Towards an Assembly Plan from Observation: Task recognition with polyhedral objects'

Katsushi Ikeuchi and Takashi Suehiro
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Currently, most robot programming is done either by manual programming or by the "teach-by-showing" method using a teach pendant. Both of these methods have been found to have several drawbacks.

We propose a novel method to program a robot, the assembly-plan-from-observation (AFO) method. The AFO method aims to build a system that has the capability of observing a human performing an assembly task, understanding the task based on the observation, and generating the robot program to achieve the same task.

In particular, this paper defines assembly relations which serve as the basic representation of each assembly task. Then, we verify that such assembly relations can be recovered from the observation of human assembly tasks, and that from such assembly relations, it is possible to generate robot motion commands to repeat the same assembly task. Finally, we demonstrate an APO system based on the assembly relations.

1 Introduction

The key characteristic of robots is their versatility. They can be used to perform a large variety of **tasks** without a major re-design of the robot. This versatility is due to the generality of the robot's physical structure, but a robot's generality can be exploited only if the robot can be easily programmed.

Several methods to program a robot have been proposed. Such methods include: teach-by-showing, teleoperation [17, 12, 31, textual programming[2], and automatic programming [6, 9, 71. In teach-by-showing methods, an engineer stores, using a teach pendant in teaching mode, a path along which a robot should move repeatedly. In run mode, the robot follows the path it was previously taught. This is the most common method to program a robot in industrial applications. This method is suitable for programming a robot to repeat simple movements. Moreover, this method is excellent because a robot can learn complicated paths from a trained engineer. However, this method requires that an engineer is in the same environment **as** the robot. Thus, we cannot use this method in hazardous environments such **as** in nuclear plants, underwater, or in outer space.

To remedy this problem, teleoperation methods have been proposed. This method uses a master manipulator for teaching and a slave manipulator for execution. An engineer controls the master manipulator in a safe environment while monitoring the hazardous environment with a remote TV camera and display. The slave manipulator in the hazardous environment executes real operations based on control signals from its master manipulator. Since this method does not require an operator in the execution environment, it is suitable for the operation in hazardous environments. However, by using this method, we can only teach a robot trajectory information. It is difficult to build a flexible robot system able to use force control with error recovery capabilities. It is also true that we have to reconstruct entire programs, even when a very minor change in the program is desired.

Textual programming is often used in academic environments. A programmer stores a robot command sequence in a computer **as** a textual program. By using a compiler or an interpreter, a command sequence in a textual program is converted into a form that the robot can execute. This method is quite flexible because we can store any kind of control programs. However, it requires a long development period and expert programmers.

In order to speed up the programming process. automatic programming has been proposed. The method tries to develop geometric reasoning systems which can generate textual programs to control a robot from geometric information given by geometric models and **task** specifications. This direction is quite promising, however, there are many issues to be addressed before we have a complete automatic programming system. Such issues include: how to generate a sequence of operations, how to determine a grasp point for each operation, how to determine a global path to move an object while avoiding collisions with other objects. It is quite difficult to build a complete automatic programming system. though perhaps not impossible.

We propose a novel method that combines automatic programming and teleoperation. We propose to add a vision capability that will observe human operations to an automatic programming system (a geometric reasoner). In particular. we propose a system that observes a human performing an assembly **tasks** while a geometric reasoner analyzes and recognizes such tasks from observation. and generates the same assembly sequence for a robot. We will refer to this paradigm **as** Assembly Plan from Observation (APO).

Due to the geometric reasoning capability, the APO system understands the operations that the operator is performing. Thus, the system for example can discard unnecessary motions which are often introduced by a human teleoperator. The

'This research was sponsored by the Avionics Laboratory. Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. *Air* Force, Wright-Patterson AFB, Ohio **45433-6543** under Connact **F33615-90-C-1465** Order No. **7597.**

system can also insert error recovery routines into the generated assembly plans. In this regard, APO is superior to the teleoperation method.

Due to the vision capability, the system can solve several otherwise extremely difficult problems, such as path planning and determining the optimal-assembly sequence, by simply observing a human performing the operation. In this regard. APO is superior to the automatic programming method.

2 Assembly plan from observation

In an APO system, a human operator performs assembly tasks in front of a video camera. From the camera, the system obtains a continuous sequence of images recording the assembly tasks. In order for the system to recognize assembly tasks from the sequence of images, the system has to perform the following six operations (See Figure 1.):

- *Temporal Segmentation* - dividing the continuous sequence of images into meaningful segments which correspond to separate human assembly tasks,
- *Object Recognilion* - recognizing objects and determining object configurations in a given image segment.
- *Task Recognirion* - recognizing assembly tasks based on the results of an object recognition system.
- *Grasp Recognition* - recognizing where and how the human operator grasps an Object for achieving the assembly task
- *Global Path Recognirion* - recognizing the path along which the human operator moves an object while avoiding collision
- *Task Instantiation* - collecting necessary parameters from object recognition, grasp recognition. and global path recognition results for performing the recognized assembly tasks, and setting up assembly plans to perform the same task using a robot manipulator.

In this paper, we will concentrate on the task recognition and task instantiation modules, because these two parts form the main loop for the assembly plan from observation

The outline of the modules are as follows.

Our Object recognition module identifies each object using the object models from a given image segment The module represents the recognition results in a world model, as shown in Figure 1, by using the geometric modeler, Vantage

Our task recognition module recognizes Object relations in two image segments and extracts the transition between two object relations from the two segments The task recognition system has *abstract iask models* in a data base Each abstract task in the data base describes a transition between two different object relations. From the task models in the data base, the system identifies a task model that describe the transition needed to achieve the observed object relations. as shown in Figure 1

Our task instantiation module represenrs the recognition result as an instantiated task model An instantiated task model associates a transition with an action capable of causing the transition It also includes appropnate parameters to achieve the action based on the given scenes Such parameters include object locations and the grasping locations for the action The instantiated task model also includes the global path along

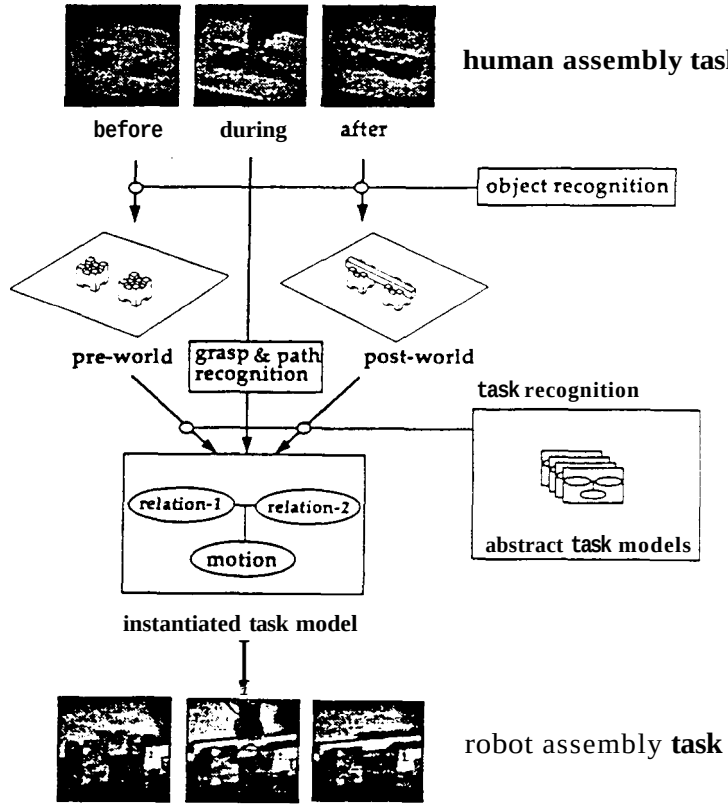

Figure 1: Assembly plan from observation

which to move an object. The system, then, inserts the obtained grasp and stack locations into the command sequence. Finally. the command sequence is sent to the robot.

3 Defining Task Models

In order to develop task models for an APO system, we have to define representations to describe assembly tasks In this section. we will define *assembly relations* for such representations. Then, we will examine that such assembly relations satisfy the two requirements.

- *recoverability* - assembly relations can be extracted from observation,
- *inferability* - a human assembly task can be inferred from an assembly relation. and it is possible to generate assembly operations for a manipulator from the assembly relation.

Finally, we will consider how to define assembly usk models using the assembly relations.

3.1 Assembly relation

In each assembly task, at least one object is manipulated. We will refer to the object as the *manipulated* object The manipulated object is attached to other stationary objects, which we refer to as *environmental* objects. so that the manipulated object achieves a particular relation with environmental objects.

We will define assembly relations with respect to lace contacts between a manipulated object and its stationary environmental objects. The essential goal of an assembly task is

to establish a new face contact between a manipulated object and environmental objects. For example, the goal of a peg-insertion is to achieve face contacts at the side and bottom faces of the peg against the side and bottom faces of the hole. Thus, it is effective to use face contact relations as the central representation for defining assembly task models.

To make the overall problem manageable, we concentrate on a world of polyhedral objects in which only one polyhedron may be moved by one assembly task. An assembly relation will be defined between a manipulated polyhedron and several stationary environmental polyhedra. This restriction still leaves a diverse range of interesting relationships, actions, and resulting assemblies.

Such face contact relations satisfy the recoverability requirement.

- *Face contact relations can be obtained by analyzing geometric models.* An object recognition program, such as in [4] can recognize a manipulated object, determine its configuration, and represent the recognition result in a geometric modeler, as well as the geometric representations of other stationary environmental objects. By examining each face pair between the manipulated and environmental objects, new face contact relations can be determined as they occur.

Face contact relations also satisfy the inferability requirement.

- *Each face contact relation constrains possible motions.* At contacting faces, the orientations of surface normals are suffice for characterizing relative object movement constraints. For example, consider a box resting on a table. At the contact faces surface normals are parallel and opposing. In this position the box can only move up or parallel to the table. A more constraining case is a square bar inserted in a matching shaped hole. The bar's four faces contact their hole counterparts with opposing normals and the only possible motion lies along the hole's axis. Thus, from a face contact relation, it is possible to infer the assembly actions that cause such face contact relations.
- *Face contact relations characterize a control strategy necessary to maintain such relations.* Each face contact relation provides a constraint to motion. As long as the motion constraint is constant, the same mode of control is applicable. When the motion constraint changes, a different mode of control is required. For example, let us consider a box to be placed on a table and then slide on the table. Position control can be used to lower the box towards the table while the box is in the air (the box does not have any face contact.). When the box is about to make contact with the table (about to have one-face contact), force control is necessary to detect the collision which ensures that the box is on the table. Combined force and position control is necessary to slide the box on the table (for maintaining one-face contact). Face contact relations have been found to characterize required control strategies [14, 13]. Thus, such face contact relations can be used to determine a control strategy necessary to achieve such face contact relations in assembly actions.

Using such face contact relations as the basic representations, we will describe an assembly task with a transition between pre-assembly relations and post-assembly relations. Based on the description, we will build an APO system in the following steps:

- classifying all possible face contact relations (assembly relations) between manipulated and environmental objects,
- considering what kinds of transitions in assembly relations occur and building a tree in which each branch corresponds to one possible transition and tach leaf node corresponds to an assembly relation, and
- assigning manipulator motions to achieve such assembly relation uansitions (the completed tree is referred to as a procedure tree).

3.2 Taxonomy for Assembly Relation

For geometric objects in a polyhedral world. our taxonomy identifies all possible assembly relations based on the directions of contact surface normals. First, we will analyze a two-dimensional polygonal world and then a three-dimensional polyhedral world. Some related issues are found in [10].

3.2.1 Two-dimensional cases.

Assembly relations will be considered between polygons (not polyhedra) by using normal directions at contact edges. Figure 3.2.1 shows an assembly relation having unidirectional contact. Even if polygons have several contact edges with the same normal direction, they are considered as having unidirectional contact.

Normal direction of an edge can be represented as a point on the Gaussian circle by translating the unit normal so that its starting point sits at the origin of the coordinate system. Its tail then lies on the unit circle whose center is the origin. This mapping is referred to as a Gauss mapping. Equivalently, possible movement directions of the object polygon can be represented on the Gaussian circle. Points on the semicircle around the contact direction correspond to the possible movement directions of the object polygon. Points on the other semicircle correspond to the prohibited motion directions.

Bidirectional contact has two possible assembly relations as shown in Figure 3. Assembly relation 2d-b in Figure 3 has two maps located opposite one another on the circle. This case has two possible movement directions on the Gaussian circle. Relation 2d-c in Figure 3 has two oblique contact directions on the Gaussian circle. This case has several possible movement directions corresponding to a small arc.

Tridirectional contact at first seems to have three cases: relation 2d-d, relation 2d-e, and relation 2d-f, as shown in Figure 4. Relation 2d-d has two opposite points and one intermediate point; resulting in only one possible movement direction. Relation 2d-e has three points whose maximum arc is larger than π, allowing no movement of the object at all.

Relation 2d-f has three arbitrary points whose maximum arc is less that π. As Figure 4(c) illustrates, the middle contact direction does not affect the possible movement directions on the Gaussian circle. Thus, relation 2d-f is considered equivalent to relation 2d-c and is not considered an independent relation. When a relation has more than three directions of contact, it can be mapped to one of the relations mentioned above.

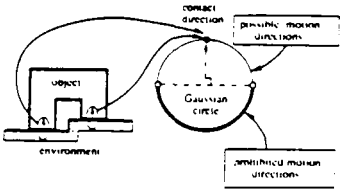

Figure 2: Unidirectional contact.

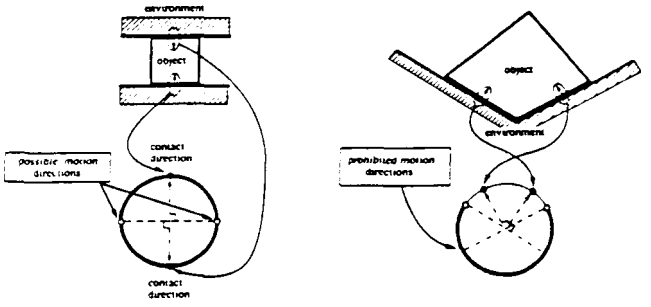

Figure 3: Bidirectional contact: (a) relation 2d-b; (b) relation 2d-c.

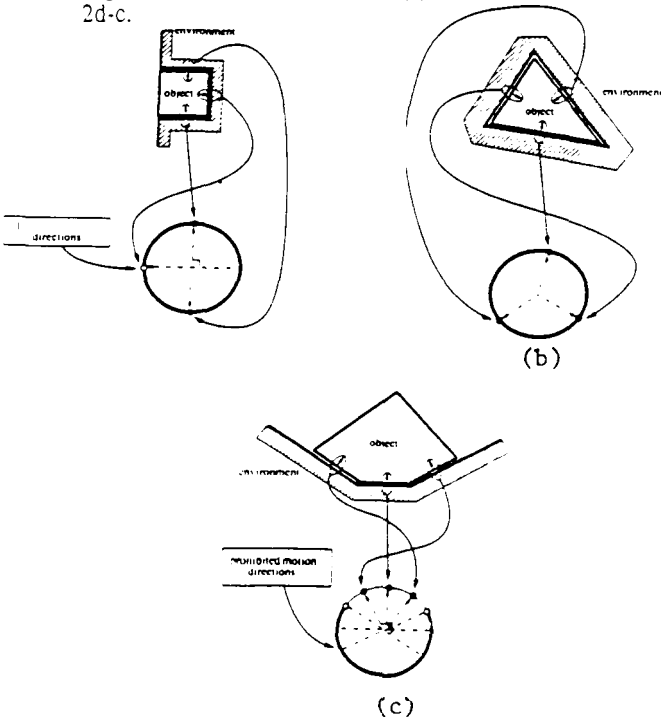

Figure 4 Tridirectional contact: (a) relation 2d-d; (b) relaуon 2d-e; (c) relation 2d-f. This is equivalent to relation 2d-c in terms of possible movement directions, and thus, is not considered an independent relation.

Table 1: 3D assembly relations

Class	Relation	Explanation
uni	3d-s	3d-a in Figure 5 shows unidirectional contact. The contact direction is represented as a potnt on the Gaussian sphere. Let us suppose the contact direction is mapped to the north pole of the Gaussian sphere. The possible directions of object monon can be represented as the northern hemisphere of the Gaussian sphere. the prohibited directions can be represented as the southern hemisphere.
bi	3d-b 3d-c	Bidirectional contacts have two different relations: 3d-b and 3d-c, dependingon whether contact directions are in opposite directions or not. See Figure 5 Relaуon 3d-b has possible movemenrc represented as a great circle an the Gaussian sphere. Relaoon 3d-c. on the other hand. has possible movements represented as an area bounded by two great circles on the Gaussian sphere.
tri	3d-d 3d-e 3d-f	The tridirectional contacts have three different relations: 3 d 4 3d-e, and 3d-f. Relaуon 3 d d and relaуon 3d-e have three coplanar contact directions Relaуon 3d-d has possible movements corresponding to a great semi circle Relaуon 3d-e has possible movements corresponding to two points Between relaoon 3d-e and relaуon 3d-d, there exists a relation which has identical possible movement directions as relation 3d-c Relaуon 3d.f has possible movements corresponding to a spherical area bounded by three great circles
tetra	3d-g 3d-h	Relaуon 3d-g has possible movements corresponding to an arc, while relaуon 3d-h has one possible movement. Addingonemore contact direction to relanon 3d-e gives relaуon 3d-h.
hexa	3d-i	3d-i in Figure 5 has no possible movement directions.

3.3 Assembly relation transitions

We will consider a sequence of manipulator operations to achieve each assembly relation from assembly relation 3d-s. Such a sequence of manipulator operations is grouped into a motion macro, i.e., a template of manipulator operations, which, when applied to an Object. yields the desired assembly relation. This is possible because each assembly relation is defined so that we can apply the same manipulator control strategy to achieve the relation by changing only controller parameters. not the strategy.

In order to reduce the number of necessary templates, we will analyze each assembly relation in an iterative manner. We will analyze simpler relations earlier and more complicated relations later. Also. instead of considenng a template to directly achieve a complicated relation from 3d-s. we will

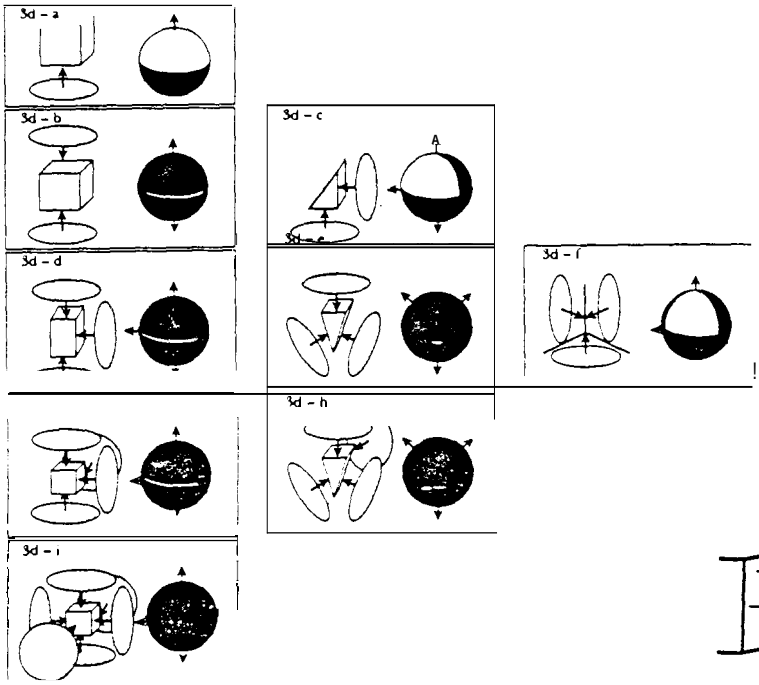

Figure 5: 3-D assembly relation taxonomy.

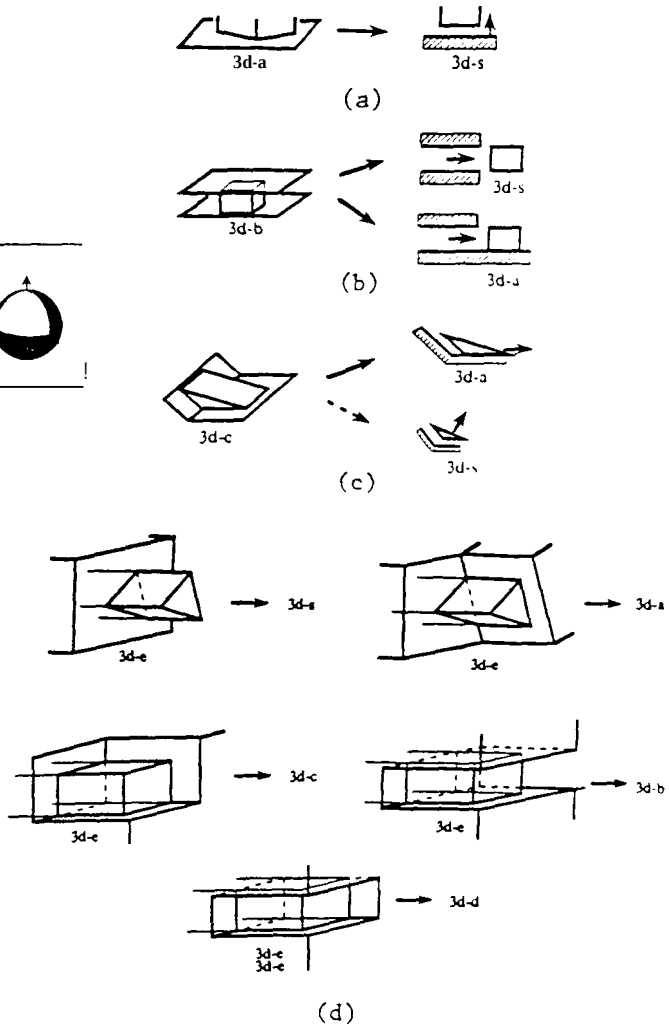

Figure 6: Examples of assembly relation transitions

consider an intermediate relation, and then try to achieve the complicated relation. First, we try to achieve an intermediate relation from 3d-s by using the templates already considered Then we try to achieve the final relation from the intermediate relation using a newly considered template.

In order to find an appropriate intermediate relation, for each assembly relation, we consider disassembly actions from the assembly relation, and extract all possible immediate intermediate assembly relations just pnor to the assembly relation We do this because considenng disassembly actions is easier than considenng assembly actions.

Several intermediate relations sometimes occur from the same assembly relation due to 1) the variation in shapes of contact faces. and *2)* the variety of possible disassembly operations

In case that due to variations in the shapes of contact faces, we have to analyze all intermediate relations and assign appropriate motion templates to all uansitions from the intermediate relations to the desired relation

In case that due to the variety of possible disassembly operations. we can choose one appropriate intermediate relation among the several intermediate relations We choose the one which is achieved by the simplest and most robust operation under uncertainty in positional information. In order to select such intermediate relation, we use the following criteria.

1. In the case that a *direct detach* motion (a motion which immediately breaks a face-contact) exists, choose it.
2. In the case that a *lateral* motion (a motion maintain the same contact relation) that would break face-contacts by crossing a certain boundary exist. choose it
3. In the case in which several candidate motions satisfy criterion 1 or criterion 2, choose the motion which least reduces the number of face contacts.

By using these criteria, we will analyze each assembly relation. extract all possible assembly relation uansitions, and prune unnecessary relation uansitions.

We can represent relation transitions as a tree structure, as shown in Figure 7. Each node in the *tree* represents one particular assembly relation, and each arc represents corresponding assembly relation uansitions.

3.4 Procedure free

A procedure tree (Figure 8) is created by placing a template of manipulator operations (motion macro) at each arc separating the assembly relation nodes of Figure 7. See Tab:; 9. The manipulator operations chosen are those which can correctly achieves an assembly relation on one node from the assembly relation on the other node.

From this analysis in Table 3, the following four motion macros are extracted:

Table 2: Possible assembly relation transitions

Relation	Possible immediate simpler relanon	Transition
3d-a	A direct detach monon giver the transinon from the assembly relation 3d-s to 3d-s. See Figure 6(a) for an example of the *direct* detach motion which causes an assembly relation transition from 3d-a to 3d-s.	s-to-a
3d-b	No direct detach motion can be applied to the assembly relanon 3d-b. Lateral motions parallel to the contact faces can be applied. Depending on the shape of contact faces, it reaches either 3d-s or 3d-a. Since this variation is due to the shape of the contact face, we have to consider both cases. Figure 6(b) shows two possible relation transitions.	s-to-b a-to-b
3d-c	By applying direct detach motions, the 3 d c relation becomes either 3d-s or 3d-a. The two possibilities are not due to the shape of the contact facer; they are due to monondirections The relation transinon bum 3 d c to 3d-s reduces the number of face-contacts by two. while the relation transition from 3 d c to 3d-a reduces the number by one. The latter relanon transinon is chosen as the desirable one by the criterion 3. Figure 6(c) shows two possible transitions due to monon directions.	a-to-c
3d-d	A direct detach monon the transition from the assembly tion 3d d to 3d b.	b-to-d
3d	*No* direct detach motions exist in case of 3d- Lateral motions along the axis parallel to the surrounding contact faces cause several relations, 3d-s, 3d-a, 3d-b, 3d-c and 3 d 4 depending on the shape of contact facer. We have to consider there five possible relanon transinom. We will refer to the axis as the *insertion axis*. See Figure 6(d) for the example.	s-to-e a-to-e b-to-e c-to-e d-to-e
3d-f	The assembly relanon 3d-f becomes to 3d-s, 3d-a and 3 d c by a detach rmonon depending on monon directions. The relauon transitions, 3d-f to 3d-s, 3d-f to 3d-a. and 3d-f to 3d-c, reduces the number of face-contacts by three, two. and one, respectively. Thus. following criterion 3, the relanon transinon. 3d-f to 3 d c is chosen as the desirable one.	c-to-f
3d.g	Among the two possible relauon transinom. 3d-g to 3d-b and 3d-g to 3d-d, The relation transition 3d-g to 3 d d is chosen by criterion 3	d-to-g
3d-b	The assembly relauon 3d-h *can* become *w* 3d-e by monon dong the insertion axis.	e-to-h

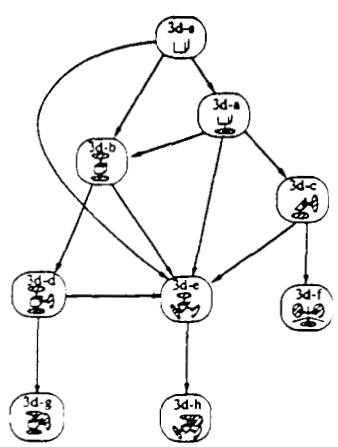

Figure 7: Assembly relation transitions represented as a tree

Table 3: Morion macro

s-to-a	The relation transition from 3d-s to 3d-a is realized by an attach monon which contains a monon component toward the contact direction. Among several attach monons, pure monon towards the *contact* direction until face contact is the easiest. Thus. we assign the corresponding template of monons to the relanon transinon from 3d-s to 3d-a and refer to this template as *move-to-contact* motion macro.	move-to-contact
[illegible]	that we can translate it between the two contact faces. Then, we have to translate the object parallel to the contact facer. We assign the corresponding template of [illegible] to the rclanon transinon from 3d-s to 3d-b and refer to this temp[illegible] as *insert-between* mou macro.	between
a.to-b	Since the configuration of the object is already aligned, *it* is only necessary to translate the object parallel to the two contact faces. We assign the corresponding template of motions to the relauon transinon and refer to this template as *move* rnonon macro	ove
a-to-c	At 3d-a. one face contact is already achieved The relation transition from 3d-a to 3d-c is realized by an attach motion along the contact face of 3d-a toward another contact face of 3 d c until two-face contact occurs Among several such monons. pure monon perpendicular to the intersection lines between two contact faces is selected. We achieve this relation transition by usmg the same template of operations for the relanon transinon from 3d-s to 3d-a, *move-to-contact*. Thus. we assign the *move-to-contact* monon macro to the relation transinon from 3d-a to 3d-c.	move-to-contact
b.to-d	The relanon transition from 3d-b to 3 d d is realized by an attach monon parallel to the two opposite contact faces. Among several attach monons. as is the case in 3d-a to 3d-c, pure monon is selected. For this rclanon transition we us[illegible] macro *n to-contact*.	move-ID-contact
s-to-e	First we have to align *the* configur[illegible] of the obje[illegible] so that it can be translated dong the insert axis Then. we have to translate the object along the axis We assign the corresponding template of motions to the relanon transition from 3d-s to 3d-e and refer to this template as *insert-into* macro. Note that the *insert-between* macro only aligns the object parallel to a pair of contact faces The macro allows the rotation and translation freedom dong the contact faces On the other hand. the *insert-into* macro does not allow such freedom; it only allows the object to translate along	insert into
to-e	contact direction in the previous 3d-a relauon Thus. by using the *insert-between* motion macro. we align the object dong the men axis and translate it into the hole	between
b-to-e	ditto	insert-between
c-to-e	The monon of the object is already constrained; the object is only movable along the insert uis We Use the *move* macro to make the relation transitions	move
d-to-e	ditto	move
c-to-f	The relation transition bum 3d-c to 3d-f is realized by an attach mouon along the intersection line of the nvo contact faces of 3 d s until tndirectional contact occurs For this relauon transinon we use the same macro *move-to-contact*	move-to-contact
.d-lo-	At 3 d 4 the object can only translate along intersection lines among contact faces. By using the *move-to-contact*	move. lo-
to-h	the *move-to-contact* macro dong the insert axis unnl the tetradirectional contact occurs.	to-contact

- *move* - a motion sequence for this macro is realized by translating a manipulated object from the starting configuration to the ending configuration.
- *move-to-contact* - a motion sequence for this motion macro is realized by translating a manipulated object until it contacts a face of an environmental object, then fitting a manipulated object face to the contact environmental face.

  If we have precise configurations. we can achieve the contact and fitting operations by using such configurations. Otherwise, these operations require some sensory feedback to detect the Occurrence of contact and fitting. See [14] for a detailed implementation of the macro as a skill in a force feedback type manipulator.
- *insert-between* - a motion sequence for this motion macro is realized by first aligning a manipulated object between a pair of contact environmental faces, and then translating it between the pair of contact faces to the ending configuration.

  If we have precise configurations, we *can* achieve align motion and translation motion using the configurations. Otherwise. the align motion requires some sensory feedback. See [14].
- *insert-into* - a motion sequence for this motion macro is realized by aligning a manipulated object along the insert axis, and then translating along the axis to the ending configuration.

  If we have precise configurations, we can achieve align motion and translation motion using the configurations. Otherwise, the align motion requires some sensory feedback. See [14].

Figure 8 represents a completed procedure tree.

3.5 Task models

A task model consists of an assembly relation transition, a motion macro. and the necessary parameters required to expand the motion macro into a sequence of manipulator commands. For example, Figure 9 shows the task model corresponding to the transition from 3d-s to 3d-a. The starting and end relation slots contain 3d-s and 3d-a, respectively. The action slot contains the move-to-contact motion macro. In order to achieve the mouon, it is necessary to know the previous configuration and end configuration of the manipulated Object. The corresponding parameters are prepared as task parameters. The values corresponding to these parameters are obtained by the task instantiation module at run time.

Thirteen task models corresponding to all arcs in the tree are prepared. They are attached to the procedure **tree.**

4 Implementation **of APO** system

How are task models used to recover human assembly tasks in the APO system? The task recognition mechanism will be explained in the following examples. The example system Consists of three classes of objects, (any of which can appear in the scene): castle, block. and stick Figure 10).

4.1 Temporal Segmentation

The system assumes that at the beginning of each assembly **task** human intervention occurs in the scene and at end of

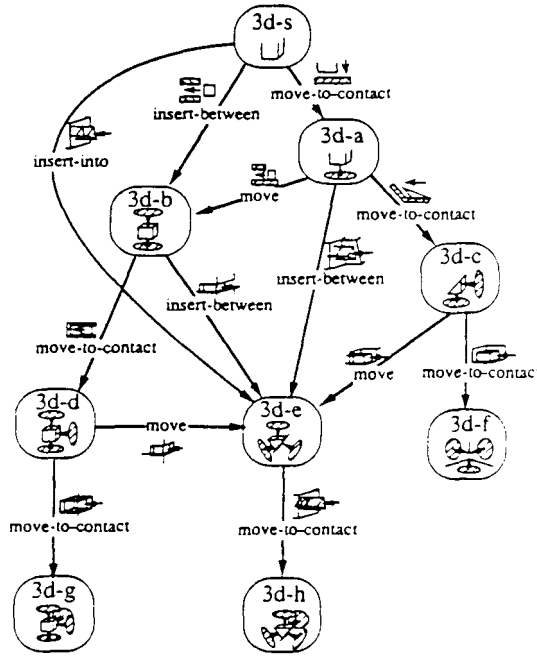

Figure 8: Procedure tree.

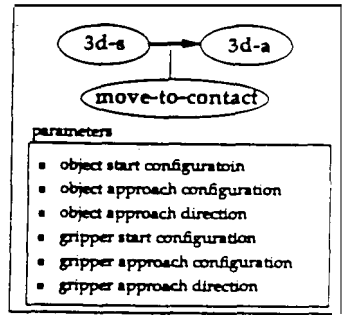

Figure 9: The **task** model corresponding to the assembly relation transition from 3d-s to 3d-a

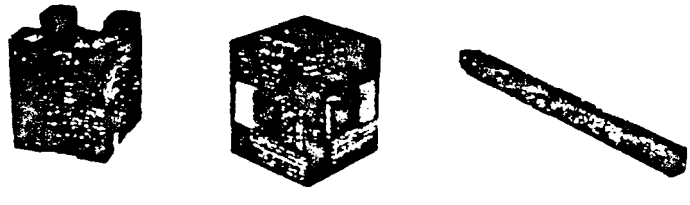

Figure 10: Castle, block and stick

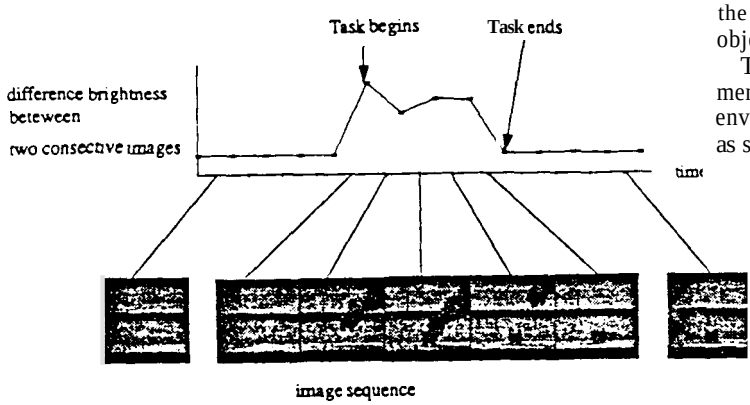

Figure 11: The system detects human intervention from the change in brightness values

the assembly task the human disappears from the scene By using this assumption, the XPO system segments acontinuous image sequence given by a TV camera from the scene into a finite number of meaningful chunks

By using the level change in the brightness difference, the system can detect human intervention Figure 11 shous a continuous image sequence of a scene given by a TV camera, while the human operator is putting a castle on the table Before human intervention. the scene consists of only still objects, thus the difference between two consecutive images is at the quite level When human intervention occurs, the brightness difference is large due to the motion of human and manipulated object in the scene This disturbance continues until the end of theassembly operation After the human hand disappears. the scene consists of only still objects Thus, the brightness difference returns to the quite level

We have been using this method for detection for several live demos repeated continuously for several days, and the method never failed.

4.2 Object Recognition

Objects in the scene are recognized from range data in our current implementation. b/w images are used only for detecting the completion of one assembly **task** More reliable range data are used for analyzing the scene After a certain period after the detection of the completion of one assembly task. the APO system invokes the range finder and measures range intormation in the scene. The APO system then generates a difference image between the range image from the previous step (before the assembly **task)** and the range image from the current step (after the assembly task)

The system applies a segmentation program to the difference image and obtains any newly appearing regions These new regions correspond to the faces of the manipulated object by the assembly **task** See Figure 12

An object recognition program is applied *to* the new regions. After recognizing a manipulated Object from *the* regions by fitting a geometric model *to* range data on the regions. the program determines the configuration of the manipulated object.

The system maintains the configurations *of* other environmental objects. The system represents these manipulated and environmental objecrs in the Vantage geometric modeler [11 as shown in Figure 13

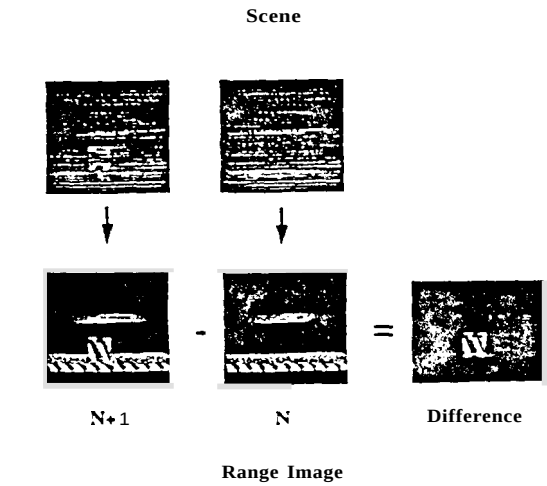

Figure 12: The difference in range data

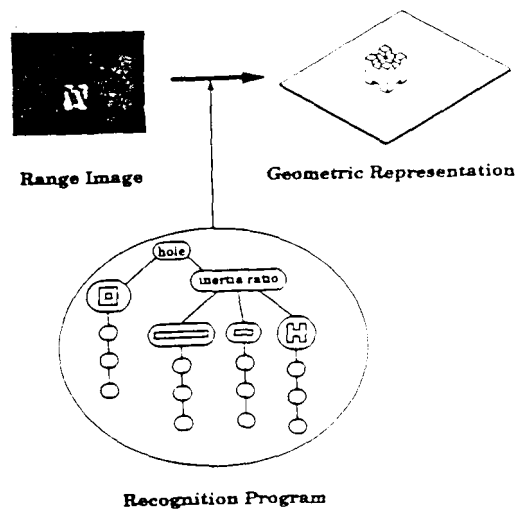

Figure 13 Object recognition a recognition program is applied only to any newly appearing regions, and recognizes only manipulated object The recognition results are represented by Vantage

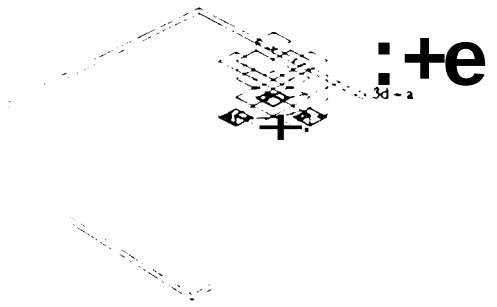

Figure 14: Extracted contact faces and assembly relation.

4.3 **Task** Identification

By using the transformation from body coordinate systems to face coordinate systems, (available from the Vantage geometric modeler), the configurations of the faces of the manipulated and environmental objects are obtained.

The system extracts contacting face pairs from the face configurations. Here, a contacting face pair is a face from the manipulated objects and a face from an environmental object, which have the same face equations and whose surface normals are opposite to each other.

The system determines the assembly relation based on the contacting face pairs by analyzing the contact directions of pairs. Here, the contact direction is defined as the normal direction from the environment faces to the manipulated object faces as previously defined. The contact pairs are grouped into a set of contact directional groups so that each group has face pairs with the same contact direction. By examining the occurrence of directions, we can determine which assembly relation occurs by the assembly task.

The system recognizes the contact faces and contact directions as shown in Figure 14. From the contact faces in Figure 14, the system determines that the current assembly relation is 3d-a.

Before the assembly **task,** the castle does not exist in the scene. Thus, before the assembly task, the assembly relation between the castle and the table was 3d-s. After performance of the assembly task, the manipulated castle established a 3d-a assembly relation with the environmental object, the table.

From this observation. the system recognizes that the assembly relation transition, 3d-s to 3d-a, occurs due to the assembly task. The corresponding task mode 3d-s to 3d-a is extracted from the corresponding arc along the procedure tree.

4.4 **Task** Instantiation

In this example, at the previous step, the castle was stored on the warehouse table. Thus, the assembly relation transitions during the entire assembly task are

- 3d-a to 3d-s: detach the castle from the warehouse table to the departure configuration.
- 3d-s to 3d-s: bring the castle from the departure configuration to the approach configuration in free space.
- 3d-s to 3d-a: move-to-contact the castle to the working table from the approach configuration.

Thus, the corresponding three **task** models are instantiated: *a-io-s, s-io-s,* and *s-io-a.*

The following procedure is executed to instantiate a task model:

- obtain an abstract task model from the data base.
- obtain necessary parameters for the motion-macro (i.e. motion direction and translation distance) derived from the object recognition results.
- obtain the necessary motion macro (a sequence of manipulator motions) by consulting the action slot of the **task** model,

The instantiationof task models occurs in the reverse order. *s-to-a, s-to-s,* and *a-to-s.*

The *s-to-a* task model has a ***move-io-contact*** motion macro in the action slot. The **task** model examines each object model and determines grasp configurations. how to grasp the object with respect to the body coordinate system. and the specified grasping method. In the current implementation. each object model has predetermined grasping configurations. The task model chooses an appropriate grasping configuration and recalculates it based on the current body configurations. The task model determines the grasping configuration of the castle based on the observed castle configuration. The task model also determines the stack configuration of the castle on the table in a similar manner. The system then inserts these parameters to the corresponding slots in the instantiated **task** model.

The global motion is also implemented as a **task** model. *s-to-s.* This task model has a motion macro, ***move.*** The current implementation does not consider collision between the manipulated object and environmental objects. It assumes that space above a certain level of height is free space. The task model incorporates the path from the departure configuration to the high position, the high position to another high position above the approach configuration, and the second high position to the approach configuration. These configurations arc obtained from the old and new configurations of the manipulated objects. These values are inserted into their slots in the instantiated task model.

The disassembly task is also implemented as a task model. The current implementation does not observe the warehouse table due to the field of view of the range finder. Thus, the assembly relation transition, 3d-a to 3d-s, which occurs at the warehouse table, is given to the system as a priori knowledge. The system instantiates a disassembly **task** model. *a-to-s.* This task model has **a** motion macro, *move* in the action slot. The grasp configuration for the disassembly task is obtained from the geometric model in a similar manner to the assembly task model. This value is stored in the corresponding slot in the instantiated task model.

The system finally performs the operations given by the three **task** models sequentially: *a-to-s, s-to-s,* and *s-to-a.* Figure 15 shows the final move-to-contact operation by a manipulator.

4.5 Additional example-

Figure 16(a) shows a human operation for inserting a stick in **a** hole of the block. The system recognizes the contact faces (Figure 16(b)). From the normal direction of contact faces, the system generates tetra directional contact. By examining

the directions of the contacts, the system determines that the observed assembly relation is 3d-e.

Currently, the vision system cannot detect intermediate relation transitions such as from 3d-b to 3d-e due to our temporal segmentation method. It can only detect the relation transition from 3d-s to 3d-e. Thus, the system explores all the possible paths in the procedure tree between 3d-s and 3d-e. Then, by examining the shape of contact pairs, the system infers which path occurs.

More precisely, the relation transition from 3d-s to 3d-e corresponds to five paths; direct path, via 3d-b, via 3d-a and 3d-b, via 3d-a and via 3d-a and 3d-c. All the arcs to the 3d-e, however, have the same assembly action (and disassembly action), translation along the axis. In Vantage, the disassembly action is applied to the current geometric representation of the manipulated and the environment objects to find the previous assembly relation. The system examines the vertex coordinates of all the contact faces, projects them to a plane parallel to the translation directions, and determines which assembly relation occurs due to this translation action. In this example, the system finds that all the boundary edge vertices on the contact faces have the same coordinate system along the translation directions. From this, it concludes that the 3d-s to 3d-e relation transition occurs.

The *s-to-e* task model has a motion macro, insert-into in the action slot. Using the predetermined grasp configuration and the observed stick position. the system performs the insert operation as shown in Figure 16(c).

Figure 17 shows other examples constructed successfully by the system.

5 Conclusion

We have described an Assembly-Plan-from-Observation (APO) system that can observe an assembly task performed by a human, recognize scene objects, relations among those objects, and actions on them, and produce corresponding operational plans for a robot. Our work will open a new domain of object recognition applications and provide a revolutionary way of programming robots.

The current system analyzes human operation and generates the fine motion plan from observation among polyhedral objects. Future directions include how to generate grasp plans and global motion plans from observation.

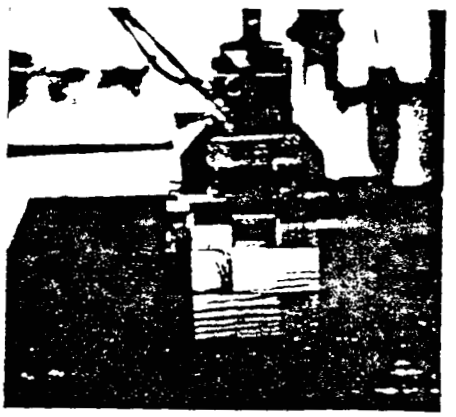

Figure 15: Put a block on the table with a manipulator.

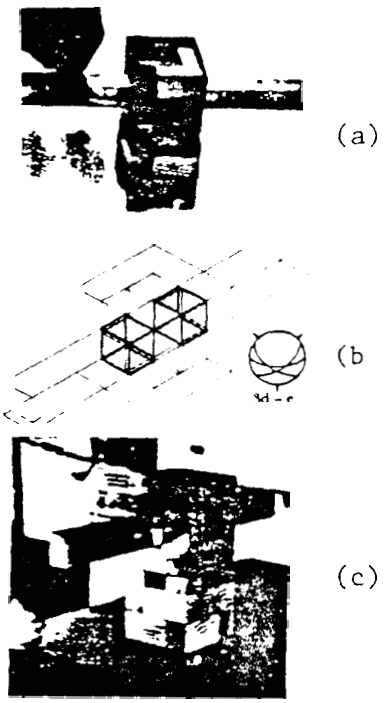

Figure 16: Insert a stick to the block; (a) input scene, (b) face contact, (c) system performance.

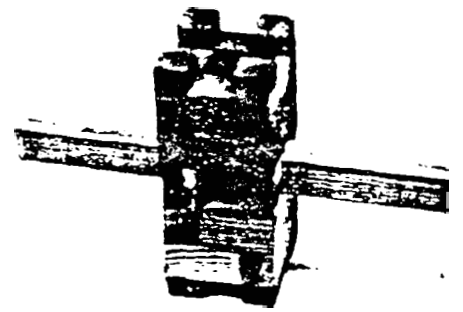

Figure 17: Additional examples.

Acknowledgement

Raj Reddy and Takeo Kanade provided useful comments and encouragements. Bradley Nelson proofread the draft of this manuscript and provided useful comments.

Takashi Suehiro was on leave of absence from Electrotechnical Laboratory, Tsukuba, Japan.

References

[1] P. Balakumar, J.C. Robert, R. Hoffman, K. Ikeuchi, and T. Kanade. Vantage: A frame-based geometric/sensor modeling system - programmer/user's manual v 1.0. Technical report, Carnegie Mellon University, Robotics Institute, 1989.

[2] R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman. AI, a programming system for automation. Technical Report AIM-177, Stanford University, Artificial Intelligence Laboratory, Stanford, CA, 1974.

[3] S. Hirai and Sato. T. Motion understanding for world model management of telerobot. In *IEEE Intern. Conf. on Intelligent Robots and Systems*, pages 124–131, 1989.

[4] K. Ikeuchi and K. S. Hong. Determining linear shape change: Toward automatic generation of object recognition programs. *Computer Vision, Graphics. and Image Processing: Image Understanding*, 53(2), March 1991. a longer version, containing programs, is avaiable as CMU-CS-88-188.

[5] Y. Kuniyoshi, H. Inoue. and M. Inaba. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. In *Proc. of IEEE Intern. Workshop on Intelligent Robots and Systems*, pages 567–574, August 1990.

[6] L.I. Lieberman and M.A. Wesley. Autopass: an automatic programming system for computer controlled mechanical assembly. *IBM Journal of Res. Develop.*, 21(4):321–333, 1977.

[7] T. Lozano-Perez. Automatic planning of manipulator transfer movements. *IEEE Trans. System Man and Cybernetics*, SMC-11(10):681–689, 1981.

[8] T. Lozano-Perez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. In M. Brady and R. Paul, editors, *Robotics Research 1*, pages 65–96. MIT Press, Cambridge, MA, 1984.

[9] T. Lozano-Perez and P.H. Winston. Lama: a language for automatic mechanical assembly. In *Proc. of 5th Intern. Joint Conf. on Artificial Intelligence*, pages 710–716, 1977.

[10] L.S.H. Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. In *Proc. of IEEE Intern. Conf. on Robotics and Automation*, pages 56–61, 1989.

[11] R.J. Popplestone, A.P. Ambler, and I. Bellos. An interpreter for a language for describing assemblies. *Artificial Intelligence*, 14(1):79–107, 1980.

[12] N.O. Sliwa and R.W. Will. A flexible telerobotic system for space operations. In *Proc. Space Telerobotics Workshop*, pages 285–292, Pasadena, 1987.

[13] T. Suehiro and K. Takase. Representation and control of motion in contact and its application to assembly tasks. In *Proc. of Intern. Symposuim of Robotics Reseach*, pages 367–374, Tokyo, Japan, August 1989.

[14] T. Suehiro and K. Takase. Skill based manipulation system. *Journal of the Robotics Society of Jupan*. 8(5):47–58, October 1990. (in Japanese).

[15] R. Thibadeau. Artificial perception of actions. *Cognitive Science*, 10(2):117–149, 1986.

[16] S. Tsuji, A. Morizono, and S. Kuroda. Understanding a simple cartoon film by a computer vision system. In *Proc. of 5th Intern. Joint Conf. on Artificial Intelligence*, pages 609410, 1977.

[17] D.E. Whitney. State space models of remote manipulation tasks. In *Proc. of 1st Intern Conf Artificial intelligence*, pages 495–507, 1969.

[18] P.H. Winston. The mit robot. In Machine Intelligence 7, editor, *Meltzer, B. and Michie, D.M.* Edinburgh Univ. Press, Edinburgh, U.K., 1972.

[19] P.H. Winston. Learning structural descriptions from examples. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.