# An Intelligent Environment For Simulating Mechanical Assembly Operations

**Article** · October 1999

Source: CiteSeer

CITATIONS
16

READS
279

**5 authors**, including:

Satyandra K Gupta
University of Southern California
**451** PUBLICATIONS **6,781** CITATIONS
SEE PROFILE

Christiaan Paredis
Georgia Institute of Technology
**254** PUBLICATIONS **5,043** CITATIONS
SEE PROFILE

Rajarishi Sinha
Google Inc.
**30** PUBLICATIONS **436** CITATIONS
SEE PROFILE

**Proceedings of DETC'98**
**1998 ASME Design Engineering Technical Conferences**
**September 13-16, 1998, Atlanta, Georgia, USA**

# DETC98/DFM-5739

# AN INTELLIGENT ENVIRONMENT FOR SIMULATING MECHANICAL ASSEMBLY OPERATIONS

**Satyandra K. Gupta**[1]
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: skgupta@ri.cmu.edu
Tel.: 412-268-8780

**Christiaan J.J. Paredis**
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: paredis+@cmu.edu

**Rajarishi Sinha**
Institute for Complex Engineered Systems
1212 Hamburg Hall, Carnegie Mellon University
Pittsburgh, PA 15213
Email: rsinha+@cmu.edu

**Cheng-Hua Wang**
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: cwwang@cs.cmu.edu

**Peter F. Brown**
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: pbrown@cmu.edu

## ABSTRACT

Rapid technical advances in many different areas of scientific computing provide the enabling technologies for creating a comprehensive simulation and visualization environment for assembly design and planning. We have built an intelligent environment in which simple simulations can be composed together to create complex simulations for detecting potential assembly problems. Our goal in this project is to develop high fidelity assembly simulation and visualization tools that can detect assembly related problems without going through physical mock-ups. In addition, these tools can be used to create easy-to-visualize instructions for performing assembly and service operations.

## INTRODUCTION

Developing high-performance electro-mechanical products is a very challenging task. In order to improve efficiency and reduce the product weight and volume, designers need to pack a large number of components in a very small space. At the same time, in order to make products easier to assemble and service, designers need to leave enough room for performing assembly and disassembly operations. These requirements are quite often in conflict and make design of electro-mechanical products a highly iterative process. In the absence of high fidelity simulation tools, most product development teams are forced to include physical prototyping in the design loop to verify proper functioning and ease of assembly. Physical prototyping is a major bottleneck. It slows down the product development process and seriously constrains the number of design alternatives that can be examined. After a prototype has been successfully built and tested, a significant amount of time is spent creating instructions for performing assembly and service.

Rapid technical advances in many different areas of scientific computing provide the enabling technologies for creating a comprehensive simulation and visualization environment for assembly design and planning. We believe that developing and maintaining a single monolithic system for assembly simulations will not be practical. Instead, we have built an environment in which simple simulation tools can be composed together to create complex simulations. Our goal in this project is to develop high fidelity assembly simulation and visualization tools that can detect assembly related problems without going through physical mock-ups. In addition, these tools will be used to create easy-to-visualize instructions for performing assembly and

[1]Author to whom all correspondence should be addressed.

service operations.

In Intelligent Assembly Modeling and Simulation (IAMS) environment, the designer creates an assembly design using a commercial CAD package. After adding information about articulation and assembly features, the designer stores the design in the assembly format. The designer then selects a number of simulation tools and composes them together to form a customized simulation. In parallel, process engineers create a model of the work cell in which the parts will be assembled. The designer proposes an initial sequence in which this assembly can be performed — either interactively or through the use of assembly planning software. He uses the simulation environment to analyze the assembly and makes changes in the assembly after discovering problems. The simulation environment currently includes the facilities for performing interference detection, tool accessibility analysis, and detailed path planning.

When designer is satisfied with the design and hands it over to the process engineer. The process engineer can optimize the workspace and create a detailed animation of the assembly process. This sequence is down-loaded to the operator's desktop computer, where it can be consulted using a browser. The operator can start assembling the parts immediately, without the need for extensive training or tedious creation of documentation.

## REVIEW OF RELATED WORK

**Assembly Representation:** In order to use computer-aided assembly design and planning systems, we need to have computer-interpretable assembly representations. A number of different techniques have been developed to represent assemblies (Lee and Gossard, 1985; Lin and Chang, 1990; Shah and Rogers, 1993; Seow and Devanathan, 1994). In many of these techniques, each individual part is represented by its geometric model, and the assembly is represented by the relative position and orientation of each individual part in the final assembled configuration. Most non-geometric information such as fits, joints, etc. is stored as text labels. At present, there is no comprehensive scheme for representing articulated assemblies. In this project, we have developed assembly representations that allow us to capture articulation.

**Design for Assembly:** This area grew out of the desire to create products that are easier to assemble (Jakiela, 1989; Hsu et al., 1992; Lee et al., 1993; Boothroyd, 1994). It was started by the pioneering work of Boothroyd and Dewherst. Several different variations of Boothroyd and Dewherst techniques have been developed over the years. Most systems formulate this problem in the following way: given the description of an assembly, evaluate its goodness and offer redesign advice based on the bottleneck areas. Boothroyd and Dewherst Inc. offers commercial software based on the early research of its founders. Most DFA tools are currently not integrated with CAD software, lack abilities to perform geometric reasoning, and fail to detect problems due to interaction with workspace and tools. Our approach address these short-comings and complements the capabilities of DFA software.

**Assembly/Disassembly Operation Sequencing:** This body of research mainly focuses on generating feasible assembly or disassembly sequences (De Fazio et al., 1989; de Mello, 1989; Khosla and Mattikalli, 1989; Swaminathan and Barber, 1995). This research can be further divided into two groups. The first group developed techniques for finding optimal or near-optimal sequences. The second group developed techniques for efficiently enumerating all feasible sequences. In most cases, assembly time or cost is used as the objective function (Millner et al., 1994) to find the optimal sequence. In many systems, the first step is to create an assembly/disassembly dependency network. This network represents the blocking characteristics of the assembly. Information for creating the assembly/disassembly dependency network can either be supplied by the human or by an automated geometric reasoning system. The next step is to use state-space search techniques for finding the desired solution. Assembly sequences are used for assembling the part. Disassembly sequences are used for creating a maintenance plan or recycling the product (Navin-Chandra, 1993). This work currently does not address tool and process related issues. Our research complements this work by providing a much more extensive set of analysis capabilities.

**Motion Planning:** This research mainly deals with robot motion planning for robotic assembly (Shin et al., 1995). The problem is defined as: given a sequence of assembly operations to be performed, determine a series of robot moves that are capable of executing the given operation sequence and optimize the given objective function (i.e., operation time). In general, the problem of finding the optimal motion plan is shown to be P-space hard. Therefore, quite often the following two approaches are used to simplify the problem at hand. In the first approach, an attempt is made to find an optimal motion path under restricted set of robot moves. In the second approach, faster algorithms are developed to find near-optimal solutions.

**Tool Accessibility** To analyze tool accessibility in machining operations, two different approaches have been developed. The first approach checks for interference between

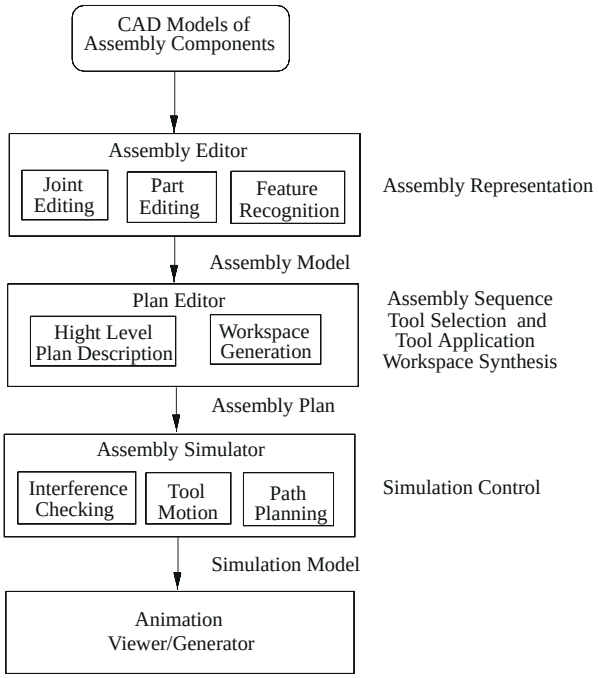



Figure 1. System Overview

the workpiece and the tool accessibility volume, which is defined as the non-cutting portion of the tool (Gupta, 1994; Henderson et al., 1997). The second approach uses visibility maps (Laxmiprasad and Sarma, 1997); a machining operation for a particular face is selected such that the tool approach direction is inside the visibility region for that face. Both approaches have also been applied for coordinate measuring machines, to determine the accessibility of inspection points. The tool accessibility problem has also been investigated for the assembly domain. Homem de Mello and Sanderson's (de Mello and Sanderson, 1991) model of assemblies include attachments, which describe fastening methods. Four types of attachment are considered: clip, pressure, screw, and glue. Even though the attachments can be used to generate disassembly sequences, the detailed tool movements required to carry out these attachment are not modeled. Diaz *et al.* (Diaz-Calderon et al., 1995) present a method that automatically determines the complexity of an assembly task based on the tool use. The most detailed work is reported by Wilson (Wilson, 1996). Wilson developed a tool representation that includes the tool's *use volume*, i.e. the minimum space that must be free in an assembly to apply the tool. He further divides tools into pre-tool, in-tool, and post-tool classes, based on the time at which the tools are applied relative to when the parts are mated in the assembly operation. In addition, he provides algorithms for finding feasible tool placements.

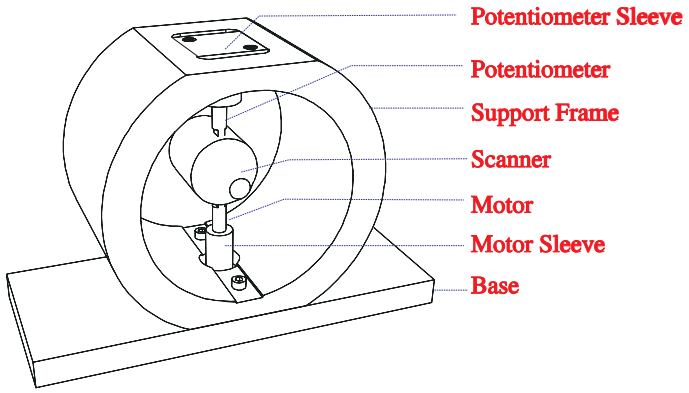



Figure 2. A One Degree of Freedom Scanner

## SYSTEM OVERVIEW

Our environment consists of four major components: (1) an assembly editor, (2) a plan editor, (3) an assembly simulator, and (4) an animation generator/viewer. The assembly editor imports CAD files of individual components from other ACIS-based solid modeling system and organizes them into an assembly representation. Using feature recognition techniques, the assembly editor recognizes joints between parts and assembly features on individual parts. The plan editor allows users to synthesize assembly plans interactively. The assembly sequence and tooling information (i.e., macro plans) entered by the user are automatically converted into low level tool and part motions (i.e., micro plans). Using the assembly simulator, the user selects and controls various simulations (such as interference and tool accessibility). The animation viewer allows the assembly operators to view the modeled assembly process interactively. The users can randomly access any particular operation in the assembly sequence and interactively change their 3D viewpoint. Figure 1 shows a high level overview of our system.

Practically, these components can be used in the following manner. A designer creates an assembly design using a commercial CAD package. The design is imported into our environment using the assembly editor. The designer than uses the plan editor to enter a specific assembly sequence. The designer selects a number of simulation agents in the simulation controller and composes them into a customized simulation. Based on the feedback from the simulations he may have to change the assembly design. After several design iterations, he is satisfied with the design and hands it over to the process engineer. In parallel, using the workspace editor, the process engineer has created a model of the work-cell in which this assembly will be performed. After incorporating the assembly in the workspace, the process engineer performs a detailed simulation to check for any

problems in the final assembly plan. He then generates an animation of the assembly process which is down-loaded to the operator's desktop computer where it can be viewed by the operator using animation viewer. The operator can start assembling the parts immediately, without the need for extensive training or tedious creation of documentation.

During our recent field trips to Allied Signal and Raytheon, we found that most assembly operators already have computers on their workbenches to display digitized drawings and images illustrating the assembly operations. As a result, we do not expect any major economic or social obstacles to adopting this technology in the workplace.

Our system has been implemented using C++ and runs on Sun and SGI workstations. We use ACIS for representing part geometry, RAPID and ACIS for performing interference tests and OpenInventor for graphics.

Figure 2 shows the assembly of a scanner. This scanner has one degree of freedom and is used to detect the presence of an object by scanning horizontally. We will be using this example throughout this paper to illustrate the capabilities of our system.

## ASSEMBLY EDITOR

Assemblies are a collection of components, which by virtue of their relative positions and orientations, satisfy certain functional requirements of the designer. However, specifying the part geometry and the relative positions and orientations of the parts is not sufficient for performing assembly simulations.

To address this issue, we have developed an Assembly Editor which assists the designer in creating a representation of the assembly. Apart from grouping all the parts into a single data structure, the Assembly Editor also provides the designer with the ability to attach attributes to parts (such as material name, density and color, etc.). The Editor automatically identifies features on the part which can aid in subsequent tool selection, and incorporates the ability to populate a representation of articulation for each joint in the assembly. Once this is done, two very important tasks during assembly sequence planning and simulation happen seamlessly, with minimal user interaction:

- Tool selection is based on the part features that have been detected by the Assembly Editor. Depending on the type of feature recognized (such as a Phillips slot), the appropriate tool will be automatically selected and will know how to position itself on the part.
- Simulation of articulation can now proceed at a very high level, namely, the designer needs only to specify a value for the degree of freedom for a joint, and the system will proceed to articulate it; it is no longer required to mimic articulation by a relative motion of the part by providing a transformation.

### Joint Specification

Most electro-mechanical products have built-in articulation. Examples include household appliances, computer disk drives, tape drives, satellites, VCRs, etc. Most existing assembly representations do not explicitly model kinematic behavior of the assembly. In order to correctly generate assembly or disassembly plans for such products, we need to take articulation information into account.

Before joints can be defined, it is helpful to know which parts are in contact with which other parts. Some subset of these contacts will become joints in the final assembly. Beginning with the set of part geometry, we generate all the possible contacts between these parts, and populate an undirected *contact graph*, with parts represented as nodes, and contacts as edges. Some of these contacts will become legitimate joints; others can be deleted by the designer.

By default, all the contacts are fixed joints. Those contacts which are not fixed joints in the final assembly can be converted to one of the following lower pairs: Revolute (1 Rotational Degree of Freedom (RDOF)), Cylindrical (1 RDOF, 1 Translational Degree of Freedom (TDOF)), Prismatic (1 TDOF), Planar (1 RDOF, 2 TDOF) and Spherical (3 RDOF).

We define three levels of articulation representation. The highest level is one of body-to-body contacts, and is obtained from the graph. The contacts can be classified into two types - constraint contacts and incidental contacts. Constraint contacts limit the degrees of freedom of the component, and incidental contacts limit the range of motions allowed for each degree of freedom. The intermediate level uses the body-to-body contact information to obtain body-to-body constraints. The result of each intersection is examined for features such as cylindrical faces, opposing planar faces, etc. The point and line features on these faces form the low-level representation.

A complete representation of a joint between two parts in an assembly requires two pieces of information - the position of the joint on each part and a description of the degrees of freedom of the joint. The position is specified by a position vector. The degrees of freedom are specified by defining features on the parts which constrain the relative movement of the parts and by defining limits on how much constrained movement is allowed. Figure 3 shows a revolute joint being specified in the assembly editor. Representation of a revolute joint is given by:

```
{ Joint { Name <name> } { Type REVOLUTE }
        { Features { Axis <part_name> <position>
                          <unit_vector> <unit_vector> }
```

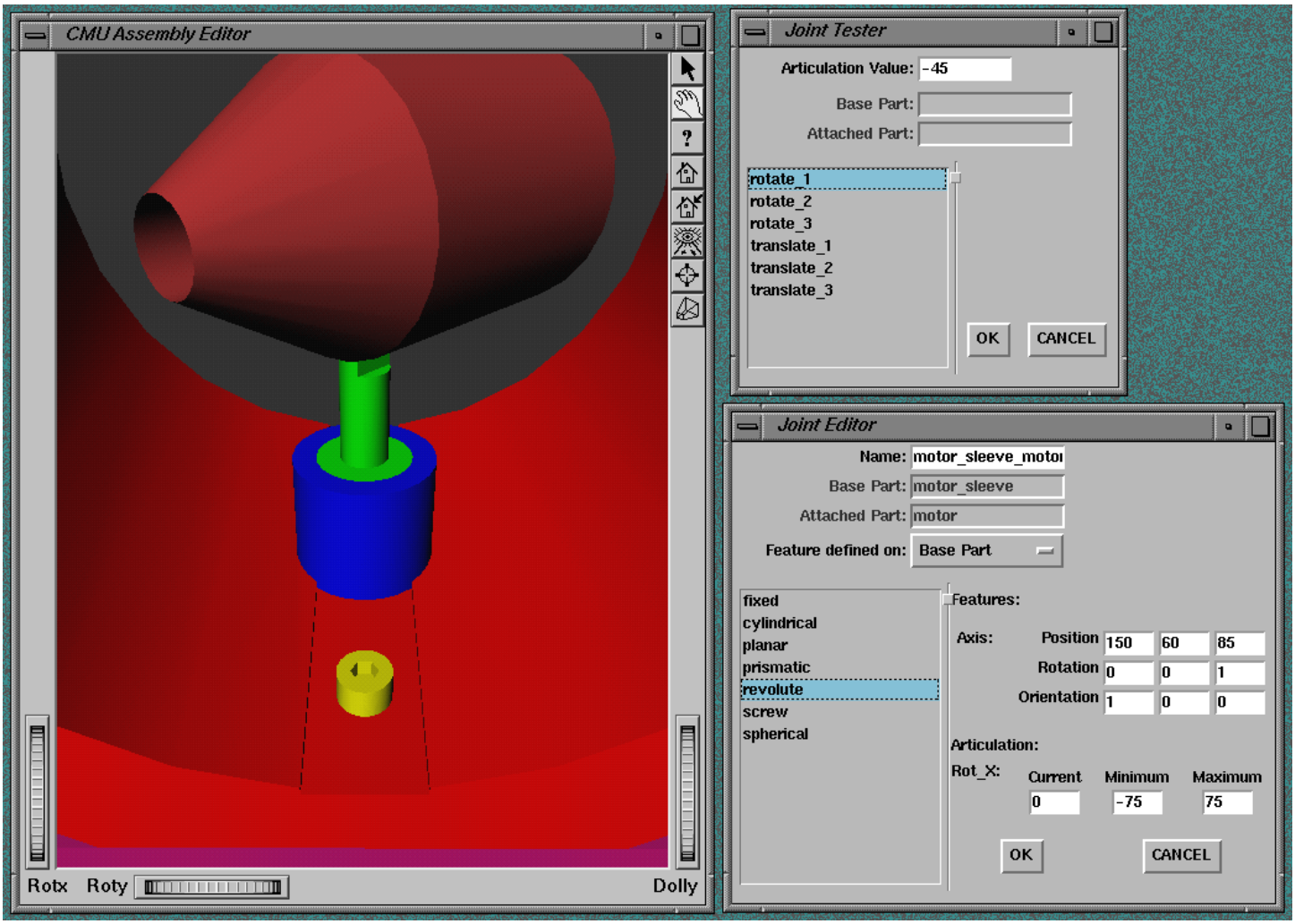



Figure 3. Specifying a revolute joint in the assembly editor.

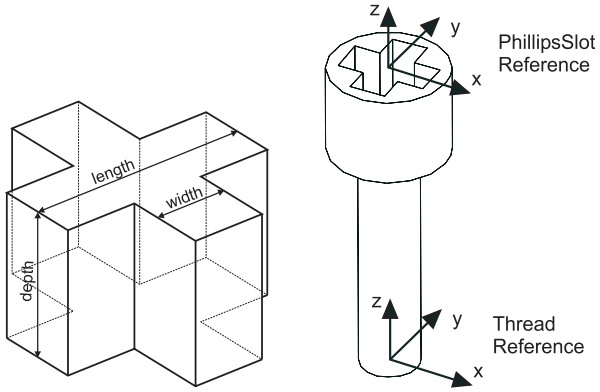
```
        { Axis <part_name> <position>
               <unit_vector> <unit_vector> }}
{ Articulation { Theta <real_number> } }
{ Limits { Rotation <real_number> <real_number> }}}
```

Once all the joints are defined and tested for correctness, and unwanted contacts have been removed from the contact graph, the designer is able to save the newly created assembly.

### Feature Recognition

Whether a tool is applicable for a part or not depends on the features of the part to which it is applied. For example, a Phillips screwdriver can only be used for screwing in a screw with a Phillips slot of the appropriate dimensions.

We have developed an algorithm that automatically recognizes assembly features. The feature recognition algorithm is quite generic because of our canonical representation of features. The representation consists of a parametric base face (with a particular surface type, number of edges, interior angles, and accessibility condition) and a set of neighboring faces. The algorithm loops over all the faces in the solid model and compares each of them with the parametric representation of the features' base face. If there is a match and the base face needs to be accessible, the algorithm checks for intersections between the normally swept volume of the base face and the rest of the part. Finally, the neighboring faces of the base face are matched with the neighboring faces of the feature template. If all of the above conditions are met, the algorithm extracts features' reference frame and the values of its parameters. As an example consider the `PhillipsSlot` feature defined in Figure 4(a) and




(a): A Phillips slot feature (b): Phillips slot feature on a screw

Figure 4. An example of an assembly feature.

shown in Figure 4(b) as part of a screw.

## PLAN EDITOR

To construct assembly plans for simulation, we need to satisfy two important criteria. On the one hand, we need to create a description of an assembly plan at a level at which humans are able to provide high level expertise. On the other hand, we need a very detailed description for performing high fidelity simulations. For instance, for a given assembly operation, it is very easy for the user to propose the appropriate tool. However, for simulation purposes, this operation needs to be augmented with detailed information about the position of each of the parts and tools at any time. Our approach is to create automatic plan completion capabilities that bridge the gap between these two requirements and relieve the user from tedious entry of low-level details; these details will be automatically generated by the system.

### Workspace Generation

Whether an assembly plan is acceptable or not depends not only on the geometry of the parts, but also on the workspace in which the assembly will be created, and on the tools with which the assembly operations will be performed. It is important that we build comprehensive representations for both the workspace and the tools. For instance, in addition to the tool geometry, our representation captures information about the proper tool usage (e.g., orientation, movement, etc.).

The workspace generator takes the assembly and the tool models as input and synthesizes a working environment which includes: an assembly table, a part rack, and a tool rack. The layout of the environment and the configurations of each of the parts and tools are also determined. All parts and tools are oriented so that their principal axes are aligned with the X, Y, and Z axes in the world coordinate. Identical parts are grouped and placed in the same cell of the part rack. Figure 5 shows the workspace for the scanner.

### Entering High Level Plans

To facilitate the generation of detailed assembly process specifications, we have implemented an assembly plan editor. With this editor, the process engineer can specify a high-level assembly sequence, as is illustrated in Figure 6. Such a sequence consists of high-level assembly operations performed on individual parts or subassemblies. The operations that are currently supported include: relative and absolute positioning of parts, fitting, extracting, screwing, tightening, soldering, brazing, and coating. When one of these operations is selected, the plan editor will display a customized form in which the user can specify additional information. For instance, for a screwing operation, the user is asked for the name of the screw, the tool name, and the duration of the operation.

A micro-planner (described in the Section on Micro Planning) will later convert the high-level assembly plan into low-level detailed assembly primitives. For example, a single screwing operation will result in a sequence of assembly primitives starting with the positioning of the screwdriver, engaging of the screwdriver with the screw, the actual screwing, the disengagement, and finally, the return of the screwdriver to the shelf. All the additional information needed to generate such a low-level sequence is extracted automatically from the tool, part and assembly models. Once the process engineer has entered a high-level assembly operation, he can verify the corresponding micro-plan immediately in the graphical animation window.

## ASSEMBLY SIMULATION

Assembly simulation module consists of a simulation controller and three tools. In the following subsections, we describe various simulation tools.

### Interference Detection

In order to ensure successful assembly, it is very important to avoid part-part and part-tool interference during assembly operations. Such interference can damage parts and tools. We have developed an agent that will detect part-part interference during assembly operations and allow designers to either modify the design or change the assembly plan to eliminate such problems. Figure 7 shows an example of detecting part-part interference in a scanner assembly.

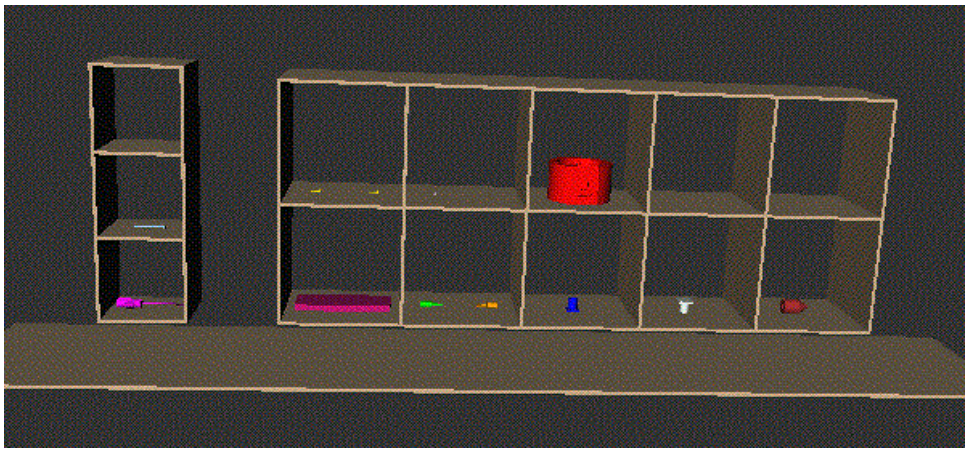


Figure 5. Workspace for the scanner.

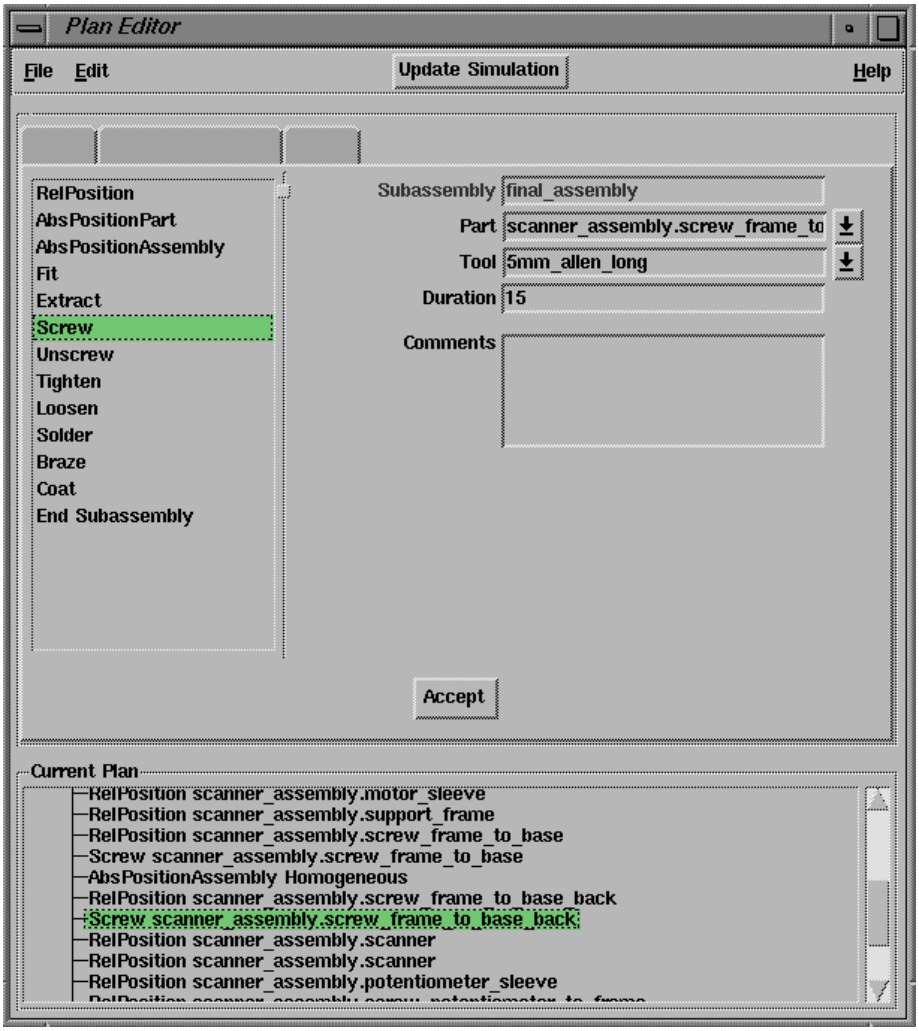



Figure 6. A screw operation being specified in the plan editor.

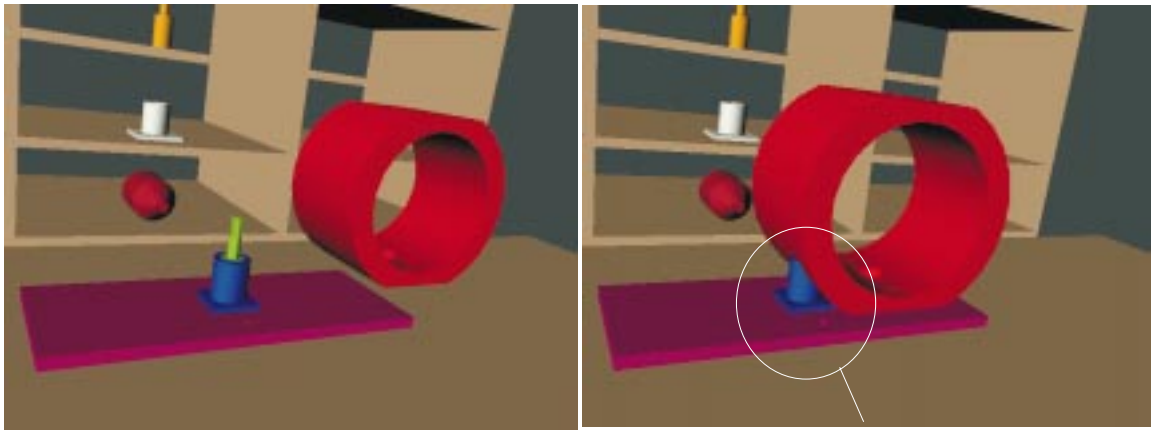



Figure 7. Detecting part part interference.

In real-world problems, assemblies tend to contain a large number of parts, each with complicated geometry. For most interference checking algorithms, performance deteriorates rapidly with the increase in the number of parts in the assembly and the number of faces per part. Most assembly design and planning problems exhibit several levels of nested combinatorial explosion, making scalability an important research topic. Computation time for detecting interference depends on the size of geometric models. Quite often, geometric models can be simplified before interference checks. We use hierarchical approximations of the geometric models of individual parts. For instance, checking for interference between the bounding enclosures of parts can be a quick first test for part-part interference. Only when the bounding boxes intersect, do we then need to proceed with more complex tests.

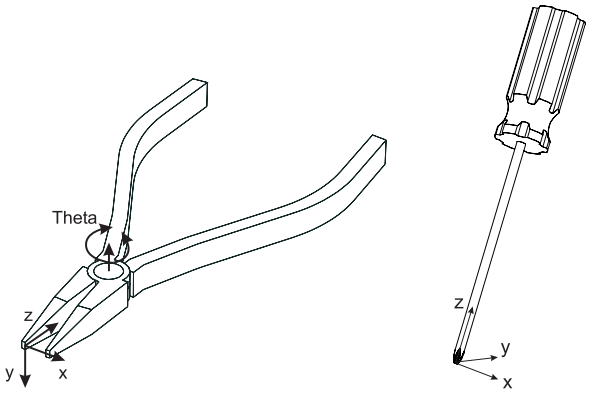
### Micro Planner for Analyzing Tool Accessibility

The ability to account for tooling requirements is critical in both design and assembly of complex electromechanical products. The designers need to make sure there is no interference between tools and parts. Likewise, process engineers need to ensure that an assembly operator has enough room to manipulate the tools. Moreover, both the execution time and the quality of the assembly depend on the tool type and its particular application. Without adequate software-aids, designers currently rely on physical prototypes to investigate tool accessibility issues. We have built micro planning software that helps designers and process engineers to select and evaluate tool applications within assembly plans. This work is described in detail in (Gupta et al., 1998).

**Tool Representation** To enable reasoning about tool usage, we have developed tool models that capture the following three types of information.

1. *Tool Geometry:* To investigate accessibility issues, we need to represent the geometry of a tool. In general, we model tools as articulated devices. During the assembly process, one can control the joint values for each of the DOFs of the tool. As illustrated in Figure 8, we also define a *tool application frame* that indicates the tool's position relative to the part to which it is applied.
2. *Tool Parameters:* For a given assembly operation, the size and shape of the tool need to match the size and shape of the part feature to which it is applied. To check whether this condition is satisfied without having to rely on geometric reasoning with solid models, the size of the tool is abstracted in the tool parameters. As illustrated in Figure 9, a flathead screwdriver is characterized by the length, width, and height of its tip. These parameters are compared to the parameters of the corresponding part feature in the *applicability condition*.
3. *Tool Use:* The final component of our tool model lists, for each type of assembly operation supported by the tool, the sequence of motions describing the proper tool use. Some of the types of assembly operations that are currently supported are: position, screw, unscrew, tighten, loosen, reorient, fit, and extract. A tool will typically support one or more of these operations.

As shown in Figure 9, the motion primitives are expressed in TCL syntax and may contain references to both the part and tool parameters. This parametric description allows complicated tool movements to be expressed as a




(a): plier (b): screw driver

Figure 8. Examples of assembly tools.

simple sequence of elementary motions. Some elementary motions, such as relative and absolute linear moves of parts or tools, translate directly into low-level simulation primitives; others, such as *screw with retraction*, require a large number of simulation primitives. The *screw with retraction* motion-macro divides a single screwing operation into multiple applications over 60 or 120 degrees each. This is useful in case a wrench is used in a confined area in which it cannot complete a full 360-degree rotation. Figure 10 shows an example, where tool motion range is restricted due to obstruction.

For a given tool and assembly operation, the tool motion script models the motion sequence of the tool required to execute the assembly operation. In our environment, the micro planner automatically converts the high-level assembly operations and tooling information into low-level part and tool motions.

**Path Planning**

To move a tool from its initial position in the workspace environment to the application position, requires in general a sequence of translations/rotations. However, there is currently no convenient mechanism available to enter a 6-DOF path into the computer; specifying via-points textually is tedious and error-prone, graphical entry is difficult because a mouse has only two DOFs, and VR environments with 6-DOF hand tracking are still very expensive and therefore not readily accessible.

It is our goal to relieve the user from the path planning task all together, and generate tool and part paths automatically. The 6-DOF path planning problem is very challenging, however, due to the following characteristics. First, in the application position, the tool tends to be in contact with the part to which it is applied. That means that in the configuration space the goal configuration may

```
{ Tool
  { Name screwdriver }
  { Body
    { URL screwdriver1.asm }
    { Transform identity } }
  { PartFeatures
    { Name FlatSlot }
    { Name Thread } }
  { Parameters
    { Length 3 }
    { Width 0.8 }
    { Height 1.2 } }
  { ApplicabilityCondition
    # TCL script
    "expr $Width < $FlatSlot(Width)
       && $Height > $FlatSlot(Depth)" }
  { Operation { Type screw }
    # TCL script
    "# engage motion
     AbsMoveTool $FlatSlot(Reference)
                 RelativeTo $part(transform);
     RelMoveTool Transform translation 0 0 -$FlatSlot(Depth);
     Attach $tool(name) $part(name);
     # operate motion
     RelMoveTool Transform screw 0 0 0 0 0 1
         [expr -360*$Thread(Depth)*$Thread(hand)
               $Thread(Pitch)]
         -$Thread(Depth);
     # disengage motion
     Detach $tool(name) $part(name);
     RelMoveTool Transform translation 0 0
                 $FlatSlot(Depth);" }
  { Operation { Type unscrew }
    # TCL script
    "# engage motion
     AbsMoveTool $FlatSlot(Reference)
                 RelativeTo $part(transform);
     RelMoveTool Transform translation 0 0 -$FlatSlot(Depth);
     Attach $tool(name) $part(name);
     # operate motion
     RelMoveTool Transform screw 0 0 0 0 0 1
         [expr 360*$Thread(Depth)*$Thread(hand)
               $Thread(Pitch)]
         $Thread(Depth);
     # disengage motion
     Detach $tool(name) $part(name);
     RelMoveTool Transform translation 0 0
                 $FlatSlot(Depth);" }
}
```

Figure 9. A file describing a flathead screwdriver.

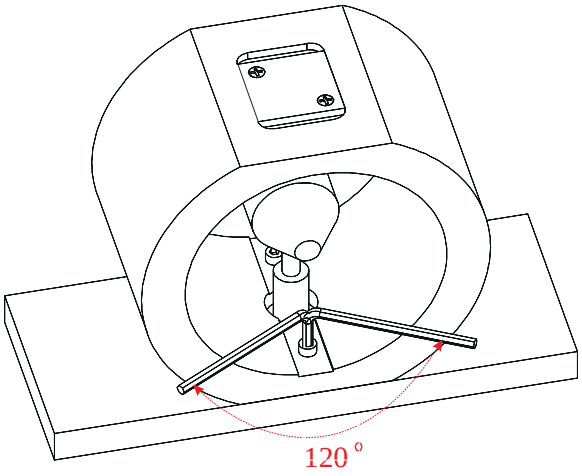



Figure 10. An example of restricted tool motion range.

be almost completely surrounded by obstacles, which is a situation that is typically difficult to handle for path planning algorithms. Second, one only computes a path once for each part in the assembly. This means, that one cannot amortize any of the pre-computations required by some of the path planning algorithms (e.g. pre-computation of the configuration space).

We have implemented a simple potential field method that is an extension of the method described in (Overmars and Svestka, 1994). The method lets the tool make small moves to 36 neighboring positions (6 pure translations, 6 pure rotations, and 24 combined rotations/translations), and ranks these positions according to their distance from the goal position. The tool is moved to that direct neighbor that is closest to the goal without colliding with any obstacles. The algorithm terminates when the goal is reached or when no neighboring positions closer to the goal are collision free. Our method is described in detail in (Gupta et al., 1998).

## ANIMATION GENERATOR/VIEWER

We envision that a number of applications can benefit from our visualization agent. We create an interactive animation of the proposed assembly plan. This animation includes the visualization of the tools and assembly. Such a complete and high fidelity visualization environment can be used by the design and process engineers while editing and simulating an assembly plan, but also by operators on the shop-floor to learn the assembly process. The idea is to store the visualization information in a compact assembly-movie-format that can be viewed interactively with a simple multi-media viewer (separate from the editing and simulation environment) on a low-cost PC.

Most companies today use engineering drawings and textual instruction sheets as the primary medium for giving instructions to their assembly operators. This means that the assembly operator needs to be trained to read engineering drawings and textual instruction sheets. Moreover, for complex assemblies, following engineering drawings and textual instruction sheets takes a very long time and is also prone to misinterpretation. From the perspective of the manufacturing engineer, the creation of the engineering drawings and textual descriptions for each assembly operation is a very time-consuming process. Moreover, as parts undergo changes, the assembly instructions need to be updated and redistributed to all the operators. This results in a huge version control problem.

With current multi-media technology and solid modelling based CAD, one can automatically create audio visual animations illustrating the entire assembly sequence. We believe that such a capability will allow even an untrained operator to follow assembly instructions without making errors. Automatic generation and distribution of these interactive 3D instructions will result in a significant workload reduction for manufacturing and process engineers, and will increase reliability through improved version control of the instructions.

The shop-floor visualization environment that we have developed provides the assembly operator with a random-access 3D interactive animation of the complete assembly process. In a compact format, the animation viewer stores the state of each component or tool as a function of time. This allows us to regenerate the state of the complete system (assembly workspace, tools, and the assembly components themselves) almost instantaneously, resulting in a random-access animation. The operator can jump forward or backward in time to a particular assembly operation of interest. To investigate the details of this operation, he can adjust the speed of the simulation continuously, pause it, or even move backwards in time. Furthermore, at any time during the animation, the operator can change the camera's viewpoint or zoom in to better study the details of an operation. Figure 11 shows a snap shot of our animation viewer.

## CONCLUSIONS

In this paper we describe a system for performing a variety of assembly simulations. Our simulation environment incorporates the following new features.

- *Articulated Tools and Products:* Most EM products are articulated. However, most assembly planning systems do not properly handle articulated products and tools. Our assembly simulator will be able to handle products and tools with built-in articulation. This is important

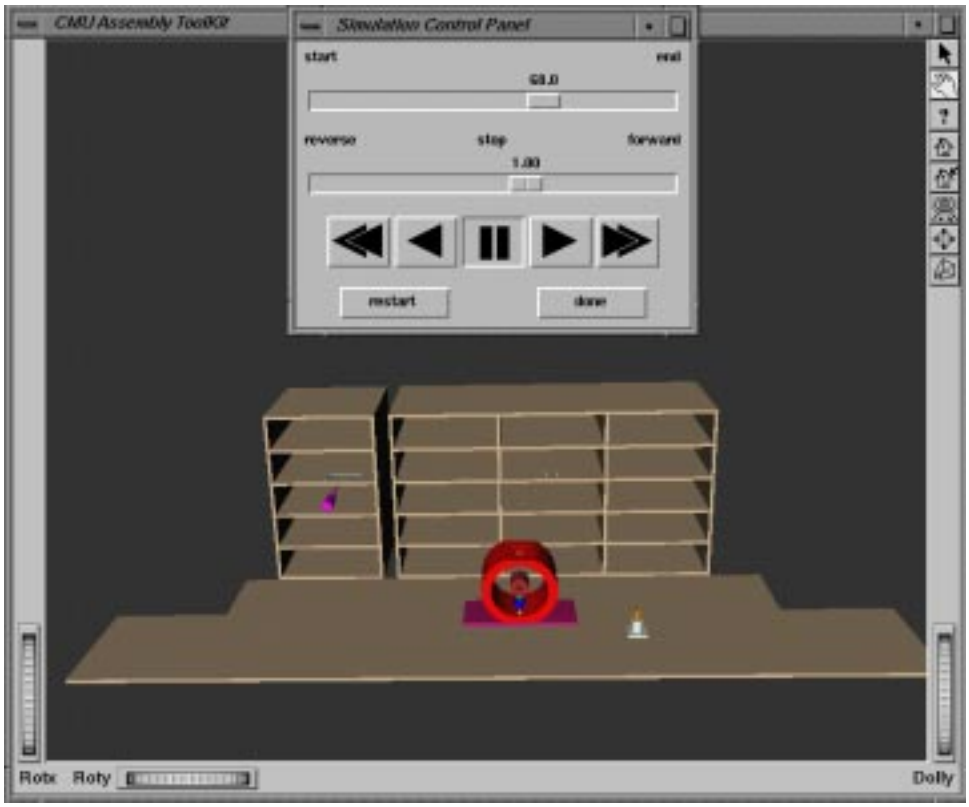



Figure 11. Animation Viewer

for a large variety of designs, for which the articulated components need to be moved to perform the assembly operations.

- *Automatic Plan Completion:* When designing a complex electro-mechanical product, the designer usually already has a coarse assembly sequence in mind. However, to perform a high fidelity simulation, it is important to specify an assembly plan in complete detail. Our framework provides plan completion features that automatically fill in the details of high level assembly operations specified by the design and process engineers.
- *Assembly Process Modeling:* Most research efforts have focussed on the geometric aspects of the assembly (i.e., finding a sequence of assembly operation without part-part interference). We believe that assembly tools and the workspace play a very significant role. Many of the problems related to assembly cannot be recognized without taking process models into account. We therefore model the workspace. This allows the process engineers to evaluate various types of environments in which the assemblies can be performed.

We believe that our assembly modeling and simulation infra-structure described in this paper will allow the creation of much more complex products in a much shorter time. Specifically, we envision the following three main advantages:

- *Reduction in Physical Prototyping:* By reducing the need for physical prototyping, we will be able to complete each design iteration much faster and significantly reduce the cost of prototyping.
- *Agile Work-force:* Ability to provide easy-to-follow instructions eliminates the need for work-force training in specialized activities. Instead, we can have a agile work-force that can be deployed to handle a wide variety of tasks.
- *Better Assembly Analysis/Planning Software:* We believe that our simulation environment can be combined with a number of assembly analysis/planning tools to create much better software. In particular, we see the following three potential applications of this research: (1) automated assembly planners, (2) optimum design for assembly workspaces, and (3) automated assembly redesign to improve manufacturability.

## ACKNOWLEDGMENT

This research was funded in part by DARPA under contract ONR #N00014-96-1-0854 , by Raytheon Company, the Robotics Institute, and the Institute for Complex Engineered Systems at Carnegie Mellon University. We would like to thank other people in our group who have contributed to the implementation of IAMS: Tim Bruce, Antonio Diaz-Calderon, and Sripal Mehta.

## REFERENCES

Geoffrey Boothroyd. Product design for manufacture and assembly. *Computer Aided Design*, 26(9):505–520, 1994.

L Homem de Mello and A Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. In *IEEE International Conference on Robotics and Automation*, pages 56–61, May 1989.

L.S. Homem de Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 7(2):228–240, April 1991.

Antonio Diaz-Calderon, D. Navin-Chandra, and Pradeep K. Khosla. Measuring the difficulty of assembly tasks from tool access information. In *Proceedings of the IEEE International Symposum on Assembly and Task Planning*, pages 87–93, 1995.

T De Fazio, D Whitney, Man-Cheung Max Lui, T Abell, and D Baldwin. Aids for the design of choice of assembly sequences. In *IEEE International Conference on Systems Machines and Cybernetics*, Nov 1989.

S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: a hierarchical structure for rapid interference detection. In *Proceedings of ACM Siggraph'96*, 1996.

S. K. Gupta, C. J. J. Paredis, and P. F. Brown. Micro planning for mechanical assembly operations. In *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 1998.

S.K. Gupta. *Automated Manufacturability Analysis of Machined Parts*. PhD thesis, University of Maryland, College Park, MD, September 1994.

W Hsu, C S G Lee, and S F Fu. Feedback evaluation of assembly plans. In *IEEE International Conference on Robotics and Automation*, pages 2419–2424, 1992.

Mark J. Jakiela. Intelligent suggestive CAD systems : Research overview. Technical report, Massachusetts Institute of Technology, 1989.

P Khosla and R Mattikalli. Determining the assembly sequence from a 3-D model. *Journal of Mechanical Working Technology*, pages 153–162, Sep 1989.

Putta Laxmiprasad and Sanjay Sarma. A feature free approach to 5-axis tool path generation. In *ASME Design for Manufacturing Conference*, Sacremento, CA, September 1997.

Kunwoo Lee and D C Gossard. An hierarchical data structure for representing assemblies: part 1. *Computer-Aided Design*, 17(1):15–19, Jan 1985.

S Lee, G. Kim, and G. Bekey. Combining assembly planning with redesign: An approach for more effective DFA. In *IEEE International Conference on Robotics and Automation*, May 1993.

A C Lin and T C Chang. Product modeling and process planning for 3-dimensional mechanical assemblies. In *Proc of the NSF Design & Manufacturing Grantees Conference*, pages 633–640, Jan 1990.

Joseph Millner, Stephen Graves, and Daniel Whitney. Using simulated annealing to select least-cost assembly sequences. In *IEEE International Conference on Robotics and Automation*, pages 2058–2063, San Diego, May 1994.

D Navin-Chandra. ReStar: A design tool for environmental recovery analysis. In *9th International Conference on Engineering Design*, 1993.

Mark. H. Overmars and Petr Svestka. A probabilistic learning approach to motion planning. In *Proceedings of the First Workshop on the Alogrithmic Foundations of Robotics*, pages 19–37. A. K. Peeters, Boston, MA, 1994.

Venkat N. Rajan, Kevin W. Lyons, and Raj Sreerangam. Generation of component degrees of freedom from assembly surface mating constraints. In *ASME Design Theory and Methods Conference*, Sacremento, CA, September 1997.

K T Seow and Rajagopalan Devanathan. A temporal framework for assembly sequence representation and analysis. *IEEE Transactions on Robotics and Automation*, 10(2):220–229, April 1994.

J. J. Shah and M. T. Rogers. Assembly modeling as an extension of feature-based design. *Research in Engineering Design*, 5(3,4):218–237, 1993.

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

Roger Stage, Mark Henderson, and Chell Roberts. A framework for representing and computing tool accessibility. In *ASME Design for Manufacturing Conference*, Sacremento, CA, September 1997.

A Swaminathan and K S Barber. Ape: An experience-based assembly sequence planner for mechanical assemblies. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1278–1283, May 1995.

Randall H. Wilson. A framework for geometric reasoning about tools in assembly. In *Proceedings of the 1996 International Conference on Robotics and Automation*, pages 1837–1844, Minneapolis, MN, 1996.