Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Robotic Assembly Operation based on Task-Level Teaching in Virtual Reality

Tomoichi Takahashi and Hiroyuki Ogata
NTT Human Interface Laboratories
Autonomous Robot Systems Laboratory
3-9-11 Midori-Cho, Musashino-Shi, Tokyo 180, JAPAN

Abstract

We propose a robot teaching interface which uses virtual reality. Our teaching method provides a user interface with which a novice operator can easily direct a robot. The operator performs the assembly task himself in a virtual workspace generated by a computer. The operator's movements are recognized as robot task-level operations by using a finite automaton. The system interprets the recognized operations into manipulator-level commands using task dependent interpretation rules and a world model. A robot executes the assembly task in the real workplace by replicating the operator's movements in the virtual workspace.

1 Introduction

Two methods have been used to program robot motion: the teaching play-back and programming. They require either a skilled worker or a robot programmer.

The amount of robot teaching data is one factor of a good interface. We assume that the amount of teaching required to manipulate a robot depends on the robot's intelligence. We define the intelligence of the robot by what kinds of sensors or how many motion control routines the robot has. For example, for robots with vision sensors for locating an assembly part, we can reduce the amount of teaching position data by specifying points in the motion with symbolic labels rather than three-dimensional positions. With macro command interpretation ability, many motion control programs can be replaced with simple commands.

Another factor of a good interface is how a human operator can direct a robot without having specific task skills or robot programming knowledge. Asada et al. [1] proposed a method of directly teaching human worker skills in a grinding task. In assembly tasks, Kuniyoshi et al. [2] and Hamada et al. [3] described an auto-programming method in which a human instructor demonstrates a task to a robot by performing it himself. Their methods of recognizing the operator's movements are based on vision and require much computer power. A new kind of tele-robotics interface has been proposed [4]. Wearing a head-mounted display which generates virtual reality, the operator manipulates a robot as if he were at the robot's location. This method requires that the operator perform the task, but it shows that an interface using virtual reality provides a good media for robot manipulation.

We propose an assembly-task teaching method which uses virtual reality to teach an intelligent robot. The method provides an interface similar to teaching by a human expert, who teaches a novice his skill by demonstrating the task himself. Section 2 introduces a teaching interface which uses the operator's movements in a virtual workspace. Section 3 describes the method of recognizing operator movements. In section 4, three functions for executing the assembly task in the actual workplace are discussed. The results of the experiments and future development are presented in section 5.

2 Interface using Virtual Reality

The operator demonstrates an assembly task in a computer generated virtual reality space, which we call the *teaching environment.* Virtual reality is a computer-based virtual environment in which operators interact with the system by means of a three-dimensional pointing device and a hand gesture interface device [5].

There are three kinds of interfaces which use virtual reality. One involves no interaction with the physical world; the other two do involve interaction with the physical world. In one of the latter two types of interface, an operator's action in the teaching environment results in actions in the physical world. The

0-8186-2720-4/92 $3.00 ©1992 IEEE

operator monitors the action's effect in the teaching environment. Telerobotics is involved in this type. Our method is the third interface type. The operator demonstrates a sequence of assembly operations in the teaching environment without considering the robot which performs the task autonomously in the physical world, which we call the *task environment*. The operations in the teaching environment and the actions in the task environment are separated in time and space.

2.1 Teaching and execution

The system consists of three phases.

phase 1: Setting the teaching environment
The operator specifies the assembly components. He selects part names and their color attributes from a menu. The color attribute is used to identify the part in the task environment in phase 3. After selecting the component, he specifies its position by clicking the mouse button.

phase 2: Performing in the teaching environment
The operator wears a hand gesture interface device and stereoscopic glasses and performs the assembly task in the teaching environment. The teaching environment is created on the basis of the layout information specified in phase 1. The system recognizes the operator's movements from the interface device data and the layout information of the teaching environment. The recognition results are a sequence of task-level commands.

phase 3: Execution in the task environment
The system interprets the command sequence into robot commands using task-dependent interpretation rules defined in advance. The part's geometrical data such as position and orientation are bound to the physical data in the task environment by means of sensor data. Using the interpreted robot commands, the robot performs the assembly task in the task environment, which does not necessarily have the same layout as the teaching environment of phase 2.

2.2 System overview

Figure 1 shows the system configuration. The teaching environment is implemented on a workstation (IRIS^{TM}) and makes use of the VPL DataGlove^{TM}. DataGlove outputs the 10 finger joint angles, orientation and three-dimensional position of the operator's hand. These values represent the operator's hand

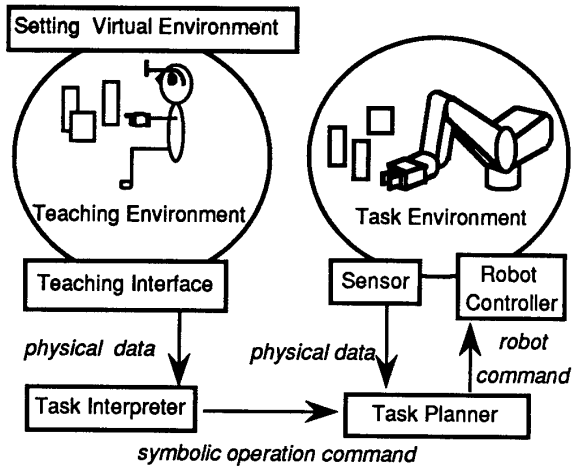


Figure 1: System Configuration

(a) in open space (b) near an object

Figure 2: Handshape recognition

shape and movements. Task-level commands are recognized from the operator's movements using a task dependently defined finite automaton. The commands are transferred to a LISP workstation (ELIS^{TM}) and are interpreted into manipulator-level commands. The robot (MOVEMASTER^{TM}) has two sensors — a color image processing system and a force-torque sensor — and performs the assembly task in the task environment.

3 Recognition in Virtual Reality

Figures 2(a) and 2(b) show the same hand shape *closed hand*, but the purposes of the hand movements are different. The hand movement in figure 2(b) is recognized as *grasp the object*. We modified a method which generates task-level commands from hand movements in a teaching environment [6].

The modified method separates hand shape recognition from hand gesture interpretation. The hand shape recognition is independent of the assembly task, and the recognized results (Σ) become the input of an automaton. The automaton state (Q) is defined task-dependently. The effective assembly operations be-

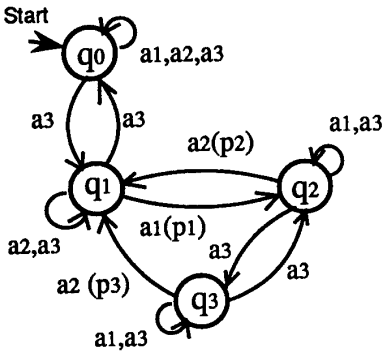


Figure 3: A Hand Gesture Model

come a subset of the operator hand shapes accepted by the automaton.

In our system, the robot has a two-digit gripper. The automaton, M=$< Q, \Sigma, \delta, q0, F >$, which accepts an operator's pick and place operation, consists of the following.

Q={q0, q1, q2, q3}

q0=grasping no object in open space (initial state)
q1=grasping no object near another object(s)
q2=grasping an object in open space
q3=grasping an object near another object(s)

Σ={a1, a2, a3}

a1=close hand shape
a2=open hand shape
a3=moving hand action

δ: state transition function (Figure 3)

F: final state (q0, q1)

The effective pick and place operations (p1, p2, p3) correspond to three state transitions. The *pick_up* an object, *place* the grasping object *on* the table, and *place* the grasping object *on* an object(s) operations respectively correspond to δ(q1,a1)=q2, δ(q2,a2)=q1 and δ(q3,a2)=q1.

Figure 4 shows the sequence of the operator teaching the robot to insert a peg into a hole in a teaching environment. The rectangular objects in upper left part of the picture of the initial state represent the virtual robot grippers. Visual feedback occurs when the grippers or the grasped object collide with another object. The gripper or the grasped object turns white and the other object turns gray. The recognized operations are shown under each picture.

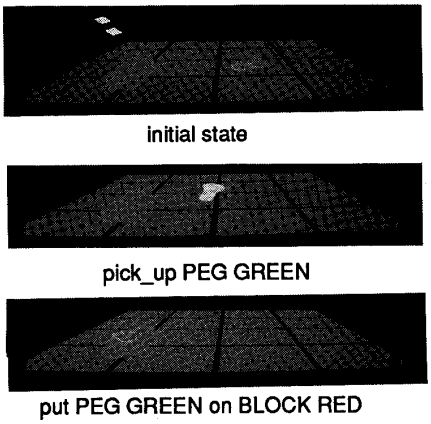


Figure 4: Operator's actions in teaching environment and recognized commands

4 Execution in the Task Environment

Three functions are needed to manipulate the robot in a task environment which is not the same as the teaching environment.

1) Mapping the teaching environment into the task environment
The operator sets up the teaching environment to simulate the task environment. The number and the kinds of parts are same, but the sizes and the positions of the parts are different in the two environments. At the time of task execution, the symbolic representation of the part's position must be converted into the physical values of the task environment.

2) Interpreting task-level operations into robot commands
The task-level commands recognized from the operator's movements are independent of the robot performing the task. The task-level commands are interpreted into the commands of the robot.

3) Replanning the sequence of operations
The operator demonstrates the assembly task in the teaching environment without considering the robot mechanism. The robot does not have the same degrees of freedom as the operator, so the assembly operation sequence must be replanned according to the robot's mechanical constraints.

The overall structure of the execution system is shown Figure 5.

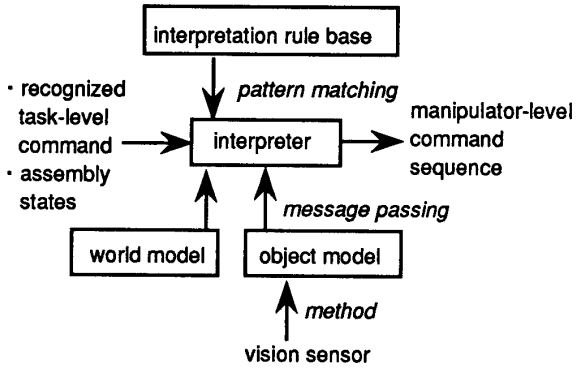


Figure 5: Block diagram of execution system

4.1 World model representation and mapping to the actual layout

A characteristic of task-level commands is that they are independent of the manipulated parts and the robot performing the task [7]. The interpretation of the task-level command varies for different parts. For example, *pick_up red object* is interpreted differently for red pegs and red poles, because the grasping positions are different for the two objects. This difference comes from the part's function as well as its shape. The part's name represents its function. The name *peg* implies an inserting operation, thus the grasping position must be suited to the inserting operation.

Object-oriented representation is a natural way to represent the assembly parts and to encapsulate the differences in attributes. The object attributes are represented as instance variables in an object-oriented representation. The instance variable has a procedure called *method*, which calculates the values. Sending the same message to different objects returns the values suited to each object.

4.2 Interpretation into robot commands

The interpreting process involves representation of an operator's knowledge concerning performance of the operation. For example, the *pick_up red object* command is a composite operation consisting of *move to red object* and *grasp the object.* If-then rules are useful means of representing such knowledge.

The task-level command interpretation process involves pattern matching of if-then rules. The pattern matching continues until all commands in the matching work area become robot commands.

The syntax of rule representation is the following. The parts in brackets { } can be omitted.

```
(defrule move_to_pole
  ( pattern  (move_to pole ?A))
  ( into     (MP  ?X))
  ( where  (setq ?X [?A position-get])))

(defrule put_ring_to_pole
  ( pattern (put ring ?A  to pole ?B))
  ( into     (move_along_down ?B)
             (GO)
             (move_along_up ?B)))
```

Capital letters indicate manipulator commands.
MP: Move Position GO: Gripper Open

Figure 6: Examples of interpretation rules

(defrule comment

(pattern: defines higher-level command name)

(into: sequence of lower-level commands performing the defined command)

{(where: specifies the variables in < *into* > part by message-passing to the object) }

{(mode: specifies the allowable range of the force-torque sensor value)})

The example of rule representation is shown in Figure 6. MP in < *into* > part stands for the *move position* command of the robot and the variable X is object A's position. The binding of variable X to the actual object position is specified by (setq ?X [?A position-get]) in the < *where* > part. [?A position-get] means sending a *position-get* message to object A. *Move_to red pole* is interpreted into a *MP x, y, z, pitch, roll* command where x, y, z are the red pole's position in the robot's coordinates and pitch and roll are the end-effector's orientation.

4.3 Replanning the sequence of operations

The system performs the task in the task environment according to the sequence of movements in the operator's demonstration. However, replanning the sequence of operation is made necessary by the robot's mechanical constraints. For example, Figure 7 shows an operator's sequence of placing blocks. First, block A and block B are placed on the table (op1, op2). Block C is then placed on blocks A and B (op3). Block D is placed under block C by setting it on the surface and pushing it (op4). A robot with limited degrees of freedom cannot do op4 after op3. Block C becomes an obstacle to placing block D (Figure 8).

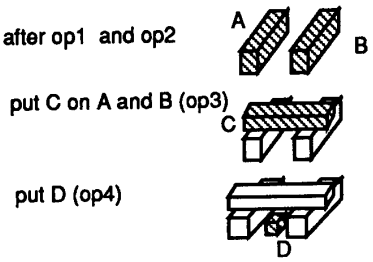


Figure 7: An operation sequence of placing blocks

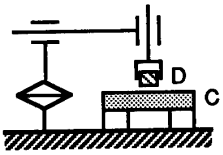


Figure 8: Operation by a robot with 3 DOF

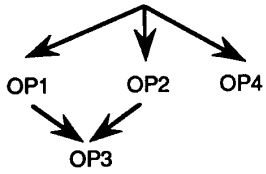


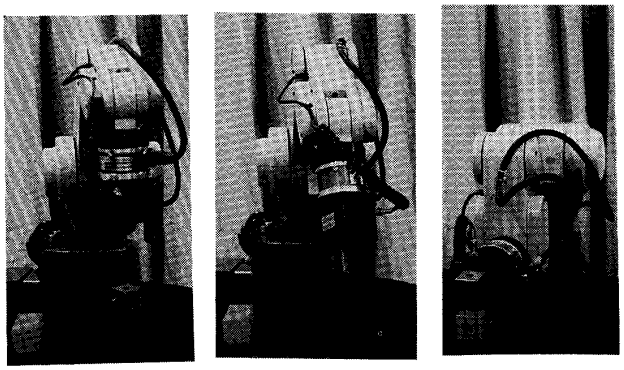
Figure 9: Constraint-network of operations

The feasibility of the robot performing the required operation sequence is checked. If the check fails, the sequence is replanned according to the constraints on operations. A constraint network is created from the recognized task-level command. Figure 9 shows the constraint network of the above example.

The arrows in the network show that the root command must be prior to the top command. In this case, op1 and op2 must be performed before op3. The precedence among op1, op2 and op4 are same. The system changes the operation sequence from op1-op2-op3-op4 to op1-op2-op4-op3 and checks its feasibility.

5 Experiment and Discussion

Three tasks in a block world are programmed by the method described above and are performed by a robot. The first task is to pick up a peg and insert it into a hole (Figure 4). In the task environment, the robot moves its end-effector to the peg's position, rotates the end-effector, picks up the peg, moves to the hole's position, directs the peg's top to the hole, moves down to the hole and releases the peg. Figure 10 shows the initial state of the task environment, an intermediate state, and the final state of assembly.



initial state intermediate state final state

Figure10: Robot's movement of inserting a peg into a hole

put RING BLUE on POLE GREEN
(a) Operator's actions in teaching environment

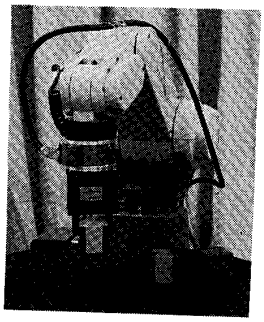
(b) Robot put operation

Figure 11: Placing rings on a pole

The second task is to pick up two different rings and place them on a pole one by one. The third task is to place a block on two other blocks. Figure 11 and 12 show an operator's actions in the teaching environment and the corresponding robot movements in the task environment.

The recognized command sequences of the experiments, such as *pick_up PEG GREEN*, *put RING BLUE on POLE GREEN* and *put BLOCK YELLOW on BLOCK RED and BLOCK GREEN*, are similar on the text level. However, the grasping point and the placing position of the object are calculated differently according to the method defined in each object class. Thus, the interpreted commands move the robot differently according to the task, using the task-dependent interpretation knowledge and the methods

pick_up BLOCK YELLOW
(a) Operator's action in teaching environment



(b) Robot movement pick_up operation

Figure 12: Placing an object on other objects

encoded in the object representation.

The above experiments involve robotic assembly tasks that need no replanning. A method of checking the feasibility of operation sequence has already been developed and simulated for simple cases [8]. However, it is not incorporated in the system at present. That method checks the sequence according to the surfaces of the objects. A three-dimensional range sensor for measuring object surfaces is currently under development.

6 Conclusion and Future Work

We proposed a teaching method in which robots learn to perform a task from an operator's movements in a teaching environment. The task-level assembly operations are recognized and interpreted into robot commands using task-dependent knowledge and object representation. Experiments show that a beginner can use the system to program a sequence of routines without knowledge of robot programming.

Future work involves teaching the parameters for motion routines, such as grasping points, fine motions such as *align these faces*, and force control parameters. This requires the system to adjust itself to what the operator wants to teach by his performance. The system will emphasize certain parts of the teaching environment or add relevant information according to the teaching context.

We believe that robot teaching using virtual reality will provide an efficient interface as robots become autonomous.

References

[1] H. Asada and Y. Asari, "The Direct Teaching Tool Manipulation Skills via the Impedance Identification of Human Motions," *Proc. of IEEE Int. Conf. on Robotics and Automations* , pp. 1269-1274, 1988.

[2] Y. Kuniyoshi, H. Inoue, M. Inaba, "Design and Implementation of a System that Generates Assembly Programs from Visual Recognition of Human Action Sequences," *Proc. of IEEE Int. Workshop on Intelligent Robots and Systems* , pp. 567-574, 1990.

[3] T. Hamada, K. Kamejima, I. Takeuchi, "Image Based Operation; A Human-Robot Interaction Architecture for Intelligent Manufacturing," *Proc. of IECON*, pp.556-561, 1989.

[4] S. Tachi, H. Arai, T. Maeda, "Tele-Existence Simulator with Artificial Reality," *Proc. of IEEE Int. Workshop on Intelligent Robots and Systems* , pp. 719-724, 1988.

[5] D. J. Sturman, D. Zelter, S. Pieper, "Hands-on Interaction With Virtual Environments," *Proc. of ACM SIGGRAPH Sym. on User Interface Software and Technology*, pp. 19-24, 1989.

[6] T. Takahashi, T. Sakai, "Teaching Robot's Movements in Virtual Reality," *Proc. of IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 1583-1588, 1991.

[7] T. Lozano-Peréz, J. L. Jones, E. Mazer, P. A. O'Donnell, "Task-Level Planning of Pick-and-Place Robot Motions," *COMPUTER*, pp. 21-29, March, 1989.

[8] H. Ogata, T. Takahashi, "An assembly state description based on part geometrical data," *Proc. of Annual Conf. of Robotic Society of Japan (in Japanese)*, pp. 937-938, 1991.