Model-Based Strategy Planning for Recognizing Partially Occluded Parts

Radu Horaud, LIFIA

Thomas Skordas, D-LETI

Recognizing partially occluded objects requires off-line modeling and planning, and runtime recognition. After deriving a customized method, we can compare recognition with and without off-line planning.

reprinted from

COMPUTER magazine

Recognizing partially occluded parts has great interest for automating batch manufacturing and thus has been an active research topic for the last decade. Initial work in the domain of object recognition has concentrated on solving simple situations such as isolated parts lying on a planar surface: The parts can be recognized using binary images from which silhouettes are extracted and global features measured. A part is said to be recognized whenever a set of measured features matches a set of prestored features that best describes the part. This strategy supposes that the objects have been previously separated by mechanical devices such as shakers, bowl-feeders, conveyors, or other special-purpose arrangements.

A universal solution to the problem of object separation based on a purely mechanical system does not seem to exist. Moreover, it would be economically expensive because every object requires the design of a specific mechanical configuration, and ill-adapted whenever an object needs to be manipulated with care. Therefore, a vision system capable of recognizing parts when they touch or partially overlap each other could be very useful for achieving flexible automation.

When parts overlap, image analysis becomes a much more complex process. Since global features are not available from a partial view, we must consider local features. The identification of an object depends in general on some prominent features characterizing it. The absence from the image of one or more of these features due to occlusion or noisy data increases the difficulty of correct recognition, since hidden components cannot guide the analysis. We must find alternative solutions. Finally, there is no one-to-one correspondence between image and object features. We must build and search an interpretation space for possible solutions: image-to-object matches.

Several object recognition systems coping with partially hidden objects have already been developed. The best performing ones are model-based: they match observed local features to stored models. The basic differences between these systems are the type of object representations they employ and the matching strategies they apply.

A first approach consists of describing the shape of an object boundary in terms of a one-dimensional function and of matching this description with pieces of contours extracted from an image.[1,2] It identifies objects in terms of a few salient features in the shapes of their boundaries. This approach is powerful only in the case where there exist specific boundary fea-

0018-9162/87/0800-0058$01.00©1987IEEE

tures that allow us to distinguish one object from another one. It does not explore the full range of information available with feature combinations.

A second approach consists of considering simple generic features such as straight lines and circular arcs. These features well suit describing a wide range of shapes. Each shape is characterized by its intrinsic feature parameters and by the mutual relationships between features. Many systems have based their matching strategy on this type of description.[3-6] We can paraphrase such a strategy as follows:

- Locate a distinctive feature of the object to be found;
- use the feature's position to suggest where to look for a second feature to verify the first; and
- use the two features to predict a third feature which, together with the first two, completely constrains the position and orientation of the object.

Since some of the predicted features may not be visible, we must provide alternatives. Therefore, a complete strategy consists of an ordered list of features, with each feature having two sets of deductions, one to be made if the feature is found and the other if it is not. The process of recognizing and locating an object is thus a tree search.

A key to the success of a strategy like this is the off-line modeling and planning: the selection, classification, and ordering of those object features to be considered at each stage of the search. The work reported herein concentrates on automatic techniques for performing this planning, given a geometric database describing the objects we want to locate and identify. We also attempt to determine the criteria for asserting that an object has been correctly identified and located; we can use the criteria as a heuristic for reducing the combinatorial explosion of the search space.

The task of recognizing an object in a random position and orientation is not trivial. Generally an object has six degrees of freedom associated with it: three rotations and three translations. If an object is constrained to lie on a planar horizontal surface (on a tabletop, for example) in a stable position, three degrees of freedom remain to be determined: one rotation and two translations. In this article we constrain the image plane to be parallel to the tabletop. This reduces distortion of scene features by the perspective effect inherent with the image formation process. For the more general case see Horaud.[7]

Related work

Previous approaches to object recognition have concentrated on the runtime system and not on the off-line planning. The LFF system[3] is the only system we know of that performs off-line planning in conjunction with a search strategy. However, LFF cannot derive stable positions and 2D silhouettes associated with them from a general-purpose 3D modeling system.

The 3DPO system[6] locates objects in range data. Range data are less ambiguous than intensity data because they directly encode the geometry of the scene. Similarly, the LFF system deals with binary images that keep the image edge description as simple as possible. Therefore, the graph search associated with either range or binary images is less complex than the one associated with grey-level images. A certain number of heuristics that did not appear important with range and binary images become crucial if intensity data are to be properly interpreted.

The modeling system

A modeling system should enable us to construct models for a large variety of industrial parts and enable vision algorithms to recognize them. No single representation scheme can efficiently support these two basic requirements. A modeling system, to be powerful, should contain multiple representations of objects.[8]

Computer-aided design (CAD) systems have facilitated the construction and visualization of objects. Although these systems are potentially very powerful since they embed a lot of implicit (computable) information, they do not contain in an explicit form the geometrical and topological properties useful for recognition and positioning. A vision-oriented representation must contain explicit lists of these properties and must express relationships between object features. It must be able, for example, to rapidly provide answers to questions such as

- How many features of a certain type or size does the model include?
- Which is the local configuration of a specific feature, i.e., which are its nearby features?
- Which are the sets of simultaneously visible features?

In order to meet the above requirements, we have implemented a two-part extended CAD model representation: a standard geometric model and a vision-oriented model.

The standard geometric model. The standard CAD geometric model contains a volumetric description (solid modeling), a surface-edge-vertex description (boundary representation), and a network of pointers linking topologically connected features.

The volumetric description is a binary tree representing elementary primitives (the leaves of the tree) together with the Boolean operations and rigid transforms that allow construction of the solid model (the root of the tree). This representation, known as constructive solid geometry (CSG), has been described elsewhere[8]; it is a basic component of many geometric modeling systems.

In our implementation, each volume primitive has associated with it a *planar patch description*, which is an exact description for polyhedra (parallelepipeds, pyramids, etc.) for which the patches are the faces themselves and an approximate description for primitives bounded by curved surfaces (cylinders, cones, and spheres). In the latter case, each surface type has an associated planar patch approximation algorithm that facilitates the boundary evaluation process. Indeed, a keystone component of any boundary evaluation method is the computation of surface intersections.

The complex problem of computing the intersection of two quadrics reduces to the problem of computing the intersection of two sets of polygons, a polygon being associated with a planar patch and a set of planar patches being associated with each quadric surface.

The resulting boundary representation is a description in terms of a surface list, an edge list, and a vertex list. Within these lists, each feature (surface, edge, or vertex) has a property list as well as a set of pointers linking it to topologically connected features. For example, associated with each surface is a list of edges bounding this surface.

The vision-oriented model. The vision-oriented model component of the two-part extended CAD representation requires every boundary feature to be explicitly listed under several classifications and evaluated as to its expected utility. For example, a linear edge appears in the two lists describing the edges bounding the two surfaces that meet and form this edge, in

the list of all edges and their parameters, in the two lists describing the two vertices delimiting the edge, and possibly in such lists as parallel or colinear edges, edges forming angles of the same value, edges with similar length, simultaneously visible edges, and so forth.

The derivation of these property lists is straightforward, with the exception of the lists of simultaneously visible features. In general, a set of simultaneously visible features is constrained by the geometry of the object and by the orientation of the image plane (the camera) relative to this object. Existing methods for determining such sets make use of hidden surface elimination algorithms. We can produce a list of visible features (or potentially visible features) for each orientation of the image plane in an object-centered frame where we properly sampled the space of possible orientations. Goad included such a procedure in a model-based recognition system.[9]

In our case, we constrained the object to lie on a horizontal plane. Therefore, we can associate a list of *potentially* visible features with each stable position of the object on the premise that the object's stable positions can be predicted. An object model (and not the physical object that it describes) has a finite number of stable positions. This is a direct consequence of the fact that curved surfaces have been approximated by planar patches, hence any object is treated as a polyhedron with a finite number of planar faces. A practical algorithm for computing the stable positions of a polyhedron is described below.

If we know the orientation of the camera relative to the tabletop in advance, we can further constrain the list of potentially visible features associated with every stable position. The task of the runtime system now reduces to hypothesizing a stable position (equivalent to matching a set of image features to a set of object features), computing the missing rotation and translations, and verifying (counting) how many predicted visible features are actually present in the image.

To sum up, the vision-oriented model has to include a list of all the stable positions of the object. We produce such a list as follows:

An object model may lie on any of the faces associated with its convex hull. The convex hull is a polyhedron containing among its faces all the object's convex faces. A convex face is such that the object must lie on one side of the plane defined by this face. Therefore, we first determine such a convex hull. The problem now reduces to determining the stable positions of the convex hull. Each one of its faces is constrained to lie flat on the tabletop; this constraint determines a potential stable position. The system verifies this stability by projecting the center of gravity of the object down onto the table and checking whether it falls within the boundaries of the supporting face. For an accepted stable position and for each possible viewing direction, the system establishes a list of visible features.

Figure 1 shows the global architecture of the modeling system including the three main representations: volumetric (solid), boundary, and the list of stable positions. Figure 2 shows a view of an object model (a), a view of the same object in one stable position (b), and a view of the object in the second stable position (c).

The planning system

The goal of the planning system is to derive a customized object representation from the general-purpose representation provided by the extended CAD model. For example, the task of recognizing an object with one stable position is quite different from the task of recognizing one with many stable positions. In the first case, only a list of features has to be determined and no a priori importance is given to a specific feature. In the latter case, it would be interesting to determine in advance which features best suit rapidly predicting a stable position. If the object has a three-centimeter circular hole on each one of its faces, this feature does not help recognition because the system cannot discriminate between the various faces of the object. We are interested in developing an algorithm that analyzes the object model and finds the object features best suited for a specific visual task.

Let us describe briefly what we mean by object recognition. It is a process that establishes a set of image-to-model correspondences. Both the model and the image are described in terms of the same features, namely edge segments such as straight lines and circular arcs. An image-feature-to-model-feature assignment is produced on the basis of similarity of their properties. Two such assignments are said to be mutually consistent if the relative position and orientation of the two model features are the same as the relative position and orientation of the two corresponding image features. Hence, looking for an image-to-model correspondence is equivalent to seeking sets of mutually consistent image-feature-to-model-feature assignments. This recognition process, as we will see, not only assigns an object model to a set of image features, but also determines the parameters of the rotational and translational transform that allow this object to overlap onto the image features, i.e., the camera-to-object transform.

Unfortunately, an image-to-model feature assignment is inherently ambiguous: Not only does an object have many similar features, these features may appear many times in the image. Object recognition based on feature-to-feature correspondence is therefore a search process.

The set of correspondences described above can be conveniently mapped into a graph representation.[10,3] A node in this graph represents an image-feature-to-model-feature assignment. An arc links two nodes whenever they represent two mutually consistent assignments. A clique in this graph is a completely connected subgraph, i.e., each node in the clique connects directly to all the other nodes in this clique. A maximal clique is a clique that cannot be extended to include other nodes of the graph. We can interpret a maximal clique as an instance of an image object.

Several algorithms locate maximal cliques in an undirected graph, including the one used by Bolles and Cain.[3] First we build the graph, and second, we perform an exhaustive search in order to make sure that we find all the maximal cliques. The largest maximal clique corresponds to the best available solution. Since the planning system may provide knowledge that can speed up the search, we need an algorithm that avoids systematic recourse to exhaustive search. Consider the following example. At some point of the search a new node is sought for inclusion in a clique. If the feature corresponding to this assignment does not increase the knowledge about the current match, or if this feature is very unlikely to occur in the image, or if it has many image assignments, it is probably too costly to include this node.

In order to overcome these difficulties, we use a hypothesize-and-test strategy implemented as a depth-first tree search (see also Bolles and Horaud,[6] Goad,[9] and Horaud[7]). We use a few nodes (one or two) to generate a hypothesis (a clique), then consider other nodes one at a time. If

a new node is considered beneficial and if it verifies the current hypothesis, the clique is extended. Otherwise, the algorithm backtracks to the last choice point. A choice point arises whenever a model feature has more than one image feature assignment. Since the features are not considered at random but according to their expected utility, we can abandon the search whenever a set of model features, even if not the largest maximal clique, defines with no ambiguity an object's appearance and its locational parameters. The tree is thus generated dynamically as the search goes on. It is important to make a clear distinction between the (graph) definition of a maximal clique and a particular algorithm that detects maximal cliques.

To summarize, in order to make the runtime tree search more efficient, the planning system should be able to provide answers to the following questions:

(1) How many features do we need to generate a hypothesis?

(2) Which criteria classify the features from the most useful to the least distinctive?

(3) Which features are the most useful for each specific subtask: object recognition and pose (position and orientation) determination?

A three-dimensional object has six degrees of freedom: three rotations and three translations. Since the object is constrained to lie on a flat horizontal surface, three degrees of freedom remain to be determined: two translations and one rotation whose axis is perpendicular to the horizontal surface (the other two rotations and one translation being determined for each stable position and for a fixed camera-to-table transform).

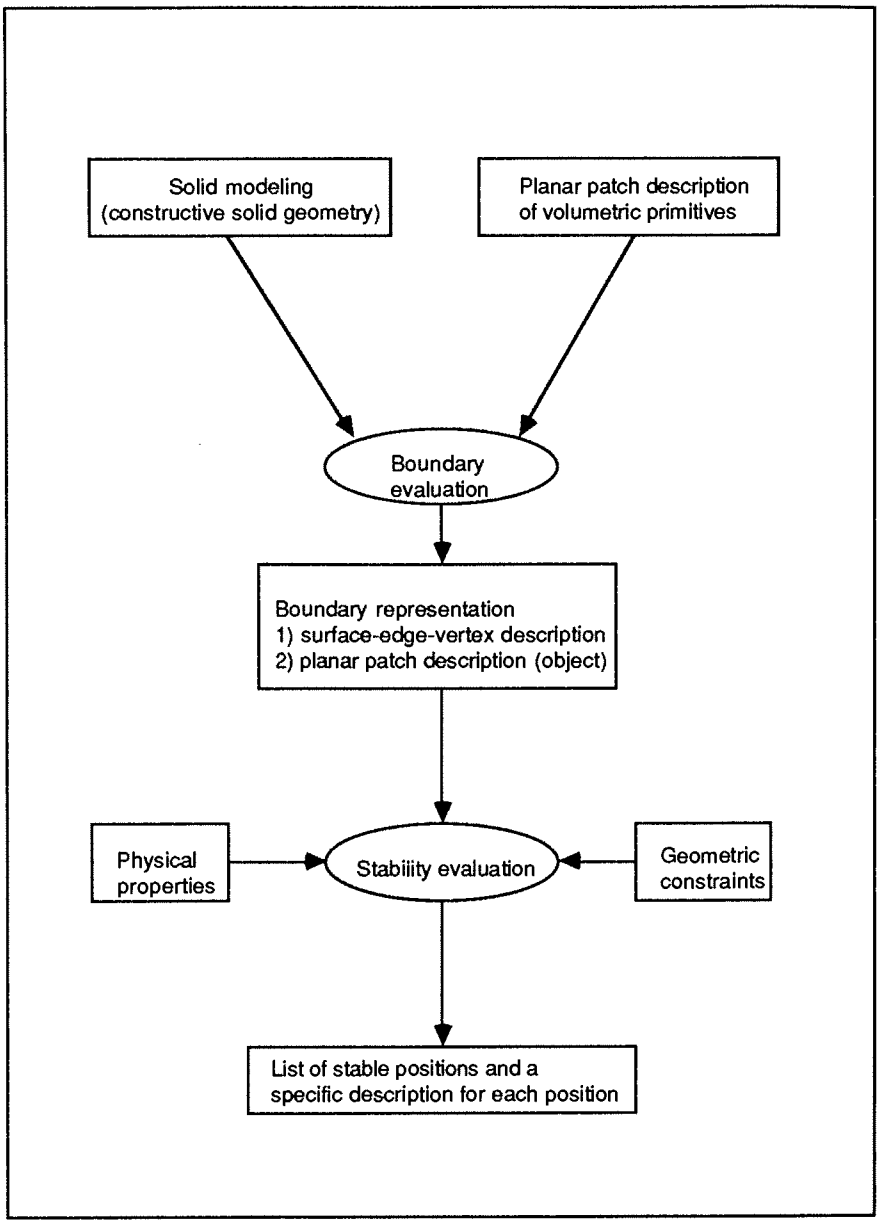

Figure 1. The architecture of the modeling system.

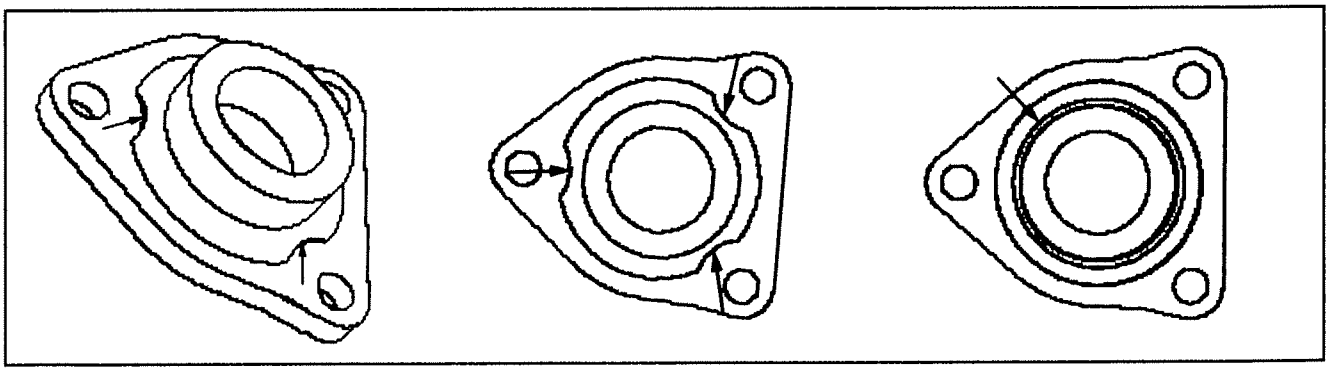

Figure 2. The geometric model of an object (a) and its stable-position sets of visible features (b) and (c). The arrows point to the most salient features characterizing each stable position as found by the planning system.

Consider now that a model linear segment is assigned to an image straight line. In practice, this is equivalent to moving the object model (rotating and translating the object) until the linear segment overlaps the image line. This will determine the rotation up to 180 degrees and the two translations. However, since an object may be partially occluded, some features are only partially visible. Bad lighting and preprocessing imperfections may cause the disintegration of one scene feature into several small image features. This means that we cannot reliably measure the length of the image line. We can determine one translation and one rotation exactly and estimate bounds on the second translation from an image-line-to-model-line assignment. Similar considerations lead to the conclusion that an image-arc-to-model-arc assignment determines two translations and leaves the rotation unknown. Nevertheless, the arcs should be considered first because they have one more intrinsic parameter than the lines, namely the radius.

To sum up, any two features (two lines, two arcs, or a line and an arc) not tied into a specific configuration such as concentric circles or parallel lines are sufficient for

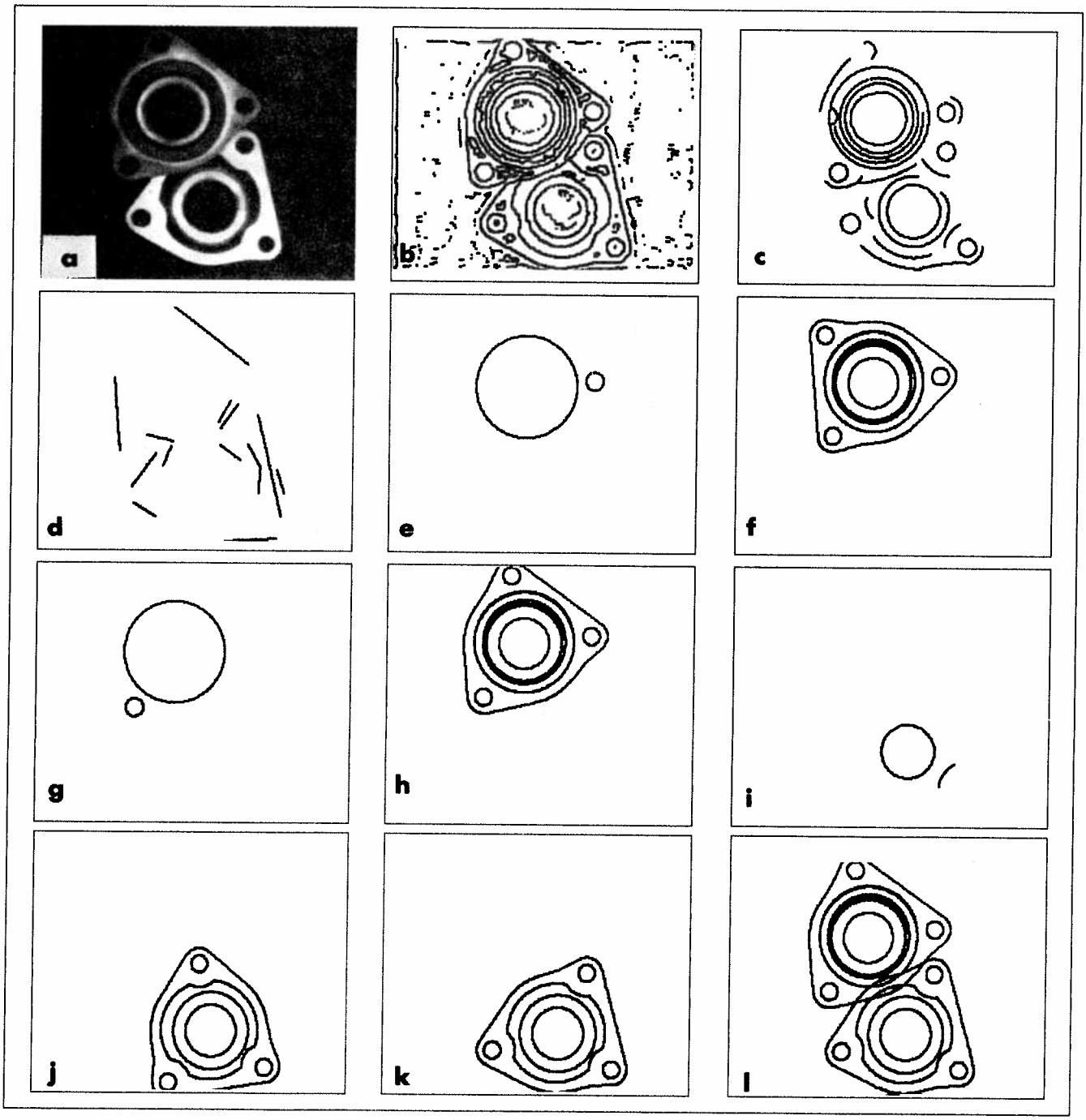

Figure 3. An example of the recognition strategy on a grey-level image.

generating a hypothesis. However, the configurations mentioned above (parallel and/or concentric features) may be very useful because their presence in the image confirms or denies a hypothesis without ambiguity.

We have implemented an algorithm that performs feature classification. A stable position of an object is characterized by a list of simultaneously visible features. First, decompose this list into four sublists:

- A sublist of circular edges (SList 1) and a sublist of linear edges (SList 2) visible for this stable position and hidden for the other stable positions.
- A sublist of circular edges (SList 3) and a sublist of linear edges (SList 4) visible for several stable positions including the stable position under consideration.

Second, score each feature within each such list according to the following criteria:

- Uniqueness: The features and feature configurations less frequent in the model are ranked first. Very often a stable position is characterized by a unique feature configuration. For example, a circle of radius three centimeters may appear on many faces, but such a configuration as two concentric circles, one of radius three centimeters and one of radius five centimeters, appears on one face only.
- Likelihood of being seen: Long lines are more likely to be seen in the image. Whole circles are more reliable than short circular arcs. Convex edges are more likely to be detected than concave ones.
- Discriminating power: Circular arcs have one more intrinsic parameter than lines—their radius. In theory, the radius of an image arc is independent of how much of the arc is actually seen. In practice, longer arcs are more reliable than short ones because the reliability of the least squares criterion used for approximating a set of edge points increases with the number of approximated data points.

The runtime system

The runtime system comprises two processes: image segmentation and image-to-object matching. We use standard methods for performing edge detection, linking, and segmentation. The straight lines and circular arcs thus detected are further classified according to their expected utility. The criteria used for this *runtime classification* are the same as the ones used for off-line planning.

Table 1. The results of recognition with planning.

Hypothesis	Expanded Nodes	Matched Features	CPU	Decision
1	655	8	1.38	Rejected
2	70	10	0.14	Accepted
3	108	7	0.30	Rejected
4	89	10	0.15	Accepted
Total	922	—	1.98	—

Next, we seek image-to-object matches using a tree search procedure. This strategy embeds a standard hypothesize-predict-verify-update loop implemented as a depth-first tree search. The *hypothesize step* selects two mutually consistent image-feature-to-model-feature assignments not yet considered. This two-node clique determines a stable position and the missing parameters (one rotation and two translations). Hence, there is a list of visible model features associated with the current hypothesis as well as an object-to-camera transform. If the object-to-camera transform associated with the current hypothesis equals any of the already considered hypotheses (whether they failed or not), the current hypothesis is rejected and a new one is generated.

The *predict step* considers a visible model feature not yet included in a match and considered beneficial by the off-line planning—next in the hierarchical ordering of the lists of visible features. The *verify step* projects the predicted model feature onto the image using the object-to-camera transform available with the current hypothesis and checks the list of image features to see whether any has the predicted qualities. If such a feature is found, the *update step* uses its image position to refine the parameters of the current hypothesis and the current clique is expanded to include this new node. If such a feature is not found, the algorithm backtracks to the last choice point.

We have implemented a certain number of heuristics asserting either that an object has been found or that an object cannot be associated reliably with the current hypothesis. In either situation, we abandon the search and generate a new hypothesis.

Let N be the number of visible features associated with a stable position being hypothesized and tested. One way to evaluate the match is to compare the size of the clique with the size of the minimum set of features that we need to detect in order to assert that we have found a stable position. Let M be the size of this minimum set. We have $M \leq N$. It is quite difficult to evaluate M analytically. In all our experiments we set $M = N/2$.

Another criterion for evaluating a match is an error measure between image and model features:

$$\text{error} = \sum_{i=1}^{n} \frac{(m_i - l_i)^2}{n}$$

where m_i is the length (the arc length for a circular arc) of a model feature, l_i is the length of the image feature assigned to the model feature within a match, and n is the size of the clique (the number of nodes in the match).

The *heuristic for accepting a match* requires that n is greater than or equal to M and that the error measure is less than an allowed tolerance. The *heuristic for rejecting a match* follows: If, at any stage of the search, the number of nodes in the current match (n) augmented by the number of model features that remain to be verified is smaller than M, the search is abandoned.

Experimental results

The recognition strategy described above has been applied to the image of Figure 3a. The edges (b) are approximated by circular arcs (c) and straight lines (d). Two selected image features (e) are used to generate the first hypothesis (f), which is rejected after expanding 655 nodes. The next two image features (g) generate the second hypothesis (h), which is accepted after expanding 70 nodes. Two new image features are selected (i). A third hypothesis (j) is rejected, while a fourth hypothesis is accepted (k). Finally, the system correctly recognized two objects (l).

Table 1 summarizes the experimental results obtained using off-line planning. For each generated hypothesis, Table 1 indicates the number of expanded nodes, the number of matched features, and the CPU time (in seconds) on a VAX 11/780. The time estimates do not include low-level image segmentation.

Table 2. First image: recognition with and without planning.

Example 1	Hypotheses	Expanded Nodes	Detected Objects	CPU
With Planning	4	922	2	1.98
No Planning	6	2326	2	4.89

Table 3. Second image: recognition with and without planning.

Example 2	Hypotheses	Expanded Nodes	Detected Objects	CPU
With Planning	16	1423	3	3.14
No Planning	15	1404	3	3.13

Table 4. Third image: recognition with and without planning.

Example 3	Hypotheses	Expanded Nodes	Detected Objects	CPU
With Planning	23	8125	3 (one mismatch)	15.17
No Planning	37	14389	3 (two mismatches)	29.59

Table 2 compares the global results of recognizing the objects of Figure 3 in the presence of off-line planning with the results obtained with the same runtime strategy omitting the planning process. In this case, the planning increases the speed of the recognition by a factor of 2.5.

The recognition strategy has been tested on many different objects. Figure 4a shows three overlapping parts and Figure 4b shows the result of recognition. In this case, the global results are almost the same with or without planning (see Table 3). This is explained by the fact that the features allowing one face to be distinguished from the other are less likely to be seen in the image than the features that are identical for both faces.

The results obtained with the last example (Figure 4c and 4d) are shown in Table 4. Here again the planning procedure increases the speed of recognition by a factor of 2.

In this last example, the algorithm correctly found two objects and misinterpreted the third one: the locational parameters are correct, but the stable position is not. This occurs because, first, the two stable positions exhibit almost the same feature configurations and, second, the recognition process is a trade-off between the ability to make a quick decision and the correctness of this decision. In the case of an exhaustive search, correctness is maximal, but the computation time lies well beyond acceptable limits.

Any model-based vision system confronts the combinatorial explosion of its search space. This is an inherent problem with this approach. We believe that "image feature grouping" is the key to an efficient reduction of the search space, and we plan to investigate general-purpose feature grouping techniques.

In the future we plan to implement models of more objects and to evaluate our system more deeply. However, we do not believe that a single technique will be general enough to recognize a large class of objects efficiently. For this reason our system and other systems, such as LFF,[3] Hyper,[5] and 3DPO[6] complement each other. □

References

1. W.A. Perkins, "A Model-based Vision System for Industrial Parts," *IEEE Trans. Computers*, Feb. 1978, Silver Springs, Md., pp. 126-143.
2. J.L. Turney, T.N. Mudge, and R.A. Volz, "Recognizing Partially Occluded Parts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, July 1985, Silver Springs, Md., pp. 410-421.
3. R.C. Bolles and R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *Int'l J. Robotics Research*, Fall 1982, MIT Press, Cambridge, Mass., pp. 57-82.
4. P. Rummel and W. Beutel, "Workpiece Recognition and Inspection by a Model-based Scene Analysis System," *Pattern Recognition*, Pergamon Press, Elmsford, N.Y., 1984, pp. 141-148.
5. N. Ayache and O.D. Faugeras, "Hyper: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Jan. 1986, Silver Springs, Md., pp. 44-54.
6. R.C. Bolles and R. Horaud, "3DPO: A Three-Dimensional Part Orientation System," *Int'l J. Robotics Research*, Fall 1986, MIT Press, Cambridge, Mass., pp. 3-26.
7. R. Horaud, "New Methods for Matching 3D Objects with Single Perspective Views," *IEEE Trans. Pattern Analysis and Machine Intelligence*, May 1987, Silver Springs, Md., pp. 401-412.
8. C.M. Brown, "Some Mathematical and Representational Aspects of Solid Modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, July 1981, Silver Springs, Md., pp. 444-453.

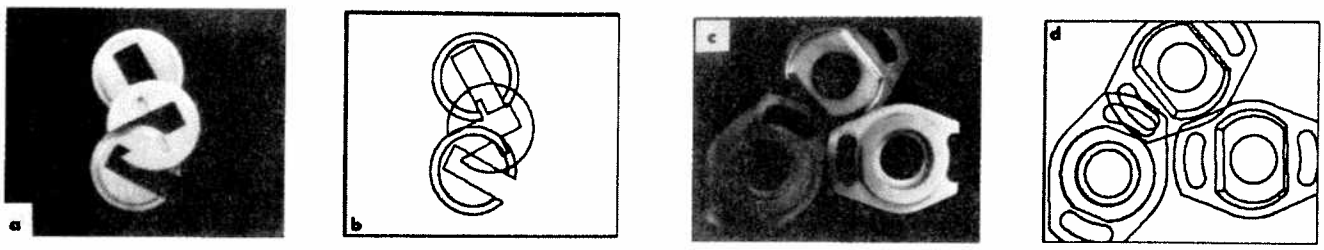

Figure 4. Two more examples.

9. C. Goad, "Special Purpose Automatic Programming for 3D Model-based Vision," *Proc. Image Understanding Workshop*, June 1983, Science Applications Inc., DARPA, Arlington, Va., pp. 94-104.
10. R.C. Bolles, "Robust Feature Matching Through Maximal Cliques," *Proc. SPIE Technical Symp. Imaging Applied to Automated Industrial Inspection and Assembly*, April 1979, Washington, D.C.

Radu Horaud is currently a research scientist at the Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle (LIFIA) in Grenoble. At the moment his research interests include computer vision, computational geometry, 3D object recognition and modeling.

Horaud received both the Diplôme d'ingénieur and the Diplôme de docteur-ingénieur from the Ecole Nationale Supérieure d'Ingénieurs Electriciens de Grenoble, France, in 1977 and 1981, respectively. He spent two years as an International Fellow at SRI International in the robotics department.

Thomas Skordas has since 1985 been with the Laboratoire d'Electronique et de Technologie de l'Informatique (LETI) in Grenoble, France. His current research interests include computer vision and graph theory.

Skordas received the Diploma in Electrical Engineering from the University of Salonica, Greece, in 1984. He is currently preparing a doctorate degree in computer science at the Institut National Polytechnique de Grenoble.

Readers may write to Horaud at LIFIA, BP 68, 38402 Saint-Martin d'Hères, Cedex, Grenoble, France. Readers may write to Skordas at D-LETI, Ave. des Martyrs, 85X, 38041 Grenoble, France.