Toward an Assembly Plan from Observation Part I: Task Recognition With Polyhedral Objects

Katsushi Ikeuchi, *Member, IEEE,* and Takashi Suehiro

Abstract—**Currently, most robot programming is done either by manual programming or using a teach pendant as part of the "teach-by-showing" method. Both of these methods have been found to have several drawbacks. We are developing a novel method with which to program a robot: the assembly-plan-from-observation (APO) method. The APO 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 subsequently generating a robot program to achieve the same task.**

This paper focuses on the task recognition module (TRM), the main component of a complete APO system. The TRM recognizes object configurations before and after an assembly task, detects a configuration transition, and infers the assembly task that causes such a configuration transition. We assume that each assembly task aims to achieve a face contact relation between an object that has just been manipulated and stationary environmental objects. We prepare abstract task models that associate transitions of face contact relations with assembly tasks that achieve such transitions. Next, we implement TRM in order to verify two issues: 1) that such a contact transition can be recovered from the output of the object recognizer, and 2) that given these relation transitions, it is possible to use the abstract task models to effect the generation of robot motion commands; the execution of these commands will culminate in a repetition on the original assembly task.

I. Introduction

SEVERAL methods to program a robot have been proposed. Such methods include: teach-by-showing, teleoperation [33], [25], [8], textual programming [6], and automatic programming [15], [19], [16]. In teach-by-showing methods, an engineer uses a teach pendant in teaching mode to store 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 an engineer to be 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 repeat the entire motions, even when a very minor change in the operation 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 that can generate textual programs to control a robot from geometric information given by geometric models and task specifications. Although this direction is quite promising, 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 exceedingly difficult to build a complete practical automatic programming system.

We have been developing a new programming paradigm that we call an assembly-plan-from-observation (APO). An APO system would be capable of observing a human performing an assembly task, recognizing such tasks from observation, and generating the same assembly sequence for a robot.

The APO paradigm is related to several bodies of research. Kuniyoshi, Inoue and Inaba developed a system that tracks movements of a human hand for program generation [14]. Since this system tracks operations using only vision and does not employ the knowledge given by analyzing geometric

Manuscript received March 1, 1993; revised March 22, 1994. This research was supported in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597, and by the National Science Foundation under Contract CDC-9121797. A preliminary version of this paper was presented at the 1992 IEEE International Conference on Robotics and Automation, Nice France.

Katsushi Ikeuchi is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

Takashi Suehiro is with Electrotechnical Laboratory, Tsukuba, Japan.

IEEE Log Number 9402039.

models of objects, it is restricted to perform simple pick-and-place operations. Asada and Hirai proposed to monitor assembly operations based on face contact patterns [2]. They derive a taxonomy of contact patterns based on geometric models. Force and displacement information acquired in the process is interpreted by using the taxonomy in order to estimate the state of the assembly process. Hirai and Sato described a system that is capable of recognizing manipulator motions in order to maintain the consistency between an internal world model and the real world [8]. Their main emphasis is on the monitoring of the assembly process.

This paper overviews the paradigm of APO in Section II. Then, Section III and Section IV report the design and implementation of the task recognition module that forms a core component of the APO system. Section V summarizes the work, describes the remaining parts of APO, and discusses the relation of the APO paradigm with the traditional paradigms such as automatic programming and teleoperation.

II. Assembly Plan from Observation (APO) Paradigm

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:

1) *Temporal Segmentation:* Dividing the continuous sequence of images into meaningful segments that correspond to separate human assembly tasks,
2) *Object Recognition:* Recognizing objects and determining object configurations in a given image segment,
3) *Task Recognition:* Recognizing assembly tasks based on the results of an object recognition system,
4) *Grasp Recognition:* Recognizing where and how the human operator grasps an object for achieving the assembly task,
5) *Global Path Recognition:* Recognizing the path along which the human operator moves an object while avoiding collision, and
6) *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 task recognition, because it constitutes the main component for the assembly plan from observation.

The outline of the object and task recognition is as follows:

Object recognition, including our object recognition module, extracts object features such as edges or faces from a given images, compares them with those of abstract object models in the database, and identifies each object. The recognition results are represented in a world model with instantiated object models, for example, by using the geometric modeler Vantage [3] in our system as depicted in Fig. 1.

Task recognition fulfils a similar function at the higher level. Task recognition extracts object relations from instantiated world models (such as pre- and post-world models in Fig. 1). By comparing two sets of object relations, it extracts the transition between them. Task recognition uses *abstract task models* just as object recognition does abstract object models. From abstract task models in the data base, the system identifies and instantiates a task model that describes the current transition.

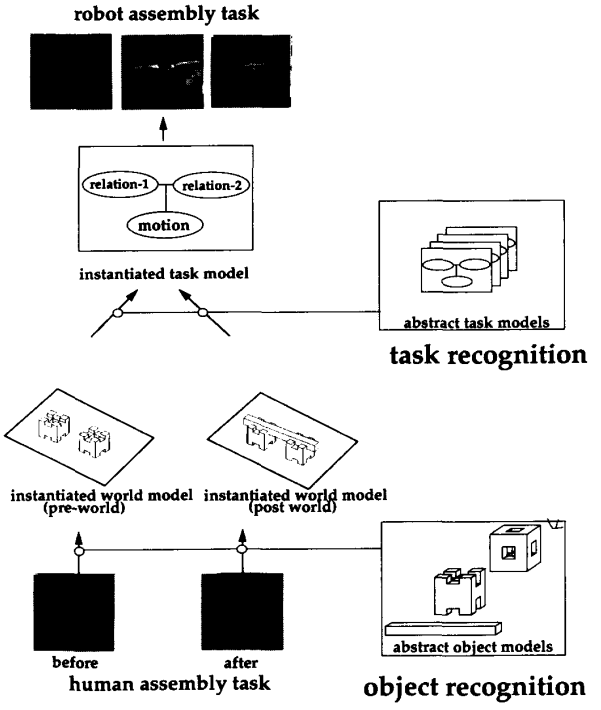


Fig. 1. Object recognition and task recognition.

The instantiated task model associates the transition with a template of a robot command sequence capable of effecting the transition. The model calculates appropriate command parameters to complete the template using object dimensions available in the current and previous instantiated world models. Such command parameters include object locations and grasping locations. The system subsequently inserts the extracted parameters into the command sequence before it sends the completed command sequence to the robot.

III. Defining Abstract Task Models

In order to develop abstract task models, we have to define representations to describe assembly tasks. In this section, we will define assembly relations for such representations. Then, we will consider how to define abstract task models using the assembly relations.

A. Assembly Relation

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

manipulated object achieves a particular relation with the environmental objects.

The primal 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 between the side and bottom faces of the peg (a manipulated object) and the side and bottom faces of the hole (an environmental object). Thus, it is effective to use face contact relations as the central representation for defining assembly task models.

We will define *assembly relations* as face contact relations between a manipulated object and its stationary environmental objects. Note that we do not exhaustively consider all of the possible face contact relations between all of the objects; this would result in a combinatorial explosion of possibilities. We can avoid the exponential complexity by concentrating on a select group of face contacts, namely, those that occur between the manipulated object and the environmental objects.

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

Assembly relations should satisfy two requirements:

1) *Recoverability:* Assembly relations can be extracted from observation, and
2) *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.

Assembly relations satisfy the recoverability requirement.

- *Assembly relations can be obtained by analyzing geometric models.* An object recognition program, such as in [10], can recognize a manipulated object, determine its configuration, and use a geometric modeler to represent the recognition result. In addition, a geometric modeler represent all the other stationary environmental objects. By examining each face pair between the manipulated and environmental objects in the representation given by a geometric modeler, face contact relations between these objects (and thus their assembly relation) can be determined.

Assembly relations also satisfy the inferability requirement:

1) *An assembly relation constrains possible motions:* An assembly relation consists of several face contact relations. At contacting faces, the orientations of surface normals are sufficient 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 move only up or in 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 and from an assembly relation, it is possible to infer the assembly actions that cause such face contact relations.
2) *Assembly relations characterize a control strategy necessary to maintain such relations:* Each face contact relation, a component of an assembly relation, provides a constraint on possible motions. 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 that is first placed on a table and then slid across 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 that 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 also been found to be effective in characterizing required control. Thus, such face contact relations can be used to determine a control strategy necessary to achieve such face contact relations in assembly actions. One of the authors has already implemented a system that uses the face contact relations as the basis for choosing a control strategy [28], [27].

We use such face contact relations as the basic representations in describing an assembly task with a transition between pre- and post-assembly relations. Based on the description, we build abstract task models in the following steps:

1) Classify all possible assembly relations,
2) Consider the kinds of transitions in assembly relations that occur, and subsequently build a graph in which each branch corresponds to one possible transition and each node corresponds to an assembly relation, and
3) Assign manipulator motions to achieve such assembly relation transitions.

B. Taxonomy for Assembly Relation

When considering possible contact relations, we mainly take into account the kinds of translation operations that are necessary for achieving these relations.[1,2]

1) A Constraint Given by a Surface Pair: Let us suppose a surface patch of the manipulated object have a face contact to a surface patch of an environmental object. This surface contact pair constrains the manipulated object's possible translation motion by:

$$\mathbf{N} \cdot \Delta T \geq 0, \tag{1}$$

where ΔT denotes possible translational motion vectors of the manipulated object and $\mathbf{N}$ denotes the normal direction of an environmental surface patch.

[1] By adding a small number of special rotations, such as might be obtained by various screw transformations, we can cover a relatively large number of assembly tasks by pure translations.

[2] All the analysis in this paper are based on kinematics. For mechanical analysis, see [20], [4].

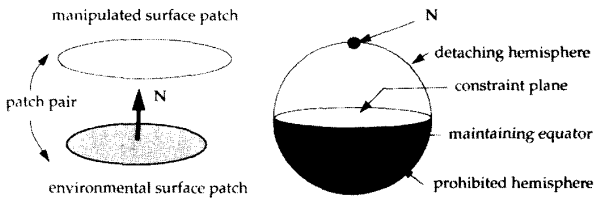


Fig. 2. Constraint inequality depicted on the Gaussian sphere.

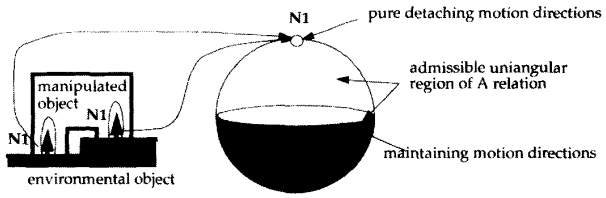


Fig. 3. An A assembly relation. Although the contact patch pairs may have different physical points, they should all have the same direction.

We use points on the Gaussian sphere to specify both a constraint vector and all possible translation vectors. Each vector is translated so that its starting point is located at the center of the Gaussian sphere and its end point exists at some point on the surface of the Gaussian sphere. This point is unique to the vector. We use this point to denote the vector. We can assume that the constraint normal, $\mathbf{N}$, points to the north pole of the Gaussian sphere without loss of generality; the normal is represented as the north pole of the Gaussian sphere.

The constraint from a patch pair defines several regions in the Gaussian sphere. We refer to the plane perpendicular to the normal, $\mathbf{N}$, as the *constraint plane*; this plane divides the Gaussian sphere into two hemispheres. The points on the northern hemisphere, referred to as the *detaching hemisphere* of the constraint, satisfy the strict inequality, $\mathbf{N} \cdot \Delta T > 0$ and denote the motion vectors that break the face contact of the surface pair. The points on the southern hemisphere, referred to as the *prohibited hemisphere* of the constraint, satisfy $\mathbf{N} \cdot \Delta T < 0$ and correspond to prohibited motions that make the manipulated object run into an environmental object. The points on the equator, referred to as the *maintaining equator* of the constraint, satisfy $\mathbf{N} \cdot \Delta T = 0$; the corresponding motions of the manipulated surface patch maintain the face contact. See Fig. 2 for the definitions.

When several surface patches of different orientations make contact, possible motion directions are constrained through simultaneous linear inequalities. These constraints are represented as a combined region in the Gaussian sphere.

2) Unidirectional Contact Relation: Let us consider the relation in which all the contact patch pairs have the *same* direction but may have *different* physical positions. See Fig. 3 for this relation. The constraint inequalities given by the all patch pairs have the same coefficient, $\mathbf{N}_1$, and provides exactly one constraint inequality equation.

$$\mathbf{N}_1 \cdot \Delta T \geq 0 \tag{2}$$

We label this assembly relation as the **A** relation. Each assembly relation provides, as the result of the constraint equation(s), admissible motion directions, all the motion directions of the manipulated object that do not result in collision between the manipulated and environmental objects. The admissible motion directions of the **A** assembly relation form an *uniangular region (hemisphere)*, including its boundary, on the Gaussian sphere. Since the **A** relation has only one constraint inequality equation, its admissible directions are a union of the detaching hemisphere and the maintaining equator of the constraint.[3]

Motions of the directions corresponding to the boundary of the uniangular region (the equator) maintain the **A** assembly relation. Those motion directions will be referred to as the *maintaining* motion directions of the **A** relation. The degrees of freedom of the maintaining motion directions (maintaining DOF) is two. Motions of the directions corresponding to the inside of the uniangular region (hemisphere) break the **A** relation and will be referred to as the *detaching* motion directions of the **A** relation. Any detaching motion can be represented as a spherical convex set of a maintaining motion and a pure detaching motion, which does not contain any maintaining motion component. The pure detaching motion of the **A** relation is along the constraint normal, $\mathbf{N}_1$; its degrees of freedom (detaching DOF) is one.

See Fig. 3 for the **A** assembly relation.

3) Bidirectional Contact: A bidirectional contact provides two constraint inequality equations, namely,

$$\mathbf{N}_1 \cdot \Delta T \geq 0 \tag{3}$$

$$\mathbf{N}_2 \cdot \Delta T \geq 0 \tag{4}$$

The rank of the coefficient matrix of the inequality equations, given by $\mathbf{N}_1$ and $\mathbf{N}_2$, is either one or two.

1) (*Rank* = 2) Two constraint planes, given by $\mathbf{N}_1$ and $\mathbf{N}_2$, intersects and forms an admissible *biangular region* (a crescent) on the Gaussian sphere. We will refer to this relation as a **C** relation. The admissible biangular region of the manipulated object is enclosed by two half circles and two vertices. At the two vertices, both $\mathbf{N}_1 \cdot \Delta T = 0$ and $\mathbf{N}_2 \cdot \Delta T = 0$ are satisfied. Corresponding motion directions maintain this **C** relation; maintaining DOF is one. The points along one of the half circle correspond to the motion directions that maintain one of the two contact relations and break the other relation. Motion directions corresponding to the points inside of the boundary are detaching motion directions. The pure detaching motion directions are denoted by those along the arc that connects $\mathbf{N}_1$ and $\mathbf{N}_2$ on the Gaussian sphere; detaching DOF is two. See Fig. 4(a).
2) (*Rank* = 1) Two normal vectors have either the same or opposite directions. If two vectors have the same

[3]Note that the detaching hemisphere and the maintaining equator are the concept defined for each constraint inequality, while admissible motion directions are the concept defined at each assembly relation as the the results of combined constraint inequalities. This is also true for the concepts of maintaining, detaching and prohibited motion directions defined in the next paragraph.

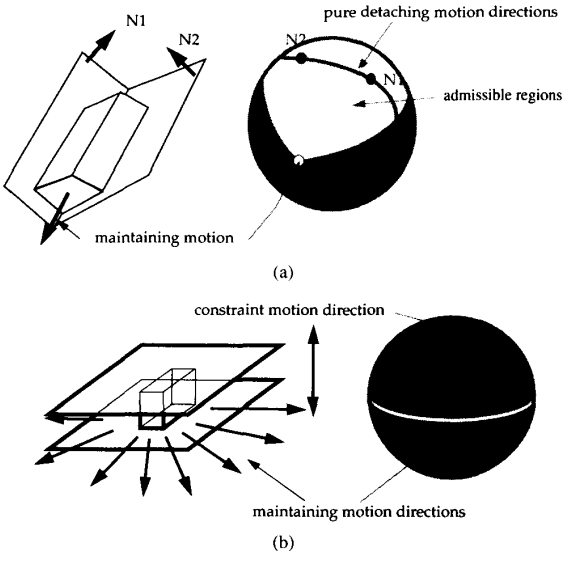


Fig. 4. Bidirectional contact; (a) C assembly relation. (b) B assembly relation.

direction, this contradicts the definition of bidirectional contact. Then, the two vectors should have the opposite directions with each other. These two directions are represented as a pair of poles on the Gaussian sphere. The admissible directions of the manipulated object can be represented as *the entire great circle* perpendicular to the axis connecting the two poles. Namely,

$$\mathbf{N}_1 \cdot \Delta T = 0$$
$$\mathbf{N}_2 \cdot \Delta T = 0$$
$$\mathbf{N}_1 = -\mathbf{N}_2$$

We refer this assembly relation as a **B** relation. See Fig. 4(b) for this relation.

Any motion direction corresponding to a point along the admissible great circle satisfies the previous two equations; all the admissible directions of a **B** relation maintain a **B** relation; and the maintaining DOF is two. There are no detaching motions; the detaching DOF is zero. One direction along the axis connecting $\mathbf{N}_1$ and $\mathbf{N}_2$ is completely constrained. We will refer to this as the constraint direction; the degree of freedom of the constraint directions (constraining DOF) is one.

4) Tridirectional Contact: A tridirectional contact provides three constraint inequality equations listed below:

$$\mathbf{N}_1 \cdot \Delta T \geq 0 \quad (5)$$
$$\mathbf{N}_2 \cdot \Delta T \geq 0 \quad (6)$$
$$\mathbf{N}_3 \cdot \Delta T \geq 0 \quad (7)$$

The rank of this coefficient matrix is either three, two, or one.

1) (*Rank 3*) The three constraint planes intersect at the center of the Gaussian sphere, and form an admissible *triangular region* on the Gaussian sphere. We will refer this relation as an **F** relation. See Fig. 5(a). All admissible motion directions are detaching motions; motion directions corresponding to the inside region break contact relations simultaneously; those along the boundary of the triangular break two of the three contact relations and maintain the remaining one; those at one of the vertices break one and maintain two contact relations. Thus, detaching DOF of **F** is three; the maintaining DOF and constraining DOF are both zero.
2) (*Rank 2*) The three normal vectors are coplanar. We can further subdivide this case into two sub-cases.
   a) If all of the two vectors are mutually independent, (all sub-matrices given by the two constraint inequality equations have a rank of 2), the three vectors form either an admissible biangular region

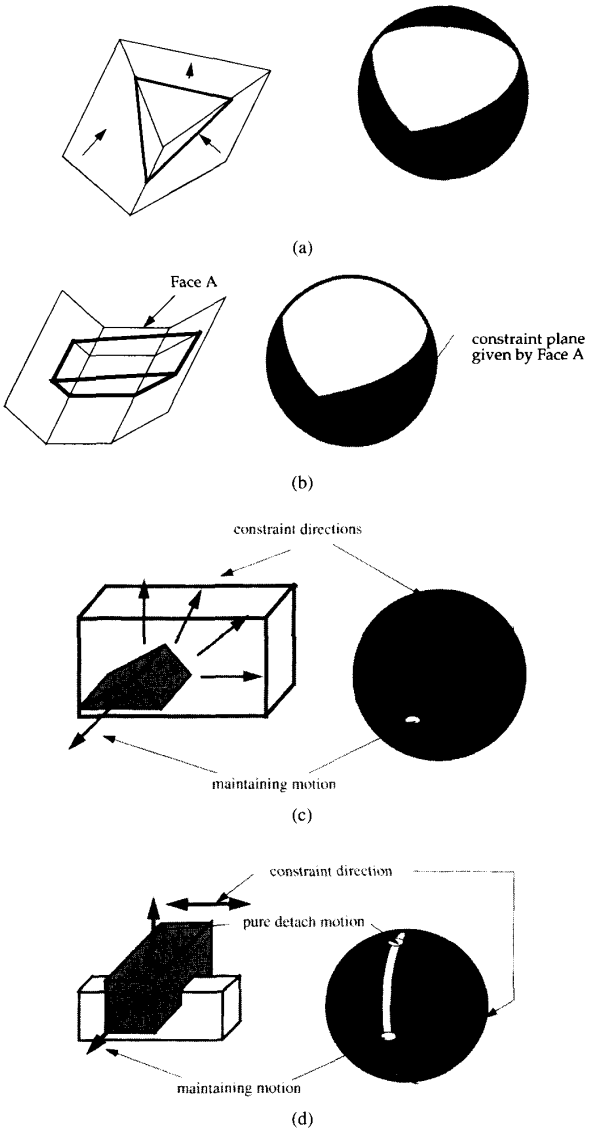


Fig. 5. Tridirectional contact; (a) F assembly relation. This relation is a general case; (b) a tridirectional contact relation equivalent to a C bidirectional assembly relation; (c) E assembly relation; (d) D assembly relation.

(which is equivalent to **C**, see Fig. 5(b)) or a pair of admissible pole points whose axis is perpendicular to the coplanar plane, i.e., the **E** relation. See Fig. 5(c). In the **E** relation, all the admissible motions are maintaining motions; the maintaining DOF is one. The constraint directions are on the coplanar plane; the constraining DOF is two. Since **E** does not have any detaching motions, the detaching DOF is zero.

b) If one of the two vectors are opposite each other (one sub-matrix has rank 1), then these two vectors form an admissible great circle that is perpendicular to the vectors, $\mathbf{N} \cdot \Delta T = 0$. The third vector, which is linearly independent from the previous two vectors, further cuts the whole great circle into one half great circle. Thus, a *half great circle* is the admissible direction; **D**. See Fig. 5(d). The maintaining motion directions are those corresponding to the two end points of the half great circle; maintaining DOF is one. The pure detach motion is along the third vector direction; detaching DOF is one. The pair of the opposition vectors provide the constraint direction; constraining DOF is one.

3) (*Rank 1*) Due to the definition of tridirectional contact, this case does not occur.

5) Tetradirectional Contact: A tetradirectional contact provides four constraint inequality equations. We will use the inductive method to consider this contact. We can arbitrarily choose three vectors from the vectors forming the tetradirectional contact. These three vectors form a tridirectional contact that has four different classes of admissible regions: triangular region, biangular region, a half great circle, and a pair of polar points. By adding a fourth vector, we will examine how these admissible regions form their shapes.

1) *Triangular Region:* The constraint plane of the fourth constraint $\mathbf{N}_4 \cdot \Delta T = 0$ can either be intersecting or non-intersecting with the triangular region.

a) *Non-Intersection:* Depending on whether the triangular region exists in the detaching hemisphere or in the prohibited hemisphere, one of the following two cases occur (see Fig. 6):

- Detaching hemisphere: $\mathbf{N}_4 \cdot \Delta T > 0$. The new admissible region has the same shape as the previous triangular region. This is the same as the **F** relation.
- Prohibited hemisphere: $\mathbf{N}_4 \cdot \Delta T < 0$. There is no admissible region. We will label this relation as **I**. Since all the degrees of freedom are constrained, the constraining DOF is three.

b) *Intersection:* Depending on how the fourth constraint plane crosses the triangular region, the following three cases occur:

- General Case: The constraint plane divides the triangular region into two subregions. Both subregion shapes are either triangular or tetra-angular. Depending on the direction of the detaching hemisphere, one of them becomes the new admissible region; the admissible region is either a triangular or tetra-angular. See Fig. 7(a).

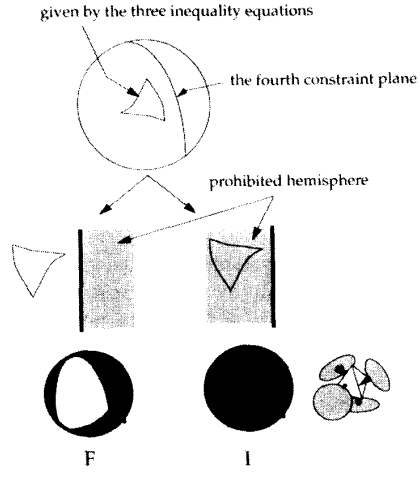


Fig. 6. Tetradirectional contact (non-intersection case).

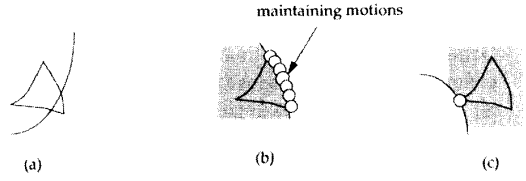


Fig. 7. Tetradirectional relation (intersection case).

In particular, a tetra-angular admissible region over-constraints the manipulated object simultaneously from four different directions. For our purposes, this tetradirectional contact (tetra-angular) has the same effect as an **F** tridirectional contact; the same control strategy is applicable to both types of contact. Namely, by achieving any three contacts among the tridirectional contact, we can also achieve the remaining contact automatically. Thus, we consider this relation equivalent to **F**.

- The fourth constraint plane coincides with one of the edges of the admissible triangular region. Then, the fourth constraint vector is either in the same or in the opposite direction as the constraint vector of the edge. However, given the definition of the tetradirectional contact (that is, contact in four different directions), the fourth constraint vector cannot be in the same direction as the vector; it should be in the opposite direction from the vector. The previous admissible triangle exists in the prohibited hemisphere of the new constraint. The new admissible region will be only along the edge of the triangular region, *an arc of a great circle*. We will label this relation as **G**. See Fig. 7(b). Any motion directions within the arc is a pure detach motion; detaching DOF is two. The direction perpendicular to the arc is the constrain direction; the constraining DOF is one. There is no maintaining motion; the maintaining DOF is zero.

- The fourth constraint plane passes through one of the vertex of the triangular region. The fourth and the other two normal vectors that form the vertex are coplanar.

  The entire triangular region exists either in the detaching or prohibited hemisphere of the new fourth constraint. On the admissible sphere, the new constraint does not effect; it is equivalent to **F**.

  On the prohibited sphere, the new admissible region is only the vertex (*a point*). We will label this relation as an **H** relation. See Fig. 7(c). Motion directions toward the point break the contact; detaching DOF is one. The plane perpendicular to the detach motion is the constraint plane; the constraining DOF is two. There is no maintaining motion; the maintaining DOF is zero.

2) *Biangular Region:* This case occurs when three vectors are coplanar and two of them form the constraint biangular region; an example of this is the relation in Fig. 5(b). Depending on the direction of the fourth vector, one general case and three special relations occur. However, all four relations are equivalent to one of the previously discussed relations.
   a) The fourth constraint normal is not coplanar to the other three normals. The fourth constraint plane crosses the biangular region and forms a new triangular admissible region. This contact relation is equivalent to **F**.
   b) The fourth constraint normal is coplanar to the three normals. Depending on the direction of the fourth normal, the admissible region becomes either the same biangular region (equivalent to **C**), the boundary of the biangular region (equivalent to **D**), or a pair of polar points (equivalent to **E**).
3) *A Half Great Circle:* This is a special case of the previous biangular region. Not only must the three vectors be coplanar, but, additionally, two of them must be directionally opposite from each other. Basically, these three constraints form a region of common detaching motions that are equivalent to the **D** relation. If the fourth vector is not coplanar with the previous three vectors, the admissible region becomes a partial arc of the half great circle (equivalent to **G**). If the fourth vector is coplanar, there is a vector in the opposing direction that is not part of the previous opposites pair; the fourth vector form a pair of opposites with the vector. So, the four vectors are coplanar and form two pairs of opposing vectors. Then, the admissible region becomes a pair of pole points (equivalent to **E**).
4) *A Pair of Pole Points:* Depending on the direction of the fourth constraint normal, the admissible two pole points remain either unchanged or one of them falls into the prohibited hemisphere (a single admissible point that is equivalent to **H**).

By using a similar inductive method as in the tetradirectional contact relation, we can prove that these nine relations are sufficient for describing the general directional constant ($n > 4$).

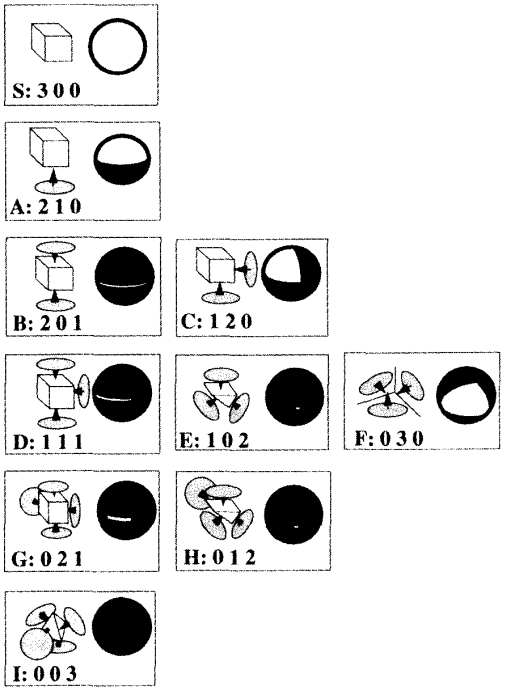


Fig. 8. Ten assembly relations; the white areas in the Gaussian sphere denote admissible regions of the assembly relations. The three digits denote maintaining DOF, detaching DOF, and constraining DOF, respectively.

For the sake of completeness, we add the no-face contact relation (**S**) to our taxonomy. All motions will continue to maintain the same **S** relation; the maintaining DOF is three. Both the constraining DOF and detaching DOF are zero.

In general, an n-directional contact relation is classified as one of the ten contact relations as shown in Fig. 8. The three digits in the figure are the maintaining DOF, the detaching DOF, and the constraining DOF, respectively. Note that the sum of the maintaining DOF (D_m), the detaching DOF (D_d) and the constraining DOF (D_c) is three: $D_m + D_d + D_c = 3$.

C. Assembly Relation Transitions

This section determines a sequence of manipulator operations necessary to achieve each assembly relation from **S** assembly relation. 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 with appropriate control parameters, 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 the control parameters, but not the strategy.

We utilize intermediate relations in order to reduce the number of necessary motion macros. We first consider macros for simpler relations. For more complicated relations, instead directly achieving it from **S**, we attempt to achieve a

TABLE I
POSSIBLE ASSEMBLY RELATION TRANSITIONS

Relation	Possible intermediate relations	Selected disassembly transition(s)	Selected assembly transition(s)
A	A direct detach motion gives the transition from the assembly relation **A** to **S**. See Fig. 9(a) for an example of a direct detach motion that causes such a transition.	**A-to-S**	**S-to-A**
B	No direct detach motions can be applied to assembly relation **B**. Lateral motions parallel to the contact force can be applied. Depending on the shape of the contact faces, it reaches either **S** or **A**. Since this variation is due to the shape of the contact faces, we include both transitions. Fig. 9(b) shows these transitions.	**B-to-S** **B-to-A**	**S-to-B** **A-to-B**
C	By applying direct detach motion, the **C** relation reaches either **S** or **A**, depending on the motion directions. The transition from **C** to **S** reduces the number of constraints by two, while the transition from **C** to **A** reduces the number by one. The latter relation transition is chosen by virtue of criterion 3. Fig. 9(c) shows two possible transitions due to motion directions.	**C-to-A**	**A-to-C**
D	A direct detach motion gives the transition from **D** to **B**.	**D-to-B**	**B-to-D**
E	No direct detach motion exists in the case of **E**. Lateral motions along the axis parallel to the surrounding contact faces (the insertion axis) cause several relations, **S**, **A**, **B**, **C**, and **D**, depending on the shape of the contact faces. We include these five possible transitions. See Fig. 9(d) for an example.	**E-to-S** **E-to-A** **E-to-B** **E-to-C** **E-to-D**	**S-to-E** **A-to-E** **B-to-E** **C-to-E** **D-to-E**
F	The assembly relation **F** reaches either **S**, **A**, or **C** by detaching motion, depending on its direction of motion. The relation transitions **F-to-S**, **F-to-A**, and **F-to-C** reduce the number of constraints by three, two, and one, respectively. Thus, following criterion 3, the relation transition **F-to-C** is chosen as the desirable one.	**F-to-C**	**C-to-F**
G	Among the two possible relation transitions **G-to-B** and **G-to-D**, the relation transition **G-to-D** is chosen using criterion 3.	**G-to-D**	**D-to-G**
H	By motion along the insertion axis, assembly relation **H** reaches **E**.	**H-to-E**	**E-to-H**

known intermediate relation using a known template, and then achieve the goal relation using a new template. We begin this approach with the unidirectional contact (**A** relation) and iteratively increase the number of contacts through tetradirectional contacts (**G** and **H** relations) in the assembly relation taxonomy.

Disassembly actions are considered for finding an appropriate intermediate relation. We consider disassembly actions from the assembly relation, and extract all possible immediate intermediate assembly relations just prior to the assembly relation. This approach is taken because disassembly actions are easier to infer than assembly actions.

Non-unique intermediate relations may exist from the same assembly relation due to 1) the variation in shapes of contact faces, or 2) the variety of possible disassembly operations.

In the case of contact face shape variations, we have to analyze all intermediate relations and assign appropriate motion macros to all transitions from the intermediate relations to the desired relation.

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

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

The results of the application of these criteria to the analysis of each assembly relation, extraction of all possible assembly relation transitions, and pruning of unnecessary relation transitions are shown in Table I.

We can represent relation transitions as a directional graph, as shown in Fig. 10. Each node in the graph represents an assembly relation, and each arc represents an assembly relation transition.

D. Abstract Task Models

This section creates thirteen task models corresponding to all possible relation transitions (represented by the arcs in Fig. 10). Each abstract 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. Table II explains the detailed method to select appropriate macros. The basic structure of a motion macro is to perform the simplest maintaining motion of an assembly relation until the next assembly relation occurs.

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

1) *Move:* A motion sequence for this macro is realized by translating a manipulated object from the starting configuration to the ending configuration.
2) *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,

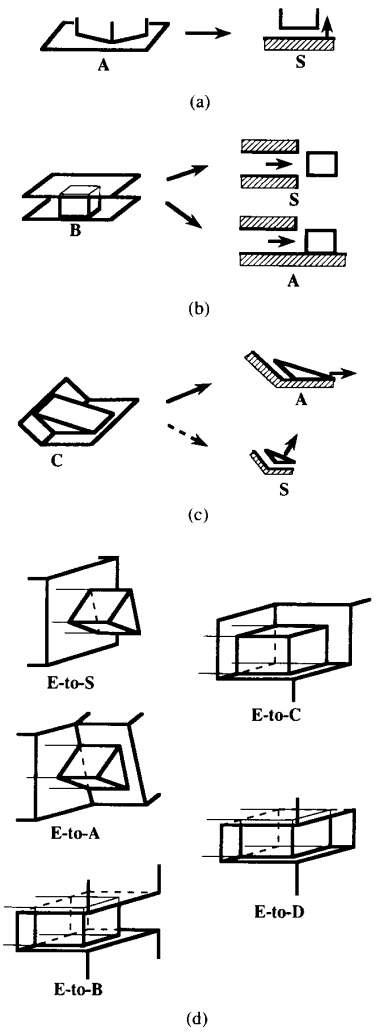


Fig. 9. Examples of assembly relation transitions.

then aligning a manipulated object face to the contact environmental face.

If we have precise configurations, we can achieve the contact and aligning operations by using such configurations. Otherwise, these operations require some sensory feedback to detect the occurrence of contact and aligning. See [28] for a detailed implementation of the macro as a skill in a force feedback type manipulator.

3) *Insert-Between:* A motion sequence for this motion macro is realized by first fitting 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 fitting motion and translation motion using the configurations. Otherwise, the align motion requires some sensory feedback. See [28].

4) *Insert-Into:* A motion sequence for this motion macro is realized by fitting a manipulated object along the insert axis, and then translating along the axis to the ending configuration.

If we have precise configurations, we can use this information to achieve the fitting and translation motions. Otherwise, the fitting motion requires some sensory feedback. See [28].

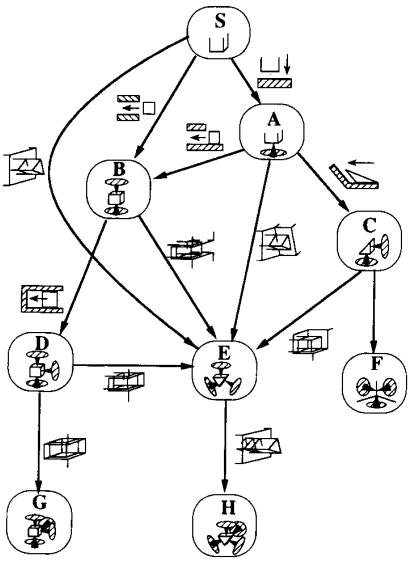


Fig. 10. Assembly relation transitions represented as a directional graph.

Fig. 11 depicts motion macros attached in the graph of relation transitions.

Each task model is represented as a frame. Each frame contains slots; some are filled and others are empty. For example, Fig. 12 shows the frame for the **S-to-A** abstract task model. The starting and end relation slots contain values, **S** and **A**, respectively. The action slot contains the move-to-contact motion macro. Several parameter slots for the motion macro are empty; the values are obtained by the demons attached to each slots when this abstract model is instantiated. Such parameter slots include:

1) The entry configuration—starting configuration of the task model to be obtained from the object configuration in the instantiated pre-world model;
2) The approach configuration—the configuration to transit from the simple move operation to the one to move with sensing the contact given from the end configuration;
3) The goal configuration—end configuration to be obtained from the object configuration in the instantiated post-world model;
4) The approach direction—the command direction to be obtained from the direction of the contact face in the instantiated post-world model,
5) The grasp point—where to grasp; in the current implementation, several candidate grasp points are registered to each abstract object model. One is chosen by considering the collision among the environmental objects in the instantiated post-world model.

Other abstract task models have similar structures.

TABLE II
MOTION MACRO

Task model	Assigning motion macro	Macro name
S-to-A	The relation transition from **S** to **A** is realized by an attaching motion containing a component toward the contact face(s) of environmental object(s). Among several attaching motions, we select pure attaching motion, a motion along the inverse direction of the contact normal, for simplicity. The pure attaching motion will be continued until face contact occurs. We refer to this template as the *move-to-contact* motion macro.	move-to-contact
S-to-B	In order to effect the relation transition from **S** to **B**, we first have to fit the configuration of the object across the gap between the two contact faces. Then we have to translate the object along the contact faces. We refer to this template as the *insert-between* motion macro.	insert-between
A-to-B	The maintaining motions of the **A** relation are those parallel to its constraint plane. It is only necessary to translate the object along its constraint plane until the goal configuration of the object is reached; the bidirectional contact should occur automatically. We refer to this template of simple translation motion as the *move* motion macro.	move
A-to-C	Among the maintaining motions of the **A** relation, attaching motions toward the new constraint planes realize the transition from **A** to **C**. Among several such motions, pure attaching motion perpendicular to the intersection lines between two constraint planes is selected. We achieve this relation by using the same template of operations for the **S-to-A** task model, move-to-contact. Thus, we assign the move-to-contact macro to the **A-to-C** task model.	move-to-contact
B-to-D	The maintaining motions of relation **B** are those parallel to the two opposite constraint planes. Attaching motions toward the third constraint plane realize the relation transition from **B** to **D**. Among several attaching motions, as is the case in **A** to **C**, pure attaching motion is selected. For this task model, we use the same macro as in **A-to-C**, move-to-contact.	move-to-contact
S-to-E	For the relation transition from **S** to **E**, we first have to align the configuration of the object so that it can be translated along the insert axis. Then we have to translate the object along the axis. We refer to this template as the *insert-into* macro. Note that the insert-between macro only fits the object parallel to a pair of contact faces. The macro allows rotation and translation freedom along the contact faces. On the other hand, the insert-into macro does not allow such freedom, it only allows the object to translate along the insert axis.	insert-into
A-to-E	The maintaining motions of the relation **A** already constrain one degree of freedom (DOF) among the two degrees of freedom of the constraint DOF of the relation **E**; the increment of the constraint DOF is one from **A** to **E**. Thus, we can use the insert-between motion macro to satisfy the increment of the constraint DOF; we use the macro to fit the object to the hole by aligning the configuration perpendicular to the constraint normal of relation **A**.	insert-between
B-to-E	This is the same as in **A-to-E**.	insert-between
C-to-E	The maintaining motions of relation **C** are those along the insert axis. We use the move macro to make the relation transitions.	move
D-to-E	This is the same as in **C-to-E**.	move
C-to-F	We can realize relation **F** by performing the maintaining motion of relation **C** until tridirectional contact occurs. We use the move-to-contact macro.	move-to-contact
D-to-G	The maintaining motions of relation **D** are those to translate along the line connecting the two end points of the half great circle. We use the move-to-contact macro along this maintaining motion until the object achieves tetradirectional contact.	move-to-contact
E-to-H	The relation transition from **E** to **H** is achieved by the move-to-contact macro along the insert axis (the maintaining motion of the **E** relation) until tetradirectional contact occurs.	move-to-contact

IV. IMPLEMENTATION OF TASK RECOGNITION MODULE (TRM)

How are abstract task models used to characterize human assembly tasks? Basically, the task recognition module (TRM) collects two different kinds of information from observation:

1) Which assembly relation transition occurs, and
2) Where the assembly task should be performed.

The TRM selects and instantiates the abstract task model corresponding assembly relation transitions between two instantiated world modes. The instantiated task models contain several empty slots for motion parameters with their demons. Each demon at the slot calculates necessary motion parameters to complete the task model from the object configurations in the instantiated world model, when the task model is instantiated.

The task recognition mechanism will be explained in the following examples. In the example, the system consists of three classes of objects, (any of which can appear in the scene): castle, block, and stick (Fig. 13).

A. Temporal Segmentation

The task recognition module (TRM) assumes that at the beginning of each assembly task, human intervention occurs in the scene, and at end of the assembly task, the human disappears from the scene. With this assumption, the TRM segments a continuous image sequence given by a TV camera from the scene into a finite number of meaningful chunks.

The system can detect human intervention by monitoring the brightness difference between consecutive images. Fig. 14(a) shows an example scene where a human operator is putting a castle on the table, while Fig. 14(b) shows a continuous brightness image sequence of the scene. Prior to human intervention, the scene consists of only stationary objects, hence, the difference between two consecutive images is insignificant. When human intervention occurs, the brightness difference is significant due to the appearance of the human arm and hand, and the motion of the manipulated object in the scene. This disturbance continues until the end of the assembly operation. After the human hand disappears, the scene once

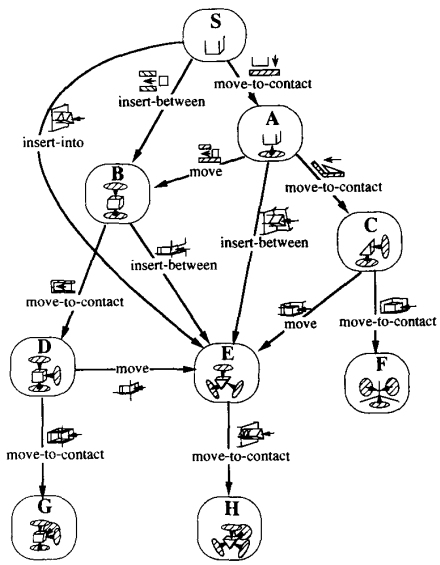


Fig. 11. Motion macros assigned to the relation transitions.

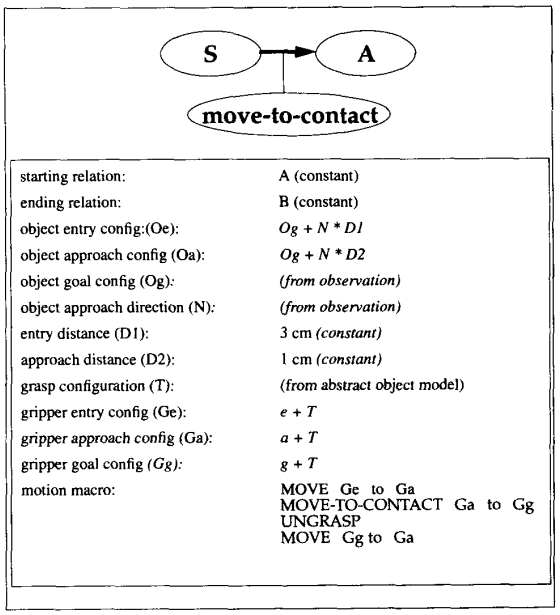


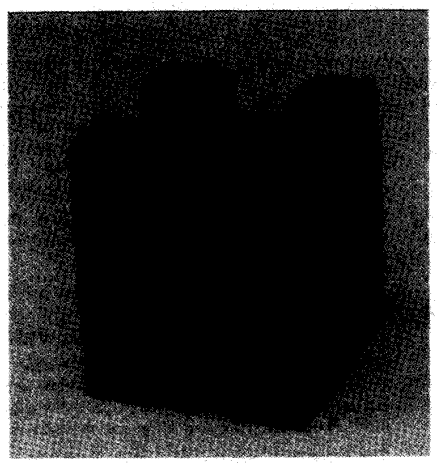
Fig. 12. The S-to-A task model.

again consists of only stationary objects. Thus, the brightness difference returns to an insignificant level.

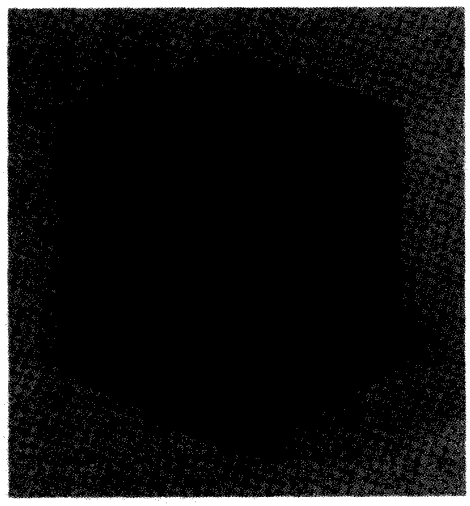
We have been successfully using this method for temporal segmentation on many occasions.

B. Object Recognition

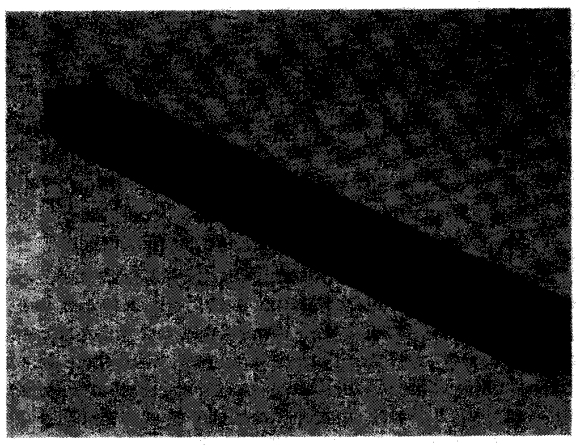
Objects in the scene are recognized from range data. In our current implementation, b/w images are used only for detecting the completion of one assembly task. Range data, which are more reliable, are used for analyzing the scene. After a certain



(a)



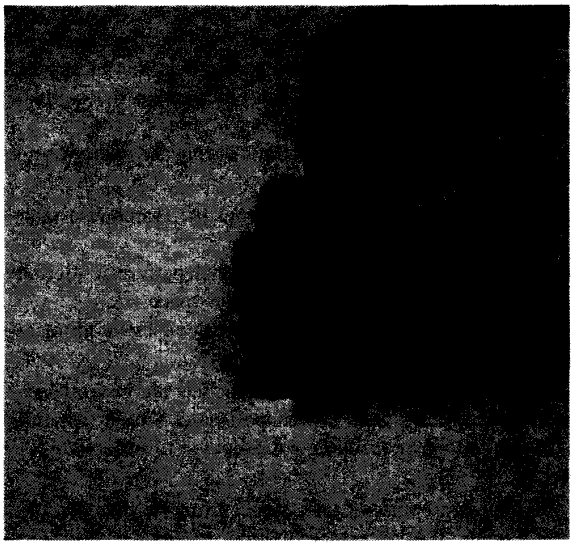
(b)



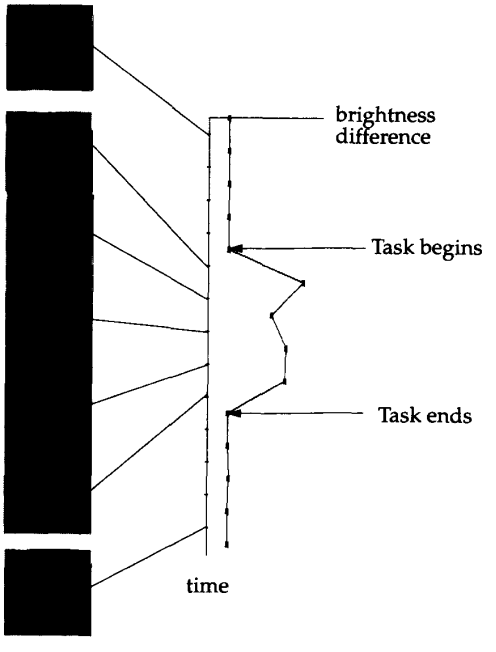
(c)

Fig. 13. Castle, block, and stick.

period after the detection of the completion of one assembly task, the TRM triggers the range finder and collects range



(a)



(b)

Fig. 14. The system detects human intervention from the change in brightness values; (a) A human operator is putting a castle on the table, (b) A continous brightness image sequence of the scene.

data of the scene. The TRM 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).

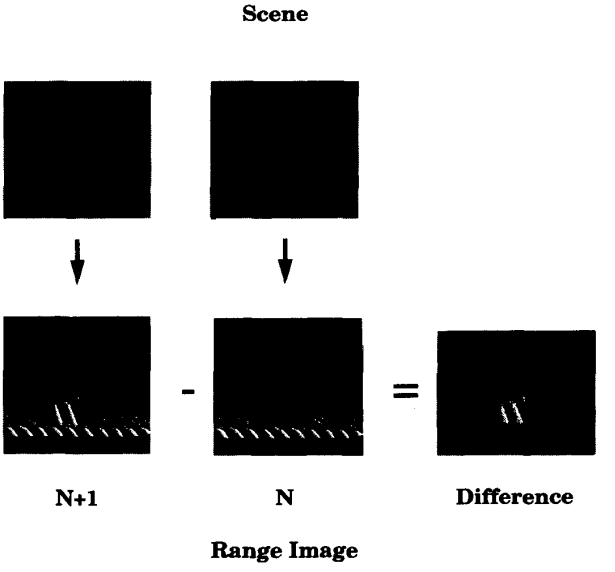


Fig. 15. The difference in range data.

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. Use of the difference image provides two benefits to the system. Since there are few appearing regions in the image, the recognition becomes quicker. Even if there are unrelated background objects, including environmental objects, in a image, they are weeded out in the difference image because they are stational. Secondly, the recognizer can concentrate for the recognition of the manipulated object; we can expect more reliable recognition results. See Fig. 15.

An object recognition module analyzes the new regions and identifies the manipulated object by comparing model features with image features. Subsequently, it determines the configuration of the manipulated object by fitting a geometric model to range data on the regions [32]. The system represents these manipulated and environmental objects in the Vantage geometric modeler [3]. The resulting representation is referred to as the current instantiated world model. Fig. 16 shows one of such instantiated world models.

C. Task Identification

The configurations of the faces of the manipulated and environmental objects are obtained by transforming body coordinate systems to face coordinate systems (available from the Vantage geometric modeler).

The system extracts contacting face pairs from the face configurations. Here, a contacting face pair is formed by 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 in direction to each other.

The system determines the assembly relation by analyzing the distribution of contact directions, the normal directions from the environment faces to the manipulated object faces. The contact pairs are grouped into a set of groups so that each

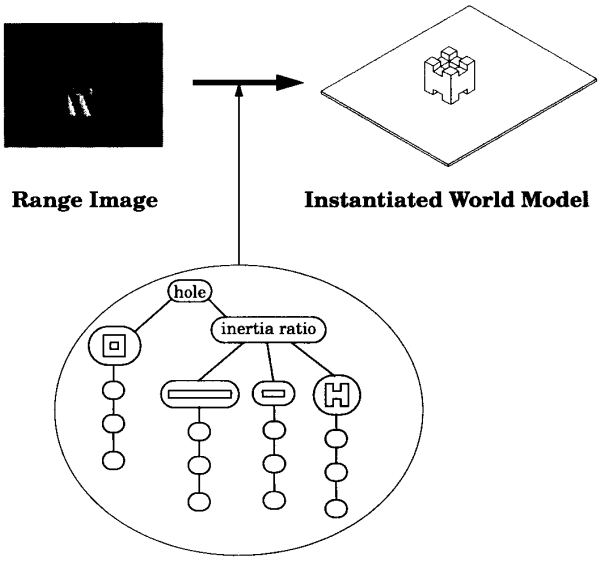


Fig. 16. Object recognition: a recognition program is applied only to any newly appearing regions, and recognizes only manipulated objects. The recognition results are represented in Vantage as an instantiated world model.

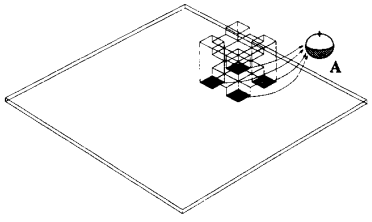


Fig. 17. Extracted contact faces and assembly relation.

group has face pairs with the same contact direction. We iteratively examine the linear independence of the contact groups generated, change the current relation one after another by finding a linear independent group, and identify the assembly relation that occurs in the assembly task.

The system recognizes the contact faces and contact directions as shown in Fig. 17. From the contact faces in Fig. 17, the system recognizes the current assembly relation as **A**.

Prior to the assembly task, the castle was not in the scene. Thus, before the assembly task, the assembly relation between the castle and the table was **S**. After the assembly task has been completed, the manipulated castle established an **A** assembly relation with the environmental object, namely, the table.

From this observation, the system recognized that the assembly relation transition, **S-to-A**, occurred in the assembly task. The **S-to-A** task was subsequently extracted from the corresponding arc along the procedure graph.

D. Task Instantiation

In the current implementation, the vision system only observes human assembly actions that occurred on the table; it does not observe how the human picks up a new object from the warehouse table (disassembly action from the warehouse). This knowledge together with the number and location of the objects, are given to the system *a priori*. In this example, the system knows that the castle was located on the warehouse table with **A** assembly relation.

The assembly relation transitions during the entire assembly task are:

1) **A-to-S:** Detach the castle from the warehouse table to the departure configuration.
2) **S-to-S:** Bring the castle from the departure configuration to the start configuration in free space.
3) **S-to-A:** Move-to-contact the castle to the working table from the start configuration.

Thus, the corresponding three task models are instantiated: **A-to-S** (disassembly), **S-to-S** (transport), and **S-to-A** (assembly).

The following procedures are executed to instantiate a task model:

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

The instantiation of task models occurs in the reverse order, i.e., **S-to-A** (assembly), **S-to-S** (transport), and **A-to-S** (disassembly).

The **S-to-A** task model has a *move-to-contact* motion macro in the action slot. The **S-to-A** task model has five major motion parameters: *starting configuration*, *approaching configuration*, *goal configuration*, *grasping configuration*, and *approach direction*. The task model determines the *goal configuration* motion parameter from the configuration of the castle in the post-world model. The approach-direction parameter is given from the contact normal of the castle in the post-world model. The approach configuration is given by the contact normal in the post-world model. The approach and starting configuration is the one translating the goal configuration along the contact normal by appropriate constant distance (currently three centimeters and one centimeter). Each abstract object model has several candidate configurations for grasping. The task model recalculates them based on the current body configurations, examines them against collision with environmental objects, and determines an appropriate one. The demon 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 end configuration of the disassembly operation (**A-to-S**) at the warehouse table to the high position, the high position to another high position above the starting configuration of the assembly operation (**S-to-A**) at the working table, and the second high position to the configuration. These configurations are obtained from the

Fig. 18. Putting a block on the table with a manipulator.

old and new configurations of the manipulated objects. These values are inserted into their slots in the instantiated task model by demons.

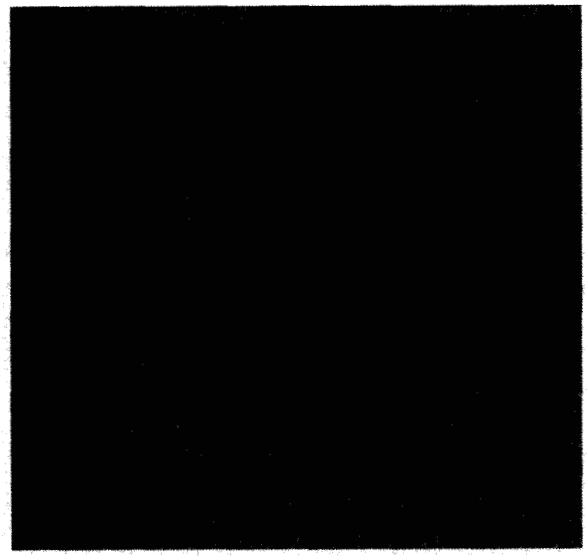
The disassembly task (**A-to-S**) is also implemented as a task model. The current implementation does not observe the warehouse table due to the limited field of view of the range finder. Thus, the assembly relation transition, **A-to-S**, which occurs at the warehouse table, is given to the system as *a priori* knowledge. The system instantiates an **A-to-S** disassembly task model. This task model has a motion macro, *move* in the action slot.

The system finally performs the operations given by the three task models sequentially: **A-to-S**, **S-to-S**, and **S-to-A**. Fig. 18 shows the final move-to-contact operation by a manipulator.

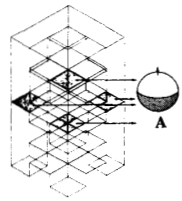
Fig. 19(a) shows the human operation to put a block on top of the castle. The system recognizes the contact faces and the task model corresponding to **A**. The system subsequently performs the put-on operation.

Fig. 20(a) shows a human operation of inserting a stick into a hole of the block. The system recognized the contact faces (Fig. 20(b)). From the normal direction of contact faces, the system then recognized the existence of a tetradirectional contact. By examining the directions of the contacts, the system identified that the observed assembly relation as **E**.

The relation transition from **S** to **E** corresponds to four paths: direct path, via **A**, via **B**, and via **C**. All the arcs to the **E** relation, however, have the same assembly action (and disassembly action) of 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.



(a)

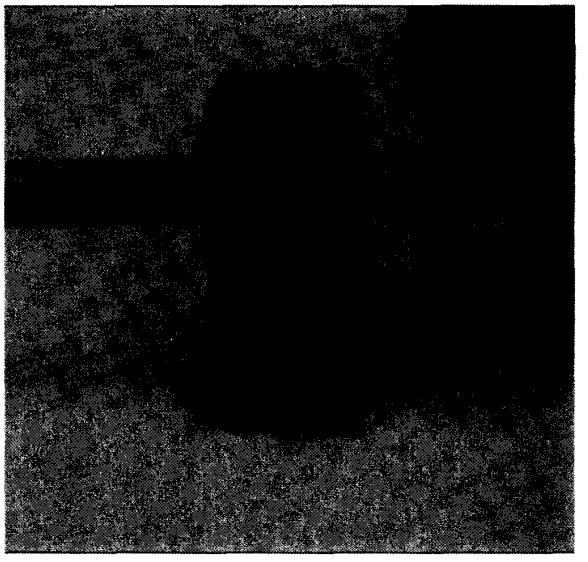


(b)

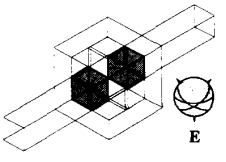
(c)

Fig. 19. Put a block on the castle; (a) input scene, (b) face contact, (c) system performance.

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 **S-to-E** relation transition has occurred.



(a)



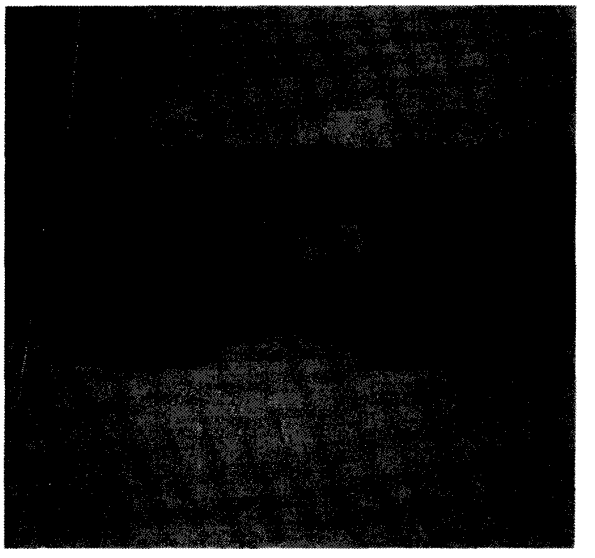
(b)

(c)

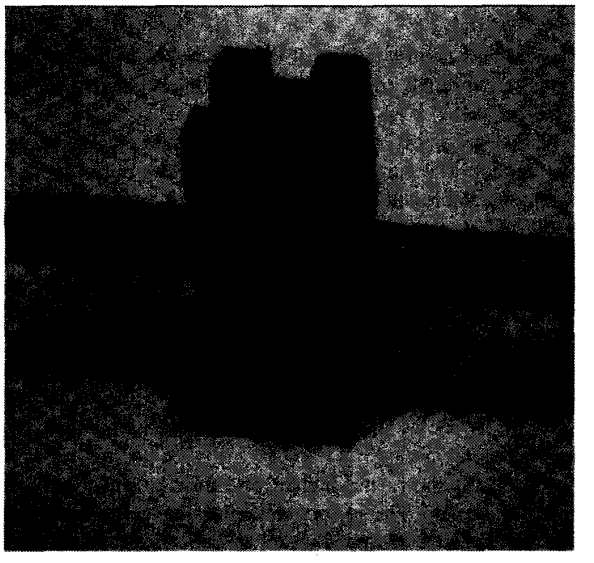
Fig. 20. Insert a stick to the block; (a) input scene, (b) face contact, (c) system performance.

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 Fig. 20(c).

Fig. 21 shows other examples that have been successfully constructed by the system.



(a)



(b)

Fig. 21. Additional examples.

V. Concluding Remarks

A. Summary

We have described the task recognition module, or TRM, that can observe an assembly task performed by a human, recognize scene objects, recognize relations among those objects, and associate these relations with the corresponding assembly task. The TRM is a component of our larger effort to develop a complete APO system.

B. Toward a Complete APO System

Some of current and future research directions toward a complete APO system include:

1) Error Correction: The current system extracts two different kinds of information from observation: face contact relations and motion parameters. It often occurs that, although the current face contact relation is correctly obtained, there are errors in the motion parameters that ultimately produce a failure in the manipulator's performing the assembly. Since we classify the current relation into one of the ten distinct patterns, this process is robust due to this discretization. On the other hand, the motion parameters are obtained by simply converting object configurations; observation errors are propagated to the final motion parameters.

We are developing a method to clean up noise-contaminated motion parameters from the correctly obtained face-contact relations [26]. A manipulated object has a face contact relation with other environmental objects through several contact face pairs. To each contact face pair, one face contact equation is established. Thus, we can set up simultaneous face contact equations among the manipulated and environmental objects. These equations are redundant and non-linear.

Observation provides two key pieces of information: which object is the manipulated object and what are the approximated object configurations. We need to consider, either directly or indirectly, only those face contact relations related to the manipulated object. Thus, by identifying the object, we can reduce the number of relations necessary to be considered.

Approximated object configurations allow us to linearize a set of simultaneous non-linear equations. The original equations contain redundant parameters. We will linearize and solve the equations only with respect to those parameters that correspond to non-redundant degrees of freedom. We can assign the values obtained from observation to the parameters that correspond to the redundant degrees; we do not need to modify these parameters. Using these newly obtained solutions, we can update the motion parameters, thereby achieving a system that is highly robust to observation noises.

2) Grasp Recognition: Currently, predetermined grasping strategies of an object are stored with its abstract object model. When a task model is being instantiated, the stored grasping strategies are retrieved. The system individually examines each strategies to determine if implementing that strategy would result in a collision between the robot fingers and environmental objects, or not. This process produces a candidate set of collision free grasping strategies. Then, one of the collision free grasping strategies is selected using an evaluation function. Hutchinson and Kak also reported a similar system, SPAR [9]. A human operator provides SPAR with all the stable poses and all the permissible grasp points on each object. Next, SPAR examines the initial poses of the objects on the work table and automatically synthesizes all the grasping and regrasping operations required to bring about a given assembly. However, these pre-stored methods rapidly become inefficient with an increase in either the number of objects or the number of grasping strategies.

We recognize obtaining grasping strategies by analyzing image sequences during human assembly operations [13]. Currently, we classify human grasping strategies using a proposed hand representation called the *contact web* in conjunction with a grasp taxonomy. From a sequence of images of a (human) assembly operation, we track human finger movements, and then determine the contact web associated with the grasp. The grasp is recognized based on the analysis of the spatial distribution of the contact web and a grasp taxonomy that we have developed.

3) Object Recognition: This system needs object recognition capability as one of its basic components. The current recognition program was hand-coded by a programmer. We need to avoid hand-coding object recognition programs in order to ensure that we do more than simply substitute one difficult problem (hand-coding assembly programs) for another difficult problem (hand-coding recognition programs). We have been developing vision algorithm compilers (VAC's) that automatically convert CAD models into recognition programs [11], [32].

4) Skill Library: The current TRM employs task models that contain manipulator control sequences based on pure position control. For the complete manipulation system, we have to expand our task models so that they also include control sequences based on other modes of control, namely force and compliant control. One of the authors is investigating such an advanced set of templates for control sequences; he refers to it as a skill library [28].

C. Relations with Traditional Paradigms

We regard the APO paradigm as a combination of automatic programming and teleoperation. This subsection discusses the similarities and differences in the APO paradigm from these two paradigms.

1) Teleoperation: A complete APO paradigm can be regarded as a symbolic-level teleoperation. Traditional teleoperation systems connect the master and slave manipulator through the signal level link. In the APO paradigm, perceptual information (signal) is recognized and converted into symbolic level representations such as task models. Then, the APO paradigm maps task models to appropriate signal-level manipulator commands. APO connects human operations with manipulator operations through this signal-symbolic-signal link.

The complete APO system conceptually understands an operator's performance. Thus, the system can adjust motion errors by human operators and can generate consistent manipulator commands. It is also possible to clean up unnecessary motions introduced by a human operator and to generate efficient command sequences. Some of these capabilities are reported elsewhere [26].

2) Automatic Programming Paradigm: The APO paradigm has a strong relation to the automatic programming paradigm. For example, our TRM employs the face-contact relations that have been originally invented in the automatic programming community. Our grasp planner has similar attributes as some of the automatic programming systems.

Historically, the Edinburgh group led by Popplestone proposed RAPT as a language for describing assemblies [1], [22], [23]. In the RAPT project, they derived a method for inferring the positions of bodies from spatial relationships among them. Further, they extended the system so that it translates a collection of statements about bodies, relations,

and actions into a "tree of knowledge." These states and actions are the key components for solving such problems as in our system. On the other hand, their systems need to be given these relations and actions from descriptions through RAPT; our system discovers these relations, as well as motion parameters, from observation and then infers the necessary actions to cause such relation transitions.

Sanderson, Homen-de-Mello, and Zhang report a system that employs the face-contact relations as the basic representation, builds an AND/OR graph among parts, and, determines the assembly plans of a machine part [24]. Laugier and Willson report similar systems [5], [35]. He, Abe and Kitahashi proposed a method to derive assembly planning from assembly illustrations [7]. Wilson proposed a non-blocking graph to describe admissible directions of an object [34]. In comparison to those automatic programming systems, our task recognition module have two advantages.

When the number of parts increases, search trees in those automatic programming systems grow exponentially. On the other hand, since our task recognition system does not need a search, the system remains in the same order of execution complexity. Secondly, although these automatic programming systems can determine the assembly order of machine parts, these systems cannot provide configuration plans of parts for assembly and, as a result, cannot generate the complete robot motion commands. Our TRM collects motion parameters from observation and can generate the entire robot command sequences that need to be performed by the manipulator.

Several skeleton-based methods have been proposed in automatic synthesis of fine-motion strategies for robots. Taylor developed a technique for propagating the effect of errors and uncertainties through a model of a task [29]. These error estimates were used to make decisions for filling in the *strategy skeletons*. Lozano-Perez proposed a method for selecting the motion parameters in strategy skeletons [19]. Each motion in a skeleton was specified symbolically by the relationship among parts that it was designed to achieve. Our abstract task models correspond to their skeletons. Our selection method of an task model is done based on observation with the procedure graph. Our method to obtain motion parameters is also based on observation.

Lozano-Perez, Mason, and Taylor presented a formal approach to synthesizing a class of fine-motion strategies [18]. They computed the *pre-image* of a goal region, that is, a set of configurations that can reach the goal using a single, compliant motion using geometric descriptions of the task. Explicit bounds are placed on errors in sensing and motion. In the development of the system, they argued that small changes in the parts' geometry can have a significant impact on the fine-motion strategies. In the automatic programming paradigm, this may be a difficult problem to solve. Yet, in the APO paradigm, this is not the case. Depending on the geometry of parts, the human operator will change the fine-motion strategies. The necessary capability to the APO paradigm is to detect these strategy changes. Thus, theoretically, the APO system can select appropriate fine-motion strategies based on observation, provided that we have a skill library for fine-motion components. Some of the directions are being explored in [12].

If we compare the APO paradigm with the automatic programming paradigm, we can regard the APO paradigm as an automatic programming system with observation capability. By using this capability, the APO system can observe human operators and can obtain the crucial hints necessary for solving otherwise seemingly intractable problems.

ACKNOWLEDGMENT

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

This research was conducted in the Task-oriented Vision Laboratory, Carnegie Mellon University.

REFERENCES

[1] A. P. Ambler and R. J. Popplestone, "Inferring the positions of bodies from specified spatial relationships," *Artificial Intelligence*, vol. 6, no. 1, pp. 157–174, 1975.

[2] H. Asada and S. Hirai, "Towards a symbolic-level force feedback: Recognition of assembly process states," in *Proceedings of 5th Int. Symp. Robotics Res.* Cambridge, MA: MIT Press, 1990, pp. 341–346.

[3] P. Balakumar, J. C. Robert, R. Hoffman, K. Ikeuchi and T. Kanade, "Vantage: A frame-based geometric/sensor modeling system—programmer/user's manual v1.0," Technical Report CMU-RI-TR-91-31, Carnegie Mellon University, Robotics Institute, 1991.

[4] R. C. Brost, "Automatic grasp planning in the presence of uncertainty," *Int. J. Robotics Res.*, vol. 7, no. 1, pp. 3–17, 1988.

[5] C. Laugier, "Planning fine motion strategies by reasoning in the contact space," in *Proc. IEEE Intern. Conf. Robotics Automat.*, Scottsdale, AZ, 1989, pp. 653–659.

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

[7] S. He, N. Abe and T. Kitahashi, "Assembling plan generation from an assembly illustration," in *Proc. of Int. Conf. Intelligent Robots Syst.*, Raleigh, NC, July 1992, pp. 2122–2127.

[8] S. Hirai and T. Sato, "Motion understanding for world model management of telerobot," in *Proc. IEEE/RSJ Intern. Workshop on Intelligent Robots and Systems*, 1989, pp. 124–131.

[9] S. A. Hutchinson and A. C. Kak, "SPAR: a planner that satisfies operational and geometric goals in uncertain environments," *AI Magazine*, vol. 11, no. 1, pp. 31–61, 1990.

[10] 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*, vol. 53, no. 2, 1991. [A longer version, containing programs, is available as CMU-CS-88-188.]

[11] K. Ikeuchi and T. Kanade, "Towards automatic generation of object recognition program," *Proc. IEEE*, vol. 76, no. 8, pp. 1016–1035, 1988.

[12] S. B. Kang and K. Ikeuchi, "Determination of motion breakpoints in a task sequence from human hand motion," in *Proc. Int. Conf. Robotics Automat.*, San Diego, CA, 1994, pp. 551–556.

[13] S. B. Kang and K. Ikeuchi, "Grasp recognition using the contact web," in *Proc. IEEE/RSJ Intern. Conf. on Intelligent Robots and Syst*, Raleigh, NC, July 1992, pp. 194–201.

[14] 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. IEEE/RSJ Intern. Workshop on Intelligent Robots and Syst.*, August 1990, pp. 567–574.

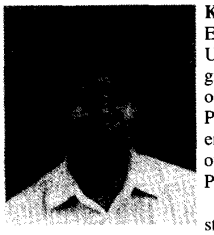
[15] L. I. Lieberman and M. A. Wesley, "AUTOPASS: an automatic programming system for computer controlled mechanical assembly," *IBM J. Res. Develop.*, vol. 21, no. 4, pp. 321–333, 1977.

[16] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst. Man Cyber.*, vol. 11, no. 10, pp. 681–689, 1981.

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

[18] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *The Int. J. Robotics Res.*, vol. 3, no. 1, pp. 3–24, 1984.

[19] T. Lozano-Perez and P. H. Winston, "Lama: A language for automatic mechanical assembly," in *Proc. of 5th Intern. Joint Conf. on Artificial Intelligence*, 1977, pp. 710–716.
[20] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *Intern. J. Robotics Res.*, vol. 5, no. 3, pp. 53–71, 1986.
[21] 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 Automat.*, 1989, pp. 56–61.
[22] R. J. Popplestone, A. P. Ambler, and I. Bellos, "Part, a language for describing assemblies," *Industrial Robot*, vol. 5, no. 3, pp. 131–137, 1978.
[23] R. J. Popplestone, A. P. Ambler, and I. Bellos, "An interpreter for a language for describing assemblies," *Artificial Intelligence*, vol. 14, no. 1, pp. 79–107, 1980.
[24] A. C. Sanderson, L. S. Homen de Mello, and H. Zhang, "Assembly sequence planning," *AI Mag.*, vol. 11, no. 1, pp. 62–81, 1990.
[25] N. O. Sliwa and R. W. Will, "A flexible telerobotic system for space operations," in *Proc. Space Telerobotics Workshop*, Pasadena, CA, 1987, pp. 285–292.
[26] T. Suehiro and K. Ikeuchi, "Towards an assembly plan from observation, part II: Correction of motion parameters based on face contact constraints," in *Proc. of IEEE Int. Conf. on Intelligent Robots and Syst.*, Raleigh, NC, July 1992.
[27] T. Suehiro and K. Takase, "Representation and control of motion in contact and its application to assembly tasks," in *Proc. of Int. Symp. of Robotics Res.*, Tokyo, Japan, August 1989, pp. 367–374.
[28] T. Suehiro and K. Takase, "Skill based manipulation system," *J. Robotics Society of Japan*, vol. 8, no. 5, pp. 47–58, 1990 (in Japanese).
[29] R. H. Taylor, "The synthesis of manipulator control programs from task-level specifications," Technical Report AIM-282, Stanford University, Artificial Intelligence Laboratory, Stanford, CA, 1976.
[30] R. Thibadeau, "Artificial perception of actions," *Cognitive Science*, vol. 10, no. 2, pp. 117–149, 1986.
[31] S. Tsuji, A. Morizono and S. Kuroda, "Understanding a simple cartoon film by a computer vision system," in *Proc. of 5th Int. Joint Conf. on Artificial Intell.*, pp. 609–610, 1977.
[32] M. D. Wheeler and K. Ikeuchi, "Towards a vision algorithm compiler for recognition of partially occluded 3-D objects," Technical Report CMU-CS-92-185, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, November 1992.
[33] D. E. Whitney, "State space models of remote manipulation tasks," in *Proc. of 1st Int. Conf. Artificial Intell.*, pp. 495–507, 1969.
[34] R. H. Wilson, "Assembling polyhedra with single translation," in *Proc. of Int. Conf. on Robotics Automat.*, Nice, France, May 1992, pp. 2392–2397.
[35] R. H. Wilson and T. Matsui, "Partioning an assembly for infinitesimal motions in translation and rotation," in *Proc. of Intern. Conf. Intell. Robots Syst.*, Raleigh, NC, July 1992, pp. 1311–1318.
[36] P. H. Winston, "The MIT robot," in *Machine Intelligence 7*, B. Meltzer and D. M. Michie, Eds. Edinburgh, U.K.: Edinburgh Univ. Press, 1972.
[37] P. H. Winston, "Learning structural descriptions from examples," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975.



Katsushi Ikeuchi (M'78–M'89) received his B. Eng. degree in mechanical engineering from Kyoto University, Kyoto, Japan in 1973, and a Ph.D. degree in information engineering from the University of Tokyo, Tokyo, Japan in 1978. Currently, he is a Principal Research Scientist in the Computer Science Department and the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

His research accomplishments in image understanding include the development of "smoothness constraints," that is, neighboring points of a surface that have similar surface orientations from which shape-from-shading and shape-from-texture can be iteratively recovered. He also pioneered the use of specular reflections to recover surface orientations. Instead of discarding specular reflections, he actively used them for recovering shape and reflectance.

In 1983, he developed a bin-picking system based on photometric stereo, a method to recover surface orientations by using structured lighting, and the extended Gaussian image, a spatial histogram of surface orientations. Dr. Ikeuchi's recent work has been on developing a vision algorithm compiler that automatically converts an object and sensor model into a vision program.

Dr. Ikeuchi has received several awards, including the David Marr Prize in Computational Vision and an IEEE outstanding paper award. In addition, in 1992, his paper, "Numerical shape from shading with occluding boundary," was selected as one of the most influential papers that had appeared in *Artificial Intelligence Journal* in the last ten years.

Takashi Suehiro received the B.E. degree, the M.E. degree and the D.E. degree in electronic engineering in 1978, 1980, and 1990, respectively, from the University of Tokyo. He joined the Electrotechnical Laboratory in 1980. He was a Visiting Research Scientist at the Robotics Institute, Carnegie Mellon University, from June 1990 to June 1991. Currently, he is on loan to the Real World Computing Partnership as the chief of the Active Inteligence Laboratory, Theory and Novel Functions Department. His research interests include skillfull motion of manipulators, fine motion planning, and man-robot cooperation.