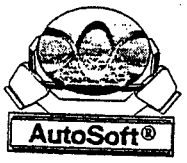

Intelligent Automation and Soft Computing, Vol. 2, No. 1, pp. 1-14, 1996

A VIRTUAL MANUFACTURING WORKCELL FOR AUTOMATED ASSEMBLY†

AKIHIRO SATO
Production Engineering Development Laboratory
NEC Corporation
Kawasaki, Kanagawa, Japan

ANTHONY A. MACIEJEWSKI
Purdue University
West Lafayette, Indiana, USA

ABSTRACT—This work describes the implementation of a novel robot workcell programming interface that allows an assembly designer to obtain immediate feedback regarding the manufacturability of his/her design. The interface allows the user to manipulate the three-dimensional CAD/CAM models of the components and "assemble" them into the final product. The computer then analyzes the relevant assembly operations and translates them into low-level commands for the robots in the specific workcell under consideration. This work is motivated by the complexity and time-consuming nature of manually programming flexible assembly cells for the manufacture of different products, particularly when they involve the cooperation of multiple robot manipulators.

Key Words: virtual environment, automated assembly

1. INTRODUCTION

The introduction of robots into assembly lines has resulted in a significant improvement in both the speed and quality of automated assembly. However, the goal of using multiple robots and ancillary automation equipment in flexible workcells that can automatically adapt to different products produced in small batches has been largely unrealized. One impediment to the realization of this goal is that, in spite of the fact that robots are by definition adaptable machines, the human effort required to reprogram workcells for different tasks has been considerable. The advent of programming languages for manufacturing and automation was one step in addressing this issue.[1] The utility of adding a graphical simulation component to these programming languages was quickly realized.[2,3,4] Recently, the graphical interaction associated with assembly planning has been enhanced to provide virtual environments for planning automated assembly.[5,6,7] It is this progression of interaction with a prospective product design to assess its manufacturability that motivates the work described here.

Since the human designer has the most knowledge concerning the assembly of a prospective product, the focus of this work is to glean from him/her the necessary knowledge for automating the assembly process. We are particularly interested in three pieces of information that are trivial for the designer and yet very difficult to automatically calculate given only the product geometry. These are: (1) the desired order of assembly, (2) stable grasp configurations for the components, and (3) a fine-motion strategy that

† This work was supported by the NEC Corporation and in part by the National Science Foundation under grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems. An earlier conference version of this work was presented at the 1994 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, October 2-5, 1994.

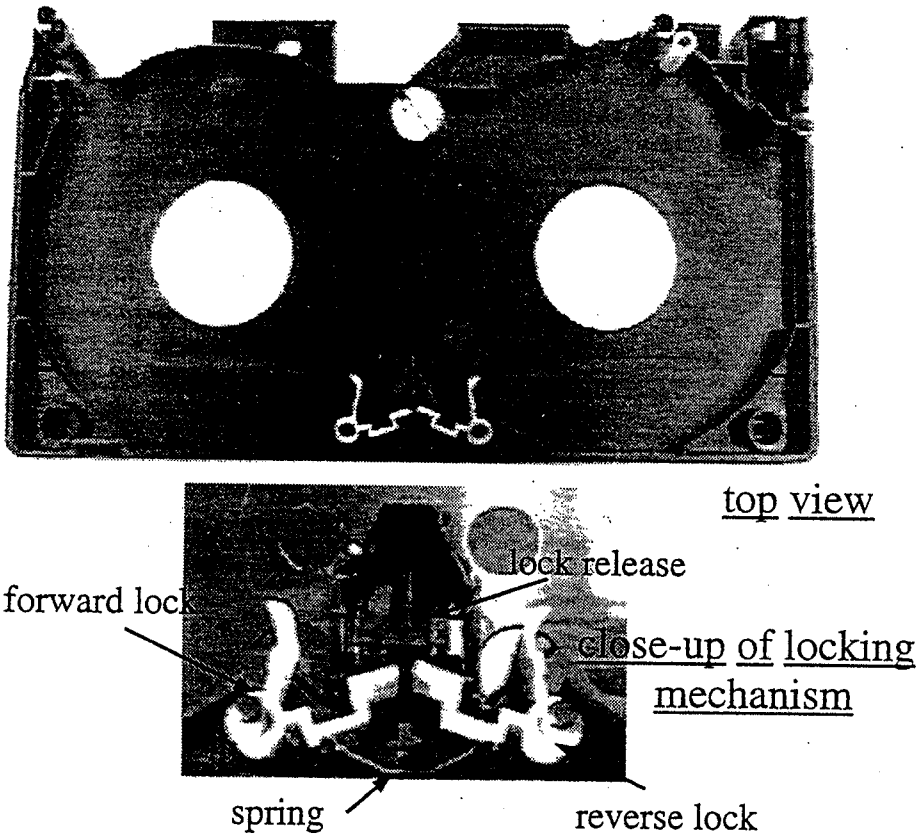

Figure 1. This figure shows a VHS cassette tape assembly that is used as an illustrative example throughout this work.

would allow a compliant robot to successfully complete the assembly.[8] The goal here is to make the task of providing this desired assembly information as natural as possible for the human designer. Therefore, the interface selected is one which provides a "virtual assembly environment" for the designer. The user of this system can see their hands in a three-dimensional relationship with the graphical CAD/CAM components of the assembly and "assemble" them together. Using this system, the user can concentrate on the high-level tasks required to complete the assembly and let the computer transform those commands into the motion of the specific robots in the workcell. This provides the designer with a natural and intuitive interface for programming robot motions without requiring any knowledge of the specific robot that is to perform the assembly. In fact, we wish to explicitly avoid the representation of any specific robot, in direct contrast to telerobotic VR systems used for remote manipulation.[9] The goal is to obtain a generic assembly strategy that is only a function of the product design which can be translated into a variety of possible workcells that might contain robots, hard automation, and/or humans. This provides the designer with immediate feedback regarding the manufacturability of his/her design as well as providing a tool for evaluating different production lines.

The principals of the system described here are illustrated by using the simple example of the VHS cassette tape shown in Figure 1. This is only an illustrative example in order to prevent the presentation from being too abstract. The procedure is identical for any other generic assembly task. To illustrate the difficulty in automatically computing an assembly strategy for even this simple assembly, consider the

tape locking mechanism shown in the close-up. To assemble the tape locking mechanism the lock release component can only be inserted after the spring and both the forward and reverse lock components are in place. Also, the lock release component must be manipulated to provide a horizontal force simultaneously against both locking components, thus compressing the spring, while being vertically inserted. Note that this assembly procedure requires the specification of an assembly sequence, a stable grasp, and a fine-motion strategy that are trivial for the mechanism's designer, but would be exceedingly difficult for the most state-of-the-art algorithm for geometric analysis. The three-dimensional path that the lock release component must take is provided by the physical interaction of the designer's hands with the graphical CAD/CAM model of the VHS cassette assembly in a virtual environment. The sequential set of three-dimensional paths for the components are then used as desired trajectories for the end effector of the robot that is to perform the actual assembly. The control of a specific robot's joints is automatically calculated by using a combination of global path planning for guaranteeing collision-free trajectories and Jacobian control for fine-motion planning.

The remainder of this article is organized as follows. Section II gives an overview of the entire experimental testbed, including both the equipment to implement the virtual assembly environment as well as the real assembly workcell. Section III gives a brief description of the CAD/CAM database used to describe the components of an assembly and their relationship to one another. Section IV presents an account of the interaction that occurs between the designer and the virtual assembly environment during the specificationof an assembly. A description of how these high-level assembly commands are then converted into specific low-level robot joint trajectories is provided in Section V. This section also discusses the software available to allow the designer to preview the resulting robot motion control commands before sending them to the actual robot. Finally, the conclusions of this work are presented in Section VI.

2. OVERVIEW OF EXPERIMENTAL SETUP

The system described here was implemented and evaluated on the experimental testbed shown in Figure 2. The testbed can be functionally divided into two main components, namely, the virtual assembly environment and the actual robot workcell. The virtual assembly environment provides the interface for the designer to interact with and evaluate his/her prospective product design while the actual robot assembly workcell provides a means of validating the efficacy of the assembly operations generated by the system.

The virtual assembly environment is centered around a SPARC ZX graphics workstation that is responsible for generating stereo images of the CAD/CAM models of the components in the assembly. These stereo images are viewed by the user through a pair of liquid crystal eyeglasses that are shuttered at 114 HZ in synchronization with the workstation. The glasses contain ultrasonic sensors to track head position/orientation and thus allow the system to appropriately modify the images generated by the workstation to improve the three-dimensional illusion. The user interacts with the component models by using an ultrasonic 6D mouse and the electromagnetic Polhemus Fastrack system, both of which provide the position/orientation of the user's hands. The Polhemus system provides a higher degree of resolution and accuracy, however, the 6D mouse simplified the specification of discrete events, e.g., grabbing or releasing an object.

The real robot assembly workcell is centered around a five-axis Adept-I manipulator that performs the actual assembly of the components into the finished product. The Adept-I is outfitted with an XGS vision system, a tool changer, and a parallel jaw gripper. It is controlled using the standard V+ robot control language which is downloaded to the robot from the workstation via a serial link. A PUMA 560 robot which is controlled in the same manner is also available in the workcell for evaluating coordinated robot motion in multiple cooperating robot workcells. It should be emphasized that the successful completion of automatically assemblying the product from the generic fine-motion strategy that is extracted from the

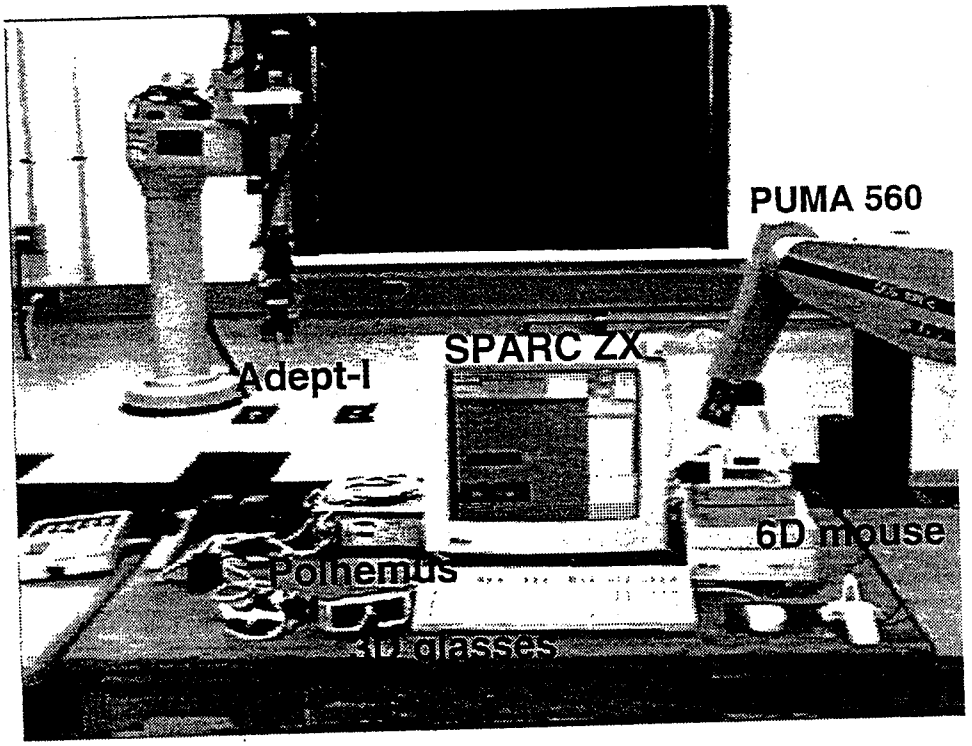

Figure 2. This figure shows a photograph of the experimental testbed for this system. The virtual assembly environment consists of a SPARC ZX workstation for image generation, 3D shuttered glasses with head tracking for stereo viewing, and a 6D mouse and Polhemus Fastrack system for hand tracking and interaction. The robot workcell consists of an Adept-I robot with an XGS vision system, tool changer, and parallel jaw gripper for performing the actual assembly. The PUMA 560 robot is available for testing cooperative assembly in multiple robot workcells.

the human designer is dependent on having some method of controlling the forces of interaction between the components being assembled. In other words, the specified trajectory is really a compliant-motion strategy as first introduced by Lozzano-Perez, et al.[8] In our testbed workcell we are unable to actively control the compliance since the robots are not currently outfitted with force/torque sensors so that we rely on a passive compliance scheme using an RCC device.

It is important to note that the virtual assembly environment with which the designer interacts is completely independent of the actual physical robot workcell that is to perform the assembly. The high-level assembly operations generated from the user's interactions with the component models are analogous to high-level computer language statements that are then compiled to machine code for a particular computer. Likewise, the system's software provides the "compilation" of the high-level assembly operations into the specific workcell platform regardless of the type or number of robots present. This feature provides portability of the high-level assembly commands and allows a comparative evaluation of various different possible platforms for the actual assembly.

3. DESCRIPTION OF CAD/CAM DATABASE

The majority of the information required by the virtual assembly environment is available from the component models stored in any typical CAD/CAM package. In particular, the virtual assembly

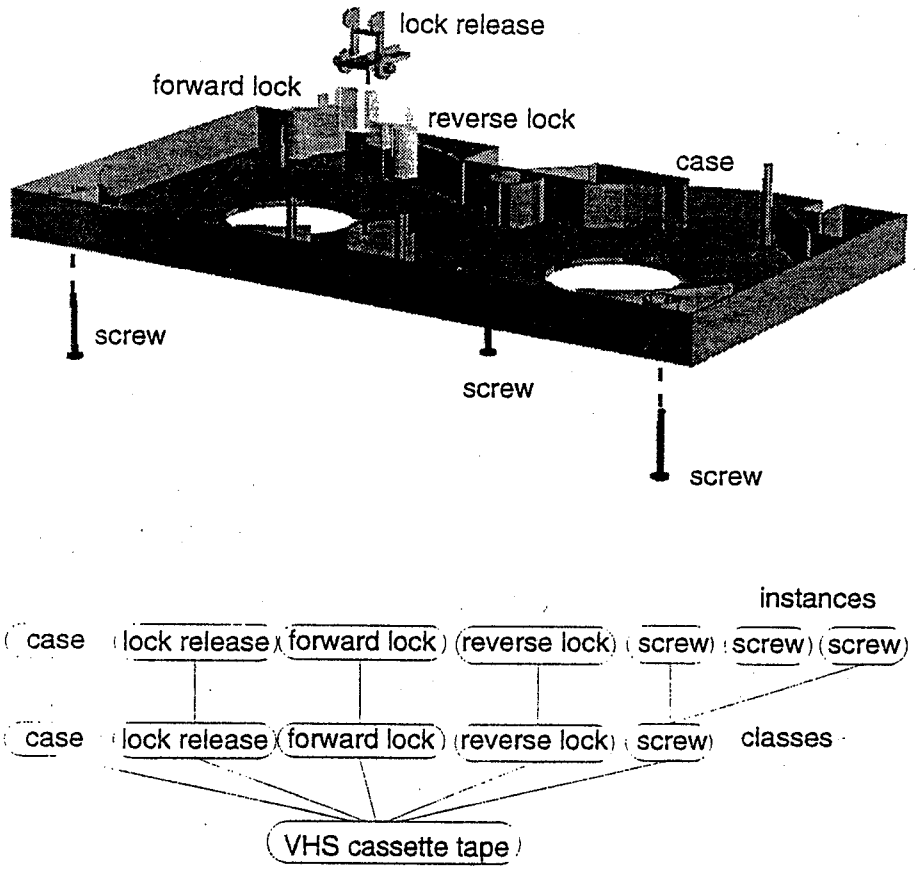

Figure 3. The upper part of this figure shows an exploded view of the CAD/CAM model for the VHS cassette tape. The lower part shows a tree representing the class hierarchy for selected components of the VHS cassette tape. (Not all parts are included in order to simplify the diagram.)

environment must have geometric information concerning the shape and location of every component in the assembly. Since many assemblies contain multiple instances of the same component, it is useful to impose a class hierarchy onto their models. As a particular example, consider the VHS video cassette shown in Figure 1. An exploded view of the CAD/CAM model for this cassette is given in Figure 3. Note that there are three identical screws, i.e. they have the same shape, but they are obviously located in different positions in the final assembly. Thus it is logical to specify a class called "screw" which contains the information common to all screws and then to specify instances of this class for information that is specific to an individual screw, such as its position. This class hierarchy of components is illustrated in the bottom half of Figure 3.

The information required by the virtual assembly environment that is common to all classes is primarily graphical information consisting of component geometry and material properties required to generate realistic images. All geometric information is specified relative to a unique class coordinate frame but is parameterized by attributes that are specific to individual instances of this class.

The particular CAD/CAM modeling package that we use is called TWIN, which is a feature-based solid modeler that uses a hybrid B-rep/CSG representation.[10] The format for its representation of

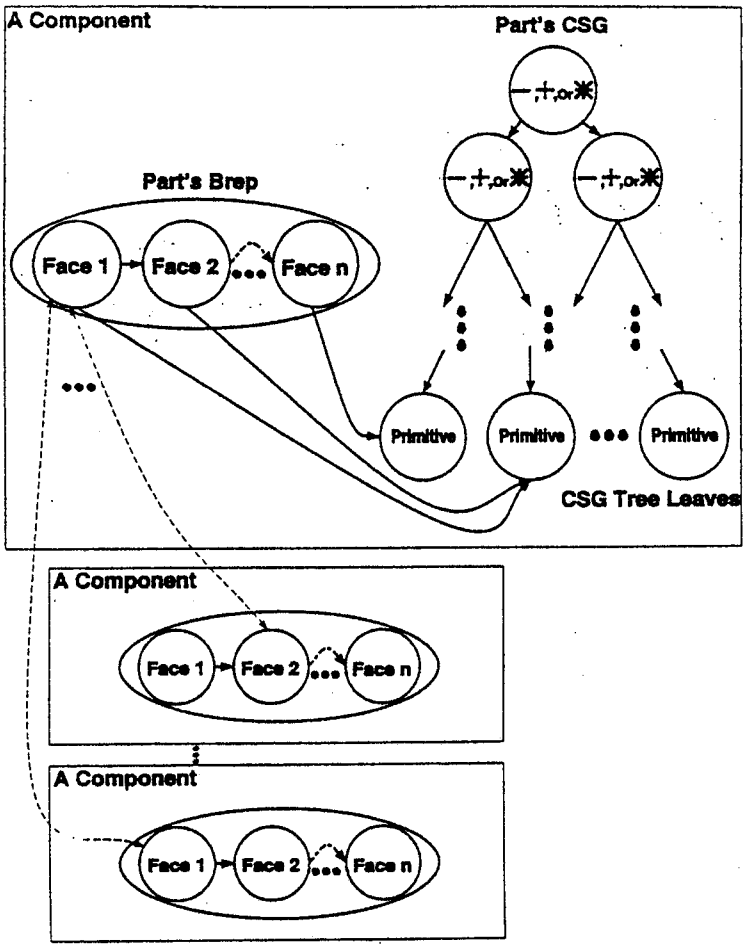

Figure 4. The CAD/CAM solid model used to define the components is a hybrid B-rep/CSG representation. Face contact information is contained between all faces, both within and between components, along with pointers to the face origin in the CSG model.[10]

component geometry is schematically illustrated in Figure 4. This representation is particularly useful for our application because it allows us to use constructive solid geometry to model the components while also maintaining a boundary representation for efficient display. The two representations are linked by pointers that identify the CSG primitive from which each boundary face originated. The CSG portion of the representation is the mechanism by which we impose the class structure. All members of a particular class must, by definition, have the same CSG tree structure. Variations between members are obtained by specifying different values for the parameters that define the CSG primitives at the leaves of the tree.

Individual instances of a class inherit all of the general characteristics of their parent class, however, specific information such as a component's position/orientation relative to the world coordinate frame is also required. The initial positions/orientations for different instances of the same class may or may not have identical values. For example, if all of the components for the VHS cassette are provided to the workcell in a parts kit then the individual screws will have different positions. However, if the screws are introduced to the workcell from a parts feeder then the initial position and orientation of all instances will be identical and are initialized as a property of this class.

It is important to note that there is one other very important piece of information available from the CAD/CAM model of an assembly, i.e., the final relative position/orientation of each component in the finished assembly. This is important because it allows the system to correctly interpret the user's manipulation of the components within the virtual assembly environment. In particular, consider a user's insertion of a pin into a shaft. If the system's interpretation of the assembly operation were to rely strictly on user input, then the user would have to insert the pin to precisely the correct depth at precisely the correct orientation. However, by knowing the ultimate destination of the user's intended insertion, the actual trajectory provided by the user can be much less precise because it can be automatically post-processed by the system as discussed in Section V. This provides the user with a natural manipulation interface without the fatigue associated with specifying extremely precise motions.

4. VIRTUAL OBJECT MANIPULATION

When a user provides the system with the CAD/CAM model of a particular product, the virtual assembly environment is initialized with the individual components of the assembly scattered throughout the virtual workspace. It is then the user's responsibility to grasp individual components and to assemble them into the final product. The grasping operation is performed by a virtual parallel-jaw gripper whose motion is controlled by the sensed motion of the user's right hand. The user may also directly grasp objects and manipulate them with his/her left hand. The process of mating two components or sub-assemblies into a single sub-assembly is described by the following eight step process where the object grasped with the right hand is denoted A and the object grasped with the left hand is denoted B.

(1) **Approach Component A** In this step the user identifies for the system the sequential order in which components are to be assembled by selecting the next component to be added to the current sub-assembly. The actual trajectory that the user follows to arrive near component A is not important, only the position/orientation of the gripper at the approach point is stored by the system. This point marks the transition from gross-motion planning, which is automatically done by the system, and fine-motion planning for which user input is utilized.

(2) **Grasp Component A** Since a component's shape may be too complicated to automatically determine a suitable grasp configuration, i.e., one that is stable and collision free, the system extracts this information from the human designer.

(3) **Designate Departure Point** Here the user lifts the grasped component A to a point where it is no longer necessary to capture the user's motion of the object for fine-motion planning. The actual path of component A to its position specified in step 5 will later be automatically determined by the global motion planner described in the following section.

(4) **Grasp Component B (optional)** The left hand is used to grasp and manipulate component B. This allows the user to specify the preferred orientation of part B to the system for the assembly process. Ideally, part B would never need to be manipulated in the actual workcell but would be placed in the preferred orientation when originally introduced to the workcell.

(5) **Specify Approach Point (A to B)** This step is similar to step 2 in that the user brings component A to a point near component B where the actual user movement will start being stored in order to assist in fine-motion planning.

(6) **Assemble Component A with B** The user manipulates component A in close proximity or contact with B to arrive at the final mated configuration. The exact trajectory of the user's hands is stored and post-processed to specify the fine motion of the robot which ultimately

performs the assembly. This step concludes with the user releasing component A.

(7) **Designate Departure Point** The system continues to store the trajectory of the user's hands as they extract the virtual parallel-jaw gripper from close proximity with the sub-assembly (A+B). In the above process, the approach and departure points are stored as 4 x 4 homogeneous transformations with respect to the appropriate local component coordinate frame with the fine-motion trajectories additionally including velocity information.

From the above description of the manipulation interface, it should be clear that the motions can be broadly separated into two categories, namely fine motion and gross motion. The fine motion requires delicate movements in a relatively localized area, such as in steps 2,3,6, and 7, whereas the gross motion is characterized by rather large movements across the entire virtual workspace, as in steps 1, 4, and 5. To deal with the conflicting requirements of these two types of motion, the system provides the user with two modes of interaction with the objects, namely position control and velocity control.

The position control method is suitable for specifying the fine motion associated with actual assembly operations because it is intuitive for the user. The component that the user is grasping will move in the same manner as his/her hand moves thus giving the impression that he/she really is grasping the component. The drawback of this intuitive control method is that it is limited by the range of the user's physical reach. While increasing the scale factor between the real and the virtual world can alleviate this problem to some extent, doing so reduces the intuitiveness of the interface as well as the resolution of the motion. To address these issues, the system switches to velocity control whenever gross motion across large regions of the virtual workspace are desired. In this mode a constant displacement of the user's hand will create a constant velocity of the virtual gripper. It is important to note that the views generated in the virtual environment are automatically adapted to deal with these two different modes. In particular, for fine motion under position control, the view angle is automatically reduced to provide a close-up view of the assembly whereas it is automatically increased to ultimately include the entireworkspace for gross motion.

5. ROBOT TRAJECTORY GENERATION

After the user has completed manipulating all of the components into the final desired assembly, the system will have accumulated a sequential profile of alternating fine and gross motion data. The system then processes this data into a form that can be used to control the robots that are to perform the actual assembly. While the fine and gross motion data are processed differently, the output in each case is a set of trajectories in joint space for each robot in the workcell that can be sent directly to the robot controller.

5.1 Fine Motion

The steps required to process the fine-motion data acquired from the human user's hands into robot joint angle trajectorieswill be illustrated through a specific example. Consider the insertion of the lock release component shown in the close-up of Figure 1. This lock release component is shown at its approach point in Figure 5 which is the start of a fine-motion phase. The actual motion data for the assembly operation acquired from the user is shown in part (a) of the figure. Note that the general characteristics required for a successful mating of the various components is clearly visible in the captured trajectory. In particular, the motion starts with a lowering of the lock release component in the y direction from the approach point, followed by a motion in z that puts it in contact with the forward and release lock components, compressing the spring (see Figure 1), before it is completely lowered into its final position. In addition to these desirable characteristics, however, there are several undesirable artifacts present as well, primarily due to the jerky and inconsistent motion of the human.

To extract only those characteristics required for a successful assembly the raw motion data is first

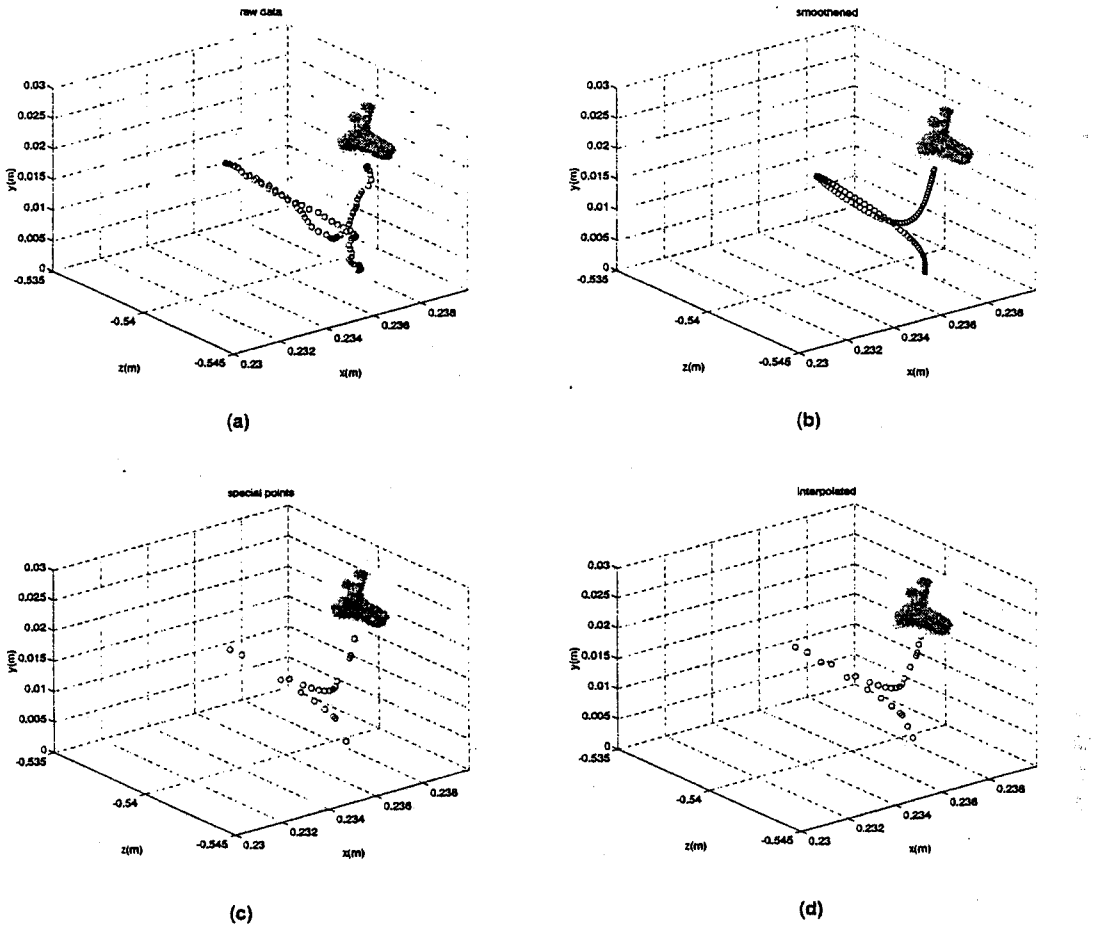

Figure 5. This figure illustrates the post processing performed on fine-motion trajectories obtained from directly tracking a user's manipulation of the virtual objects. This particular example is from the insertion of the lock release component (see Figure 1), which is shown at its approach point. In (a) the raw data obtained from the user's hand motion is shown. Filtering of this data results in the smoothed trajectory illustrated in (b). The trajectory is then compressed by identifying key locations along the trajectory as illustrated in (c). The key locations can then be interpolated with any desirable velocity profile in order to meet the requirements of the specific robot that is to perform that actual assembly (d).

filtered to remove the high frequency oscillations. This is done by applying an FFT to the data and then applying a low-pass filter in the frequency domain. The cut-off frequency of this filter is variable to accommodate different users (10Hz is used in Figure 5). The filtered signal is then transformed back into the time domain by performing an inverse FFT. The resulting trajectory for this example is shown in Figure 5(b).

The next step in processing the trajectory is to extract the geometric properties of the fine-motion strategy from the speed at which they were performed by the human. This is done by reparameterizing the filtered hand motion to obtain an equal arc length representation of the path. This path is further compressed by decreasing the number of locations as the radius of curvature increases. Our approach is to vary the rate at which locations are stored from a minimum of 10% of the total arc length in areas of infinite curvature, i.e., straight line motion, to a maximum of every 1% of the total arc length in areas of minimum curvature. For the example in Figure 5(c) the total arc length for the fine-motion strategy is approximately 4 cm so that in the straight portions of the trajectory key locations are separated by

approximately 4 mm and key locations are never closer than 0.4 mm.

The required trajectory to perform this phase of the assembly is now available as a discrete set of n homogeneous transformations for the gripper that is to carry the component, denoted $x(k_0)$ through $x(k_n)$. From this representation it is easy to calculate individual joint set points for any robot's controller. In particular, given a specific robot with a particular limit on its maximum tool velocity, the key points of the trajectory are converted into a desired hand velocity $\dot{x}(t)$. The robot joint velocities $\dot{\theta}$ required to achieve this trajectory are then calculated by solving

$$\dot{x} = J\dot{\theta}$$

where J is the manipulator Jacobian for this particular robot. Integrating $\dot{\theta}$ and sampling at the control cycle time of the robot being considered provides the joint positions $\theta(t)$ that are used by the commercial robot controller. Details of this inverse kinematics process are provided in Macijewski and Klein.[11]

5.2 *Gross Motion*

Each fine-motion trajectory that is performed is preceded by a gross motion that positions the robot's gripper at the appropriate approach point (see steps 1 and 5 in Section IV). The data obtained from the user of the virtual assembly environment for a gross motion phase consists only of a starting homogeneous transformation for the gripper S_i, and a goal homogeneous transformation G_i corresponding to an approach point. It is important to appreciate why these gross motion trajectories are not directly obtained from the motion of the human user as he moves the virtual gripper throughout the virtual assembly environment. The first reason is that the virtual assembly environment is intentionally made independent of the actual robot workcell that is to perform the assembly. Thus the user has no knowledge of any ancillary equipment that may be located within the real workcell with which collisions must be avoided. The advantages of this approach are that the user can intuitively assemble the product and then later evaluate different realizations of possible workcells without repeating the virtual assembly process. Second, it is very difficult for a human to manually determine a collision-free trajectory for an articulated robot.

To deal with the issue of generating a collision-free robot joint angle trajectory $\theta(t)$ from only a start and goal configuration, a global path planning algorithm based on the approach presented in Maciejewski and Fox[12] is used. This algorithm takes all of the physical objects present in the workcell of the real robot and transforms them into the joint space coordinates of the robot, also commonly referred to as configuration space. The algorithm then analyzes the configuration space to determine which portions of it are connected, which physically represents all possible collision-free paths within the workcell. When the algorithm receives a start and goal configuration for a gross motion trajectory, it simply maps these configurations into their representations in the robot's configuration space, validates that these two configurations are actually connected, and then generates a collision-free joint angle trajectory $\theta(t)$ that can be used by the real robot's joint controller. This process is perhaps best illustrated through an example. Consider the bottom half of Figure 6 which shows the top view of the Adept-I robot in a workcell that consists of four polyhedral obstacles. The top half of Figure 6 is a map of the configuration space for the Adept-I in which every point represents a unique configuration of the robot. Once the obstacles in the workspace are mapped into the configuration space the process of determining a collision-free gross motion is reduced to connecting the start configuration S with the goal configuration G without intersecting any of the obstacles. One such path that was automatically generated by the system is shown in the top half of Figure 6 using a dotted line, with the resulting robot motion shown in the workcell. The average total time for calculating such collision-free motions on the SPARC ZX is on the order of a few milliseconds.

5.3 *Preview and Robot Control*

Robot joint angle trajectories, $\theta(t)$, are the output from processing both the fine and the gross motion segments. After processing all of the motion segments successive joint angle trajectories are concatenated

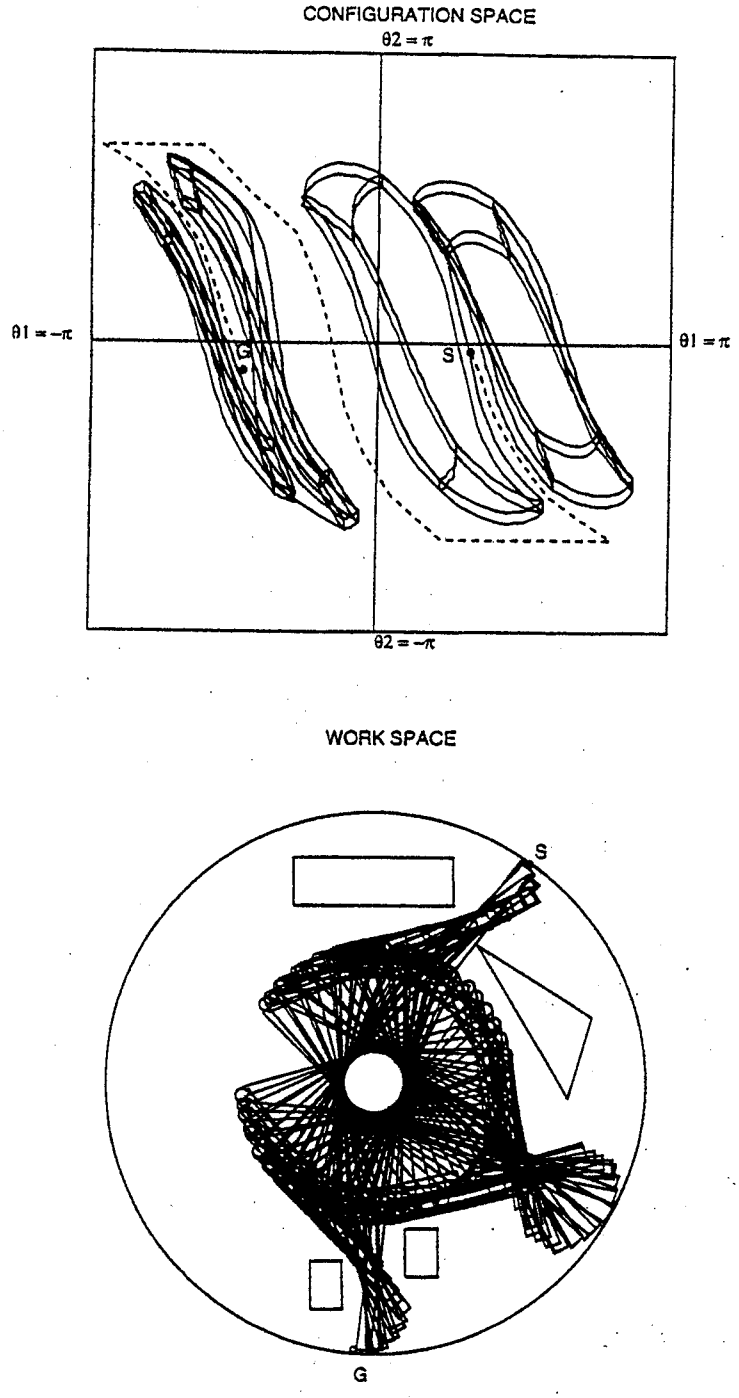

Figure 6. This figure illustrates the algorithm used to automatically calculate collision-free gross motions for a robot. The bottom half of the figure shows the top view of the Adept-I robot in a workcell containing four polyhedral obstacles. The top half shows the configuration space for this robot along with the desired start (S) and goal (G) configurations. The algorithm automatically determines the robot motion, shown with a dotted line, that avoids collisions with the four obstacles that have been transformed into configuration space. The motion of the robot represented by the dotted-line path is shown in the workcell.

into a single robot joint motion profile for the entire assembly. This can be done because the gross and fine motion components are designed to be continuous at their transitions, i.e., the approach and departure points. If multiple robots are used, each robot will have its own respective joint motion profile which is synchronized with the others to avoid collisions. These robot motion profiles can now be previewed in a graphical simulation of the virtual workcell to validate the resulting robot motions.[11] At this point the user can modify the virtual workcell to change or add robots in order to evaluate the efficiency of different workcell configurations by simply rerunning the trajectory generation software. Once the designer is satisfied with the motions of the robots within the virtual workcell, the robot joint angle trajectories are translated into the robot's programming language. The control of the gripper motion, identified by the discrete events in step 2 and 6 of the object manipulation process, is provided by a call to either the open or close gripper subroutines.

6. CONCLUSION

This article has described a prototype system that has been implemented to assist design and manufacturing engineers in automating the assembly process. The system provides a virtual assembly environment that allows the design engineer to manipulate the CAD/CAM models of his/her prospective design and then automatically preview the assembly of that product in a prospective robot workcell and ultimately generate the robot controller commands for the real physical robot workcell. The major contribution of this work is a technique for extracting a generic assembly plan based solely on product geometry that specifies (1) a preferred order of assembly, (2) stable grasp locations for all components, and (3) a fine-motion strategy that would allow a compliant robot to successfully complete the assembly. This procedure explicitly avoids the representation of any specific robot so that the resulting assembly plan is generic and can be applied to a variety of possible workcells that might contain robots, hard automation, and/or humans. This is in direct contrast to virtual environments applied to remote teleoperation.[9]

The virtual assembly environment described here appears to be an intuitive interface for generating assembly plans. Initial performance by human user's did exhibit noticeable hesitance during the many phases of the assembly process but this is attributable to the novelty of the interface. Future planned improvements in the interface include a more sophisticated dynamic model for the components as well as force feedback to the user,[13] however, maintaining the rapid response time of the system is of primary importance.

REFERENCES

1. Gruver, W.A., B.I. Soroka, J.J. Craig, and T.L. Turner, "Industrial robot programming languages: A comparative evaluation." *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-14, no. 4, July/August 1984, pp. 1-7.
2. Liegeois, A., P. Borrel, and E. Dombre. "Programming, simulating and evaluating robot actions," in H. Hanafusa and H. Inoue, editors, *Robotics Research: The Second International Symposium* , MIT Press, 1985, pp. 309-316.
3. Hornick, M., and B. Ravani. "Computer-aided off-line planning and programming of robot motion." *International Journal of Robotics Research* , Vol. 4, no. 4, Winter 1986, pp. 18-31.
4. Pertin-Troccaz, J. "Task level robot programming: On the HANDEY system," in O. Khatib, J. J. Craig, and T. Lozano-Perez , editors, *Robotics Review* 2, MIT Press, 1992, pp. 31-36.
5. Takahashi, T., and H. Ogata. "Robotic assembly operation based on task-level teaching in virtual reality." *Proc. IEEE 1992 InternationalConference on Robotics and Automation* (Nice, France), May 10-15 1992, pp. 1083-1088.
6. Takahashi, T., H. Ogata, and S.Y. Muto. "A method for analyzing human assembly operations for use in automatically generating robot commands." *Proc. IEEE 1993 International Conference on Robotics and Automation* (Atlanta, Georgia), May 2-6 1993, pp. 695-700.
7. Ro, P.I., and B.R. Lee, "An optimum path and posture planning for fixtureless assembly." *Proc. IEEE 1993 International Conference on Robotics and Automation*, (Atlanta, Georgia), May 2-6 1993, pp. 808-813.

8. Lozano-Perez, T., M.T. Mason, and R.H. Taylor. "Automatic synthesis of fine-motion strategies for robots." *International Journal of Robotics Research* , Vol. 3, no. 1, 1984, pp. 3-23.
9. American Nuclear Society, *ANS 6th Topical Meeting on Robotics and Remote Systems*, Monterey, CA, February 5-10 1995.
10. Anderson, D.C., and T.C. Chang. "Geometric reasoning in feature-based design and process planning," *Computers & Graphics*, Vol. 14, no. 2, 1990, pp. 225-235.
11. Maciejewski, A.A., and C. A. Klein. "SAM—Animation software for simulating articulated motion." *Computers & Graphics*, Vol. 9, no. 4, 1985, pp. 383-391.
12. Maciejewski, A.A., and J.J. Fox. "Path planning and the topology of configuration space." *IEEE Transactions on Robotics and Automation*, Vol. 9, no. 4, August 1993, pp. 444-456.
13. Burdea, G.C., and N.A. Langrana. "Virtual force feedback: Lessons, challenges, future applications." *Proc. Advances in Robotics*, Annual Meeting of The American Society of Mechanical Engineers (Anaheim, CA), November 8-13 1992, pp. 41-47.