Three-Dimensional Object Recognition Systems
A.K. Jain and P.J. Flynn (Editors)

Part-Based Modeling and Qualitative Recognition

Sven J. Dickinson[a]

[a]Department of Computer Science
University of Toronto
6 King's College Road
Toronto, Ontario, M5S 1A4
Canada

1. Introduction

A multitude of object recognition paradigms have been proposed, each differing in their primitive (feature) extraction, matching, model representation, verification, or overall control strategies. Despite the tremendous variety of approaches, however, there is a very powerful metric that can be used to compare them. By examining the *indexing primitives* (image structures that are matched to object models) 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 Figure 1. In the left column are various indexing primitives ranging in complexity[1] 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 example 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 Figure 1, we see that as the complexity of indexing primitives increases, the number of primitives 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 [19], Huttenlocher and Ullman [15], Thompson and Mundy [24], and Lamdan et al. [17], are applied to small databases typically containing only a few objects.

[1]By complexity, we mean a primitive's descriptive power, typically proportional to the number of bits used to represent the primitive.

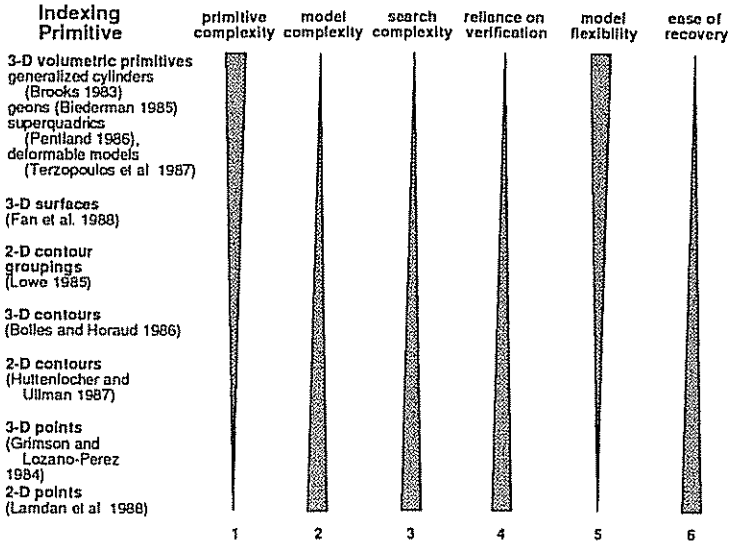


Figure 1. A comparison of object recognition systems according to their indexing primitives (reprinted from [7], ©1992 Academic Press)

Since simple indexing primitives imply 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, discriminatory indexing primitives to the model-based verification of simple indexing primitives. 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.

Reliance 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 primitives such as points or lines requires an accurate determination of the object's pose with respect to the image. If the pose is incorrect, the search of a local vicinity of the image for some model feature may fail. Needless to say, accurately solving for the object's pose can be computationally complex, particularly when a perspective projection camera model is used, e.g., Lowe [19].

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 [10], Huttenlocher [13], and Lowe [18]. 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 Figure 1 clearly indicate the advantages of using complex indexing features over simple ones. What is the trade-off? Why are most 3-D from 2-D recognition systems using simple indexing primitives?[2] 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 chapter addresses this challenge. From a single 2-D image, we present an approach to the recovery and recognition of 3-D objects using 3-D volumetric indexing primitives.

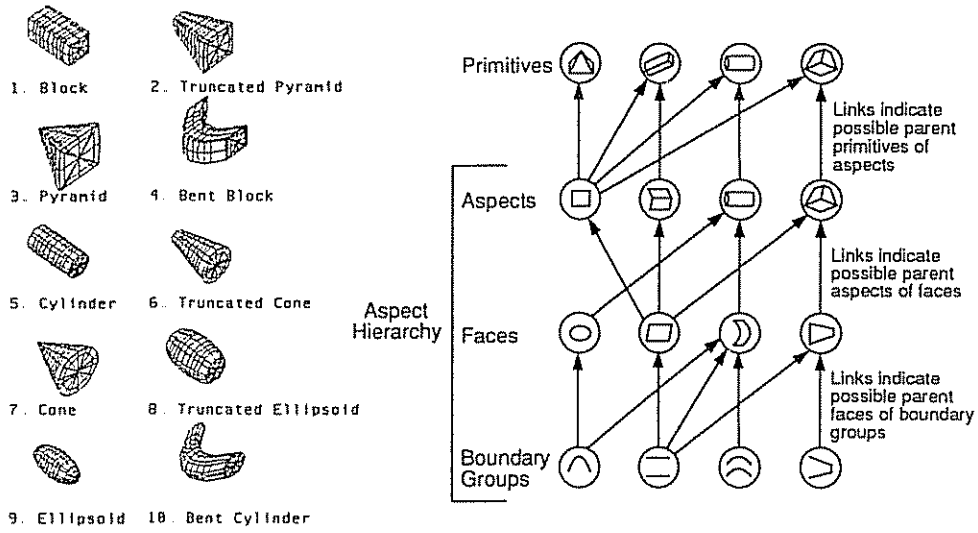
2. Object Modeling

2.1. Choosing the 3-D Primitives

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. In addition, the chosen primitives should be qualitatively defined so that the object models are invariant to minor changes in shape of the primitives. To demonstrate our approach, we have selected an object representation similar to that used in Biederman's Recognition by Components (RBC) theory [2]. 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*.

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 including cross-section shape, axis shape, and cross-section sweep. The

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



(a) (b)

Figure 2. (a) The ten modeling primitives (reprinted from [8], ©1992 IEEE), (b) The aspect hierarchy (reprinted from [8], ©1992 IEEE)

values of these properties give rise to a set of ten qualitative volumetric primitives.[3] 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 ten primitives were modeled using Pentland's SuperSketch 3-D modeling tool [21], as illustrated in Figure 2(a).[4] We believe that this taxonomy of volumetric primitives is sufficient to model a large number of objects; however, nothing in our approach is specialized for superquadrics or geons. If necessary, our approach can easily accommodate other sets of volumetric primitives bearing little resemblance to geons or superquadrics.

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 [16]). Our approach differs

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

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.

To minimize the number of aspects needed to represent the primitives, we constrain the aspects to be invariant to minor changes in primitive shape. By encoding only region topology and qualitative region shape, a particular aspect of a primitive becomes invariant to changes in primitive size, curvature, taper, etc. As a result, a small set of qualitatively different aspects describes a small set of qualitatively different volumetric primitives; each primitive, in turn, describes an enormous range of 3-D shape. The advantage of this approach is that since the number of qualitatively different primitives used to build objects 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 further compounded when objects are articulated.

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. Clearly, 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; Figure 2(b) illustrates a portion of the aspect hierarchy.

- *Aspects* constitute the top level of the aspect hierarchy and represent all possible views of the primitives in terms of visible faces. 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* constitute the second level of the aspect hierarchy and represent all possible component faces making up the aspects. 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* constitute the third and lowest level of the aspect hierarchy and represent all subsets of the faces' bounding contours. 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. To generate these conditional probabilities, we first model our 3-D volumetric primitives using the SuperSketch modeling tool [21], as shown in Figure 2(a). 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.[5] For each view, we 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 [6]).

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 spatial orientation distribution of a primitive is 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 more 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. Nevertheless, the computation of the aspect hierarchy is performed off-line and is *independent* of the contents of the object database.

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 [8] 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

[5]Rotating the superquadric about its z axis in 10° intervals results in 36 views for a given x-y orientation. If we consider 18 x-y orientations by fixing either the x or the y orientation and varying the other at 10° intervals, we can effectively cover the viewing sphere with 648 views.

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.

3.1. Extracting Faces

The first step in extracting faces consists of extracting the bounding contours of image regions. We begin by applying Canny's edge detector [5] to the image followed by Beymer's algorithm [1] to fill gaps in the detected edges; minimal cycles in the resulting edge map correspond to the bounding contours of regions (or faces) in the image. The next step is to partition the contours at significant curvature discontinuities. We apply Saint-Marc and Medioni's scale-space adaptive smoothing algorithm [22] to come up with a partitioned contour at low, medium, and high scales.

Once the faces have been extracted, we must classify each face according to the faces in the aspect hierarchy. Both an image face and an aspect hierarchy face are represented by a graph in which nodes represent the face's bounding (partitioned) contours, and arcs represent relations between contours. Contours are characterized as straight or curved depending on how well a straight line can be fitted to them; furthermore, curves are characterized as convex or concave. Two non-coterminating lines are considered parallel if the angle between their fitted lines is small, while two non-coterminating curves are considered parallel if one is convex, one is concave, and the angle between their directions is small.[6] Two two non-coterminating, non-parallel lines are considered symmetric if there is sufficient overlap when one line is projected onto the other.[7]

Each of the three scales in the curve partitioning step gives rise to a graph representing an image face. Consequently, the classification of an image face consists of comparing its three graphs to those graphs representing the faces in the aspect hierarchy. We begin with the graph corresponding to the low scale in the curve partitioning step, often representing an oversegmentation of the face's bounding contour. If there is an exact match, as shown in Figure 3, then we immediately generate a *face hypothesis* for that image face, identifying the label of the face. If for any reason (e.g., occlusion, segmentation errors, noise, etc.), there is no match, we must descend to the boundary group level of the aspect hierarchy, as shown in

[6]The direction of a curve is computed as the vector whose head is defined by the midpoint of the line joining the two endpoints of the curve, and whose tail is definied by the point on the curve whose distance to the line joining the endpoints is greatest.

[7]Two non-parallel vectors will have an intersection point. When one vector is rotated about that point it can be brought into correspondence with the other. If the resulting overlap of the two lines is a large portion of the smaller of the two lines, the lines are said to be symmetric.

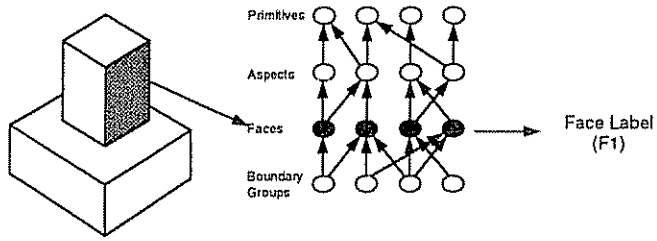


Figure 3. Labeling an unoccluded face (reprinted from [7], ©1992 Academic Press)

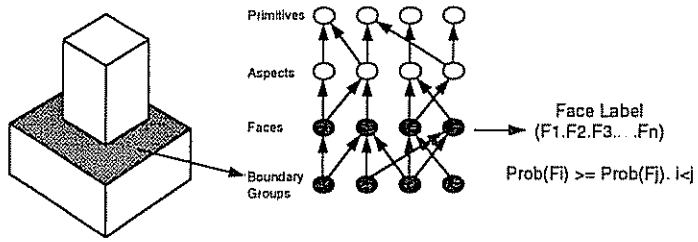


Figure 4. Labeling an occluded face (reprinted from [7], ©1992 Academic Press)

Figure 4. 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. This process is repeated for each of the other two graphs (medium and high scales). If, for any scale, a graph matches an aspect hierarchy face, the graphs representing all other scales are discarded; otherwise, we collect together the face hypotheses generated at all three scales. Each face hypothesis defines one or more *seed contour sets* representing those bounding contours of the image face (entire face or specific boundary groups) which define the label of the face.[8]

[8]In the case of a face hypothesis whose image face exactly matches an aspect hierarchy face (defined by the label of the face hypothesis), there will be a single seed contour set containing all the bounding contours of the image face. However, in the case of a face hypothesis whose image face does not match an aspect hierarchy face, there will be one seed contour set for every boundary group supporting the face hypothesis.

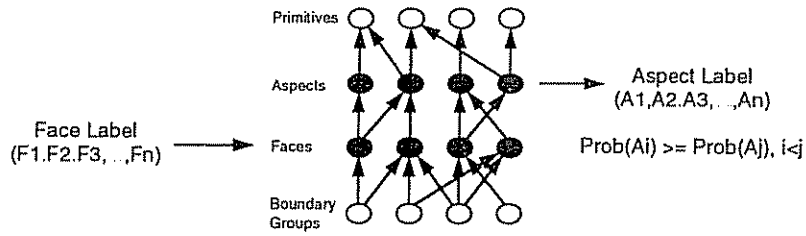


Figure 5 Generating the aspect labels of a face (reprinted from [7], ©1992 Academic Press)

3.2. Extracting Aspects

3.2.1. Problem Definition

The image can now be represented by a *face graph*, in which nodes represent faces (or regions) and arcs represent face adjacencies. Furthermore, following face extraction, each node (face) in the face graph has one or more face hypotheses associated with it. 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 [8] for a discussion on the problem's computational 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 Figure 5; 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.

3.2.2. Aspect Instantiation

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 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 criteria, 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.

Let us explore the aspect instantiation process in more detail. Consider a face graph, FG, and an aspect hypothesis, ah, with label l, seeded at face f in FG. The aspect hierarchy aspect corresponding to label l, herein called the aspect definition, specifies that the aspect contains k faces, each with specified label and adjacency relations. We first collect together all neighboring faces of f (including f itself) in FG. Next, we generate all face subsets of size $\leq k$ from this collection; recall that there is an upper bound on k which is fixed (specified by the aspect hierarchy) and independent of the size of FG. For each subset, we check to see if the subgraph of FG (i.e., face subgraph) induced by the face subset is isomorphic to the aspect definition. For each matching subset, we instantiate an aspect; the result is a (possibly empty) list of instantiated aspects.

An aspect can be instantiated from an aspect hypothesis and a face subgraph if and only if the following conditions are satisfied:

- For each face in the face subgraph, there must exist, among its list of face hypotheses, a hypothesis whose label agrees with the label of its matching face in the aspect definition; if such a face hypothesis is found, it is *assigned* to the face in the subgraph.

- For each arc (or face adjacency relation) in the face subgraph, there must exist a corresponding arc in the aspect definition. Similarly, for each arc in the aspect definition, there must exist a corresponding arc in the face subgraph.

- For each arc in the face subgraph involving two faces, A and B, there must exist a seed contour set belonging to the face hypothesis assigned to A, and a seed contour set belonging to the face hypothesis assigned to B, such that each of the two seed contour sets includes the contours shared by A and B. Or, more intuitively, the contour(s) shared by two faces must be seed contours of both faces.

- For each face in the face subgraph, there must exist at least one seed contour set belonging to its assigned face hypothesis that satisfies all face adjacency relations involving that face.

If an aspect with k faces cannot be instantiated from an aspect hypothesis, it may be due to the fact that the aspect is occluded in the image. In this case, our goal is to find subsets of image faces that match portions of the aspect definition. Consider the set S of all subsets of image faces such that for each s in S, $|s| < k$ and the face subgraph induced by s matches some portion of the aspect definition (according to the above set of conditions). In addition, according to the partial match, let the valid seed contour sets at face i in s be $SC_1^i, SC_2^i, \ldots, SC_w^i$. Finally, let r represent the faces

in the aspect definition not included in s (presumably occluded). We instantiate the aspect encompassing s provided the following conditions are satisfied:

- There exists no other subset t in S such that s is a proper subset of t and the aspect encompassing t has been instantiated. Or, more intuitively, if a set of faces satisfies an aspect, we ignore its subsets (which may also satisfy the aspect).
- For each arc in the face graph involving a face, fs, in s, and a face, fr, in r, there must exist a valid seed contour set SC_j^{fs} belonging to the face hypothesis assigned to fs such that the contours shared by fs and fr do not appear in the seed contour set. Or, more intuitively, if face A is occluded by face B, then the contours shared by faces A and B (which belong to face B) should not be seed contours of face A.

The above restrictions have a significant impact upon the selection of boundary groups. If we have a weak (i.e., low probability) face hypothesis, then it is likely that each of its seed contour sets represents a small fraction of the contours comprising the face. Consequently, instantiation of an aspect including such a face hypothesis may fail since it is likely that the required neighboring faces do not border at seed contours. However, with the lack of seed contours, smaller subgraphs may match the aspect definition since it is likely that neighboring faces do not border at seed contours. We conclude that there is a trade-off between selecting only the best boundary groups and exhaustively selecting all boundary groups. In the former case, a strong face hypothesis supported by strong boundary groups will likely match few aspect definitions, pruning out many interpretations of the face. However, if weaker boundary groups are not included in the face hypothesis, a correct interpretation may be impossible. Conversely, the presence of weak boundary groups allows occluded aspects to be instantiated. Although this may guarantee a solution, the increased number of interpretations may lengthen the search for a solution, and may result in less likely solutions being prematurely generated.

3.2.3. Algorithm

We can now reformulate our problem as a search through the space of aspect labelings of the faces in our face graph.[9] 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 6 illustrates the correct aspect covering of the face graph representing a scene containing an object composed of two blocks; one aspect label, in this case, A27 (see [8] 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.

[9]The size of this space is $O(A^F)$, where A is the number of aspects in the aspect hierarchy, and F is the number of faces in the image.

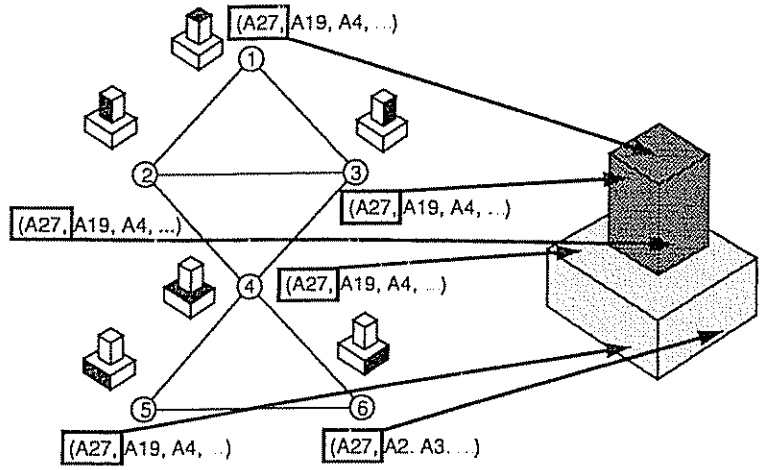


Figure 6. Covering the face graph with aspects (the aspect labeling problem) (reprinted from [7], ©1992 Academic Press)

For our search through the possible aspect labelings of the face graph, we employ Algorithm A (Nilsson [20]) 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 (using a depth-first search) 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 is only continued if the recovered shapes (inferred from the aspect covering) can not be recognized.

Before adding a successor state to the open list, it is evaluated using a heuristic function. The 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 an algorithm for evaluating a state, as shown in Figure 7; note that a node consists of a number of *indices*, one per face, with each *index* referring to a particular aspect hypothesis for that face.[10]

[10]The values of c_1 and c_2 were empirically chosen to be 0.25 and 0.50, respectively.

input: *node*
output: *value*
value = 0
for each *index* **in** *node* **do**
    aspect set = set of instantiated aspects pointed to by *index*
    aspect hypothesis = aspect hypothesis from which *aspect set* was instantiated
    value = *value* -
        probability(*aspect hypothesis*) -
        $(c_1 * \frac{\text{maximum number of visible faces of any aspect in } \textit{aspect set}}{\text{number of faces in } \textit{aspect hypothesis} \text{ definition}})$ -
        (c_2 * maximum number of visible faces of any aspect in *aspect set*)
return *value*

Figure 7. Heuristic for Evaluating a State (reprinted from [8], ©1992 IEEE)

3.3. Extracting Primitives

We can represent an aspect covering by a graph in which nodes represent aspects and arcs represent aspect adjacencies. For each aspect in the aspect covering, we can use the aspect to primitive mapping to hypothesize a set of *primitives*, as illustrated in Figure 8.[11] 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*.[12] 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 primitive, thereby favoring higher probability interpretations.

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 connected in the aspect covering, their corresponding primitives are not connected in the primitive covering. However, if two aspects are connected in the aspect covering, this does not mean that their corresponding primitives are necessarily connected in 3-D; one primitive may be occluding the other without being attached to it. A primitive connection

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

[12] The number of possible primitive coverings is $O(P^A)$, where P is the number of primitives in the aspect hierarchy, and A is the number of aspects in the aspect covering.

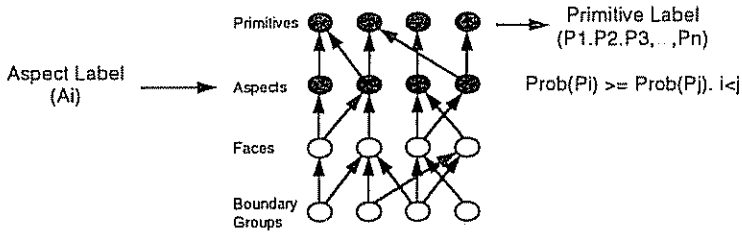


Figure 8. Generating the primitive labels of an aspect (reprinted from [7], ©1992 Academic Press)

between primitives P_1 and P_2 is said to be visible if the following condition is satisfied:

- 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.

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 covering, 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 covering, 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. If a strongly connected subgraph is common to two object models, these heuristics can then be used to rank order the candidates for verification.

Hypothesizing the attachment surfaces proceeds as follows. Let S_1 be the set of faces belonging to the aspect corresponding to P_1 which are adjacent to a face belonging to the aspect corresponding to P_2. Similarly, let S_2 be the set of faces belonging to the aspect corresponding to P_2 which are adjacent to a face belonging to the aspect corresponding to P_1. There are three cases to consider:

1. *Sets S_1 and S_2 each contain a single face.* The attachment surface for P_1 is among the set of attachment surfaces that are adjacent to, and including, the surface representing the face in S_1. The attachment surface for P_2 is among the set of attachment surfaces that are adjacent to, and including, the surface representing the face in S_2. More intuitively, we believe that the attachment surface is in the local vicinity (on the primitive) of the attachment surface corresponding to the single visible face.

2. *Set S_1 contains a single face and set S_2 contains multiple faces.* In this case, the attachment surface for P_1 is the surface that the face in S_1 maps to. The attachment surface for P_2 is among the set of surfaces that are adjacent to, but not included in, the surfaces representing the faces in S_2. More intuitively, we believe that P_2 penetrates P_1; since the connection is visible, the attachment surface for S_1 is therefore attached to an occluded surface of P_2. (The same holds true when set S_2 contains a single face and set S_1 contains multiple faces.)

3. *Sets S_1 and S_2 both contain multiple faces.* In this case, the attachment surface for P_1 is among the set of surfaces that are adjacent to, but not included in, the set of surfaces representing the faces in S_1. The attachment surface for P_2 is among the set of surfaces that are adjacent to, but not included in, the set of surfaces representing the faces in S_2. More intuitively, we believe that although the attachment of P_1 and P_2 is visible, both their attachment surfaces are occluded.

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.

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 [11]. Second, this approach assumes that a primitive covering represents a single object. If the scene contains multi-

ple 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?

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) map 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.

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 using 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 larger than if we index using 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 3.3, we hypothesized that if a connection between two primitives was *visible*, i.e., a *strong* connection, the two primitives were connected in 3-D. Our model indexing strategy therefore consists of identifying all the *strongly* connected components in the primitive graph, each hypothesizing a set of object candidates according to the two-level hash function described in Figure 9(a).

At the first level, we hash on the basis of a string formed from the labels of the primitives in the strongly connected component. Each entry in this table points to a separate hash table at the second level which encodes primitive connections; each hash table at the second level corresponds to objects that contain a particular set of primitives (number and type). Once at the second level, we hash on the basis of a string formed from the connections in the strongly connected component. It is

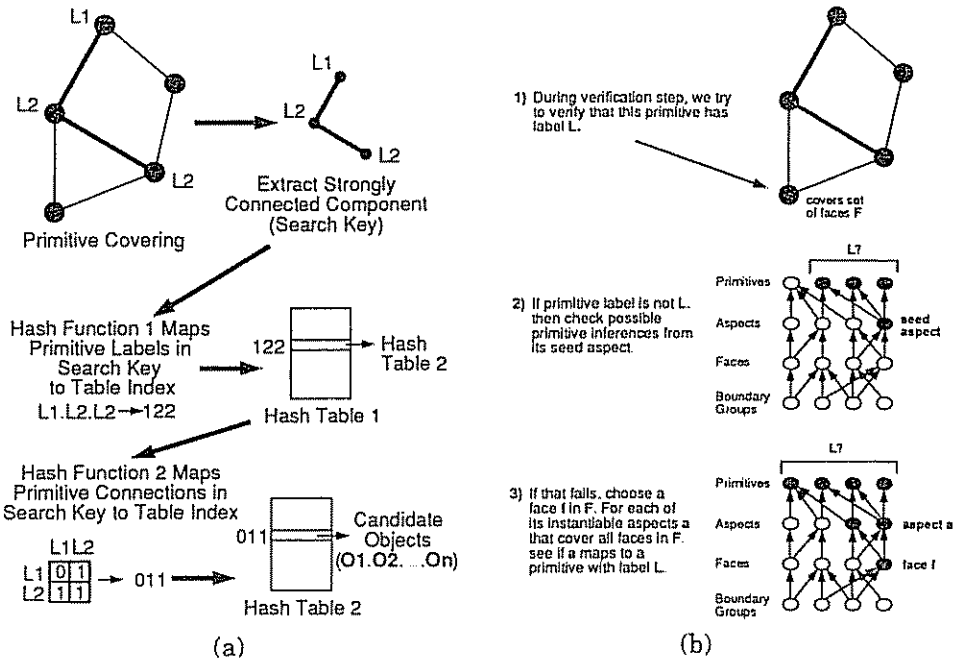


Figure 9. Object recognition strategies: (a) Unexpected object recognition (reprinted from [7], ©1992 Academic Press), (b) Expected object recognition (reprinted from [7], ©1992 Academic Press)

at this level that we further discriminate the objects on the basis of their primitive connections. The entire process is repeated for each strongly connected component in the primitive covering, resulting in a set of candidate object hypotheses.

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 finds maximal correspondences between recovered primitives and model primitives.

Since our hash function ignores the connections in the strongly connected subgraph, the first step is to check that the strong connections in the strongly connected subgraph 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 subgraph in the primitive graph is isomorphic to a subgraph of the candidate model. We then grow this correspondence according to the following steps:

```
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 + c1
    else (arc is strong)
      if arc correctly specifies attachment surfaces then
        goodness of fit = goodness of fit + c2
      else
        goodness of fit = goodness of fit + c3
```

Figure 10. Goodness of fit algorithm for model candidates (reprinted from [7], ©1992 Academic Press)

1. Given a correspondence between a primitive (covering) 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 covering 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

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

The final step ranks the correspondences according to a goodness of fit measure defined by the algorithm shown in Figure 10. 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*.[13]

Once the correspondences are ranked according to the goodness of fit measure, we choose the best correspondence and remove those aspects from the image that cor-

[13] 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.

respond 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 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 to what object the image features will be matched. 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. The assumption here is that the first aspect and primitive coverings of the scene represent a correct interpretation for much of the scene, and provide a good starting point for object search.

Figure 9(b) 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 (aspect used to infer primitive) 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 alternate interpretation and add it to the correspondence. If not, we can probe deeper for the correct interpretation.

Descending one more level of the aspect hierarchy, there may be other aspect interpretations of the faces belonging to the seed aspect. Furthermore, one of these interpretations may be used to infer the primitive we are searching for. Therefore, our strategy consists of searching the aspect labels of the faces belonging to the seed aspect for an aspect which not only can be verified, but whose mapping to the desired primitive has a nonzero conditional probability. If such an aspect is found, the desired primitive is inferred and added to the correspondence. If the search is unsuccessful, the correspondence will not include the faces belonging to the seed aspect.

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^{TM} workstation. The image preprocessing which takes an image and returns a gap-filled skeletonized image is performed in the KBVision environment. In this section we apply OPTICA to the six images presented in Figure 11; images (c) and (d) are real images of objects constructed out of clay, while (a), (b), (e), and (f) were generated using Pentland's Thingworld modeling tool.

Figures 12(a) through (d) present the results of applying OPTICA to the images in Figures 11(a) through (d). There are four windows in each figure. At the top, the image window contains the contours extracted from the image, along with the face numbers. To the left is the diagnostic window describing the recovered primitives (primitive covering). The mnemonics PN, PL, PP, and PS, refer to primitive number (simply an enumeration of the primitives in the covering), primitive label (see Figure 2(a)), and primitive probability, respectively. The mnemonics AN, AL, AP, and AS refer to the aspect number (an enumeration), aspect label (see [8]), aspect probability, and aspect score (how well aspect was verified), respectively. The mnemonics FN, FL, FP, and PS refer to face number (in image window), face label (see [8]), face probability, and corresponding primitive attachment surface (see [8]), respectively, for each component face of the aspect.

To the right are the "Recognized Objects" and "Primitive Connections" windows. The "Recognized Objects" window indicates the aspect covering iteration and primitive covering iteration (given the aspect covering). In addition, this box lists all objects currently (at the above iterations) identified in the image, including their corresponding primitive numbers (PN). The "Primitive Connections" window 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.

In each case ((a) through (d)), the first aspect and primitive coverings represent

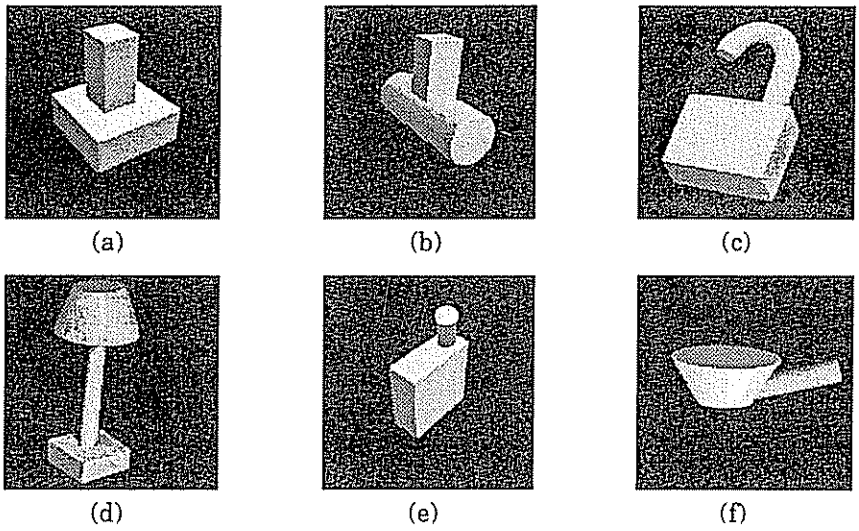


Figure 11. The six images input to OPTICA ((c) and (d) are real images, while (a), (b), (c), (d) were generated using Pentland's Thingworld modeling tool) ((c) and (d) reprinted from [8], ©1992 IEEE)

the correct interpretation of the scene. Total execution time, excluding the time required to extract the contours (90 seconds), was approximately 60 seconds in each case; the majority of that time was spent in building a graph representation of an image face, i.e., connected components analysis, curve partitioning, curve classification, and symmetry and parallelism detection.

Figure 13(a) presents the results of applying OPTICA to the image in Figure 11(e). In this case, the first aspect and primitive coverings do not represent the correct interpretation of the scene (arm of lock is misinterpreted as a cylinder). If we let the algorithm continue, we arrive at the correct interpretation with the second primitive covering given the first aspect covering, as shown in Figure 13(b). When we apply the expected object recognition algorithm (searching for a lock), the search for the bent cylinder arm is constrained to the area covered by the cylinder faces (first covering). In this case, the correct primitive interpretation was found by descending only one level of the aspect hierarchy, i.e. the aspect was correct but the primitive inference was wrong.

Similarly, Figure 13(c) presents the results of applying OPTICA to the image in Figure 11(f). Again, the first aspect and primitive coverings do not represent the

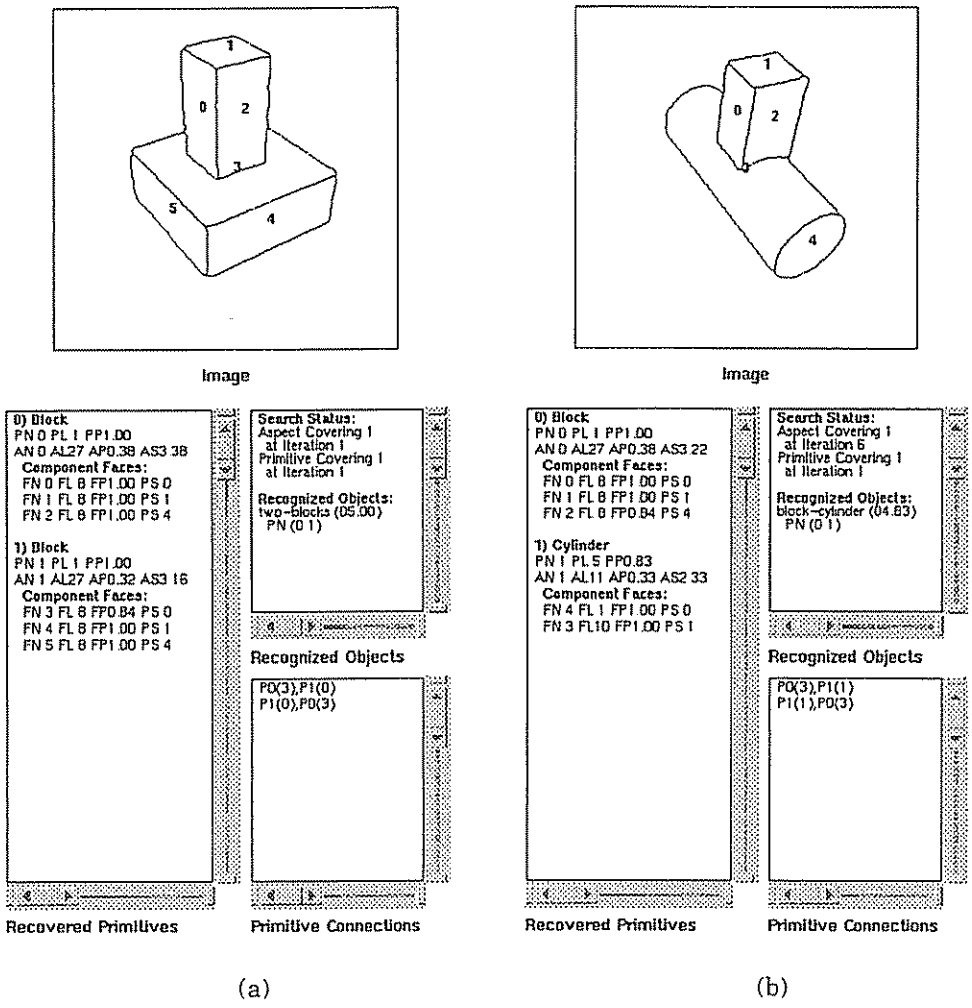


Figure 12. Results of applying OPTICA to images in Figures 11(a) and (b)

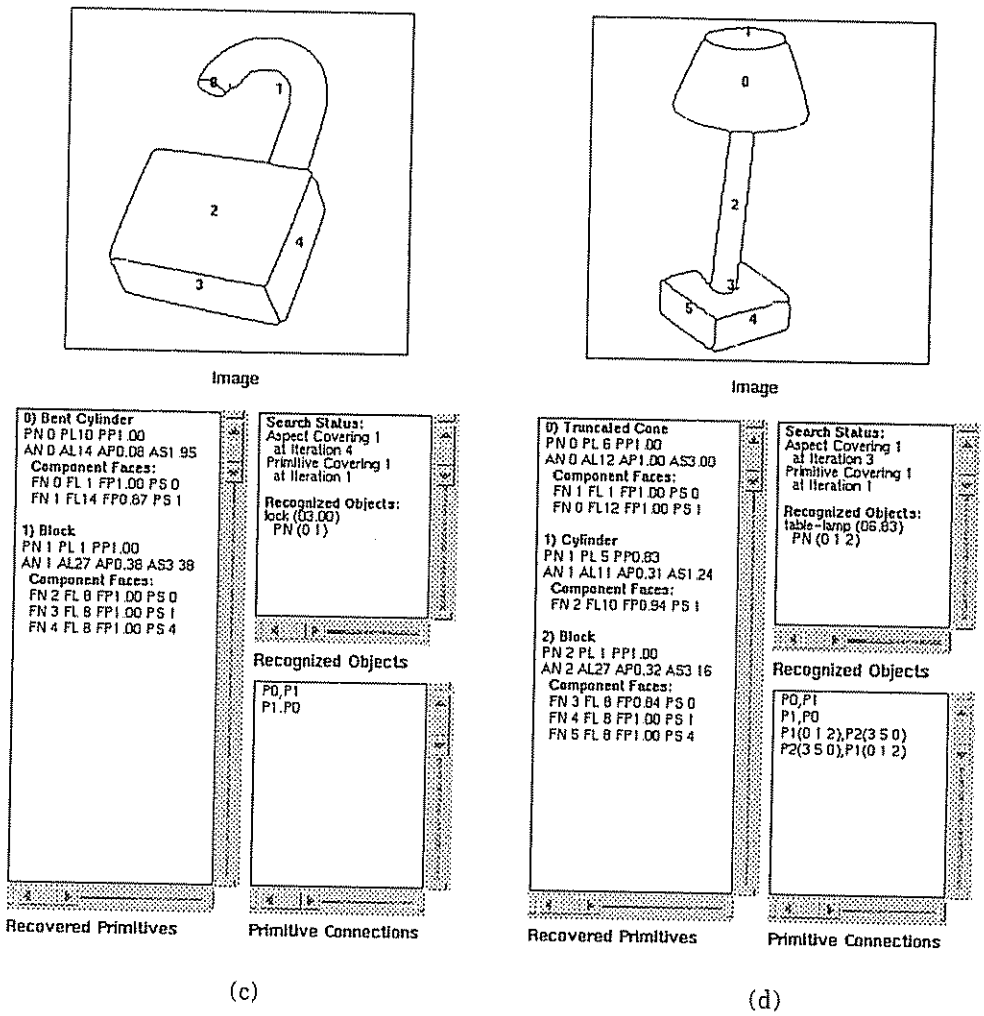


Figure 12 Results of applying OPTICA to images in Figures 11(c) and (d)

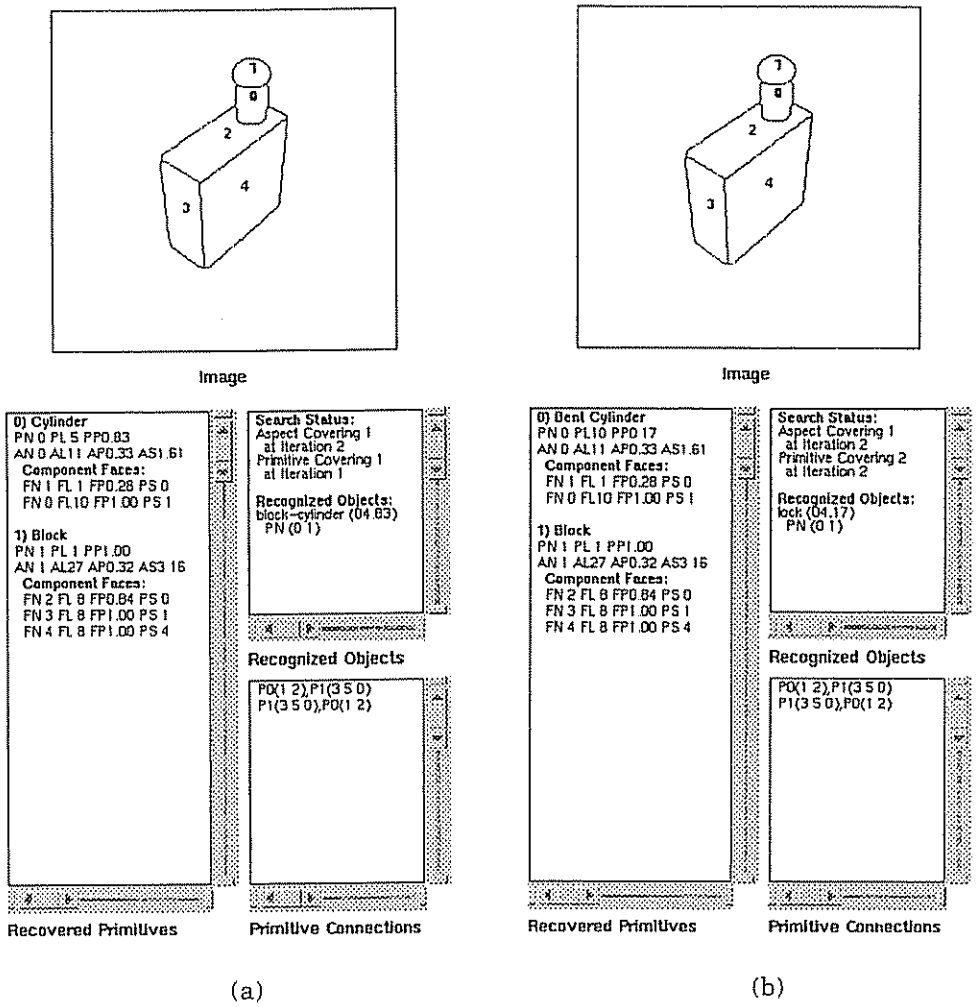


Figure 13. Results of applying OPTICA to image in Figure 11(e)

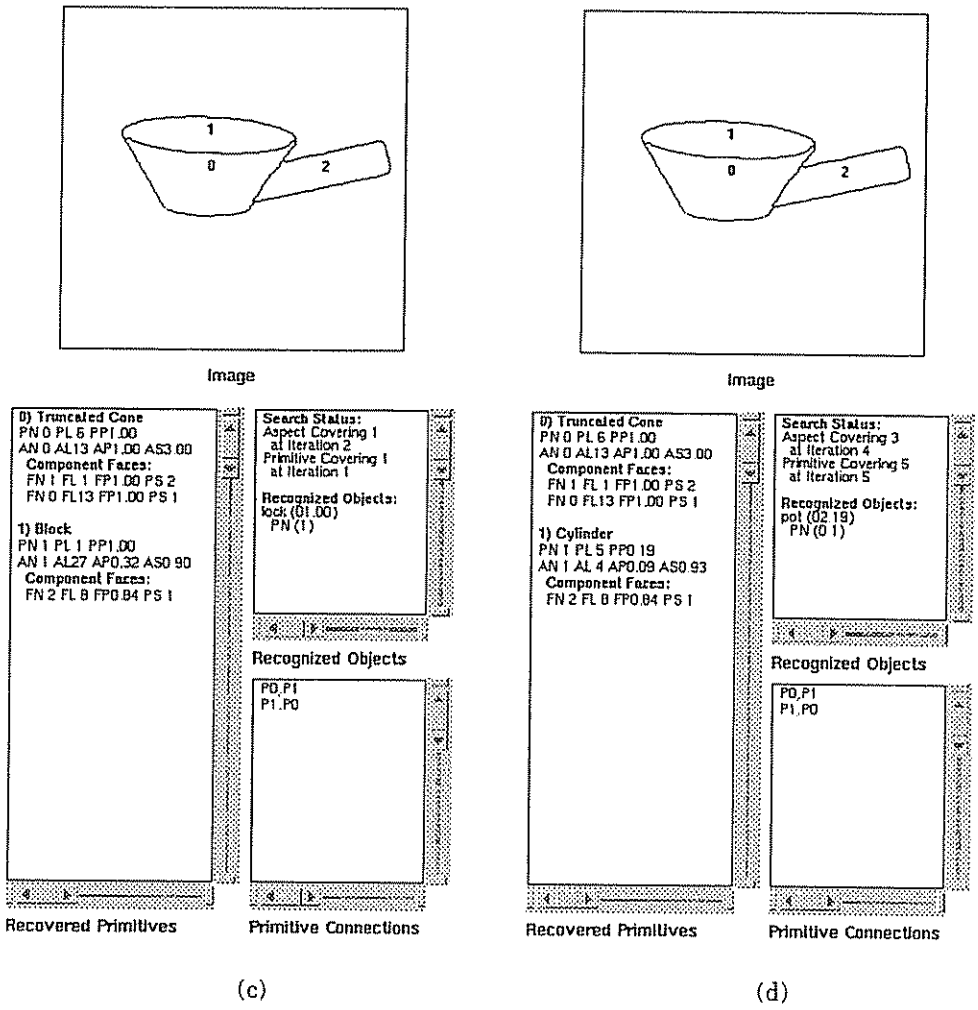


Figure 13. Results of applying OPTICA to image in Figure 11(f)

correct interpretation of the scene (handle of pot is misinterpreted as a block). If we let the algorithm continue, we eventually arrive at the correct interpretation with the fifth primitive covering given the third aspect covering, as shown in Figure 13(b). When we apply the expected object recognition algorithm (searching for a pot), the search for the cylinder handle is constrained to the area covered by the block faces (first covering). In this case, the correct primitive interpretation was found by descending two levels of the aspect hierarchy, i.e. a new aspect was recovered before the correct primitive inference could be made.

6. Conclusions

The inefficiency of most 3-D object recognition systems is reflected in the relatively small number of objects in their databases; 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, resulting in 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 [2]; 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 [8]. 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.

7. Acknowledgements

I would like to thank Azriel Rosenfeld, Sandy Pentland, and more recently, John Tsotsos for their generous support and guidance throughout this work. I would also like to 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. Finally, I would like to thank Gérard Medioni for supplying the curve partitioning software, and Ken Shih for implementing the Beymer gap-filling algorithm.

REFERENCES

1 D. Beymer. Finding junctions using the image gradient. Memo 1266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991.

2 I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.

3 R. Bolles and P. Horaud. 3dpo: A three-dimensional part orientation system. *The International Journal of Robotics Research*, 5(3):3–26, 1986.

4 R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.

5 J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

6 S. Dickinson, A. Pentland, and A. Rosenfeld. 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, 1991.

7 S. Dickinson, A. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3-D object recognition. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(2), 1992.

8 S. Dickinson, A. Pentland, and A. Rosenfeld. 3-D shape recovery using tributed aspect matching. *IEEE Transactions on Pattern Analysis and Mach Intelligence*, 14(2):174–198, 1992.
9 T. Fan, G. Medioni, and R. Nevatia. Recognizing 3-D objects using surf descriptions. In *Proceedings, Second International Conference on Compu Vision*, pages 474–481, Tampa, FL, 1988.
10 W. Grimson. Recognition of object families using parameterized models. *Proceedings, First International Conference on Computer Vision*, pages 93–1 London, UK, 1987.
11 W. Grimson. The effect of indexing on the complexity of object recognit Memo 1226, Artificial Intelligence Laboratory, Massachusetts Institute of Te nology, 1990.
12 W. Grimson and T. Lozano-Pérez. Model-based recognition and localiza from sparse range or tactile data. *International Journal of Robotics Resea* 3(3):3–35, 1984.
13 D. Huttenlocher. Three-dimensional recognition of solid objects from a t dimensional image. Technical Report 1045, Artificial Intelligence Laborat Massachusetts Institute of Technology, 1988.
14 D. Huttenlocher and S. Ullman. Object recognition using alignment. In *P ceedings, First International Conference on Computer Vision*, pages 102– London, UK, 1987.
15 D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment v an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.
16 J. Koenderink and A. van Doorn. The internal representation of solid sh with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.
17 Y. Lamdan, J. Schwartz, and H. Wolfson. On recognition of 3-D objects f 2-D images. In *Proceedings, IEEE International Conference on Robotics Automation*, pages 1407–1413, Philadelphia, PA, 1988.
18 D. Lowe. Fitting parameterized three-dimensional models to images. *IE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441– 1991.
19 David Lowe. *Perceptual Organization and Visual Recognition*. Kluwer A demic Publishers, Norwell, MA, 1985.
20 N. Nilsson. *Principles of Artificial Intelligence*, chapter 2. Morgan Kaufm Publishers, Inc., Los Altos, CA, 1980.
21 A. Pentland. Perceptual organization and the representation of natural fo *Artificial Intelligence*, 28:293–331, 1986.
22 P. Saint-Marc and G. Medioni. Adaptive smoothing: A general tool for e vision. *IEEE Transactions on Pattern Analysis and Machine Intellige* 13(6):514–529, 1991.
23 D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and object recovery. *International Journal of Computer Vision*, 1:211–221, 1987
24 D. Thompson and J. Mundy. Model-directed object recognition on the connec machine. In *Proceedings, DARPA Image Understanding Workshop*, pages 106, Los Angeles, CA, 1987.