

Recognizing 3-D Objects Using Surface Descriptions

TING-JUN FAN, MEMBER, IEEE, GERARD MEDIONI, MEMBER, IEEE, AND
RAMAKANT NEVATIA, SENIOR MEMBER, IEEE

Abstract—Object recognition is a key problem in machine vision. The issues to be addressed relate to the representation of the data and the method used to establish correspondences. We present a system which takes as input dense range data and automatically produces a symbolic description of the objects in the scene in terms of their visible surface patches. This segmented representation may be viewed as a graph whose nodes capture information about the individual surface patches and whose links represent the relationships between them, such as occlusion and connectivity. Based on these relations, a graph for a given scene is decomposed into subgraphs corresponding to different objects.

For the purpose of matching, a model is represented by a set of such descriptions from multiple viewing angles, typically 4–6. Models can therefore be acquired and represented automatically.

Matching between the objects in a scene and the models is performed by three modules: the *scraper*, in which we find the most likely candidate views for each object, the *graph matcher*, which performs a detailed comparison between the potential matching graphs and computes the 3-D transformation between them, and the *analyzer*, which takes a critical look at the results and proposes to split and merge object graphs. We present results on a variety of scenes containing multiple complex objects with occlusion.

Index Terms—Object description, object recognition, range image analysis.

I. INTRODUCTION

RECOGNIZING objects in a scene is a primary goal of computer vision. The difficulty of this task depends on several factors such as: the number and complexity of objects in the scene, the number of objects in the model database, and the amount of *a priori* information about the scene. The appropriate techniques for object recognition depend on the difficulty of the task. In our case, we attempt to recognize rather complex objects in range images.

The simplest tasks assume that objects are unoccluded and occur in one or more of standard viewing positions. This essentially reduces to a 2-D recognition task. In another set of tasks, model-based techniques are used.

Manuscript received July 22, 1988; revised May 1, 1989. Recommended for acceptance by O. D. Faugeras. This work was supported by the Defense Advanced Research Projects Agency under Contract F33615-87-C-1436, monitored by the Air Force Wright Aeronautical Laboratories, DARPA Order No. 3119.

T.-J. Fan was with the Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, CA 90089. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

G. Medioni and R. Nevatia are with the Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, CA 90089.

IEEE Log Number 8929959.

There, objects are recognized by finding some distinguishing features and relations between them [6], [15]. However, to handle more complex situations where the number of models may be large and little *a priori* information about the scene may be available, it becomes necessary to first use sophisticated descriptions of the scene and then use these descriptions for recognition. This is the scenario assumed in the system described in this paper.

One can match two scenes at many different levels of descriptions with some tradeoffs: the lower the level of the descriptions, the easier it is to compute them. For example, a range array may be available directly as input and an Extended Gaussian Image [19] requires only the computation of surface normals. However, low level descriptions are not invariant to viewing directions and little tolerant to occlusion. The higher level descriptions, on the other hand, maintain their invariance but the known algorithms to compute them are often weak and error-prone. The appropriate level of description to be used for matching thus depends on the expected variations in the scenes and on the state-of-art in computing descriptions of the scene.

We have decided, in this paper, to use object descriptions in terms of their surfaces. The surface of an object is described by segmenting it into *surface patches* and the complete description consists of the description of each patch separately, and their interrelationships. Such a description can be viewed as an *attributed graph* with the patches as the *nodes* and relations between them as the *links*. The segmentation and description of the surface is based on measured curvature properties of the surface. We describe this process briefly in Section III, complete details are given in [12], [13].

We believe that our chosen representation has many advantages for scene and object matching. The description is *rich*, so that similar objects can be identified, *stable*, so that local changes do not radically alter it, and has *local support* so that partially visible objects can be identified. It also enables us to recreate, from its features, a shape reasonably close to the original one. The surface descriptions are much higher level than pointwise or edge descriptions, but not as high as volumetric descriptions. The surface descriptions are invariant for smaller changes in viewing angle than would be the case for volume descriptions. However, techniques for volume descriptions are not yet fully developed, whereas our previous work allows us to have high-quality surface descriptions.

In general, there are two steps involved in object recognition: *building models* and *recognizing scene objects*. We think that a good recognition system should provide the following.

- It should automatically build object models from range images.
- The descriptions used for models and for scenes of unknown objects should be compatible, or at least it should be easy to go from one to the other.
- The search should be efficient in finding correspondences between models and scene objects.

We have decided to use the same description format for both model and scene objects. As a result, the computation of the model descriptions can be achieved automatically, and there is no need to *translate* one description into another. We have tested our system on a number of scenes containing multiple complex objects with occlusion, and obtained very good recognition results. Some examples are given later in Section V.

In the next section, we review existing object recognition systems. Section III provides a brief summary of how we go from a dense range map to a set of surface patches, and how these patches are further organized into partial objects. Section IV presents the details of our recognition system, which uses the descriptions given above. Section V shows results of our recognition system on real range images. Finally, Section VI summarizes our contribution.

II. SURVEY OF RECOGNITION SYSTEMS

In this section, we review the most successful object recognition systems to date, and point out how they address the issues of model representation, choice of scene features, and matching methods. We also discuss their limitations.

A. Extended Gaussian Image (EGI)

Horn *et al.* use multiview EGI models to recognize 3-D objects [19]–[21]. Each model object is represented by its mapping on the Gaussian sphere, and each scene object is also represented by an EGI. The EGI's of scene objects and model views are compared. To constrain the search space, the two EGI's are first aligned along the directions of minimum EGI mass inertia, then a match measure is specified by comparing the similarity in their mass distributions. The model that maximizes this measure is chosen as the matched model. This method has the advantage that EGI can be computed directly; no complicated description stage is needed. The main disadvantage is that EGI is sensitive to occlusion and is unique for convex objects only. Furthermore, when multiple objects are present in a single scene, it would be necessary to segment the EGI into regions corresponding to separate objects, and it is not clear how to achieve this, except for simple-shaped objects.

B. 3DPO

Horand and Bolles [18] and Bolles *et al.* [6] developed the 3DPO system for recognizing and locating 3-D parts

in range data. The model consists of two parts: an augmented CAD model and a feature classification network. The CAD model describes edges, surfaces, vertices, and their relations. The feature-classification network classifies observable features by type and size. The system recognizes unknown objects by searching for features that match features for some model, for example, a cylindrical curve with a given radius. Then objects are hypothesized by determining whether a pair of observed segments are consistent with a given model's feature. This system has been shown to recognize objects in highly complex scenes, but with very few models. Since the models are represented in 3-D, it is very difficult to compute them automatically, therefore a CAD model is required that usually needs help from a user, it also needs a very complex network. Furthermore, this system relies heavily on detecting circular arcs and straight dihedral edges, so the shape of the objects it can recognize is restricted.

C. Grimson and Lozano-Pérez

Grimson and Lozano-Pérez [15], [16] discussed how local measurements of 3-D positions and surface normals can be used to identify and locate objects from among a set of unknown objects. Models consist of polyhedral objects represented by their planar faces. The information about these faces (such as their equations) and the relations between faces (such as distance) are also computed. Sparse range or tactile data of 3-D objects are used as scene features. The matching process contains two steps: in the first step, a set of feasible interpretations of the sensory data is constructed. Interpretations consist of pairings of each sensed point with some object surface of one of the models. Interpretations inconsistent with local constraints are discarded. In the second step, the feasible interpretations are verified by a transformation test. An interpretation is accepted if it can be used to solve for a transformation that would place each sensed point on an object surface. Only polyhedral objects or objects with a sufficient number of planar surfaces can be used in this system.

D. Ikeuchi

Ikeuchi [22] developed a method for object recognition in bin-picking tasks. Object models are generated under various viewer directions, apparent shapes are then classified into groups. Since this system is mainly designed for tasks for bin-picking, only *one* type of object, which is the same one as in the model, appears in the scene. The same surface features used in models are extracted and classified by the help of the model. In the recognition process, an *interpretation tree* is generated according to various model views. The orientation and location of the scene object are then decided by comparing their surface features and classified by the interpretation tree. Strictly speaking, this is not a recognition system, but a system which identifies a given 3-D object to be picked up.

E. Faugeras and Hebert

Faugeras and Hebert [14] developed a system to recognize and locate rigid objects in 3-D space. Model objects are represented in terms of linear features such as points, lines, and planes. The same features such as significant points, lines, and planes are used to describe scene objects. The system uses *rigidity constraints* to guide the matching process. At first, possible pairings between model and scene features are established, the transformation is estimated using quaternions. Then, further matches are predicted and verified by the rigidity constraints. This system has been shown to recognize fairly complex objects, but in scenes without occlusion from other objects. As the segmentation of objects in this system results in a large number of surface patches, and since the segmentation is not guided, we believe it is going to be highly sensitive to small changes in the surfaces.

F. Oshima and Shirai

Oshima and Shirai [25], [26] developed a model-based recognition system for objects with planar and curved surfaces. Each model is represented by a *relational-feature graph* whose nodes represent planar or smoothly curved surfaces, and links represent relations between adjacent surfaces. Matching is achieved by a combination of data-driven and model-driven searching processes. At first, *kernel nodes* which consist of large, planar surfaces with no occlusion are extracted. Next, an exhaustive search of all model graphs is performed and those containing regions which match the kernel nodes are selected as candidates. Finally, a *depth-first search* is applied to build the correspondences for remaining surfaces. Since only one view is used for each model object, if objects may be viewed from multiple directions, then a separate relational graph must be constructed for each view, and these models must be treated independently by the matching process. Furthermore, no occlusion is allowed for curved surfaces.

G. Nevatia and Binford

Nevatia and Binford [24] developed a system in mid-1970's that used generalized cones as the basic representation. This system uses range data as input. Model acquisition and scene description follow the same procedure, hence the models can be acquired automatically. This system also handles articulation of parts of objects. It is tested on objects of the complexity of a doll and a horse. This system also uses an *indexing* scheme based on gross features so that recognition does not require a search through all models in the database. The main limitation of this system would come from the ability to generate appropriate generalized cone descriptions for complex objects. This is a difficult problem as we discuss later.

A. ACRONYM

Brooks [9], [10] developed an image understanding system called ACRONYM which uses generalized cylin-

ders for descriptions of model and scene objects. The model objects are represented by hierarchical graphs of primitive volumes described by generalized cylinders (GC). The user constructs a tree for each object, where nodes contain parts of objects represented by GC, and links represent their subpart relation. The user is also required to construct a model class hierarchy called a *restriction graph*. This graph contains sets of constraints for different classes of objects and is used later to guide the match between model and scene objects. *Ribbons* and *ellipses* are used to describe scene objects. The descriptions are finally represented by an *observation graph* whose nodes contain ribbon and ellipse descriptions and links specify spatial relations between nodes. Matching is performed at two levels: first, predicted ribbons must match image ribbons, and second, these local matches must be globally consistent. The main restriction of ACRONYM is that models and restriction graphs are constructed by the user, which is very expensive and restricts the possibility of automatic model building. Furthermore, since both models and scenes are represented by GC's that consist of ribbons and ellipses, the shape of model and scene objects is restricted; in addition, the viewing direction is assumed to be approximately known.

Other contributions in object recognition are due to Bolles and Cain [5], Bhanu [4], Ayache [1]. A more detailed survey can be found in [2], [7], [11].

III. COMPUTING SYMBOLIC SCENE DESCRIPTIONS

As discussed in Section I, we have chosen segmented surface descriptions for representing objects in this work. Discontinuities such as *jump boundaries*, *limbs*, and *creases* are chosen to segment surfaces as they are explicit features of the surfaces of 3-D objects. Our method consists of two stages. In the first stage, surfaces objects are segmented at discontinuities which can be detected by examining the *zero-crossings* and *extremal* values of surface curvature measures. Then these detected discontinuities are used to segment a complex surface into simpler meaningful components called *surface patches* or *patches*. These patches can then be approximated by simple surface models. Finally, these surface patches are grouped into meaningful 3-D objects, and *attributed graphs* are generated to describe these objects.

A. Extracting and Describing Surface Patches

Given a complex surface, we wish to decompose it into simple surface patches. Our approach is to determine the boundaries of the surfaces by computing *local* properties and then inferring the patches from them. Similar ideas for segmentation have been used in [3], [8]. In particular, we seek to find the following kinds of boundaries.

- 1) *Jump boundaries*—where the surface undergoes a discontinuity.
- 2) *Creases*—which correspond to surface orientation discontinuities.

We infer these boundaries from curvature properties of the surface. In particular, we use curvature *zero-crossings*

and *extrema* [12]. A *jump boundary* creates a zero-crossing of the curvature in a direction normal to that of the boundary; a *crease* causes a local extremum of the curvature at that point. Crease boundaries may also create zero-crossings away from the location of the boundary itself.

These features, when connected using contiguity, give us *partial* boundaries for patches in which the surface should be segmented but not necessarily a *complete* segmentation. These partial boundaries are then completed by simple extension [12]; we have found this process to be satisfactory for the large number of examples we have tested it on. The resulting *regions* are assumed to correspond to elementary surface patches. These regions could be segmented further, either based on the region shape or on the results of surface fitting; we have not found it necessary to do so for the examples we have tried.

The surface patches are approximated by a second-order polynomial in (x, y, z) , whose coefficients are computed by a least-squares method. The details are given in [12]. Figs. 1 and 2 illustrate this early processing on a synthetic and a real range scene, respectively.¹ In both figures, (a) shows the shaded image, (b) shows the curves obtained by our feature detection process, and (c) shows the regions bounded by the previous curves.

B. Object Inference

At the end of the above process, we have a symbolic representation of a scene as a *graph* whose nodes represent the patches and whose links express geometric relationships between patches, making it an appropriate level to use in a matching process. We can further group these patches into *objects*, or rather visible faces of objects, by reasoning on the type of connections between adjacent patches. This higher level of description facilitates the matching process. The procedure used to obtain these *objects* is described in detail in [12], we only provide the outline below.

Up until now, we have classified the boundaries into two classes only: jump boundaries and creases. Once each patch is approximated by an analytic function, it becomes possible to distinguish true jump boundaries from *limbs* (also called axial contour generators [28]), for which the normal to the surface becomes perpendicular to the viewing direction. This distinction is important for matching, as limbs are *not* intrinsic properties of an object, but depend on the viewing direction. We therefore end up with the following set of four labels: *convex creases* (+), *concave creases* (-), *limbs* (L), and *jumps* (J).

¹Most researchers display range images by encoding depth by grey level, but this produces images with very poor dynamic range; in the rest of the paper, we present range images by borrowing a technique from computer graphics: we assume that the object is Lambertian, compute the normal to the surface in a small (3×3) neighborhood, and generate a *reflectance* image in which the intensity is inversely proportional to the angle between the light source and the surface normal. (The brightness of a point on the surface is proportional to the cosine of the angle between the light source and the surface normal under the Lambertian assumption.)

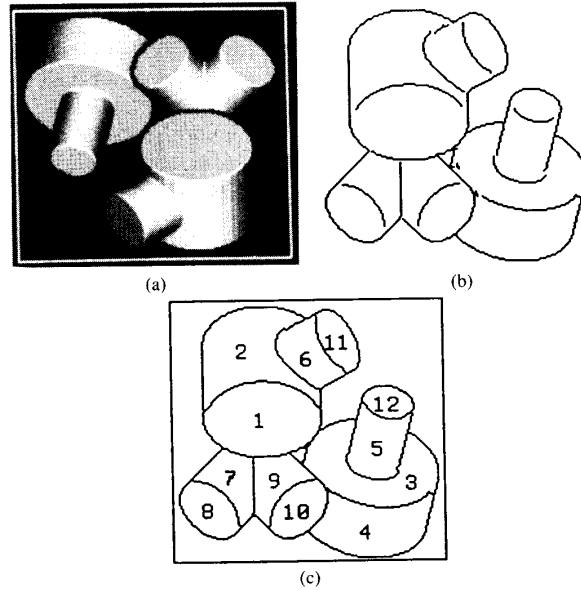


Fig. 1. Segmentation of a synthetic range image: (a) original scene, (b) detected discontinuities, (c) result of segmentation.

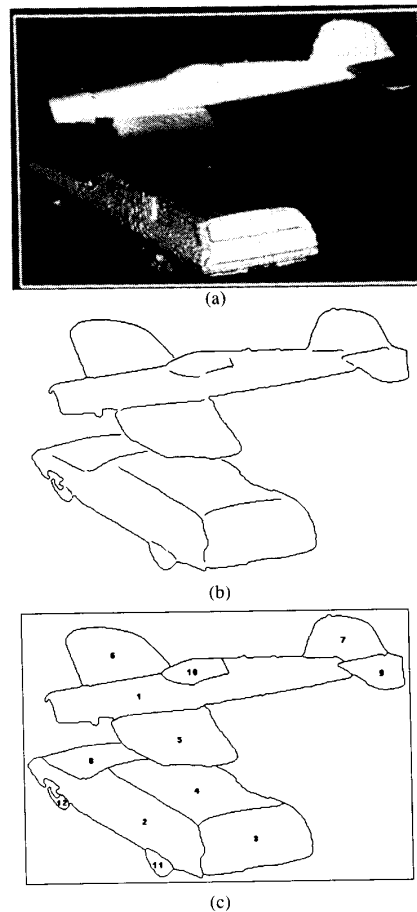


Fig. 2. Segmentation of a real range image: (a) original scene, (b) detected discontinuities, (c) result of segmentation.

The match primitives are composed of graphs and subgraphs. We create a *node* for each surface patch, which contains the *unary* information about the surface patch such as the shape, orientation, and location. Then, for each pair of nodes that share a common boundary, we create a *link* between them. This link contains the *binary* information between these two nodes such as the boundaries and the *possibility* that these two nodes (which represent two surface patches) belong to the same object. Note that one node contains one surface patch only, but one link may include multiple boundaries.

Surfaces are only parts of solid objects. In a segmented scene, the type of adjacency between two patches conveys strong information regarding whether or not these two patches belong to the same object. Based on our labels, the adjacency information we can derive is that of *occlusion* or *connectivity*.

From the type of adjacency relationships, it is possible to generate hypotheses about objects. We do this by looking at each triplet (S_i, S_j, b_k) of surface patches S_i and S_j connected by boundary b_k with the label k . Whenever k is a convex crease, we directly conclude that S_i and S_j belong to the same object. Such a strong conclusion, however, cannot be made about the other junction types. We have chosen to compute the possibility p that two patches belong to the same object. This number p , between 0 and 1, encodes our belief that two patches belong to the same object. p is 1 for a convex crease, 0.75 for a concave crease, and between 0 and 0.5 for a limb or jump, depending on the distance between the two patches in the vicinity of this boundary. The precise value in the last case is computed as follows: let D be the smallest distance between the two patches S_i and S_j , along the common boundary b_k (for simplicity, we compute the distance along the line of sight, rather than the true Euclidean distance). Let D' be the "thickness" of the occluding patch along b_k . Then the possibility of connection is given by $0.5 \times (1 - \min(1, D/D'))$.

When two nodes are connected by more than one type of boundary, the value of the link between them is the maximum of the individual values (an example of this can be seen in Fig. 1(c) where regions 3 and 5 are linked by a concave crease *and* a jump boundary).

Finally, the links with p less than some threshold (0.3 for all of our examples) are removed, which means the two nodes are considered not likely to belong to the same object. Thus, we obtain a partition of the original graph into a set of subgraphs with no links between them, each subgraph representing one (partial) object. Figs. 3 and 4 show the results of object inference on the previous images where (a) shows the inferred objects in which different objects are represented by different textures, (b) shows the graph where circles represent the nodes whose numbers refer to the patch numbers shown in the result of segmentation, and lines represent links with a value corresponding to the connection possibility p . It should be noted that the grouping may not be perfect, as is the case in Fig. 4 where the right wing of the plane is inferred as

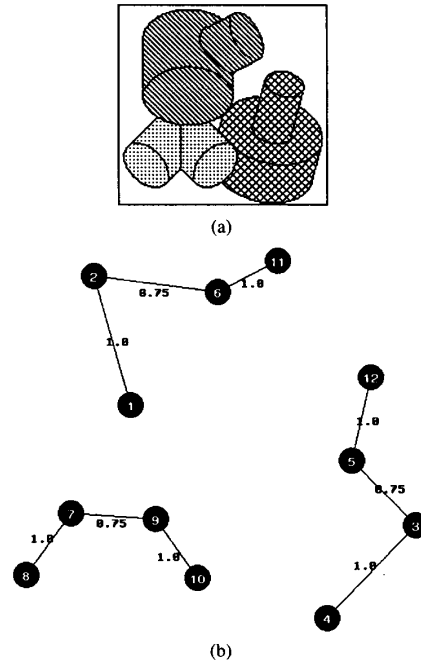
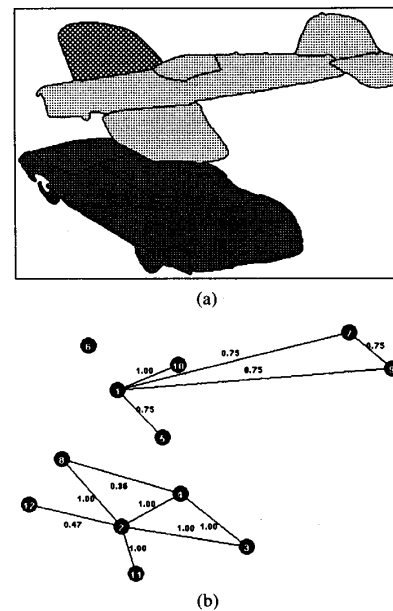


Fig. 3. Objects and graphs for the synthetic range image: (a) inferred objects, (b) graphs.



the corresponding surface patch, and a link I_{ij} between *connected* nodes n_i and n_j represents the relationships between them. The choice of these attributes is largely intuitive. However, their descriptive power and robustness has been demonstrated by testing on a large number of examples with no changes to the parameter values. Some of the attributes (such as the averages of the principal curvatures, given below) are global and could change with varying amounts of occlusion. However, as explained later, these attributes are used only to select a subset of likely models in the matching process; recognition itself involves evaluating a geometrical transformation between the data and the model and is not directly dependent on these gross measures. The attributes we use are described below.

1) *Node Attributes*: For each visible surface patch n_i , we compute the following:

- *Visible area $A(i)$* : This is the 3-D area of the visible surface patch. It is computed from the image of the patch in 2-D and the knowledge of surface normal at each point of the patch, by making a piece-wise planar approximation of each pixel. The 3-D area is given by $\sum_j 1/\cos(\alpha_j)$, where α_j is the angle between the surface normal and the line of sight at the j th pixel and the sum is over all pixels in the image region corresponding to the surface.

- *Orientation $\vec{n}(i)$* :

- For planar surfaces: the direction of the normal.
- For cylindrical or conic surfaces: the direction of the axis.
- For other surfaces: the direction of least curvature, which is defined as follows: let $\kappa_\theta(p)$ denote the curvature at point p in direction θ , and let $\kappa_\theta = \sum_{p \in n_i} |\kappa_\theta(p)|$. Then the orientation is chosen as the direction θ for which κ_θ is the smallest, the direction along which the patch is the least curved.

- *Average of Principal Curvatures $K_1(i)$ and $K_2(i)$* : $K_1(i)$ is the average of the maximum curvature at each point of the patch n_i ; similarly $K_2(i)$ is the average of the minimum curvature at each point. Note that these are not the same as the “mean” curvatures commonly used in differential geometry.

- For planar surfaces: $K_1(i) = K_2(i) = 0$.

- For other surfaces: let $\vec{n}(i)$ represent the orientation of the surface, and V_1 the projection of $\vec{n}(i)$ on the x - y -plane. Let V_2 be the vector on the x - y -plane perpendicular to V_1 . Then $K_1(i)$ is the average curvature at every point of the surface patch along the direction of V_1 and $K_2(i)$ that of V_2 .

Hence $K_1(i)$ and $K_2(i)$ reflect the *flatness* of the patch.

- *Estimated Ratio of Occlusion $R(i)$* : A surface patch n_i is said to be occluded by another surface patch n_j if there exists a limb or jump contour c_{ij} between n_i and n_j where the depth value of n_i in the vicinity of c_{ij} is less than that of n_j (note that depth is encoded such that the higher the value, the closer the point is to the viewer). Let $L_{\text{occ}}(i)$ denote the total length of set of boundaries of n_i that are occluded by other patches, and $L_{\text{tot}}(i)$ be the total length

of the boundaries of n_i , then the estimated ratio of occlusion R of n_i is equal to $L_{\text{occ}}(i)/L_{\text{tot}}(i)$.

- *Centroid $\vec{C}(i)$* : The centroid of n_i is given by the average of the (3-D) coordinates of the visible surface points belonging to n_i .

2) *Link Attributes*: For each pair of nodes n_i and n_j connected by at least one boundary, the link l_{ij} expresses the following.

- *The Type of Adjacency $t(i, j)$* : The adjacency between n_i and n_j can be any combination or none of the following:

- n_i is occluded by n_j at a jump or a limb,
- n_j is occluded by n_i at a jump or a limb,
- n_i and n_j are connected by a convex crease,
- n_i and n_j are connected by a concave crease.

- *The Connection Possibility $p(i, j)$ as Given Earlier*:

- $p = 1$ if n_i and n_j are connected by a convex crease, otherwise,
- $p = 0.75$ if n_i and n_j are connected by a concave crease, otherwise,
- $0.3 \leq p \leq 0.5$ if n_i occludes (is occluded by) n_j .

These attributes that we associate to nodes and links are the ones which we found most appropriate for establishing correspondences.

IV. OBJECT RECOGNITION

Matching between the objects in a scene and the database of the models is performed by three modules: the *screener*, in which we find the most likely candidate views for each object, the *graph matcher*, which performs a detailed comparison between the potential matching graphs and computes the 3-D transformation between them, and the *analyzer*, which takes a critical look at the results and proposes to split and merge object graphs. Since the matching program is rather intricate, we start by giving an overview of the core ideas, then describe each module in detail. Finally, we present a detailed case study on one of our test scenes in Section V.

A. Overview of the Matching Process

We have decided to use multiview surface models in our recognition system. Each model consists of several views (2–6, chosen by us based on the complexity of the object). These views are taken so that most of the significant surfaces of the model objects are contained in at least one of these views.

We want to recognize *objects* in occluded *scenes*. Each scene S_j may contain multiple (unknown) objects $S_j^1, S_j^2, \dots, S_j^{N_j}$ with self and mutual occlusion. The scene is processed and described using the method presented in the previous section. Each scene object S_j^k is represented by an attributed graph.

We have access to a database of several known objects, called *models*, M_1, M_2, \dots, M_N . Each model M_i consists of several *views* $M_i^1, M_i^2, \dots, M_i^{N_i}$, and each view is represented by an attributed graph computed as before.

The goal of the matching process is to find, for each object in the scene, the most similar model view. The ob-

ject is then recognized as the model which contains that view.

The matching process contains three major blocks: the *screener*, the *graph matcher*, and the *analyzer*. The top-level block diagrams of these three modules are shown in Figs. 5, 6, and 7, respectively.

• *Screener*: This module serves to find the likely model view candidates for each object in the scene. It is a fast search involving the computation of coarse differences in properties of the nodes of the graphs. The output is an ordered list (with at most 5 elements) of candidates. The details are given in Section IV-B.

• *Graph Matcher*: Once this list of candidates has been computed for each scene object, we perform an extensive comparison between the graphs representing the model view and the object, until one match is found to be "good enough" (defined later), or there are no candidates left.

The strongest constraint imposed by the matching process involves the geometric transform (rotation and translation) of a rigid object. However, this constraint cannot be applied until *after* a candidate match has been computed! As a result, we rely on weaker, partial constraints, and on an incremental estimate of the true transform.

The graph matching procedure consists of finding the pairs (*model node*, *object node*) forming the largest set consistent with a single rigid 3-D transform. We begin by finding all the possible pairs $\langle m, s \rangle$ where m and s are the model and object nodes, respectively. Then we compare the attributes of these nodes. For instance, a planar patch in the model cannot correspond to a cylindrical patch in the object. Once we have these pairs, we incrementally group them into sets consistent with a 3-D transform. This is done by tree search, updating the transformation as we go. The details of this process are given in Section IV-C.

• *Analyzer*: The previous steps can be improved with critical feedback. Two possible deficiencies of previous steps are:

- The segmentation of the scene graph may not be perfect, especially if objects are close together.
- The heuristic arguments invoked during matching may have led to wrong pairings or discarded valid ones.

There is a module in our system, called the *analyzer*, which tries to detect and correct such errors by merging unmatched objects with existing matches, or by splitting objects with many unmatched nodes into smaller objects. The details are given in Section IV-D.

B. Module 1: Screener

In principle, the number of model views and scene objects may be large, and evaluating each pair to find possible correspondences would be prohibitively expensive. Instead, we use a heuristic method to order the model views for every scene object S , according to coarse differences between S and these views, and use only the highly ranked views for detailed matching. This process significantly reduces searching time.

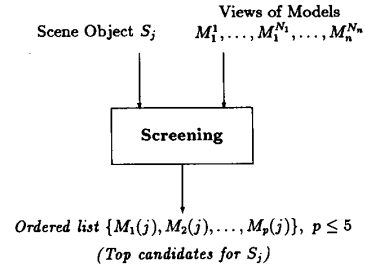


Fig. 5. Block diagram for the screener.

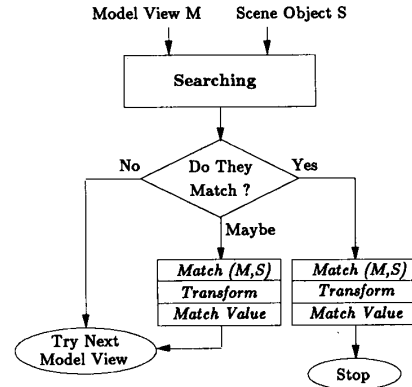


Fig. 6. Block diagram for the graph matcher.

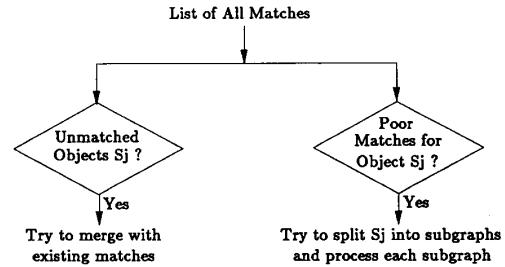


Fig. 7. Block diagram for the analyzer.

To measure the difference between two graphs, we first introduce a normalized measure between 0 and 1 as follows:

$$d(a, b) = \frac{|a - b|}{\max(a, b)} \quad (1)$$

The following *differences* are computed between each model view \mathfrak{M} and scene object S :

- *Number of Nodes*: Let $N(\mathfrak{M}, S) = d(N_{\mathfrak{M}}, N_S)$ where $N_{\mathfrak{M}}$ and N_S denote the number of nodes in \mathfrak{M} and S , respectively.
- *Number of Planar Nodes (Surface Patches)*: Let $R(\mathfrak{M}, S) = d(R_{\mathfrak{M}}, R_S)$ where $R_{\mathfrak{M}}$ and R_S denote the number of planar nodes in \mathfrak{M} and S , respectively.
- *The Visible 3-D Area of the Largest Node*: Let $A(\mathfrak{M}, S) = d(A_{\mathfrak{M}}, A_S)$ where $A_{\mathfrak{M}}$ and A_S denote the visible 3-D area of the largest nodes in \mathfrak{M} and S , respectively.

If any of the following conditions occurs, \mathfrak{M} and \mathfrak{S} are considered not similar enough and the model view \mathfrak{M} is discarded:

- $N(\mathfrak{M}, \mathfrak{S}) > 40$ percent
- $R(\mathfrak{M}, \mathfrak{S}) > 30$ percent
- $A(\mathfrak{M}, \mathfrak{S}) > 30$ percent.

If they pass the above test, the *difference* between \mathfrak{M} and \mathfrak{S} is measured by

$$\alpha(\mathfrak{M}, \mathfrak{S}) = N(\mathfrak{M}, \mathfrak{S}) + R(\mathfrak{M}, \mathfrak{S}) + A(\mathfrak{M}, \mathfrak{S}). \quad (2)$$

A model view \mathfrak{M}_1 is said to be more similar to scene object \mathfrak{S} than model view \mathfrak{M}_2 if $\alpha(\mathfrak{M}_1, \mathfrak{S})$ is smaller than $\alpha(\mathfrak{M}_2, \mathfrak{S})$.

The output of the screening module is an ordered list $P(\mathfrak{S}) = \{M_1, M_2, \dots, M_N\}$ of model views such that $\alpha(\mathfrak{M}_i, \mathfrak{S}) \leq \alpha(\mathfrak{M}_j, \mathfrak{S})$ for all $i < j$. In addition, we impose the restriction that $N \leq 5$.

C. Module 2: Graph Matcher

The purpose of this module is to find, from a small list of candidate model views produced by the screening module, the most likely view, if any, to correspond to a given scene object. The problem is therefore to find the largest subgraph in the model view for which every node maps onto a node of the object graph according to the geometric operations which transform the view onto the object.

Exhaustive search on all possible sets of pairs is an exponential process, so we perform a two-stage depth-first exploration on this tree, whose block diagram is shown in Fig. 8.

a) Computing All The Possible Pairs: For each pair $\langle m, s \rangle$, where m and s are model and scene nodes, respectively, we check whether or not they are compatible (according to relation ξ_0 defined later), and if they are, we assign a measure of goodness to this pair.

b) Search Stage 1: We try to incrementally build a set with 4 pairs. This is done by depth-first tree search: we expand, that is, we try to add a pair to, the (ordered) pairs of this tree in a depth-first manner until a “good enough” set of pairs is found. Since we cannot enforce the compatibility relation based on the true 3-D transform, we rely instead on weaker constraints (ξ_1 – ξ_6 as defined later) and on the current estimate of the true transform (ξ_7). We associate a measure of goodness with each set; if a set of size 4 has a high enough measure, the search terminates, otherwise it continues. This restriction to four pairs is imposed in order to focus the search on the most promising paths only.

c) Search Stage 2: We now look for the largest set containing the *best* set found in the previous step. The search tree is expanded from that set only, in depth-first fashion, using the same compatibility constraints as before. It may appear that the search is too focused, and that we may miss promising nodes at earlier stages, but one should remember that we are interested in all the pairs in the path from the root to a leaf, regardless of their order. Therefore, if an unexpanded pair at an earlier level of the tree

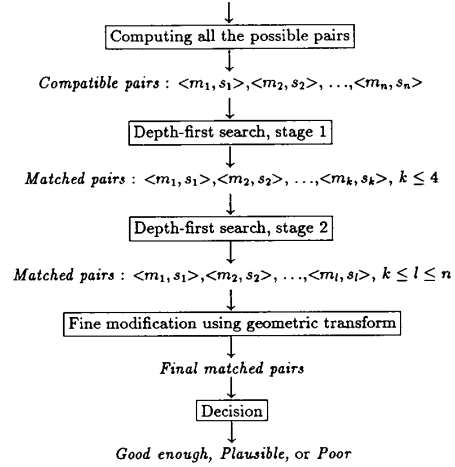


Fig. 8. Block diagram for the graph matcher.

is compatible with the expanded path, it will eventually be appended to it.

In both stage 1 and stage 2, the search terminates if one of the following conditions occurs.

- i) The current path is “good enough” (defined precisely in Section IV-C-5).
- ii) There is no more possible expansion.
- iii) The number of expanded nodes is large enough.

d) Fine Modification: We now have a good estimate of the actual transform, and can therefore enforce a strong compatibility constraint between pairs in a set of matches. As a result, we may include pairs that were rejected by the approximate constraints, or reject pairs that were included. We update the transform and the measure of goodness after such modifications.

e) Decision: We have two thresholds, $\mathcal{J}C_1$ and $\mathcal{J}C_2$, with $\mathcal{J}C_1 < \mathcal{J}C_2$, on the measure of goodness $\mathcal{J}C$ of the maximal set of matches.

- If $\mathcal{J}C < \mathcal{J}C_1$, the object is considered not to correspond to the model.
- If $\mathcal{J}C \geq \mathcal{J}C_2$, the match is considered good, the object matches the model, and the search stops.
- If $\mathcal{J}C \leq \mathcal{J}C_2$, the match is considered plausible, but the next candidate model views are also evaluated. Finally, the match with the highest $\mathcal{J}C$ is selected.

We now give in detail the definition of our compatibility relationships, our similarity measures, and the fine modification procedure.

1) Compatibility Between Nodes of Model View and Scene Graph (ξ_0): In measuring the similarity between two nodes m of the model view and s of the scene object, we first compute the normalized measure (1) of the *difference* for each of the following properties:

- a) $d_{m,s}(1) = d(A_m, A_s)$, where A_m and A_s represent the 3-D visible area of m and s , respectively.
- b) $d_{m,s}(2) = d(K_m, K_s)$, where K represents the average curvature K_1 .

c) $d_{m,s}(3) = d(k_m, k_s)$, where k represents the average curvature K_2 .

The nodes m and s are said to be ξ_0 -compatible if and only if:

- $d_{m,s}(1) < 0.30 + 0.70 \times \max(R_m, R_s)$, where R_m and R_s represent the estimated ratios of occlusion of nodes m and s , respectively.
- $d_{m,s}(2) < 0.30$.
- $d_{m,s}(3) < 0.30$.

If any of the above criteria does not hold, the two nodes are considered not ξ_0 -compatible, otherwise, the *similarity measure* of the two nodes m and s is defined by

$$d_{m,s} = \frac{\sum_{i=1}^3 w_i d_{m,s}(i)}{\sum_{i=1}^3 w_i} \quad (3)$$

where w_i represents the weight for each item $D_{m,s}(i)$. In our experiments, we chose $w_2 = 2$, and $w_1 = w_3 = 1$.

At the end of the above process, all the ξ_0 -compatible pairs are ordered according to their similarity measure. Then, in the graph matching module, the pairs are examined according to this order.

2) *Compatibility Between Two Pairs of Matching Nodes*: Everytime a pair of nodes $\langle m_i, s_i \rangle$ is selected, it is compared to all the already matched pairs $\langle m_j, s_j \rangle$ using a compatibility constraint. If this constraint is not satisfied, the chosen pair $\langle m_i, s_i \rangle$ is discarded. The constraint contains the following *consistency checks*.

a) *Uniqueness Consistency* (ξ_1): $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_1 -compatible if and only if $m_i \neq m_j$ and $s_i \neq s_j$.

b) *Connection Consistency* (ξ_2): Let l_1 and l_2 represent the links between m_i and m_j and between s_i and s_j , respectively. Then the types (i.e., limb, jump, convex, or concave) of l_1 and l_2 , denoted as t_1 and t_2 , respectively, are said to be ξ_2 -compatible if and only if one of the following satisfies:

- t_1 is equal to t_2 , or
- one of them is a jump and the other one is a convex crease (a change of view point may transform one into the other)
- either one or both of them is NULL (the nodes are not adjacent, this is possible especially when shadows occur).

Note that l_1 and l_2 may have multiple types, as explained in Section III-B earlier. In this case, we require that *any* of the multiple types match.

c) *Direction Consistency* (ξ_3): Let θ_1 and θ_2 denote the angles between the orientation of $\langle m_i, m_j \rangle$ and $\langle s_i, s_j \rangle$, and let $\theta = |\theta_1 - \theta_2|$, then the pairs $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_3 -compatible if and only if θ is less than a fixed threshold θ_{tol} . Here we choose $\theta_{tol} = 25^\circ$.

d) *Distance Consistency* (ξ_4): Let L_1 and L_2 denote the distance between the centroid of inertia of m_i and m_j , and s_i and s_j , respectively. Let

$$L = \frac{|L_1 - L_2|}{\max(L_1, L_2)}, \quad (4)$$

then, the pairs $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are said to be ξ_4 -compatible if and only if L is less than a threshold. Here we choose the threshold as 0.30.

e) *3-D Geometry Consistency* (ξ_5): For all the matched pairs $\langle m_k, s_k \rangle$ other than $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$, let \vec{U}_{ij} , \vec{V}_{ij} , \vec{U}_{ik} , and \vec{V}_{ik} represent the vector connecting the centroid of m_i to that of m_j , s_i to s_j , m_i to m_k , and s_i to s_k , respectively. Let θ_1 and θ_2 denote the *directed angle* from \vec{U}_{ij} to \vec{U}_{ik} , and from \vec{V}_{ij} to \vec{V}_{ik} , respectively. And let

$$\theta = |\theta_1 - \theta_2|. \quad (5)$$

Then the three pairs $\langle m_i, s_i \rangle$, $\langle m_j, s_j \rangle$, and $\langle m_k, s_k \rangle$ are said to be ξ_5 compatible if and only if θ is less than a threshold. Here we choose the threshold to be 25° . This consistency check is mainly used to remove a match between two objects which are geometrically similar by mirror effect, as shown in Fig. 9.

If these four consistency conditions are fulfilled, we say that $\langle m_i, s_i \rangle$ and $\langle m_j, s_j \rangle$ are mutually consistent. This only happens, however, when no abrupt changes occur as a result of a difference in viewing angle. To take into account such changes, we define an additional condition as follows.

• *Enclosure* (ξ_6): To determine whether a surface patch m is (*partially*) *enclosed* in 2-D by another surface patch s , we start from the center point $C = (x, y)$ of m , where x and y are the first two coordinates of the centroid of m , and search outward in 8 directions, each 45° apart, if 6 or more out of the 8 searches encounter any point in surface s , we say m is enclosed by s . With this definition, we accept two pairs even if they fail all of the consistency conditions above except for condition 1, as long as m_i encloses (is enclosed by) m_j and s_i encloses (is enclosed by) s_j .

The motivation for this constraint is to allow looser matching of smaller regions contained inside larger and already matched regions (such as pushbuttons on the telephone in one of the examples shown later). The constraints ξ_2 - ξ_5 tend to be not reliable in such cases. Note that the matches accepted at this stage must still agree with the geometric transformation computed later.

It should be noted that the above criteria are not perfect, especially when parts of the objects are occluded, and mostly serve to prune the search tree. The true compatibility between nodes is established by the computation of the geometric transform.

3) *Computing the Geometric Transform*: Computing the geometric transform between matched objects not only indicates how to bring matched objects in correspondence, but also helps to verify the matching process. If the error in the transform for the current partial match is too large, the match should be abandoned.

The matching process gives the correspondences between surfaces of model view \mathfrak{M} and scene object S . By extracting the orientation and location of the corresponding surfaces, we can compute the (geometric) transform which brings \mathfrak{M} in registration with S . Here we introduce a noniterative method in which the axis of the rotation is first computed using the orientation of the matched nodes,

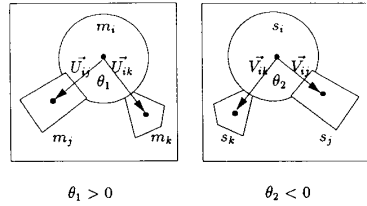


Fig. 9. Geometric similarity by mirror effect.

then the angle of the rotation is obtained using a two-level search. Finally, the translation is computed from the centroids of the matched nodes, using a least-square method. Since this method is noniterative, we believe it is faster than iterative methods which are used by many researchers [14], [17]. The details of the geometric transform computation are given below.

The rigid transform between objects can be represented in homogeneous coordinates by a 4×4 matrix as follows:

$$\begin{pmatrix} R_{(3 \times 3)} & T_{(3 \times 1)} \\ 0_{(1 \times 3)} & 1 \end{pmatrix} \quad (6)$$

where R represents the rotation and T the translation, respectively. A rotation can be represented by a rotation of angle θ about a unit vector k (axis of revolution) located at the origin. Let k_x , k_y , and k_z represent the three components of the axis k , then the relation between the coefficients of R and θ , k_x , k_y , k_z can be expressed as follows [27]:

$$R(k, \theta) = \begin{pmatrix} k_x k_x v + c & k_y k_x v - k_z s & k_z k_x v + k_y s \\ k_x k_y v + k_z s & k_y k_y v + c & k_z k_y v - k_x s \\ k_x k_z v - k_y s & k_y k_z v + k_x s & k_z k_z v + c \end{pmatrix} \quad (7)$$

where $s = \sin \theta$, $c = \cos \theta$, and $v = 1 - c$.

To find the axis k , the orientation vectors for each matched node pairs are first retrieved. Let $\langle m_i, s_i \rangle$, $i = 1, \dots, N$ be the matched node pairs between two objects \mathfrak{M} and \mathfrak{S} where $m_i \in \mathfrak{M}$, $s_i \in \mathfrak{S}$, and N is the number of matched node pairs, respectively. Let $\langle P_i, Q_i \rangle$ denote the orientation vectors of the matched node pair $\langle m_i, s_i \rangle$, as shown in Fig. 10 for $i = 1, 2$, where both orientation vectors have been brought to the origin O . Assuming the rotated angle is $\theta = 2a$ as indicated in Fig. 10, it is easy to show that the axis of revolution k for P_1 and Q_1 must lie on the plane L_1 that bissects the angle $P_1 O Q_1$. In other words, the angle between P_1 and its projection on L_1 is equal to that between Q_1 and L_1 , and this angle is equal to a (unless P_1 and Q_1 coincide at k , in this case, however, L_1 is an arbitrary plane that contains k). The same reasoning can be applied to all the other $\langle P_i, Q_i \rangle$ pairs. Thus to find the axis k , we only have to find all the planes L_i , then k is at their intersections. In our implementation, we find all the intersections k_{ij} of each two pairs L_i and L_j , then for each k_{ij} we compute the sum of the angle be-

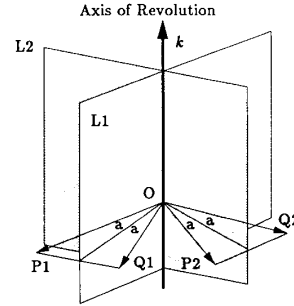


Fig. 10. Computing transforms.

tween all other k_{ij} , the one with the least sum is our choice of axis k .

The angle θ is found as follows. Let $\Theta(a, b)$ denote the angle between vector a and b . Given a rotation θ , for each matched pair $\langle m_i, s_i \rangle$, $\Theta(R(\theta) P_i, Q_i)$ is computed where $R(\theta) P_i$ represent the vector of P_i after rotating it by an angle θ . Then the best θ is found by minimizing the following equation:

$$E_\theta = \sum_i \Theta(R(\theta) P_i, Q_i). \quad (8)$$

After the rotation matrix R is computed, the translation T can be obtained easily. Let $\langle E_i, C_i \rangle$ denote the centroids of each matched pair $\langle m_i, s_i \rangle$ and $C'_i = R(\theta) E_i$. Then the translation T is as follows:

$$\frac{1}{N} \sum_i C_i - C'_i. \quad (9)$$

In our experiments, we have found that this computation gives highly accurate results; an example is shown later (in Section V-B).

4) Modifications Based on the Geometric Transform: Since the compatibility constraints presented previously are not perfect, some corresponding nodes may not have fulfilled the conditions imposed and may therefore have been rejected. Using the transform for the match allows us to rectify this situation. To achieve this, the transform between the model view and the scene object is computed first, using the current match L , then the model object is transformed and *superimposed* on the scene. Each surface s of the scene object is then checked by the following constraint ξ_7 :

- If s is not yet matched and there exists an unmatched model surface m whose transformed surface m' is close enough to s , then include $\langle m, s \rangle$ into the match L . Surfaces m' and s are said to be close enough if and only if the distance between the centroids of m' and s are less than twice the smaller *average width* of m and s , where the average width W of a surface m is computed as follows:

$$W = \frac{w_1 + 3w_2}{4} \quad (10)$$

where w_1 and w_2 indicate the distance from the 2-D center of the surface m to its farthest and closest boundaries, respectively.

• If s is a matched node and the corresponding transformed model node m' is not close enough, then $\langle m, s \rangle$ is removed from the match L .

After the modification, the transform is recomputed.

5) *Measuring the Goodness of a Match*: Throughout the graph matching procedure, a *match value* \mathcal{J} is attached to the current match which is computed as follows:

$$\mathcal{J} = \frac{\max(N_m, N_s) + \max(A_m, A_s)}{2} \quad (11)$$

where N_m , N_s , A_m , and A_s represent the ratio between the number of matched model nodes and the number of total model nodes, the number of matched scene nodes and the number of total scene nodes, the 3-D area of matched model nodes and that of all the model nodes, and the 3-D area of matched scene nodes and that of all the scene nodes, respectively. The highest value of \mathcal{J} is 1, which represents a complete match, and the lowest value of \mathcal{J} is 0, which means that nothing is matched. A match is considered "good enough" when \mathcal{J} reaches $\mathcal{J}_2 = 0.80$, and the graph matching step is immediately terminated. Otherwise, the search continues until the search tree can no longer be expanded. Finally, if the match value \mathcal{J} of the resulting match is less than $\mathcal{J}_1 = 0.60$, the match is discarded.

D. Module 3: Analyzer

Input scenes may contain multiple objects, and, as we have mentioned before, the object inference step may not produce perfect results. For example, when two objects touch, it is possible that we would consider them as one object instead of two, i.e., the result of object inference may produce just one *object shell* (i.e., a single graph) for these two touching objects. On the other hand, due to occlusions, shadows, or a special view point, real objects may appear as separate pieces in range image. In this case, more than one *object shell* is generated for different parts of the same object. In both cases, the match between model views and such objects will not be satisfactory. To correct this, a refining process which *splits* and/or *merges* objects according to current matches and their geometric relationships is applied next; we call this module the *analyzer*.

1) *Splitting Objects*: Suppose that our scene contains two objects A and B which are inferred as one object shell (represented by graph \mathcal{S}). Since we allow only one model view being matched for one object shell, let us assume that the model view \mathcal{Q} matches \mathcal{S} (as the result after step 2) where \mathcal{Q} is the view of the model object which is similar to scene object A . We also assume that there is another model view \mathcal{R} whose model is similar to scene object B . At this time, the match between \mathcal{R} and B has not yet been found. Fig. 11(a) illustrates this situation, where all surface patches are numbered for convenience. From the figure, we see that patch 3 of the scene is touching

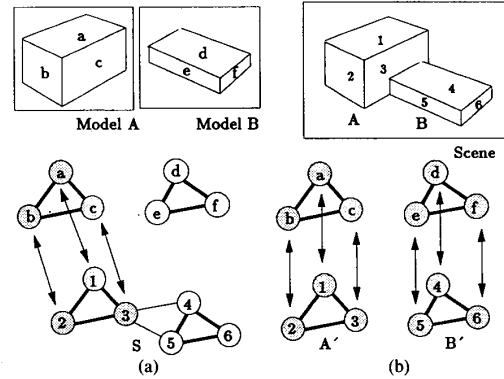


Fig. 11. Splitting objects.

patches 4 and 5. The scene graph \mathcal{S} and the model graphs \mathcal{Q} and \mathcal{R} are also shown. The shaded nodes represent the currently matched nodes (the correspondences between two matched nodes are indicated by bidirectional arrows). After matching, we find that half of the nodes in \mathcal{S} are not matched (surfaces 4, 5, and 6); however, by examining \mathcal{S} , it is possible to *split* the two objects in the scene.

The following two rules are followed in splitting objects.

- \mathcal{R}_1 : Splitting can only be achieved by removing links for which p is strictly less than 1.
- \mathcal{R}_2 : The set of already matched patches should not be split into subsets.

By looking at the scene and the graph \mathcal{S} , we know that surfaces 1-2, 2-3, 1-3, 4-5, 4-6, and 5-6 are connected by convex creases ($p = 1$) and from Rule 1 they can not be split (displayed by thick lines); however, surfaces 3-4 and 3-5 are connected by concave creases ($p = 0.75$) and may be split (displayed by thin lines). By removing the links 3-4 and 3-5 from \mathcal{S} , we obtain two separate graphs \mathcal{Q}' (which contains surfaces 1, 2, and 3) and \mathcal{R}' (which contains surfaces 4, 5, and 6). Furthermore, by looking at the match we notice that the *matched surfaces* (surfaces 1, 2, and 3) are contained only in \mathcal{Q}' , thus the result of splitting does not break Rule 2.

After splitting, we now can match scene object \mathcal{R}' to model view \mathcal{R} and the result is shown in Fig. 11(b).

From the above reasoning, we have derived the following procedure to split objects:

- After graph searching, if there exists a *matched* scene object \mathcal{S} such that more than 40 percent of its nodes are not matched, try to split it.
- Split \mathcal{S} according to the two rules \mathcal{R}_1 and \mathcal{R}_2 stated above. Let \mathcal{S}_1 represent the split object which contains all the already matched nodes, and $\mathcal{S}_2, \mathcal{S}_3, \dots, \mathcal{S}_N$ represent the remaining split objects sorted in the descending order of number of containing nodes.
- Reconnect* $\mathcal{S}_2, \mathcal{S}_3, \dots, \mathcal{S}_N$ into one object \mathcal{S}' , try to match it with model views, if no model views can be matched, remove \mathcal{S}_N (which contains the least number of nodes) and retry the match. The idea is that we want to match from the largest possible objects.

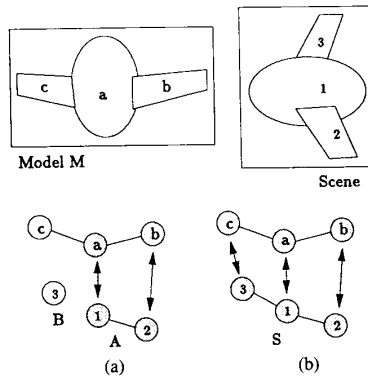


Fig. 12. Merging objects.

d) Repeat steps a)–c) until no more splitting is achievable.

2) *Merging Objects*: Suppose that our scene contains only one object S ; however, due to self occlusion, it is inferred as two object shells (represented by graphs \mathcal{A} and \mathcal{B} , respectively). Also assume that after graph searching, \mathcal{A} is matched to a model view \mathcal{M} (which is supposed to match the entire object S) and \mathcal{B} is not matched (usually because it contains too few nodes). This is illustrated in Fig. 12(a) where \mathcal{A} contains two *matched* surfaces 1 and 2, \mathcal{B} contains one *unmatched* surface 3, and model view \mathcal{M} contains three surfaces a , b , and c . In the current match, surfaces a and b match surfaces 1 and 2, respectively, while surface c is not matched (which is supposed to match surface 3 of \mathcal{B}).

To refine the match, merging can be used to bring separate pieces that actually belong to the same object. The idea is that by merging two objects \mathcal{A} and \mathcal{B} into one object S (which matches \mathcal{M}), not only the similarity constraints between corresponding *matching* nodes ($\langle c, 3 \rangle$ in this example) should be satisfied, but the binary constraints and the transform constraints between each pair of matching nodes ($\langle a, 1 \rangle$, $\langle b, 2 \rangle$, and $\langle c, 3 \rangle$ in this example) should also be satisfied.

In the example we show here, \mathcal{A} and \mathcal{B} are merged into S and the new match result is shown in Fig. 12(b).

From the above reasoning, we have derived a general procedure to merge objects as follows:

a) After graph searching, try to merge any *unmatched* scene object \mathcal{B} .

b) Select among matched objects the one which is geometrically closest to \mathcal{B} . Let \mathcal{A} , \mathcal{M} , and L denote the selected object, its matched model view, and the match between \mathcal{A} and \mathcal{M} , respectively.

c) Use the graph searching process presented in the previous section to find the correspondences between the nodes in \mathcal{B} and the *unmatched* nodes in \mathcal{M} , assuming that the match L has already been established. More matches can be added to L only if the resulting match meets the constraints of node similarity, binary constraints between each pair of matching pixels, and the transformation constraint.

d) Repeat steps a)–c) until either no more unmatched scene objects exist or no more merging is achievable.

V. EXPERIMENTAL RESULTS

In this section, we show the results and an evaluation of our recognition process, using a database of 10 objects, resulting in 32 views. We first show a detailed case study, then show results on a few more scenes.

A. The Models

In this work, we have selected 10 objects to build models. Then scenes which consist of these objects are acquired and recognized using these models. The model objects include a car, a chair, a telephone, a table, a mask, a hatchback car, a wagon, two boats, and an airplane. In order to save space, we only show *one* view for each of these objects in Fig. 13, but it is important to remember that the database consists of *all* the views for all the objects, 32 in all.

Our data comes from an active range finder using a light-stripe and triangulation technique, it is described in detail in [23]. The system consists of a laser, a video camera, a video monitor, a terminal, and a computer-driven rotary table.

Our composite scenes were obtained by first scanning each object in an arbitrary position and orientation, then combining these objects (synthetically) to generate the scene as it would have appeared if we had scanned it. This is necessary because of the technical difficulty in actually scanning the scene using a rotary table, which would create too many shadow regions.

B. A Detailed Case Study

In order to better understand how the system works, we take one of our test scenes, shown in Fig. 2(a), and follow each step of the system as it processes this scene. Table I shows the computation times for the various steps of processing on a Symbolics 3645 computer (with a floating point accelerator and 1 Megaword of memory). Note that the recognition takes only a small fraction of the total time. This is consistent with our design philosophy that good representations vastly simplify recognition. Also note that most of the time is spent on relatively simple, local computations that can be performed in parallel in a straightforward way.

1) *Search Nodes Expanded in Recognition*: As mentioned before, the object inference cannot be expected to always generate perfect results. In the scene, the right wing of the airplane is segmented into a separate object as the result of object inference, shown in Fig. 14, where different textures represent different objects. Thus, at the beginning of the recognition process, the scene consists of three “objects”: the airplane without the right wing, the wagon, and the wing.

The first step of the recognition process is to screen the model views for each of the three objects. The single wing, which consists of only one surface patch, does not find any candidate model views, thus is ignored tempo-

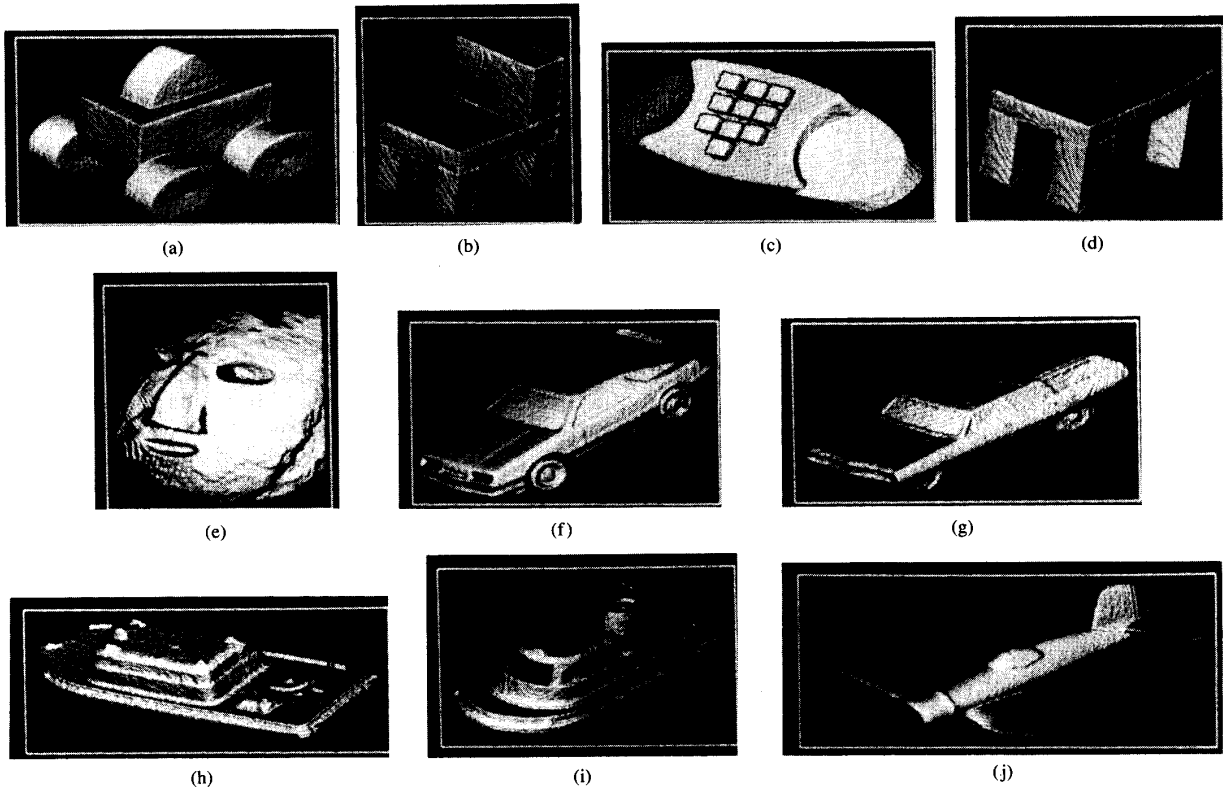


Fig. 13. Model views: (a) model car (4 views), (b) model chair (4 views), (c) model phone (4 views), (d) model table (4 views), (e) model mask (6 views), (f) model hatchback (2 views), (g) model wagon (2 views), (h) model Boat 1 (2 views), (i) model Boat 2 (2 views), (j) model plane (2 views).

TABLE I
DETAILED TIMING INFORMATION FOR THE FIRST SCENE

	Step		Overall
CV	Time: 693 seconds		17%
	Gaussian convolution	50%	
	$L \circ G$ convolution	50%	
SG	Time: 433 seconds		10%
	Zero-crossing detection	18%	
	Extrema detection	22%	
	Space grouping	17%	
	Surface segmentation	43%	
AP	Time: 2851 seconds		68%
	Planar surface approximation	7%	
	Quadric surface approximation	41%	
	Approximation error computation	52%	
GR	Time: 37 seconds		1%
	Computation of graph	50%	
	Object inference	50%	
RC	Time: 147 seconds		4%
	Screener (Module 1)	1%	
	Graph Matcher (Module 2)	92%	
	Analyzer (Module 3)	7%	

CV : Time for Gaussian and LoG convolution
 SG : Time for feature detection and segmentation
 AP : Time for surface approximation
 GR : Time for computation of graph and object inference
 RC : Time for recognition

rarily. The screening results for the other two objects are as follows:

- The wagon: 1) plane, view 1, 2) hatchback, view 2, 3) wagon, view 1, 4) chair, view 2, 5) wagon, view 2.

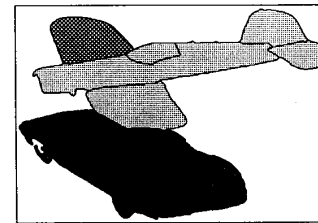


Fig. 14. Result of object inference for the first scene.

- The plane: 1) plane, view 1, 2) plane, view 2, 3) chair, view 1.

The second step examines each of the selected candidates in the order shown above. As the result of graph matching, the first four candidates for the wagon are discarded. In this case study, we show the search trees expanded by matching between the scene object "wagon" and the second view of the model "wagon," and between the scene object "airplane" and the first view of the model "plane," where both models are selected as the final match.

Fig. 15 shows the surface patches of the first view of the model plane, the second view of the model wagon, and the scene, respectively, with each surface patch (node) having a unique number. Figs. 16 and 17 show the search trees in matching the wagon and the airplane, respec-

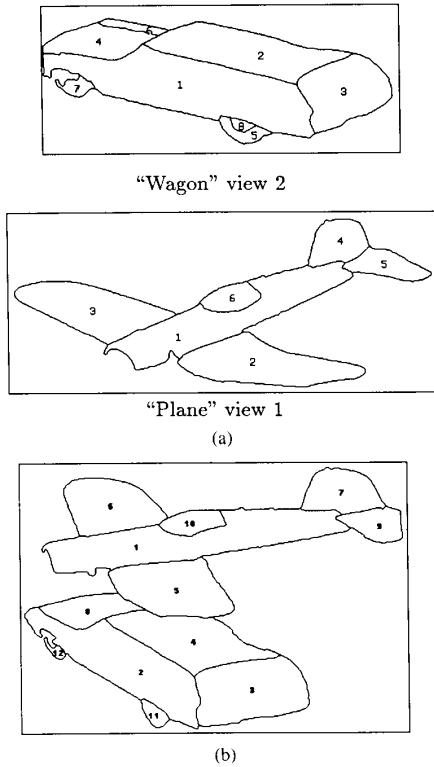


Fig. 15. Node numbers of the models and the first scene: (a) matched model views, (b) matched objects}.

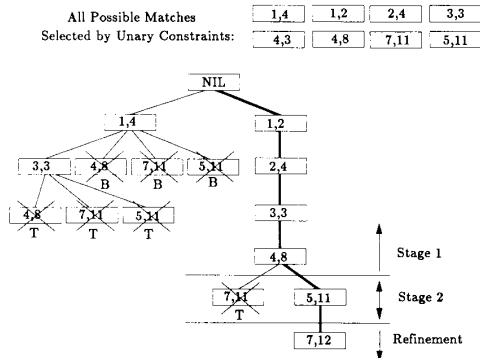


Fig. 16. Search tree for the wagon.

tively. In these figures, each tree node, represented by a pair of numbers inside a box a, b , indicates a possible match between model node a and scene node b . The order of expansion is from right, then top to bottom. In Fig. 16, for example, the order of expansion is $1, 4$, $3, 3$, $4, 8$ at level 2, $7, 11$ at level 2, $5, 11$ at level 2, $4, 8$ at level 3, $7, 11$ at level 3, $5, 11$ at level 3, $1, 2$, and so on. A cross over a tree node indicates that this match is rejected, either by binary constraints (marked "B") or by transform constraints (marked

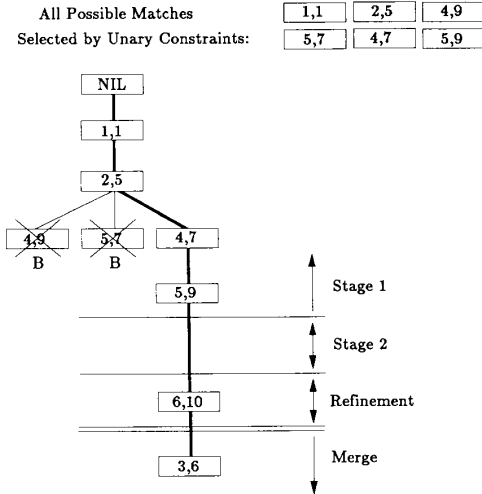


Fig. 17. Search tree for the airplane.

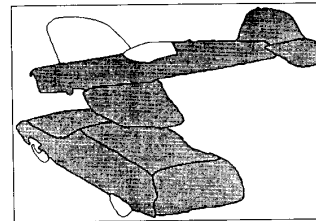


Fig. 18. Nodes expanded in stage 1.

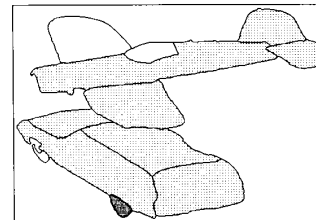


Fig. 19. Nodes expanded in stage 2.

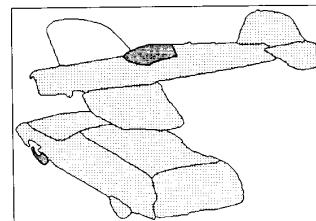


Fig. 20. Nodes expanded after refinement.

"T"). Note that the unary constraints are applied to select all the possible match pairs *before* the search begins. The final selected matches are indicated by thicker links. Figs. 18, 19, and 20 show the scene nodes expanded in

each stage where heavily shaded regions indicate currently expanded patches (nodes) and lightly shaded regions indicate expanded patches at previous stages. Note that we show both scene objects (wagon and airplane) in the same figures, but in fact they are processed in sequence. From these figures we see that the major (larger) surfaces are always selected in the first stage while smaller surfaces are matched in later stages.

After the graph search, the correspondences of the wagon are correctly established while the right wing of the airplane has not been recognized. In the next step, merging is applied to the scenes. Since the wing (scene node 6) is closer to the airplane than to the wagon, and there is only one unmatched node in the matched airplane model (model node 3), the match $[3, 6]$ is thus selected as a possible match. The transform and binary constraints are then checked, and in this case, both of them are satisfied, thus the wing is merged to the airplane and the match $[3, 6]$ is included. It should be noted that to verify the transform constraint, we simply transform the model views of the airplane and *superimpose* it on the scene, which is shown in Fig. 21, and then find that the two nodes $[3, 6]$ are close enough. Fig. 22 shows the expanded node in the merging step.

Fig. 23 shows the recognition results where Fig. 23(a) shows the matched model views and Fig. 23(b) shows the matched scene objects. In this figure, corresponding textures are used to represent corresponding objects, and corresponding numbers are used for matched nodes. Table II summarizes the recognition results. The entries are to be interpreted as follows.

- *Object*: The object in the scene.
 - *Model*: The selected model view by the screener.
- The views are sorted by order of selection.
- *Nodes Expanded*: The number of *search nodes* expanded in the search tree. It is limited to 100.
 - *Max. Depth*: The maximum depth of the search tree.
 - *Max. Width*: The maximum width of the search tree, it is limited to 5.
 - *Decision*: The final decision of the recognition:
 - 1) *Matched*: The object and the model view are considered matched (good enough in Module 2 (Graph matcher), i.e., match value $\mathcal{I}C \geq 0.8$). If this happens, the remaining selected model views are ignored.
 - 2) *Ignored*: The view is ignored because a good-enough match has already been found.
 - 3) *Rejected*: The match is too poor to be accepted ($\mathcal{I}C < 0.6$).
 - 4) *Plausible*: The match between the object and the view is acceptable; however, it is not good enough to ignore remaining candidate views ($0.6 \leq \mathcal{I}C < 0.8$). In this case, the match and its match value $\mathcal{I}C$ are kept for further comparison. If a good-enough match is found during the subsequent exploration, this plausible match is discarded, otherwise, the plausible match with the largest $\mathcal{I}C$ is selected as the final match.

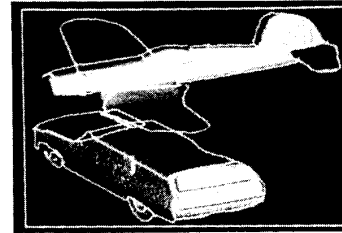


Fig. 21. Transform results of the first scene.

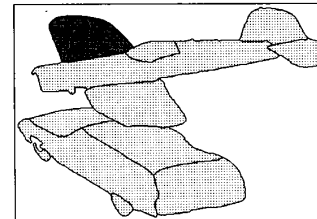
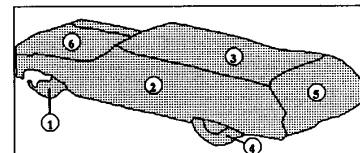
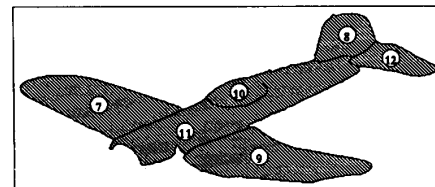


Fig. 22. Nodes expanded after merging.

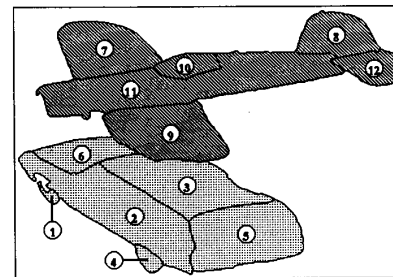


"Wagon" view 2



"Plane" view 1

(a)



(b)

Fig. 23. Result of recognition on the first scene: (a) matched model views, (b) matched objects}.

C. Results for Other Scenes

We have performed successful experiments on several such composite scenes. The complete set can be found in [13]; here, we only present results for two other scenes in order to save space. They are shown in Figs. 24 and 25,

TABLE II
SUMMARY FOR THE FIRST SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The wagon	plane, view 1	8	2	3	Rejected
	hatchback, view 2	3	2	2	Rejected
	wagon, view 1	4	1	2	Rejected
	chair, view 2	5	2	2	Rejected
	wagon, view 2	15	6	4	Matched
The plane	plane, view 1	7	5	3	Matched
	plane, view 2	-	-	-	Ignored
	chair, view 1	-	-	-	Ignored

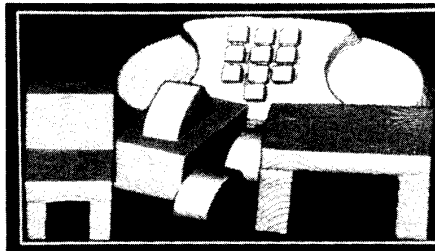


Fig. 24. The second scene.

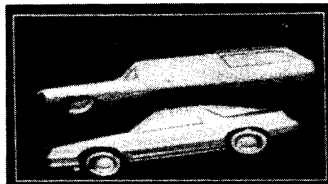


Fig. 25. The third scene.

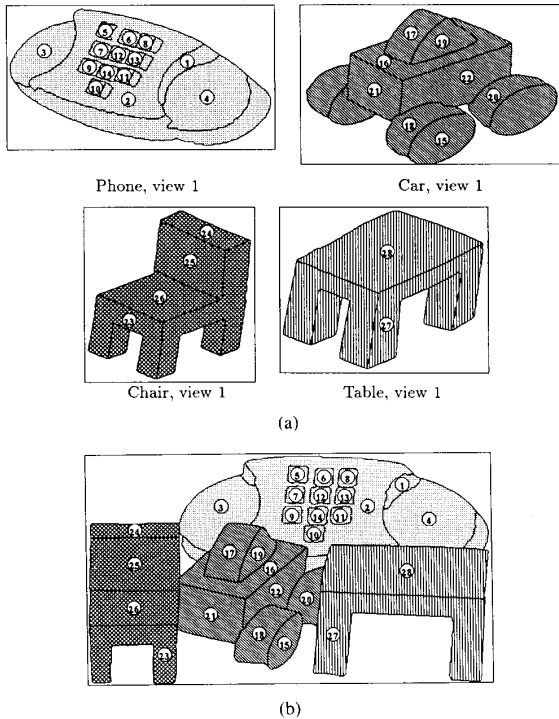


Fig. 26. Result of recognition on the second scene: (a) matched model views, (b) matched objects.

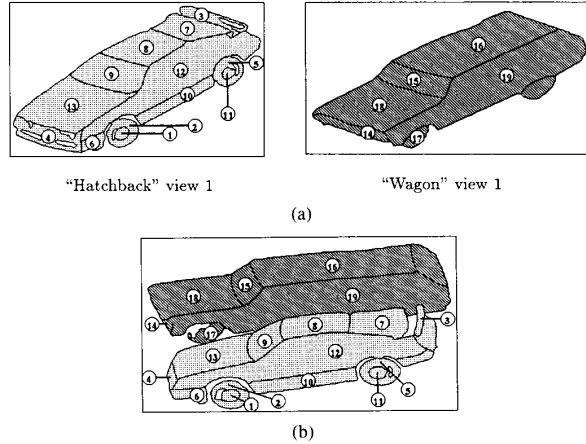


Fig. 27. Result of recognition on the third scene: (a) matched model views, (b) matched objects.

TABLE III
SUMMARY FOR THE SECOND SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The phone	phone, view 1	20	10	4	Matched
	phone, view 2	-	-	-	Ignored
	chair, view 3	-	-	-	Ignored
	hatchback, view 1	-	-	-	Ignored
The car	hatchback, view 2	5	2	2	Plausible
	plane, view 2	3	2	2	Rejected
	chair, view 1	17	3	3	Plausible
	car, view 1	18	4	5	Matched
	plane, view 1	-	-	-	Ignored
The chair	chair, view 1	7	4	3	Matched
The table	table, view 1	2	2	1	Matched
	mask, view 6	-	-	-	Ignored

TABLE IV
SUMMARY FOR THE THIRD SCENE

Object	Model	Expanded Nodes	Max. Depth	Max. Width	Decision
The hatchback	table, view 2	2	1	2	Rejected
	chair, view 3	14	3	5	Rejected
	hatchback, view 1	15	8	3	Matched
The wagon	wagon, view 1	7	4	2	Matched
	wagon, view 2	-	-	-	Ignored
	chair, view 2	-	-	-	Ignored
	hatchback, view 2	-	-	-	Ignored
	chair, view 1	-	-	-	Ignored

together with the recognition results shown in Figs. 26 and 27. Tables III and IV summarize the recognition process.

It should be noted that if a good-enough match cannot be found after graph matching, further analysis may be necessary to find the best match from the plausible matches. Several possibilities include increasing the resolution, focusing on one part, or changing viewing angles.

Several conclusions can be drawn from the following results.

- Most of the correct matches are found at the first or second selected views, this proves that our screening is powerful.

- Whenever a correct view is selected, the number of expanded nodes is small. This proves that our graph matching is efficient.

- The inferred objects do correspond to the physical objects, which indicates that our object inference and splitting/merging methods are powerful.

VI. CONCLUSION

This paper provides a complete method to describe and recognize 3-D objects, using the surface information of these objects. The key contributions are summarized as follows.

- *It provides a complete system to describe and recognize 3-D objects.* We present a set of techniques to retrieve significant features of 3-D objects, describe them, match the descriptions, build the models in multi-views, and finally recognize them.

- *It is data-driven in that no a priori scene knowledge is required.* The descriptions of the objects are computed without any knowledge about existing models, which is important when the environment is unknown, or the number of interesting objects is large.

- *Moderately complex objects can be well described and matched.* A large number of different objects were tested and good results were obtained. These objects vary in sizes, shape, and complexity. Some of them are highly symmetric while others are different from every view point. This shows the generality and robustness of our method.

- *Partially occluded objects can be well described and matched.* Occlusion is one of the major problems in computer vision. The method presented here tackles this problem in a very effective way. More than two thirds of the surfaces can be heavily occluded or completely missing and a plausible match is still achievable.

Our system can be improved in several ways:

- Better segmentation methods will help. Specifically, additional methods for segmenting "smooth" surfaces are needed.

- Under heavy occlusion, it may be necessary to further group fragmented surfaces before matching is attempted.

- It would be nice to have a single, 3-D volumetric representation of the models, rather than the multiple views. The multiple views could then be computed from the single model. Alternately, volume descriptions may be computed from the data. The latter is a difficult problem, but we believe that our surface descriptions are a key step for this process also.

REFERENCES

- [1] N. Ayache, "A model-based vision system to identify and locate partially visible industrial parts," in *Proc. Conf. IEEE Computer Vision and Pattern Recognition*, Washington, DC, 1983, pp. 492-494.
- [2] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Comput. Surveys*, vol. 17, no. 1, pp. 75-145, Mar. 1985.
- [3] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 2, pp. 167-192, Mar. 1988.
- [4] B. Bhanu, "Representation and shape matching of 3-D objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 3, pp. 340-350, May 1984.
- [5] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local feature-focus method," *Int. J. Robotics Res.*, vol. 1, no. 3, pp. 637-643, 1982.
- [6] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 3-26, Fall 1986.
- [7] M. Brady, "Computational approaches to image understanding," *ACM Comput. Surveys*, vol. 14, no. 1, pp. 3-71, Mar. 1982.
- [8] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing surfaces," In H. Hanafusa and H. Inoue, editors, *Proc. 2nd Int. Symp. Robotics Research*, H. Hanafusa and H. Inoue, Eds. Cambridge, MA: M.I.T. Press, 1985.
- [9] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intell.*, vol. 17, pp. 285-348, 1981.
- [10] —, "Model-based three-dimensional interpretations of two-dimensional images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 2, pp. 140-150, Mar. 1983.
- [11] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surveys*, vol. 18, no. 1, pp. 67-108, Mar. 1986.
- [12] T. J. Fan, G. Medioni, and R. Nevatia, "Segmented descriptions of 3-D surfaces," *IEEE J. Robotics Automation*, pp. 527-538, Dec. 1987.
- [13] T. J. Fan, "Describing and recognizing 3-D objects using surface properties," Ph.D. dissertation, Dep. Comput. Sci., Univ. Southern California, Los Angeles, Aug. 1988.
- [14] O. D. Faugeras and M. Hebert, "The representation recognition and locating of 3-D objects," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 27-52, Fall 1986.
- [15] W. E. L. Grimson and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Res.*, vol. 3, no. 3, pp. 3-35, Fall 1984.
- [16] —, "Localizing overlapping parts by searching the interpretation tree," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 4, July 1987.
- [17] K. T. Gunnarsson, "Optimal part localization by data base matching with sparse data and dense data," Ph.D. dissertation, Dep. Mech. Eng., Carnegie-Mellon Univ., Pittsburgh, PA, Apr. 27, 1987.
- [18] P. Horaud and R. C. Bolles, "3DPO's strategy for matching three-dimensional objects in range data," in *Proc. Int. Conf. Robotics*, Atlanta, GA, Mar. 13-15, 1984, pp. 78-85.
- [19] B. K. P. Horn, "Extended Gaussian images," *Proc. IEEE*, vol. 72, pp. 1656-1678, Dec. 1984.
- [20] B. K. P. Horn and K. Ikeuchi, "The mechanical manipulation of randomly oriented parts," *Science America*, vol. 251, no. 2, pp. 100-111, Aug. 1984.
- [21] K. Ikeuchi, "Recognition of 3-D objects using the extended Gaussian image," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, Vancouver, B.C., Canada, Aug. 24-28, 1981, pp. 595-600.
- [22] —, "Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks," in *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 321-339.
- [23] J. L. Jezouin, P. Saint-Marc, and G. Medioni, "Building an accurate range finder with off the shelf components," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 5-9, 1988.
- [24] R. Nevatia and T. O. Binford, "Description and recognition of complex-curved objects," *Artificial Intell.*, vol. 8, pp. 77-98, 1977.
- [25] M. Oshima and Y. Shirai, "A scene description method using three-dimensional information," *Pattern Recognition*, vol. 11, pp. 9-17, 1979.
- [26] —, "Object recognition using three-dimensional information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 3, no. 4, pp. 353-361, July 1983.
- [27] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: M.I.T. Press, 1984.

- [28] K. Rao and R. Nevatia, "Generalized cone descriptions from sparse 3-D data," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami Beach, FL, June 22-26, 1986, pp. 256-263.



Ting-Jun Fan (M'88) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, in 1980, the M.S. degree in computer engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1982, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1986 and 1988, respectively.

He is currently a Research Staff Member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His research interests include computer graphics, user interface design, computer animation, and computer vision.

Dr. Fan is a member of the Association for Computing Machinery.



Gerard Medioni (S'82-M'83) received the Diplome d'Ingenieur Civil from the Ecole Nationale Supérieure des Telecommunications, Paris, France, in 1977, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in 1980 and 1983, respectively.

He is currently an Assistant Professor in the Departments of Electrical Engineering and Computer Science, University of Southern California. His research interests include computer vision, artificial intelligence, and robotics.

Dr. Medioni is a member of the Association for Computing Machinery and the American Association for the Advancement of Science.

Ramakant Nevatia (S'71-M'74-SM'86), for a photograph and biography, see this issue, p. 1139.