Computer Vision and Image Understanding **79,** 364–407 (2000)
doi:10.1006/cviu.2000.0865, available online at http://www.idealibrary.com on IDEAL®

3D Object Recognition and Pose with Relational Indexing[1]

Mauro S. Costa

Boeing Commercial Airplane Group, Seattle, Washington
E-mail: mauro.s.costa@boeing.com

and

Linda G. Shapiro

University of Washington, Seattle, Washington
E-mail: shapiro@cs.washington.edu

Received September 14, 1998; accepted June 1, 2000

This paper addresses the problem of recognizing 3D objects from 2D intensity images. It describes the object recognition system named RIO (relational indexing of objects), which contains a number of new techniques. RIO begins with an edge image obtained from a pair of intensity images taken with a single camera and two different lightings. From the edge image, a set of new high-level features and relationships are extracted, and a technique called relational indexing is used to efficiently recall 2D view-class object models that have similar relational descriptions from a potentially large database of models. Once a model has been hypothesized, pairs of 2D–3D corresponding features, including point pairs, line–segment pairs, and ellipse–circle pairs, are used in a new linear pose estimation framework to produce a hypothesized transformation from a 3D mesh model of the object to the image. The transformation is either accepted or rejected by a verification procedure that projects the 3D model wireframe to the image and computes a Hausdorff-like distance measure between the projected model and the edge image. The resultant object recognition system is able to recognize 3D objects having planar, cylindrical, and threaded surfaces in complex, multiobject scenes.

[1] This research was supported by the National Science Foundation under Grant IRI-9023977, by the Washington Technology Center, and by the Boeing Commercial Airplane Group.

1077-3142/00 $35.00

1. INTRODUCTION

Three-dimensional object recognition has seen a great deal of activity in the past decade, as has been pointed out in recent surveys [3, 6, 13, 46, 53]. Most systems fall into three main categories: (1) systems that use intensity data alone [1, 7, 9, 11, 35, 42, 43, 48, 59], (2) systems that use range data alone [5, 10, 24, 26, 27, 38, 30, 37, 40, 50], and (3) systems that use both range and intensity (sometimes including color) data [31, 34, 52].

In intensity-image-based systems, points and straight line segments are still the most commonly-used features. In fact, the recent popularity of the alignment method [36] has led to a significant number of systems that blindly match triples of points or line segments from the image to similar triples from the model, using little or no contextual information. These algorithms all make the assumptions that (a) the points or line segments are reasonably reliable features of the class of objects to be recognized or located and (b) the pose of the object can be uniquely determined from a small set of these features. These assumptions are only true for polyhedral objects or those with a number of sharp, straight edges. They fall apart for most curved-surface objects. Figure 1a shows a polyhedral object where points and line segments make good features, and Fig. 1b shows another simple object with both curved and planar surfaces where they are not very useful. Our system can recognize objects that have planar, cylindrical, and threaded surfaces, and it is designed to handle occlusion.

In range-image-based systems, primitive surfaces (usually planar or quadric) are the most common features, but 3D line segments and points are also used. Because surfaces from range data are more reliable than surface regions from grayscale, a number of systems use the properties of and relationships among surfaces in matching algorithms. This type of approach has worked well for simple objects with a small number of simple surfaces. Systems that work with more complex, free-form surfaces generally look for interest points and perform point matching. Again, the reliable detection of feature points is crucial to success. The recent work of Johnson and Hebert [40] provides a new and powerful way to hypothesize point correspondences.

This work addresses the problem of recognizing 3D objects from 2D intensity images. It describes the object recognition system named RIO (relational indexing of objects), which

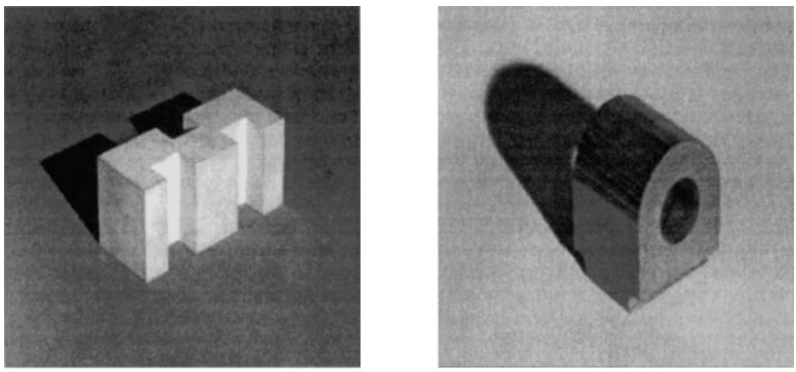

FIG. 1. (a) Image of a polyhedral object whose junctions and line segments make useful features for recognition and pose estimation. (b) Image of a nonpolyhedral object for which line segments and junctions alone are virtually useless as recognition features.

performs feature-based alignment and contains a number of new techniques. RIO begins with an edge image obtained from a pair of intensity images taken with a single camera and two different lightings. From the edge image, a set of new high-level features and relationships are extracted, and a technique called *relational indexing* is used to efficiently recall 2D view-class object models that have similar relational descriptions from a potentially large database of models. Once a model has been hypothesized, pairs of 2D–3D corresponding features, including point pairs, line–segment pairs, and ellipse–circle pairs, are simultaneously used in a new linear pose estimation framework to produce a hypothesized transformation from a 3D mesh model of the object to the image. The transformation is optimized (and subsequently accepted or rejected) by a verification procedure that projects the 3D model wireframe onto the image and minimizes a Hausdorff-like distance measure between the projected model and the image edges. The resultant object recognition system is able to recognize 3D objects having planar, cylindrical, and threaded surfaces in complex, multiobject scenes.

This paper describes the major research contributions of the RIO system. Section 2 discusses the related literature. Section 3 defines the features and relations used for recognition. Section 4 gives the relational indexing algorithm and some initial experiments. Section 5 describes the new pose-estimation algorithm, and Section 6 discusses the full experiments and results.

2. RELATED WORK

This section briefly describes a few existing systems which are closely related to the system developed in this work. Whenever possible, an attempt has been made to compare or relate characteristics of the system being described to the philosophical aspects of the work described in this paper.

Features can be predicted analytically or by applying graphics software to CAD models [9, 29, 30, 56]. In [30], Gremban and Ikeuchi use the term appearance-based vision to refer to methods in which the recognition system analyzes and predicts the appearances of the object models based on CAD data and on physical sensor models. The prediction can be either analytical or based on synthesized images of the objects in the model database. The predicted appearance is the set of features that are visible under a specific set of viewing conditions. The analysis of the predicted appearance allows for the generation of an object recognition program to be used in the online phase of the recognition process. This process is also called VAC (vision algorithm compiler), because it takes a set of object and sensor models and outputs an executable object recognition program. The framework is general in the sense that it does not require any specific type of sensor. Their system has successfully recognized simple objects from range data in a bin-picking environment. However, there are two drawbacks to this approach: (1) analytical prediction is impractical in some domains; and (2) synthetic images are not yet realistic enough for general use. More recently, Dorai and Jain [22] have developed a method of view grouping for free from objects, using range images.

The use of synthetic images also affected the performance of the PREMIO system of Camps *et al.* [9]. This system utilizes artificially rendered images to predict object appearances under various environmental conditions (sensor, lighting, and viewpoint location). The predictions generated by the system did not agree well with the real images acquired under the same set of conditions. In order to improve PREMIO's predictions, Pulli [47]

developed the TRIBORS system. He initially attempted to improve the predictions by using a better ray tracer, but that was also insufficient. The solution he found was to bootstrap the prediction process with synthetic images and to train on real images. These new predictions led to better and faster object recognition. The use of real training images seems to be a step in the right direction. In this paper, the "predictions" are derived exclusively from real images of the objects. These predictions are described in detail in Section 2.

Despite the fact that it only deals with two-dimensional objects, Bolles and Cain's local-feature-focus method [4] is very relevant to the work herein. Their method automatically analyzes the object models and selects the best features for recognition. Typical features include holes and corners. The basic principle is to locate one relatively reliable feature and use it to partially define a coordinate system within which a group of other key features is located. If enough of these secondary features are located and if they can uniquely identify the focus feature, then the hypothesized position and orientation of the object (of which this feature is a part) is determined. A verification step that utilizes template matching is then performed to prove or disprove the hypothesis. The system has been proven to efficiently recognize and locate a large class of partially visible two-dimensional objects.

The work of Murase and Nayar [43] also involves appearance of objects and the training is performed on real images. They argue that since the appearance of an object is dependent on its shape, its reflectance properties, its pose in the scene, and the illumination conditions, the problem of recognizing objects from brightness images is more a problem of appearance matching than of shape matching. They define a compact representation of object appearance that is parameterized by pose and illumination only, since shape and reflectance are intrinsic (constant) properties. This representation is obtained by acquiring a large set of real images of the objects under different lighting and pose configurations and then compressing the set into an eigenspace. A hypersurface in this space represents a particular object. At recognition time, the image of an object is projected onto a point in the eigenspace and the object is recognized based on the hypersurface on which it lies. The exact location of the point determines the pose of the object. The major drawback of this method is that it cannot handle multiple-object scenes. Occlusion also adversely affects the performance of the system.

Though the work of Bergevin and Levine [2] on generic object recognition does not make use of the specific model-based paradigm, it is philosophically related to the work herein. They utilize coarse, qualitative models that represent classes of objects. Their work is based on the recognition by component (RBC) theory of Biederman. The system is divided into three main subsystems: part segmentation, part labeling, and object model matching. The part segmentation algorithm is boundary-based, and it is independent of the specific shape of the parts making up an object. The part (geon) labeling algorithm makes use of the concept of faces to further categorize the geons into generalized solids. At the matching stage, the labeled geons are used to index into the database of models. A measure of similarity is defined in order to discriminate among the models. An important observation made by the authors themselves is that it is not clear that suitable line drawings may eventually be obtained from real images. All their examples and tests have made use of ideal line drawings.

The evidence-based recognition technique proposed by Jain and Hoffman [38] defines an object representation and a recognition scheme based on salient features in range images. The objects are represented in terms of their surfaces, boundaries, and edges. The recognition scheme makes use of an evidence rulebase, which is a set of evidence conditions and their corresponding weights for various models in the database. The similarity between a set

of observed image features and the set of evidence conditions for a given object determines whether there is enough evidence that the particular model is in the image. The model features must be carefully chosen in order to make possible the distinction between object classes.

Geometric hashing [15, 41] is a matching scheme that achieves rapid online matching performance via a large offline preprocessing step. In the offline database creation step of geometric hashing, salient feature points in the object models are converted into an affine-invariant model representation by using three points as a basis and transforming the coordinates of the remaining points. These new coordinates are encoded and used as an entry to a hash table where the basis triplet and the model from which the coordinates came are recorded. This is done for all possible basis triplets in the models. In the matching step, salient points in the image are detected, a basis triplet is chosen, and the remaining points are transformed with respect to this basis and are used to access the table. For each bin accessed, votes are cast for the model-basis pairs associated with the bin. After all the points are used in this fashion, if a particular model-view has high enough votes, an object match hypothesis is declared found. The recognition part of the system described in this paper is accomplished by utilizing the high-level features and relationships of an object in a paradigm called relational indexing. This indexing technique is related to the original geometric hashing technique, except that the database of models is indexed by encoding (without quantization) small relational graphs of features, as opposed to affine-invariant point coordinates. Because we use symbolic information and because the information we store is much smaller than the information required for geometric hashing, our current implementation uses only an array for table lookups, instead of a hash table.

The work of Stein and Medioni [50] is particularly related to our indexing scheme. In their structural indexing technique boundaries of objects are approximated by polygons and groups of consecutive segments are encoded and used to index the database and to retrieve possible hypotheses. In their MULTIHASH system, Grewe and Kak [31] propose an interactive framework for learning the structure of a multiple-attribute hash table for use in the recognition and localization of 3D objects. The system makes use of both qualitative and quantitative attributes, such as shape of a surface and color, respectively. Decision trees and uncertainty modeling are used in the construction of the hash tables, after a human trainer shows objects to the vision system and tells the system the identities of the models corresponding to the several attributes considered.

The work of Chen and Stockman [12] uses a hypothesize-and-test approach to generic object recognition. Their recognition system uses an alignment paradigm consisting of three stages: modeling, indexing, and matching. In the first stage, model aspects are constructed for predicting the object contours visible from an arbitrary viewpoint. Model aspects and pose hypotheses are generated by the indexing module, which makes use of the concept of "parts." In the matching stage, verification of model hypotheses is carried out by an alignment technique that makes use of Newton's method along with a Levenberg–Marquardt minimization, to estimate or refine the object pose iteratively. The hypotheses are refuted or supported by the matching results.

3. FEATURES AND RELATIONS

The features used in this work are derived from edge images. These edge images actually come from a pair of intensity images taken from the same viewpoint, but with the light source

on the left in one and on the right in the other. This aspect of the work will be discussed briefly when we describe the experimental system used to evaluate the algorithms.

3.1. Features

Once the edge image is obtained, it is used for feature extraction. The feature extraction module of the system makes use of the ORT (object recognition toolkit) public domain software package developed by A. Etemadi which extracts line segments and circular arc segments from edge images [23]. These primitive features are used to generate higher-level features such as pairs of parallel lines, junctions, triplets, clusters of coaxial arcs, and ellipses. All higher-level features are generated by the ORT package with the exception of ellipses and clusters of arcs. The procedures for detecting those features were developed in cooperation with Yu-Yu Chou and are reported in [14]. The complete list of features used in this work is given below, and their pictorial representation can be seen in Fig. 2.

- Ellipses
- Coaxials-3: cluster of three coaxial arcs
- Coaxials-multi: cluster of four or more coaxial arcs
- Parallel-far: pair of parallel lines 40 pixels or more apart
- Parallel-close: pair of parallel lines 40 pixels or less apart
- U-triples: set of three line segments in a U shape
- Z-triples: set of three line segments in a Z shape
- L-junctions
- Y-junctions
- V-junctions

3.2. Relationships Among Features

The relationships used for matching must be translation, scale, and rotation invariant. Note that they are relationships among two-dimensional features in a two-dimensional view class of the 3D object. Different features appear and different relationships among features hold in different view classes of the same object. The list of relations among features used in this work is given below.

- Share one arc
- Share one line
- Share two line
- Coaxial
- Close at extremal points
- Bounding box encloses
- Enclosed by bounding box

Each of the above relations is defined between a pair of features. All of the relations are symmetric relations, except for *bounding box encloses* and *enclosed by bounding box*, which are both one-way relations. Figure 3 shows examples of the above relations.

3.3. View-Class Models

A feature-based model of an object is a description of the object in terms of features that are detectable in real images of the object. A feature is *detectable* if there is a computer

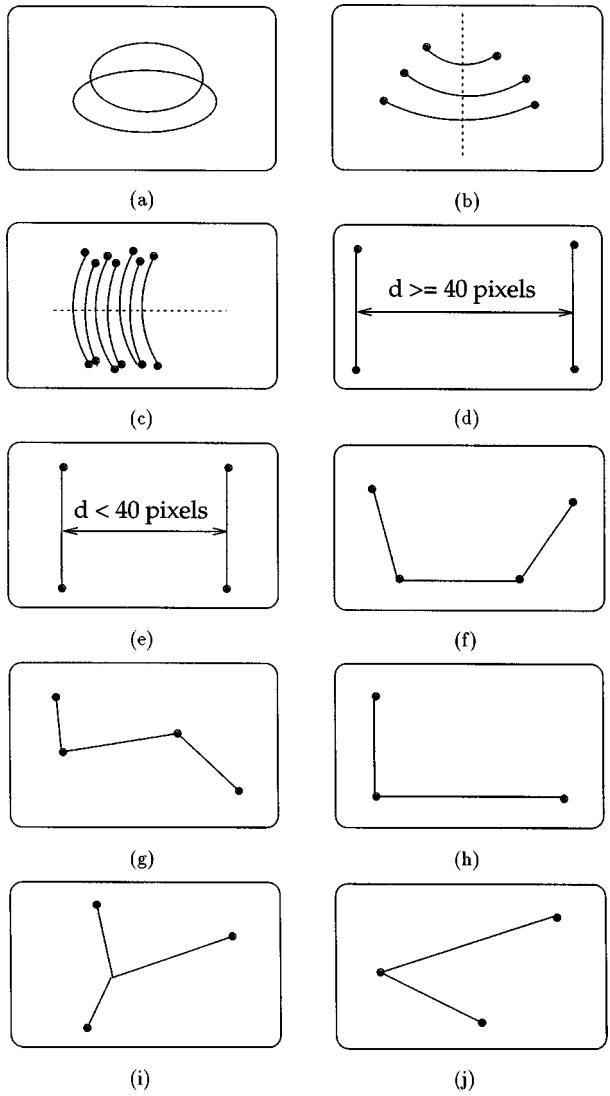

FIG. 2. Features used in this work. (a) Ellipses; (b) coaxials-3; (c) coaxials-multi; (d) parallel-far; (e) parallel-close; (f) U-triple; (g) Z-triple; (h) L-junction; (i) Y-junction; (j) V-junction.

program that can extract the feature from an image of the object, through some well-defined procedure. Feature-based models can be *full-object models* including all the features that appear in any view of an object or they can be *view-class* models in which an object is represented by a small set of characteristic views, each having its own distinct feature set. In this work, view-class models of industrial parts made of metal with planar, cylindrical, and threaded surfaces are utilized. The view classes used were generated manually by a human designer, but the same can be done automatically as proposed by Thornton in [54].

Let C be the class of objects to be recognized, and let T be the set of feature types to be used in the recognition task. Features may be 2D or 3D, depending on the sensors used to detect them. Each type of feature has a 3D source, an explanation for the appearance of the

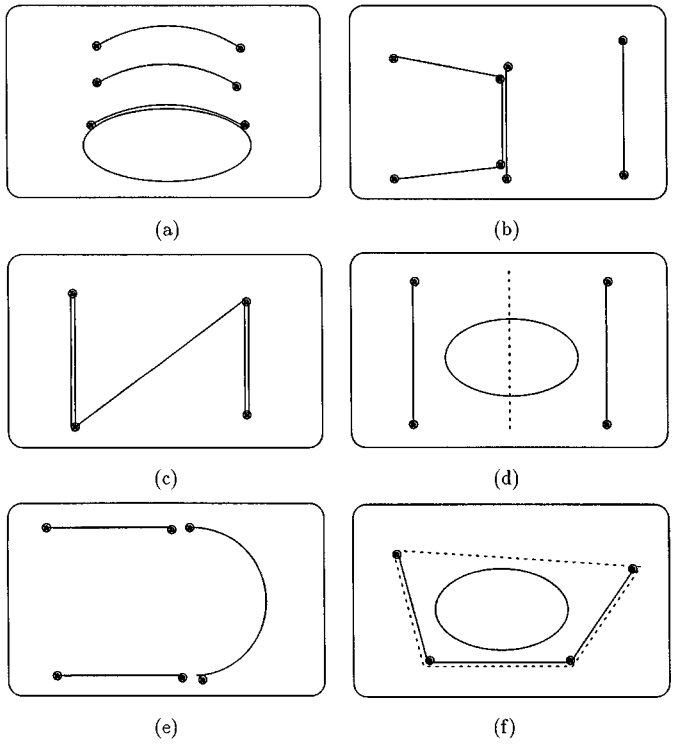

FIG. 3. Relations between sample pairs of features. (a) Share one arc; (b) share one line; (c) share two lines; (d) coaxial; (e) close at extremal points; (f) bounding box encloses/enclosed by bounding box.

feature in an image. In general, 3D image features correspond to 3D object features. For example, 3D surfaces detected in range images correspond to object surfaces, and 3D edges correspond to object edges or boundaries between objects. Some 2D features correspond directly to 3D features of the object. Straight and curved 2D segments, for instance, can correspond directly to straight and curved 3D edges. 2D ellipses can correspond directly to 3D circles or ellipses. Other 2D features come about due to a mixture of factors including the geometry of the object, the viewpoint, and the lighting. Limb edges are viewpoint dependent and have no corresponding 3D edge at all on the object. Highlight edges are caused by the shape and reflectance properties of the object and the lighting; they are also highly viewpoint-dependent.

Let $S_{V,M}$ be a set of training images for view class V of object model M. The images are all taken from a connected volume of the viewing sphere, which is assumed to be centered at the origin of the object and to be of a fixed radius equal to the maximum viewing distance. If the object has symmetries, then the connected volume can be replaced by the union of several connected volumes. Each image $I \in S_{V,M}$ is processed to yield a set of features F_I. A feature f_n^I from image I is *equivalent* to another feature f_m^J from image J if they have the same type and are judged to have come from the same 3D source. The set of features that represent the view class is the set $F_{V,M}$ of equivalence classes of the union of the feature sets. The feature types used in this work are coaxial circular arcs (two-cluster, three-cluster, and multicluster), ellipses, triples of line segments (U-shaped and Z-shaped), junctions (V-junction, T-junction, Y-junction, and Arrow), and parallel line segments (close and far apart).

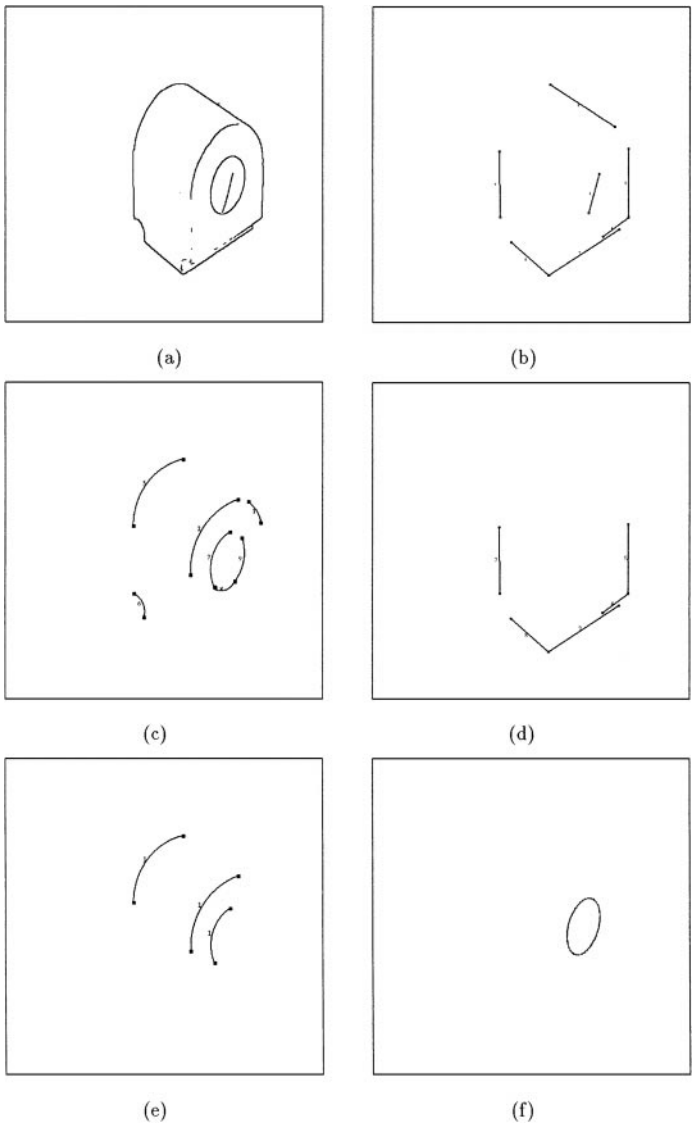

FIG. 4. A set of features extracted from the training image of Fig. 1b. (a) Edges; (b) straight lines; (c) circular arcs; (d) line features; (e) arc clusters; (f) ellipses.

Figure 4 illustrates some of the features that were extracted from the edge image of the nonpolyhedral object of Fig. 1b. Figure 4a shows the raw edge image. Figures 4b and 4c show the straight line segments and circular arcs extracted. Figure 4d shows the line features identified by the system. Line segments 5 and 7 form a far-apart, parallel pair, while line segments 8 and 9 form a V-junction as do line segments 5 and 6. Figure 4e shows the single cluster of three coaxial arcs detected by the system, and Fig. 4f shows the single ellipse detected.

The features shown in Fig. 4 are from one training sample of View Class 1 of Model 1 of the model database. Of the five samples of that view class that were analyzed, the three coaxial arcs and the ellipse were detected in all five; the V-junctions were detected in three;

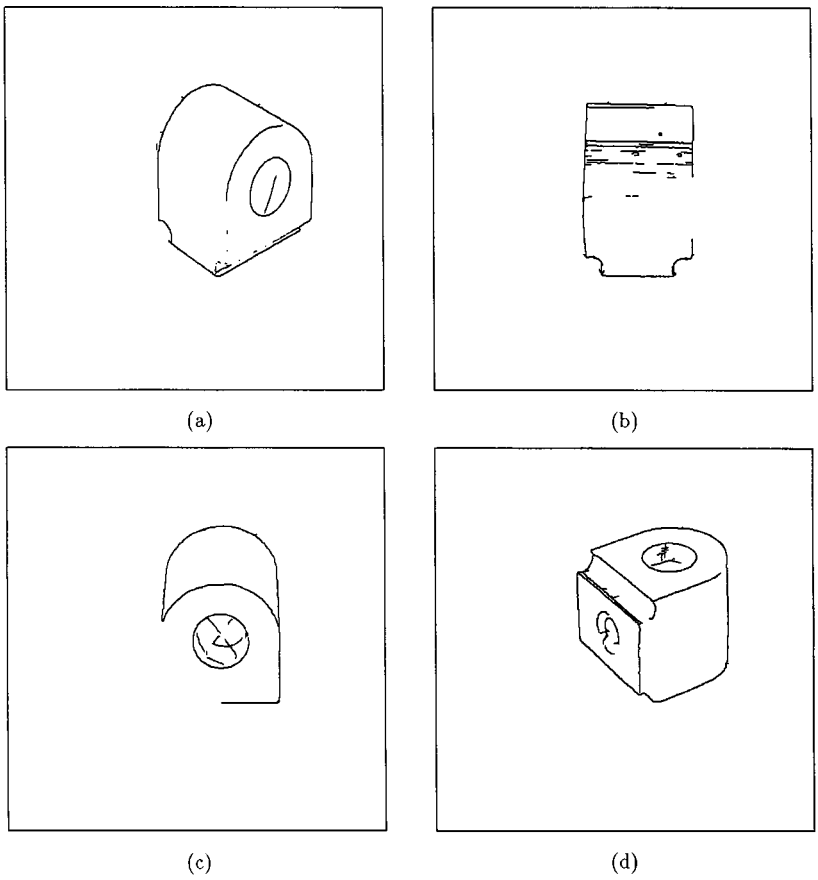

FIG. 5. Sample model views in the database. (a) Model 1 view class 1; (b) model 1 view class 2; (c) model 1 view class 3; (d) model 1 view class 4.

the far-apart, parallel lines were detected in three; another pair of parallel lines was detected in two; and a U-shaped triple of line segments was detected in two. Figure 5 illustrates samples of the edge images obtained from training images of the four view classes of Model 1.

3.4. 3D Mesh Models

In addition to the feature-based models described in the previous section, full geometric CAD models are utilized by the system. The CAD models are used in the verification and pose estimation steps as described in Sections 5.7 through 5.9.

4. RELATIONAL INDEXING FOR HYPOTHESIS GENERATION

In a model-based object recognition system [49], the task of matching image features to model features, in the general case, implies searching the space of all possible correspondences. Indexing is one of the techniques that has been utilized to reduce this search space. In recent years, several systems have made use of different approaches to indexing [8, 41, 44, 50]. In this section a novel approach to indexing into a database of models that makes

use of features and the relationships among them is described. This new technique is called *relational indexing*.

In relational indexing each view-class model in the database is described by a relational graph of all its features, but small relational subgraphs of the image features are utilized to index into the database and retrieve appropriate model hypotheses. In this section the use of this new technique for hypothesis generation is demonstrated.

4.1. Relational Indexing Notation

An *attributed relational description* D is a labeled graph $D = (N, E)$ where N is a set of attributed nodes and E is a set of labeled edges. For each attributed node $n \in N$, let $A(n)$ denote the attribute vector associated with node n. Each labeled edge $e \in E$ will be denoted as $e = (n_i, n_j, L_{i,j})$, where n_i and n_j are nodes of N, and $L_{i,j}$ is the label associated with the edge between them. $L_{i,j}$ is usually a scalar, but it can also be a vector.

A relational description $D = (N, E)$ can be broken down into subgraphs, each having a small number of nodes. Subgraphs of two nodes, called *2-graphs*, are considered. In the worst case, a complete graph of k nodes has $\binom{k}{2}$ 2-graphs, each consisting of a pair of attributed nodes and the labeled relationship between them. The relationship between the two nodes may be a meaningful spatial relationship or the null relationship *none*. The set of 2-graphs with nonnull relationships of a relational description D_l is denoted as T_l. Figure 6 illustrates a partial graph representing an object and all the 2-graphs for the given relational graph.

4.2. Relational Indexing Algorithm

Let $DB = \{M_1, M_2, \ldots, M_m\}$ be the database of view-class models, where each $M_i = (N_i, E_i)$ is an attributed relational description. Let $D = (N, E)$ be a relational description that has been extracted from an image and T be the set of all 2-graphs of D. In order to

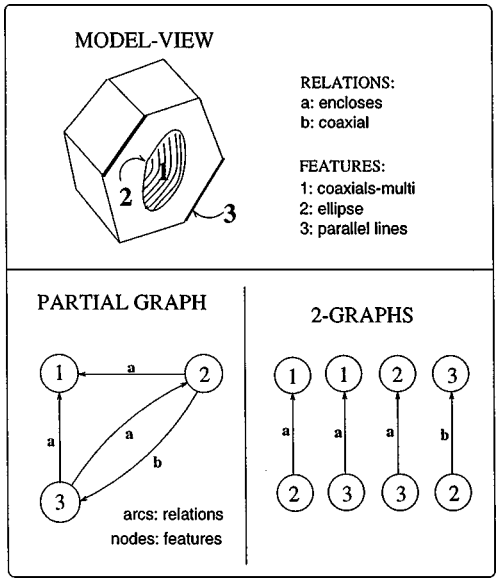

FIG. 6. Sample graph and corresponding 2-graphs for the hexnut object.

find the closest models to D two steps are performed: an offline preprocessing step that sets up the indexing mechanism and an online hypotheses generation step. The offline step is as follows. Let T^{M_i} be the set of 2-graphs of M_i. Each element $G_l^{M_i}$ in this set is encoded to produce an index $I_l^{M_i}$, which is used to access a lookup table. The bin corresponding to an encoded 2-graph G_l stores information about which models gave rise to that particular index. Whenever a particular 2-graph $G_l^{M_i}$ of model M_i produces an index that accesses a bin B, model M_i is added to the list of models in B. This encoding and storing of information in the lookup table is done offline and for all models in the database DB.

In the online step the relational indexing procedure keeps an accumulator A_i for each model M_i in the database; all the accumulators are initialized to zero. Each 2-graph G_l in T is encoded to produce an index I_l. The procedure then uses that index to retrieve from the precomputed lookup table all models M_i that contain a 2-graph that is identical to G_l. Identical means that the two nodes have the same attributes and the edge has the same label. For each 2-graph G_l of T, the accumulator A_i of every retrieved model M_i is incremented by one. After the entire voting process, the models whose accumulators have the highest votes are candidates for further consideration. Since the procedure potentially goes through all $\binom{k}{2}$ 2-graphs of T and for each one can retrieve a maximum of m models, the worst-case complexity is $O(m\binom{k}{2})$. However, the work performed on each model is very small, merely incrementing its accumulator by one. This is very different from methods that perform full relational matching on each model of the database. Furthermore, in real imaging applications, many of the 2-graphs have the null relationship and are not included in the voting process. The relational indexing algorithm is given below.

RELATIONAL INDEXING ALGORITHM.

Preprocessing (offline) Phase

1. For each model M_i in the database DB do:
   - Encode each 2-graph $G_l^{M_i}$ to produce an index.
   - Store M_i and associated information in the selected bin of the lookup table.

Matching (online) Phase

1. Construct a relational description D for the scene.
2. For each 2-graph G_l of D do:
   - Encode it, produce an index, and access the lookup table.
   - Cast a vote for each M_i associated with the selected bin.
3. Select M_i's with enough votes as possible hypotheses.

Since some models share features and relations, it is expected that some of the hypotheses produced will be incorrect. This indicates that a subsequent verification phase is essential for the method to be successful. It is important to mention that the information stored in the lookup table is actually more than just the identity of the model that gave rise to a particular 2-graph index. It also contains information about which specific features (and their attributes) are part of the 2-graph. This information is essential for hypothesis verification and eventual pose estimation.

4.3. Encoding, Indexing, and Voting Schemes

In the scope of this work, only subgraphs of size two nodes were used. This means that each index, in the general case, is made up of a combination of two features and two

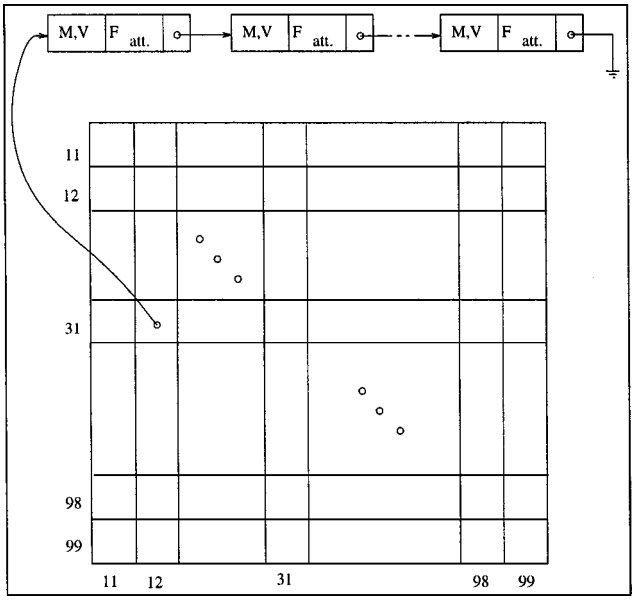

FIG. 7. Lookup table and indexing scheme.

relations, indicating the two-way relationship between the features. In the implementation, each feature type and each relation are represented by distinct labels (integers). Therefore, each subgraph index is uniquely represented by a 4-tuple of integers (f_1, f_2, r_1, r_2), which is used to access a lookup table, as shown in Figure 7. Each entry of the lookup table holds a linked list of all model-views that gave rise to the particular 2-graph corresponding to the table indices.

The overall voting scheme associated with the relational indexing technique is illustrated in Fig. 8. In the 2-graph shown, the ellipse and the coaxial cluster of arcs are related by the

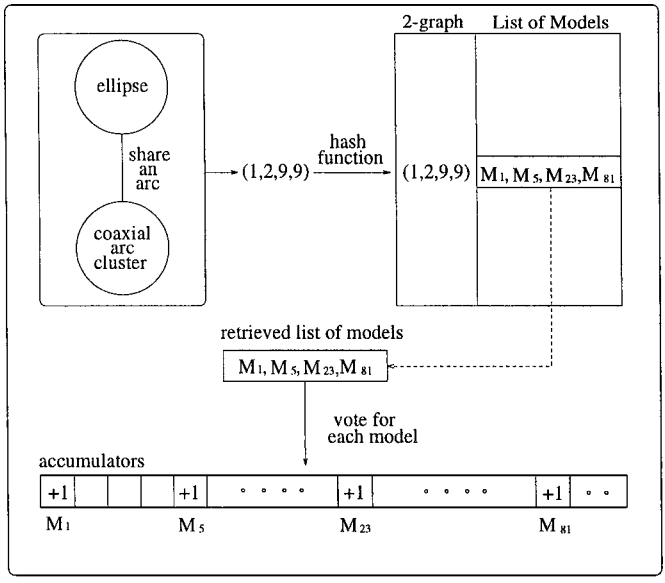

FIG. 8. Overall voting scheme.

two-way relation *share an arc*. The ellipse is represented by the label "1" while the coaxial arc cluster is represented by label "2". The relation *share an arc* is represented by label "9". Therefore, the particular 2-graph illustrated is uniquely represented by the 4-tuple (1, 2, 9, 9). This gives rise to the indices used to access the lookup table: ($x = 12, y = 99$). The indexed bin contains a list of all models in the database which contain the subgraph used to generate the indices. In the example shown, models M_1, M_5, M_{23}, and M_{81} are retrieved. The accumulators of these models are incremented by one, meaning that each model receives one vote from the specific 2-graph used.

4.4. Matching with Relational Indexing: An Example

This section illustrates an example of the experiments conducted to demonstrate the use of the relational indexing technique for 3D object recognition with feature-based models [16]. In order to illustrate the relational indexing technique, nine test images of both single and multiple object scenes were matched to the database of model-views. The nine test images used are shown in Fig. 9. The database of models was created by encoding all 2-graphs for each of the model-views. For each test scene, features and relations were detected, the relational description was built, and all 2-graphs were encoded. Relational indexing was then performed and the generated hypotheses were normalized by the number of 2-graphs in the original models and ranked in order of strength. Hypotheses that exceeded a preset strength threshold of 50% were dubbed "strong hypotheses." These hypotheses are to be passed to the verification procedure for further consideration. The results obtained for the nine test images are summarized in Table 1.

In each of the nine tests, the strong hypotheses were classified as type A, type B, or type C. Type A hypotheses are those where the correct model and the correct (closest) view class were identified. Type B hypotheses are those where the correct model was identified, but the chosen view class was not closest to the view in the image. Type B hypotheses can still be verified and used to determine pose if enough corresponding features are found. Type C hypotheses are those where an incorrect model was selected. These incorrect hypotheses

TABLE 1
Results of the Hypotheses Generation Process for Nine Test Scenes

Scene	Number of objects	Strong hypotheses		
		Type A	Type B	Type C
1	1	1	1	0
2	1	1	0	1
3	1	1	1	1
4	1	1	0	0
5	2	2	3	0
6	2	2	1	1
7	3	3	1	1
8	3	3	1	0
9	4	4	1	0

Note. Hypotheses types are as follows: A, correct model, correct view; B, correct model, incorrect view; C, incorrect model.

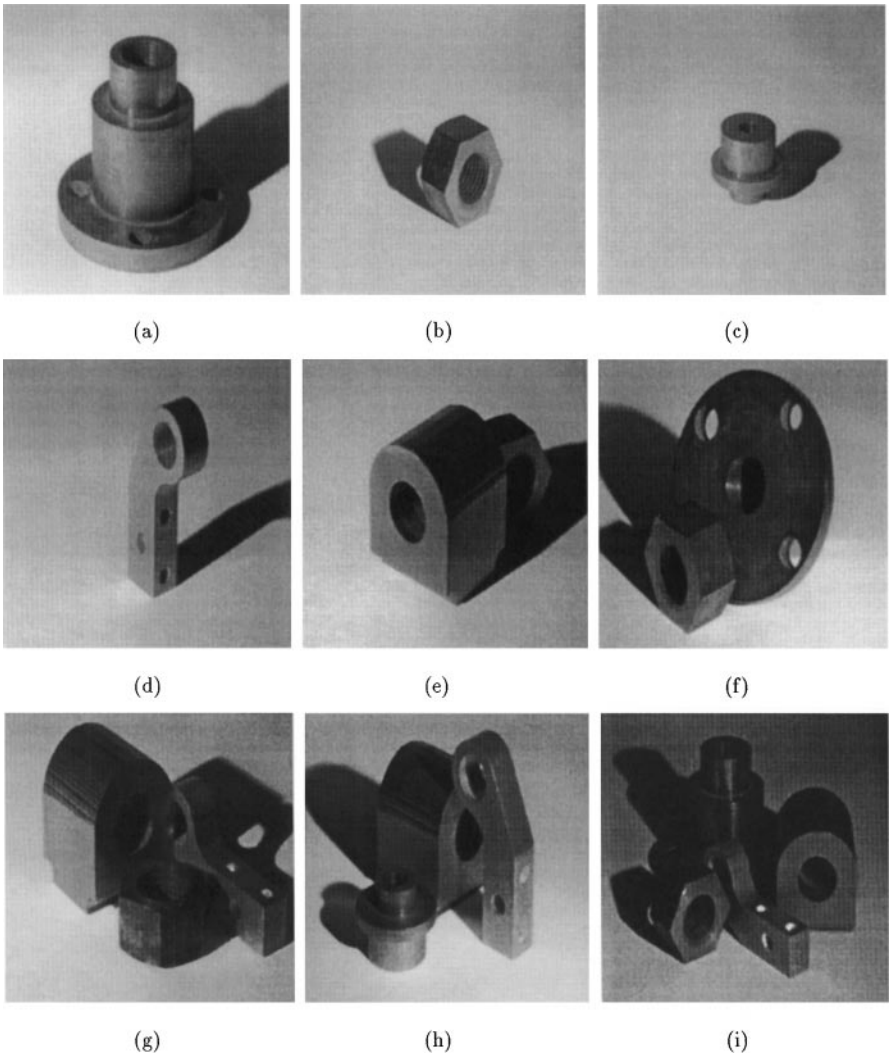

FIG. 9. The nine test scenes used. The labels left and right indicate the direction of the light source. (a) Image 1 (left); (b) image 2 (right); (c) image 3 (left); (d) image 4 (left); (e) image 5 (left); (f) image 6 (right); (g) image 7 (left); (h) image 8 (right); (i) image 9 (right).

should be ruled out in the verification step. The results of the nine tests are as follows: all the objects in the scenes have been correctly recognized (18 type A hypotheses); there were nine type B hypotheses and four type C hypotheses.

Figure 10a shows the results for test scene 9, which contains four objects: the stacked cylinder, the hexnut, the wrench, and the cylinder-block. The system produced five strong hypotheses; four were correct and are overlaid on the image. These hypothesized models were taken through pose computation (affine correspondence of model features and scene features) without verification. The fifth strong hypothesis (not shown) matched the object hexnut to an incorrect view of the correct object model. The subgraph indices shown in Fig. 6 were among those that were used in the matching process.

Figure 10b illustrates the correct (type A) hypotheses generated for test scene 5. Of the three type B hypotheses generated, one was for the cylinder-block object and two were for

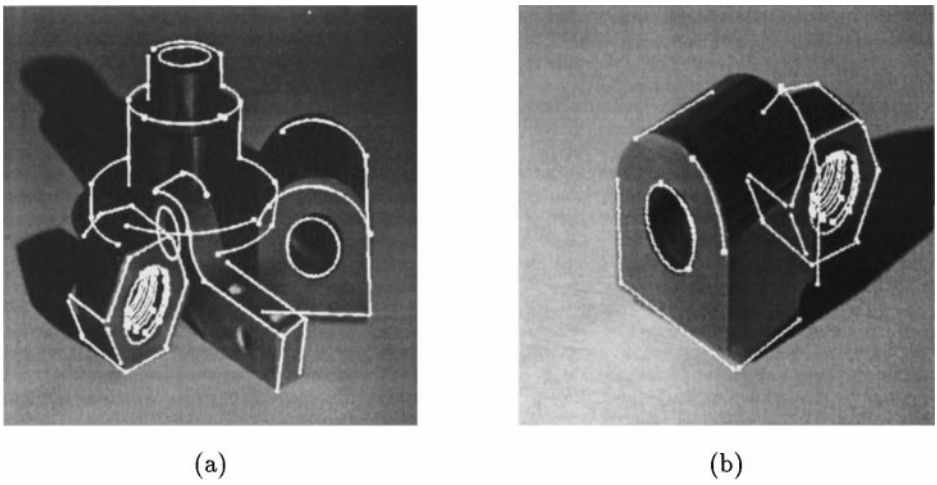

FIG. 10. (a) Right image of test scene 9 overlaid with the features of the hypothesized model matches. The objects in this scene are the stacked cylinder, the hexnut, the wrench, and the cylinder-block. (b) Left image of test scene 5 overlaid with the features of the hypothesized model matches. The objects in this scene are the cylinder-block and the hexnut.

the hexnut object, both of which are present in the scene. These are preliminary results; the full evaluation of the RIO system is described in Section 5.

5. POSE ESTIMATION USING MULTIPLE FEATURE TYPES

This section addresses the problem of estimating the position and orientation (pose) of an object given a set of some of its 3D features and the set of corresponding 2D image features, and its use in the context of hypotheses verification. In the most common case addressed in the literature, the feature sets are merely point sets. In a few cases line correspondences have been used as features to determine pose and so have ellipse–circle correspondences. But apart from the work of Phong [45] on computing pose from points and lines, a *linear* solution to computing pose from different types of feature correspondences *simultaneously* has never been addressed in the literature. Stockman's work on pose clustering [51] is one example of a method that can use the computed poses from different types of feature correspondences, but the pose computations are separate. Wong, Rong, and Liang [58] use points, line segments, and ellipse–circles for matching, but only points are used to compute pose; the other features are only used in the verification step.

In this section, a linear method is proposed to directly compute pose from point correspondences and ellipse–circle correspondences simultaneously. A new technique for linear pose computation from point correspondences is also presented, as well as a generalization of an existing algorithm for computing pose from an ellipse–circle correspondence, in order to handle nonrotationally symmetric objects.

5.1. The Camera Model

The model used is that of a pinhole camera, as illustrated in Fig. 11. It is assumed that the camera has been calibrated, that is, all the interior parameters are known. The camera

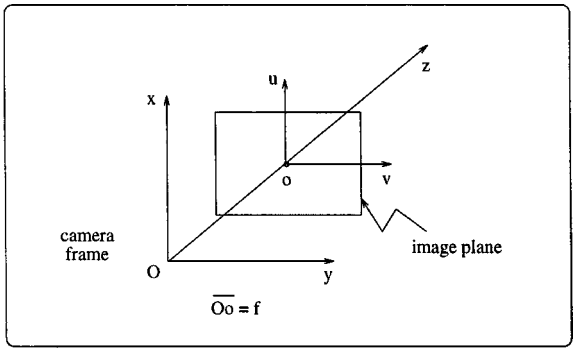

FIG. 11. The camera model used.

coordinate system is at the center of projectivity and its z-axis coincides with the optical axis. The optical axis is perpendicular to the image plane and it intersects that plane at the origin of the (u, v) image coordinate system. Thus, $u_0 = v_0 = 0$. The focal length is $f = 25$ mm.

5.2. Pose from 2D–3D Point Correspondences

The problem of determining 3D pose from sets of matched 2D image points and 3D object points has been vastly addressed in the literature [28, 55, 32]. This problem is inherently a nonlinear one, and nonlinear methods for estimating the pose parameters are necessary. However, under some conditions, an approximate, linear solution can be found.

Let (x, y, z) be the coordinates of model point P in its object coordinate system. Also, let the object coordinate system and the camera coordinate system be related by a transformation $T = \{R, t\}$, described in the form of a rotation matrix R and a translation vector t, where

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad \text{and} \quad t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}.$$

Then, the perspective projection of P onto the image plane yields image plane coordinates (u, v), where

$$u = f \frac{r_{11}x + r_{12}y + r_{13}z + t_x}{r_{31}x + r_{32}y + r_{33}z + t_z} \tag{1}$$

and

$$v = f \frac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z}, \tag{2}$$

and f is the focal length of the camera.

The transformation between object frame and camera frame corresponds to the pose of the object with respect to the camera frame. Thus, there are 12 unknowns: nine rotation matrix entries and three translation parameters. Since for each point there are two equations in the form of Eqs. (1) and (2), six 2D–3D point correspondences are needed to determine

all the pose parameters. The resulting system of equations is of the form

$$Bw = 0, \tag{3}$$

where

$$B = \begin{pmatrix} fx_1 & fy_1 & fz_1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ 0 & 0 & 0 & fx_1 & fy_1 & fz_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ fx_2 & fy_2 & fz_2 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & f & 0 & -u_2 \\ 0 & 0 & 0 & fx_2 & fy_2 & fz_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ fx_6 & fy_6 & fz_6 & 0 & 0 & 0 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ 0 & 0 & 0 & fx_6 & fy_6 & fz_6 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix} \tag{4}$$

and

$$w = (r_{11} \quad r_{12} \quad r_{13} \quad r_{21} \quad r_{22} \quad r_{23} \quad r_{31} \quad r_{32} \quad r_{33} \quad t_x \quad t_y \quad t_z)^T. \tag{5}$$

However, if one is interested in finding the true pose parameters, and not simply a transformation that aligns the projected model points well to the image points, conditions need to be imposed on the elements of R such that it satisfies all the criteria a true 3D rotation matrix must satisfy. In particular, a rotation matrix needs to be orthonormal: its row vectors must have magnitude equal to one, and they must be orthogonal to each other. This can be written as:

$$\begin{aligned} \|R_1\| &= r_{11}^2 + r_{12}^2 + r_{13}^2 = 1 \\ \|R_2\| &= r_{21}^2 + r_{22}^2 + r_{23}^2 = 1 \\ \|R_3\| &= r_{31}^2 + r_{32}^2 + r_{33}^2 = 1 \end{aligned} \tag{6}$$

and

$$\begin{aligned} R_1 \circ R_2 &= 0 \\ R_1 \circ R_3 &= 0 \\ R_2 \circ R_3 &= 0. \end{aligned} \tag{7}$$

In fact, theoretically, there is an infinite number of transformations of the form $T = \{R, t\}$ that will produce coordinates (u, v) for a given P that yield an acceptable "alignment," but there is only one $T = \{R, t\}$, for which R and t correspond to the true 3D position and orientation of the object relative to the camera frame.

It can be seen that the conditions imposed on R turn the problem into a nonlinear one. If the conditions on the magnitudes of the row vectors of R are imposed one at a time, and computed independently, a linear constrained optimization technique similar to the one used by Faugeras [25] can be used to compute the constrained row vector of R.

5.3. Linear Constrained Optimization

Given the system of equations (3), the problem at hand is to find the solution vector w that minimizes $\|Bw\|$ subject to the constraint $\|w'\|^2 = 1$, where w' is a subset of the elements of w. If the constraint is to be imposed on the first row vector of R, then

$$w' = \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \end{pmatrix}.$$

To solve the above problem, it is necessary to rewrite the original system of equations $Bw = 0$ in the following form

$$Cw' + Dw'' = 0,$$

where w'' is a vector with the remaining elements of w. Using the example above, i.e., if the constraint is imposed on the first row of R,

$$w'' = (r_{21} \quad r_{22} \quad r_{23} \quad r_{31} \quad r_{32} \quad r_{33} \quad t_x \quad t_y \quad t_z)^T.$$

The original problem can be stated as: minimize the objective function $O = Cw' + Dw''$, that is

$$\min_{w',w''} \|Cw' + Dw''\|^2, \tag{8}$$

subject to the constraint $\|w'\|^2 = 1$. Using a Lagrange multiplier technique, the above is equivalent to

$$\min_{w',w''} \left[\|Cw' + Dw''\|^2 + \lambda(1 - \|w'\|^2)\right]. \tag{9}$$

The minimization problem above can be solved by taking partial derivatives of the objective function with respect to w' and w'' and equating them to zero:

$$\frac{\partial O}{\partial w'} = 2C^T(Cw' + Dw'') - 2\lambda w' = 0 \tag{10}$$

$$\frac{\partial O}{\partial w''} = 2D^T(Cw' + Dw'') = 0. \tag{11}$$

Equation (11) is equivalent to

$$w'' = -(D^T D)^{-1} D^T C w'. \tag{12}$$

Substituting Eq. (12) into Eq. (10) yields

$$\lambda w' = [C^T C - C^T D(D^T D)^{-1} D^T C]w'. \tag{13}$$

It can be seen that λ is an eigenvector of the matrix

$$M = C^T C - C^T D(D^T D)^{-1} D^T C. \tag{14}$$

Therefore, the solution sought for w' corresponds to the smallest eigenvector associated with matrix M. The corresponding w'' can be directly computed from Eq. (12). It is important to notice that since the magnitude constraint was imposed only on one of the rows of R, the results obtained for w'' are not reliable and therefore should not be used. However, solution vector w'' provides an important piece of information regarding the sign of the row vector on which the constraint was imposed. The constraint imposed was $\|w'\|^2 = 1$, but the sign of w' is not restricted by this constraint. Therefore, it is necessary to check whether or not the resulting w' yields a solution that is physically possible. In particular, the translation t_z must be positive in order for the object to be located in front of the camera as opposed to behind it. If the element of vector w'' that corresponds to t_z is negative, it means that the magnitude of the computed w' is correct, but its sign is not, and it must be changed. Thus, the final expression for the computed w' is

$$w' = \text{sign}(w''_9)w'.$$

5.4. *Computing the Transformation* $T = \{R, t\}$

Row vector R_1 is computed first by computing w' as described above, since in this case $R_1 = w'$. Matrices C and D are

$$C = \begin{pmatrix} x_1 & y_1 & z_1 \\ 0 & 0 & 0 \\ x_2 & y_2 & z_2 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ x_6 & y_6 & z_6 \\ 0 & 0 & 0 \end{pmatrix} \tag{15}$$

and

$$D = \begin{pmatrix} 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ fx_1 & fy_1 & fz_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & 0 & f & -u_2 \\ fx_2 & fy_2 & fz_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ fx_6 & fy_6 & fz_6 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix}. \tag{16}$$

Then row vector R_2 is computed using the same technique, except that now the constraint is imposed on its magnitude; thus, $R_2 = w'$. In this case, matrices C and D are

$$C = \begin{pmatrix} 0 & 0 & 0 \\ fx_1 & fy_1 & fz_1 \\ 0 & 0 & 0 \\ fx_2 & fy_2 & fz_2 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ fx_6 & fy_6 & fz_6 \end{pmatrix} \tag{17}$$

and

$$D = \begin{pmatrix} fx_1 & fy_1 & fz_1 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ 0 & 0 & 0 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ fx_2 & fy_2 & fz_2 & -u_2x_2 & -u_2y_2 & -u_2z_2 & f & 0 & -u_2 \\ 0 & 0 & 0 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ fx_6 & fy_6 & fz_6 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ 0 & 0 & 0 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix}. \tag{18}$$

R_3 could also be computed the same way as R_1 and R_2 above, but that would not guarantee it to be normal to R_1 and R_2. Instead, R_3 is computed as follows:

$$R_3 = \frac{R_1 \times R_2}{\|R_1 \times R_2\|}. \tag{19}$$

All the constraints on the row vectors of R have been satisfied, except one: there is no guarantee that R_1 is orthogonal to R_2. In order to solve this undesired situation, R_1, R_2, and R_3 need to go through an orthogonalization process such that the rotation matrix R is assured to be orthonormal. This can be accomplished by fixing R_1 and R_3 as computed above and recomputing R_2 as:

$$R_2 = R_3 \times R_1. \tag{20}$$

This way, all the rotation parameters have been calculated and they all satisfy the necessary constraints. The translation vector t is computed using a least squares technique on a new, nonhomogeneous, overconstrained system of 12 equations:

$$At = b, \tag{21}$$

where

$$A = \begin{pmatrix} f & 0 & -u_1 \\ 0 & f & -v_1 \\ f & 0 & -u_2 \\ 0 & f & -v_2 \\ \vdots & \vdots & \vdots \\ f & 0 & -u_6 \\ 0 & f & -v_6 \end{pmatrix} \tag{22}$$

and

$$b = \begin{pmatrix} -f(r_{11}x_1 + r_{12}y_1 + r_{13}z_1) + u_1(r_{31}x_1 + r_{32}y_1 + r_{33}z_1) \\ -f(r_{21}x_1 + r_{22}y_1 + r_{23}z_1) + v_1(r_{31}x_1 + r_{32}y_1 + r_{33}z_1) \\ -f(r_{11}x_2 + r_{12}y_2 + r_{13}z_2) + u_1(r_{31}x_2 + r_{32}y_2 + r_{33}z_2) \\ -f(r_{21}x_2 + r_{22}y_2 + r_{23}z_2) + v_1(r_{31}x_2 + r_{32}y_2 + r_{33}z_2) \\ \vdots \\ -f(r_{11}x_6 + r_{12}y_6 + r_{13}z_6) + u_1(r_{31}x_6 + r_{32}y_6 + r_{33}z_6) \\ -f(r_{21}x_6 + r_{22}y_6 + r_{23}z_6) + v_1(r_{31}x_6 + r_{32}y_6 + r_{33}z_6) \end{pmatrix}. \tag{23}$$

5.5. Pose from Ellipse-to-Circle Correspondence

In order to find the 3D position and orientation of a circle from its projection onto the image plane (an ellipse in the general case) the solution described in [20] and [21] is utilized. The solution consists of a series of 3D transformations to the circle and it is carried out in two stages: first the orientation of the 3D plane on which the circle lies is determined; and then the center of the circle is computed.

An ellipse in the image plane is of the following form

$$a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 = 0. \tag{24}$$

Since one of the assumptions about the geometry is that the origin is at the principal point, this ellipse defines a cone in 3D,

$$a_1x^2 + a_2xy + a_3y^2 + \frac{a_4}{f}xz + \frac{a_5}{f}yz + \frac{a_6}{f}z^2 = 0, \tag{25}$$

where f is the focal length of the camera. Equation (25) above can be written in the form $X^TCX = 0$, where

$$C = \begin{pmatrix} a_1 & \frac{a_2}{2} & \frac{a_4}{2f} \\ \frac{a_2}{2} & a_3 & \frac{a_5}{2f} \\ \frac{a_4}{2f} & \frac{a_5}{2f} & \frac{a_6}{f^2} \end{pmatrix} \tag{26}$$

and

$$X = (x \quad y \quad z)^T. \tag{27}$$

In order to find the orientation of the plane on which the circle lies, it is necessary to first reduce the cone equation to

$$\frac{x^2}{b_1} + \frac{y^2}{b_2} + \frac{z^2}{f^2} = 0.$$

This is accomplished by a 3D rotation to the eigenvector frame such that C becomes a diagonal matrix of the form

$$C' = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}, \tag{28}$$

where λ_1, λ_2, and λ_3 are the eigenvalues of C in ascending order. The rotation matrix that accomplishes the above is given by

$$R_\lambda = (V_1 \quad V_2 \quad V_3), \tag{29}$$

where V_1, V_2, and V_3 are the eigenvectors of C. Then, another rotation about the new y-axis is performed such that the intersection of the cone with the plane $z = k$ is a circle. Thus,

the coefficients of x^2 and y^2 are identical. In this case, the normal to the circle becomes (0, 0, 1). This second rotation is of the form

$$R_\theta = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, \tag{30}$$

where

$$\theta = \pm\tan^{-1}\sqrt{\frac{\lambda_2 - \lambda_1}{\lambda_3 - \lambda_1}}. \tag{31}$$

However, because the circle corresponds to a boundary of the physical object it comes from, there must be a direction associated with it and not only an orientation. Hence, there is more than the two-fold ambiguity seen above. One must consider the other two cases for which the circle is "flipped" 180 degrees about its diameter. Thus, there are four possible solutions for the angle theta,

$$\begin{aligned} \theta_1 &= |\theta| \\ \theta_2 &= -|\theta| \\ \theta_3 &= \pi + |\theta| \\ \theta_4 &= \pi - |\theta|, \end{aligned}$$

but only one of them is physically correct. At this point, it is not possible to determine which of the four solutions above is correct, but later in this section this problem will be addressed and a solution given.

In the camera coordinate system, the normal to the plane on which the 3D circle lies is given by applying, in reverse, the transformations computed

$$N_c = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} (V_1 \quad V_2 \quad V_3)(0 \quad 0 \quad -1), \tag{32}$$

where the negative z component of the normal is due to enforcing a right-handed camera coordinate system. To compute the coordinates of the 3D circle in camera coordinates, it is necessary to first determine the position of the center after rotations R_λ and R_θ have been applied. After the second rotation, R_θ, the equation of the cone is given by

$$(x \quad y \quad z) R_\theta^T C' R_\theta \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

or

$$(x \quad y \quad z) \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0. \tag{33}$$

If the desired circle has radius ρ, its center must be located at $y = 0$,

$$z = k = \frac{\lambda_2 \rho}{\sqrt{\lambda_1 \lambda_3}}, \tag{34}$$

and

$$x = w = -2\lambda_1 \left(1 - \frac{\lambda_3}{\lambda_2}\right). \tag{35}$$

Applying the transformations in reverse order yields the desired coordinates of the center of the circle in the camera frame

$$O_c = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} (V_1 \quad V_2 \quad V_3)(w \quad 0 \quad k). \tag{36}$$

Therefore, by obtaining N_c and O_c, the 3D circle normal, and center with respect to the camera coordinate system, the pose-from-ellipse problem is solved. However, use of the above solution was previously reported only for the determination of pose of solids of revolution. Most of the objects used in the work herein do not fall in that category. Therefore, the above solution had to be augmented to include an extra transformation step, namely a rotation about the perpendicular axis of the 3D circle. This additional rotation ensures that the nonsymmetrical characteristics of the object have been taken into account.

5.6. Pose from Ellipses for Objects Other Than Solids of Revolution

There is an infinite number of ways a circle can be rotated about its center, on its plane, and still look the same in its 2D projection. Thus, the correspondence between a 2D ellipse on the image and a 3D circle on the model is not enough to determine the pose of a generic object. An additional constraint is needed, and the above method must be augmented. Figure 12 illustrates the effect of the original solution on a nonrotationally symmetric object. There are four possible solutions and none of them is correct because the additional orientation constraint was not used. The correct solution, when the additional constraint was used, is shown in Fig. 13.

The augmented pose-from-ellipse method uses an ellipse-to-circle correspondence plus an additional point-to-point correspondence to determine an initial pose estimate. The additional point used may lie anywhere in the 3D space of the circle, except along the perpendicular axis that passes through its center. Due to the nature of the objects used, and the fact that for nonrotationally symmetric objects most detected ellipses are the boundaries of holes, a point lying on the plane of the circle is used.

In order to describe the approach to computing the necessary rotation it is assumed that the original solution of [20] and [21] is given in terms of a rotation matrix R_e and a translation vector t_e. It is also assumed that the model has already been transformed in such a way that the center of the circle in question lies on the origin and the circle is on the yz-plane; that is, the x-axis is perpendicular to and passes through the center of the circle. Let the additional point correspondence between 3D model point and 2D image point be ($P_m = (x, y, z)$, $P_i = (r, c)$). In order for P_m to project onto P_i, the following set of transformations must be applied to it:

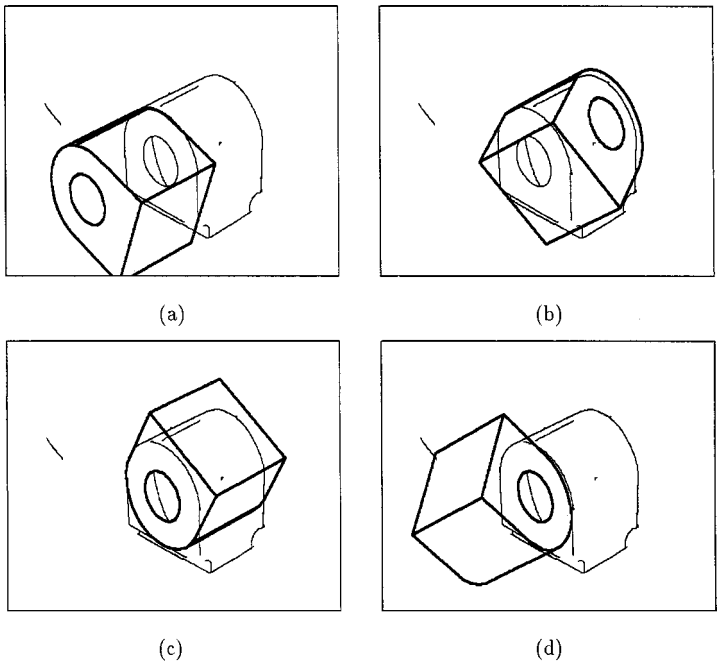

FIG. 12. The four incorrect solutions from the original pose-from-ellipse algorithm. (a) Solution 1; (b) solution 2; (c) solution 3; (d) solution 4.

1. Rotate the model about the x-axis by the unknown angle ϕ:

$$(x', y', z') = (x', y' \cos\phi - z' \sin\phi, y' \sin\phi + z' \cos\phi).$$

2. Apply rotation R_e to the model:

$$(x'', y'', z'') = (x', y', z')R_e.$$

3. Translate the model by t_e:

$$(x''', y''', z''') = (x'', y'', z'') + t_e.$$

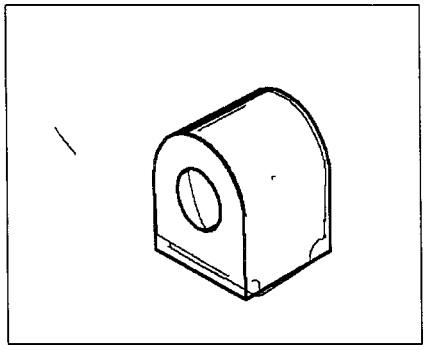

FIG. 13. The correct initial pose using the additional point correspondence constraint.

4. Project the translated model onto the image plane:

$$(r, c) = \left(f\frac{x'''}{z'''}, f\frac{y'''}{z'''} \right), \quad \text{where } f \text{ is the focal length of the camera.}$$

In order to determine the unknown rotation angle ϕ, notice that:

$$\frac{r}{x'''} = \frac{c}{y'''}.$$

The above leads to a trigonometric equation on ϕ of the form $A \sin\phi + B\cos\phi + C = 0$, which yields the following two solutions,

$$\phi = 2 \arctan \frac{(-A + \sqrt{(A^2 + B^2 - c^2)})}{-B + C} \tag{37}$$

and

$$\phi = 2 \arctan \frac{(-A - \sqrt{(A^2 + B^2 - c^2)})}{-B + C}, \tag{38}$$

where

$$\begin{aligned} A &= y\left(r_{e_{31}} - kr_{e_{32}}\right) + z\left(kr_{e_{22}} - r_{e_{21}}\right), \\ B &= y\left(r_{e_{21}} - kr_{e_{22}}\right) + z\left(r_{e_{31}} - kr_{e_{32}}\right), \\ C &= x\left(r_{e_{11}} - kr_{e_{12}}\right) + t_{e_x} - kt_{e_y}, \end{aligned} \tag{39}$$

and $k = r/c$.

Therefore, there is a total of eight possible solutions for the pose, but only one of them is physically correct. Figure 14 illustrates the eight pose estimates, computed as described above, for an image pair of the hexnut object. The verification procedure described in the next section can rapidly rule out the incorrect solutions by means of a directed distance computation between the projected model edges and the detected image edges.

5.7. Verification of the Eight Pose Candidates

In order to have a quantitative measure of how good the estimated model pose is, it is necessary to evaluate how well the projected model edges align with the detected scene edges. This is done by computing a unidirectional version of a modified Hausdorff distance [18] between the image of the projected model and the image of the detected edges.

The most common way of defining the distance between a point a and a point set $B = \{b_1, \ldots, b_{N_B}\}$ is

$$d(a, B) = \min_{b \in B} \|a - b\|. \tag{40}$$

The directed distance d_6 [18] is used to quantitatively evaluate how well the projected model point set (A) overlays the edge image point set (B), and it is defined as

$$d_6(A, B) = \frac{1}{N_A} \sum_{a \in A} d(a, B), \tag{41}$$

where N_A is the number of points in set A.

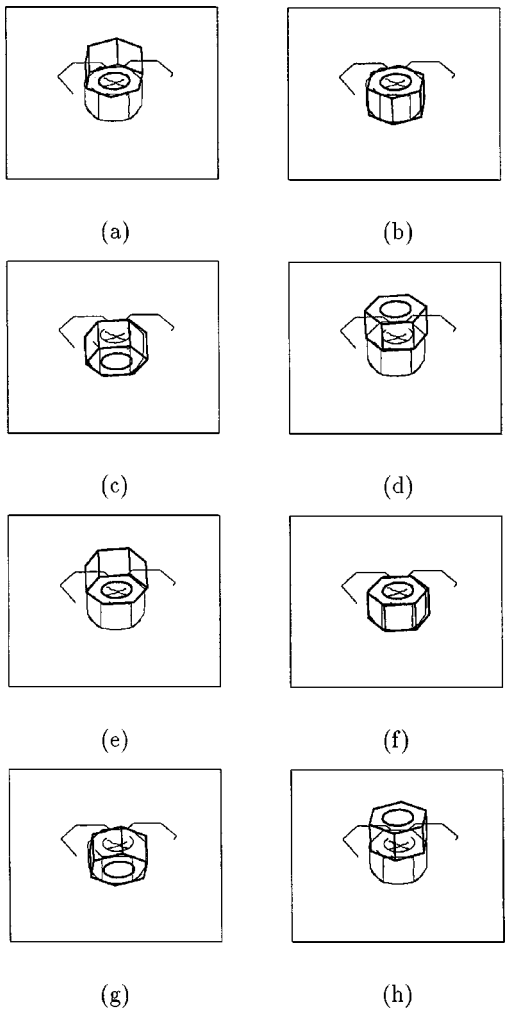

FIG. 14. The eight solutions from the constrained pose-from-ellipse algorithm. (a) Solution 1; (b) solution 2; (c) solution 3; (d) solution 4; (e) solution 5; (f) solution 6; (g) solution 7; (h) solution 8.

For the specific verification task required in this work, the directed distance measure is utilized because it is necessary to account for occlusion as well as extra points in set B, since, in the general case, there are multiple objects in the image. The above measure has been shown to rule out the pose hypotheses that are obviously incorrect. Results to this effect are given in Section 6.

5.8. Generalized Pose Computation from Ellipses and Points

In the two previous sections two different methods for computing pose from different features were described. However, it is desirable to be able to compute pose from more than one type of feature *simultaneously*. The reason is quite obvious: it will provide a more accurate solution. This section is devoted to formulating a method of computing pose from 2D–3D point correspondences and ellipse–circle correspondences, simultaneously.

To exemplify the issue of accuracy addressed in the paragraph above, the results of the pose estimation from points alone and from one ellipse are shown in Figs. 15a and 15b, respectively. Notice that due to the localized concentration of detectable feature points and

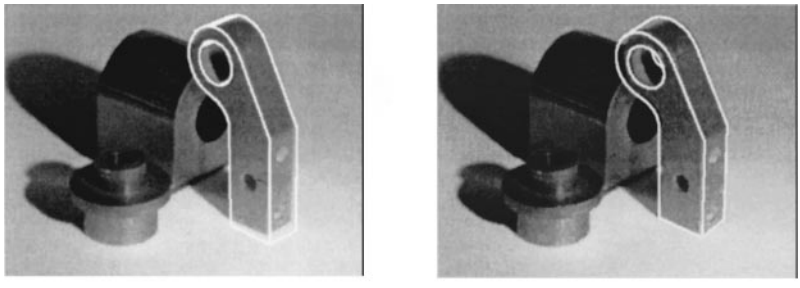

FIG. 15. The poses computed using algorithms described in Sections 4.3 and 4.4. (a) Pose from a single ellipse–circle; (b) pose from six point correspondences.

the physical distance between the circle and these points, the poses computed align well only in the areas where the features used are located. Specifically, the result in Fig. 15a shows good alignment in the upper portion of the object where the circle is located. On the other hand, the result in Fig. 15b shows good alignment only at the lower part of the object where the concentration of detectable feature points is located.

It can be seen that both results are close to what is expected, but neither can be considered a good result. This is the main motivation behind formulating the pose problem in a way that the information from both point correspondences and ellipse–circle correspondences can be used simultaneously.

In the pose-from-points problem, the system of equations to be solved, as given by Eq. (3), is homogeneous so no algebraic solution can be found. Furthermore, constraints must be imposed in order to ensure the correctness of the solution found in terms of the physical and spatial conditions. However, if the system of equations obtained was not homogeneous, it could be directly solved by a simple direct method. The results obtained from the pose-from-ellipses algorithm can be used in order to augment the system of equations that arises from point correspondences and the resulting system of equations can be solved by direct methods. This is done in the following way.

Let N_c and O_c be the 3D circle normal and center (in camera coordinates), respectively. Also, let N_o and O_o be the normal and center of the same circle, but in the object coordinate system. The two normals and the two centers are related by the transformation $T_e = \{R_e, t_e\}$, where the subscript e indicates that it has been found from an ellipse-to-circle correspondence. However, a *generalized* solution from points and ellipses combined is sought. Therefore, the N_c and O_c computed are taken as *observations* and it is assumed that the true transformation, $T = \{R, t\}$, is unknown. Thus,

$$N_c = \begin{pmatrix} N_{c_x} \\ N_{c_y} \\ N_{c_x} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} N_{o_x} \\ N_{o_y} \\ N_{o_s} \end{pmatrix} = RN_o, \tag{42}$$

and

$$O_c = \begin{pmatrix} O_{c_x} \\ O_{c_y} \\ O_{c_x} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} O_{o_x} \\ O_{o_y} \\ O_{o_x} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = RO_o + t. \tag{43}$$

Furthermore, since $R^{-1} = R^T$, N_o can be written as

$$N_o = \begin{pmatrix} N_{o_x} \\ N_{o_y} \\ N_{o_z} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{pmatrix} \begin{pmatrix} N_{c_x} \\ N_{c_y} \\ N_{c_z} \end{pmatrix} = R^T N_c. \tag{44}$$

Notice that Eqs. (42) and (44), if used simultaneously, enforce the condition that the unknown rotation matrix be orthonormal. Equations (42)–(44) involve the same 12 unknowns as in the system of equations (3). Hence, that system is augmented by those three equations giving rise to the following new system

$$B'w = k, \tag{45}$$

where

$$B' = \begin{pmatrix} fx_1 & fy_1 & fz_1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ 0 & 0 & 0 & fx_1 & fy_1 & fz_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ fx_2 & fy_2 & fz_2 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & f & 0 & -u_2 \\ 0 & 0 & 0 & fx_2 & fy_2 & fz_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ fx_6 & fy_6 & fz_6 & 0 & 0 & 0 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ 0 & 0 & 0 & fx_6 & fy_6 & fz_6 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \\ N_{o_x} & N_{o_y} & N_{o_x} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & N_{o_x} & N_{o_y} & N_{o_z} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & N_{o_x} & N_{o_y} & N_{o_z} & 0 & 0 & 0 \\ O_{o_x} & O_{o_y} & O_{o_z} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & O_{o_x} & O_{o_y} & O_{o_z} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & O_{o_x} & O_{o_y} & O_{o_z} & 0 & 0 & 1 \\ N_{c_x} & 0 & 0 & N_{c_y} & 0 & 0 & N_{c_z} & 0 & 0 & 0 & 0 & 0 \\ 0 & N_{c_x} & 0 & 0 & N_{c_y} & 0 & 0 & N_{c_z} & 0 & 0 & 0 & 0 \\ 0 & 0 & N_{c_x} & 0 & 0 & N_{c_y} & 0 & 0 & N_{c_z} & 0 & 0 & 0 \end{pmatrix}, \tag{46}$$

$$w = (r_{11} \quad r_{12} \quad r_{13} \quad r_{21} \quad r_{22} \quad r_{23} \quad r_{31} \quad r_{32} \quad r_{33} \quad t_x \quad t_y \quad t_z)^T, \tag{47}$$

and

$$k = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & N_{c_x} & N_{c_y} & N_{c_z} & O_{c_x} & O_{c_y} & O_{c_z} & N_{o_x} & N_{o_y} & N_{o_z} \end{pmatrix}^T. \tag{48}$$

The system above can be solved by a direct least-squares solution of the form

$$w = (B'^T B')^{-1} B'^T k. \tag{49}$$

It can be seen that since nine new equations have been added to the system, there is no need for all six point correspondences in order to solve for the 12 transformation unknowns. However, matrix B' must be full rank and, therefore, at least three points must be used.

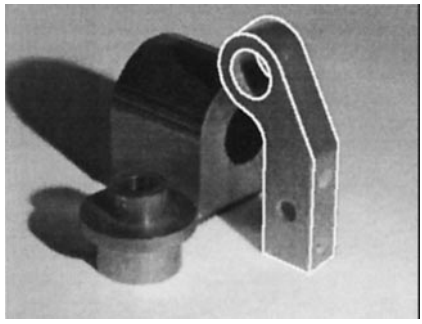

FIG. 16. The pose computed from six point correspondences and one ellipse–circle correspondence using the generalized methodology. Note the gap between the object and the projected model at the very top of the object.

Figure 16 illustrates the pose computed for the same image of Figs. 15a and 15b using the generalized pose estimation technique and making use of the same six point correspondences and the same ellipse correspondence previously used.

Visual inspection of the results in Figs. 15a, 15b, and 16 shows the superiority of the new technique over the other single-feature methods. The projection of the CAD model onto the image using the transformation matrix obtained using the new technique yields a better alignment than those projections obtained using only point correspondences or a single ellipse–circle correspondence. In order to compare the results quantitatively, Table 2 shows the pose transformations obtained for each linear method used. Also shown in the last row of Table 2 is the final transformation obtained after convergence of the iterative, nonlinear Gauss–Newton method [33]. The nonlinear method requires an initial guess; the results of the linear methods (points only, circle–ellipse only, and points and circle–ellipse together) have been used as such. The final results after convergence of the iterative method are the same regardless of which initial guess was used; the difference lies in the number of iterations it took for the method to converge to the final solution. These results are reported in [39]. The model projection for the solution obtained using the nonlinear method is shown in Fig. 17.

TABLE 2
Pose Results from Different Methods

Method	R	t
Points only	$\begin{pmatrix} 0.410 & -0.129 & -0.902 \\ 0.606 & -0.700 & 0.376 \\ -0.681 & -0.701 & -0.208 \end{pmatrix}$	$(-43.125\ -25.511\ 1232.036)$
Circle only	$\begin{pmatrix} 0.302 & 0.302 & -0.932 \\ 0.692 & -0.628 & 0.355 \\ -0.655 & -0.753 & -0.054 \end{pmatrix}$	$(-35.161\ -15.358\ 1195.293)$
Points and circle	$\begin{pmatrix} 0.398 & -0.142 & -0.902 \\ 0.554 & -0.667 & 0.336 \\ -0.700 & -0.684 & -0.201 \end{pmatrix}$	$(-43.077\ -26.400\ 1217.855)$
Nonlinear	$\begin{pmatrix} 0.341 & -0.156 & -0.927 \\ 0.631 & -0.693 & 0.349 \\ -0.697 & -0.704 & -0.137 \end{pmatrix}$	$(-43.23\ -28.254\ 1273.07)$

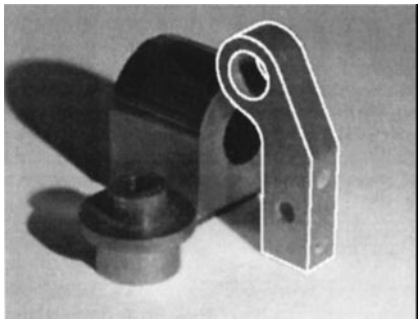

FIG. 17. The final pose obtained using the solution in Fig. 16 as the initial solution to a nonlinear least-squares pose estimation procedure. Note that the gap between the object and its features is now gone.

Additional sample results showing the superiority of a method that combines information from different features for pose estimation are given in Figs. 18–20. As it can be seen, the result obtained when simultaneously using point correspondences and an ellipse–circle correspondence is far more accurate than those results using points alone or an ellipse–circle correspondence alone.

It is very important to emphasize that, in the general case, the pose results obtained from the generalized technique using several feature types can only be taken as initial guesses or strong hypotheses. It may still be necessary to improve and/or optimize the solution found. One option, as discussed above, is to use the solution as an initial guess to a generic iterative nonlinear pose estimation procedure. Another option, which can be employed after the use of the nonlinear estimation or in place of it, is to submit the solution to a constrained optimization procedure, as described in the next section.

5.9. Verification and Pose Optimization

The pose solution found either by the generalized linear pose finding algorithm or by a subsequent nonlinear least-squares procedure must still be verified and/or optimized. In this work, verification and optimization of the solution is performed by minimizing the one-directional distance d_6 between the image of the projected model and the edges found in the scene. In order to make the constrained optimization more efficient, the rotation matrix R associated with the pose transformation found, $T = \{R, t\}$, is represented by its

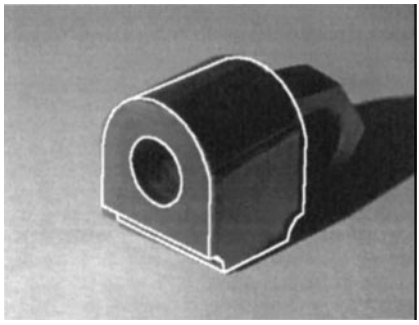

FIG. 18. The pose computed from six point correspondence using the algorithm described in Section 5.2.

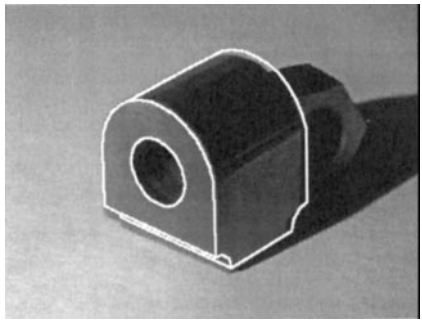

FIG. 19. The pose computed from an ellipse–circle correspondence using the algorithm described in Section 5.6.

corresponding quaternion vector Q, where

$$Q = (s \quad l \quad m \quad n)^T, \tag{50}$$

$$s^2 + l^2 + m^2 + n^2 = 1, \tag{51}$$

and

$$R = \begin{pmatrix} s^2 + l^2 - m^2 - n^2 & 2(lm - sn) & 2(ln + sm) \\ 2(lm + sn) & s^2 - l^2 + m^2 - n^2 & 2(mn - sl) \\ 2(ln - sm) & 2(mn + sl) & s^2 - l^2 - m^2 + n^2 \end{pmatrix}. \tag{52}$$

Powell's method [19] in the seven-dimensional space of the pose solution (four quaternion parameters and the translation t) is used, along with the constraint that the sum of the squares of the quaternion parameters must equal 1, as seen in Eq. (51). Figure 21 shows an initial pose estimate for a single-object image as well as the final result after the constrained optimization has been applied to that initial solution.

In order to be accepted by the verification step, the result obtained from the optimization algorithm has to produce a distance such that $d_6 \leq \tau$, where τ is an empirically determined threshold equal to 5. Typically, the distance for correct pose solutions in the application herein lies in the interval $1 \leq d_6 \leq 5$. Section 6 discusses this in more detail.

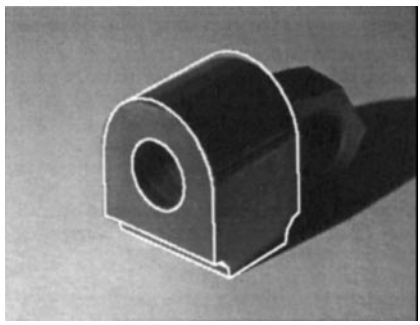

FIG. 20. The pose computed from six point correspondence and one ellipse–circle correspondence using the generalized methodology.

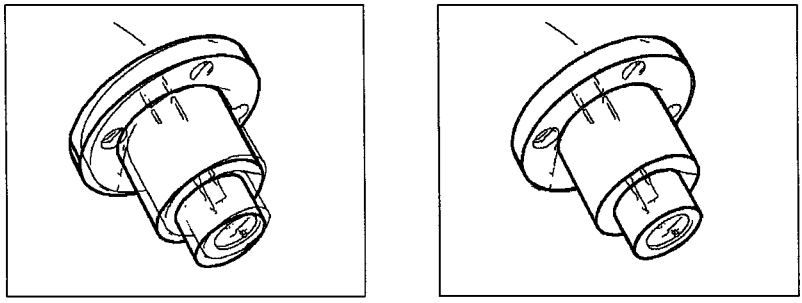

FIG. 21. Example pose hypothesis and final pose after constrained optimization: (a) initial pose and (b) final pose.

6. EXPERIMENTS AND RESULTS

The methods described in this paper were implemented in the RIO recognition system. The physical setup for the system is composed of a single CCD camera and two light sources, one placed at the right of the camera and the other at the left. Polarizers are used in order to reduce highlight effects [57], since the objects used in this work are shiny, metallic, industrial parts.

With each light source turned on at a time, two images, namely the left and the right image, are taken of the scene to be recognized. These images undergo a special processing sequence in order to generate a single, suitable edge image from which the features can be extracted. A high-level description of the sequence is given below, and details can be found in [17]. First, edges are detected in both input images. A combined edge image is generated by ORing the two edge images. Since this image contains shadows cast on the background, a mask image is used to remove them. The mask image is obtained by thresholding the original input images and by ANDing them together. The final edge image is produced by ANDing the mask image and the combined edge image.

An evaluation of the system developed is presented in this section. First, a step-by-step example of the results obtained at the different stages of the system is given. Next, the results of a set of experiments with real images are shown, and example cases of misdetection by the system are studied. Issues such as verification of occluded object hypotheses, ruling out of incorrect pose hypotheses, and recognition of multiple instances of the same object in a scene are addressed.

6.1. Sample Run of the System

In this section, an example of the steps performed at different stages of the system for a multiobject scene is given. Figures 22a and 22b show the original graytone images used as input to the system. The combined edge image extracted using the procedure described above is shown in Fig. 22c. The linear features, circular arc features, and ellipses extracted are depicted in Figs. 22d–22f, respectively.

For this case study, a total of eight hypotheses were generated. Five of them were incorrect and they are shown in Fig. 23, along with their respective average distances d and the percentage of projected model edge points that were within five or less pixels from a scene edge pixel. All five incorrect hypotheses were ruled out by the verification criteria.

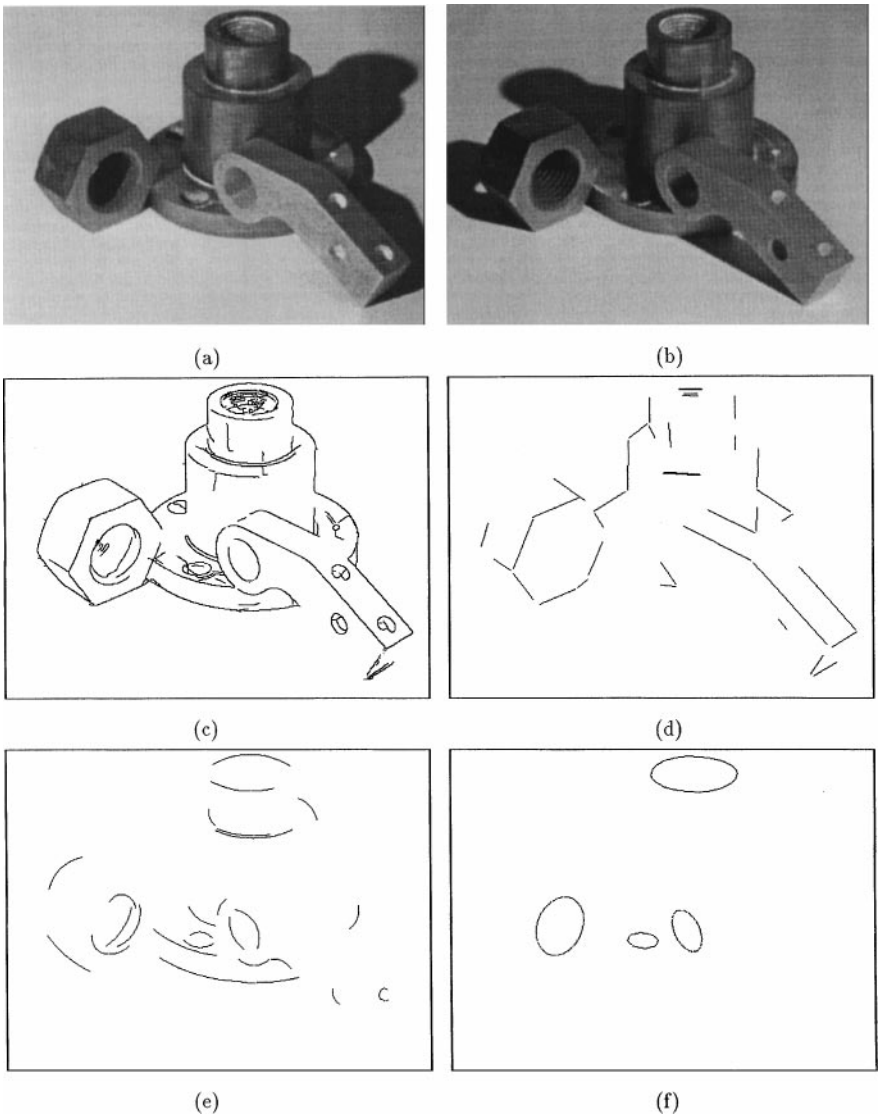

FIG. 22. Sample run of the system. (a) Original left image; (b) original right image; (c) combined edge image; (d) linear features detected; (e) circular arc features detected; (f) ellipses detected.

The correct hypotheses generated by the system are shown in Fig. 24. All three correct hypotheses were accepted as valid by the verification criteria, since they all satisfied $d \leq 5$ and $p \geq 65\%$.

6.2. Experiments on a Set of Real Images

Forty-two pairs of real images were used to evaluate the performance of the overall recognition system. These images were obtained from scenes containing up to three objects, since the camera configuration we used was not able to accurately detect features in larger scenes; 9 of them contain 3 objects, 13 contain 2 objects, and 20 contain a single object. Of the total number of 73 objects in the test scenes, 23 are partially occluded by one or two objects. These statistics about the test images are shown in Table 3. The images were

TABLE 3
Test Scene Statistics

Number of test images	42
Images with 3 objects	9
Images with 2 objects	13
Images with 1 objects	20
Total number of objects	73
Number of occluded objects	23

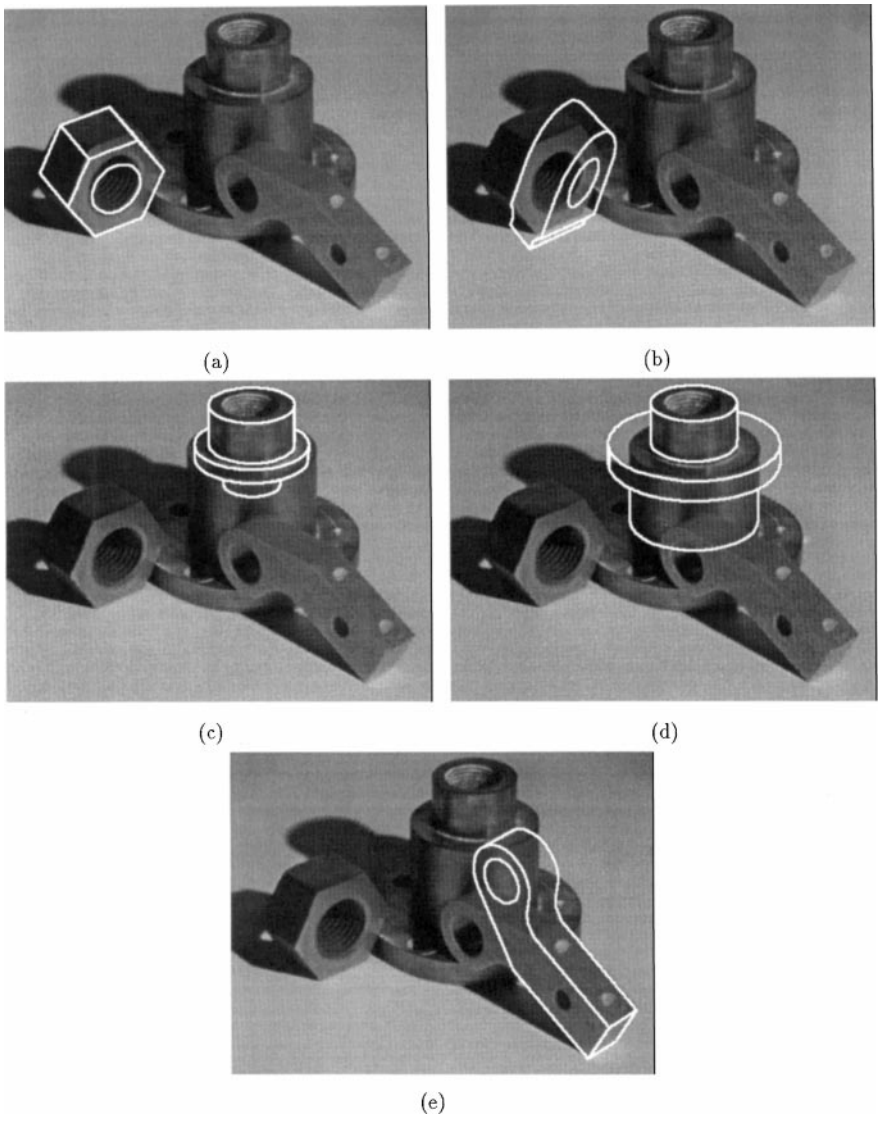

FIG. 23. Incorrect hypotheses generated for the sample case study. (a) Incorrect hypothesis 1 ($d = 6.61$, $p = 59.2\%$); (b) incorrect hypothesis 2 ($d = 17.74$, $p = 37.1\%$); (c) incorrect hypothesis 3 ($d = 6.95$, $p = 63.2\%$); incorrect hypothesis 4 ($d = 9.81$, $p = 48.9\%$); (e) incorrect hypothesis 5 ($d = 8.31$, $p = 53.2\%$).

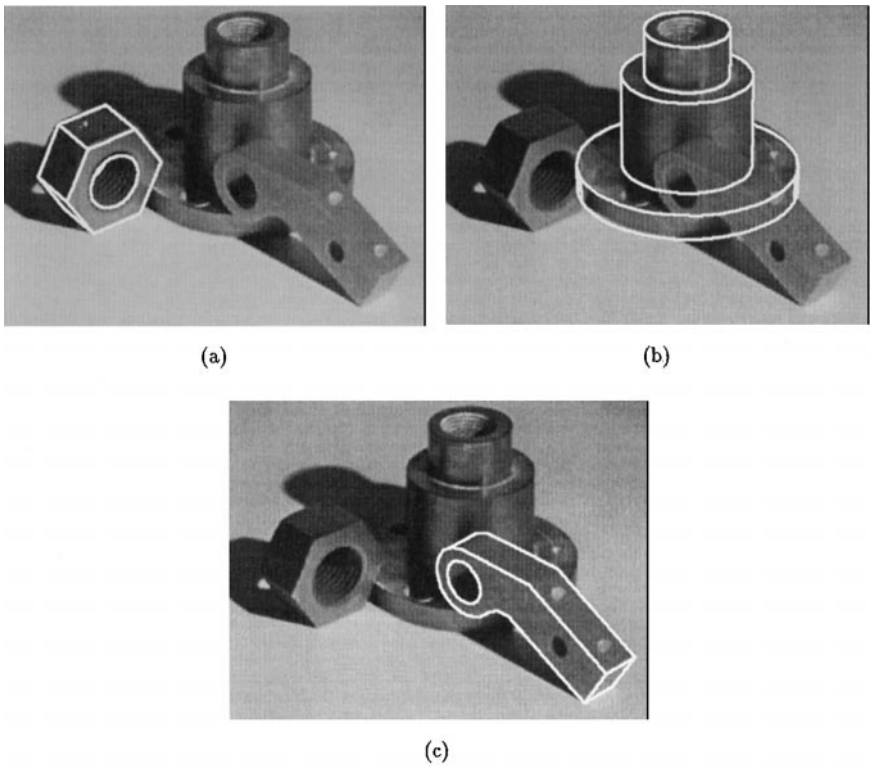

FIG. 24. Correct hypotheses generated for the sample case study (a) Correct hypothesis 1 ($d = 3.30$, $p = 83.1\%$); (b) correct hypothesis 2 ($d = 3.84$, $p = 79.5\%$); (c) correct hypothesis 3 ($d = 3.92$, $p = 83.5\%$).

matched to a database of 20 view-class models representing five 3D objects. The size of the lookup table for these experiments was 100×100.

The hypothesis-generation process, as performed by the relational indexing technique, yielded the results shown in Table 4. Hypotheses were taken as valid if the votes received reflected at least 50% of the 2-graphs of the hypothesized model or if they were among the top five ranked hypotheses, regardless of the votes received. (These "rules" were derived empirically from experimentation and common sense.) Out of the ten unoccluded cases for which the vote threshold was not met, eight were taken as valid hypotheses because they were among the top five ranked hypotheses generated for the test images they were in. Out of the 40 correct hypotheses that did meet the voting threshold criterion, nine were ranked first, and 27 were among the top five ranked, for their respective test images. From the data in Table 4 it can also be seen that out of the total of 73 instances of the objects in the test images, seven were not among the hypotheses generated by the relational indexing (some of these cases are discussed in more detail in the next section). Five of these seven were

TABLE 4
Hypotheses Generation Statistics

Objects	≥50%	<50%	≥50%, top 5	<50%, top 5	1st ranked
Unoccluded objects	40	10	27	8	9
Occluded objects	18	5	0	0	0

TABLE 5
Varification Statistics (66 Hypotheses Tested)

	Hypotheses	
	Correct	Incorrect
Successful verification	62	0
Unsuccessful verification	4	254
Average distance d	4.28 pix	20.13 pix
Avg. % points: $d \leq 5$	79.04%	39.68%

instances of occluded objects and two of unoccluded objects. The remaining 66 instances were correctly taken as valid hypotheses and passed along to the verification step.

The verification was successful in 62 out of the 66 correct hypotheses tested, and it failed in four cases. In those four instances, the features detected were not enough to compute pose, and, therefore, did not allow verification of otherwise correct hypotheses. Some of these unsuccessful verification cases are detailed in the next section. As discussed in Section 4, a given hypothesis was taken as successfully verified if the average distance between the projected model edges and the edges detected in the image was less than five pixels.[2] If the average distance was more than five, but if at least 65% of the projected model edge points were within at least five pixels of an image edge point, the verification was also taken as successful. The overall average distance among all 62 correct hypotheses successfully verified was 4.28. The average percent of projected model points that were at least five pixels from an edge point was 79.04%. These numbers indicate the accuracy of the linear pose estimation procedure. Statistics associated with the verification step are shown in Table 5.

6.3. Study of Some Misdetection Cases

As discussed in the previous section, there were scenes for which some objects were not recognized. In those cases either the hypothesis-generation process failed to generate suitable hypotheses or the verification procedure failed due to a lack of suitable features detected that could be used to compute pose.

An example of the first case, where the relational indexing failed to produce a correct hypothesis is illustrated in Fig. 25. The original right image and edges detected are shown in Figs. 25a and 25b, respectively. The primitive lines and circular arcs detected are shown in Fig. 25c. The features extracted are shown in Fig. 25d. It can be seen that the features detected for the misdetected object (the small stacked cylinder) were not sufficient to produce a suitable hypothesis.

The second type of misdetection, where the features extracted were enough to generate a hypothesis from the relational indexing procedure, but were not suitable features to allow pose computation, is illustrated in Fig. 26. It is important to notice that due to the rotationally symmetric nature of the small stacked cylinder object, the detection of ellipses is essential for successful verification of hypotheses for that particular object. Although the feature detection found several elliptical arcs, it did not find enough evidence to generate a full ellipse, which is currently required by the pose estimator.

[2] To transform this concept to the real world, note that in our tests, the average pixel/mm conversion factor was 2.43 pixels/mm.

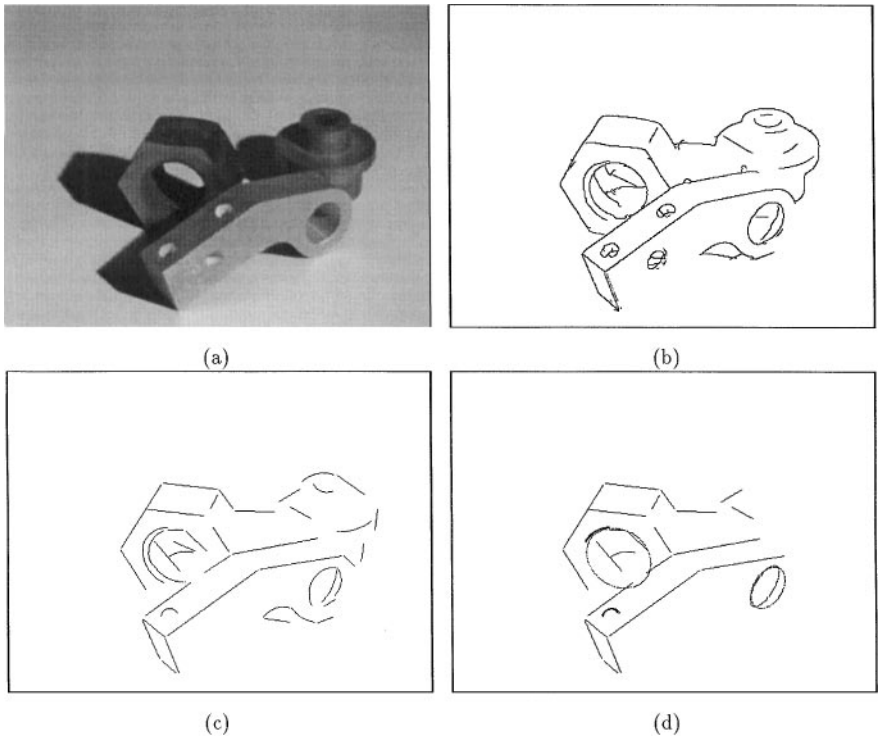

FIG. 25. Example of insufficient features to generate a hypothesis. (a) Original right image; (b) edges detected; (c) primitives lines and arcs detected; (d) features detected.

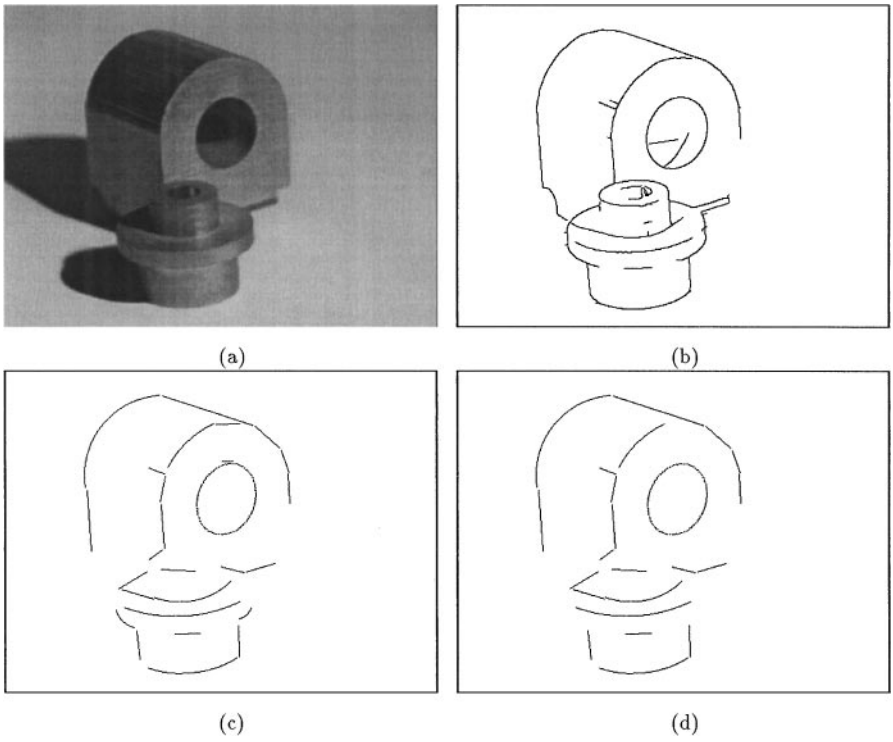

FIG. 26. Example of features not suitable for pose computation/verification. (a) Original right image; (b) edges detected; (c) primitive lines and arcs detected; (d) Features detected.

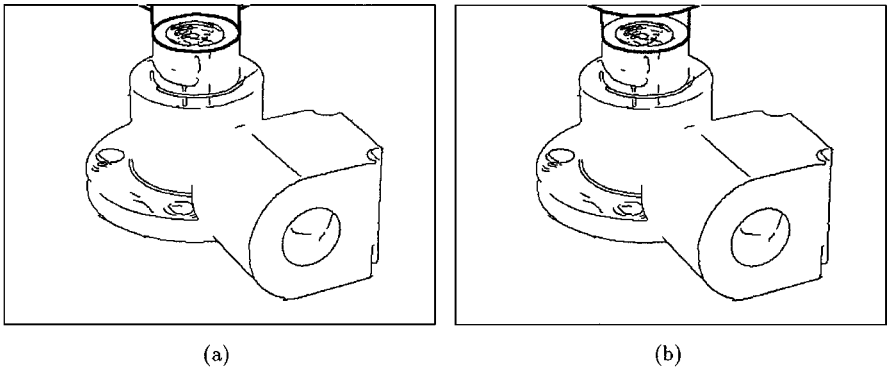

FIG. 27. Examples of pose overlays for Case 1 (see text).

6.4. Ruling Out Incorrect Pose-from-Ellipse Hypotheses

The distance measure used was efficient in ruling out incorrect pose hypotheses arising from the pose-from-ellipse algorithm. In all test cases, the correct pose yielded the smallest distance between the projected model edges and the scene edges. However, in some cases, some of the incorrect poses generated distances comparable to the the correct ones. There are two main reasons why this happened:

- Case 1: in some cases, the projected model, for the pose being tested, was almost completely outside the image bounds, but the portion that was within the image bounds aligned quite well with the scene edges. Two such examples can be seen in Fig. 27. Even though the computed distances are small in these cases, such hypotheses can be easily ruled out at the model projection time by computing what percentage of model edges lie outside image boundaries.
- Case 2: one of the incorrect pose-from-ellipse solutions aligns the outline of the projected model quite well with the outline of the object in the edge image, though the inside edges do not align well. This also produces an average distance that is comparable to, though always larger than, the distance for the correct pose. An example of this situation in shown in Fig. 28.

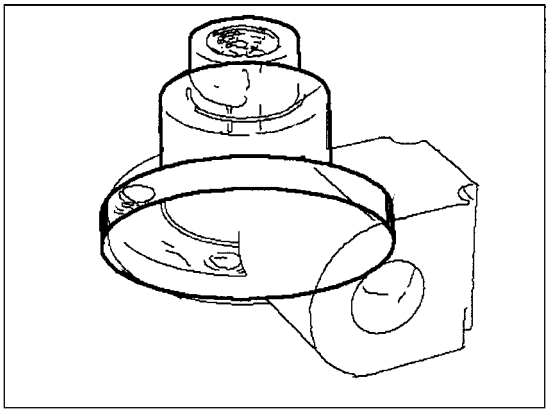

FIG. 28. Example of pose overlay for Case 2 (see text).

TABLE 6
Average Distances *d* from Pose-from-Ellipse Algorithm

Test image	Correct pose	Pose 2	Pose 3	Pose 4
1	4.232169	5.905406	38.015209	38.064182
2	3.492276	3.836853	5.438044	6.229556
3	2.565156	10.072806	75.727005	77.769081
4	6.048713	6.095455	7.518664	9.984772
5	6.500802	10.659150	12.863460	14.970914
6	4.760042	5.682881	37.694584	38.204376
7	3.650602	4.853358	22.808851	23.714478
8	5.016474	7.895664	45.042072	51.129051
9	3.737756	9.964845	15.774166	21.634424
10	5.117007	10.208114	82.480644	84.060287
11	6.213883	10.301037	90.775963	92.624016
12	2.749343	4.488708	6.710792	11.896962
13	6.487325	15.678705	45.408619	49.991577
14	1.485621	11.119739	14.899862	18.397236
15	5.114830	11.136388	12.420178	14.751463

Table 6 shows the distance measure computed for each four possible pose cases for 15 different test images. In all cases, the correct pose yielded the smallest distance.

6.5. Improving the Verification of Occluded Object Hypotheses

It can be seen from the data shown in Fig. 24b that a large part of the projected model edges do not align well with the scene edges due to the fact that the corresponding portion of the model is occluded by another object. Therefore, it is apparent that occlusion adversely affects the verification procedure devised. In order to make the verification more effective for occluded objects, the following steps may be added to it. First, all unoccluded objects are identified. This can be done not only by means of the distance measure (d) and the percentage of points that align well (p), but also from the pose parameters themselves which include the exact 3D location of each object. Thus, it is possible to determine which objects in the scene are occluded by one or more object. Once the unoccluded objects have been identified, their corresponding projected edges are used to generate masks. The masks generated for the case study introduced in Section 5.2 can be seen in Figs. 29a and 29b. These masks are used to mask out the predicted occluded regions in the projected model of the occluded object as shown in Fig. 29d. Then the masked project model is used to compute d and p, which will determine whether the verification succeeds or not. It is shown in the captions of Figs. 29e and 29f that both d and p are greatly improved by this masking process, making the verification more reliable.

7. DISCUSSION AND FUTURE WORK

This paper described the RIO object recognition system. RIO utilizes a number of new techniques to recognize 3D objects in 2D intensity images. Real images are used in the stage of learning the feature-based relational models of the objects to be recognized. A new technique called relational indexing is used as an effective database retrieval tool, generating suitable model hypotheses. A new linear pose estimation technique is used to

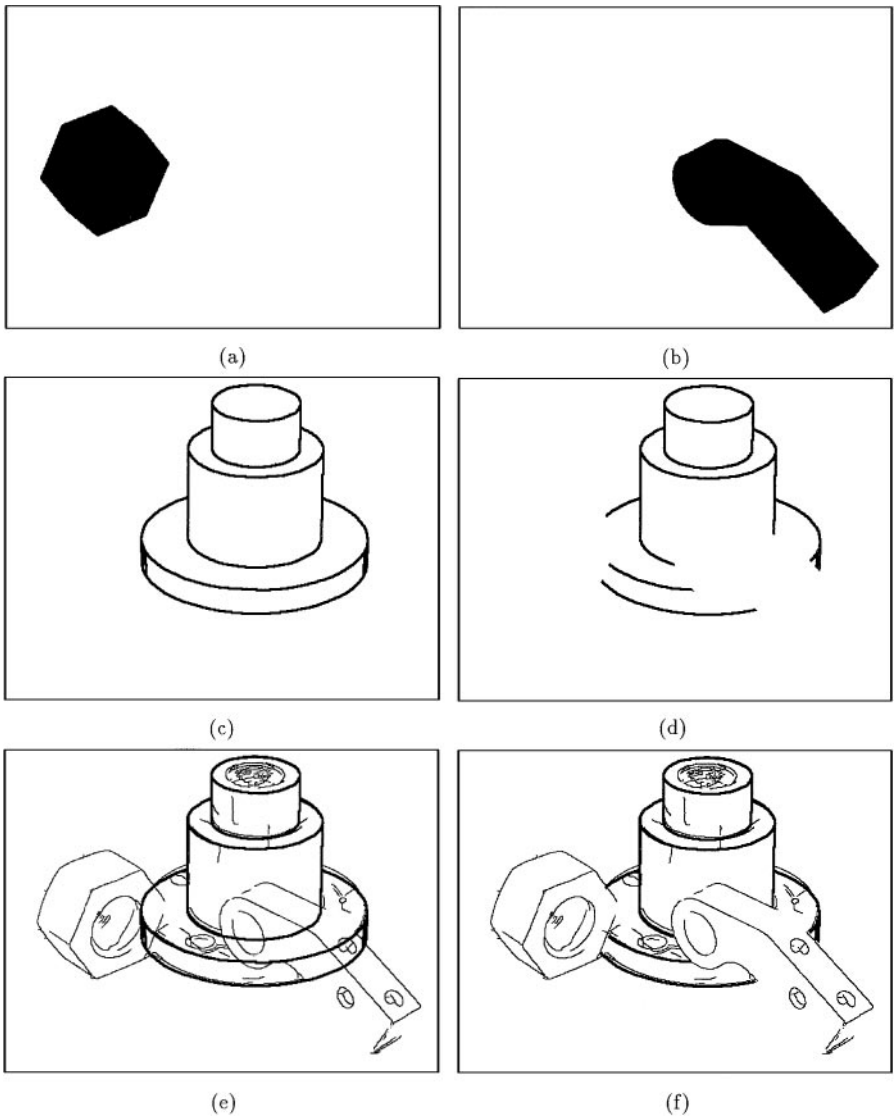

FIG. 29. Masking out occluded hypothesis for verification. (a) Mask 1; (b) mask 2; (c) original projected model; (d) masked projected model; (e) original model overlaid on image edges ($d = 3.84$, $p = 79.5\%$); (f) masked model overlaid on image edges ($d = 2.42$, $p = 91.3\%$).

verify or disprove the hypotheses and makes simultaneous use of multiple types of 2D–3D feature correspondences.

As the experiments have shown, the performance of the RIO system was very good with the exception of a few misdetection cases. These cases were mainly due to poor feature detection, a well-known problem in most feature-based recognition systems. Issues on improvement of occluded hypothesis verification were addressed and a method for effectively handling the verification of occluded objects was proposed. It was shown that the distance measure utilized was appropriate for disambiguating among incorrect pose hypotheses generated by the pose-from-ellipse algorithm.

Even though the verification procedure was proven to be reliable given the criterion used, namely the average distance between projected CAD model edges and scene edges and the percentage of projected edge pixels that aligned well with scene edge pixels, it is felt that a more robust verification criterion needs to be developed. Making the assumption that the distances between projected model edge points and scene edge points have a Gaussian distribution, it seems logical to use a chi-squared test (with as many degrees of freedom as the points in the projected model edges) to decide whether or not the alignment produced by the computed pose is acceptable.

All the experiments run were limited to subgraphs of size 2. As mentioned in Section 4, the smallest size index was used in order to effectively handle occlusion. However, a hierarchical, multilevel indexing approach is also viable and should be investigated. The larger the subgraphs used, the more reliable the generated hypotheses will be, and the faster they can be generated. However, in multiobject scenes only single features or small subgraphs may be detected. Therefore, it is necessary to start at the largest subgraph level and go down as needed. Objects that are unoccluded may be recognized at the highest level of indexing.

Work on investigating the effects of occlusion on the single-object feature-based models is in order. The special case of multiple-object scenes that contain a considerable amount of occlusion needs to be studied carefully. It is intuitive that the set of features that can be reliably detected from a real image of a particular model alone is a superset of the features that can be detected in a scene where that model is partially occluded. The technique developed was designed to handle occlusions. However, when the object is partially occluded, it is possible that some of the original features detected in its training images can change and give rise to new features that often appear when occlusion is present. It is also possible that new relationships between features arise due to occlusion. It seems logical to augment the current single-object feature-based models to include the knowledge obtained by studying a set of training images where multiple objects and occlusion are present.

A discrete voting technique was associated with the relational indexing method developed in this work. Relational indexing can be performed in the context of a probabilistic framework, as described in [14]. This should considerably improve the hypotheses generated, since the probabilities associated with the features (computed at model generation time) can be used as weights when computing the votes to be cast.

REFERENCES

1. J. Ben-Arie and A. Z. Meiri, 3-d object recognition by optimal matching search of multinary relations, *Comput. Vision Graphics Image Process.* **37**, 1987, 345–361.
2. R. Bergevin and M. D. Levine, Generic object recognition: Building and matching coarse descriptions from line drawings, *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 1993, 19–36.
3. J. P. Besl and R. C. Jain, Three-dimensional object recognition, *ACM Comput. Surveys* **17**, 1985, 75–154.
4. R. C. Bolles and R. A. Cain, Recognizing and locating partially visible objects: The local-feature-focus method, *Int. J. Robotics Res.* **1**(3), 1982, 57–82.
5. R. C. Bolles and P. Horaud, 3dpo: A three-dimensional part orientation system, *Int. J. Robotics Res.* **5**(3), 1986, 3–26.
6. J. P. Brady, N. Nandhakumar, and J. K. Aggarwal, Recent progress in object recognition from range data, *Image Vision Comput.* **7**(4), 1989, 295–307.
7. J. B. Burns and E. M Riseman, Matching complex images to multiple 3d objects using view description networks, in *Proc. of the IEEE CVPR*, pp. 328–334, 1992.

8. A. Califano and R. Mohan, Multidimensional indexing for recognition of visual shapes, *IEEE Trans. Pattern Anal. Mach. Intell.* **16**, 1994, 373–482.
9. O. I. Camps, L. G. Shapiro, and R. M. Haralick, Image prediction for computer vision, in *Three-dimensional Object Recognition Systems* (A. Jain and P. Flynn, Eds.), Elsevier, Amsterdam/New York, 1993.
10. C. H. Chen and A. C. Kak, A robot vision system for recognizing 3d objects in lower order polynomial time, *IEEE Trans. Systems Man Cybernetics* **19**, 1989, 1535–1563.
11. C. H. Chen and P. Mulgaonkar, Automatic vision programming, *CVGIP: Image Understanding* **55**, 1992, 170–183.
12. J. L. Chen and G. C. Stockman, Matching curved 3d object models to 2d images, in *Second CAD-Based Vision Workshop* (A. Kak and K. Ikeuchi, Eds.), pp. 210–227, 1994.
13. R. T. Chin and C. R. Dyer, Model-based recognition in robot vision, *ACM Comput. Surveys* **18**, 1986, 67–108.
14. Y. Chou and L. G. Shapiro, Probabilistic relational indexing, in *Proceedings of the International Conference on Pattern Recognition, Brisbane, Australia, 1997*, pp. 1331–1335, 1998.
15. M. S. Costa, R. M. Haralick, and L. G. Shapiro, Optimal affine invariant point matching, in *Proceedings of 10th ICPR*, Vol. 1, pp. 233–236, 1990.
16. M. S. Costa and L. G. Shapiro, Scene analysis using appearance-based models and relational indexing, in *International Symposium on Computer Vision, November 1995*, pp. 103–108.
17. M. S. Costa, *Object Recognition and Pose Estimation Using Appearance-Based Features and Relational Indexing*, Ph.D. Dissertation, Department of Electrical Engineering, University of Washington, 1997.
18. M.-P. Dubuisson and A. K. Jain, A modified Hausdorff distance for object matching, in *Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel, 1994.*
19. W. H. Press *et al.*, *Numerical Recipes in C*, Cambridge Univ. Press, Cambridge, UK, 1992.
20. D. Forsyth *et al.*, Invariant descriptors for 3-d object recognition and pose, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**, 1991, 971–991.
21. M. Dhome *et al.*, Spatial localization of modeled objects of revolution in monocular perspective vision, in *First European Conference on Computer Vision, 1989.*
22. C. Dorai and A. K. Jain, Shape spectrum based view grouping and matching of 3D free-form objects, *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 1997, 1131–1138.
23. A. Etemadi, Robust segmentation of edge data, in *Proceedings of the IEE Image Processing Conference, 1992.*
24. T. J. Fan, G. Medioni, and R. Nevatia, Recognizing 3d objects using surface descriptions, *IEEE Trans. Patt. Anal. Mach. Intell.* **11**, 1989, 1140–1157.
25. O. D. Faugeras, *Three-Dimensional Computer Vision, a Geometric Viewpoint*, MIT Press, Cambridge, MA, 1993.
26. O. D. Faugeras and M. Hebert, The representation, recognition, and locating of 3d objects, *Int. J. Robotics Res.* **5**(3), 1986, 27–52.
27. P. J. Flynn and A. K. Jain, Bonsai: 3d object recognition using constrained search, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**, 1991, 1066–1075.
28. S. Ganapathy, Decomposition of transformation matrices for robot vision, in *Proceedings of the International Conference on Robotics and Automation, 1984.*
29. C. Goad, Fast 3D model-based vision, in *From Pixels to Predicates* (A. P. Pentland, Ed.), pp. 371–391, 1986.
30. K. D. Gremban and K. Ikeuchi, Appearance-based vision and the automatic generation of object recognition programs, in *Three-dimensional Object Recognition Systems* (A. Jain and P. Flynn, eds.), Elsevier, Amsterdam/New York, 1993.
31. L. Grewe and A. Kak, Interactive learning of multiple attribute hash table for fast 3d object recognition, in *Proceedings of the Second CAD-Based Vision Workshop, February 1994*, pp. 124–131.
32. R. M. haralick, C. N. Lee, K. Ottenburg, and M. Nolle, Review and analysis of the three-point perspective pose estimation problem, *Int. J. Comput. Vision* **13**, 1994, 331–356.
33. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Reading, MA, 1993.
34. I. Higuchi, H. Delingette, M. Hebert, and K. Ikeuchi, Merging multiple views using a spherical representation, in *Proceedings of the Second CAD-Based Vision Workshop, February 1994*, pp. 124–131.

35. D. P. Huttenlocher, *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image*, Ph.D. dissertation, Massachusetts Institute of Technology, 1988.
36. D. P. Huttenlocher and S. Ullman, Recognizing solid objects by alignment with an image, *Int. J. Comput. Vision* **5**, 1990, 195–212.
37. K. Ikeuchi and T. Kanade, Towards automatic generation of object recognition programs, *CVGIP: Image Understanding* **76**, 1988, 1016–1035.
38. A. K. Jain and R. Hoffman, Evidence-based recognition of 3-d objects, *IEEE Trans. Pattern Anal. Mach. Intell.* **10**, 1988, 783–802.
39. Q. Ji, M. S. Costa, R. M. Haralick, and L. G. Shapiro, An integrated technique for pose estimation from different geometric features, in *Proceedings of Vision Interface '98, Vancouver, June 18-20, 1998*, pp. 77–84.
40. A. E. Johnson and M. Hebert, Efficient multiple model recognition in cluttered 3-D scenes, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1998*, pp. 671–677.
41. Y. Lamdan and H. J. Wolfson, Geometric hashing: A general and efficient model-based recognition scheme, in *Second International Conference on Computer Vision*, pp. 238–249, 1988.
42. D. G. Lowe, Three-dimensional object recognition from single two-dimensional images, *Artificial Intell.* **31**, 1987, 355–395.
43. H. Murase and S. K. Nayar, Visual learning of object models from appearance, *Int. J. Comput. Vision* **14**(1), 1995, 5–24. [Also Technical Report CUCS-054-92]
44. C. F. Olson, Probabilistic indexing for object recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **17**, 1995, 518–522.
45. T. Q. Phong, R. Horaud, A. Yassine, and P. D. Tao, Objet pose from 2-d to 3-d point and line correspondences, *Int. J. Comput. Vision* **15**, 1995, 225–243.
46. A. R. Pope, *Model-based Object Recognition—A Survey of Recent Research*, Technical Report 94-04, January 1994.
47. K. Pulli, Triplet-based object recognition using synthetic and real probability models, in *Proceedings of ICPR96, 1996*, Vol. IV, pp. 75–79.
48. M. Seibert and A. M. Waxman, Adaptive 3d object recognition from multiple views, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 1992, 107–124.
49. L. G. Shapiro and R. M. Haralick, A metric for comparing relational descriptions, *IEEE Trans. Pattern Anal. Mach. Intell.* **7**, 1985, 90–94.
50. F. Stein and G. Medioni, Structural indexing: Efficient three dimensional object recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 1992, 125–145.
51. G. Stockman, Object recognition and localization via pose clustering, *Comput. Vision Graphics Image Process.* **40**, 1987, 361–387.
52. T. M. Strat and M. A. Fishler, Contex-based vision: recognizing objects using information from both 2d and 3d imagery, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**, 1991, 1050–1065.
53. P. Suetens, P. Fua, and A. J. Hanson, Computational strategies for object recognition, *ACM Comput. Surveys* **24**, 1992, 5–61.
54. K. B. Thornton, *Clustering Similar Views of an Object Using a Relational Distance Approach*, Master's thesis, 1989.
55. R. Y. Tsai, An efficient and accurate camera calibration technique for 3d machine vision, *IEEE J. Robotics Automation* **3**(4), 1987, 323–344.
56. R. C. Bolles and P. Horaud, 3DPO: a three-dimensional part orientation system, *Int. J. Robotics Res.* **5**(3), 1986, 3–26.
57. L. B. Wolff, Polarization-based material classification from specular reflection, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**, 1990, 1059–1071.
58. A. K. C. Wong, L. Rong, and X. Liang, Robotic vision: 3D object recognition and pose determination, in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1202–1209, 1998.
59. S. Zhang, G. D. Sullivan, and K. D. Baker, The automatic construction of a view-independent relational model for 3d object recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 1993, 531–544.