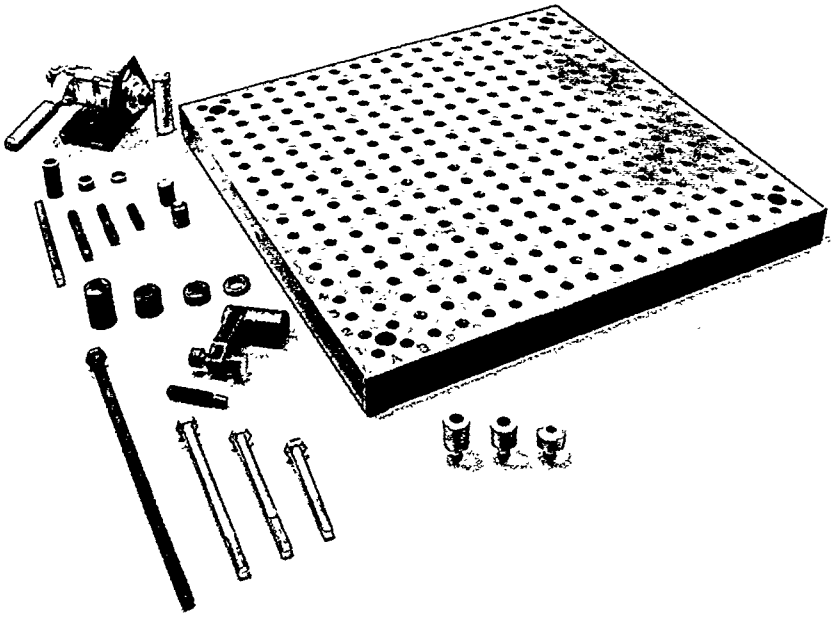

Figure 2.1. Modular fixturing kit from Qu-Co, Inc.

2.3 The Experimental Assembly

Figure 2.2 shows a cast housing and assembly components that will be used throughout this chapter to help illustrate the integration. This assembly was selected for two reasons. First, a fixture was required to hold the base component for final machining operations and not interfere with the cutting paths. Second, a fixture was required to hold the housing while the electronic components were placed into the housing; however, for clarity, the electrical components are not shown.

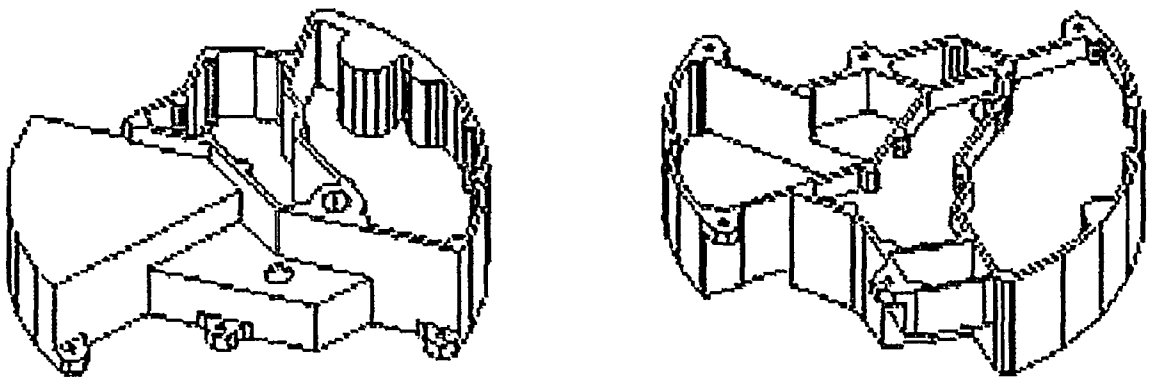

Figure 2.2. An example fixturing problem.

2.4 Automated Fixture Planning

Fixture design is a practical problem. When manufacturing products, it is often necessary to hold a work piece in place during the course of several manufacturing tasks, such as machining, assembly and inspection. The fixtures used to hold the work piece must prevent undesired part motions and avoid interfering with these tasks, often with the additional requirement that the work piece must be held in an accurate, repeatable position. These conditions must be maintained even in the face of small variations in work piece shape. The cost of fabricating assembly fixtures is also a primary concern, because assembly lines often require many copies of these fixtures to carry the assembly from station to station. Thus, assembly fixtures must be inexpensive in order to be cost-effective. To address these issues, Sandia has developed a fixture design software tool called HoldFast™.

Given a fixturing problem specified by a work piece description and a set of task constraints, the HoldFast™ algorithms generate designs that provide form closure while obeying the constraints. Each fixture is analyzed for loadability and passes to a user-defined quality metric that rates the fixture design. This quality metric can consider arbitrary aspects of the fixture in assigning a quality score, and may apply thresholds to determine whether any of these aspects is unacceptably poor. The fixture is discarded if it fails to pass some threshold; otherwise, it is assigned a scalar quality value. The algorithm employs branch-and-bound pruning to identify the global optimum fixture design while exploring only a small portion of the feasible fixture design space. If pruning is turned off, the algorithm enumerates all possible fixture designs. Either way, the algorithm is guaranteed to find the optimal fixture design, for the given set of optimality criteria and associated thresholds. The resulting optimum fixture design is the best possible design that can be constructed with the available fixture elements, subject to several small

approximations described below. The most important of these approximations is that the user may instruct the algorithm to limit its search for side locator height combinations, for example considering only height combinations where each locator is at its lowest available height. This restriction often has a minor effect on fixture quality, but allows a significant reduction in search time. These approximations may be removed at the expense of additional computation.

The quality function used to design fixtures for this example considered three fixture aspects: the fixture's ability to resist expected forces without exerting large reaction forces on the work piece, the fixture's ability to repeatably locate critical features of the work piece, and the ease of loading the fixture. This metric is sensitive to the specific manufacturing operations that will be applied to the work piece, and applies independent thresholds to each of these criteria. Each quality criterion returns a value between 0.0 and 1.0, and the total quality is determined by either a weighted sum or minimum of the three resulting values, based on user preference. This metric is appropriate for a variety of fixture applications, but other metrics may also be supplied to the program.

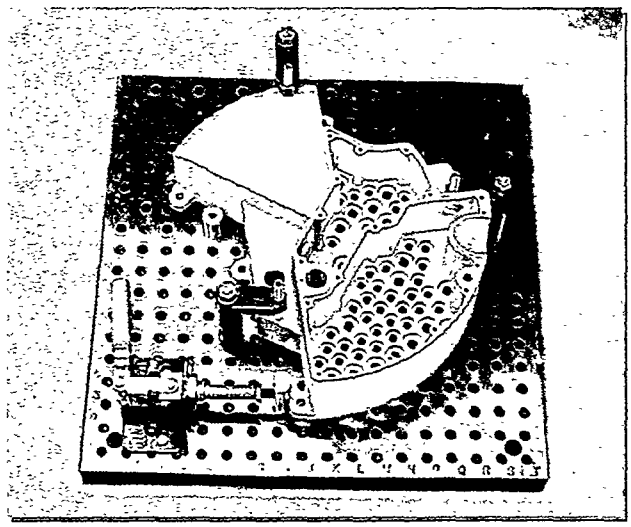

Figure 2.3. HoldFast™ designed fixture.

Figure 2.3 shows a fixture designed by the algorithms for a job-shop-machining problem and component-assembly, comprised of modular elements that may be rapidly assembled in various configurations. In this problem, a near-net-shape casting requires several finish machining operations. This problem is difficult to solve because the complex pockets and large number of holes greatly constrain the placement of support elements. The input description of this problem includes the work piece shape, regions swept by the cutter, expected machining forces, tolerance limits on critical work piece features, and the components of the fixture kit. Every fixture design returned by the algorithm is guaranteed to resist the expected machining forces without causing permanent deformation of the work piece at the fixture contacts. For this particular example assembly, the fixture design was computed in 18 seconds on an SGI workstation.

The algorithm implementation also includes an interface, shown in Figure 2.4, which interactively displays the fixture designs as they are generated. This allows the user to apply subjective evaluation criteria that are not included in the program's quality metric and to stop the computation once an acceptable fixture design is generated. Thus, the user may accept an early fixture design to reduce program run time, or obtain the global optimum by letting the program run to completion. The algorithm employs branch-and-bound pruning to restrict the enumeration and quickly find the global optimum; the user can reduce the effect of this pruning as desired. The details of the algorithm used for this integration may be found in [1,2].

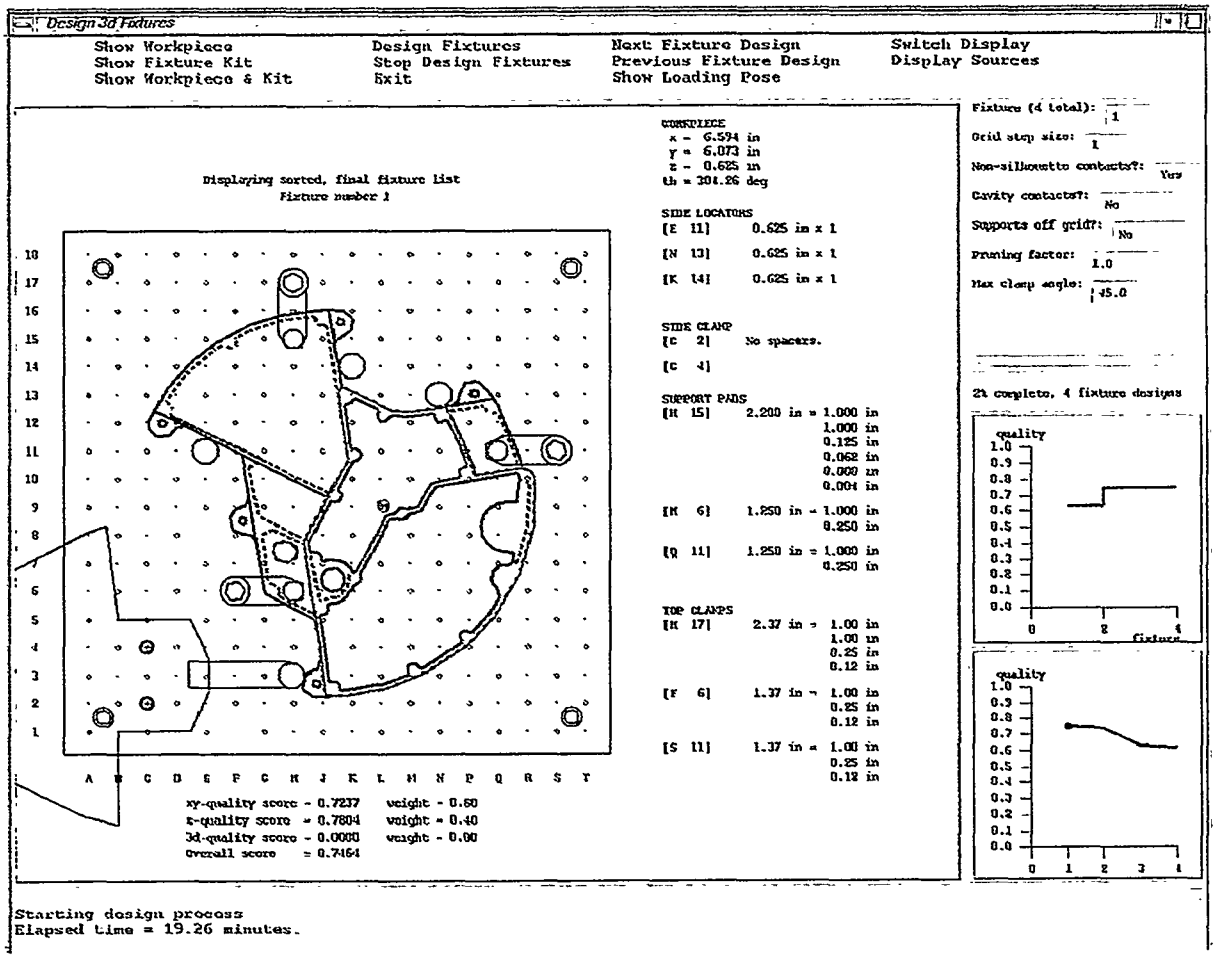

Figure 2.4. Fixture-design for fixturing the cast housing and assembling the electrical components, shown with the design tool user interface.

2.5 Automated Assembly Planning

The system used to analyze the fixture designs and assembly was Sandia's Archimedes 3.0© Automated Assembly Analysis System [3,5,7]. The Archimedes 3.0© System is a Sandia developed, constraint-based interactive assembly planning software tool used to plan, optimize, simulate, visualize, and document sequences of assembly. Given a CAD model of the product,

the program automatically finds part-to-part contacts, generates collision-free insertion motions, and chooses assembly order. The engineer specifies a quality metric in terms of application-specific costs for standard assembly process steps, such as part insertion, fastening, and subassembly inversion. Combined with an engineer's knowledge of application-specific assembly process requirements, Archimedes 3.0© allows systematic exploration of the space of possible assembly sequences. The engineer uses a simple graphic interface to place constraints on the valid assembly sequences, such as defining subassemblies, requiring that certain parts be placed consecutively with or before other parts, declaring preferred directions, etc. Archimedes 3.0© considers thousands of combinations of ordering and operation choices in its search for the best assembly sequences and ranks the valid sequences by the quality metric. Graphic visualization enables the engineer to easily identify process requirements to add as sequence constraints. Planning is fast, enabling an iterative constrain-plan-view-constrain cycle. For some restricted classes 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.

The approach taken in assembly planning is critical to the design, implementation, and performance of a user constraint system. It especially affects special-purpose routines for efficiency. Archimedes 3.0© generates two-handed monotone assembly sequences in reverse, starting from the more highly constrained, fully assembled state. The search space is an AND/OR graph of subassembly states and the operations used to construct them from smaller subassemblies. The strategy is designed to generate a first plan as quickly as possible, like a depth-first search, but to avoid being caught by bad early decisions as a depth-first search would. This is critical to achieve the desired view-constrain-replan cycle of interaction. During each pass of the search algorithm, a single assembly sequence is generated, making random choices of operations to construct each subassembly. The first time any subassembly is visited, only a single operation is generated to construct it, and the known subassemblies of that operation are then visited. Bounds on quality measures for each subassembly and operation are stored and propagated in the AND/OR graph as they are generated. This allows useless search paths to be identified and pruned, and an optimal plan to be identified when it becomes available. The same algorithm functions as an any-time algorithm to optimize the assembly sequence when the user requests. The reader is referred to Appendices A and B for a broader overview Archimedes 3.0© and the constraint framework.

2.6 The Integration

The integration of HoldFast™ with Archimedes 3.0© resulted in a two-phased approach. In the first phase, HoldFast™ was upgraded to use ACIS® 3D objects. Specific tasks required for the ACIS® upgrade included converting the HoldFast™ geometry from 2-1/2-D to 3D and developing interface capabilities between HoldFast™ and ACIS®. HoldFast™ previously used 2-1/2D geometry to interrogate an object's geometry. Those functions were rewritten in order to be compatible with the ACIS® 3D representation. Also, since HoldFast™ was initially written in LISP and ACIS® is written in C++, interface routines had to be developed to allow a seamless transition between HoldFast™ and ACIS®. When the conversion was complete, the actual

integration of the two software packages was made. Before the housing was fabricated and machine-finished, CAD designs of it, the pallet design, and additional assembly components were loaded into the Archimedes 3.0© System for analysis. Using the Archimedes 3.0© constraint framework, the fixture was declared as a base component. This meant that the fixture is the first set of parts to be assembled and will remain stationary throughout the assembly analysis. Figures 2.5 and 2.6 illustrate the use of a socket wrench and phillips-screwdriver. The Archimedes 3.0© System tested for feasibility of tool use during the pallet assembly and assembly of the components. In this particular case, the Archimedes 3.0© System identified four potential design flaws in the housing that prohibited the use of the screwdriver for attaching the cover plate. The design was corrected and a new assembly plan was generated. The assembly planner not only generated assembly plans for the assembly, but it was used to generate assembly plans for the fixture as well. The plan was captured in the form of an m-peg video for use in demonstrating manufacturability.

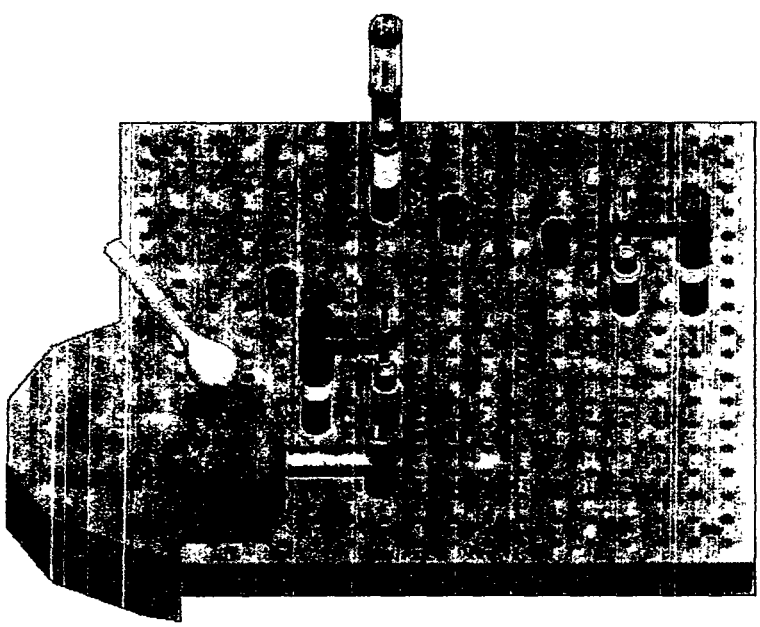

Figure 2.5. Illustration of a ratchet tool used during the assembly operation.

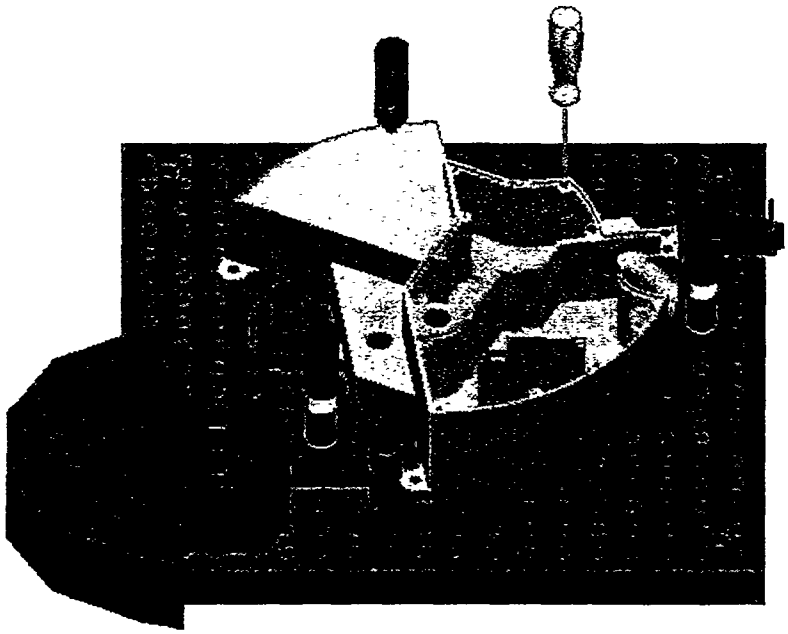

Figure 2.6. Illustration of a screwdriver tool use during the assembly operation.

2.7 Conclusions and Future Work

An integrated solution for combining automated fixture design tools and automated assembly planning has been presented. The method starts with three-dimensional CAD descriptions of an assembly whose assembly tasks require a fixture to hold the base part. It then combines algorithms that automatically design assembly pallets to hold the base part with algorithms that automatically generate assembly sequences. The designed fixtures rigidly constrain and locate the part, obey task constraints, are robust to part shape variations, are easy to load, and are economical to produce. The combined algorithms guarantee that the fixture will hold the base part without interfering with any of the assembly operations. What is lacking in this framework is the ability for the assembly planner to automatically provide feedback to the fixture planner. Future work is aimed at providing a more complete integration of the planners.

References

1. R. C. Brost and K. Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. In *IEEE Transactions on Robotics and Automation*, 12(1):31-46, February 1996.
2. R. C. Brost and R. R. Peters. Automatic design of 3d fixtures and assembly pallets. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 495-502, 1996.
3. T.L. Calton. Advancing design-for-assembly: the next generation in assembly planning. In *Proc. of 3rd Intl. Symposium on Assembly and Task Planning*, 1999.
4. E. G. Hoffman. Modular fixturing. *Manufacturing technology press.* Lake Geneva, Wisconsin, 1987.
5. R.E. Jones, R.H. Wilson, and T.L. Calton. On constraints in assembly planning. In *IEEE Intl. Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 849-863, 1998.
6. K. Quinter. Qu-co modular fixturing system. Qu-Co, Inc., Union, Ohio. 1993.
7. R. H. Wilson. Geometric reasoning about assembly tools. *SAND95-2423*, 1997.

3

Automated Assembly Analysis for Human Assisted Robotics and Automation

3.1 Introduction

For robots to work harmoniously within an integrated environment with humans the robots must have cooperative operational skills. For example, in a human only environment, humans may tolerate collisions with one another if they did not cause much pain. This level of tolerance may or may not apply to robot-human environments. Humans expect that robots will be able to operate and navigate in their environments without collisions or interference. The ability to accomplish this is linked to the sensing capabilities available. Current work in the field of cooperative automation has shown the effectiveness of humans and machines directly interacting to perform tasks. To continue to advance this area of robotics, effective means need to be developed to allow natural ways for people to communicate and cooperate with robots just as they do with one another. Computer vision, using low cost devices, makes it possible for a human to use body motions or any convenient object-as digital input devices.

One effective means for a human to communicate with a machine is to use body gestures similar to those used when communicating with another person. These gestures include such things as pointing to indicate a destination. This chapter describes methods that were implemented for extracting static pointing gesture information from Sandia's 3D Video Motion Detection System. The chapter also describes the static pointing gesture interpretation algorithms used for robot-assisted operations. The static pointing gestures were used to indicate predefined portions to which a robot should move, and perform the appropriate task at that destination.

The next section provides an overview of previous research in the field of static and dynamic gesture recognition. Section 3.3 briefly describes Sandia's 3D Video Motion Detection System used to track a human operator near or within the moving workspace of a robot. Section 3.4 summarizes the gesture recognition approach. Finally, Section 3.5 concludes and gives directions for future work.

3.2 Previous Work

Previous efforts and ongoing work in the field of cooperative automation has shown the effectiveness of humans and machines directly interacting to perform tasks. Cooperative automation places the human in the field with the robot, and therefore the human must communicate with the robot using methods other than the traditional keyboard and mouse. To continue to advance this area of robotics, effective means need to be developed to allow natural ways for people to communicate and cooperate with robots just as they do with one another.

Natural interaction consists of two major parts: natural language understanding and visual observation of the user's body pose and gestures. The ability of a machine to recognize and understand both of these modes of communication provides a powerful communication tool. While a significant amount of work has been conducted in the area of computer recognition of natural language, the recognition of human intent through body gestures is still in the early stages of development.

Virtually all of the existing gesture recognition and interpretation research involves 2-D vision (the use of one camera). The analysis techniques utilize image analysis to extract body features based on feature extraction, color and texture. Some techniques employ fairly sophisticated specialized routines that take advantage of knowing the image is that of a human. However, because only 2-D data is captured the systems are limited to 2D-gesture recognition. Further, many existing approaches require post-processing (not real-time).

There are two analysis techniques which are widely used for multi-modal pattern recognition: Fourier descriptors and Hidden Mark Models (HMMs); however, they are both restricted to 2-D. Fourier descriptors have been specifically developed for 2-D gestures. This work has shown that the gesture is not strictly constrained in space (i.e., a box gesture in space can be arbitrarily oriented in space, doesn't have to lie along x, y axes).

Hidden Markov Models have been used prominently and successfully in speech recognition and, more recently, in handwriting analysis. Consequently, they seem ideal for visual recognition of complex, structure hand and body gestures. While the continuous speech recognition community adopted HMMs many years ago, the vision community is just now accepting the

techniques. An early effort by Yamato uses discrete HMMs to recognize image sequences of six different tennis strokes among three subjects. (Yamato's work is described in [4].) This experiment is significant because it uses a 25x25 pixel quantized sub-sampled camera image as a feature vector. Even with such low-level information, the model can learn the set of motions and recognize them with respectable accuracy. Darrell and Pentland presented a dynamic time warping in [2], a technique using HMMs to match the interpolated responses of several learned image templates. While Baum-Welch re-estimation, [1], was not implemented, this study shows the continuous gesture recognition capabilities of HMMs by recognizing gesture sequences. Recently, in [3] Wilson and Bobick explored incorporating multiple representations in HMM frameworks.

3.3 3D Video Motion Detection

At Sandia a 3-Dimesional Video Motion Detection System (3DVMD) has been deployed to track the human operator near and within the moving workspace of the robot. Several cameras are deployed from the structure that surrounds the robot and monitor an area about twice the size of the robot's reachable workspace. The 3DVMD system continuously monitors for activity in discrete 3-D sub-volumes (about 1" cubes or voxels) which make up the monitored space. Figure 3.1 shows the detection of a human operator as indicated by the colored voxels that are occupied by that operator. The data from the occupied voxels is analyzed to report the person's location to the robot in real-time. The centroid of the occupied voxels is calculated, and this centroid is transmitted to the robotic system as the operator's location. The relationship of occupied voxels can be further analyzed to determine the pose of the operator. As shown in Figure 3.1, the voxel size is sufficiently small such that the person's extended hands, legs and head can be distinguished from the torso.

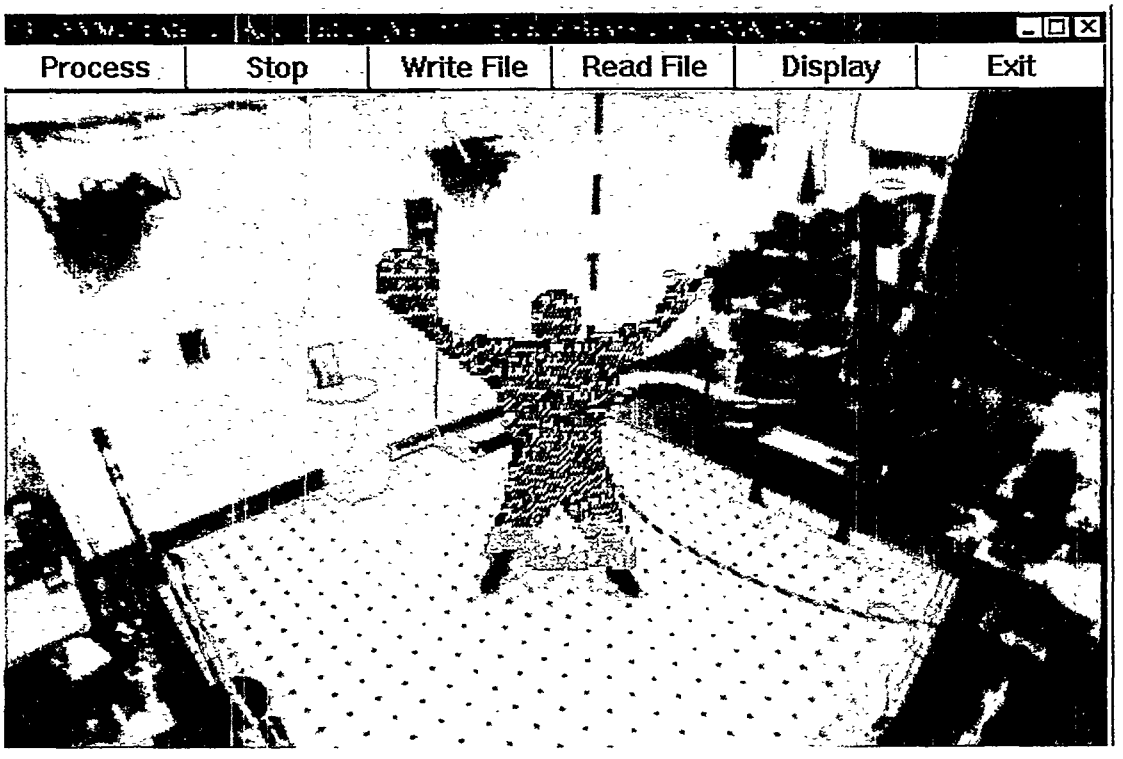

Figure 3.1. 3DVMD System detects the operator as indicated by the colored voxels.

3.4 Gesture Recognition Approach

A real-time data interpretation approach was implemented. First, the software associated with the 3DVMD system continuously calculates the centroid of the occupied voxels. The centroid is used to indicate the human operator's location. Next, the relationship of the voxels is analyzed to determine if a gesture is occurring. A vertical cylindrical sub-volume is applied around the centroid. If any voxels exist outside this cylinder, then a gesture is occurring. The purpose of the cylindrical sub-volume is to limit the gesture space such that the normal position of the legs (particularly the feet) and head do not trigger gestures. The cylinder properties (radius, length) can be adjusted to take into account specific body types of individual users. However, a single set of properties can be used for a wide variety of operators. A specific voxel density is required for the 3DVMD system to indicate that a detected object is an operator. This prohibits the cylindrical gesture volume from being applied to smaller objects, and potentially causing an erroneous indication of a gesture.

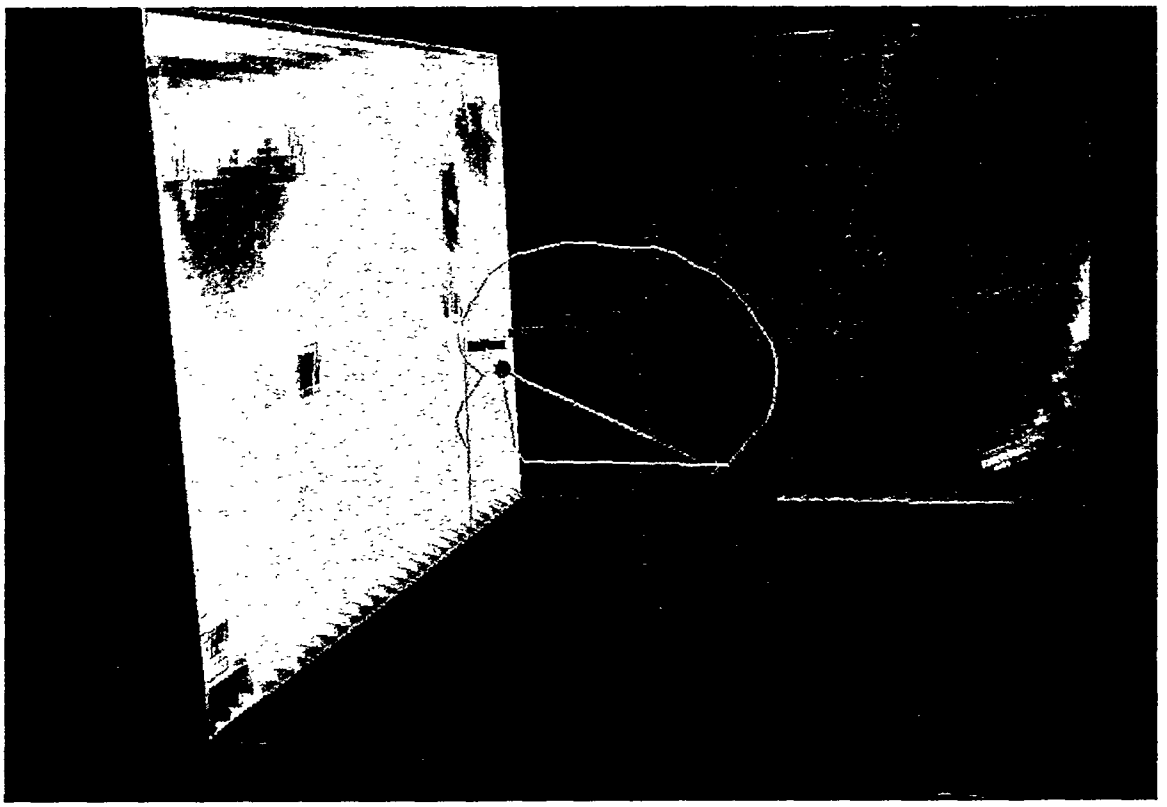

Figure 3.2. 3/4-inch resolution Image depicting the trace of a motion captured by the 3DVMD System.

When a gesture is detected, a vector is created by connecting the centroid and the outer most voxel that exceeds the cylindrical limit. The 3DVMD system reports this data to the supervisory control system in real-time. The data packet contains the centroid of the occupied voxels, a flag that indicates if a gesture is occurring, and the gesture vector in robot workspace coordinates. The supervisor then interprets the gesture vector and projects it into the robotic workspace. The destination to which the operator is pointing is determined, and the supervisor commands the robot on the particular task to be performed.

3.5 Conclusions and Future Work

The research focused on extracting static gesture information from the 3DVMD data. Several predefined positions were defined within the robotic workcell. By pointing to those

destinations, the operator could command the robot to move to that destination, and perform the task described at that position.

Future work in this area should encompass the development of an instruction language for robots using dynamic hand and arm gesture and the development of robust and real-time recognition and interpretation algorithms.

The following technical issues should addressed to create a dynamic gesture recognition system:

(1) Defining the Language – a set of dynamic gestures for human / robot communication needs to be defined.

(2) Obtaining Data – a sensor system for capturing human gestures in real-time needs to be established.

(3) Segmenting Data – algorithms for isolating the data segment which represents the data. Data segmentation involves determining which portion of the captured gesture represents the intended command to the machine, and which portion is just spurious movement of the hands.

(4) Recognition Algorithms – analysis algorithms that recognize the gesture from the segmented data. Algorithms should be developed for processing the segmented data to identify the gesture. Our initial research will investigate fitting a Fourier series to the decomposed gesture data. Decomposition involves breaking the gesture data into three components: x, y, and z versus path length. A Fourier series should be fitted to each of these components. The weighting functions for each of the series elements are descriptors. Further research should be conducted on how to analyze these descriptors to reliably recognize the gesture.

(5) Integration with Speech – algorithms which combine speech input and recognized gestures to bring context to the gesture. For example, a "go here" command would indicate that the gesture is indicating a destination in the work environment, whereas a "go this way" combined with the same gesture would indicate that the robot should move in the general direction indicated as long as the gesture continues.

(6) Communication with robot – command a robot's operation based on the combined input of speech and gesture. Gestures should be captured, analyzed and communicated to the robotic system in real-time.

References

1. L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov Processes. *Inequalities*, 3:1-8, 1972.
2. T. Darrell and A. Pentland. Space-time gestures. *CVPR*, p. 335-340, 1993.
3. A. Wilson and A. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. On Comp. Vis.*, Nov. 1995.
4. J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov Models. In *Proc. IEEE Int'l ICCV*, p379-385, 1992.

Intentionally Left Blank.

Closing

This manuscript serves as the technical report required by Sandia National Laboratories Laboratory Directed Research and Development (LDRD) Program for the project entitled "Ergonomics in Life Cycle Assembly Processes". The LDRD Program financially supported the entire project and this manuscript summarizes the technical achievements made throughout the project's two-year history. The work conducted under the purview of the LDRD focused primarily on automating analysis and planning techniques for human assisted assembly processes. However, to facilitate solutions to additional needs from within the Sensing and Intelligent Controls Investment Area, additional goals were added during the second year of the LDRD. Of particular interest was the need to more tightly couple the enabling of human skills in cooperative automation with automated assembly analysis. It was hoped that such a coupling would provide improved intelligent controls for creating cooperative automated assistants for a human's logistical capabilities and safety in hazardous operating environments.

In support of the project's original objectives, Sandia's Automated Assembly Analysis System, Archimedes 3.0©, was used as the foundation for integrating algorithms that automatically generate assembly and disassembly human assisted assembly processes. Initialy, a general framework for realizing these issues has been developed by integrating commercially available human figure modeling software packages Archimedes 3.0©. What differentiates this methodology from other approaches is the capability to automatically analyze complete assembly and disassembly by representing and reasoning about human related geometric issues and human factors for a variety of hand tools used in assembly operations. Secondly, an integrated solution for combining automated fixture design tools and automated assembly planning. This initiative focused on integrating Sandia's Fixture Design Software Package, HoldFast™, with Archimedes 3.0©. The combined algorithms guarantee the fixture will hold the base part without interfering with any of the assembly operations.

In alignment with the goals outlined in the re-direct of the LDRD, efforts were combined with equal amounts from three separate LDRDs ("Enabling Human Skills with Cooperative Automation", "Adaptive 3D Sensing", and "Volumetric Video Motion Sensing for Unobtrusive Human-Computer Interactions"). The major technological achievement under this project's purview was the development of vision algorithms to recognize (static) human gestures for use in robot communication and cooperation. Methods that were developed for extracting static pointing gesture information provided by a human from Sandia's 3D Video Motion Detection System for use in cooperative automation environments.

Intentionally Left Blank.

Appendix A

Assembly Planning Constraint Framework

A.1 Introduction

Constraints on assembly plans vary depending on product, assembly facility, assembly volume, and come from a wide variety of choices. Design requirements, part and tool accessibility, assembly line and workcell layout, requirements of special operations, and even supplier relationships can drive the choice of a feasible or preferred assembly sequence. Computer-aided assembly planning systems promise to help product and assembly system designers to manage this complexity and choose good assembly sequences. However, there so many types of assembly constraints that it is impractical for a single program to encode them all. A practical assembly planning system must have the ability to manage assembly constraints in a general way to determine how they impact the choice of assembly sequence.

This appendix describes the principles and implementation of a framework that supports a wide variety of user-specified constraints for Sandia's Automated Assembly Analysis System, Archimedes 3.0©. In this framework, the user chooses constraints from a library of standard constraint types, with a simple graphical interface for defining the specifics of the constraint.

Constraints are implemented as *filters* that either accept or reject assembly operations proposed by the planner. Any constraint that can be encoded as a filter can be added to the constraint library in a straightforward way. Some constraints are supplemented with special purpose modifications to the planner's algorithms for greater efficiency.

In the next section, the constraint framework is placed in the context of previous research on constraints and assembly planning. Section A.3 addresses manufacturing plan selection criteria as a whole while Section A.4 delves deeper into some of the issues that arise in implementing the constraint system effectively and concludes the appendix.

A.2 Previous Work

Assembly planning systems that allow user-defined constraints have generally been of two types. In the first type [5,6], users must specify all constraints before the long process of plan generation begins. In reality, this is rarely practical; the user finds it very difficult to list all constraints on assembly ordering until some possible plans are considered. In the second type of system (e.g. [2]), a large space of plans is first generated, and then undesirable operations and states are pruned interactively by the user. However, as assemblies become moderately complex the space of plans quickly becomes too large to edit or even generate. The methods implemented in the Archimedes 3.0© System solve this problem interactively by generating one or a small number of assembly plans that satisfy the current set of constraints, and then allowing the user to impose additional constraints. Users can continue to add constraints and replan until a satisfactory plan is identified. This approach is foreshadowed in a much more limited form by [16].

Previous efforts to incorporate a comprehensive set of user constraints in assembly planners were based on liaison precedence relations. Precedence relations specify logical combinations of part connections that must be established either before or after others. Precedence relations were pioneered by Bourjault [4] and greatly extended by DeFazio and Whitney [5]. Wolter *et al* [18] analyze the expressive power of precedence relations in detail. Precedence relations are quite powerful, but they can be very difficult to write correctly or understand as a user of an assembly planner. In Archimedes 3.0© consideration was given to the idea of translating all constraints into precedence relations internally, but instead a procedural approach was adopted for reasons of efficiency and simplicity of implementation. The Archimedes 3.0© System demonstrates that an assembly planning system can achieve comprehensive constraint coverage while maintaining the advantages of a procedural representation.

A.3 Manufacturing Plan Selection Criteria

There are basically two categories of manufacturing plan selection criteria: *assembly issues per se* and *planning issues.* Assembly issues arise directly from physical constraints and manufacturing concerns, such as fixturing, manipulability, or issues of assembly line layout.

Closely related are quality measures arising from a need to optimize one of the following basic aspects of assembly:

1) assembly cost;
2) time;
3) performance/reliability.

In contrast, planning issues are those that arise from the assembly process itself, such as simplifying assumptions and limitations of planning systems, or the need to handle large numbers of possible sequences practically.

The constraints on assembly plans vary depending on product, assembly facility, assembly volume, and many other factors. Assembly costs and other measures to optimize vary just as widely. Design requirements, part and tool accessibility, assembly line and workcell layout, requirements of special operations, and even supplier relationships can determine which orders of assembly are feasible. Among feasible orderings, a similarly diverse set of quality measures determines which is preferred or optimal. Together, the set of hard constraints and quality measures comprise a complex set of criteria that real assembly plans must satisfy. In Archimedes 3.0©, users specify manufacturing plan selection criteria in the form of constraints and quality measures. The constraints are implemented along the lines of the "assembly issues" classification. The Archimedes 3.0© constraint framework is discussed in detail in the next section.

A.4 Assembly Planning Constraints

The constraint framework described here was required for the Archimedes assembly planning system [10]. That system takes CAD data for a product as input and automatically determines geometrically feasible sequences of motions to assemble the product from its parts. However, geometry is only a small part of assembly planning. In [9], a survey of constraints that have appeared in the assembly planning literature was conducted, in hopes of adding many of those constraints to the Archimedes system. Two types of assembly constraints were identified. *Strategic* constraints apply to the entire assembly and its plan, while *tactical* constraints only apply to certain subsets of the parts. The result of the survey is an extensible library of simple but useful constraints that enable, highly interactive mode of assembly planning. Not all of those constraints, however, were chosen for implementation. The constraints that are implemented in the Archimedes 3.0© System were selected the following reasons:

User Friendly: Each constraint can be described simply in terms familiar to the user, has straight-forward effects, and combines with others in a very predictable way.

Maintainable: Each constraint simply provides a filter procedure that disallows some assembly operations. The filter implementations are completely independent, allowing new constraint types to be added easily.

Efficient: Because the filters are procedures, they can be implemented in the most efficient way for that constraint. They test individual assembly operations, so they are compatible

with standard state-space search and optimization methods. Furthermore, special-purpose methods can be added to improve efficiency.

The strategic constraints currently implemented in the Archimedes 3.0© System are flags for the planner. However, in theory there is no real difference between strategic and tactical constraints since tactical constraints can usually be applied to the entire assembly, and strategic constraints can always be limited to a subset of the parts. For instance, in [9], the REQ_PATHS_AXIAL constraint is identified as strategic, is also very useful applied to a subset of the assembly. Implementing it as a tactical constraint allowed both uses, was simpler to implement, and caused no loss in efficiency. Table A.1 lists the constraint types implemented in the Archimedes 3.0© System.

Constraint Name	**Purpose**	**Scope**
REQ_CONNECT*	Requires that every subassembly in the plan be connected. This common constraint is implicit in cut-set methods such as [1,8],	strategic
REQ_LINEAR*	Requires that parts be inserted one at a time [1,18]. This a common constraint as well as a common limitations in other planning systems.	strategic
REQ_VERTICAL*	Requires that all parts be placed directly downward, but unlike REQ_PATHS_AXIAL, allows the assembly to be reoriented so that any given action is considered "downward." This constraint is very useful when planning assembly with standard SCARA-type robots, where insertions must be vertical but reorientations should be minimized.	strategic
PRH_STATE	Does not allow the assembly to enter a given state [2].	tactical
PRH_SUBASSY	Prohibits use of a certain subassembly containing certain part combinations. For example, one may need to avoid a hard-to-fixture arrangement [6] containing a set of key parts.	tactical
REQ_CLUSTER	Requires that a set of parts be added to the assembly consecutively, i.e., without interruption by other parts [3].	tactical
REQ_FASTENER*	Requires that certain parts be treated as fasteners for others parts [10,13]. The fasteners must be placed immediately after the fastened parts are mated.	tactical
REQ_LINEAR_CLUSTER	A combination of REQ_CLUSTER and REQ_LINEAR_SUBSET.	tactical
REQ_LINEAR_PARTS	Requires that a given set of parts be assembled linearly when they are being mated with any of another set of parts.	tactical

Constraint Name	Purpose	Scope
REQ_ORDER_FIRST	Requires that the assembly plan start with a given part, such as a "chassis" [11], or a set of parts.	tactical
REQ_ORDER_LAST*	Requires that a specific part or set of parts be placed last in the assembly plan.	tactical
REQ_ORDER_LIAISON	Requires some ordering between two or more liaison creations; typically stated in a Boolean form such as $1 \geq (2$ and 3), or as a set of such Boolean statements involving many liaisons [5]. This is a common and powerful type of constraint, analyzed in [19].	tactical
REQ_ORDER_PART	Requires an ordering between particular part insertions.	tactical
REQ_PART_SPECIAL	Any special-purpose part constraint, such as those dealing with liquids, springs, snapfit parts, etc.	tactical
REQ_PATHS_AXIAL	Requires that each assembly action be along one of the six coordinate directions of a given coordinate system, or a selected subset of these six directions. Common special cases are *uniaxial* constraints (allowing motions in either direction along one axis) [12] and *unidirectional* constraints [11]. This constraint is a limitation of many assembly planning systems.	tactical
REQ_STACK	Specifies a set of parts to be assembled one at a time in a given direction.	tactical
REQ_STAT	Requires a set of parts to be in the stationary subassembly when mated with any of another set.	tactical
REQ_SUBASSY*	Requires that a particular subassembly be used in the plan [1,6].	tactical
REQ_SUBASSY_ WHOLE*	The same as REQ_SUBASSY but tells the planner in addition not generate a plan to construct the subassembly.	tactical
REQ_SUBSEQ	Requires that a particular assembly subsequence be used somewhere in the plan. This might be invoked because the sequence is particularly efficient or reliable. The front-fill then back-fill subsequences of [2] are relatively complex examples.	tactical
REQ_TOOL*	Requires that a collision-free placement of a given tool use-volume must exist in the assembly during a certain operation. See [17] for more details.	tactical

Table A.1. Constraints implemented in the Archimedes 3.0© System. Those marked by * have special-purpose routines for efficiency.

In the Archimedes 3.0© System, filters are an instance of *generate-and-test*, a standard paradigm in artificial intelligence [14] as well as in assembly planning [6, 8]. Attaching special-purpose methods to constraint tests is also a well-known efficiency technique [14], although the methods used to accomplish this were novel.

A filter is nothing more than a simple procedure that accepts or rejects assembly operations. During planning, each proposed assembly operation is passed to the constraint's *filter function*, which returns true or false depending on whether the operation satisfies the constraint or not.

Only an operation that satisfies all current constraints is feasible. For instance, consider an operation placing subassembly S_1 into subassembly S_2[3]. The filter function of a REQ_SUBASSY constraint with part set P will return true if and only if

$$[P \subseteq S_1] \vee [P \subseteq S_2] \vee [(S_1 \cup S_2) \subseteq P] \vee [S_1 \cup S_2) \cap P = 0].$$

In other words, the operation satisfies the constraint if it keeps the parts in P together, if only parts P are involved, or if no parts in P are involved.

As a standard interface to all constraints, the filter function provides a number of benefits. First and foremost, it makes the implementation of each constraint type independent. Interactions between constraints need not be considered, and each constraint can be implemented in its most straightforward and efficient way. This becomes crucial as the number of constraint types grows. In addition, constraints can vary in the data associated with them, their instantiation routines, various debugging outputs, and so on.

For use in a standard state-space search method (such as generating an AND/OR graph for the assembly), it is important that the filter functions take as input single assembly operations, rather than more complex information such as a sequence of operations. In a state-space search, a given operation appears only once in the state graph and is either present or not. Hence, its feasibility cannot depend on operations that come before or after it. In the Archimedes 3.0© framework, a number of constraints (e.g. REQ-SUBASSY above) must be implemented in a less natural way to apply to single operations than would otherwise be necessary.

Filter functions are flexible enough that a large subset of the constraints implemented in Archimedes 3.0©, plus some additional ones that users requested are implemented. The flexibility is further demonstrated by the REQ_TOOL constraint, which encodes tool accessibility constraints for various hand and robotic tools [17] within the framework.

Although filter functions themselves are usually quite fast, the generate-and-test abstraction can sometimes lead to an inefficient planning process overall. This is particularly true when many dead-ends appear in the search space or when a large number of assembly operations are

---

[3] An operation will typically have other specifications, such as a mating trajectory and perhaps an assembly orientation, but these are not relevant to REQ_SUBASSY.

generated, but few satisfy the constraints. In many cases, special purpose routines can increase efficiency dramatically. The constraint types for which special purpose routines have implemented are indicated with an asterisk (*) in Table A.1.

3.5.1 Interaction

In experiments with product designers and assembly process engineers, it was found that a high level of interactivity is critical to the successful application of the assembly planner. Usually the designer cannot list all the constraints on assembly order at the start of the planning session. However, many of these constraints become "obvious" when the system graphically illustrates a plan that violates them. Seeing a violation, adding a constraint to remove that violation, and then replanning becomes the main cycle of interaction in the system. In this way, the assembly planner aids constraint discovery and management as well as plan generation and optimization.

Note, however, that placing a new constraint is very different from ruling out a certain operation, as performed in some previous systems such as [2]. Although a single operation demonstrates the need for a constraint, placing the constraint prohibits similar actions from occurring in many different operations, and hence limits the allowable plans far more than prohibiting a single operation. In the best case (and in many practical cases), the constraint encodes the manufacturing constraint exactly.

This plan-view-constrain replan cycle requires that replanning be performed at interactive speeds. In the Archimedes 3.0© System, a first assembly plan for a product can usually be found in a few minutes [10]. However, the most expensive part of planning is ensuring that part insertions are collision-free. By saving collision-detection information, replanning usually requires 10 to 20 seconds for assemblies of up to 100 parts.

There is, of course, no guarantee that all of the constraints the user has instantiated can be satisfied by a single plan. In this case, the planner fails and enters a "debug" mode that helps the user to determine the cause of the failure. If the constraints are all real, then a problem with the product design may be indicated. In most cases, some constraints can be adjusted to allow planning to succeed. When there are inaccuracies or inconsistencies in the product CAD data, planning can fail before the user has entered any constraints. The debug mode also supports finding such problems, and certain problems can be fixed within Archimedes 3.0

Note that if constraints are entered inaccurately, they may over-constrain the choice of plan and rule out some plans incorrectly. However, if some plans still satisfy the entered constraints, then the error may go unnoticed. This is a difficult problem to solve in a system allowing user specification of constraints.

After all known manufacturing constraints have been entered, the user can then ask for an optimal plan, according to user-specified costs of certain standard operations. In some cases, additional unstated constraints will be violated and discovered as the planner looks through a large space of plans to find the best. In this case, the new constraints must be added and the cycle repeats.

References

1. T. E. Abell, G. P. Amblard, D. F. Baldwin, T. L. De Fazio, M.-C. M. Lui, and D. E. Whitney. Computer aids for finding, representing, choosing amongst, and evaluating the assembly sequences of mechanical products. In [7], pages 383-435.

2. 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.

3. N. Boneschanscher and C. J. M. Heemskerk. 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

4. 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.

5. 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, 1987. Errata in RA-4(6):705-708.

6. J. M. Henrioud and A. Bourjault. LEGA: a computer-aided generator of assembly plans. In [7], pages 191, 215.

7. L. S. Homem de Mello and S. Lee, editors. *Computer-Aided Mechanical Assembly Planning.* Kluwer, 1991.

8. 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.

9. R. E. Jones and R. H. Wilson. A survey of constraints in assembly planning. In *Proc. IEEE Intl. Conf. On Robotics and Automation*, pages 1525-32, 1996.

10. 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 Intl. Conf. on Robotics and Automation*, pages 3361-8, 2996.

11. H. Ko and K. Lee. Automatic assembling procedure generation from mating conditions. *Computer Aided Design*, 19(1):3-10, 1987.

12. E. Kroll, E. Lenz, and J. R. Wolberg. Rule-based generation of exploded-views and assembly sequences. *Artificial Intelligence in Engineering, Design, and Manufacturing*, 3(3): 143-155, 1989.

13. J. M. Miller and R. L. Hoffman. Automatic assembly planning with fasteners. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 69-74, 1989.

14. N. J. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, 1980.

15. B. Romney. Atlas: An automatic assembly sequencing and fixturing system. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling*: Theory and Practice, pages 397-415. Springer-Verlag, 1997.

16. R. H. Wilson. Minimizing user queries in interactive assembly planning. *IEEE Trans. on Robotics and Automation*, 11(2): 308-312, 1995.

17. R. H. Wilson. Geometric Reasoning about assembly tools. Technical Report SAND95-2423, Sandia National Labs, 1996.

18. J. D. Wolter, S. On the automatic generation of plans for mechanical assembly,"Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1988. *IEEE Trans.*

19. J. D. Wolter, S. Chakrabarty, and J. Tsao. Mating constraint languages for assembly sequence planning. *IEEE Trans. On Robotics and Automation.* To appear.

20. 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.

Appendix B

Automated Assembly Analysis

Archimedes 3.0© is a constraint-based, interactive assembly planning software system used to plan, optimize, simulate, and document sequences of assembly. Given a computer-aided (CAD) model (ACIS® representation) of the product, the program automatically finds part-to-part contacts, generates collision-free insertion motions, and chooses assembly order. The engineer specifies a quality metric in terms of application-specific costs for standard assembly process steps, such as part insertion, fastening, and subassembly inversion. Combined with an engineer's knowledge of application-specific assembly process requirements, such as defining subassemblies, requiring that certain parts be placed consecutively with or before other parts, declaring preferred directions, etc., Archimedes 3.0© allows systematic exploration of the space of possible assembly sequences. The system considers thousands of combinations of ordering and operation choices in its search for the best assembly sequences and ranks the valid sequences by a quality metric. Graphical visualization enables the engineer to easily identify process requirements to add as sequence constraints. Planning is fast, enabling an iterative constrain-plan-view-constrain cycle. This constraint-based interactive approach to automated assembly analysis and planning has

proved extremely valuable in demonstrating manufacturability of a product when starting with the original CAD data.

This chapter provides a detailed view of the Archimedes 3.0© System. For completeness, the assembly planning approach provided in Chapters 1 and 2 is again provided in Section B.1. Section B.2 describes the graphical user interface. Section B.3 covers situations when the planner might fail and describes how to recover. The efficiency of the system is provided in Section B.4 while Section B.5 concludes the chapter with an overview of the Archimedes 3.0© implementation.

B.1 Assembly Planning Approach

The approach taken to assembly planning is obviously critical to the design, implementation, and performance of a user constraint system. It especially affects special-purpose routines for efficiency. The constraint system described in the previous section was added to the Archimedes 3.0© mechanical assembly planner [2]. The Archimedes 3.0© System consists of four main elements: a user interface, a constraint system, a search engine, and an animation module. Archimedes 3.0© generates two-handed monotone assembly sequences in reverse, starting from the more highly constrained, fully assembled state. This is a standard technique in assembly planning. The search space is an AND/OR graph of subassembly states and operations to construct them from smaller subassemblies. The planner uses a non-directional blocking graph, NDBG, of each subassembly [4] to efficiently determine assembly operations that might be performed to construct that subassembly, then checks these operations for geometric collisions, which is essentially a built-in filter. Operations are then checked against the list of user constraints.

The search strategy is carefully tuned to generate a first plan as quickly as possible in the domain of mechanical assembly. This is critical to achieve the desired plan-view-constrain-replan cycle of interaction. The search algorithm is not the focus of this report, and space allows only a cursory description. An AND/OR version of iterative sampling [3] is performed: during each pass of the algorithm, a single assembly sequence is generated, making random choices of operations to construct each subassembly. The first time any subassembly is visited, only a single operation is generated to construct it, and the known subassemblies of that operation are then visited. Bounds on quality measures for each subassembly and operation are stored and propagated in the AND/OR graph as they are generated. This allows useless search paths to be identified and pruned and an optimal plan to be identified when it is known. The strategy is designed to quickly reach a first solution, like a depth-first search, but to avoid getting caught by bad early decisions as a depth-first search would. The same algorithm functions as an any-time algorithm to optimize the assembly sequence when the user requests.

B.2 User Interface

The user interface is critical to effectiveness and user acceptance of an interactive planning system. The constraints must be easy to understand, define, and manage. In this section, features of the Archimedes 3.0© user interface that are important to the success of the constraint system are described. See [1] for a better understanding of the Archimedes 3.0© constraint interface.

Figure B.1 shows the main Archimedes 3.0© user interface. The upper portion of the window on the left hand side serves and the main control window and displays the program's current status, displays any diagnostic output, and allows pausing or aborting any computation. The lower portion of the window on the left hand side serves as the constraints and control panel and displays user-defined constraints and overrides. The window on the right hand side provides graphical output and part/subassembly selection.

Constraints are added by clicking on the "Add" button at the bottom, which brings up a sequence of menus and questions that let the user pick a constraint type and specify the particulars of the desired constraint. (See Figure B.2.) Each constraint requires the user to select one or more parts of a "controlled" set in the graphic window, such as the parts making up a subassembly. An auxiliary window provides a list of named subassemblies to facilitate selection of larger sets of parts. (See Figure B.3.) The user can give each constraint a name and descriptive comment. Some simple checks are performed to catch certain obvious inconsistencies within and between constraints.

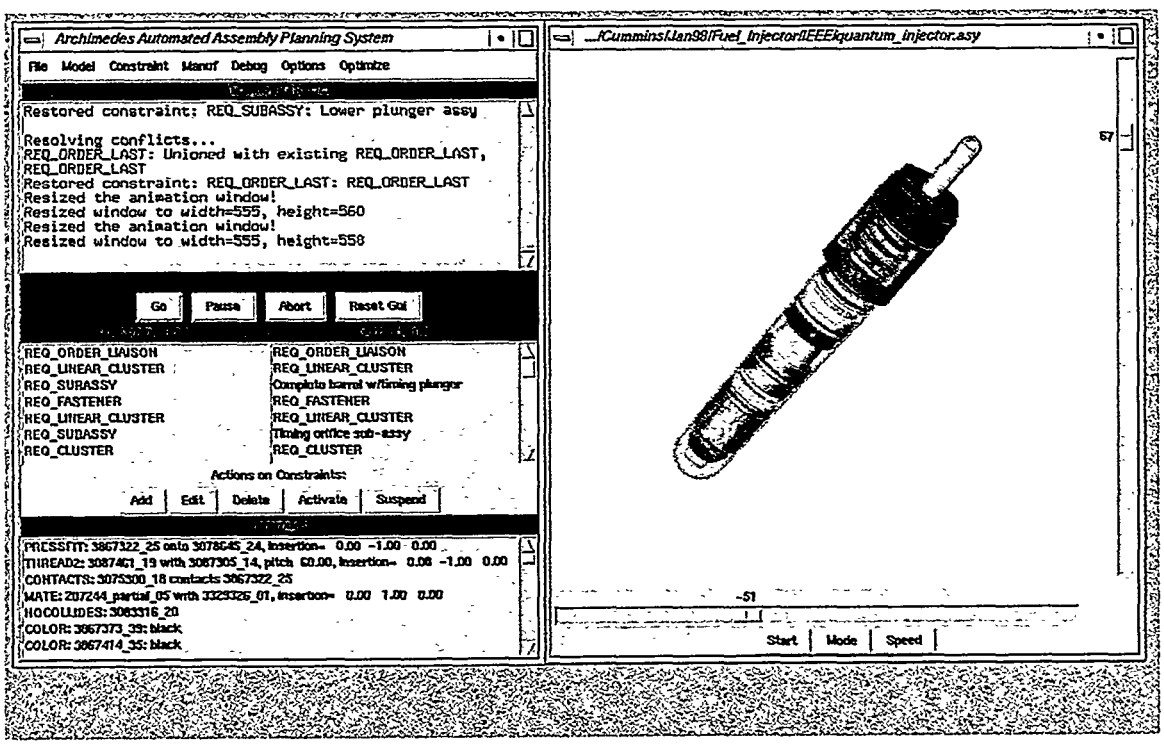

Figure B.1. The Archimedes 3.0© Automated Assembly Analysis System.

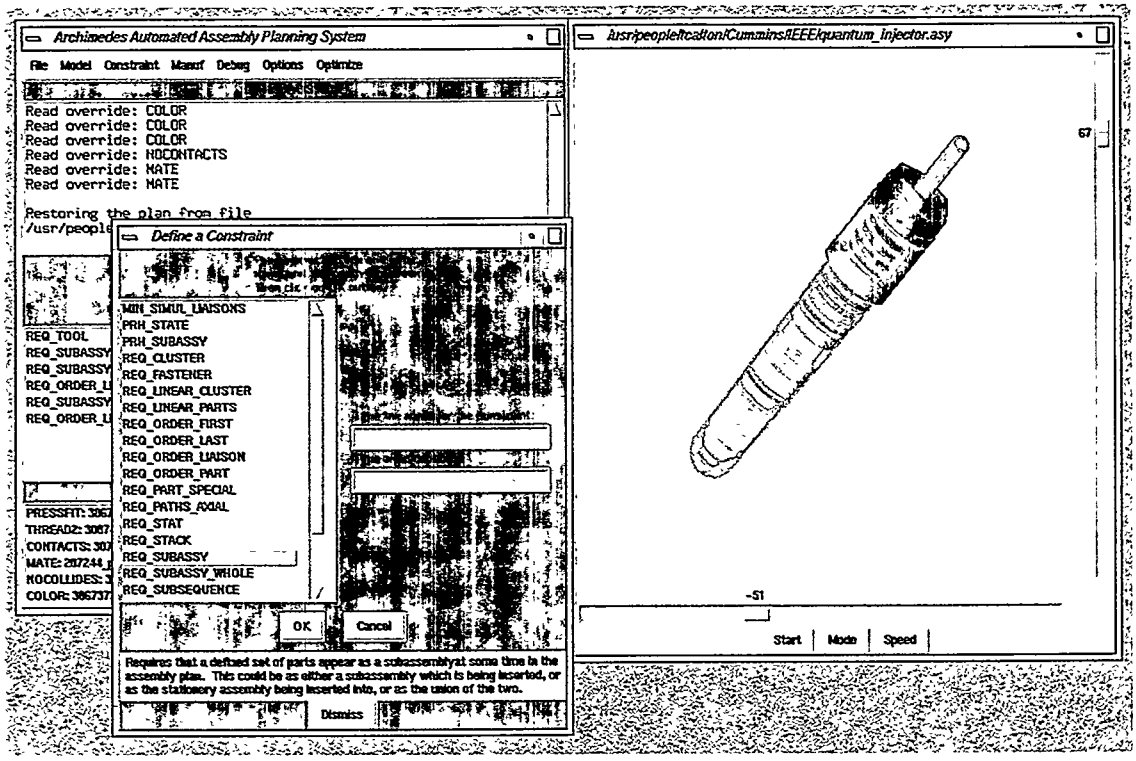

Figure B.2. Planning dialog, showing a partial list of constraints.

Figure B.3 shows the instantiation of a REQ_SUB_WHOLE. In this setting, the user selects the parts using a point-and-click method from within the animation window shown on the right-hand-side. The user is offered several access methods to selecting parts, which may be "inside" of the assembly. These options are shown in Figure B.3 at the bottom of the animation window.

Once defined, constraints are listed in the planning dialog (as shown in Figure B.1). They can be edited using a process very similar to initial definition. They can also be deleted, temporarily suspended, and re-activated. Constraint suspension is a very useful feature that allows the user to consider various scenarios for assembly. Constraints often embody assumptions about the product assembly scheme; by suspending some and replanning, the user can compare the cost of removing the assumption to the possible gains in assembly sequence efficiency that result.

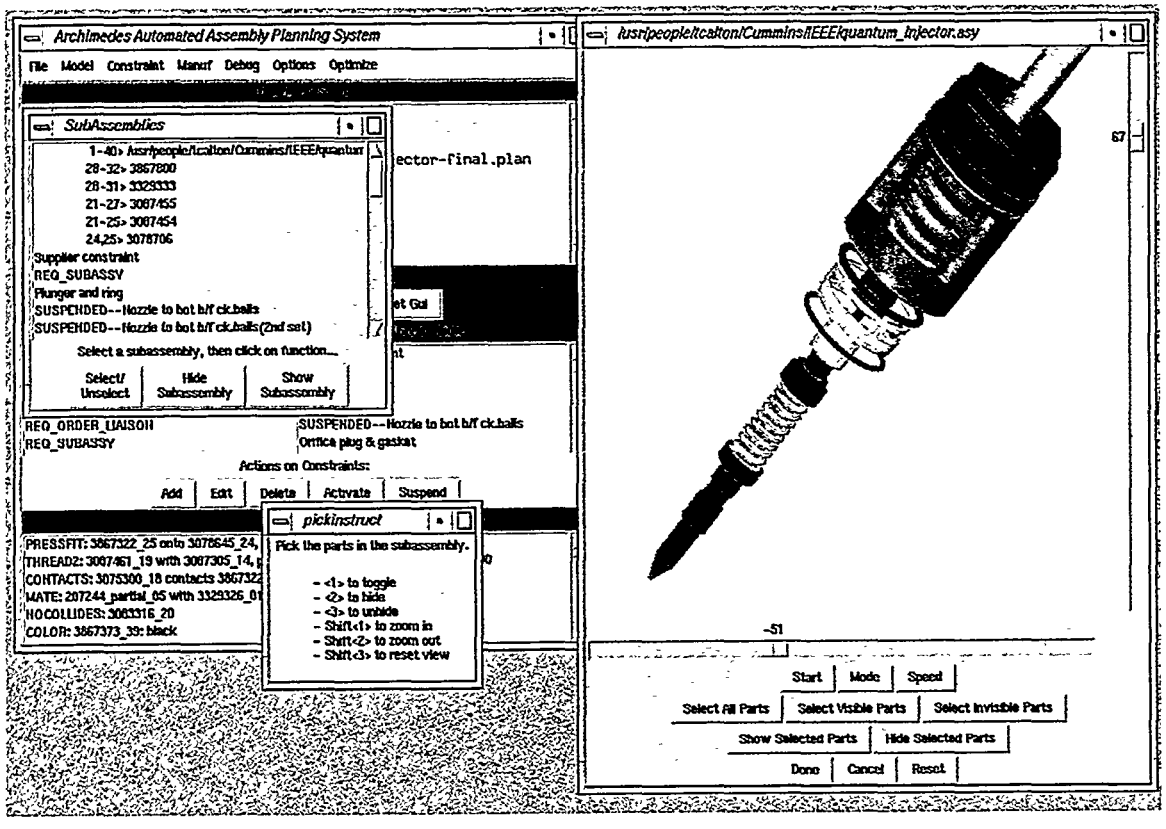

Figure B.3. Planning dialog, showing the instantiation of REQ_SUB_WHOLE constraint.

B.3 When Planning Fails

When a product cannot be assembled according to the current set of constraints, the planner fails and enters a "debug" mode that helps the user to determine the cause of the failure. For example, one can request that the planner try to remove a particular part or subassembly (from the subset of parts remaining when the planner failed) in a direction that appears possible. Collisions or constraints that disallow the operation are then posted in the status window. Other options include trying to disassemble every pair of parts in the offending subassembly, or trying to remove any parts along a given trajectory. This constraint debugging capability is very important and useful, since in some cases it may not be apparent to the user why the planner cannot find a feasible plan.

Often, the planner can fail without any user-defined constraints. This is sometimes due to limitations in the planner's algorithms, such as an inability to reason about flexible parts such as snapfits and springs. Other times, inaccuracies or inconsistencies in the product CAD data cause the planner to fail. Examples include pressfit parts and threaded parts that are modeled as

cylinders too large for their holes. Archimedes 3.0© includes a set of model adjustment features *or overrides*, which can be used to correct such problems. These include a function to effectively add threads to cylindrical contacts between parts; to specify that certain part insertions are in fact possible, even though collisions occur between the parts; and to delete a part temporarily.

B.4 Efficiency

As mentioned above, the generate-and-test abstraction can sometimes lead to an inefficient planning process where a large number of operations are generated, very few of which pass through the filters. In such cases, supplemental routines can improve planning efficiency greatly. These routines are very dependent on the internal algorithms of the planner. Because Archimedes 3.0© plans backward from the assembled state to individual parts, the supplemental routines must be implemented in reverse.

For instance, if the user has created a REQ_SUBASSY constraint with part set P, and parts not in P are present, then P cannot be "split" at that point in the plan. To implement this constraint efficiently, a supplemental routine binds the parts of P together for that stage, not considering any operations that split them. This is accomplished by placing bi-directional arcs between every pair of parts in P in every blocking graph of the subassembly [4].

Supplemental routines must be considered carefully, trading off the added speed against the increase in planner complexity. Three characteristics identify candidates for supplemental routines:

1. The constraint either leads to many dead-ends in the search space or rules out a very large proportion of generated operations,
2. An efficient method exists to implement the preprocessing, and
3. The constraint is used often.

The REQ_FASTENER constraint type is another instructive example. Fasteners are very common in mechanical assemblies. The REQ_FASTENER constraint requires that as soon as one set of parts is joined (the *fastened parts*), and then a set of *fastener parts* must immediately be placed. In reverse, this constraint means that as soon as a single fastener part is removed, then all other fasteners must be immediately removed, followed by at least one of the fastened parts. If any of the fasteners cannot be removed, a dead end appears in the search space (in fact many can appear).

The filter function for REQ_FASTENER is straightforward, but its supplemental routines are the most complex implemented. The fastener parts are removed from the assembly representation and considered *secondary parts* that implicitly must be added when the fastened parts are mated. Before generating operations to construct a certain subassembly, the planner determines which fasteners could be placed into the subassembly. For each fastener that cannot be placed, the

corresponding fastened parts are bound together as for REQ_SUBASSY. Fasteners are placed back in subassemblies for collision checking and other calculations.

Note that when a constraint has supplemental routines, the planner still calls the constraint's filter function, which should never return false. This double check is very useful to ensure correctness, because the supplemental routines are complex and interact with each other. It is conceivable that a supplemental routine would only *reduce* the number of operations rejected by the filter function, but we have not found such a case.

In almost all cases, adding constraints reduces planning time. The reduced size of the search space usually outweighs the extra time required to compute the filter functions. In fact, constraints can be used to guide the planner to a correct plan for assemblies that would otherwise have intractably large search spaces.

B.5 Implementation

Archimedes 3.0© is implemented in C++, with Tcl/Tk used for the graphic interface and OpenGL™ for 3D graphics and animation. The constraints are implemented as a hierarchy of C++ derived classes. Each type of constraint simply overrides the filter function from a base class, along with methods to define the type, name, etc. of the constraint. Each constraint type also has its own data members, such as part sets, tool choices and points of application, and so on. Some of the supplemental routines are implemented as constraint class methods; however, most cannot be separated from the planner's algorithms, and are woven directly into the planner implementation.

References

1. R. E. Jones and R. H. Wilson. An interactive assembly planning system. In *Video Proc. IEEE Intl. Conf. on Robotics and Automation*, 1997.

2. 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 Intl. Conf. on Robotics and Automation*, pages 3361-8, 2996.

3. P. Langley. Systematic and nonsystematic search strategies. *In Artificial Intelligence Planning Systems: Proc. of the First Intl. Conf.*, 1992.

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

Intentionally Left Blank.

Distribution:

1	MS	0188	LDRD Office, 4001
1		0507	C. M. Hart, 2600
1		1002	P. J. Eicker, 15200
1		1005	R. W. Shaum, 15201
1		1003	R. D. Robinett, 15211
1		1004	R. W. Harrigan, 15221
1		1010	M. E. Olson, 15222
1		1125	A. K. Miller, 15252
1		1006	P. Garcia, 15271
20		1008	T. L. Calton, 15221
10		1008	S. Blauwkamp, 15200 Library, 15221
3		1008	R. R. Peters, 15221
1		9018	Central Technical Files, 8940-2
2		0899	Technical Library, 4916
1		0612	Review & Approval Desk, 4912
			For DOE/OSTI
1		0161	Patent and Licensing Office, 11500