CVGIP: IMAGE UNDERSTANDING
Vol. 55, No. 2, March, pp. 130–154, 1992

From Volumes to Views: An Approach to 3-D Object Recognition*

SVEN J. DICKINSON†

Center for Automation Research, University of Maryland, College Park, Maryland 20742-3411

ALEX P. PENTLAND

Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

AND

AZRIEL ROSENFELD

Center for Automation Research, University of Maryland, College Park, Maryland 20742-3411

Received August 19, 1991; accepted November 19, 1991

We present an approach to the recovery and recognition of 3-D objects from a single 2-D image. The approach is motivated by the need for more powerful indexing primitives, and shifts the burden of recognition from the model-based verification of simple image features to the bottom-up recovery of complex volumetric primitives. Given a recognition domain consisting of a database of objects, we first select a set of object-centered 3-D volumetric modeling primitives that can be used to construct the objects. Next, using a CAD system, we generate the set of aspects of the primitives. Unlike typical aspect-based recognition systems that use aspects to model entire objects, we use aspects to model the *parts* from which the objects are constructed. Consequently, the number of aspects is fixed and *independent* of the size of the object database. To accommodate the matching of partial aspects due to primitive occlusion, we introduce a hierarchical aspect representation based on the projected surfaces of the primitives; a set of conditional probabilities captures the ambiguity of mappings between the levels of the hierarchy. From a region segmentation of the input image, we present a novel formulation of the primitive recovery problem based on grouping the regions into aspects. No domain dependent heuristics are used; we exploit only the probabilities inherent in the aspect hierarchy. Once the aspects are recovered, we use the aspect hierarchy to infer a set of volumetric primitives and their connectivity relations. Subgraphs of the resulting graph, in which nodes represent 3-D primitives and arcs represent primitive connections, are used as indices to the object database. The verification of object hypotheses consists of a topological verification of the recovered graph, rather than a geometrical verification of image features. A system has been built to demonstrate the approach, and it has been successfully applied to both synthetic and real imagery. © 1992 Academic Press, Inc

1. INTRODUCTION

1.1. Motivation

A major problem in computer vision is the recognition of three-dimensional objects from a single two-dimensional image. A functional decomposition of a typical object recognition system is given in Fig. 1. From an input image, a set of features, or primitives, is extracted. The choice of which primitives to recover from the image depends on the task; primitives can range in complexity from collections of 2-D points to collections of 3-D volumes. In some cases, knowledge of the objects in the database can be used to aid primitive recovery. The next step, called the database indexing step, consists of querying the database to find out which object models contain a particular image primitive; hence, we call the extracted image primitives *indexing primitives*. The indexing process matches the image primitives to primitives composing the object models, returning a set of hypothesized correspondences between image and model primitives. The final step involves a verification of the hypothesized correspondences; verified correspondences are ranked according to an appropriate goodness-of-fit.

A multitude of object recognition paradigms have been proposed (see Besl and Jain [4], Chin and Dyer [15], and Binford [8] for comprehensive reviews), all differing in their primitive extraction, matching strategy, model rep-

* The support of the Air Force Office of Scientific Research under Contract AFOSR-86-0092 is gratefully acknowledged.

† The author's current address is Department of Computer Science, University of Toronto, 6 King's College Rd., Toronto, Ontario, Canada M5S 1A4

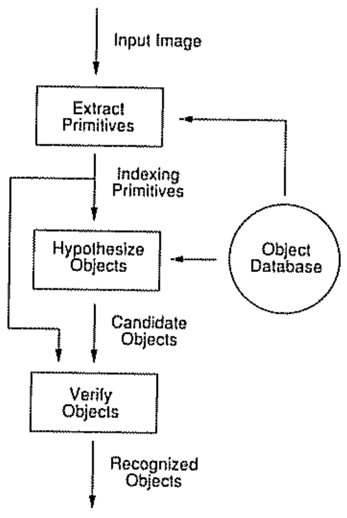


FIG. 1. The components of a typical object recognition system

resentation, verification strategy, or overall control. Despite the overwhelming variety of approaches, there is a very powerful metric that can be used to compare them. In particular, by examining the indexing primitives used in the various approaches, we can draw some powerful conclusions about building object recognition systems. In addition, we will see that the selection of indexing primitives not only affects system performance, but constrains the design of other recognition system modules.

A comparison of object recognition systems according to their indexing primitives is given in Fig. 2. In the left column are various indexing primitives ranging in complexity from low (e.g., 2-D points) to high (e.g., 3-D volumes), as depicted by the width of the leftmost bar (bar 1). Some of the indexing primitives are two-dimensional, while others are three-dimensional, often reflecting the type of input as intensity or range image data. Accompanying each indexing primitive is a reference to an exam-

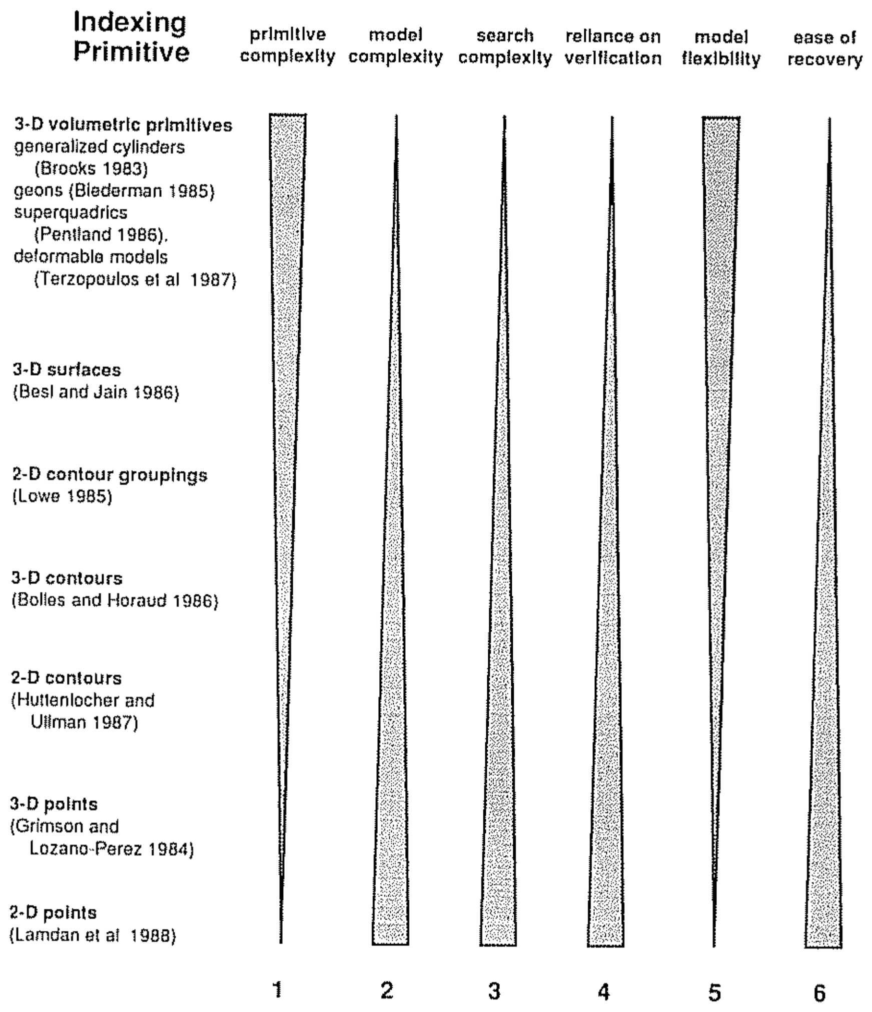


FIG. 2. A comparison of object recognition systems according to their indexing primitives

ple system that employs that primitive. Note that this list of indexing primitives is not complete; it is meant only to exemplify the range in complexity of possible indexing primitives.

Working from left to right in Fig. 2, we see that as the complexity of indexing primitives increases, the number of features making up the object models decreases (bar 2), since an object can be described by a few complex parts or by many simple parts. This, in turn, implies that the search complexity, i.e., the number of hypothesized matches between image and model primitives, decreases with increasing primitive complexity (bar 3). The high search complexity involving simple indexing primitives is compounded by large object databases. As a result, most systems using simple indexing primitives, e.g., Lowe [38], Huttenlocher and Ullman [33], Thompson and Mundy [71], and Lamdan *et al.* [36], are applied to small databases typically containing only a few objects.

Since the simple indexing primitives represent a more ambiguous interpretation of the image data (e.g., a few corners in the image may correspond to many corner triples on many objects), systems that employ simple primitives must rely heavily on a top-down verification step to disambiguate the data (bar 4). In this manner, the burden of recognition is shifted from the recovery of complex, discriminating indexing features to the model-based verification of simple indexing features. Since many different objects may be composed of the same simple features, these systems are faced with the difficult task of deciding which object to use in the verification step. However, there is a more fundamental problem with simple indexing features.

Relying on verification to group or interpret simple indexing primitives has two profound effects on the design of recognition systems. First, verifying the position or orientation of simple indexing features such as points or lines requires an accurate determination of the object's pose with respect to the image. If the pose is incorrect, searching a local vicinity of the image for some model feature may come up empty. Needless to say, accurately solving for the object's pose can be computationally complex, particularly when a perspective projection camera model is used.

Relying on verification also affects object modeling. Specifically, the resulting object models must specify the exact geometry of the object, and are not invariant to minor changes in the shape of the object (bar 5). Consider, for example, a polyhedral model of a chair. If we stretch the legs, broaden the seat, or raise the back, we would require a new model if our verification procedure were checking the position of points and lines in the image. Excellent work has been done to extend this approach to certain types of parameterized models, e.g., Grimson [29], Huttenlocher [32], and Lowe [39]. However, by nature of the indexing primitives, these models do not explicitly represent the gross structure of the object, and therefore cannot easily accommodate certain types of shape changes.

So far, bars 1 through 5 in Fig. 2 clearly indicate the advantages of using complex indexing features over simple ones. Why then do most 3-D from 2-D recognition systems use simple indexing primitives?[1] First of all, in certain domains, e.g., typical CAD-based recognition, in which the object database is very small, object models are constructed from simple primitives, object shape is fixed, and exact pose determination is required, simple indexing primitives have proven to be quite successful. However, more importantly, the reliable recovery of more complex features, particularly from a single 2-D image, is a very difficult problem (bar 6), particularly in the presence of noise and occlusion. Clearly, the major obstacle in the path of any effort to build a recognition system based on complex indexing primitives will be the reliable recovery of those primitives. This paper addresses this challenge. From a single 2-D image, we present an approach to the recovery and recognition of 3-D objects using complex 3-D volumetric indexing primitives.

1.2. Overview of the Approach

The approach presented in this paper is outlined in Fig. 3. Given a recognition task domain consisting of a database of objects, the first step is to select a set of object-centered 3-D volumetric primitives which, when assembled together, can be used to construct the objects. Next, using a CAD system, we map the set of volumetric primitives to a set of viewer-centered 2-D aspects whose number is fixed and *independent* of the size of the object database.

The aspects are represented by a hierarchy of 2-D features, called the *aspect hierarchy*, whose levels include the qualitative shapes of the primitives' projected surfaces (faces), subsets of the contours that bound the faces (boundary groups), and groups of faces (aspects). The relations between these features are then assessed from *all* viewpoints, resulting in a table of estimated conditional probabilities for each 2-D feature and primitive as a function of less complex 2-D features. For instance, one entry in this table might be the conditional probability that we are viewing a cylinder primitive given that we have found a rectangular face in the image. The computation of the aspect hierarchy, including tabulation of the conditional probabilities, is performed off-line.

[1] Many of the more complex indexing primitives, e.g., 3-D surface patches, deformable models, and superquadrics are typically recovered from range data images.

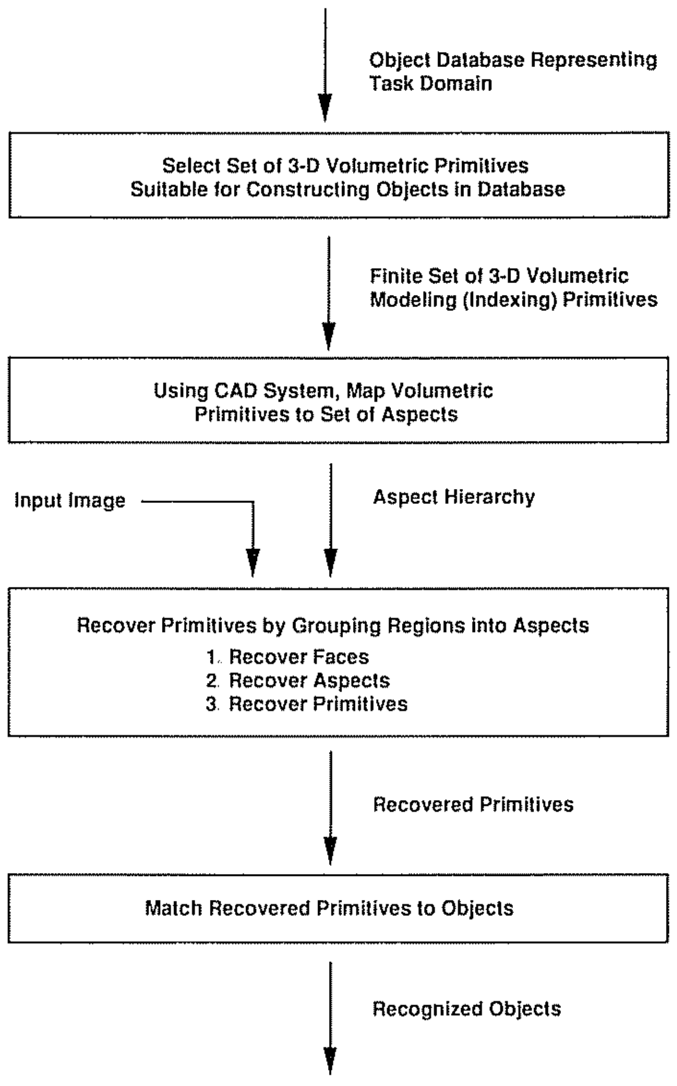

Object Database Representing Task Domain
Select Set of 3-D Volumetric Primitives Suitable for Constructing Objects in Database
Finite Set of 3-D Volumetric Modeling (Indexing) Primitives
Using CAD System, Map Volumetric Primitives to Set of Aspects
Input Image
Aspect Hierarchy
Recover Primitives by Grouping Regions into Aspects
1. Recover Faces
2. Recover Aspects
3. Recover Primitives
Recovered Primitives
Match Recovered Primitives to Objects
Recognized Objects

FIG. 3 Overview of the approach

The recovery of 3-D volumetric primitives from a 2-D image would normally entail a high search cost due to the complexity of the primitives. However, we have been able to avoid this problem by taking advantage of the computed probabilistic information about whatever set of modeling primitives the user has chosen. Given an image of a scene, this table of conditional probabilities is then used to guide a combinatorial search that yields a full and consistent interpretation of the viewed scene. The key idea is that the statistical properties of the set of user-defined primitives are used to avoid a combinatorial explosion in the search process. Knowledge about how each primitive looks from all angles makes for a more informed search, supporting the recovery of much more complex image features than are typically employed.

Primitive recovery consists of three steps. From a region segmentation of the image, each region is classified (or labeled) according to the faces in the aspect hierarchy; if a face is occluded, it may receive multiple labels. Next, the regions are partitioned into groups, each corresponding to an aspect of a primitive. We formulate the problem as a region labeling and again use the conditional probabilities within the hierarchy to avoid a combinatorial explosion. The final step consists of mapping the 2-D aspects to the 3-D primitives, and recovering the primitive connections.

From a set of recovered 3-D volumetric primitives, we then proceed to the task of recognizing the object(s) contained in the scene. Groups of recovered primitives, including their connections, are used as indices to the object database. Unlike typical 3-D recognition systems which rely heavily on geometric verification to confirm or reject object hypotheses, we rely on topological (qualitative) verification of part aspects (image faces). We show that the same mechanisms can be applied both to the problem of unexpected object recognition (Rosenfeld [58]) and to the problem of searching the image for a particular object (expected object recognition).

In previous work, we described our representation integrating object-centered and viewer-centered models [16] and our technique for recovering 3-D volumetric primitives from a 2-D image [18]. This paper focuses on the final component of our system, the object recognition component. Following a review of the representation and recovery techniques, we present the recognition algorithm in detail. The integrated system, including the recognition component, is demonstrated on both synthetic and real imagery.

2. OBJECT MODELING

2.1. Choosing the 3-D Primitives

As discussed in Section 1, volumetric primitives offer a powerful indexing mechanism for object recognition. Nevertheless, the question arises: Why volumetric primitives? Why not 3-D surface patches, or even complete objects, as in the case of typical aspect-based recognition systems? We believe that volumetric primitives represent the most intuitive decomposition of an object into parts. Object models composed of volumetric primitives easily support part articulation, and at the structural level are insensitive to dimensional changes in the parts. Another compelling reason for their selection is their ability to support functional-based object recognition (Stark and Bowyer [64]). Reasoning about object function requires not only a higher-level representation of the image data, but a good mapping between shape and functional primitives. For example, reasoning about the functionality of a recovered chair leg is better served by a volumetric representation of the leg than by a representation consisting of a series of surface patches. Additional support for volumetric primitives based on the criteria of accessibility, scope and uniqueness, and stability and sensitivity is given by Marr [41].

Many researchers use volumetric primitives to model objects. An often used class of primitives is the class of generalized cylinders (e.g., Binford [7], Agin and Binford [1], Nevatia and Binford [47], Marr and Nishihara [40], Brooks [10]) whose cross-section, axis, and sweep properties are arbitrary functions. Superquadrics (Gardiner [24]) provide a volumetric representation requiring only a few parameters. Pentland [49] first applied superquadrics to primitive modeling for object recognition, while Pentland [50] and Solina [62] have achieved considerable success in deriving superquadric primitives from range data. Active or physically-based models have been used by Terzopoulos *et al.* [68–70], Metaxas and Terzopoulos [43, 44], and Pentland and co-workers [51–54] to successfully recover 3-D shape and nonrigid motion from natural imagery. Although generalized cylinders, superquadrics, and active models provide a rich language for describing parts, their extraction from the image is computationally complex. In addition, many of the above systems rely on range data as input, and often require strong initial conditions or even manual intervention.

Given a database of object models representing the domain of a recognition task, we seek a set of three-dimensional volumetric primitives that, when assembled together, can be used to construct the object models. Whichever set of volumetric modeling primitives is chosen, they will be mapped to a set of viewer-centered aspects. Consider, for example, a rectangular block primitive which might be a component of many objects in a database. Let us assume that for each object of which it is a component, its dimensions are different. If our aspect definitions were *quantitative*, specifying the exact geometry of image features, each instance of the block would map to a different set of aspects. However, if the aspect definitions were *qualitative*, providing stability under minor changes in the shape of the primitives (e.g., scale, dimension, and curvature), a single set of aspects might represent all possible instances of a rectangular block. Our approach, therefore, has been to select a set of qualitatively-defined volumetric primitives, so that their description will be invariant under such changes in shape.

To demonstrate our approach to primitive recovery, we have selected an object representation similar to that used in Biederman's Recognition by Components (RBC) theory [6]. RBC suggests that from nonaccidental relations in the image, a set of contrastive dichotomous (e.g., straight vs. curved axis) and trichotomous (e.g., constant vs. tapering vs. expanding/contracting cross-sectional sweep) 3-D primitive properties can be determined. The Cartesian product of the values of these properties gives rise to a set of volumetric primitives called geons.

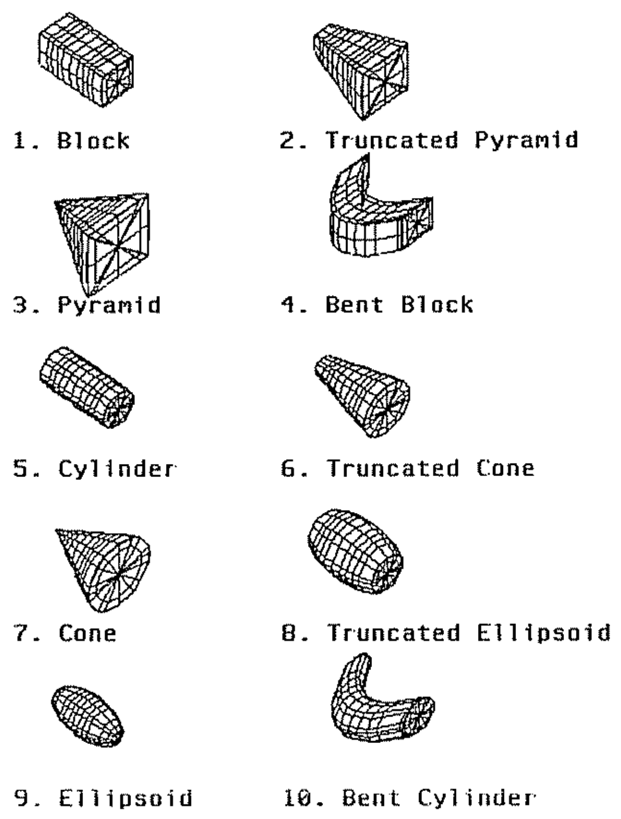


FIG. 4 The 10 primitives

Biederman's geons constitute only one possible selection of qualitatively-defined volumetric primitives; the general approach of applying the Cartesian product to a set of contrastive primitive properties can be used to generate many different volumetric primitive representations. For our investigation, we have chosen three properties: cross-section shape, axis shape, and cross-section sweep. The values of these properties give rise to a set of 10 qualitative volumetric primitives.[2] To construct objects, the primitives are simply attached to one another with the restriction that any junction of two primitives involves exactly one attachment surface from each primitive, i.e., an attachment cannot lie on a surface discontinuity.

In our system, these 10 primitives were modeled using Pentland's SuperSketch 3-D modeling tool [49], as illustrated in Fig. 4.[3] We believe that this taxonomy of volu-

[2] The Cartesian product of the values of these properties results in a set of 20 primitives; however, to simplify the investigation in terms of generating the conditional probability tables described in the next section, we have chosen a subset of 10 primitives which we believe to be a good basis for modeling a wide range of objects. If necessary, more primitives could easily be added to enhance the vocabulary.

[3] SuperSketch models each primitive with a superquadric surface that is subjected to bending, tapering, and pinching deformations.

metric primitives is sufficient to model a large number of objects; however, nothing in our approach is specialized for this particular set of primitives. If necessary, our approach can easily accommodate other sets of volumetric primitives.

2.2. Defining the 2-D Aspects

Traditional aspect graph representations of 3-D objects model an entire object with a set of aspects, each defining a topologically distinct view of an object in terms of its visible surfaces (Koenderink and van Doorn [34]). Our approach differs in that *we use aspects to represent a (typically small) set of volumetric primitives from which each object in our database is constructed, rather than representing an entire object directly*. Consequently, our goal is to use aspects to recover the 3-D primitives that make up the object in order to carry out a recognition-by-parts procedure, rather than attempting to use aspects to recognize entire objects. The advantage of this approach is that since the number of qualitatively different primitives is generally small, the number of possible aspects is limited and, more important, *independent* of the number of objects in the database. In contrast, the number of aspects required to model complete objects grows with the size of the database, and is compounded by articulating objects. The disadvantage is that if a primitive is occluded from a given 3-D viewpoint, its projected aspect in the image will also be occluded. Thus we must accommodate the matching of occluded aspects, which we accomplish by introducing a hierarchical aspect representation we call the *aspect hierarchy*.

The aspect hierarchy consists of three levels, based on the faces appearing in the aspect set; Fig. 5 illustrates a portion of the aspect hierarchy.

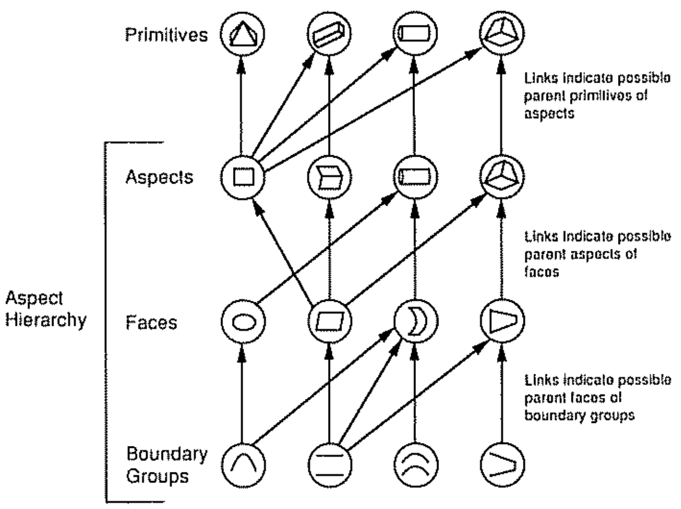

Primitives
Links indicate possible parent primitives of aspects
Aspects
Links indicate possible parent aspects of faces
Aspect Hierarchy
Faces
Links indicate possible parent faces of boundary groups
Boundary Groups

FIG. 5. The aspect hierarchy

• *Aspects* constitute the top level of the aspect hierarchy; they consist of the possible groups of faces for each of the primitives. Identification of the aspects can allow identification of the visible primitives. However, due to occlusion, some of the faces in an aspect may be partially or completely missing. When this occurs, we may need to analyze the arrangement of the remaining faces, and so we introduce the second level of the aspect hierarchy.

• *Faces* that make up the various aspects form the second level of the aspect hierarchy. Reasoning about the type and arrangement of visible faces can allow identification of an aspect even when it is partially occluded. However, again due to occlusion, some of the contours that make up a face may be partially or completely missing. When this occurs, we may need to analyze the arrangement of the remaining contours bounding the face, and so we introduce the lowest level of the aspect hierarchy

• *Boundary Groups* are subsets of the faces' bounding contours and make up the third and lowest level of the aspect hierarchy. The boundary groups provide a mechanism for identifying the face type even when the face is partially occluded.

2.3. Relating the 2-D Aspects to the 3-D Primitives

A given boundary group may be common to a number of faces. Similarly, a given face may be a component of a number of aspects, while a given aspect may be the projection of a number of primitives. To capture these ambiguities, we have created a matrix representation that describes conditional probabilities associated with the mappings from boundary groups to faces, faces to aspects, and aspects to primitives. For example, consider the mapping between faces and aspects. To describe this mapping, we create a matrix whose rows represent faces and whose columns represent aspects. If a particular face can be a component of 10 different aspects, then those 10 column entries corresponding to the 10 aspects each contain a value from 0 to 1.0, indicating the probability that the face is part of that particular aspect. Thus, the entries along each row sum to 1.0.

To generate these conditional probabilities for the boundary group to face, face to aspect, and aspect to primitive mappings, we use the following procedure. We first model our 3-D volumetric primitives using the SuperSketch modeling tool [49], as shown in Fig. 4. The next step in generating the probability tables involves rotating each primitive about its internal x, y, and z axes in 10° intervals. The resulting quantization of the viewing sphere gives rise to 648 views per primitive; however, by exploiting primitive symmetries, we can reduce the number of views for the entire set of primitives to 688. For each view, we orthographically project the primitive onto the image plane, and note the appearance of each feature (boundary group, face, and aspect) and its parent. The resulting frequency distribution gives rise to the three conditional probability matrices (which can be found in [17]).

This procedure implicitly assumes that all primitives are equally likely to appear in the image, and that all spatial orientations of the primitives are equally likely. In practice, this is a strong assumption since both the frequency of occurrence and the spatial orientation distribution of a primitive are governed by the contents of the object database. A more effective approach would be to preprocess the object database, counting the number of times each primitive appears in each object and noting the primitive's orientation. The resulting set of *a priori* probabilities of occurrence and orientation could then be easily incorporated into the analysis, providing a set of tables that accurately reflect the contents of the object database.

It should be emphasized that these results offer only a rough approximation to the true probabilities. A more thorough analysis would use a finer quantization of both the primitives' parameters and the viewing sphere, and would measure the conditional probabilities directly from image data. The resulting explosion of views would require an automated tool to perform the analysis and generate the probabilities; much of the current analysis is performed manually.

A number of algorithms exist for automatically computing aspect graphs for specific object and projection models. For example, for convex polyhedra, see Stewman and Bowyer [66] (perspective); for general polyhedra, see Plantinga and Dyer [55] (orthographic and perspective), Gigus and Malik [25] (orthographic), Gigus, Canny, and Seidel [26] (orthographic), and Stewman and Bowyer [65] (perspective); for solids of revolution, see Eggert and Bowyer [19] (orthographic) and Kriegman and Ponce [35] (orthographic); for curved objects, see Sripradisvarakul and Jain [63] (orthographic); and for articulated objects, see Sallam and Bowyer [60]. Unfortunately, our needs extend beyond the computation of aspect graphs. In fact, the transitions between aspects (called visual events) in an aspect graph are not used in our approach; only the set of distinct aspects is applicable. However, in addition to the aspects, we need the conditional probabilities associated with the aspects and their component features.

3. PRIMITIVE RECOVERY

The aspect hierarchy effectively *prunes* the mapping from boundary groups to primitives by introducing topological and probabilistic constraints on the boundary group to face, face to aspect, and aspect to primitive mappings. An analysis of the conditional probabilities [18] suggests that for 3-D modeling primitives which resemble the commonly used generalized cylinders, superquadrics, or geons, the most appropriate image features for recognition appear to be image regions, or faces. Moreover, the utility of a face description can be improved by grouping the faces into the more complex aspects, thus obtaining a less ambiguous mapping to the primitives and further constraining their orientation. Only when a face's shape is altered due to primitive occlusion or intersection should we descend to analysis at the contour or boundary group level. Our approach, therefore, first segments the input image into regions and then determines the possible face labels for each region. Next, we assign aspect labels to the faces, effectively grouping the faces into aspects. Finally, we map the aspects to primitives and extract primitive connectivity. The following sections describe the approach in greater detail.

3.1. Face Extraction

From an analysis of the conditional probabilities, we concluded that the aspect-to-primitive mapping was the least ambiguous mapping to the primitives. In addition, we concluded that the best mapping to the aspects was from the faces rather than from the boundary groups. This suggests that faces are an appropriate starting point in the primitive recovery process. Since we characterize faces by their bounding contours, our first step is to extract a set of contours from the image; this can be accomplished using either region-based or edge-based methods.[4]

Once a set of contours has been extracted from the image, the next step is to partition the contours at significant curvature discontinuities. The segmented contours are captured in a *contour graph* in which nodes represent junctions or significant curvature discontinuities, and arcs are the actual bounding face contours. Given the contour graph representation of an input scene, our next task is to construct its corresponding *face graph* in which nodes represent faces and arcs represent face adjacencies.[5] The algorithm for transforming a contour graph into a face graph is based on Roberts [57] and can be found in [18].

3.2. Face Labeling

Once the faces have been extracted, we must classify each face according to the faces in the aspect hierarchy. Each face is represented by a graph in which nodes repre-

[4] For our task of extracting faces, region-based methods are preferable since they avoid the problem of contour gaps which can break the cycle of contours bounding a region.

[5] Two faces are said to be adjacent if they share one or more contours in the contour graph representation.

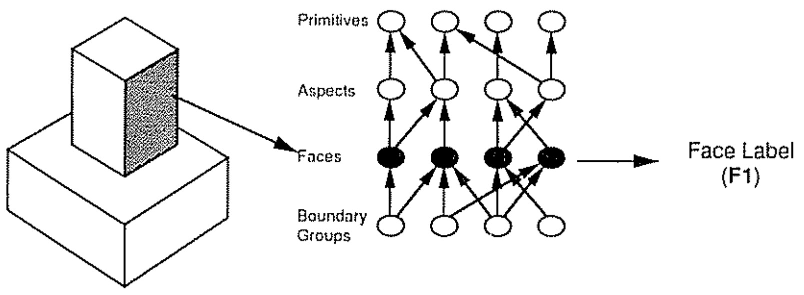


FIG. 6. Labeling an unoccluded face

sent the face's contours and arcs represent relations between contours.[6] Thus, the classification of a face in the image consists of comparing its graph to those graphs representing the faces in the aspect hierarchy. If there is an exact match, as shown in Fig. 6, then we immediately generate a *face hypothesis* for that image face, identifying the label of the face. If, due to occlusion, there is no match, we must descend to the boundary group level of the aspect hierarchy, as shown in Fig. 7. We then compare *subgraphs* of the graph representing the image face to those graphs at the boundary group level of the aspect hierarchy. For each subgraph that matches, we generate a face hypothesis with a probability determined by the appropriate entry in the conditional probability matrix mapping boundary groups to faces.

Thus, from the original contour graph representation of the image, we first construct a face graph, and then for each face in the face graph, the classification process results in a list of hypotheses about the face's label. In the simple case of an image face that exactly matches a face found in the aspect hierarchy, the list contains a single hypothesis with probability 1.0.[7] For an image face that does not exactly match a face found in the aspect hierarchy, the list contains one or more face hypotheses listed in decreasing order of probability.[8]

[6] Two adjacent collinear or curvilinear contours bounding a face may have been separated in the contour graph by a junction. If so, they are merged to form one node in the graph. In addition, all nodes are classified as either a straight line, a concave curve, or a convex curve.

[7] Due to occlusion, the fact that an image face exactly matches an aspect hierarchy face does not guarantee that the interpretation (label) of the image face is correct. A more precise analysis would go ahead and compare the image face's boundary groups to aspect hierarchy boundary groups, ensuring that the correct face hypothesis is generated. Nevertheless, the hypothesis representing the matched face would still have the highest probability.

[8] There can be no redundancy in this list since we do not allow multiple face hypotheses with the same label. If two boundary groups give rise to the same face hypothesis, the hypothesis is assigned the higher probability of the two boundary group to face mappings [18].

3.3. Extracting Aspects

3.3.1. *Problem Definition*

We now have a face graph with one or more face hypotheses at each face. We can formulate the problem of extracting aspects as follows: Given a face graph and a set of face hypotheses at each face, find a covering of the face graph using aspects in the aspect hierarchy, an *aspect covering*, such that no face is left uncovered and each face is covered by only one aspect. Or, more formally: Given an input face graph, FG, partition the vertices (faces) of FG into disjoint sets, $S_1, S_2, S_3, \ldots, S_k$, such that the graph induced by each set, S_i, is isomorphic to the graph representing some aspect, A_j, from a fixed set of aspects, $A_1, A_2, A_3, \ldots, A_n$.

There is no known polynomial time algorithm to solve this problem (see [18] for a discussion of the problem's complexity); however, the conditional probability matrices provide a powerful constraint that can make the problem tractable. After the previous steps, each face in the face graph has a number of associated face hypotheses. For each face hypothesis, we can use the face-to-aspect mapping to generate the possible *aspect hypotheses* that might encompass that face, as shown in Fig. 8; the face hypothesis becomes the *seed face hypothesis* of each of the resulting aspect hypotheses. The probability of an aspect hypothesis is the product of the face to aspect mapping and the probability of its seed face hypothesis. At each face, we collect all the aspect hypotheses (corresponding to all face hypotheses) and rank them in decreasing order of probability.

Each aspect hypothesis is merely an informed guess as to the aspect label of its seed face hypothesis. The process of verifying the hypothesized aspect label is called *aspect instantiation*. For an aspect to be instantiated from an aspect hypothesis, the relations between the seed face hypothesis and neighboring face hypotheses must be consistent with the definition of the aspect. More formally, there must exist a set of faces, S, including the face corresponding to the seed face hypothesis, such that

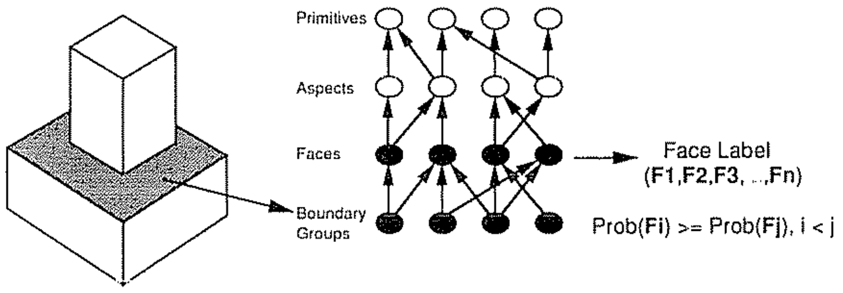


FIG 7. Labeling an occluded face.

the face subgraph induced by S is isomorphic to the graph representing the aspect. Since there may be multiple sets of faces which satisfy this criterion, there may be multiple aspects instantiated from a single aspect hypothesis. Hence, the process of aspect instantiation produces a (possibly empty) set of instantiated aspects for a given aspect hypothesis. For a detailed discussion of aspect instantiation and how occluded aspects can be instantiated, see [18].

We can now reformulate our problem as a search through the space of aspect labelings of the faces in our face graph. In other words, we wish to choose one aspect hypothesis from the list at each face, such that the instantiated aspects completely cover the face graph. Figure 9 illustrates the correct aspect covering of the face graph representing the object composed of two blocks; one aspect label, in this case, A27 (see [18] for a description of all aspects), is selected from the list at each face to completely cover the graph. There may be many labelings which satisfy this constraint. Since we cannot guarantee that a given aspect covering represents a correct interpretation of the scene, we must be able to enumerate, in decreasing order of likelihood, all aspect coverings until the objects in the scene are recognized.

3.3.2. *Algorithm for Enumerating Aspect Coverings*

For our search through the possible aspect labelings of the face graph, we employ Algorithm A (Nilsson [48]) with a heuristic based on the probability estimates for the aspect hypotheses. The different labelings are ordered in the open list according to the heuristic. At each iteration, a labeling, or state, is removed from the open list and checked to see if it represents a solution (a covering). The successor states are then generated, evaluated, and added to the open list. The actual instantiation of aspects is performed during successor generation. The algorithm continues until all possible solutions are found; i.e., all labelings are checked. However, it should be pointed out that in an object recognition framework, once a solution is found, the search would only be continued if the recovered shapes (inferred from the aspect covering) could not be recognized.

Before adding a successor state to the open list, it is evaluated using the heuristic function. This function has been designed to meet three objectives. First, we favor selections of aspects instantiated from higher probability aspect hypotheses. Second, we favor selections whose aspects have fewer occluded faces, since we are more sure of their labels. Finally, we favor those aspects covering more faces in the image; we seek the minimal aspect covering of the face graph. These three objectives have been combined to form the algorithm for evaluating a state, described in [18].

3.4. Extracting Primitives

We can represent an aspect covering by a graph in which nodes represent aspects and arcs represent aspect adjacencies. The next steps are to map the aspects in the

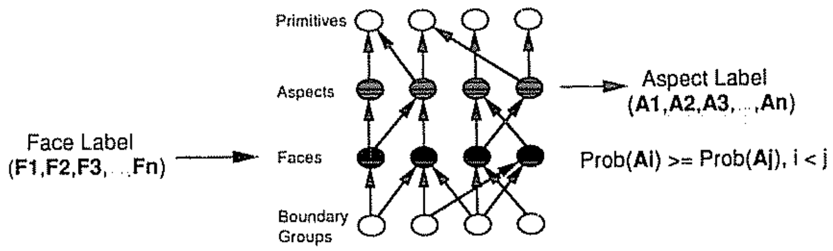

Primitives
Aspects
Aspect Label (A1,A2,A3,...,An)
Face Label (F1,F2,F3,...,Fn)
Faces
Prob(Ai) >= Prob(Aj), i < j
Boundary Groups

FIG 8. Generating the aspect labels of a face.

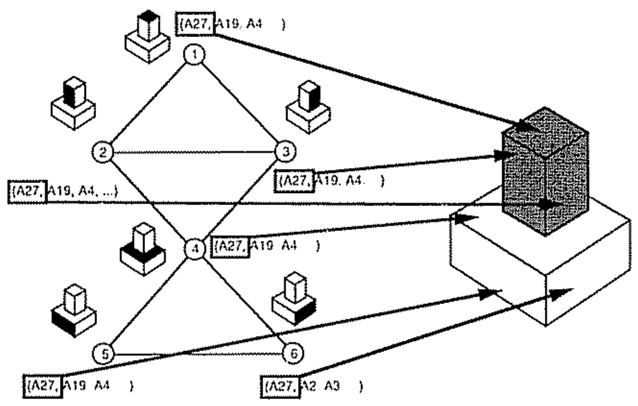


FIG. 9 Covering the face graph with aspects (the aspect labeling problem)

covering to a set of primitives and to extract their connectivity. The following sections describe this process in greater detail.

3.4.1. *Algorithm for Enumerating Primitive Coverings*

For each aspect in the aspect covering, we can use the aspect to primitive mapping to hypothesize a set of *primitives*, as illustrated in Fig. 10.[9] As in the case of aspect hypotheses generated from face hypotheses, we can rank the primitives in decreasing order of probability. A selection of primitives, one per aspect, represents a 3-D interpretation of the aspect covering; we call such a selection a *primitive covering*. Since we cannot guarantee that a given primitive covering represents a correct interpretation of the scene, we must be able to enumerate, in decreasing order of likelihood, all primitive coverings until the objects in the scene are recognized. To enumerate the selections, we employ a variation on the search algorithm used to enumerate the aspect coverings. The heuristic function simply negates the sum of the probabilities of the primitives, thereby favoring higher probability interpretations.

3.4.2. *Extracting Primitive Connectivity*

A primitive covering, represented by a graph in which nodes represent primitives and arcs represent primitive adjacencies, is then compared to the object database during the recognition process. If two aspects are not adjacent in the aspect covering, their corresponding primitives are not adjacent in the primitive covering. However, if two aspects are adjacent in the aspect covering, this does not mean that their corresponding primitives are necessarily adjacent in 3-D; one primitive may be occluding the other without being attached to it. A primitive connection between primitives P_1 and P_2 is said to be visible if the following condition is satisfied [18]:

- There exists a pair of faces, F_1 and F_2, such that F_1 belongs to the aspect corresponding to P_1 and F_2 belongs to the aspect corresponding to P_2, F_1 and F_2 are adjacent in the face graph, and F_1 and F_2 share a contour (following collinearity or curvilinearity grouping).[10]

Therefore, we define two types of primitive connectivity based on connection visibility:

- Two primitives are said to be *strongly* connected if their corresponding aspects are adjacent in the aspect graph, and the primitive connection is visible; in this case, we assume that the primitives are attached.
- Two primitives are said to be *weakly* connected if their corresponding aspects are adjacent in the aspect graph, and the primitive connection is *not* visible; in this case, one primitive occludes the other and it is not known whether or not they are attached.

A strong primitive connection strongly suggests the existence of a connection between two primitives. We can enhance the indexing power of a strongly connected subgraph if the attachment surfaces involved in each connection are hypothesized. Although it is impossible to define a set of domain independent rules which will, for any given set of primitives, correctly specify the attachment surfaces involved in a connection, we can define a set of heuristics which will specify a set of likely candidates [18]. If a strongly connected subgraph is common to two object models, these heuristics can then be used to rank order the candidates during verification.

4. OBJECT RECOGNITION

Given a primitive covering representation of the scene, in which nodes represent 3-D volumetric primitives and arcs represent strong or weak connections between the primitives, the final task is to identify the object(s) in the scene. There are two cases to consider. In an *unexpected* object recognition domain, we have no a priori knowledge of the contents of the scene. In this case, the recognition task consists of two steps: (1) identifying possible candidate models that might be present in the scene (model indexing), and (2) verifying that these models actually appear in the scene. In an *expected* object recognition domain, we search the image for one or more instances of a particular object. In this section, we present

[9] In addition, the aspect hierarchy defines a mapping from the faces in an aspect to the attachment surfaces of a primitive.

[10] Before grouping, two adjacent faces in the face graph share a contour by definition. However, following collinearity and curvilinearity grouping within their respective faces, they may not have a contour in common.

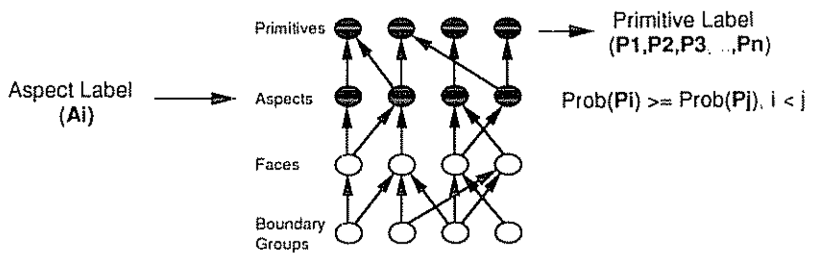

Primitives
Primitive Label (P1,P2,P3,...,Pn)
Aspect Label (Ai)
Aspects
Prob(Pi) >= Prob(Pj), i < j
Faces
Boundary Groups

FIG. 10. Generating the primitive labels of an aspect.

solutions to each of these problems, and show how the two solutions share the same underlying mechanism.

4.1. Unexpected Object Recognition

The simplest unexpected object recognition strategy is to compare the entire primitive covering to each model in the object database, i.e., verify each object model in the image. If the graph representing the primitive covering is isomorphic to the graph (or subgraph) representing an object in the database, then the object in the scene has been identified. However, there are two major problems with this naive approach. First, for large object databases, the cost of verification may be prohibitive, as was shown by Grimson [30]. Second, this approach assumes that a primitive covering represents a single object. If the scene contains multiple occluded objects, the primitive covering will not match a single object in the database. Thus, we are left with two problems: (1) How do we avoid matching the recovered primitives to each object in the database? and (2) What portions of the primitive covering likely belong to a single object, and should hence participate in the matching process? Together, these two issues lie at the heart of the model indexing problem, the subject of the next section. Following that, we discuss the verification of object models.

4.1.1. *Model Indexing*

An alternative to sequentially matching the recovered primitives to each model object is provided by *hashing* techniques. A hash table is a precomputed data structure each of whose entries (in our case) maps some recovered image feature(s) to a list of object models that contain that feature. The mapping between a recovered image feature and a location in the table is provided by a *hash function*. Once an image feature is "hashed" to an entry in the hash table, each of the objects referenced in the table entry must be verified. The advantage of hashing is that by preprocessing off-line the models in the object database, considerable on-line search can be avoided.

The hash table alone does not solve the problem, for if the recovered image features are simple, e.g., points, lines, or corners, they will be present in every object. The resulting hash table will have few entries (corresponding to a few simple indexing primitives), with each entry pointing to every object. Unfortunately, such a hash table leads us back to a sequential search of the database. Clearly, the goal in designing the hash table is to increase the size of the table, so that there are more entries, each having fewer pointers to object models.

Hash tables have been used in a variety of object recognition systems. For example, Flynn and Jain [23] propose a technique for 3-D object recognition from range data that automatically constructs invariant feature indexed tables from a CAD database. Invariant features include the angle between the normals of two planar surfaces, the angle between a plane's normal and a cylinder's axis of symmetry, the minimum distance between a plane and a sphere's center, the angle between two cylinder's axes, the minimum distance between a cylinder's axis and a sphere's center, and the distance between two spheres' centers. Breuel [11] describes a hashing technique whose hash function maps an invariant (under 2-D translation, rotation, and scale) description of a pair or triple of 2-D image features to a location in a combined feature vector representing a distinct view of a 3-D object. Each distinct view therefore requires a separate combined feature vector. Combined feature vectors recovered from the image are compared to model feature vectors, each representing a distinct view of a model object. In a similar approach, Lamdan *et al.* [36] describe a geometric hashing technique used in the recognition of 3-D planar objects. Specifically, three noncollinear image points are used to define a 2-D basis. The coordinates of the other image points are then represented with respect to this basis, and are invariant to any affine transformation. The resulting coordinates of an image point define an index into a hash table whose location contains the identity of the model(s) and the three points used to define the basis. A final example is provided by Ettinger [20] who addresses the problem of 2-D object recognition. In this approach, a 2-D subpart recovered from the image is used as an index to a table entry containing a list of 2-D objects containing the subpart. The subparts are contour features based on Asada and Brady's Curvature Primal Sketch (CPS) representation [2].

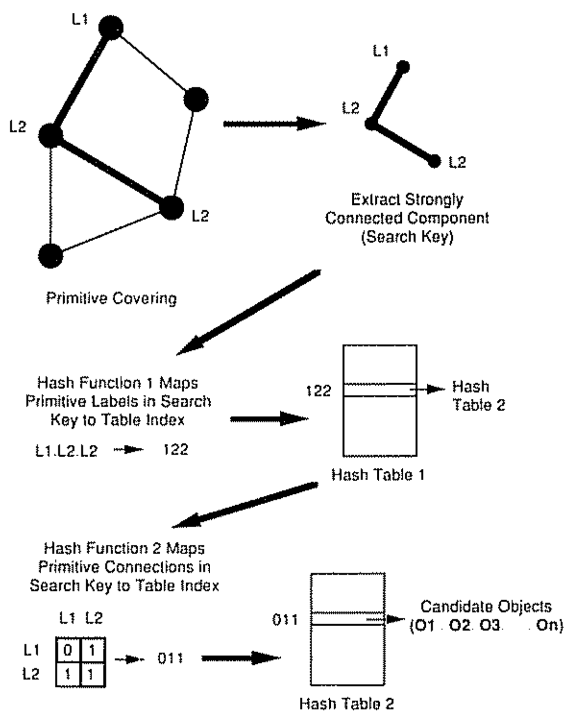

L1
L2
L2
Primitive Covering
L1
L2
L2
Extract Strongly Connected Component (Search Key)
Hash Function 1 Maps Primitive Labels in Search Key to Table Index
L1.L2.L2 → 122
122
Hash Table 2
Hash Table 1
Hash Function 2 Maps Primitive Connections in Search Key to Table Index
L1 L2
L1 0 1
L2 1 1
→ 011
011
Candidate Objects (O1 . O2 O3 . On)
Hash Table 2

FIG. 11. Hash function used to select possible object candidates

As stated in Section 1, our goal has been to recover from the image richer, more complex primitives whose combination offers a more discriminating index into the object database. Unlike simple features such as lines, points, or corners which are abundant in every object, a particular collection of 3-D volumetric primitives is unlikely to be common to many objects. Our solution, therefore, is to index with a collection of recovered primitives. Since we have a variety of primitives which can be connected in a variety of ways, the size of our hash table will be much larger than if we index with simple primitives. However, our second problem still remains: What collection of recovered primitives do we use as an index?

From a primitive covering of the input scene, we would like to index using a collection of recovered primitives that belongs to the same object. In Section 3.4, we assumed that if a connection between two primitives was *visible,* i.e., a *strong* connection, the primitives were connected in 3-D. Therefore, our model indexing strategy consists of first identifying all the strongly connected[11] components in the primitive graph.

Having decided on strongly connected primitives as our indexing structure, we can proceed to construct the hash table. First, we need all possible combinations of primitives appearing in objects in the database; this is equal to the union of the set of all object subgraphs. Unfortunately, there is no unique polynomial-time (in the number of graph nodes) mapping from a general graph to a string (or address of a location in the hash table); if there were, we could solve the subgraph isomorphism problem in polynomial time. One alternative is to encode all possible string representations of a graph (through a depth-first search) and have an entry for each string. Although this would be a polynomial-time mapping, the size of the hash table would be enormous. In the worst case, each graph would have $n!$ redundant entries, where n is the number of nodes in the graph.

[11] By strongly connected, we mean connected by *strong* arcs.

Our solution is to provide a more efficient mapping in terms of time and space at the cost of giving up some of the information encoded in a graph, i.e., a strongly connected component. We propose a two-level hash table. At the first level, we hash on the basis of a string formed from the labels of the primitives in the strongly connected component. At the second level, we hash on the basis of a string formed from the connections.

Figure 11 illustrates the hash function in more detail. The first hash function is based solely on the primitive labels and is calculated as follows. To limit the size of the table, we assume an upper bound on the size of a strongly connected component, say n.

1. Collect in a list the primitive labels of the primitives contained in the strongly connected component.
2. Sort the labels lexicographically.
3. The sorted labels represent the digits of an integer index whose base is equal to the number of primitive classes plus one.

Each entry in this table will point to a separate table at the second level which encodes primitive connections.

A hash table at the second level corresponds to objects that contain a particular set of primitives (number and type). It is at this level that we further discriminate the objects on the basis of their primitive connections. The second-level hash function is defined as follows:

1. Take the sorted list of primitive labels and remove duplicates. Let the number of remaining labels be n.
2. Construct an $n \times n$ matrix whose rows and columns correspond to the remaining labels. Initialize its elements to zero.
3. For each primitive connection, of the form (**a, b**) where **a** and **b** are the labels of the two primitives, increment by one the entries **a, b** and **b, a** in the matrix.
4. The entries in the upper triangular portion of the (symmetric) matrix visited in row-major order form an $n \times (n - 1)$ digit integer index whose base is equal to the maximum number of connections in a strongly connected component.

Each entry in this table will contain all the objects which have the same number and type of primitives, as well as the same number and type of connections. Of course, what is lost is a specification of which *particular* primitives participate in a given connection. The entire process is repeated for each strongly connected component in the primitive covering, resulting in a set of candidate object hypotheses for each strongly connected component.

Due to the relatively small number of objects in our demonstration database (order 10), the second hashing level was deemed unnecessary. Instead, entries in the first table point to objects that contain a particular numbers and types of primitives. Furthermore, to limit the size of the table, we consider only strongly connected components with three or fewer primitives; components having more than three primitives are broken down into their constituent components having three primitives. For our set of 10 primitives, the resulting size of the hash table is 1331 entries. We are currently adding more objects to the database; when hashing to the first table yields many candidate object hypotheses, we will add the second hash function and index with larger strongly connected components.

4.1.2. *Model Verification*

Given a set of candidate models, each containing a given strongly connected component (or part thereof), the final step is to evaluate how well each model fits the scene (primitive covering). This process is known as hypothesis verification and consists of two stages. The first stage is to find the maximal correspondence between recovered primitives and model primitives in terms of the number of primitives. The second stage is to rank the various correspondences according to an appropriate goodness of fit.

Growing Correspondences. Since our hash function ignores the connections in the strongly connected component, the first step is to check that the strong connections in the strongly connected component exist between the corresponding primitives in each candidate model; this may result in some candidate models being discarded. At this point, for each remaining candidate model, the strongly connected component in the primitive graph is isomorphic to a subgraph of the candidate model. We then grow this correspondence according to the following steps:

1. Given a correspondence between a primitive subgraph PS and a model subgraph MS, we first choose a model primitive M_i that is not contained in the model subgraph, but is connected to a primitive M_j in the model subgraph. In the primitive subgraph, let the primitive corresponding to M_j be P_j.
2. Among the neighbors (through strong or weak connections) of P_j in the primitive subgraph which are not contained in PS, select those whose label matches that of M_i. If more than one such neighbor exists, we create a new correspondence for each neighbor.

```
goodness of fit = 0
for each image primitive in correspondence do
    goodness of fit = goodness of fit + probability(image primitive)
for each arc in primitive subgraph do
    if arc is weak then
        goodness of fit = goodness of fit + $c_1$
    else (arc is strong)
        if arc correctly specifies attachment surfaces then
            goodness of fit = goodness of fit + $c_2$
        else
            goodness of fit = goodness of fit + $c_3$
```

FIG. 12. Goodness of fit algorithm for model candidates.

We repeat this sequence of steps for each correspondence until its size stabilizes. The entire process is then repeated for each strongly connected component in the primitive covering. The final result is a list of correspondences, each mapping a subgraph of the primitive covering to a model subgraph.

Ranking the Correspondences. The final step ranks the correspondences according to a goodness of fit measure defined by the algorithm shown in Fig. 12. The goodness of fit measure is a function of the size of the correspondence, the probability of the recovered primitive hypotheses, the visibility of the primitive connections, and the degree to which the connections are correctly specified. The input to the algorithm is a *correspondence*, consisting of a *primitive subgraph* whose nodes represent *image primitives*, and a *model subgraph* whose nodes represent *model primitives*. For the experiments described in Section 5, the values of c_1, c_2, and c_3 were chosen to be 1.0, 3.0, and 2.0, respectively; highest weight has been given to visibly connected primitives whose attachment surfaces have been correctly hypothesized.

4.1.3. *Recognizing Multiple Objects*

Once the correspondences are ranked according to our goodness of fit measure, we choose the best correspondence and remove those aspects from the image that correspond to the recognized primitives. From the remaining aspects, forming a new aspect covering, we repeat the entire process. We first apply the primitive covering algorithm, establish primitive connectivity, extract strongly connected components, determine candidate models, grow and rank the correspondences, and select the most likely correspondence. The process is repeated until no aspects remain in the image. At any stage, a primitive covering may not yield any recognizable objects, i.e., candidate models. In this case, we generate a new primitive covering from the current aspect covering and repeat

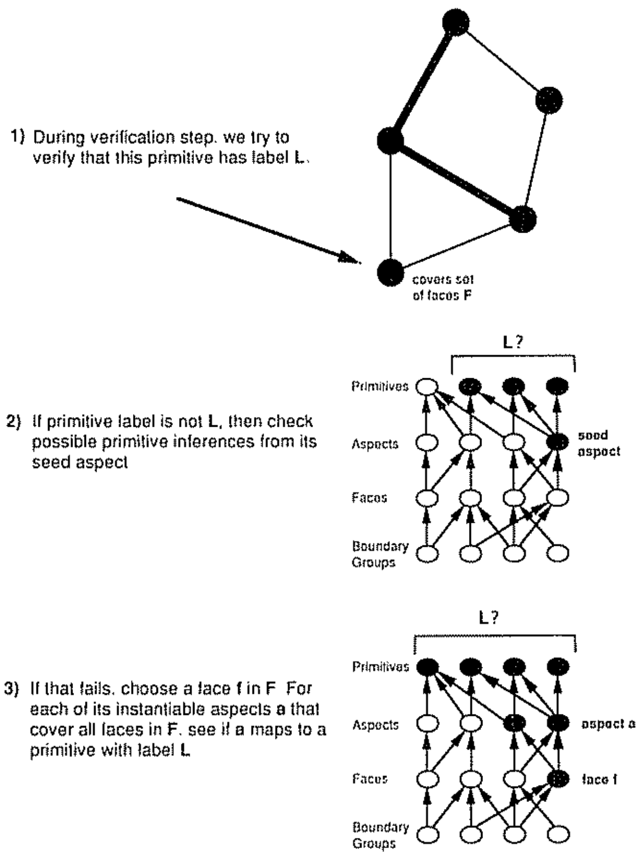

1) During verification step, we try to verify that this primitive has label L.
covers set of faces F
L?
Primitives
Aspects
Faces
Boundary Groups
2) If primitive label is not L, then check possible primitive inferences from its seed aspect
seed aspect
L?
3) If that fails, choose a face f in F. For each of its instantiable aspects a that cover all faces in F, see if a maps to a primitive with label L.
aspect a
face f

FIG. 13. Expected object recognition.

the process. Only when all primitive coverings are exhausted do we generate a new aspect covering.

4.2. Expected Object Recognition

In the domain of expected object recognition, the image is searched for one or more instances of a particular object. In this case, there is no need for a complicated indexing step since we know what object the image features will be matched to. Instead, we are faced with the question: What features of the object do we search for in the image? Our approach is to start with a primitive covering and then constrain further primitive and aspect covering generation by exploiting knowledge of the object.

Figure 13 illustrates our approach to expected object recognition. The first step is the generation of the first primitive covering given the first aspect covering; this represents the most likely interpretation of the scene in terms of recovered shape. Next, as in the case of our approach to unexpected object recognition, we extract the strongly connected components. For each strongly connected component, the indexing step returns a list of model candidates that contain the component. Since we are looking for a particular object, we can discard all candidates but the object we are searching for (if it exists as a candidate). As before, we grow the correspondence between image and model primitives. However, it is during this last step that our approach differs.

In the unexpected object recognition algorithm, we attempt to completely recognize the entire primitive covering. Only when recognition fails do we generate another primitive covering. Furthermore, only when all primitive coverings are exhausted do we generate another aspect covering. Our expected object recognition approach attempts to integrate the three processes based on knowledge of which object part we are searching for.

During the correspondence growing step, some primitive in the primitive covering is checked to see if it matches some primitive in the model. If not, the unexpected object recognition approach does not include that primitive in the correspondence, i.e., growth of the correspondence is discontinued through that primitive. However, there may be an alternative primitive interpretation of that primitive's seed aspect that matches the expected model primitive. Recall that from the aspect covering, each aspect was used to infer a list of primitives in decreasing order of probability. By checking the various primitive hypotheses (in which the current primitive is included), we may find that the primitive label we are searching for is among the possible primitive interpretations of the seed aspect. If so, we choose the alternative interpretation and continue the correspondence. If not, we can probe deeper for the correct interpretation.

Let us assume, for example, that during our correspondence growing step, we examine a particular primitive P for a particular label L. Upon examination, we find that not only does primitive P not have label L, but there is no mapping from P's seed aspect SA to a primitive with label L. Let the set of faces belonging to seed aspect SA be f. The search for the correct primitive interpretation proceeds according to the following steps:

1. Choose one of the faces in f; call it F.

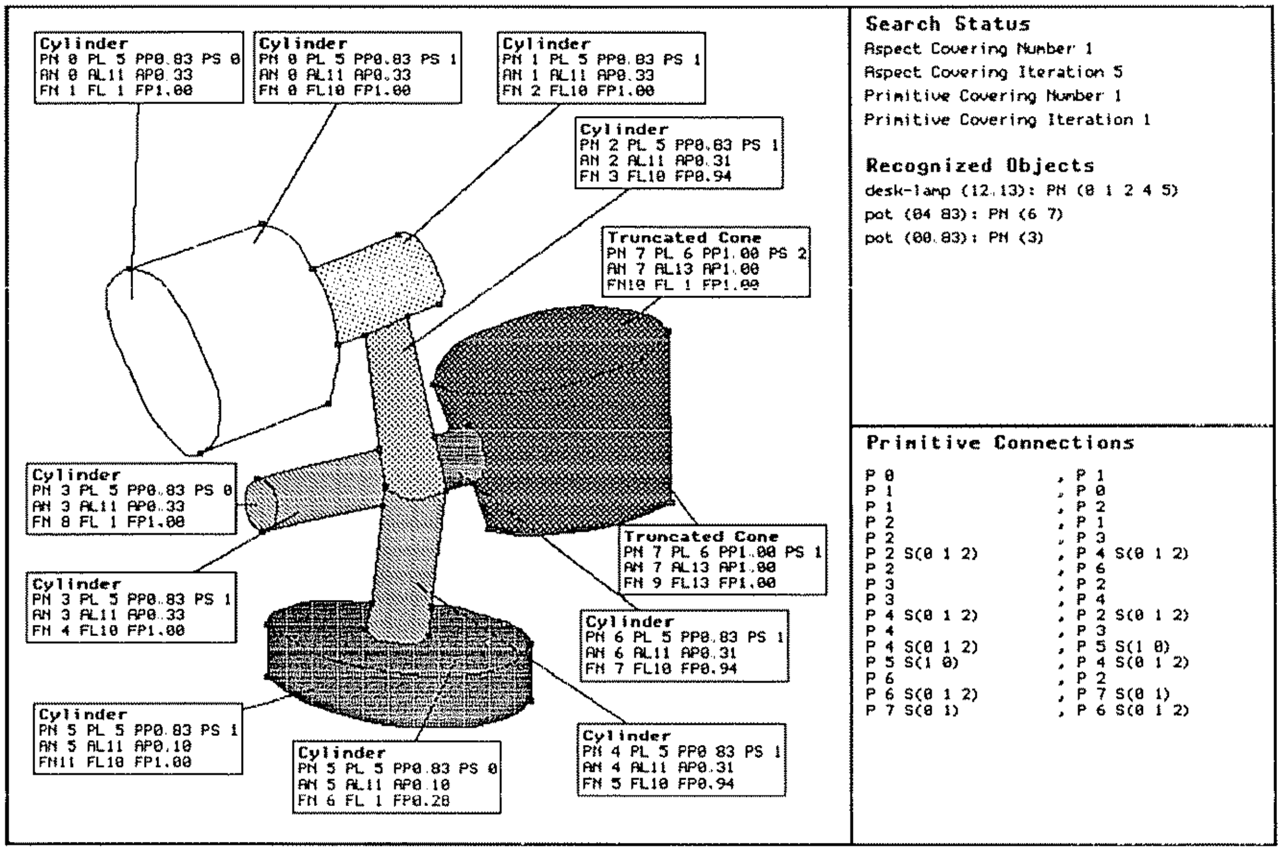
2. Recall that in the face graph, we generated a set of aspect labels for each face. Let the set of aspect labels for face F be a.

3. For each aspect label A in a do

   (a) If A in instantiable and A covers all the faces in f, then from the table mapping aspects to primitives, check the mapping between A and L.

   (b) If the probability of the mapping is nonzero, then we have found the correct primitive interpretation.

The above technique searches for the correct primitive interpretation from among all possible primitive interpretations of the faces covered by P. It implicitly assumes that although a particular aspect label in the aspect covering may be incorrect, the first enumerated (i.e., most



```
Cylinder
PN 0 PL 5 PP0.83 PS 0
AN 0 AL11 AP0.33
FN 1 FL 1 FP1.00

Cylinder
PN 0 PL 5 PP0.83 PS 1
AN 0 AL11 AP0.33
FN 0 FL10 FP1.00

Cylinder
PN 1 PL 5 PP0.83 PS 1
AN 1 AL11 AP0.33
FN 2 FL10 FP1.00

Cylinder
PN 2 PL 5 PP0.83 PS 1
AN 2 AL11 AP0.31
FN 3 FL10 FP0.94

Truncated Cone
PN 7 PL 6 PP1.00 PS 2
AN 7 AL13 AP1.00
FN10 FL 1 FP1.00

Cylinder
PN 3 PL 5 PP0.83 PS 0
AN 3 AL11 AP0.33
FN 8 FL 1 FP1.00

Cylinder
PN 3 PL 5 PP0.83 PS 1
AN 3 AL11 AP0.33
FN 4 FL10 FP1.00

Truncated Cone
PN 7 PL 6 PP1.00 PS 1
AN 7 AL13 AP1.00
FN 9 FL13 FP1.00

Cylinder
PN 6 PL 5 PP0.83 PS 1
AN 6 AL11 AP0.31
FN 7 FL10 FP0.94

Cylinder
PN 5 PL 5 PP0.83 PS 1
AN 5 AL11 AP0.10
FN11 FL10 FP1.00

Cylinder
PN 5 PL 5 PP0.83 PS 0
AN 5 AL11 AP0.10
FN 6 FL 1 FP0.28

Cylinder
PN 4 PL 5 PP0 83 PS 1
AN 4 AL11 AP0.31
FN 5 FL10 FP0.94
```

```
Search Status
Aspect Covering Number 1
Aspect Covering Iteration 5
Primitive Covering Number 1
Primitive Covering Iteration 1

Recognized Objects
desk-lamp (12.13): PN (0 1 2 4 5)
pot (04 83): PN (6 7)
pot (00.83): PN (3)
```

```
Primitive Connections

P 0                   , P 1
P 1                   , P 0
P 1                   , P 2
P 2                   , P 1
P 2                   , P 3
P 2 S(0 1 2)          , P 4 S(0 1 2)
P 2                   , P 6
P 3                   , P 2
P 3                   , P 4
P 4 S(0 1 2)          , P 2 S(0 1 2)
P 4                   , P 3
P 4 S(0 1 2)          , P 5 S(1 0)
P 5 S(1 0)            , P 4 S(0 1 2)
P 6                   , P 2
P 6 S(0 1 2)          , P 7 S(0 1)
P 7 S(0 1)            , P 6 S(0 1 2)
```

FIG. 14. Correct first interpretation of an occluded scene.

likely) aspect covering represents the correct partition of image faces into aspects. Although this technique has been applied to expected or top-down object recognition, there is no reason that it could not be applied to the verification step in our approach to unexpected recognition. Although systematic generation of all primitive coverings is necessary to guarantee a correct interpretation, it is likely that a constrained search for the correct primitive would make the unexpected object recognition algorithm more efficient.

5. RESULTS

We have built a system to demonstrate our approach to 3-D object recognition. The system is called OPTICA (*O*bject recognition using *P*robabilistic *T*hree-dimensional *I*nterpretation of *C*omponent *A*spects), and has been implemented in COMMON LISP on a Sun 4/330™ workstation. In the following sections, we demonstrate the approach on both synthetic and real imagery.

5.1. Synthetic Image Data

In the first example, we apply the recovery and recognition processes to the manually segmented contour image presented in Fig. 14. Each face in Fig. 14 is described by a small box containing some mnemonics. The mnemonics PN, PL, PP, and PS, refer to the primitive number (simply an enumeration of the primitives in the covering), primitive label (see Fig. 4), primitive probability, and primitive attachment surface (see [18]), respectively, of the primitive that includes that face. The mnemonics AN, AL, and AP refer to the aspect number (an enumeration), aspect label (see [18]), and aspect probability, respectively, of the aspect from which the primitive was inferred. The mnemonics FN, FL, and FP refer to the face number (an enumeration), face label (see [18]), and face probability, respectively, of the face from which the aspect was inferred.

The smaller box to the upper right indicates the aspect covering iteration and primitive covering iteration (given the aspect covering). In this case, the first aspect and primitive coverings represent the correct interpretation of the scene. In addition, this box lists all objects currently (at the above iterations) identified in the image, including their corresponding primitive numbers (PN). Note that the handle of the cooking pot has been interpreted as two distinct primitives; no collinearity grouping is performed at the primitive level.

The smaller box to the lower right indicates the primitive connections by primitive number (PN); if two primitives are strongly connected, a list of probable attachment surfaces appears in parentheses next to the primitive number. This list is not exclusive, but rather a list of likely candidates.

5.2. Real Image Data

The first step in applying our approach to real images is the segmentation of an input image into homogeneous regions. To extract regions, we first apply the Canny edge detector [13] to the input image to detect the projected surface discontinuities of the object. From the resulting edge pixels, our goal is to locate cycles of edge pixels that bound regions. However, since there may be small gaps in the edge pixel map that break cycles, we dilate the image to fill the gaps. The result is then skeletonized to yield an image containing single pixel width contours.

From the image of contours, we apply a connected components algorithm to extract a set of contours, each beginning and ending at a junction of three or more contours; all other contours are discarded. The next step is to partition the contours at significant curvature discontinuities. We first apply Ramer's algorithm [56] to produce an initial set of breakpoints. Although effective for the partitioning of straight lines, the algorithm overpartitions curves. However, the resulting partition points are a superset of the correct partition points (Liao [37]). To remove the false partition points, we fit circular arcs to the left and right neighborhoods of each potential breakpoint, and discard the breakpoint if the angle between the tangents to the two circles at the breakpoint is near 180°. The resulting set of contours are classified as lines or curves depending on how well a line can be fitted to them.

From the set of partitioned contours, we build the face graph using the algorithm described in [18]. For each face in the face graph, our next task is to represent the face by a graph in which nodes represent contours and arcs represent certain nonaccidental relations among the contours. For a given face, adjacent lines or adjacent curves which meet at a junction are merged (according to the criteria used to check initial partition points) if they are collinear or curvilinear. Any curves bounding the face are further classified as concave or convex by checking the angle between the lines joining the midpoint to the two ends of the curve. Nonadjacent lines are labeled parallel if the angle between them is small, and symmetric if they are opposite and nonparallel. Nonadjacent curves are labeled parallel if one is concave, the other is convex, and they face in similar directions, where the direction of a curve is defined by the vector whose head is at the midpoint of the line joining the two ends of the curve and whose tail is that point on the curve whose perpendicular distance to the line is maximum. A similar test is used for curve symmetry if both curves are concave or convex. If, for parallel or symmetric curves, the radii of the circles fitted to the curves are significantly different, the relative size of the curves is noted.

In the following two examples, we apply the approach to real images of objects taken against a black background. The time required to generate the first interpretation of the scene was approximately 7 s, excluding time required to produce the skeletonized edge image. Generation of subsequent interpretations required an average of only 0.2 s each, indicating that much of the execution time was spent in the connected components and contour segmentation and fitting routines.

The entire process was first applied to the 256 × 256 image of a table lamp shown in Fig. 15; the resulting primitive covering is presented in Fig. 16 (partitioned contours extracted from the skeleton image have been enumerated). Despite the underpartitioning of the contours (contour 11), the first aspect and primitive coverings represent the correct interpretation of the scene.

In the second example, the entire process was applied to the 256 × 256 image of a padlock shown in Fig. 17. Figure 18 shows the partitioned contours extracted from the skeleton image. Faces FN2, FN3, and FN4 were correctly grouped and interpreted as the block primitive (PL1). However, faces FN0 and FN1 were grouped and interpreted as the truncated ellipsoid primitive (PL8). In face FN1, the contours have been overpartitioned; therefore, the face could not be matched to the faces in the aspect hierarchy. Furthermore, due to noise, contour 2 was classified as a line. The strongest face inference, representing the projection of the body of a truncated ellipsoid, was generated by the boundary group consisting of contours 0 and 4. When grouped with the elliptical face FN0, the resulting aspect was mapped to the trun-

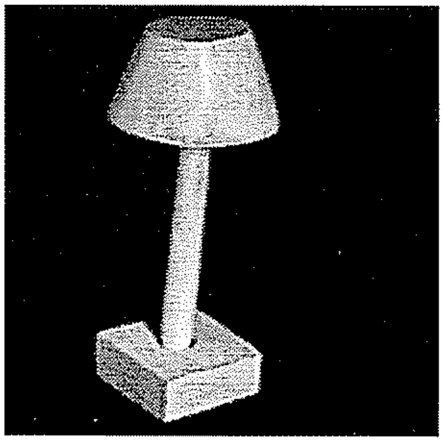
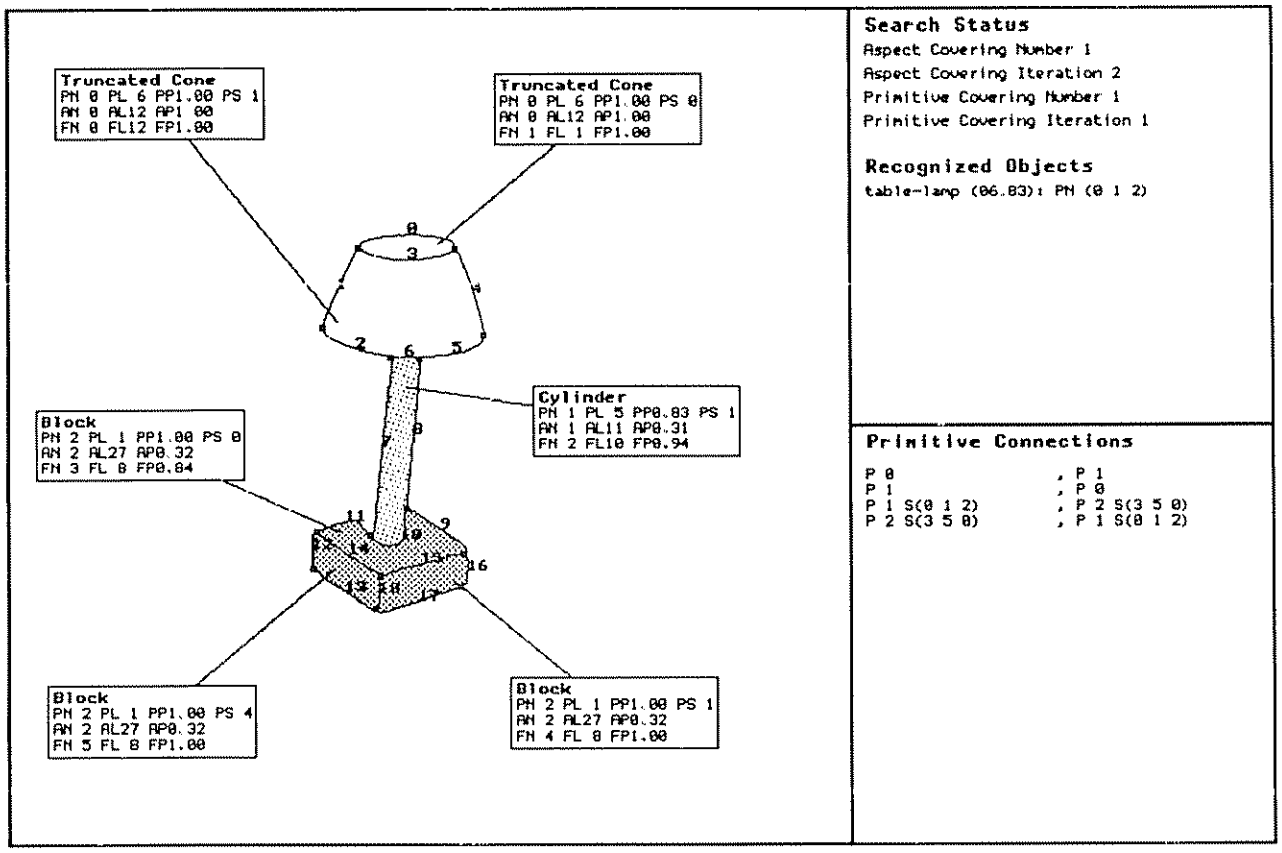


FIG. 15. Image of a table lamp (256 × 256)



```
Truncated Cone
PN 0 PL 6 PP1.00 PS 1
AN 0 AL12 AP1 00
FN 0 FL12 FP1.00
```

```
Truncated Cone
PN 0 PL 6 PP1.00 PS 0
AN 0 AL12 AP1.00
FN 1 FL 1 FP1.00
```

```
Cylinder
PN 1 PL 5 PP0.83 PS 1
AN 1 AL11 AP0.31
FN 2 FL10 FP0.94
```

```
Block
PN 2 PL 1 PP1.00 PS 0
AN 2 AL27 AP0.32
FN 3 FL 8 FP0.84
```

```
Block
PN 2 PL 1 PP1.00 PS 4
AN 2 AL27 AP0.32
FN 5 FL 8 FP1.00
```

```
Block
PN 2 PL 1 PP1.00 PS 1
AN 2 AL27 AP0.32
FN 4 FL 8 FP1.00
```

```
Search Status
Aspect Covering Number 1
Aspect Covering Iteration 2
Primitive Covering Number 1
Primitive Covering Iteration 1

Recognized Objects
table-lamp (06.83): PN (0 1 2)
```

```
Primitive Connections

P 0                 , P 1
P 1                 , P 0
P 1 S(0 1 2)        , P 2 S(3 5 0)
P 2 S(3 5 0)        , P 1 S(0 1 2)
```

FIG 16 Correct first interpretation of a table lamp.

cated ellipsoid. The second strongest (correct) face inference, representing the projection of the body of a bent cylinder, was also generated by contours 0 and 4. The second aspect covering from which the correct primitive covering was inferred is shown in Fig. 19.

If we apply the expected object recognition algorithm to the image in Fig. 17, we can constrain the search for the arm of the lock. In this case, the lock body is used as the database index, with verification searching the aspect hierarchy for a bent cylinder interpretation of the remaining faces. The successful first interpretation is shown in Fig. 20.

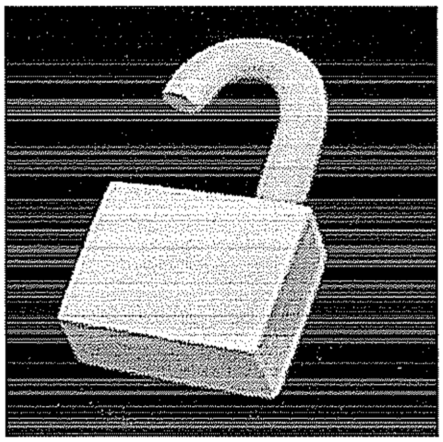
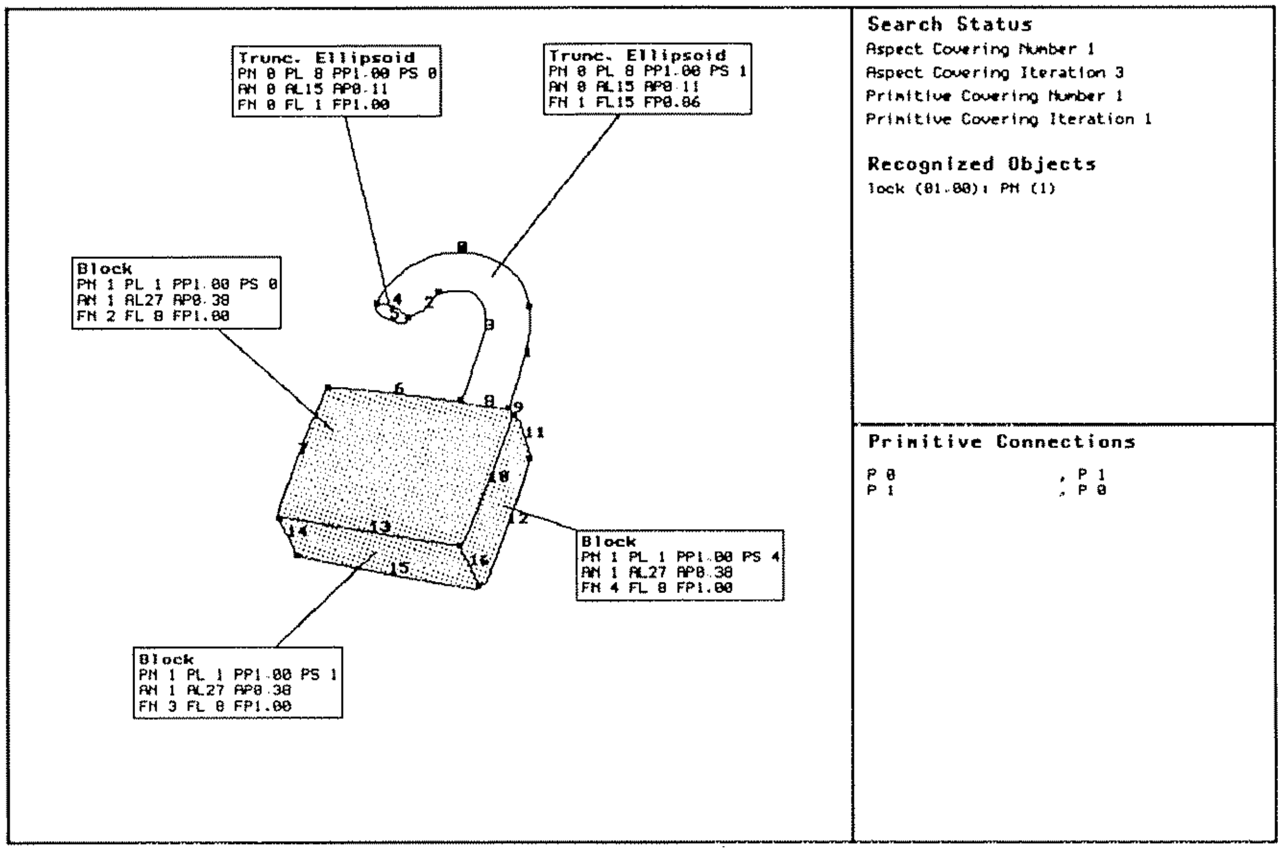


FIG 17 Image of a lock (256 × 256)

These examples illustrate the results of applying our shape recovery and recognition algorithm to real images. Despite the use of simple, standard techniques for image segmentation and contour grouping, which resulted in several segmentation errors, the algorithm was able to produce a correct interpretation of these scenes. With more effective techniques for region extraction (e.g., Meer *et al.* [42], Besl and Jain [5]), contour partitioning (e.g., Saint-Marc and Medioni [59], Fischler and Bolles [22]), perceptual grouping (e.g., Mohan and Nevatia [45], Lowe [38]), and the model-based correction of segmentation errors discussed in [18], we expect the system's performance to improve significantly.

6. RELATED WORK

In this section, we review a number of related approaches to object recognition, and compare them to our own approach. Since the field of 3-D object recognition is



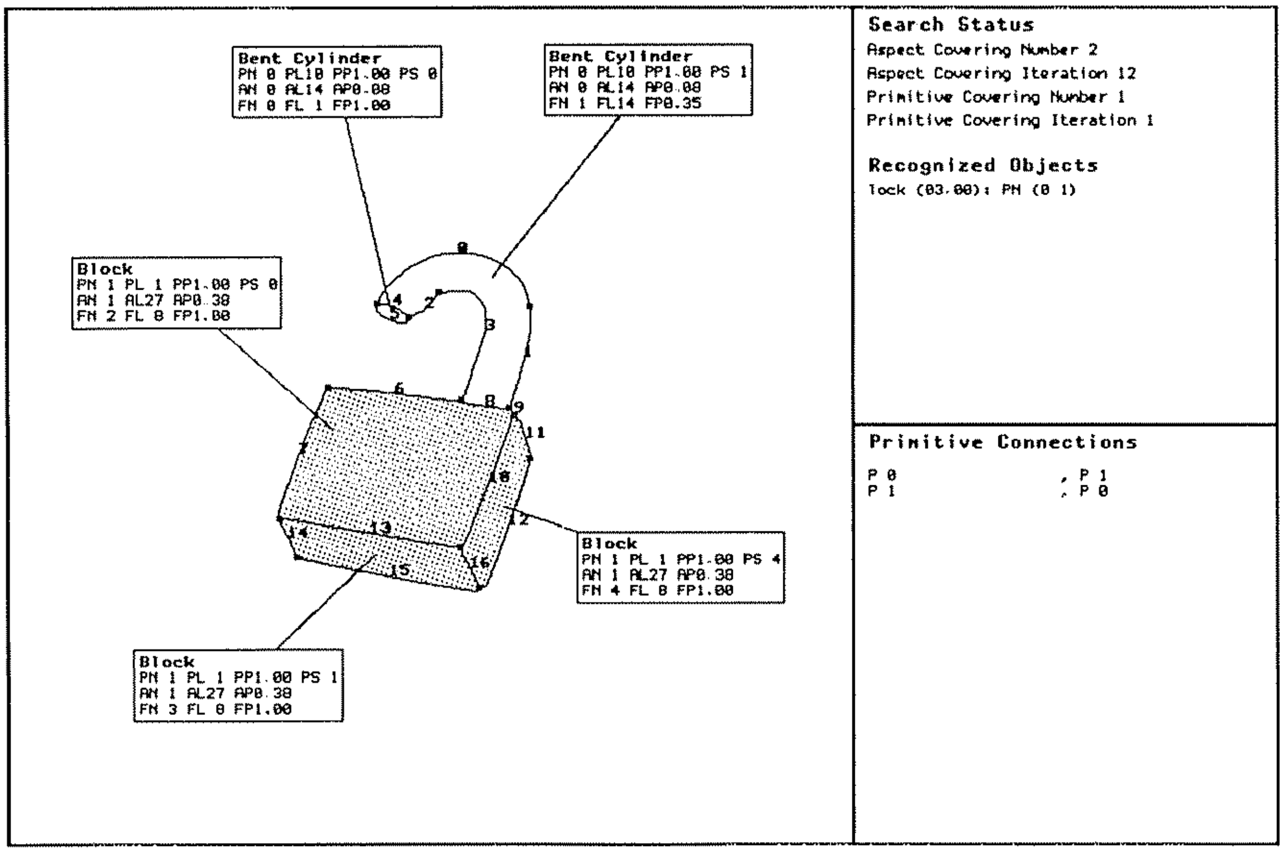
Search Status
Aspect Covering Number 1
Aspect Covering Iteration 3
Primitive Covering Number 1
Primitive Covering Iteration 1

Recognized Objects
lock (01.00): PN (1)

Primitive Connections

```
P 0                    , P 1
P 1                    , P 0
```

FIG. 18 Incorrect first interpretation of the lock



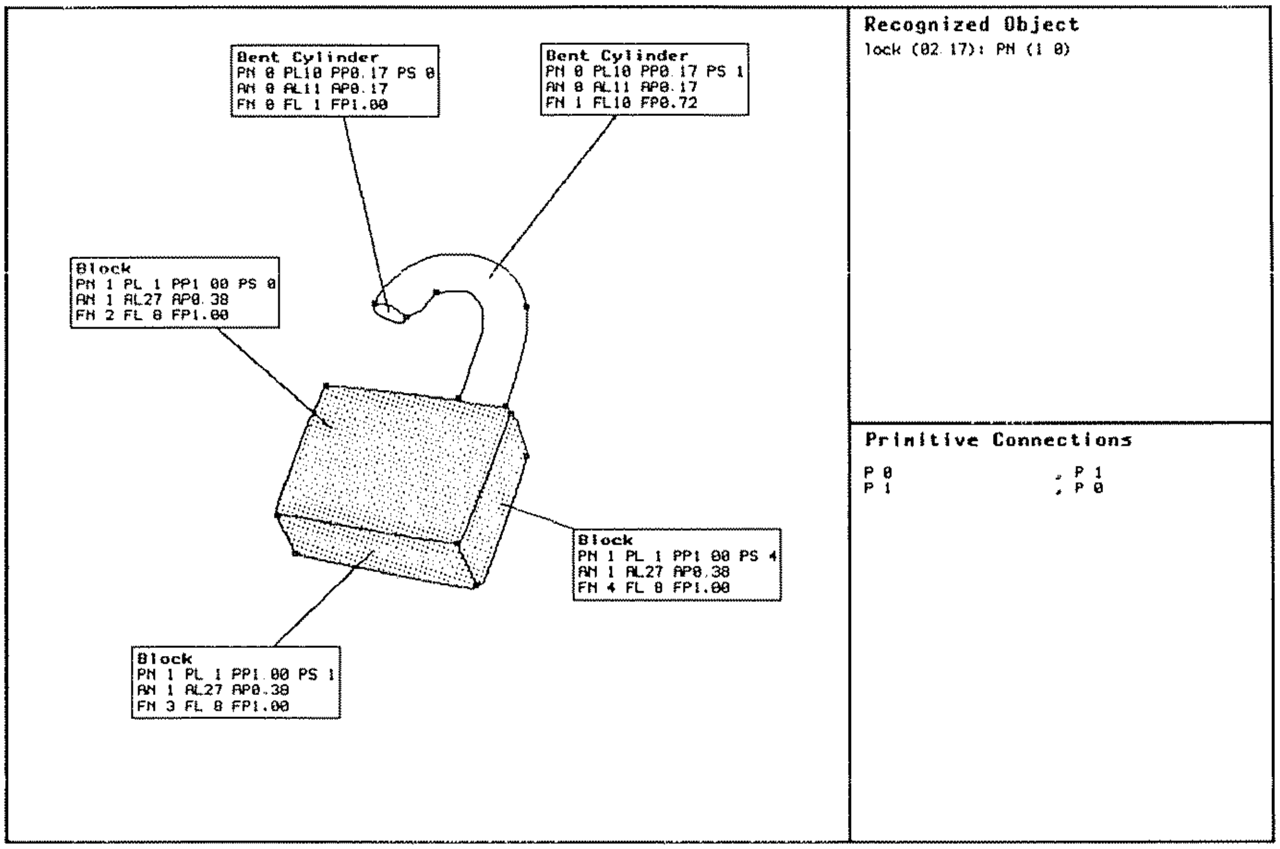
Search Status
Aspect Covering Number 2
Aspect Covering Iteration 12
Primitive Covering Number 1
Primitive Covering Iteration 1

Recognized Objects
lock (03.00): PN (0 1)

Primitive Connections

```
P 0                    , P 1
P 1                    , P 0
```

FIG. 19 Correct interpretation of the lock



```
Bent Cylinder
PN 0 PL10 PP0.17 PS 0
AN 0 AL11 AP0.17
FN 0 FL 1 FP1.00

Bent Cylinder
PN 0 PL10 PP0.17 PS 1
AN 0 AL11 AP0.17
FN 1 FL10 FP0.72

Block
PN 1 PL 1 PP1 00 PS 0
AN 1 AL27 AP0.38
FN 2 FL 0 FP1.00

Block
PN 1 PL 1 PP1 00 PS 4
AN 1 AL27 AP0.38
FN 4 FL 0 FP1.00

Block
PN 1 PL 1 PP1.00 PS 1
AN 1 AL27 AP0.38
FN 3 FL 0 FP1.00

Recognized Object
lock (02 17): PN (1 0)

Primitive Connections
P 0                    , P 1
P 1                    , P 0
```

FIG. 20. Searching the image for a lock.

so extensive, we have chosen to focus only on those techniques which address the problem of 3-D object recognition from a single 2-D image. We first review related work in object-centered recognition, followed by related work in viewer-centered recognition. The review is by no means complete, and is meant only to contrast our approach with some of the more established techniques; more comprehensive reviews can be found in Binford [8], Besl and Jain [4], and Chin and Dyer [15].

7. OBJECT-CENTERED RECOGNITION

The seminal work of Roberts [57] addresses the task of recognizing polyhedral objects composed of cubes, wedges, and hexagonal prisms. Given four points in the image and four non-coplanar points in the model, the system solves for eight transformation parameters. The computed transformation is verified by projecting the model points back into the image and checking that they do not lie outside the object's boundary. The system's indexing features are limited to vertices and their surrounding *unoccluded* polygons; no partial information is extracted from occluded polygons. In addition, the quantitative nature of the verification step restricts recognition to rigid objects whose exact geometry must be specified.

In Brooks' ACRONYM system [10], objects are represented as constructions of generalized cylinders. Recognition of a particular model object consists of predicting the projected appearance in the image of the object's components; partial constraints on the 3-D parts of the model are mapped to constraints on the 2-D projections of the parts (e.g., ribbons). The image contours are then examined, subject to these constraints, and matched contours are used to further constrain the sizes and orientations of the 3-D parts. Local matches are grouped into complete objects during the interpretation phase, further constraining the parameters of the object. Unfortunately, the top-down nature of ACRONYM makes it unsuitable for unexpected object recognition; ACRONYM can only confirm or deny the existence in the image of a user-specified object.

In contrast to ACRONYM's top-down approach, Lowe's SCERPO system [38] takes a more bottom-up approach to the recognition of polyhedral objects. Image contours are first grouped according to perceptual organization rules, including parallelism, symmetry, and collinearity, providing more powerful indexing features than simple lines or points. From these groupings, simple 3-D inferences are made about the 3-D contours comprising the object. The 3-D inferences are matched against manually identified instances of the properties in the model. Assuming a perspective projection imaging model, Lowe requires up to six pairs of image and model points to solve (using Newton–Raphson iteration) for the orientation and position of the model. Back-projected features are then used to verify the object and further constrain its position and orientation. Although SCERPO could be applied to unexpected object recognition, the complexity of polyhedral models and the simplicity of the indexing features would result in considerable indexing ambiguity. In addition, like Roberts' work, SCERPO's reliance on verification of simple image features restricts its recognition domain to objects whose exact geometry is known.

Goad [27] presents a technique for locating a known 3-D polyhedral object in a 2-D image. Like ACRONYM, the matching of model features to image features is used to constrain the viewpoint and, hence, the orientation and position of other image features. However, unlike ACRONYM, the technique exploits off-line precomputation to save time during on-line recognition. For each possible position and orientation of each model feature, Goad's approach first precomputes the relative position in the image of every other model feature. Given an initial pairing of a model edge and an image edge, a precompiled search tree chooses another edge in the model and predicts its location in the image relative to the first edge. As new edges in the image are paired with model edges, their position and orientation are used in a back-projection operation to constrain the viewpoint.

Another approach which exploits off-line precomputation to speed up on-line recognition is the geometric hashing algorithm proposed by Lamdan, *et al.* [36] for the recognition of 3-D planar objects. In their approach, three noncollinear model points are used to define a 2-D basis. The coordinates of all other model points are then represented with respect to this basis and are invariant to any affine transformation. During precomputation, the process is repeated for each coplanar noncollinear triplet of model points (e.g., corners or curvature extrema). The resulting coordinates of a model point define an index into a hash table whose location contains the identity of the model(s) and the three points used to define the basis. During on-line recognition, the same process is applied. A triple of image points is selected to form an affine basis in which all other image points are represented. Each image point then votes for a model basis; if enough image points vote for the same model basis, the object is recognized. Thompson and Mundy [71] present a similar approach to the recognition of polyhedral objects that is also based on precomputation and voting techniques. Unfortunately, like the Lamdan *et al.* approach, the precomputed tables are not invariant to minor changes in the shape of the model.

Huttenlocher and Ullman [31–33] describe another approach to the recognition of polyhedral objects under a weak perspective imaging model. The approach is divided into two phases: (1) finding possible transformations from a model to the image; and (2) verifying those transformations by aligning the model with the image. In the first phase, the system segments image contours into positive, negative, and zero curvature portions (separated by zeros of curvature which are preserved under projection). The indexing features represent sets of points extracted from one or more connected contour portions. Three features are required to solve for the transformation, and consist of either three points or a single point and an orientation. During the alignment phase, each transformation is verified by comparing the transformed projected model contours with the image contours. If a significant portion of a model's contours are matched, the transformation is accepted.

A number of researchers have moved away from the recognition of exact geometrical models and towards the recognition of coarse qualitative models. Mulgaonkar *et al.* [46] describe a recognition system based on a set of generalized relational blob models including sticks, plates, and blobs which, in three-space, are modeled as straight lines, circular disks, and spheres, respectively. From the 2-D silhouette of an object, a graph-theoretic clustering technique yields a set of convex polygonal parts; internal image contours are ignored. The projected parts are then compared to 3-D part instances in the model database, subject to quantitative geometrical and relational constraints. These and other constraints are propagated to adjacent parts to achieve a consistent labeling of the scene. Like ACRONYM, the system is primarily top-down, starting with a model and matching image structures to model-based predictions.

Finally, Bergevin and Levine [3] have developed a system, called PARVO, to recognize objects composed of Biederman's geons [6]. Their approach to grouping lines consists of pairing segmentation points resulting from concave slope discontinuities lying on the silhouette boundary of the object. From this pairing, line groups are formed, and internal contours are later assigned to the line groups on a second pass. Given a set of cross-sectional and body faces representing a line group, a discrimination tree is traversed to determine the symbolic values of the four geon attributes. The geon connections are then specified, along with relative part sizes and qualitative part aspect ratios. The resulting description is then matched to the database of object models. The technique assumes that the segmentation points can be paired, and assumes that a unique geon label can be assigned to each group of lines constituting a part. However, in the presence of occlusion or degenerate viewpoint, these assumptions may be too strong. The major limitation of the approach is that it is dependent on the choice of geons as modeling primitives.

8. VIEWER-CENTERED RECOGNITION

The viewer-centered representation of a 3-D object using a set of aspects has been applied to polygonal object recognition by Chakravarty and Freeman [14]. In their approach, each characteristic view, or aspect, is represented by its visible junctions and connecting lines. Furthermore, the set of views is organized hierarchically according to the number of labeled vertices of each type (e.g., "Y" or "T"). During the recognition process, lines and junctions are extracted from the image and labeled. Next, a projection silhouette is extracted from the line labels and used to select a set of candidate views from the database which have a similar silhouette. The silhouette junctions are then compared to the candidate views to further prune possible matches. Correlated junctions are then used to compute a transformation between the 3-D object and the image, followed by a verification procedure based on the computed transformation. This approach relies on accurate junction labeling which is difficult in the presence of noise. In addition, for recognition from large databases, the number of views may be prohibitively large.

Burns and Kitchen [12] describe an approach to polygonal object recognition from large databases based on a precompiled feature hierarchy, called a prediction hierarchy, that guides a bottom-up interpretation of the scene. Compiling the prediction hierarchy consists of starting with a set of simple image features, or predictions (parallel or coincident lines), and iteratively searching across all views of all objects for commonly occurring groups or specializations of simpler predictions. By storing all predictions in a visibility map, the compilation can find predictions that often occur together in the same projection. The predictions at the top level of the hierarchy are predictions that define the views of the objects. During the recognition stage, the system builds up correspondences between image and model segments through a simultaneous guided search of the prediction hierarchy and the image structures. Given a unique correspondence between a set of image lines and some view of an object, the system uses the view as an initial pose estimate for Lowe's iterative pose refinement algorithm [38]. Because aspects describe entire objects, the size of the prediction hierarchy is proportional to the number of objects. Furthermore, since views represent entire objects, the addition of an object to the database requires a recompilation of the prediction hierarchy.

Swain [67] describes a technique similar to that of Burns and Kitchen [12] that precompiles a decision tree for recognizing a given view of a polyhedral object from a large database. The technique exploits the probability distributions for the objects' frequencies of occurrence, the objects' views, and the errors that occur in detecting image edges. Each node in the tree represents a state consisting of a collection of mappings between image and model (aspect) features, while a node's children represent the possible expanded mappings. Since, using a dynamic programming approach, constructing the optimal search tree is intractable for large databases, the technique chooses a path, i.e., image test, at a tree node based on the expected cost of recognition in the subtrees. To calculate the expected cost, an entropy measure is used, based on the probabilities of the nodes. As in the Burns and Kitchen approach, the technique can, at any time during the search, attempt to solve for the object's viewpoint. However, like the Burns and Kitchen approach, not only is the size of the search tree proportional to the size of the object database, but the search tree must be recompiled with the addition of new objects to the database.

Shapiro and Lu [61] present an approach to view class determination of a known object from an unknown view. The approach begins by using the PADL-2 solid modeling system to create a 3-D CSG object using rectangular polyhedra, spheres, cylinders, cones, wedges, and tori. Next, the resulting CAD model is mapped to a set of view classes, each represented by a relational pyramid structure. The levels of the pyramid include points (level 0), lines and curves (level 1), junctions (level 2), and combinations of level 2 features (level 3). From a relational pyramid built from an unknown view, the pyramid is mapped to an index (summarizing the information in the pyramid) which points to a list containing all view classes possessing the summarized structure. In an accumulator-based matching algorithm, evidence is added to each of the indexed view classes. The system provides both an exact matching technique in which only indexed views are voted for, and an inexact matching technique in which views at nearby indices are also voted for (using a Gaussian distribution voting scheme). Another technique for coping with missing or extra line segments in the image involves adding imperfect views to the database.

Finally, Ullman and Basri [72] describe an approach whereby a 3-D object is represented by a small number of 2-D images of the object. They show that under orthographic projection, an object with sharp edges undergoing rigid transformation and scaling in 3-D space can be represented as a linear combination of four images of the object. Objects with smooth boundaries, undergoing similar transformation and projection, can be represented as a linear combination of six images of the object. A number of methods are presented for determining the alignment coefficients, two of which require correspondence between image features and model features. The major advantage of the approach is that unlike Chakravarty and Freeman [14], Burns and Kitchen [12], Swain [67], or Shapiro and Lu [61], a 3-D model is not required to generate the views of an object; a small set of images is sufficient. Unfortunately, each topologically distinct view (aspect) of the object requires a distinct set of images. For large databases containing complex objects, the resulting number of images could become prohibitive. Achieving four- or six-point correspondence between image and 3-D model features is computationally complex; a multitude of images representing the various aspects of a model further compounds the problem.

8.1. DISCUSSION

Most of the object-centered (or viewpoint-independent) recognition systems discussed in this section are based on simple indexing features such as points, lines, zeroes of curvature, or curvature extrema. The resulting systems must accurately solve for viewpoint in order to support the geometrical verification of image features (either through back-projection or voting techniques). The resulting models fail to capture an object's intuitive part structure, and are sensitive to minor changes in object shape. In most cases, the recognition domain consists of a single object. While there are exceptions, e.g., Brooks' ACRONYM [10] and the approach of Mulgaonkar *et al.* [46], which attempt a volumetric part-based modeling of objects, these systems are essentially top-down, searching the image for a particular model. Bergevin and Levine's PARVO system [3] attempts to recover volumetric primitives for large recognition domains, but makes strong assumptions about primitive type, primitive orientation, and part segmentation.

All the viewer-centered (or viewpoint-dependent) recognition systems discussed in this section model entire objects using a set of views, which can lead to a prohibitively large number of views for a large object database. Some approaches, e.g., Burns and Kitchen [12] and Swain [67], have attempted to streamline the search by precompiling search strategies that more efficiently match image features to model aspects. Unfortunately, the size of search trees is still proportional to the number of views in the database. Furthermore, the addition of a new object requires the recompilation of the search strategies. Although many of the approaches, e.g., Burns and Kitchen [12], Swain [67], and Shapiro and Lu [61], model aspects with a hierarchical representation, they do not capture the object's intuitive parts. Consequently, as an object's parts *articulate* or change shape, additional aspects may be required.

9. CONCLUSIONS

The inefficiency of most 3-D object recognition systems is reflected in the relatively small number of objects in their database (on the order of 10); in many cases, algorithms are demonstrated on a single object model. The major problem is that these systems terminate the bottom-up primitive extraction phase very early, resulting in simple primitives such as lines, corners, and inflections. These primitives do not provide very discriminating indices into a large database, so that normally there are a large number of hypothesized matches. Consequently, the burden of recognition falls on top-down verification, which for simple geometric image features requires both accurate estimates of the object's pose and prior knowledge of the object's geometry.

We instead index into the model database with more discriminating primitives, i.e., ones that do not require precise knowledge of model geometry or accurate estimates of pose. An appropriate choice for higher-order indexing primitives is the class of volumetric primitives which capture the intuitive notion of an object's parts. In this approach, object models are constructed from object-centered 3-D volumetric primitives. The primitives, in turn, are represented in the image by a set of viewer-centered aspects.

Unlike typical aspect-based recognition systems which model each entire object in a database using a set of aspects, we use aspects to model a finite number of volumetric parts used to construct the objects. The size of the resulting aspect set is fixed and, more important, *independent* of the contents of the object database. To accommodate the representation of occluded aspects arising from occluded primitives, we introduce a hierarchical aspect structure, called the *aspect hierarchy*, based on the faces appearing in the aspect set. The ambiguous mappings between levels of the aspect hierarchy are captured by a set of conditional probabilities resulting from a statistical analysis of the aspects. The aspect hierarchy is precomputed *once* off-line and remains fixed while objects are added or removed from the database.

We have demonstrated our approach using a vocabulary of primitives resembling Biederman's geons [6]; however, our approach is *not* dependent on geons as object modeling primitives. Although any selection of volumetric primitives can be mapped to a set of aspects, our hierarchical aspect representation is particularly appropriate for primitives with distinct surfaces, i.e., primitives whose aspects contain distinct faces. The use of a face-based aspect hierarchy is the backbone of our approach, allowing us to obtain probabilistic rules for inferring more complex features from less complex features, and for merging oversegmented contours and regions [18]. Although the individual features represented in our aspect hierarchy may change when using other types of volumetric primitives, the concepts of representing a set of 3-D volumetric primitives as a probabilistic hierarchy of image features, and exploiting these probabilities during recovery and recognition are applicable to any object representation that models objects using volumetric parts.

The cost of extracting more complex primitives from the image is the difficulty of grouping less complex features into more complex features; the number of possible groupings is enormous. Our recovery algorithm uses a statistical analysis of the aspects (explicitly represented in the aspect hierarchy) to rank-order the possible groupings. The result is a heuristic that has been demonstrated to quickly arrive at the correct interpretation. Note, however, that our approach will, if need be, enumerate *all* possible interpretations (or groupings); the correct interpretation of any scene, no matter how ambiguous or unlikely, will eventually be generated.

We have presented a database indexing scheme that maps a group of recovered primitives and their connections to a hash table location whose contents specify those models containing a similar primitive structure. The candidate hypotheses are then topologically verified and ranked based on the strengths of the hypothesized primitives. We show that for both the problems of unexpected and expected object recognition, the same recognition engine can be used.

The ability to recover a set of volumetric primitives without a priori knowledge of the object database not only supports the recognition of objects from large databases, but also supports the automatic acquisition of object models. In addition, the bottom-up recovery of an object's intuitive parts supports the recognition of objects based on functionality. You can reason about what function a collection of volumetric shapes might perform, whereas little sense can be made from a collection of points.

10. ACKNOWLEDGMENTS

The authors thank Larry Davis, Göran Olofsson, Lars Olsson, Jan-Olof Eklundh, and especially Suzanne Stevenson for many insightful discussions, and for reviewing earlier drafts of this work.

REFERENCES

1. G. Agin and T. Binford, Computer description of curved objects, *IEEE Trans. Comput.* **C-25** (4), 1976, 439–449.
2. H. Asada and M. Brady, The curvature primal sketch, *IEEE Trans. Pattern Anal. Mach. Intelligence* **8** (1), 1986, 2–14.
3. R. Bergevin and M. Levine, Generic object recognition: Building coarse 3D descriptions from line drawings in *Proceedings, IEEE Workshop on Interpretation of 3D Scenes, Austin, TX, November 1989*, pp. 68–74.
4. P. Besl and R. Jain, Three-dimensional object recognition, *ACM Comput. Surv.* **17** (1), 1985, 75–145.
5. P. Besl and R. Jain, Segmentation through variable-order surface fitting, *IEEE TRans. Pattern Anal. Mach. Intelligence* **10** (2), 1988, 167–192.
6. I. Biederman, Human image understanding: Recent research and a theory, *Comput. Vision. Graphics Image Process.* **32**, 1985, 29–73.
7. T. Binford, Visual perception by computer, in *Proceedings, IEEE Conference on Systems and Control, Miami, FL, 1971*.
8. T. Binford, Survey of model-based image analysis systems, *Int. J. Robot. Res.* **1** (1), 1982, 18–64.
9. R. Bolles and P. Horaud, 3DPO: A three-dimensional part orientation system, *Int. J. Robot. Res.* **5** (3), 1986, 3–26.
10. R. Brooks, Model-based 3-D interpretations of 2-D images, *IEEE Trans. Pattern Anal. Mach. Intelligence* **5** (2), 1983, 140–150.
11. T. Breuel, Adaptive model based indexing, in *Proceedings, DARPA Image Understanding Workshop, 1989*, pp. 805–814.
12. J. Burns and L. Kitchen, Recognition in 2D images of 3D objects from large model bases using prediction hierarchies, in *Proceedings, International Joint Conference on Artificial Intelligence, Milan, Italy, 1987*, pp. 763–766.
13. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intelligence* **8** (6), 1986, 679–698.
14. I. Chakravarty and H. Freeman, Characteristic views as a basis for three-dimensional object recognition, in *Proceedings, SPIE Conference on Robot Vision, Arlington, VA, May 1982*, pp. 37–45.
15. R. Chin and C. Dyer, Model-based recognition in robot vision, *ACM Comput. Surv.* **18** (1), 1986, 67–108.
16. S. Dickinson, A. Pentland, and A. Rosenfeld, A representation for qualitative 3-D object recognition integrating object-centered and viewer-centered models, in *Vision: A Convergence of Disciplines* (K. N. Leibovic, Ed.), Springer-Verlag, New York, 1990.
17. S. Dickinson, The recovery and recognition of three-dimensional objects using part-based aspect matching, Technical Report CAR-TR-572, Center for Automation Research, University of Maryland, August, 1991.
18. S. Dickinson, A. Pentland, and A. Rosenfeld, 3-D shape recovery using distributed aspect matching, *IEEE Trans. Pattern Anal. Mach. Intelligence*, special issue on Interpretation of 3-D Scenes, Part II, **14** (2), Febrary, 1992.
19. D. Eggert and K. Bowyer, Computing the orthographic projection aspect graph of solids of revolution, *Pattern Recognition Lett.* **11**, 1990, 751–763.
20. G. Ettinger, Large hierarchical object recognition using libraries of parameterized model subparts, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, 1988*, pp. 32–41.
21. Deleted in proof.
22. M. Fischler and R. Bolles, Perceptual organization and curve partitioning, *IEEE Trans. Pattern Anal. Mach. Intelligence* **8** (1), 1986, 100–105.
23. P. Flynn and A. Jain, 3D object recognition using invariant feature indexing of interpretation tables, in *Proceedings, IEEE Workshop on Directions in Automated CAD-Based Vision, Maui, HI, June 1991*, pp. 115–123.
24. M. Gardiner, The superellipse: A curve that lies between the ellipse and the rectangle, *Sci. Am.* **213**, 1965, 222–234.
25. Z. Gigus and J. Malik, Computing the aspect graph for line draw-

ings of polyhedral objects, in *Proceedings, Computer Vision and Pattern Recognition, Ann Arbor, MI, June 1988*, pp 654–661

26. Z Gigus, J Canny, and R Seidel, Efficiently computing and representing aspect graphs of polyhedral objects, in *Proceedings, Second International Conference on Computer Vision, Tampa, FL, December 1988*, pp 30–39.

27. C Goad, Special purpose automatic programming for 3D model-based vision, in *Proceedings, DARPA Image Understanding Workshop, Arlington, VA, 1983*, pp 94–104

28. W Grimson and T Lozano-Pérez, Model-based recognition and localization from sparse range or tactile data, *Int J Robot Res* **3** (3), 1984, 3–35.

29. W Grimson, Recognition of object families using parameterized models, in *Proceedings, First International Conference on Computer Vision, London, UK, 1987*, pp 93–100

30. W Grimson, The effect of indexing on the complexity of object recognition, Memo No 1226, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, April, 1990

31. D Huttenlocher and S Ullman, Object recognition using alignment, in *Proceedings, First International Conference on Computer Vision, London, UK, 1987*, pp 102–111

32. D Huttenlocher, Three-dimensional recognition of solid objects from a two-dimensional image, Chap 7, Technical Report 1045, MIT Artificial Intelligence Laboratory, 1988

33. D Huttenlocher and S Ullman, Recognizing solid objects by alignment with an image, *Int J Comput Vision* **5** (2), 1990, 195–212.

34. J Koenderink and A van Doorn, The internal representation of solid shape with respect to vision, *Biol Cybernet* **32,** 1979, 211–216.

35. D Kriegman and J Ponce, Computing exact aspect graphs of curved objects: Solids of revolution, *Int J Comput Vision*, **5** (2), 1990, 119–135

36. Y Lamdan, J Schwartz, and H Wolfson, On recognition of 3-D objects from 2-D images, in *Proceedings, IEEE International Conference on Robotics and Automation, Philadelphia, PA, 1988*, pp 1407–1413

37. Y Liao, A two-stage method of fitting conic arcs and straight-line segments to digitized contours, in *Proceedings, Pattern Recognition and Image Processing Conference, Dallas, TX, 1981*, pp 224–229.

38. D Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic, Norwell, MA, 1985.

39. D Lowe, Fitting parameterized three-dimensional models to images, *IEEE Trans Pattern Anal Mach Intelligence* **13** (5), 1991, 441–450.

40. D Marr and H Nishihara, Representation and recognition of the spatial organization of three-dimensional shapes, *Proc R Soc London, B* **200,** 1978, 269–294

41. D Marr, *Vision*, Freeman, San Francisco, 1982

42. P Meer, D Mintz, and A Rosenfeld, Least median of squares based robust analysis of image structure, in *Proceedings, DARPA Image Understanding Workshop, Pittsburgh, PA, 1990*, pp 231–254.

43. D Metaxas and D Terzopoulos, Constrained deformable superquadrics and nonrigid motion tracking, in *IEEE Computer Vision and Pattern Recognition Conference, Maui, HI, 1991*, pp 337–343

44. D Metaxas and D Terzopoulos, Recursive estimation of shape and nonrigid motion, in *IEEE Workshop on Visual Motion, Princeton, NJ, 1991*, pp 306–311.

45. R Mohan and R Nevatia, Perceptual organization for segmentation and description, in *Proceedings, DARPA Image Understanding Workshop, Palo Alto, CA, 1989*, pp 415–424

46. P Mulgaonkar, L Shapiro, and R Haralick, Matching 'sticks, plates and blobs' objects using geometric and relational constraints, *Image Vision Comput* 3 (2), 1984, 85–98

47. R Nevatia and T Binford, Description and recognition of curved objects, *Artificial Intelligence* **8,** 1977, 77–98

48. N Nilsson, *Principles of Artificial Intelligence*, Chap 2, Morgan Kaufmann, Los Altos, CA, 1980

49. A Pentland, Perceptual organization and the representation of natural form, *Artificial Intelligence* **28,** 1986, 293–331.

50. A Pentland, Recognition by parts, in *Proceedings, First International Conference on Computer Vision, London, UK, 1987*, pp 612–620.

51. A Pentland and J Williams, Good vibrations: Modal dynamics for graphics and animation, *ACM Comput Graphics* **23** (4), 1989, 215–222

52. A Pentland, Automatic extraction of deformable part models, *Int J Comput Vision* **4,** 1990, 107–126

53. A Pentland and S Sclaroff, Closed-form solutions for physically based shape modeling and recognition, *IEEE Trans Pattern Anal Mach Intelligence* **13** (7), 1991, 715–729.

54. A Pentland and B Horowitz, Recovery of nonrigid motion and structure, *IEEE Trans Pattern Anal Mach Intelligence* **13** (7), 1991, 730–742.

55. H Plantinga and C Dyer, Visibility, occlusion, and the aspect graph, *Int J Comput Vision* **5** (2), 1990, 137–160.

56. U Ramer, An iterative procedure for the polygonal approximation of plane curves, *Comput Graphics Image Process* **1,** 1972, 244–256.

57. L Roberts, Machine perception of three-dimensional solids, in *Optical and Electro-Optical Information Processing* (J T Tippett *et al*, Eds), pp 159–197, MIT Press, Cambridge, MA, 1965

58. A Rosenfeld, Recognizing unexpected objects: A proposed approach, in *Proceedings, DARPA Image Understanding Workshop, Los Angeles, CA, 1987*, pp 620–627.

59. P Saint-Marc and G Medioni, Adaptive smoothing for feature extraction, in *Proceedings, 1988 DARPA Image Understanding Workshop, Cambridge, MA, 1988*, pp 1100–1113

60. M Sallam and K Bowyer, Generalizing the aspect graph concept to include articulated assemblies, *Pattern Recognition Lett* **12,** 1991, 171–176

61. L Shapiro and H Lu, Accumulator-based inexact matching using relational summaries, *Mach Vision Appl* **3,** 1990, 143–158

62. F Solina, *Shape Recovery and Segmentation with Deformable Part Models*, Technical Report MS-CIS-87-111, GRASP LAB 128, University of Pennsylvania, Philadelphia, December, 1987

63. T Sripradisvarakul and R Jain, Generating aspect graphs for curved objects, in *Proceedings, IEEE Workshop on Interpretation of 3D Scenes, Austin, TX, 1989*, pp 109–115.

64. L Stark and K Bowyer, Generic recognition through qualitative reasoning about 3-D shape and object function, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Maui, HI, June 1991*, pp 251–256

65. J Stewman, and K Bowyer, Creating and perspective projection aspect graph of polyhedral objects, in *Proceedings, IEEE Second International Conference on Computer Vision, Tampa, FL, 1988*, pp 494–500

66. J Stewman and K Bowyer, Direct construction of the perspective projection aspect graph of convex polyhedra, *Comput Vision Graphics Image Process* **51,** 1990, 20–37

67. M. Swain, Object recognition from a large database using a decision tree, in *Proceedings DARPA Image Understanding Workshop, Cambridge, MA, 1988*, pp. 690–696.

68. D. Terzopoulos, A. Witkin, and M. Kass, Symmetry-seeking models and 3D object recovery, *Int. J. Comput. Vision* **1**, 1987, 211–221.

69. D. Terzopoulos, A. Witkin, and M. Kass, Constraints on deformable models: Recovering 3D shape and nonrigid motion, *Artificial Intelligence*, **36**, 1988, 91–123.

70. D. Terzopoulos and D. Metaxas, Dynamic 3D models with local and global deformations: Deformable superquadrics, *IEEE Trans. Pattern Anal. Mach. Intelligence* **13** (7), 1991, 703–714.

71. D. Thompson and J. Mundy, Model-directed object recognition on the connection machine, in *Proceedings, DARPA Image Understanding Workshop, Los Angeles, CA, 1987*, pp. 93–106.

72. S. Ullman and R. Basri, Recognition by linear combinations of models, A.I. Memo No. 1152, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, August, 1989.