

# Describing and Segmenting Scenes from Imperfect and Incomplete Data\*

KASHI RAO†

*Image Understanding, Corporate Research, Texas Instruments, MS238, P.O. Box 655474, Dallas, Texas 75265*

AND

RAM NEVATIA

*Institute for Robotics and Intelligent Systems, Powell Hall 204, MC-0273, University of Southern California, Los Angeles, California 90089*

Received January 14, 1992; accepted May 14, 1992

Usually shape description systems assume that a scene has been segmented into objects and that object boundaries are given. This, however, is not realistic when working with intensity images. Such images do not, in general, yield silhouettes corresponding to objects. The boundaries from such images are fragmented and contain surface markings and shadow and noise boundaries. The system we describe works with such input and computes shape descriptions of complex objects. Scene segmentation takes place through shape description. We use ribbons or 2D analogs of generalized cones for the basic shape representation. The key idea in our approach is the Finiteness Observation, which says that an object/object part is not infinite in extent, rather that it is terminated by a boundary or another object part. This observation is embodied in the Principle of Termination developed specifically for ribbons. We also develop the Same-sidedness and the Non-overlapping Observations to obtain parts of compound objects from imperfect and incomplete data. We exploit these observations and principle in the system that we develop. We show results for several synthetic and real intensity images of objects (simple and compound) with broken boundaries in the presence of shadows, reflectance boundaries, texture, and occlusion. The output of our system is useful for object recognition, learning new models, stereo and motion correspondence and navigation, inference of 3D orientation from a single 2D image, and grasping for robot hand-eye coordination. © 1993 Academic Press, Inc.

## 1. INTRODUCTION

Computing descriptions of shapes of objects in a scene is one of the most central problems in image understand-

\* This research was supported by the Defense Advanced Research Projects Agency under Contract F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

† This research was performed when the author was with the Institute for Robotics and Intelligent Systems, University of Southern California.

ing. Good shape description is needed for object recognition, of course, but also for other tasks requiring geometric reasoning of the scene, such as grasping and navigation.

Normally, shape description is assumed to be preceded by a *scene* segmentation process that outlines different objects in the scene; that is, the figure-ground resolution problem is assumed to be solved. Although this is possible given high quality, dense range data, it is usually not the case for realistic intensity images as the image boundaries are likely to be incomplete and imperfect, containing boundaries arising from markings, shadows, and noise, in addition to object boundaries (see Fig. 1, for example; detailed discussion on the figure follows). In such cases, the object shape itself is necessary in achieving good segmentation; that is, scene segmentation and shape description are interdependent. In the system we describe, we do not assume that we know the specific shapes expected to be perceived, but only certain generic classes of shapes as described below.

We have chosen a segmented, hierarchical representation for shape description. In this approach a scene is represented in terms of component objects; complex objects are represented by decomposing them into simpler parts and by describing the parts and the relations between them. This process can be applied hierarchically. Such representations can be rich and stable and represent occlusion and articulation in a natural way.

In this work, we have selected *generalized cones* [3] for the basic part representation. A generalized cone (GC) consists of an arbitrary planar shape, called a *cross section*, swept along an arbitrary 3D curve, called an *axis*. Further, the size and also the shape of the cross section may change along the axis; the rule describing the change is called the *cross section function*. We can also define 2D analogs of GCs, often called *ribbons*. For a

ribbon, the axis is an arbitrary 2D curve, the cross sections are simply line segments, and the cross section function defines how the cross section width changes along the axis; 2D ribbons may be viewed as projections of 3D GCs. Given only a single intensity image, it is easier to compute ribbons which may serve as a step toward inferring 3D volume descriptions.

GCs have been used for shape description in the past. Most of that work, however, assumes that the input consists of a perfect line drawing; that is, the scene has been segmented into objects and extremal object contours have been made explicit [19, 14]. Such input can be expected if dense range data is available; even then, the object surfaces may have to be prepared specially, such as being painted mat white for an indoor laser range finder. Use of intensity images invariably gives fragmented boundaries, and we must distinguish object contours from contours caused by shadows, surface markings and noise. Some typical scene boundaries are shown in Fig. 1 (ii), where (a) and (c) are from actual images and (b) is constructed synthetically. The ACRONYM system, discussed later, does deal with intensity image inputs and fragmented boundaries but it is highly model-directed [6].

In [25] we described a system called SHAPE I that computes GC descriptions from sparse and imperfect data. That system assumes that the input consists of broken boundary fragments, but the boundaries are given in 3D (as may be obtained from stereo, for example). That system is limited to the domain of the linear straight homogeneous generalized cones (LSHGCs, in Shafer's terminology [26]). This paper describes a much more general system called SHAPE II. Although we still assume the scene consists of objects well-described as GCs (or as ribbons in the image), the GCs now can be complex, i.e., they may have curved axes and the cross sections need not change linearly. Furthermore, the objects may be compound objects, that is, be composed of simpler GCs or other compound objects, as opposed to being just simple objects. Again our system has no *a priori* knowledge of the specific shapes it expects to see.

Figure 1 shows typical examples of this open problem that we wish to solve. The scenes are deceptively simple for humans but describing them requires several capabilities: the figure-ground problem needs to be resolved, the object boundaries need to be made explicit from other boundaries, and a shape description needs to be computed. These scenes are difficult to describe and segment because they are fragmented and there are *no silhouettes*. There are breaks in the boundaries of objects, there are markings, such as background and object texture and reflectance markings, and there are shadows. The breaks in the boundaries are *non-trivial*; that is, the boundaries interact closely with the markings and shadows. For exam-

ple, as the figures show (look at Fig. 1(ii)(b)—the two hammers case), the markings may be closer to a boundary fragment than is another boundary fragment. Thus simple-minded linking will not close the boundary and will not solve the figure-ground problem in such scenes. Besides fragmentation, the presence of compound objects with some complex generalized cone parts makes these scenes difficult. Occlusion accentuates these problems.

A shape description system such as the one described here is useful for object recognition [18]. It is also useful for learning new models and refining old ones. It can be used to match stereo images for depth or to match images in a motion sequence to obtain depth and motion parameters. Symmetry descriptions can be used to obtain 3D orientation of an object from a single view [12, 28]; 2D generalized cone descriptions can also be used to hypothesize 3D generalized cone descriptions, which in turn would not only make recognition and learning easier, but would also be useful for grasping [24].

The original contributions of this paper are the following:

- We work with incomplete and imperfect data and obtain shape descriptions. This is in contrast to previous researchers, as explained in Section 2 and illustrated in Figs. 1 and 2.
- We develop the Finiteness Observation, which states that an object/object part is not infinite in extent, rather that it is terminated by a boundary or another object part. This observation is embodied in the Principle of Termination, developed specifically for ribbons (Section 3.1).
- We develop a parts theory from imperfect and incomplete data unlike previous researchers who have done so with perfect closed boundaries. Our theory is based on the Samesidedness and Non-overlapping Observations that we have developed (Section 3.4).
- We develop and make explicit the following additional principles and properties for description and segmentation (Sections 3.2, 3.3, and 3.5): the Principles of Continuity, the Principle of Subsumption, and properties involving the axes, axial contour generators, and terminators of generalized cones and ribbons ("in-betweenness" and terminator-does-not-penetrate properties).
- We develop methods for hypothesizing ribbon descriptions from images (Section 4.1).
- We develop algorithms to verify the ribbon descriptions as being objects or parts of objects. The algorithms for verification are the major contributions of our method (Section 4.2).
- We have tested our method on several synthetic and real images (Section 5).

Section 4 presents some details of the method, although more details may be found in Chapter 5 of [22]. In

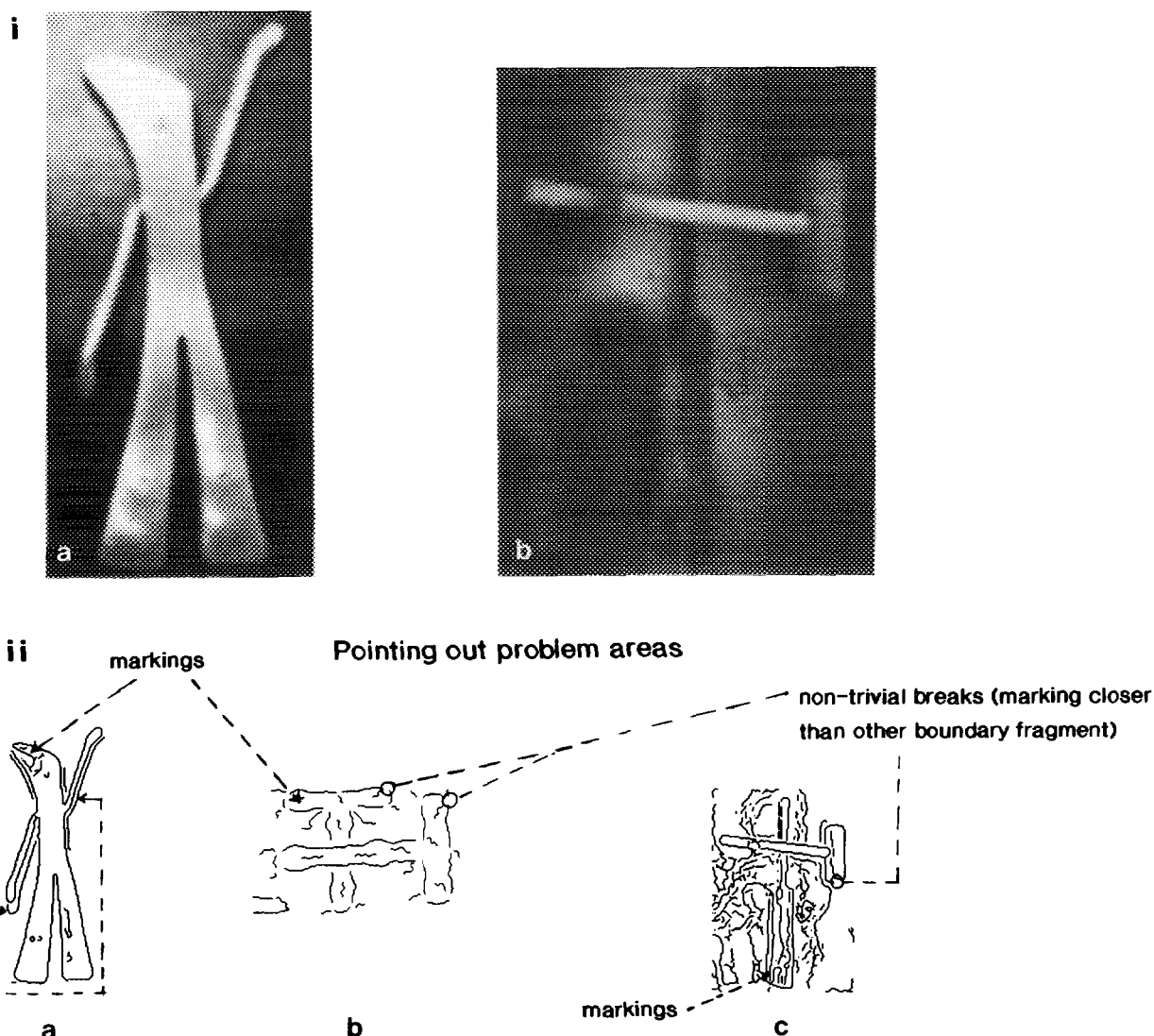


FIG. 1. Obtaining shape descriptions of compound objects in scenes like these with breaks in the boundaries, surface marks, and occlusion is an unsolved problem: (i) intensity images: (ii) edges.

Section 6 we conclude with possible applications of the method.

## 2. REVIEW OF RELATED RESEARCH

An early axis based method is the symmetric axis transform (SAT) [4]. This method needs closed boundaries with knowledge of figure-ground. Therefore, it would not work on the example scenes in Fig. 1. The 3D generalization of SAT [17] also suffers from the same problems.

The newer technique of smooth local symmetries (SLS) [5], like the SAT, also requires closed boundaries (or knowledge of figure-ground), to decide on corresponding points and to filter out undesirable axes. That

is, it needs the segmentation problem solved *before* description. Thus it, too, would not perform satisfactorily on the examples in Fig. 1 and would generate spurious axes. SLS also suffers from the problem of generalizing to a surface in 3D, rather than remaining a curve.

In [8] the authors describe a shape description system based on SLS. Their method requires knowledge of the inside and outside of objects in order to filter out axes. Thus the authors need figure-ground information, which is not available in Fig. 1. Also results shown are for closed or easily closable boundaries (trivial breaks). There are few surface markings, and the texture does not interact with object boundaries. Results are shown for single object scenes (and therefore there is no occlusion). Thus [8] will perform poorly on Fig. 1.

Another system that could be applicable to Fig. 1 is ACRONYM [6], a model-guided vision system capable of working with fragmented input data. It, however, requires detailed knowledge of the object being viewed in the form of model constraints. Some knowledge of the camera pose is also required. ACRONYM is primarily a predictive (top-down) vision system, and its principal weakness is its descriptive phase, which is our focus here. Its descriptive phase would produce several spurious ribbons on the examples shown. It would not be able to go further and filter out these undesirable ribbons because it is model-driven, and we do not have any specific models. Thus it would not be able to appropriately describe and segment the scenes in Fig. 1 and is also not adequate for the problem at hand.

Related research is that of parts theories [11, 13, 14, 19]. These methods, however, are for perfect, closed boundaries and will not apply to examples such as those in Fig. 1. The research on superquadrics and "geons" [20, 9, 1, 2] is also related to our work. Other related work is that of [21], which finds only straight (axis) homogeneous generalized cones (SHGCs) with nonparallel boundaries and without occlusion. The method is global and is based on fitting a straight line to the intersection of tangents to contours. The fitting is done using the Hough transform or using least squares error and has the standard problems of these techniques. As the scene may

have multiple objects or compound objects with parts, the method will need to know how many axes to look for. This is because the desirable axes may not appear as the only local maxima in Hough space. Also these axes may not even appear as local maxima in the presence of significant image clutter. As the method in [21] is global (as opposed to our method which is local), it will have problems with occlusion, and no results are shown on occluding objects. The method will also not work if the object contours are parallel because the tangents will not intersect then. Thus we believe that this method will also not work on our examples.

We have in this section shown how previous systems in computer vision would not perform satisfactorily for our task. In later sections we explain and demonstrate how our system successfully describes (and thus segments) scenes such as those in Fig. 1.

### 3. THEORY AND APPROACH

Two approaches to understanding images are summarized in Fig. 2. Traditionally, one first segments an image into closed boundaries and regions and then derives shape descriptions for the regions. In contrast, we obtain shape descriptions directly from the (fragmented) edges in the image. Scene segmentation, consisting of figure-ground separation and decomposing an object into parts,

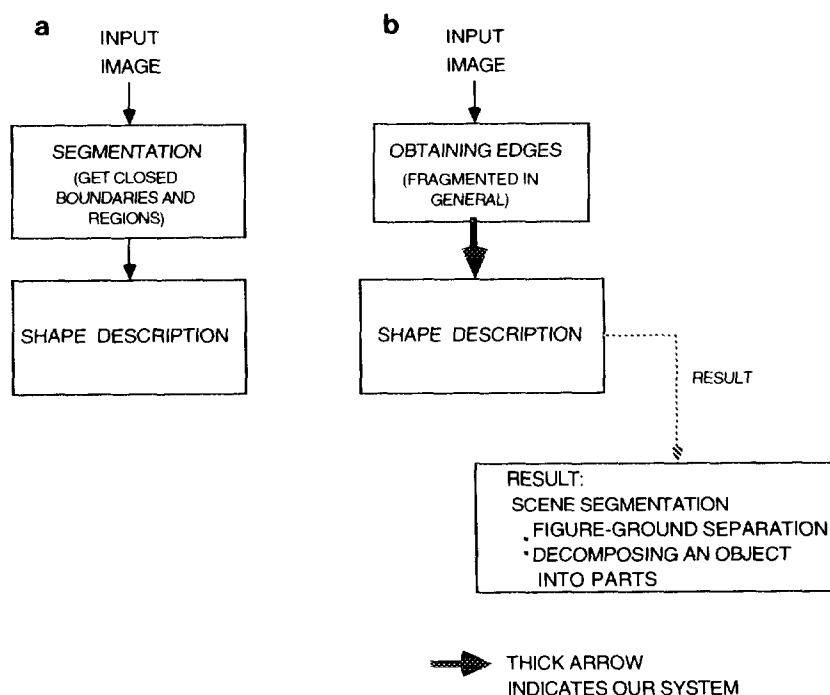


FIG. 2. Understanding images: two approaches: (a) Traditionally, researchers segment the input image into closed boundaries and regions and then describe shapes of regions. (b) In contrast, we describe shapes from the (fragmented) edges extracted. Scene segmentation, consisting of figure-ground separation and decomposing an object into parts, is a result of the more fundamental shape description process.

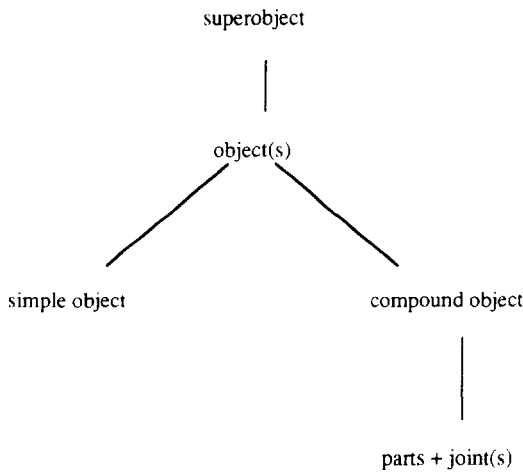


FIG. 3. Taxonomy of objects.

is not really a separate step but a result of shape description.

### 3.1. Key Ideas

In our representation, objects are of two kinds: simple and compound. A simple object is an object which cannot be decomposed into component parts. A compound object, on the other hand, is an object with parts. These parts are held together by one or more joints, which are essentially junctions of parts. Thus a compound object can be decomposed into its component parts and joint(s). See Fig. 3 for a taxonomy of objects and see Fig. 4 for examples of simple and compound objects. Our approach is based on the following *fundamental* observation:

*Finiteness Observation O1.* Objects/object parts are not infinite in extent, rather they are terminated either by a boundary or another object part.

This observation is embodied in the following specific principle:

*Principle of Termination, P1.* A ribbon represents the entire object, iff it is terminated at both extremities by terminator boundaries. A ribbon represents just a part, iff it is terminated at either (or both) extremity(ies) by another part; the extremity not terminated by another part should have a terminator boundary. (See Fig. 4 for an illustration.)

This rather obvious observation leads to a paradigm that we show is very powerful. Based on the observation we develop a theory that leads to specific ways of decomposing compound objects into component parts. The theory helps us construct algorithms to find these parts and therefore build appropriate descriptions from an image with imperfections.

At the top level, our method is simply that of hypothesizing and verifying (see Fig. 5). Hypothesis generation consists of finding groupings of boundaries that may define an object or parts of it. Verification consists of applying criteria based on the Principle of Termination to establish which hypotheses are viable.

### 3.2. Other Principles and Properties

We use some other principles and properties in different phases of the system. We use the following continuity principle in the hypothesis generation phase. The principle is justified because continuity seems to play an important role in grouping.

*Principle of Continuity of Axis Fragments, P2.* Continuous pieces of boundary give rise to continuous axis fragments.

The properties below are used in the verification phase, which is the principal component of the system. The properties are based on the following labeling of GC boundaries, shown in Fig. 6:

*Axial contour generators (ACGs).* Contour generators are the extremal points on the object surface and

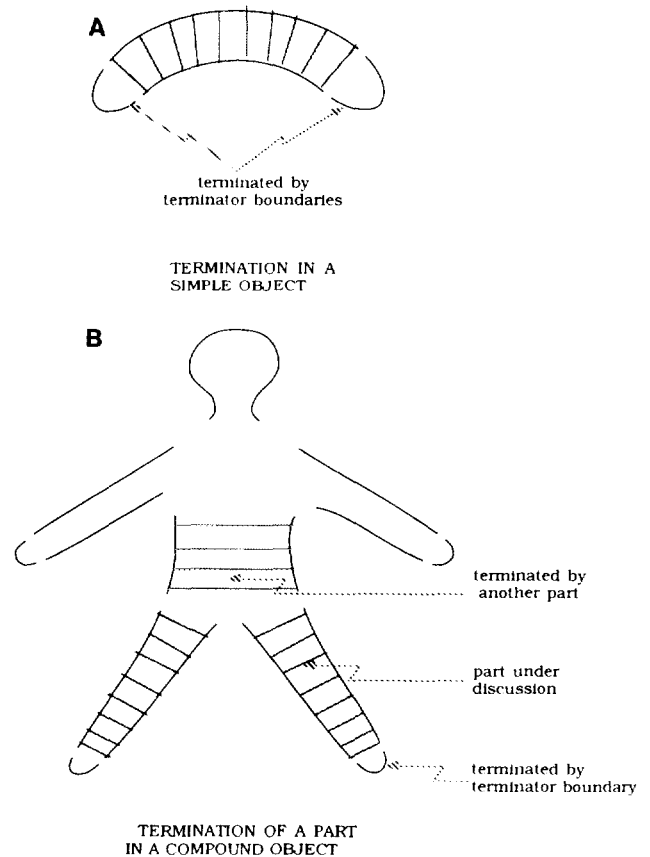


FIG. 4. The Principle of Termination.

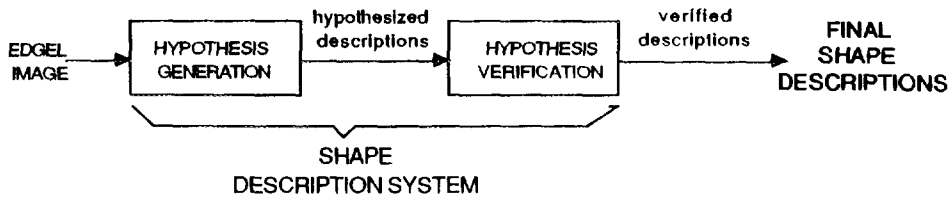


FIG. 5. Block diagram of the SHAPE II system.

enclose the visible surface (they are thus viewpoint dependent). For a smooth generalized cone, the contour generators are the extremal points on the surface, where the line of sight is tangential to the viewed surface. The axial contour generators are those portions of the contour generators that are along the length of the generalized cone.

*Terminators.* Terminators of a generalized cone are simply its ends (imagine an infinite generalized cone cut at some place across its axis).

The above definitions lead to the following properties obeyed by axial contour generators and terminators:

*“In-betweenness” Property, PT1.* The terminator boundary lies completely within the axial contour generators, extrapolated if necessary. This can be seen in Fig. 7.

*Terminator-does-not-penrate Property, PT2.* The terminator surface (boundary) does not penetrate the generalized cone volume in 3D (ribbon area in 2D). See Fig. 8.

We use the following important principle to form higher level groupings. Note that it is a higher level form of P2 and is similarly justified because of the role of continuity in grouping.

*Principle of Continuity of Ribbons and Objects, P3.* Ribbons are grouped together if they are continuous. Similarly objects are grouped together if they are continuous. We define continuity of two ribbons as continuity

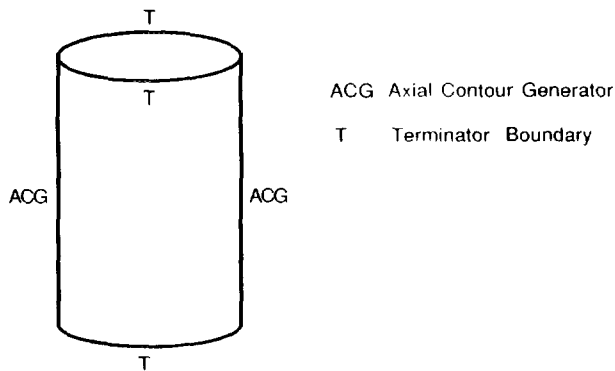


FIG. 6. A classification of generalized cone boundaries.

of their boundaries. Boundaries are continuous if their extremities are close-by and if the tangents at the extremities are continuous. Objects are continuous if the ribbons at their extremities are continuous.

### 3.3. 2D and 3D Descriptions

We now discuss in greater detail the relationship between 2D and 3D descriptions. In particular, we use the axis and the terminator to discuss the relationship.

#### 3.3.1. The Axis

In this work we have chosen right generalized cones or right ribbons for descriptions. A right generalized cone (opposite: oblique) has its cross section plane perpendicular to its axis. Similarly, a right ribbon (opposite: oblique) has its cross section straight line segment perpendicular to its axis. Right generalized cones and right ribbons are just as adequate as others for description and additionally have the following advantages: they are intu-

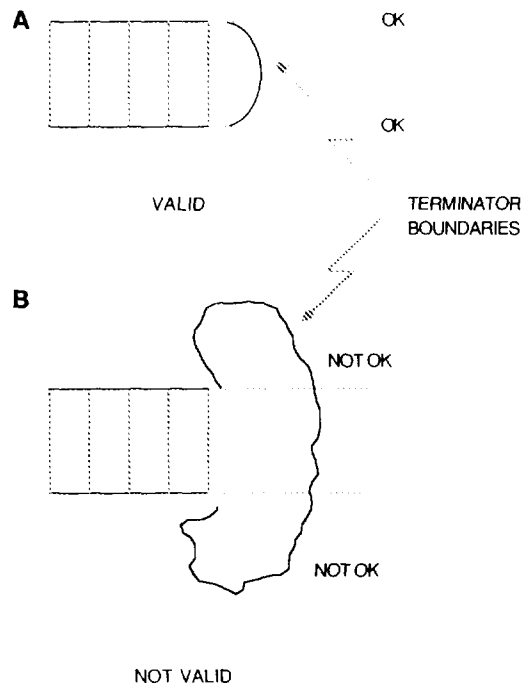
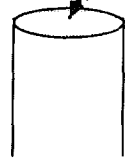
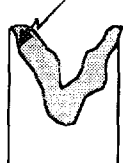
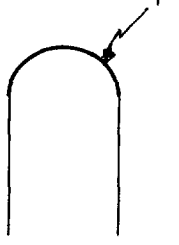



FIG. 7. “In-betweenness” property.

Dimension	valid	not valid
in 3-D		
in 2-D		

T: terminator

FIG. 8. Terminator-does-not-penetrate property: the terminator surface does not penetrate the generalized cone volume, and the terminator boundary does not penetrate the ribbon area.

itively appealing and result in simpler descriptions; they are easier to compute; and the axes of some right generalized cones, like solids of revolution, project to axes of right ribbons (see [23] for an analytical study).

Therefore we have chosen to compute axes of orthogonal symmetry (axes of right ribbons) instead of choosing other forms of axes of symmetry: for example, keeping the angle between the join of the boundary points and the axis nonconstant, or even if it is kept constant, not equal to  $90^\circ$ . A point of orthogonal symmetry,  $A$ , between two points,  $P$  and  $Q$ , is defined as the midpoint between  $P$  and  $Q$  such that the local tangent at  $A$  is perpendicular to the straight line segment  $PQ$ . See Fig. 9.

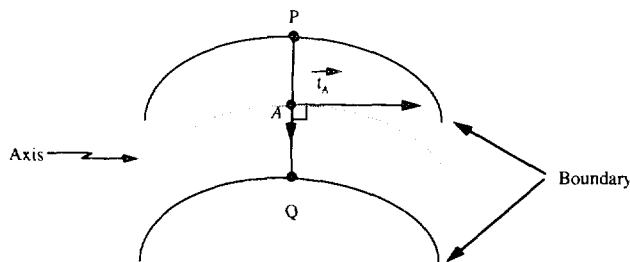


FIG. 9. Definition of the axis of orthogonal symmetry. For  $A$  to be a point of orthogonal symmetry between points  $P$  and  $Q$  we require the following:  $AP = AQ$  and  $PQ \cdot t_A = 0$ . Here  $t_A$  is the tangent to the axis at point  $A$ .

### 3.3.2. The Terminator

We have earlier discussed some properties of terminators and axial contour generators. Here we further discuss how the terminator can be used to relate a 2D description to a 3D description. Referring again to Fig. 8 (see figures in the first row), we can see it is the terminator that makes the object appear three-dimensional. If we can extract the appropriate ribbon from such a figure and associate it with the correct terminator, we have a good description for the object. In the process we should not confuse the terminator for another ribbon. The 2D description of the image (in terms of ribbons) corresponds to the appropriate 3D interpretation of the object because of the correct interpretation and use of the terminator. (Figure 31 in Section 5 shows an example.) Note that this does not mean the complete 3D description has been recovered; rather, it goes a long way toward recovering one. Note also that the terminator gives us some idea of the cross section of the generalized cone that the 2D ribbon represents. We were able to use this idea easily in our previous work [25] because we had 3D data.

### 3.4. Parts from Imperfect Data

We next state some observations in our approach to obtaining parts from imperfect data. We use these observations in the verification stage. Our objective is similar to that of parts theories [11, 13, 14, 19] referred to in Section 2. The difference, however, is that our approach is designed to also work with boundaries that are broken and with the input data having noise and surface markings.

*Same-sidedness Observation, O2.* Two ribbons connected by a boundary and lying on opposite sides of it cannot both be simultaneously in the figure or in the background: one of the ribbons has to be in the figure and the other in the background. That is, the boundary is an *occluding boundary* between the two ribbons. (This observation is true except for accidental alignments.)

Figure 10 explains this observation. Consider three ribbons  $R1$ ,  $R2$ , and  $R3$  in the picture. Let  $b_r$  be the *ribbon-inbetween boundary* joining  $R1$  and  $R2$ ;  $b_r$  is the list of edgels joining the extremity points of  $R1$  and  $R2$  as shown in the figure. For convenience let  $b_r$  also be the ribbon-inbetween boundary joining  $R1$  to  $R3$ .  $R1$  and  $R2$  lie on the same side of  $b_r$  and are, therefore, acceptable as being together in the figure or in the background.  $R1$  and  $R3$  lie on opposite sides of  $b_r$  and cannot both be accepted simultaneously: one of them has to be in the figure and the other in the background.

Joints are portions of an object where parts come together or become attached. In 2D a joint is thus an area or a region. To describe a compound object, we not only

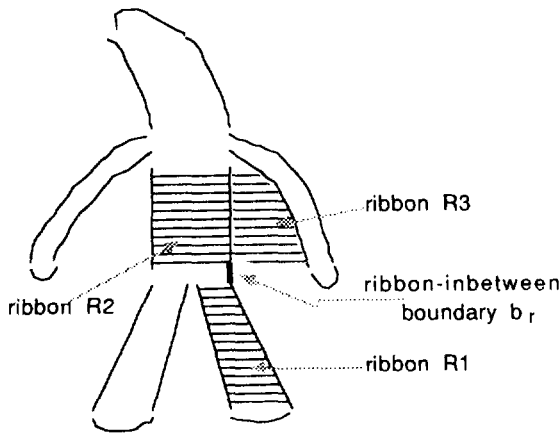


FIG. 10. The same-sidedness (of a curve) observation.

have to describe its parts, but also its joints because joints express an interrelationship between parts. We now state an observation on joints and parts called the Non-overlapping Observation (see Fig. 11).

*Non-overlapping Observation, O3.* 1. The joint formed by ribbons does not share the area with the composing ribbons. 2. Two ribbons participating in a joint do not share any area, they may share boundary edges though.

To explain this observation, we note that ribbon extremities are represented as nodes in a graph. (Section 4.2 presents details on this representation.) We define the *node-inbetween boundary*,  $B_n$ , between two nodes as the set of edges joining the ribbon extremities corresponding to those nodes. Let  $B(\cdot)$  be a function that returns the set of boundary edges of a ribbon; let  $E(\cdot)$  be a function that returns the set of extremity pixels of a ribbon; let  $A(\cdot)$  be a function that returns the set of pixels in the area of a ribbon; and let  $\emptyset$  denote the null set. Consider two rib-

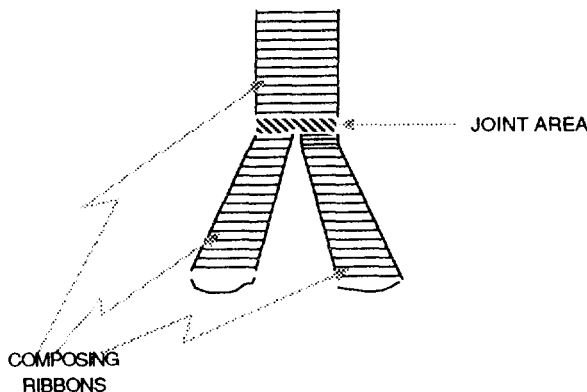


FIG. 11. The non-overlapping observation.

bons,  $R1$  and  $R2$ . The Non-overlapping Observation has the following details:

1. The node-inbetween boundary between two nodes of the two ribbons,  $R1$  and  $R2$ , should not share any edges with the boundaries of the ribbons, except at the extremities of the ribbons. That is,  $(B_n \cap (B(R1) \cup B(R2))) - E(R1) - E(R2) = \emptyset$ .

2. The node-inbetween boundary should *not* lie in the area of the two ribbons under consideration (except the ribbon extremities). That is,  $(B_n \cap (A(R1) \cup A(R2))) - E(R1) - E(R2) = \emptyset$ . Note that the first item is a special case of the second item.

3. The two ribbons should *not* share any area. (They may share boundary edges.) That is,  $(A(R1) \cap A(R2)) - (B(R1) \cup B(R2)) = \emptyset$ .

As the joint is a region in 2D, its boundary is cyclic or nearly cyclic. Thus the method for finding a joint is to find candidate parts and to see if they are connected with other parts by boundary fragments. (This rudimentary method can be improved by negotiating breaks in boundaries.) Finding joints becomes easier if we transform this problem into one of finding cycles in a graph formed by object parts. We discuss the method for this in hypothesis verification (Section 4.2).

### 3.5. Hierarchical Descriptions

As mentioned in Section 1, our approach is to obtain hierarchical descriptions of scenes. By hierarchical descriptions we mean descriptions at multiple levels of detail. We do not smooth the boundaries or subsample the image to obtain coarse boundaries and images the way researchers in multiresolution analysis do [27]. Instead we obtain higher level descriptions by using the following principle:

*Principle of Subsumption, P4.* Subsuming descriptions are higher in a hierarchy than subsumed descriptions with subsumption meaning containment. In 2D containment is by area, and in 3D containment is by volume.

Since we have 2D data we use subsumption by area. The above principle is justified because subsumption seems to play an important role in determining grouping hierarchy.

We utilize the Principle of Subsumption in various ways. We use subsumption of a ribbon (or superribbon), where that ribbon (superribbon) is contained in another ribbon (superribbon). We use subsumption of a cycle, where the ribbons and joint(s) of that cycle are contained in the ribbons and joint(s) of another cycle. Similarly, we use subsumption of objects (or superobjects), where that object (superobject) is contained in another object (or superobject). We exploit these ideas of subsumption to obtain a few higher level descriptions from a large num-

ber of lower level descriptions, while retaining substructural descriptions.

### 3.6. Discussion

In concluding this section, we observe that the broad approach in the SHAPE II system is similar to that in our earlier SHAPE I system for LSHGCs [25]. Our problem here in SHAPE II is more unconstrained because we are working with 2D data from scenes, and we have compound objects. Thus we must work with many more hypotheses and the verification step needs to do much more reasoning. This is in contrast to SHAPE I, where hypotheses were more constrained, and we were able to use the property that the contour generators of LSHGCs are linear segments and must be coplanar. We had 3D data available in SHAPE I, and this also made the verification step simpler.

## 4. THE METHOD

The method is basically that of hypothesizing and verifying (see Fig. 5). The input to the method is an edgel image: we link the edgels and use linked edgels (edge boundaries) in our method. Hypothesis generation then consists of producing possible ribbon descriptions from edge boundaries. Verification consists of verifying those descriptions using constraints from Section 3.

### 4.1. Hypothesis Generation

As mentioned above, in the hypothesis generation step we form ribbons from boundary fragments. Figure 12

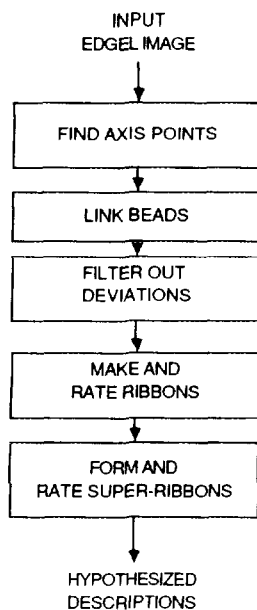


FIG. 12. Block diagram of the hypothesis generation phase of SHAPE II.

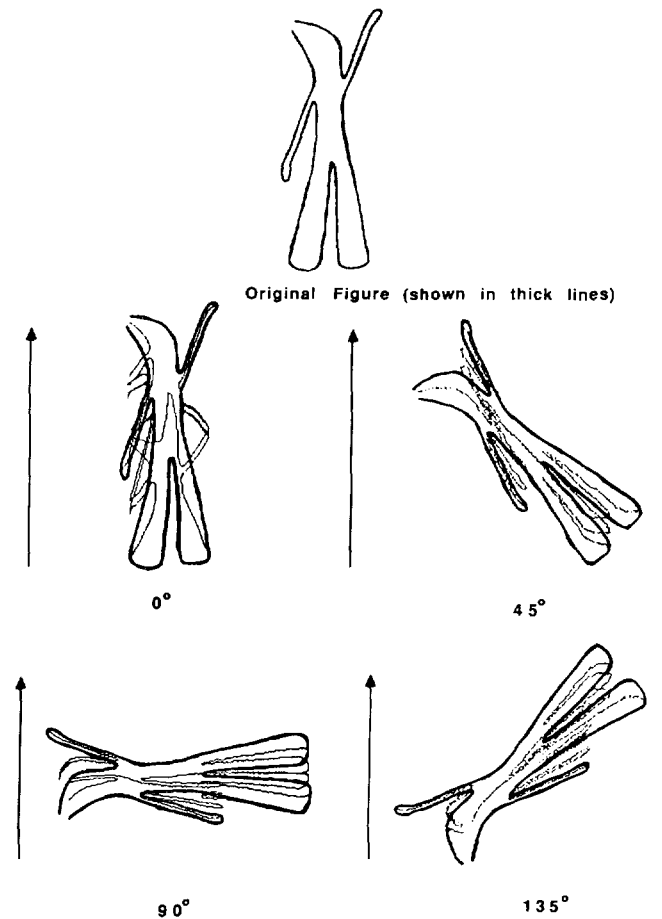


FIG. 13. Illustrating the axis finding procedure (the "dancing Gummy").

gives the block diagram of the hypothesis generation phase of the SHAPE II system.

We look for axes of orthogonal symmetry, which consist of linked points of orthogonal symmetry. To find such axes we first find midpoints between edges in different directions. To find axes in the horizontal direction, for example, we first draw vertical lines in the image at fixed intervals, say every  $s$  pixels (we take consecutive pixels, i.e.,  $s = 1$ , for accuracy). We then locate the edgels on each such vertical line. Axis points are placed midway between every pair of edgels in the line. Similarly, axis points are found in other directions. This is done by rotating the image, or merely the edgel list, and drawing the "vertical" lines again. We find axis points in  $m$  equally spaced directions,  $0 \leq \text{direction} < 180^\circ$ . (Note that we need not do  $0 \leq \text{direction} < 360^\circ$ .) This technique is basically the method of projections [19] and is illustrated in Fig. 13. The original figure is shown in thick lines; it is rotated in  $m = 4$  different directions,  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ , and axis points are found in each direction. The axis points are shown light; they appear to be con-

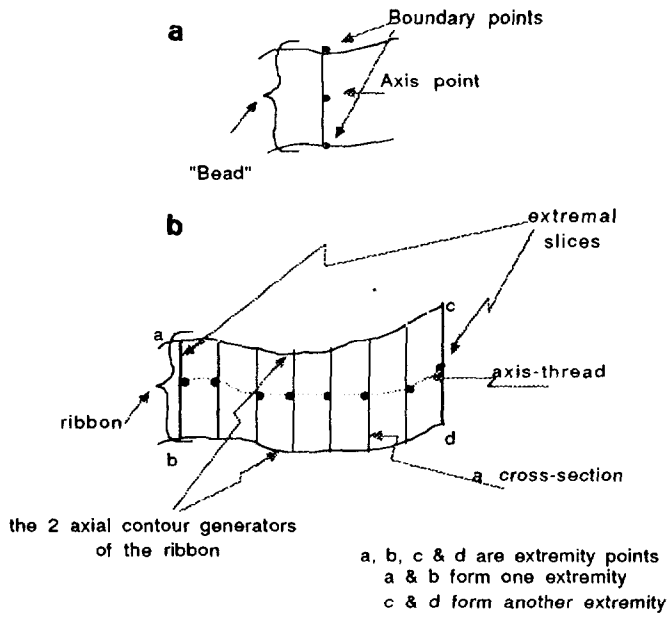


FIG. 14. The terminology used in axis finding.

nected but are actually not yet grouped into lines. Figure 14 explains some of the terminology used. Some of the terms illustrated in the figure are defined and used later.

By using  $m$  directions for axis finding we quantize the axis direction and use that approximation to compute the axis. We choose  $m = 4$  because we are working with digital imagery, and we are looking for axis points every pixel ( $s = 1$ ). When we take  $s = 1$ , we link with the neighboring axis point in a 3 by 3 neighborhood (discussed later). In such a neighborhood the natural quantization of the directions is  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ . This implies  $m = 4$ . In general, the maximum number of useful directions for axis finding is  $4s$ . We have taken  $m = 4$  for  $s = 1$ . For  $s = 1$ ,  $m = 2$  will also do well in most cases but gives a crude approximation if the axis of a ribbon is along the  $45^\circ$  or  $135^\circ$  lines. For  $s = 1$ ,  $m > 4$  is not any more useful than  $m = 4$  and is also computationally more expensive. Note that this kind of quantization in axis finding is similar to that in edge finding using directional edge detectors [18].

The data structure associated with an axis point consists of its  $x$  (column) and  $y$  (row) locations, its two defining edgels (called the *boundary points*), the distance between the edgels, the angle of rotation of the image (from which the axis point came), and the intermediate pixels between the boundary points, if necessary. The axis point with its associated data structure is called a *bead*.

For each direction, we order and group beads by linking them into ribbons (alternatively, axis points are linked into axis threads) by the algorithm explained below. The principle behind linking and neighbor checking (restated from P2 in Section 3.2) is the following:

*Principle for Linking Beads, P2.* Linked edgels in the image give rise to linked beads. The conditions for two beads,  $P1$  and  $P2$ , to be neighbors are (Fig. 15) the following:

1. The boundary points of the beads should be linked.
2. The beads should be ordered correctly. For example, if the boundary points of  $P1$  and  $P2$  are all linked, then the boundary points of  $P1$  should both succeed or precede the boundary points of  $P2$ . In other words, this specifies an *ordering* constraint on the beads. The next two conditions are special and important cases of this ordering constraint.
3. The cross sections of the beads should not cross. Two beads may cross because of interpolation errors due to rotation of edges in finding axis points. For the same reason they may share a boundary point; hence the condition below.
4. The two beads should not have a common boundary point.

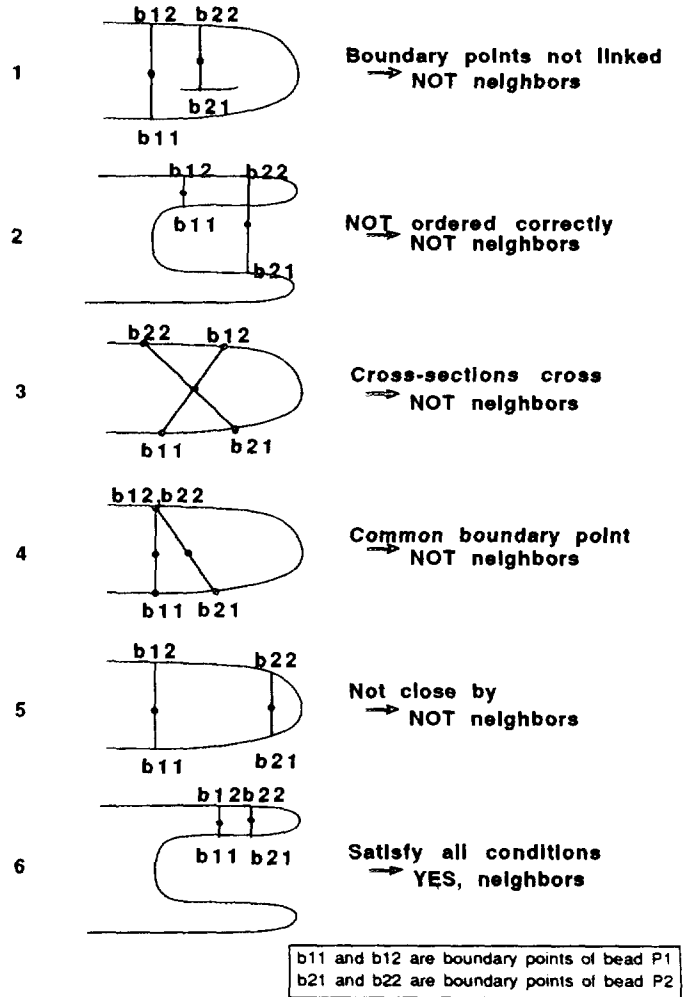


FIG. 15. Neighboring beads.

5. The two beads should be close-by. That is, the boundary points and the axis points of the beads should be within preset thresholds. Close-by-ness depends on the distance between successive lines used to find axis points.

Axis point of bead  $P1$  is close to axis point of bead  $P2$  iff it is within a  $q$  by  $q$  neighborhood of axis point of bead  $P2$ . We take  $q = 2s + 1$ , where  $s$  is the distance between successive lines to find axis points. Since we take  $s = 1$ ,  $q$  is equal to 3, and we search in a 3 by 3 neighborhood. The same method is used for boundary points.

Axis threads with less than five axis points are filtered out. This is because we need at least five points on the axis to compute tangent and curvature using the five-point tangent and curvature formulae explained next. If we use a larger threshold, we may filter out desirable axes; and if we use a smaller threshold, we cannot compute tangent and curvature anyway.

We next compute the tangent and curvature at the axis points of the axis threads. Tangents and curvatures are computed from five-point formulae using B-splines [16]. We skip the extremal two points: we start computing from the third point and stop at the third last point. If the axis thread has less than five points, tangents cannot be computed; but we anyway filter out the axis threads with less than five points immediately after linking.

The above procedure of linking gives us several axis threads, but not all of them are orthogonal to their cross sections. An axis point where the local tangent to the axis is not nearly perpendicular to the cross section it came from is filtered out. The maximum allowed angle,  $\tau$ , between a cross section and the normal to an axis is taken as  $180^\circ/2m$ , where  $m$  is the number of equally spaced directions, between 0 and  $180^\circ$ , for axis finding. For ex-

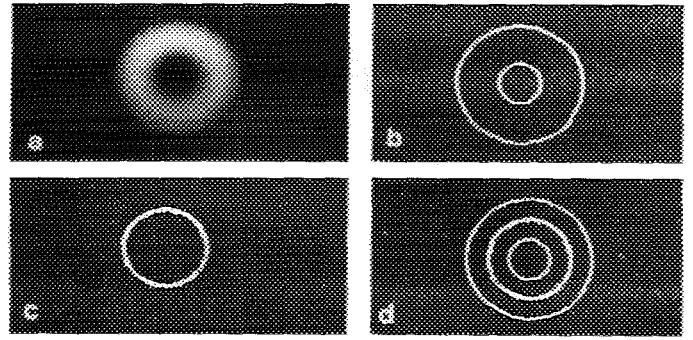


FIG. 17. Result on a real image of a torus: (a) input image; (b) edges extracted; (c) axis found; (d) superposition.

ample, if  $m = 4$ , the threshold =  $22.5^\circ$ . This filtering step is carried out on every axis thread from every direction in axis finding. It is illustrated in Fig. 16.

The justification for this threshold is as follows. Let the tolerance in the angle allowed be an unknown,  $\tau$ . This tolerance is in both the clockwise and counterclockwise sides of each direction scanned for finding axis points. Thus the total tolerance for each direction is  $2\tau$ . The total number of directions scanned is  $m$ , covering  $180^\circ$ . The total tolerance for all directions is  $2\tau m$ , since there is no overlap in the tolerance angle regions among the directions scanned. But the total tolerance should equal  $180^\circ$ . Therefore,  $2\tau m = 180^\circ$ , or  $\tau = 180^\circ/2m = 90^\circ/m = 90^\circ/4s = 22.5^\circ/s$ , which is the threshold used.

The threshold thus derived is desirable. If we have a smaller threshold, we will filter out a large number of axis points, thus breaking axis-threads into too many small pieces. If we have a larger threshold, we will retain undesirable axis-threads deviating a large amount from  $90^\circ$  from the cross section.

Axes and the regions they cover form ribbons. Ribbons are then joined to one another by the Principle of Continuity, P3 (explained in Section 3.2) to form larger ribbons or *superribbons*. Superribbons are rated according to length-to-width ratio and uniformity of width. Superribbons are the output of the hypothesis formation stage.

We next show examples of the output of the hypothesis generation phase. Figure 17 shows an output of the hypothesis generation phase for a real image of a torus. The torus is an interesting extreme case because it is continuously curving (the curvature of the axis is always non-zero), and the superribbon shown consists of some ribbons obtained from each direction of rotation.

As another example, refer to the result of hypothesis generation on the two hammers example. Figure 32(b) shows the large number of hypotheses (261 superribbons) generated, most of them nonintuitive. (For clarity, we show just the axes of the superribbons rather than their regions.) In the next section we discuss the verification stage, which chooses from this large number of hypotheses.

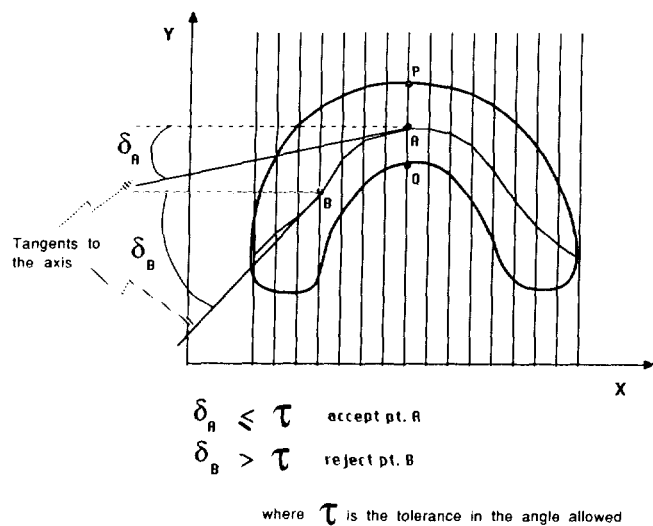


FIG. 16. Illustrating the filtering method.

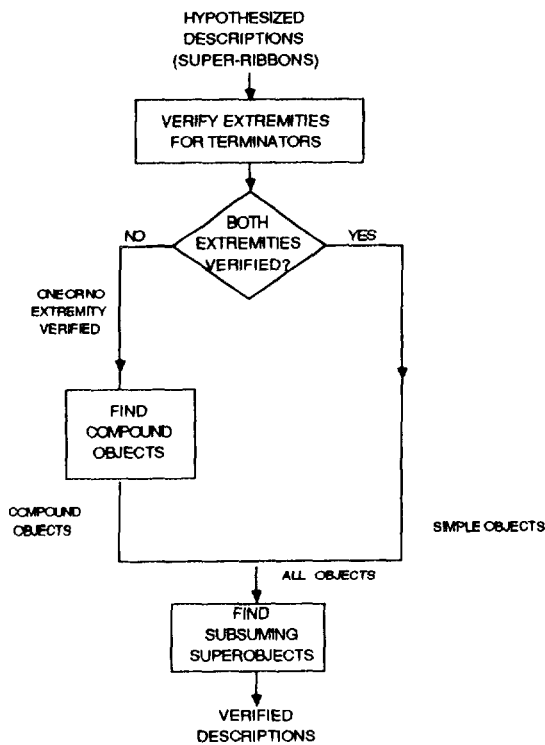


FIG. 18. Block diagram of the verification stage of SHAPE II.

#### 4.2. Hypothesis Verification

The hypothesis generation phase produces a rather large number of hypotheses, most of which do not correspond to objects we perceive. We now choose from these based on a verification phase. The underlying concept is that the hypothesis generation stage considers only boundary fragments that define axes. If a ribbon, however, does correspond to an object or a part, it must have terminator boundaries or attach to another part, by the Principle of Termination, P1. Figure 18 gives the block diagram of the verification stage.

As the figure shows, the verification module proceeds by first checking extremities of all ribbons for terminator boundaries. If a ribbon is terminated<sup>1</sup> at both extremities, it is completely verified and declared an object. It is a simple object, with no parts, as opposed to a compound object.

A ribbon is verified at an extremity if edges can be found to traverse from one side of the ribbon extremity to the other. Figure 19 shows an example of the terminator boundary tracing process. We traverse from one side to the other negotiating gaps using a backtracking algorithm. The terminator should satisfy the "in-between-

<sup>1</sup> The word terminated will now on be used to mean terminated by a boundary, unless otherwise stated.

ness" property, PT1 (Section 3): it should lie completely within the axial contour generators. The terminator should also satisfy the *terminator-does-not-penetrate* property, PT2 (Section 3): it should lie outside the extremal slices of the ribbons (within a tolerance). If we find several possible terminator boundaries, we pick the best one (the shortest, the smoothest, and the one with the least number of gaps).

For the rest of the ribbons we check if they form parts of compound objects. Figure 20 gives the block diagram for this step. We first check if these ribbons possibly form joints with one another. We then explore if the joints could be grouped to describe a compound object verified at all its extremities. This method of grouping ribbons into joints is performed by first forming a graph of ribbons called the *scene graph*. The objective is to represent the scene so that joints of objects are cycles in the graph. To do this, we make each extremity point of each side of a ribbon a node in the scene graph. Thus each extremity is represented by two nodes and each ribbon by two pairs of nodes. The two pairs of nodes for a ribbon form a supernode and may be considered a single entity. We form nodes for all ribbons found in the scene. An *eligible node* is a node whose corresponding ribbon extremity has not been verified by the prior terminator boundary finding process.

Our next objective is to form links between nodes in the scene graph. There are three types of links, classified by the way one can traverse from one node to another:

*Same-ribbon same-extremity (SRSE)* links between nodes at the same extremity of the *same* ribbon;

*Same-ribbon same-side (SRSS)* links between nodes at opposite extremities of the *same* ribbon but the same side of the ribbon; and

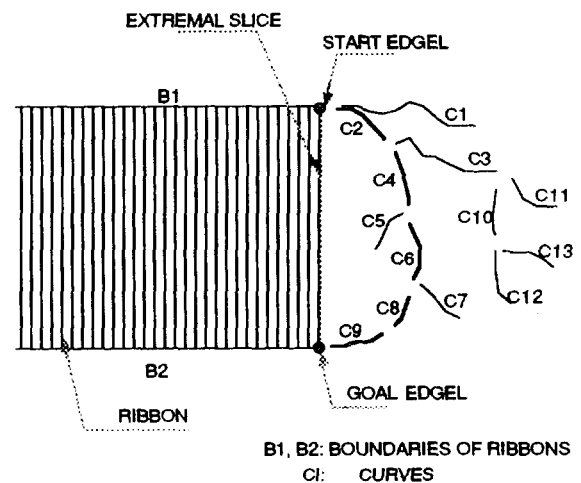


FIG. 19. Terminator boundary tracing using depth first search.

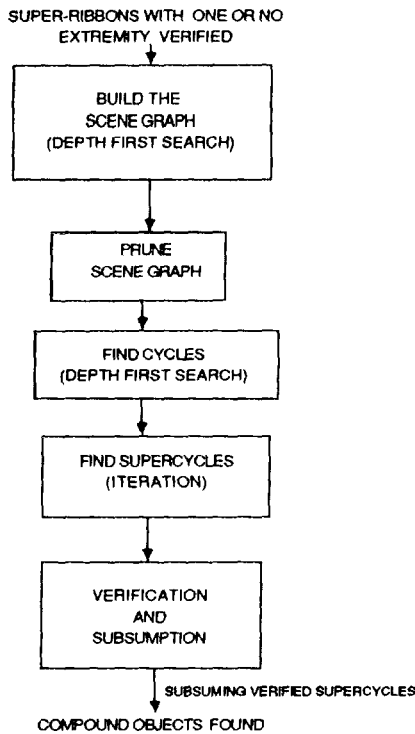


FIG. 20. Block diagram for finding compound objects in the verification stage.

*Between-ribbon (BR)* links between nodes of *different* ribbons.

See Figs. 21 (a) and (b) for some simple ribbons, their corresponding nodes, and SRSE and SRSS links.

Among the three links defined above, BR links are the difficult ones to obtain, and they play a crucial role in forming joints and compound objects. For verification purposes we need to form these links among eligible nodes only. Links from each node are obtained by traversing from it to all the other nearby eligible nodes. This traversal is like the linking problem for finding terminator boundaries, except that we now have several possible goal nodes (the other eligible nodes in the scene graph). We do this again by a depth-first search algorithm to negotiate breaks in the boundaries and stray surface marks.

As we traverse along a boundary, we do *not* establish links with all ribbons on the path. Only some of these ribbons are acceptable. The acceptability criteria are based on the Same-sidedness and Non-overlapping Observations, O2 and O3, stated in Section 3.4.

Next, we explain the formation of the remainder of the scene graph using between-ribbon links. Consider a node, called *current node*, representing an extremity of a ribbon called the *current ribbon*. Consider another node called the *other node*, representing an extremity of another ribbon, called the *other ribbon*. For the other node to form a link with the current node, the other ribbon and the current ribbon should satisfy the constraints set by the above observations. Figure 22 shows a schematic example with shadows, markings, and breaks in the boundaries; for simplicity we consider only some ribbons in the image that are due to object parts, shadows, and markings. The goal is to retain ribbons that are good candidates for object parts and to filter out those due to shadows and marks. For this we first form the scene

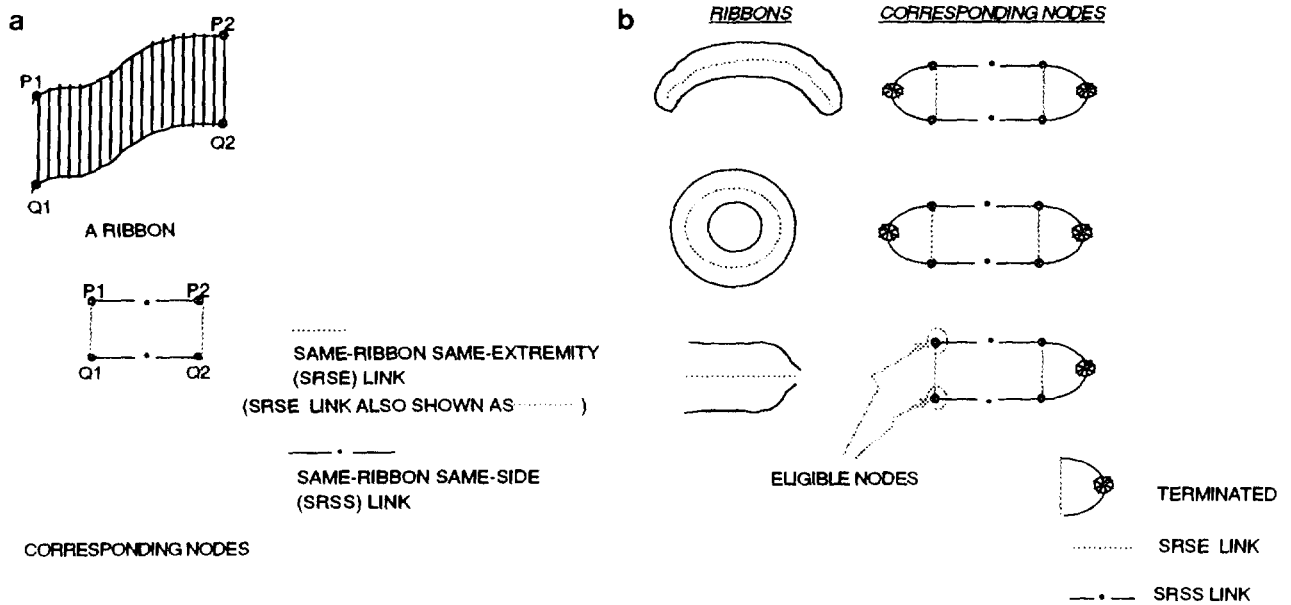


FIG. 21. The nodes for a ribbon in the scene graph.

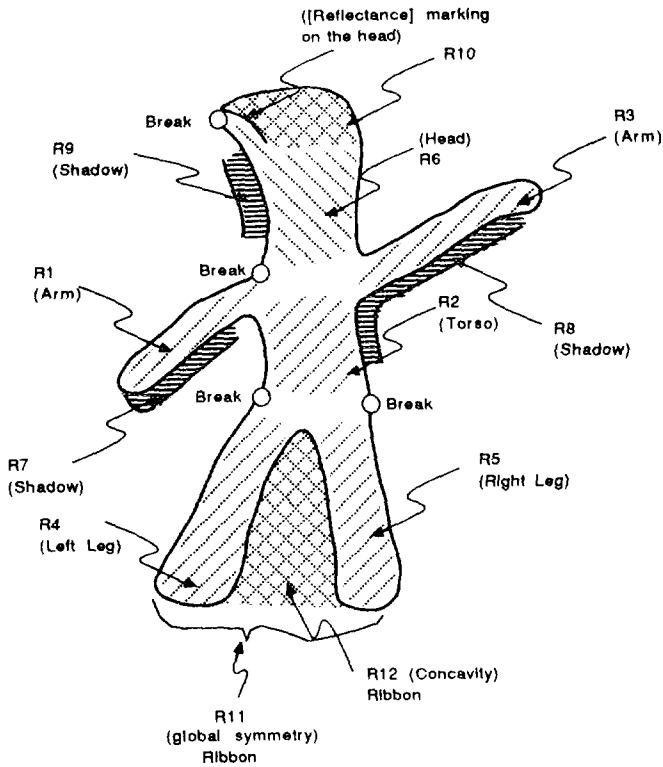


FIG. 22. A schematic example of a compound object (Gumby) with shadows, markings and breaks in the boundaries.

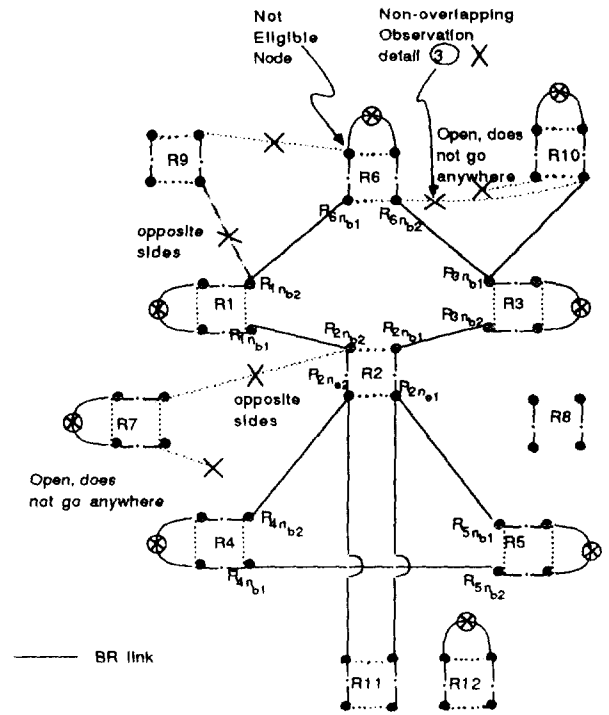


FIG. 23. The graph for the Gumby example. Some of the arcs are shown light with cross-marks and a note explaining why those arcs are not valid.

graph, Fig. 23. Some of the arcs are shown light with cross marks and a note explaining why they are not valid.

The scene graph is pruned so that the search space becomes smaller when finding cycles. For this, we keep only those nodes corresponding to ribbons that have at each extremity either a terminator boundary or connections to other ribbons on each side of that extremity. The other nodes are pruned and links involving them are removed from the graph. Figure 24 shows the pruned graph. Ribbons R7, R8, R9, R10, R11, and R12 have extremities without terminator boundaries and without connections to other ribbons. They are pruned and arcs connecting them to the remaining ribbons are removed.

Our next objective is to find cycles in the scene graph because they correspond to joints in objects. We form cycles by depth-first traversal of the pruned graph. Nodes are expanded using same-ribbon same-extremity and between-ribbon links (and *not* same-ribbon same-side links). By forming cycles we find joints in objects. Forming joints may be considered a first level of grouping ribbons. Figure 25 shows cycles found for the example scene and ribbons corresponding to those cycles. For each cycle we check if its ribbons can be joined by the Principle of Continuity, P3. Ribbons are merged by linear interpolation of axes and boundaries, followed by smoothing.

From cycles we form supercycles, which are combinations of cycles. A supercycle corresponds to a group of joints in an object. (A compound object may be considered a group of joints with *outer ribbons* in the group verified at their *outer extremities*. These terms are defined later.) We thus form supercycles and take the grouping of ribbons one step further. Supercycle forming may be understood as propagating the verification check

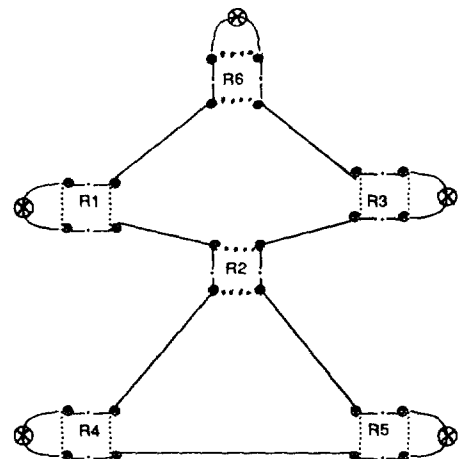


FIG. 24. The pruned graph for the Gumby example.

of an object from one extremity to another. Supercycles are formed by combining cycles in the graph using same-ribbon same-side links. That is, we combine two cycles linked by a ribbon (with the opposite extremities of the ribbon being in the two cycles).

The algorithm for forming supercycles is the following:

1. *Initial condition.* Initially all the cycles formed are supercycles.
2. *Merging.*
  - (a) For each of these supercycles check if it can be merged with any other supercycle.
  - (b) If so form a new supercycle.
  - (c) Keep a list of new supercycles and the old ones that have not been merged.
  - (d) Do this merging in each iteration and keep a marker indicating whether new supercycles have been formed.
  - (e) Stop when there is no change between successive iterations.

This algorithm returns a list of all supercycles found.

For each supercycle returned, we check if the outer extremities of its outer ribbons (ribbons away from joints, defined more precisely in the next paragraph) are verified with a terminator boundary. If so the supercycle is verified. We throw away the supercycles not verified, keeping only the verified ones. Of the verified supercycles we

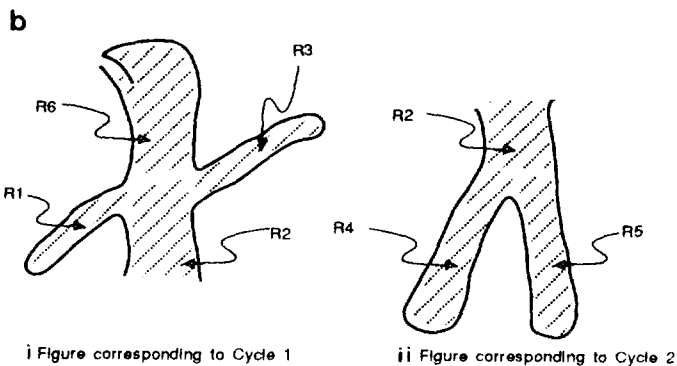
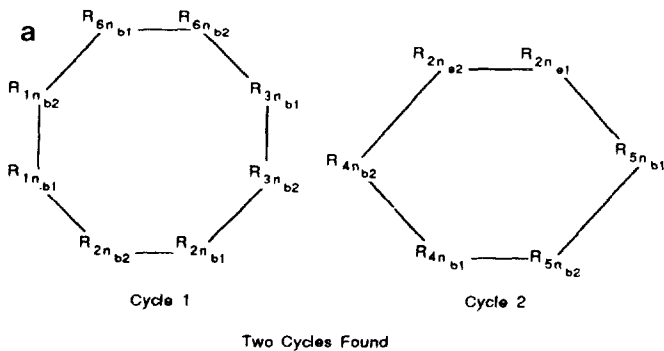


FIG. 25. Cycles found for the Gumby example. Also shown are the ribbons corresponding to the cycles.

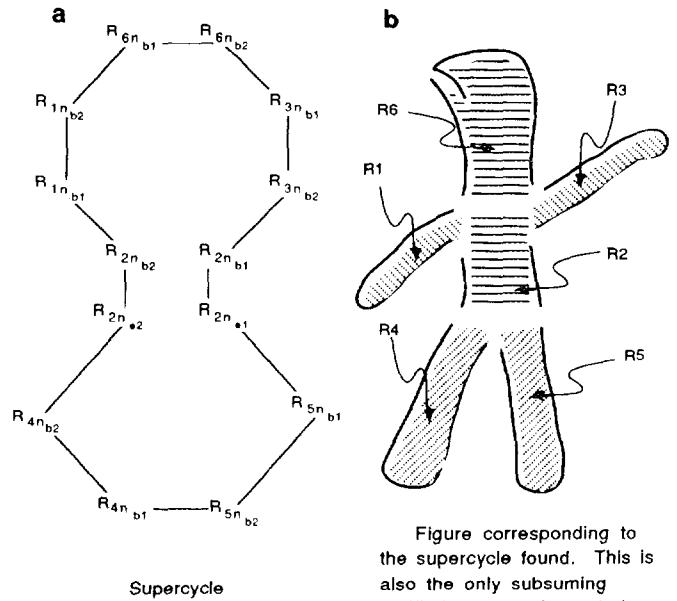


Figure corresponding to the supercycle found. This is also the only subsuming verified supercycle and the only superobject

FIG. 26. The supercycle found for the Gumby example. Also shown are the ribbons corresponding to the supercycle.

take only those subsuming others in area<sup>2</sup> by the Principle of Subsumption, P4. These are the subsuming verified supercycles. The *subsumed* verified supercycles may be understood as *substructures* of the *subsuming* verified supercycles. Note that the system has the desirable capability of describing scenes at different levels; that is, the system is capable of substructure and superstructure descriptions. These are thus segmented, hierarchical descriptions, as desired in Section 3.

Figure 26 illustrates the supercycle formation process. It shows a supercycle found in the scene graph of Fig. 22 using same-ribbon same-side links to merge the cycles in Fig. 25. For clarity, we introduce some general terminology for supercycles. An *inner ribbon* of a supercycle is a ribbon that has all its nodes connected to other ribbons in the supercycle and none of whose extremities is verified by a terminator boundary. In this example, R2 is an inner ribbon. The ribbons away from the inner ribbons are *outer ribbons*, and their extremities away from the inner ribbons are *outer extremities*. Here ribbons R1, R3, R6, R4, and R5 are outer ribbons. The outer extremities need to be verified for the supercycle to be verified. They are indeed verified for this supercycle.

We next find and merge objects. Figure 27 presents the block diagram for this stage. The objects found are the subsuming verified supercycles (compound objects) and

<sup>2</sup> Cycle  $C_1$  subsumes cycle  $C_2$  iff the area of  $C_1$  subsumes the area of  $C_2$ . Area of a cycle is the union of the area of its ribbons and its joint. The test and definition are similar for supercycles.

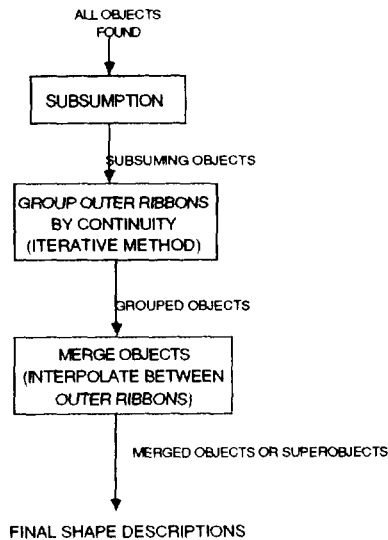


FIG. 27. Block diagram for finding subsuming superobjects in the verification stage.

ribbons verified earlier by terminator boundaries at both extremities (simple objects). From these we filter out objects subsumed by others in area<sup>3</sup> again by the Principle of Subsumption, P4. Note that we again have substructure descriptions. Subsuming objects are then grouped by continuity of outer ribbons<sup>4</sup> by the Principle of Continuity, P3. Objects are then grouped iteratively by a process similar to that of finding supercycles. Grouped objects may have missing portions, possibly due to occlusion, and these portions are hypothesized by linearly interpolating and smoothing the ribbons to be joined. This process of grouping objects and merging them across occlusion is similar to Guzman's matched T's heuristic [10]. The merged objects or *superobjects* are the final object level descriptions of the scene.

Figure 32 explains this process for the two hammers example. Figure 32 (a) shows the original scene. Figure 32 (d) shows the objects found, which are also the subsuming objects found; b1 and b2 are the compound objects obtained by joint finding, whereas b3 is a simple object. Figure 32 (e) shows these objects grouped by continuity of the outer ribbons. Note that objects b1 and b3 have been grouped together as object c1. Finally ribbons in object c1 found continuous are interpolated, and we obtain object d1. The final merged objects or superobjects are shown in Fig. 32 (f).

## 5. RESULTS AND DISCUSSION

Using the above ideas, we have implemented SHAPE II in Common Lisp on a Symbolics 3600 series machine

<sup>3</sup> The test for subsumption of objects is the same as that for cycles.

<sup>4</sup> The outer ribbon of a simple object is the single ribbon itself describing the object.

under Genera 7.2. SHAPE II has been tested on a number of scenes with good results. We show some examples here, more are reported in Chapter 5 of [22]. For each example we show the input edge data and the final descriptions obtained by our system. Each ribbon is displayed by its axis, the boundaries along the axis, the extremal cross sections, and the terminator boundary at each extremity, when found. In all examples we have breaks in the boundaries and markings on the object and background. In some cases the marks on the background are shadows. The breaks in the boundaries are nontrivial; i.e., mere linking of edges does not generate closed boundaries, due to markings being closer to boundary fragments. For the real data, we show original intensity images and their Canny edges [7].

The first example (Fig. 28, synthetic data, a plane) shows a result on a compound object. Here a supercycle is formed by combining two cycles. Part (a) shows the input edge data of the plane, (b) shows all the axes generated by the hypothesis generation phase. Only those ribbons that are terminated (either by boundary or by a part) are verified, and these are shown in part (c). Part (d) shows the ribbons after interpolation.

Figures 29 (synthetic data, ship) and 30 (real data, Gumby) show cases where the supercycles are formed by combining two cycles and then combining with a third. These examples thus show we can propagate the verification of extremities one step further. Note that the Gumby example has shadows and reflectance markings. Yet our

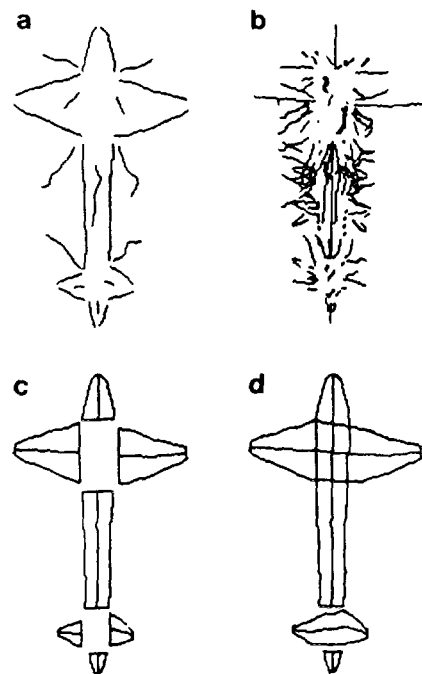


FIG. 28. (a) Input edge data for plane; (b) all hypotheses generated; (c) supercycle found (grouping of two cycles); (d) final result (superobject).

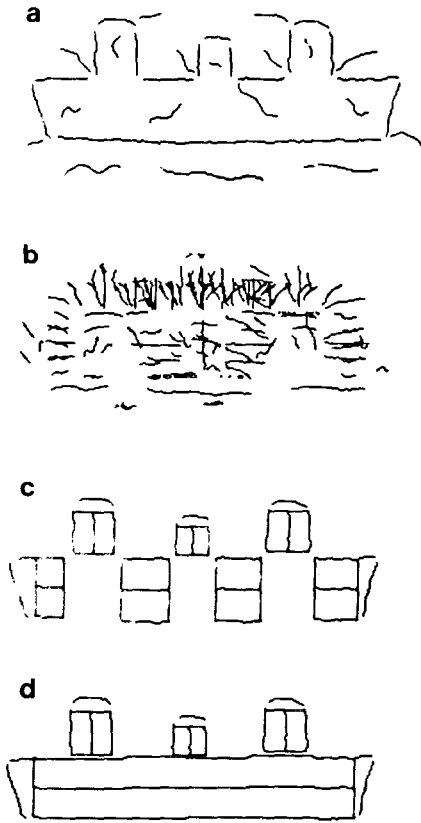


FIG. 29. (a) Input edge data for ship; (b) all hypotheses generated; (c) supercycle found (grouping of three cycles); (d) final result (superobject).

system is capable of describing such a scene appropriately.

Figure 31 shows an object with both the front and back terminator boundaries being seen and with flutings too. (A fluting is a tracing where the cross section attains an extremum or where the tangent to the cross section has a discontinuity. A tracing is a space curve formed by a point on the cross section contour as the cross section is drawn along the axis. An example of a fluting is a line of surface orientation discontinuity along the axis of the object.) Our system does generate ribbon descriptions due to the front and back terminator boundaries and due to the flutings, but these descriptions become subsumed by the description of the entire object.

Figures 32 (synthetic data) and 33 (real data) demonstrate the capability of the system in describing compound objects with *occlusion*. We display all the axes hypothesized and then all the cycles found. These cycles are also the supercycles; there is no propagation of verification here. The supercycles (compound objects) and simple objects are then grouped by continuity; continuous objects are merged by interpolation across occlusion and are shown in the figures as the final result. These

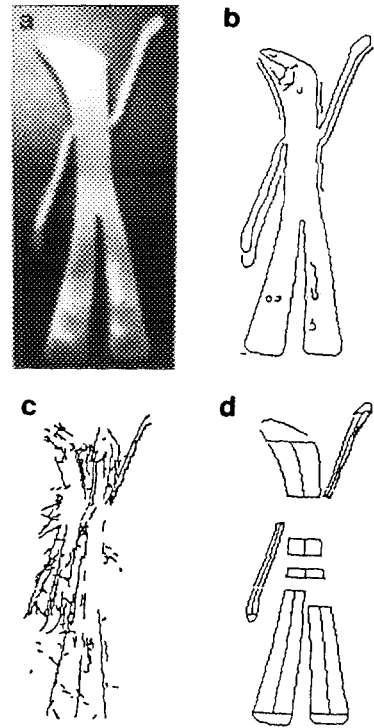


FIG. 30. (a) Original image for Gumby; (b) input edge data; (c) all hypotheses generated; (d) final result (supercycle found—grouping of three cycles).

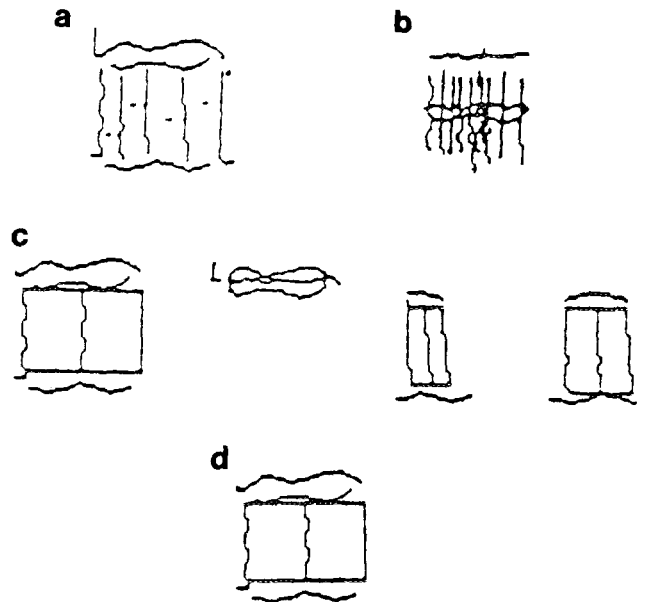


FIG. 31. (a) Input edge data for object with flutings and terminators (in the front and the back); (b) all hypotheses generated; (c) some intermediate descriptions; (d) final result.

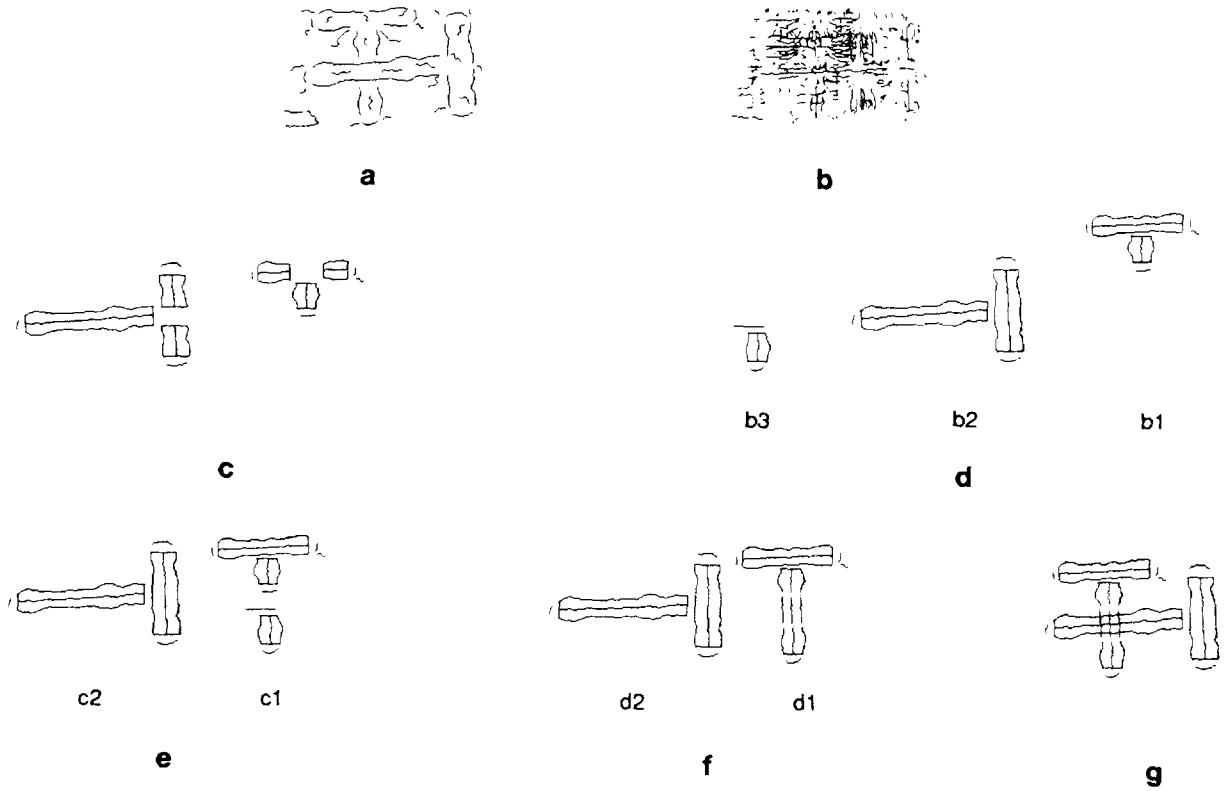


FIG. 32. (a) Input edge data for two hammers with occlusion; (b) all hypotheses generated; (c) supercycles found (just one cycle each); (d) all objects found; (e) grouped objects; (f) superobjects; (g) final result (superposition of superobjects found).

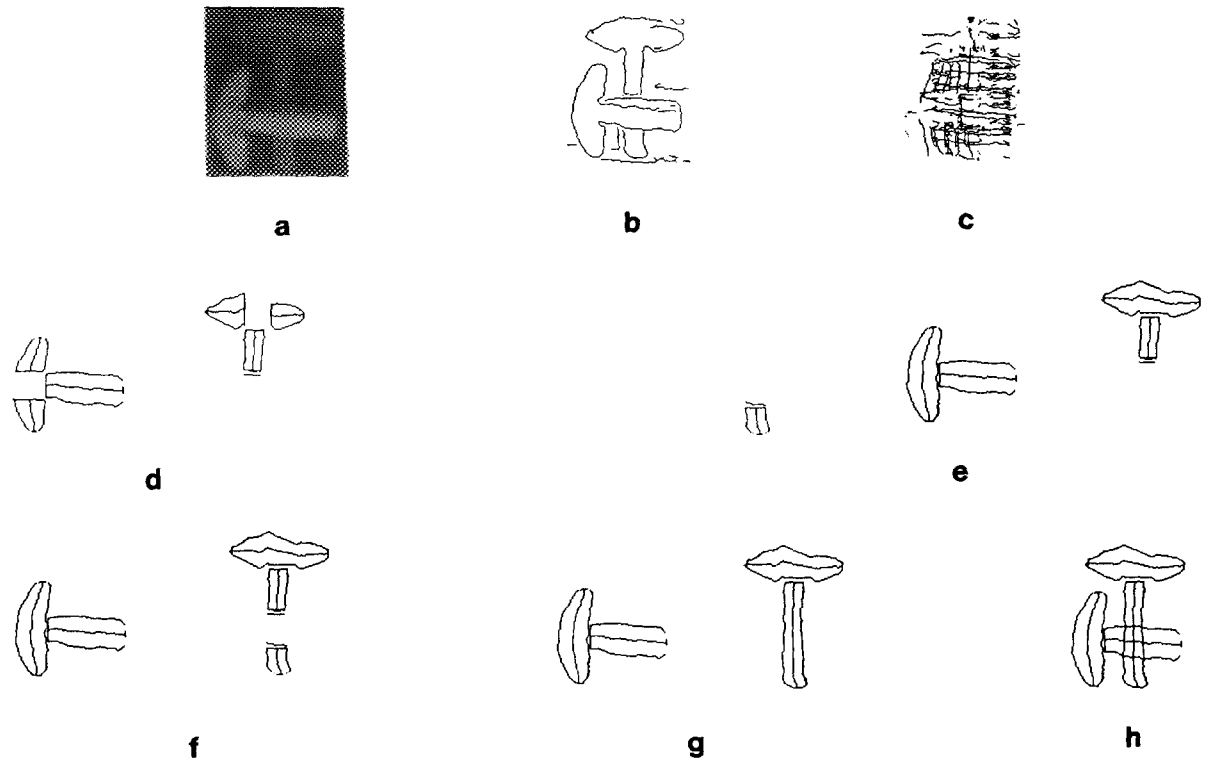


FIG. 33. (a) Original image of two hammer-like (mushroom-like) objects with occlusion; (b) input edge data; (c) all hypotheses generated; (d) supercycles found (just one cycle each); (e) all objects found; (f) grouped objects; (g) superobjects; (h) final result (superposition of superobjects found).

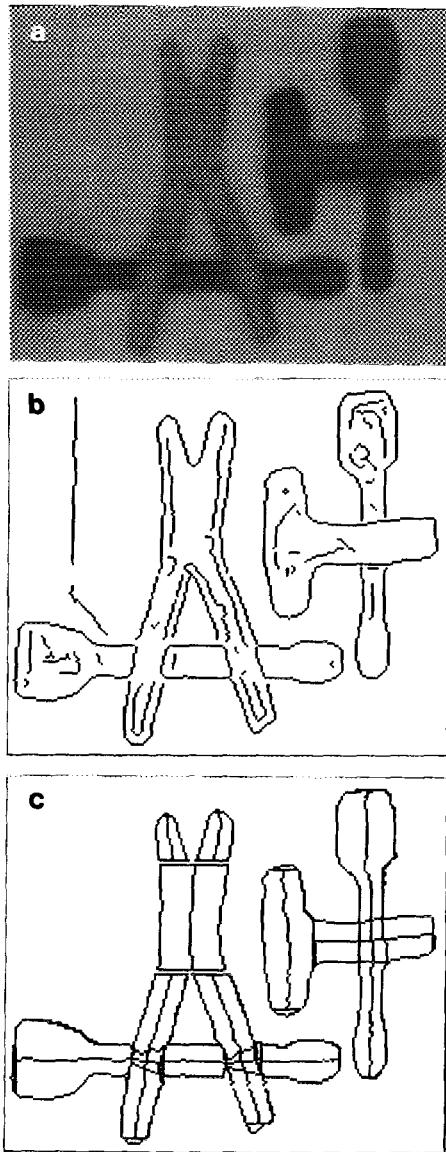


FIG. 34. (a) Original image of some tools; (b) input edge data; (c) final result.

examples show that we can *hypothesize missing parts* due to occlusion.

Figures 34 and 35 show results on more complex real images with the difficulties discussed above, in particular with more complex objects, with more occlusion and also more texture in the background and on the objects. We show Canny edges for these images. A large number of hypotheses are generated for these examples, and only the verified hypotheses are displayed.

The results here demonstrate that the system is capable of working with scenes with complex objects with surface markings on the object and background, shadows, highlights, occlusion, and multiple objects. The system is not given figure-ground information, and thus object boundaries are not known *a priori*. Also, the sys-

tem does not have models of the objects. Thus we are capable of working with imperfect 2D data without models, and we are capable of obtaining good shape descriptions, unlike previous methods ([4, 17, 5, 8, 6], for example), as discussed in Section 2.

The descriptions we obtain are high-level, segmented, and hierarchical. Besides selecting the desirable ribbons, we draw inferences about objects, their parts, the relationship between parts, and the joints that hold the parts together. In our processing we develop a graph structure that makes explicit the connectivity relationship between object parts. We also make intelligent guesses in cases of occlusion and hypothesize occluded parts. These descriptions are more sophisticated than those reported in literature (for example, [4, 17, 5]). Note also that the descriptions are at multiple levels (subsuming and subsumed descriptions), with the system working with one image at only one level of resolution. This is also more desirable than the totally global and single description obtained by the symmetric axis transform, for example.

To give an idea of the detailed running of the system, in Table 1 we present numbers at various stages of the pro-

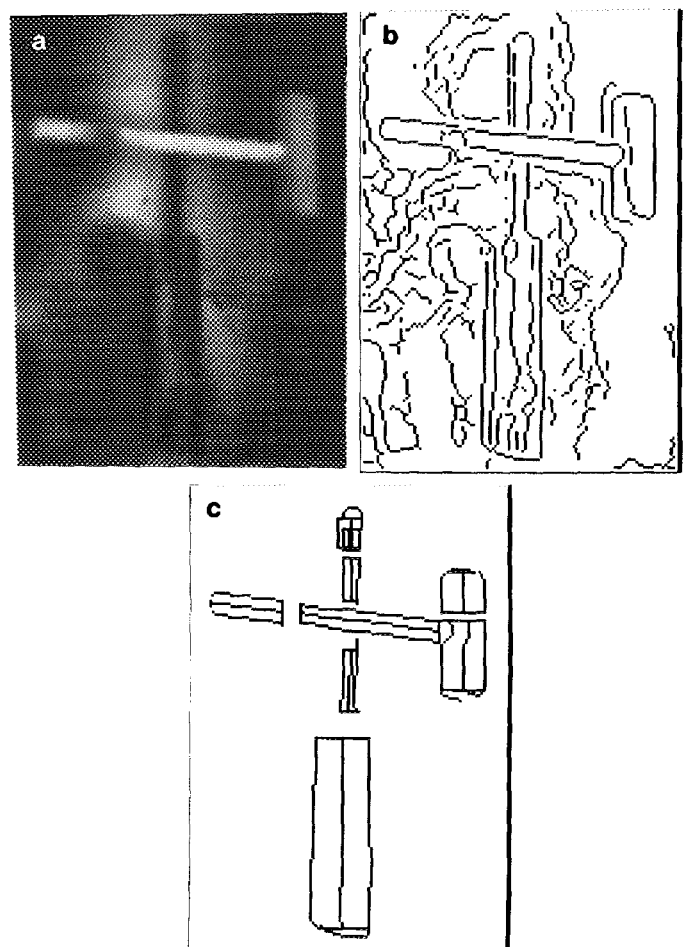


FIG. 35. (a) Original image with a lot of texture of a screw driver and hammer; (b) input edge data; (c) final result.

TABLE 1  
The Values of Some Variables at Various Stages of Processing  
for the Two Hammers Example

	Image name Figure number Image dimension	Two hammers Figure 32 181 by 111
No. of pixels		20091
No. of edges		1034
No. of axis points		7240
No. of linked lists of axis points		872
No. of super-ribbons		261
No. of nodes in the graph (all super-ribbons)		1044
No. of nodes after pruning		32
No. of ribbons after pruning		8
No. of cycles		4
No. of supercycles		4
No. of verified supercycles		4
No. of subsuming verified supercycles (compound objects)		2
No. of simple objects		1
No. of objects (simple and compound)		3
No. of subsuming objects		3
No. of superobjects		2

gram on the two hammers examples of Fig. 32. The image, 181 columns by 111 rows, has 1034 edgels. 7240 axis points are found, which give 872 ribbons and then 261 superribbons. These superribbons form 1044 possible nodes in the scene graph, which is pruned to 32 nodes corresponding to ribbons that have a potential to be verified. From these nodes four cycles are found; these are also the supercycles found (as there is no merging of cycles in this example, unlike in the plane and ship cases). All these supercycles are verified at their extremities. From these we obtain only two supercycles that subsume the others. These are the subsuming verified supercycles and are the compound objects in the scene. In this example, there is one simple object; the total number of objects, simple and compound, is three. These are also the subsuming objects. Two of these objects are grouped together and are merged to form one superobject. Finally, two superobjects are found.

A further idea of the running of the system may be obtained by looking at the order of magnitude run times of the major steps on a typical 256 by 256 one-bit edge image of the type shown in the results. Note that the run times are for code written for proof of concept and are not for optimized code. Axis point-finding takes 1 min. Linking of axis points takes an hour. Joining of ribbons to form superribbons is relatively fast and takes 1 min. Terminator finding is by depth-first search and is quite expensive and takes an hour. Scene graph formation is also expensive and takes an hour. Pruning, cycle-finding, and supercycle-finding take 1 min each. Finding objects, filtering out some by subsumption, grouping them by continuity, and merging them to form superobjects take 10 min

together. The total typical time is 3 h and is determined primarily by linking of axis points, and terminator and scene graph formation.

Complexity analysis (Appendix A of [22]) of the algorithms discussed here indeed reflects domination of run time by the above-mentioned more expensive operations. The overall time complexity is  $O(m^2n^5/r)$ , and the total number of descriptions is  $O(mn^2/r)$ . Here  $n$  is the number of edgels in an image of size  $r \times r$ , and  $m$  is the number of directions searched-for axes. If  $m$  is small and constant and  $r = O(n^{0.5})$ , the time complexity is  $O(n^{4.5})$ , and the number of descriptions is  $O(n^{1.5})$ . Though the expressions for complexity are fairly high, the bright side is that they are not exponential in the number of edgels in the image.

The system described here has some limitations. One limitation is that it works with objects that are well described as generalized cones. This means it prefers elongated objects. Objects like discs and spheres have several equally good descriptions and are not uniquely described as generalized cones. This, however, is not serious because it implies the system sometimes comes up with several equally good descriptions rather than a single description.

Another limitation arises from the principle that a ribbon is declared an object or a part iff it is terminated at both extremities by a boundary or another part (Principle of Termination, PI). A ribbon having terminations at both extremities may not necessarily correspond to a real physical object. It may merely be a drawing or a painting, but it will still pass our object test. For an example, see Fig. 36. It is difficult, however, for us to draw further

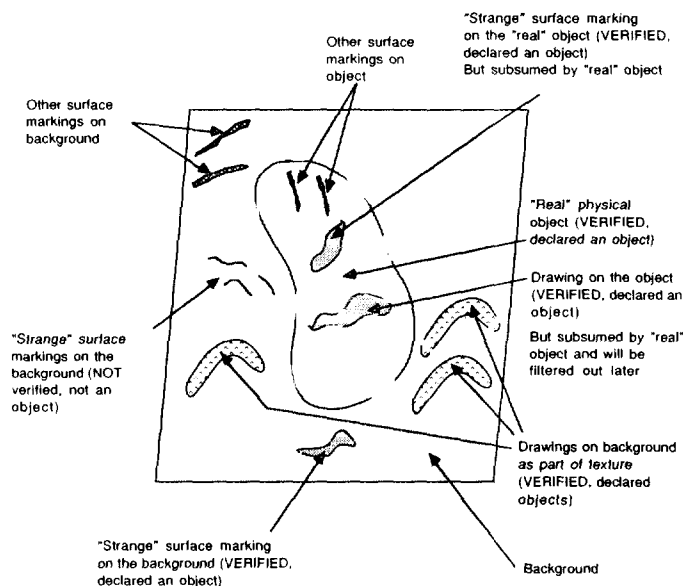


FIG. 36. An example scene of an object sitting on a textured background illustrating some limitations of the system.

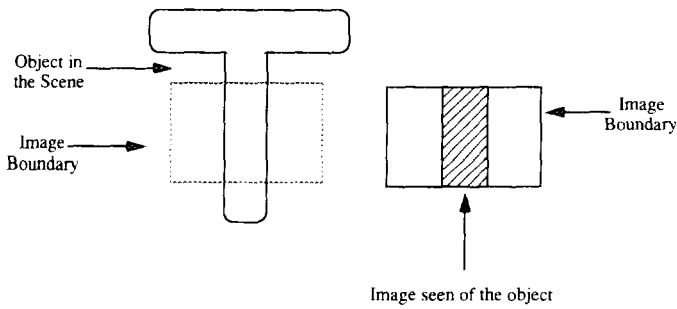


FIG. 37. Another example scene explaining some system limitations. As shown in the figure, an object only partially seen in the image due to image boundaries is not declared an object/object part. (a) Scene and imaging position: the object in the scene is shown by an unbroken line; the image boundary is shown broken. (b) The resulting image: the image boundary is shown by an unbroken line; the portion of the object in the image is shown shaded. The shaded portion is not declared an object/object part because it does not satisfy the Principle of Termination.

inferences on objects from images such as in Fig. 36. More texture and breaks will confuse the program, and it will come up with a larger number of possible descriptions and may even be led astray. If these extraneous descriptions are subsumed by the desirable ones, the final output of the system will be less confused and more useful. Note that we do not do much low-level processing (for example, segmenting edges into curves) because it is difficult to do so correctly without knowing the shape of the objects in the first place! But working with edges creates problems of generating a large number of candidates for higher level reasoning and description, and slows down the system.

Working conversely with the Principle of Termination, we note that some objects in a scene may not be completely seen in the image because of image boundaries. Ribbons of an object from such an image will not be terminated (by boundary fragments or by a part), and the object would thus not be found. See Fig. 37 for an example. Note that this case is different from the case of an object occluded by another object. In the latter case the occluding boundary provides the boundary for termination, as in some of our results, for example, Fig. 32. (Our system, however, could be extended to allow use of the

image boundary as a possible termination and to take care of the above limitation.)

Overall, SHAPE II is stable in its choice of descriptions because of the verification procedure. The preference attributes for ribbons in hypothesis generation (Section 4.1) are not used anywhere in their selection, except when we have two or more *verified* descriptions for the same region.

We now discuss the parameters and thresholds used in SHAPE II. Table 2 summarizes them. A high value of  $s$  results in coarser axes and lower computation time. Similarly, a low value of  $m$  results in coarser axis directions and lower computation time. A high value of  $q$  leads to larger axis fragments because more pieces become joined; but this may also result in incorrect joining. A high value of  $q$  also means more computation time. A high value of  $\tau$  results in larger axis fragments because it allows greater deviation from orthogonality of the cross section from the axis. It does not directly affect the computation time. As the third column in Table 2 indicates, all these parameters are related, and there is really one free parameter,  $s$ .

With respect to its different component steps, SHAPE II is modular. For example, the axis-finding method could be replaced with some other method for computing axes without affecting the rest of the system.

We have designed our system so that it is robust to noise. For example, to compute the tangent and curvature at axis points we use five-point B-spline formulae [16]. Also, to take care of noise in computing tangent and curvature, we skip the extremal points of each axis because they are unreliable: we start computing tangent and curvature at the third point from the beginning and stop at the third last point. If the axis has less than five points, the tangent and curvature cannot be computed; but we anyway filter out axes with less than five points immediately after linking.

We have in this section discussed results and the strengths and weaknesses of SHAPE II. The next section discusses conclusions and applications of this research.

## 6. CONCLUSIONS AND APPLICATIONS

In this article we have presented several novel ideas for shape description and segmentation of scenes with

TABLE 2  
Summary of Parameters and Thresholds Used in the SHAPE II System

Parameter/threshold	Notation	Value used
Pixel interval between axis points	$s$	1
No. of directions for axis finding	$m$	$m = 4s = 4$
Nearness of beads (for linking)	$q \times q$ neighborhood	$q = 2s + 1 = 3$
Threshold for axis point filtering	$\tau$	$\tau = \frac{180^\circ}{2m} = \frac{22.5^\circ}{s} = 22.5^\circ$

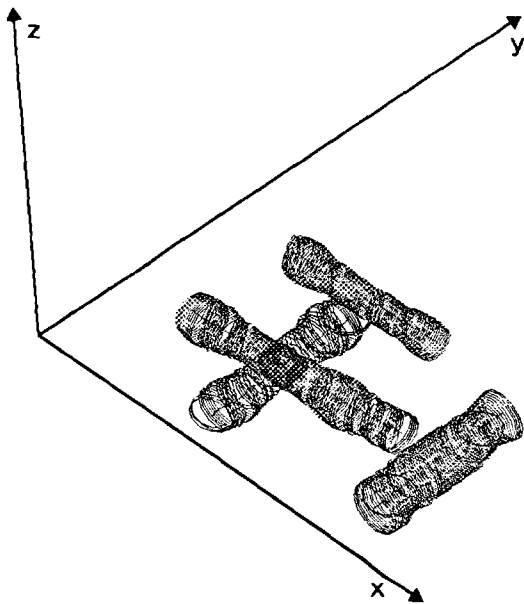


FIG. 38. The 3D generalized cone description for the two hammers example in Fig. 32.

compound objects, occlusion, non-trivial breaks in the object boundaries, and surface markings on the objects and the background. We have developed algorithms based on our ideas and have implemented them in a system. We have shown results on many real and synthetic images of objects. Previous systems would fail or produce undesirable descriptions from the imperfect and incomplete data that we work with. We thus believe our research here is a significant contribution to the vision field.

The descriptions we obtain can be used for a variety of purposes. They can be used for recognition of objects from their shapes in 2D images. For example, as has been observed in [19, 18, 15], objects can be recognized reasonably well from their 2D stick figures. Quite often the connectivity information in such stick figures is sufficient for recognition. It is these stick figures and their connectivity that we obtain from our system.

Such a shape description system can also be used for building object models and hence for learning. Stick figures can be used to learn new models [19, 29]. They can refine older ones by using different views. They can also help build robust models, despite occlusion and articulation of objects in some views. Ribbon descriptions are also useful for higher level matching between 2D images. For example, ribbons in two stereo images can be matched to obtain depth. Similarly, matched ribbons in a motion sequence can be used to obtain depth and estimate motion parameters. This can aid in navigation for a mobile robot.

Ribbon and symmetry descriptions can be used to esti-

mate 3D orientation of objects from a single 2D image. For example, one can use the heuristic that skew symmetry in 2D arises from orthogonal symmetry in 3D [12]; or one can use constraints on parallel and mirror symmetries, as in [28].

Two-dimensional ribbon descriptions may be utilized to obtain 3D generalized cone descriptions under some conditions. For instance, the contours of a solid of revolution are symmetric about the projection of its 3D axis [23]. If we can find the 2D axis of the projection of a solid of revolution, we can obtain its 3D axis using one more constraint/assumption or one more piece of information. We can then obtain a 3D GC description by, for example, assuming a circular cross section. We have thus obtained 3D generalized cone descriptions for several objects like the hammer. Figure 38 shows the generalized cone description for the two hammers example of Fig. 32.

The 3D generalized cone descriptions can then be used for several purposes. They can be used for better recognition performance and also for improved model building and learning. The 3D shape descriptions can also be used to preshape a robot hand for grasping [24], an application requiring such object-centered volumetric shape descriptions.

## REFERENCES

1. R. Bergevin and M. D. Levine, Part decomposition of objects from single view line drawings. *Comput. Vision Graphics Image Process.: Image Understanding*, **55**(1), 1992, 73-83.
2. I. Biederman, Human image understanding: Recent research and theory. *Comput. Vision Graphics Image Process.*, **32**(1), 1985, 29-73.
3. T. O. Binford, Visual perception by computer. in *IEEE Conference on Systems and Controls, Miami, December 1971*.
4. H. Blum, Biological shape and visual science (part 1). *J. Theoret. Biol.*, **38**, 1973, 205-287.
5. J. M. Brady and H. Asada, Smoothed local symmetries and their implementation. *Internat. J. Rob. Res.*, **3**(3), 1984, 36-61.
6. R. A. Brooks, Symbolic reasoning among 3-D models and 2-D images. *Artif. Intelligence*, **17**, 1981, 285-348.
7. J. Canny, A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-8**(6), 1986, 679-698.
8. J. H. Connell and M. Brady, Generating and generalizing models of visual objects. *Artif. Intelligence* **31**(2), 1987, 159-183.
9. S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, From volumes to views: An approach to 3-D object recognition. in *IEEE Computer Society Workshop on Directions in Automated CAD-Based Vision, Maui, Hawaii, June 1991*, pp. 85-96.
10. A. Guzman, Decomposition of a visual scene into three-dimensional bodies. In *AFIPS Proceedings Fall Joint Comput. Conf., 1968*, Vol. 33.
11. D. D. Hoffman and W. A. Richards, Parts of recognition. *Cognition*, **18**, 1985, 65-96.
12. T. Kanade, Recovery of the three-dimensional shape of an object from a single view. *Artif. Intelligence*, **17**, 1981, 409-460.

13. M. Leyton. A process-grammar for representing shape. in *Workshop on Spatial Reasoning and Multi-Sensor Fusion, Chicago, Illinois, October 1987*. pp. 148–157.
14. D. Marr. Analysis of occluding contour, *Proc. R. Soc. London Ser. B* **197**, 1977, 441–475.
15. D. Marr and K. Nishihara. Representation and recognition of the spatial organization of three dimensional shapes, *Proc. R. Soc. London Ser. B* **200**, 1978, 269–294.
16. G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic B-splines, *Comput. Vision Graphics Image Process.* **39**, 1987, 267–278.
17. L. R. Nackman and S. M. Pizer. Three-dimensional shape description using the symmetric axis transform I: Theory, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-7**(2), 1985, 187–201.
18. R. Nevatia, *Machine Perception*. Prentice–Hall, Englewood Cliffs, NJ, 1982.
19. R. Nevatia and T. O. Binford, Description and recognition of complex-curved objects, *Artif. Intelligence* **8**, 1977, 77–98.
20. A. P. Pentland. Perceptual organization and the representation of natural form, *Artif. Intelligence* **28**, 1986, 292–331.
21. J. Ponce, D. Chelberg, and W. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours, *IEEE Trans. Pattern Anal. Mach. Intelligence* **11**(9), 1989, 951–966.
22. K. Rao, *Shape Description from Sparse and Imperfect Data*, Ph.D. thesis (computer engineering), University of Southern California, Los Angeles, CA, December 1988; Technical Report USC-IRIS-250, USC, Institute for Robotics and Intelligent Systems.
23. K. Rao and G. Medioni. Generalized cones: Useful geometric properties, in *Progress in Computer Vision and Image Understanding* (L. Shapiro and A. Rosenfeld, Eds.), pp. 185–208, Academic Press, San Diego, CA, 1992.
24. K. Rao, G. Medioni, H. Liu, and G. Bekey. Shape description and grasping for robot hand-eye coordination, *IEEE Control Systems Magazine: Special Issue on Robotics and Automation* **9**(2), 1989, 22–29.
25. K. Rao and R. Nevatia. Computing volume descriptions from sparse 3-D data, *Internat. J. Comput. Vision* **2**(1), 1988, 33–50; also in *Advances in Spatial Reasoning*, Vol. 2, Chap. 2, Ablex, Norwood, NJ, 1989.
26. S. A. Shafer, *Shadow Geometry and Occluding Contours of Generalized Cylinders*, Technical Report CS-83-131, Carnegie-Mellon University, May 1983.
27. D. Terzopoulos, *Multiresolution Computation of Visible-Surface Representations*, Ph.D. thesis, Massachusetts Institute of Technology, Department of Computer Science and Electrical Engineering, January 1984.
28. F. Ulupinar and R. Nevatia. Using symmetries for analysis of shape from contour, in *International Conference on Computer Vision, December 1988, Tampa, Florida*, pp. 414–426.
29. P. H. Winston, *Learning Structural Descriptions from Examples*, Ph.D. thesis (computer science), Massachusetts Institute of Technology, January 1970.