A Virtual Object Manipulation Interface for Automated Assembly Programming

Akihiro Sato* Anthony A. Maciejewski†
*Production Engineering Development Lab., NEC Corporation,
3-484, Tsukagoshi, Saiwai-ku, Kawasaki, Kanagawa, Japan 210
†Purdue University, 1285 Electrical Engineering Bldg.,
West Lafayette, Indiana 47907-1285

Abstract— **This work describes the implementation of a novel robotic workcell programming interface that allows an assembly designer to obtain immediate feedback regarding the manufacturability of his 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.**

I. 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 the human effort required to reprogram workcells for different tasks. The advent of programming languages for manufacturing and automation [1] along with graphical simulation capabilities [2] mediated some of these difficulties. Recently, the graphical interaction associated with assembly planning has been enhanced to provide virtual environments for planning automated assembly [3]. It is this progression of interaction with the prospective product design to assess and facilitate its manufacture 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 the necessary knowledge for automating the assembly process. The goal here is to make the task of providing the 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.

The principals of this system are illustrated by automatically assembling a VHS cassette tape (see Fig. 1) using a SCARA-type robot. To assemble the tape locking mechanism, the lock release component should be inserted only 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 while this assembly procedure is trivial to determine for the mechanisms designer, it is exceedingly difficult to determine strictly from an automated analysis of the mechanism. The three-dimensional path that the lock release component must take is thus provided by the physical interaction of the designer's hands with the graphical CAD/CAM models of the entire 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 the robot that is to perform the actual

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.

0-7803-2129-4/94 $3.00 © 1994 IEEE

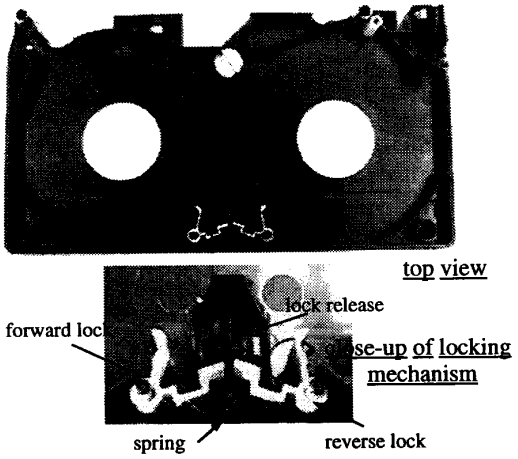


Fig. 1. This figure shows a VHS cassette tape assembly which is used as an illustrative example throughout this work.

assembly. The control of the robot's joints is then automatically calculated by the computer without any user intervention.

The remainder of this paper is organized as follows. Section II gives an overview of the entire experimental testbed. 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 specification of a products 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. Finally, the conclusions of this work are presented in section VI.

II. Overview of Experimental Setup

The system described here was implemented and evaluated on the experimental testbed shown in Fig. 2. The testbed can be functionally divided into two main components, namely, the virtual assembly environment and the actual robotic workcell. The virtual assembly environment provides the interface for the designer to interact with and evaluate his prospective product design while the actual robotic 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 which is responsible for generating stereo images of the CAD/CAM models of the components of 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 are also outfitted with ultrasonic sensors in order to track eye position and orientation and thus allow the system to appropriately modify the images generated by the workstation in order to improve the three-dimensional illusion. Interaction with the component models is provided through an ultrasonic 6D mouse and the electromagnetic Polhemus Fastrack system, both of which provide a means of tracking the position and orientation of the user's hands. The Polhemus system provides a higher degree of resolution and accuracy, however, the 6D mouse was required in order to simplify the specification of discrete event transitions, such as grabbing or releasing an object, by using its buttons.

The real robotic 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 a 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 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.

III. 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 environment must have geometric information concerning the shape and locations 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 Fig. 1. An exploded view

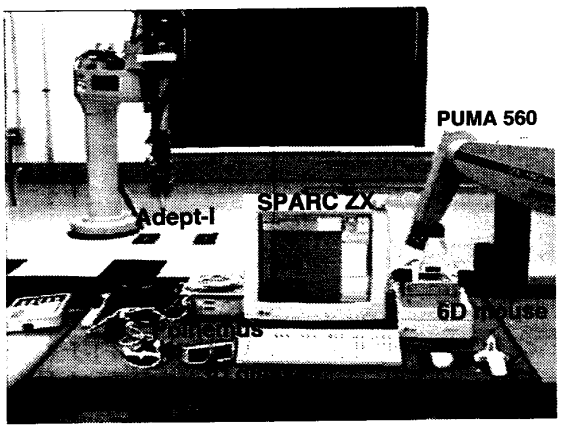


Fig. 2. This figure shows a photograph of the experimental testbed for this system. The virtual assembly environment consists of a SPARC ZX workstation for image generation, 3D shuttered glasses with head tracking for stereo viewing, and a 6D mouse and Polhemus Fastrack system for hand tracking and interaction. The robot workcell consists of an Adept-I robot with a 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.

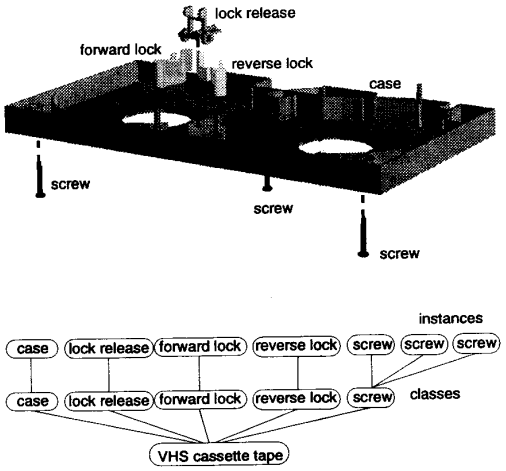


Fig. 3. The upper part of this figure shows an exploded view of the CAD/CAM model for the VHS cassette tape. The lower part shows a tree representing the class hierarchy for selected components of the VHS cassette tape. (Not all parts are included in order to simplify the diagram.)

of the CAD/CAM model for this cassette is given in Fig. 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 Fig. 3.

The information required by the virtual assembly environment that is common to all classes is primarily graphical information consisting of component geometry and visible material properties required to generate realistic images. The geometric information is all specified relative to a unique class coordinate frame but is possibly parameterized by attributes that are specific to individual instances of this class. The geometric information is ultimately transformed into a boundary representation for rendering the objects, however, it is primarily stored in a more useful CSG format wherever possible.

Individual instances of a class naturally inherit all of the general characteristics of their parent class. However, specific information such as a component's position and orientation relative to the world coordinate frame are also required. Note that initial positions and 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 could all be 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 and orientation of each component. 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 an 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.

IV. Virtual Object Manipulation

To "assemble" a prospective product, the user loads the CAD/CAM model of the design into the system and then manipulates their graphical models with his hands. Grasping of the components 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 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 objects denoted A and B are grasped with the right and left hand, respectively.

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 add to a current sub-assembly. The actual trajectory that the user follows to arrive near component A is not important, only the position and 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 be later 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 is simply to allow the user to specify for the system the preferred orientation of part B 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 are 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.

The above description defines two types of motion, i.e., fine motion which requires delicate movements in a relative localized area (steps 2,3,6, and 7), and gross motion which is characterized by rather large movements across the entire virtual workspace (steps 1, 4, and 5). To deal with the conflicting requirements of these two types of motion, the system provides two modes of interaction with the objects, namely position or 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 hand moves thus giving the impression of really grasping the object. 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. Here 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 of the virtual environment must also be automatically adapted to deal with these two different modes. In particular, for fine motion under position control, the view angle is automatically reduced in order to provide a close up view of the assem-

bly whereas it is automatically increased to ultimately include the entire workspace for gross motion.

V. Robot Trajectory Generation

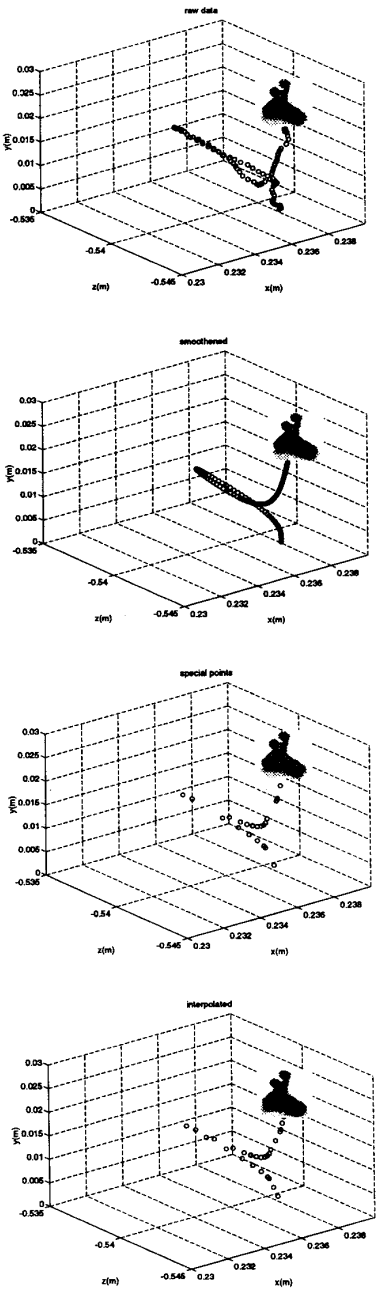
After the user has completed manipulating all of the components into the final desired assembly, the system is left with a sequential profile of alternating fine and gross motion data. The system must then processes this data into a form that can be used to control the robots that are to perform the 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.

A. Fine Motion

The steps required to process the fine motion data acquired from the human user's hands into robot joint angle trajectories will be illustrated through a specific example. Consider the insertion of the lock release component shown in the close-up of Fig. 1. This lock release component is shown at its approach point in Fig. 4 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 thus compressing the spring (see Fig. 1), before it is completely lowered into its final position. in addition to these desired 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 filtered to remove the high frequency oscillations. The resulting trajectory for this example is shown in Fig. 4(b). The data is then further compressed by capturing only key features in the trajectory. These key features are identified by examining the filtered trajectory and determining locations where the rate of change in curvature surpasses a threshold. The resulting key points are illustrated in Fig. 4(c).

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



(a) (b) (c) (d)

Fig. 4. Post processing performed on the fine motion trajectory obtained from a user's insertion of the lock release component (see Fig. 1). The raw data obtained from the user's hand motion is shown in (a). Filtering of this data results in the smoothed trajectory (b). Compression by identifying key locations along the trajectory is illustrated in (c). Appropriate interpolation then satisfies the physical constraints of the robot performing the assembly (d).

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 most commercial robot controllers. Additional details of this inverse kinematics process are provided in [4].

B. 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 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 simply a start and goal configuration, a global path planning algorithm based on the approach presented in [5] 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 this configuration space to determine which portions of it are connected, this physically represents all possible collision-free paths within the workcell. Thus when the algorithm receives a start and goal configuration for a gross motion trajectory, it can simply map these configurations into their representations in the robot's configuration space, validate that these two configurations are actually connected, and then generate a collision-free joint angle trajectory $\theta(t)$ that can be used by the real robots joint controller.

C. Preview and Robot Control

After processing all of the fine and gross motion segments, successive joint angle trajectories are concatenated into a single continuous collision-free joint motion profile for each robot that is to cooperate in the assembly process. These robot motion profiles can now be previewed in a graphical simulation of a virtual workcell [4] and potential changes to the workcell such as adding or changing robots can be quickly evaluated by only rerunning the trajectory generation software. Once the designer is satisfied with a specific workcell the system automatically translates the desired robot motion into the robot programming language to perform the assembly.

VI. Conclusion

This article has described a prototype system which has been implemented to assist design and manufacturing engineers in automating the assembly process. While the interface to the virtual assembly environment is intuitive, the initial user performance did exhibit noticeable hesitance during the many phases of the assembly process. This is probably attributable to the novelty of the interface and it is expected that user proficiency will vastly increase with wide-spread utilization.

References

[1] W. A. Gruver, B. I. Soroka, J. J. Craig, and T. L. Turner, "Industrial robot programming languages: A comparative evaluation," *IEEE Trans. Syst., Man, Cybern.*, SMC-14(4):1-7, July/Aug. 1984.

[2] M. Hornick and B. Ravani, "Computer-aided off-line planning and programming of robot motion," *Int. J. Robot. Res.*, 4(4):18-31, Winter 1986.

[3] T. Takahashi and H. Ogata, "Robotic assembly operation based on task-level teaching in virtual reality," *Proc. 1992 IEEE Int. Conf. Robot. Automat.*, pp. 1083-1088, (Nice, France), May 10-15 1992.

[4] A. A. Maciejewski and C. A. Klein, "SAM—Animation software for simulating articulated motion," *Computer Graphics*, 9(4):383-391, 1985.

[5] A. A. Maciejewski and J. J. Fox, "Path planning and the topology of configuration space," *IEEE Trans. Robot. Automat.*, 9(4):444-456, Aug. 1993.