

Perceptual Grouping for Generic Recognition*

PARAG HAVALDAR, GÉRARD MEDIONI AND FRIDTJOF STEIN†

*Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles,
California 90089-0273*

havaldar@iris.usc.edu

medioni@iris.usc.edu

Received May 14, 1993; Revised May 24, 1994 and October 11, 1994

Abstract. We address the problem of recognition of generic objects from a single intensity image. This precludes the use of purely geometric methods which assume that models are geometrically and precisely designed. Instead, we propose to use descriptions in terms of features and their qualitative geometric relationships. To succeed, it is clear that these features need to be high level, rather than points or lines. We propose to detect groups using perceptual organization criteria such as proximity, symmetry, parallelism, and closure. The detection of these features is performed in an efficient way using proximity indexing. Since many groups are created, we also perform selection of relevant groups by organizing them into sets of similar perceptual content. Finally we present an implementation of a recognition system using these sets as primitives. It is an efficient colored graph matching algorithm using the adjacency matrix representation of a graph. Using indexing, we retrieve matching hypotheses, which are verified against each other with respect to topological constraints. Groups of consistent hypotheses represent detected model instances in a scene. The complete system is illustrated on real images. We also discuss further extensions.

1. Introduction

Most object recognition systems today address the problem of finding the location and orientation of an exactly known rigid object in a scene. The tools used to achieve this task are *geometric constraints*, and a lucid treatment of this class of approaches can be found in Grimson's book (Grimson, 1990). The presence of a model is inferred by the verification that such a model could indeed produce some of the observed data under an appropriate geometric transform. In this approach, low level primitives such as edgels or their approximations, as produced by state of the art edge finders can be used and an exact geometric model is required. Such an

approach is therefore very appropriate when evolving in a controlled environment, such as a factory, where the number of possible objects is small and their geometry is precisely known.

However, this approach cannot be extended to more general scenarii because objects may be very similar while being geometrically different. Consider for instance two different airplanes which have similar features but different geometries. In other words, generic recognition should not make use of methods based purely on the exact geometric structure of the object. It is clear that the only way to solve this difficult problem is to reason about parts and their arrangements. This argument is supported by Biederman (1987). According to his RBC theory, a limited number of volumetric components (or *geons*) are sufficient for the task of recognition. Recovery of parts and their arrangements can help quick recognition of objects even if the objects are occluded, novel, rotated in depth or extensively degraded. We therefore have three problems to solve: the

*This research was supported by the Advanced Research Projects Agency of the Department of Defence and was monitored by the Air Force office of Scientific Research under Contract No. F49620-90-C-0078 and by a NSF Grant under award No. IRI-9024369.

†Fridtjof Stein is currently with Daimler Benz, Germany.

extraction of primitives, the description of scenes in terms of these primitives, and the actual recognition of objects. In this paper, we propose the use of perceptual grouping to approach the problem of generic recognition.

Use of perceptual groups is not new, as it was proposed in the 1970s but was not very successful because of the failure to obtain reliable primitives in the first place. Using groups explicitly for recognition was done by the following authors: the study of object recognition of three dimensional objects was launched by the classic work of Roberts (1968). Brooks (1983) developed an image understanding system called ACRONYM which uses a restricted class of generalized cylinders (GC) for descriptions of model and scene objects. Lowe's SCERPO (Lowe, 1987) system takes a bottom-up approach to object-centered recognition. Rothwell (1993) explains the need for computing local invariants and for tying them together to form complete object descriptors as opposed to computing a single global descriptor. Clemens and Jacobs (1991) claim that indexing provides a significant speedup only when accompanied by a grouping process.

We propose to use perceptual grouping laws to extract *groups* from initial primitives, organize them into sets which have similar "perceptual" content and use the sets for recognition. As explained in some of our previous work (Stein et al., 1993), such groups serve as an intermediate level representation of the data, in a hierarchical fashion, and can be used to retrieve likely candidate objects from a library. The models can be extracted from multiple views of an object, rather than from a complete model (as such they are "quasi-invariants"). This idea of "perceptual grouping" can be found in the psychology literature of the 1920s, under the *Gestalt* name. As an explanation of how the perception of individual objects are formed, the Gestalt theory proposed an organization of parts of an image into wholes, based on laws of grouping. Elements in the image are grouped based on proximity, similarity, closure, symmetry, and continuation. These groups themselves can be used as tokens and grouped with the same laws. Unfortunately, the implementation of such observations is difficult, as these laws conflict, even for simple stimuli. In computer vision these ideas can be found in Lowe (1987) and Witkin and Tanenbaum (1983). The issues we have to tackle in order to generate groups relate to the choice of such groups, their discriminative power, their robustness to viewpoint and noise, and their efficient computation. Here, we address

these issues and develop a feature hierarchy which can be used for object recognition of three-dimensional objects from a two-dimensional scene. In this hierarchy, we propose specific groups based on proximity, parallelism, symmetry, and closure. The detection of these features is performed in an efficient way using *proximity indexing*. First, we generate features with multiple representations to overcome the unreliability of local algorithms during preprocessing, and to handle noise and capture different levels of detail. Later, we merge perceptually similar features at higher levels of the feature hierarchy.

The groups, as extracted from real images, not only contain perceptually salient groups but also accidental groups which do not yield any natural descriptions. These undesired groups increase the complexity of the recognition process and provide no useful information. Therefore, we need to perform some selection process at this stage. We propose a way to select the "good groups" and, if possible, discard "bad groups". We organize the groups detected into sets such that each set is perceptually similar and make use of junctions to select the perceptually valid sets.

For recognition, we follow a graph-based matching approach. We use a set of non-registered views of a 3-D object as models. We use sets of perceptually similar groups to model the object and perform recognition. As such, every valid group that the object gives rise to has *multiple representations*. While most other systems use spatial correspondences to verify matching hypotheses, we use high level features and their *topological* relationships for the recognition process. These topological relationships can be represented by a graph. The graph vertices represent sets of features. The edges represent topological attributes between the features, such as inclusion and adjacency. We use the paths in this graph as basic token for the process of structural indexing. Groups of consistent hypotheses represent detected model instances in a scene.

The paper is organized as follows. Section 2 describes the feature hierarchy, discussing the detection algorithms which we developed to derive the different features from the intensity edges of an image. We then explain the need for the selection of good features and how they are selected in Section 3. Finally in Section 4, we talk about representation issues, give an outline of a object recognition system, discuss the matching, and show some results in Section 5. Some topics, such as large data bases, are beyond the scope of

this presentation and will be hopefully addressed in future research. Finally, we discuss the computational efficiency of our perceptual grouping algorithms.

2. Going from Edgels to Groups

In our previous work (Stein and Medioni, 1992a), the super segment was introduced in two or three dimensions as a feature which represents a part of a curve. It is based on the assumption that the underlying structure embodies continuity. Therefore the edge primitives are grouped with respect to their co-curvilinearity into a curve which is then approximated by line segments. Consecutive line segments are grouped into super segments. Here we face a more difficult problem. The above cited co-curvilinearity is no longer a reliable criterion for a stable grouping strategy. The solution which we pursue in this paper is to go to higher level groups which take into account other grouping criteria besides co-curvilinearity, such as parallelism, closure, and symmetry. We now explain the steps involved in going from an image to a high level representation of it in terms of “perceptual” groups. This chain of processing is sketched in Fig. 1. We start with an image. In the *preprocessing* stage, we reduce the amount of data: starting from images we compute curves, which consist of linked edgels. In the *local grouping* stage, we generate many line segments based on multiple linear approximations with different fitting tolerances. For each approximation tolerance, we perform a vertex collapse and compute super segments and parallels with exhaustive grouping. By creating a large set of features at this point we gain robustness in our further groups, and we significantly reduce the unreliability of the preprocessing. The *perceptual grouping* stage no longer distinguishes between features of different fitting tolerances. The reduction of data is based on two strategies:

1. merging perceptually similar features from the local grouping stage, and
2. grouping features into higher level features using geometric relationships such as symmetry, closure, and proximity.

The applied rules for the grouping process are not guaranteed to be mutually exclusive. In Fig. 2 an example illustrates some of the conflicting interpretations which are possible with the underlying data. Without any further organization of the detected groups, we

cannot decide the correct and incorrect ones. Therefore we further organize the groups into sets of groups such that each set contains groups which are similar in perceptual content. We then use these sets as *high level groups* to build a topological graph. The graph edges are defined by topological relationships such as adjacency or inclusion. The following sections focus on the details of the perceptual hierarchy.

We start with the elementary primitives, the edgel curves, and discuss the implementation of the different grouping rules:

- the co-curvilinearity criterion produces *super segments*,
- the parallelism criterion produces *parallels*, which consist of two parallel line segments,
- the symmetry criterion produces either *parallel symmetries* or *skewed symmetries* which are computed from parallels or super segments respectively,
- and the closure criterion produces *U-Shapes*, *closed curves*, and *skewed symmetries*.

Finally, we discuss the computational efficiency of our perceptual grouping algorithms. In computer vision, many authors have focused on computing perceptual groups (see e.g., (Kanade, 1981; Lowe, 1987; Kriegman and Ponce, 1990; Rom and Medioni, 1993; Saint-Marc and Medioni, 1990; Ulupinar and Nevatia, 1993)). Most of these algorithms tackle the detection of *all* perceptual groups by either assuming perfect data; or by applying exhaustive search. Our algorithms try to compromise: we do not assume perfect data and therefore we find most (but not necessarily all) perceptual groups. But on the other hand we do this in an efficient way by using *proximity indexing*.

2.1. Preprocessing

We first detect edges in the image (we use the Canny edge detector (1986)). The resulting edgels are further linked into *curves*. Curves are then approximated with a line fitting algorithm to compute the polygonal approximations. Instead of just using one representation we approximate each curve with a set of line fitting tolerances to get a robust representation. For every approximation, we then merge vertices, so that vertices which are close together (typically three to five pixels) are merged into one vertex. In Fig. 3 show an example scene and the detected edges. Note that although this is

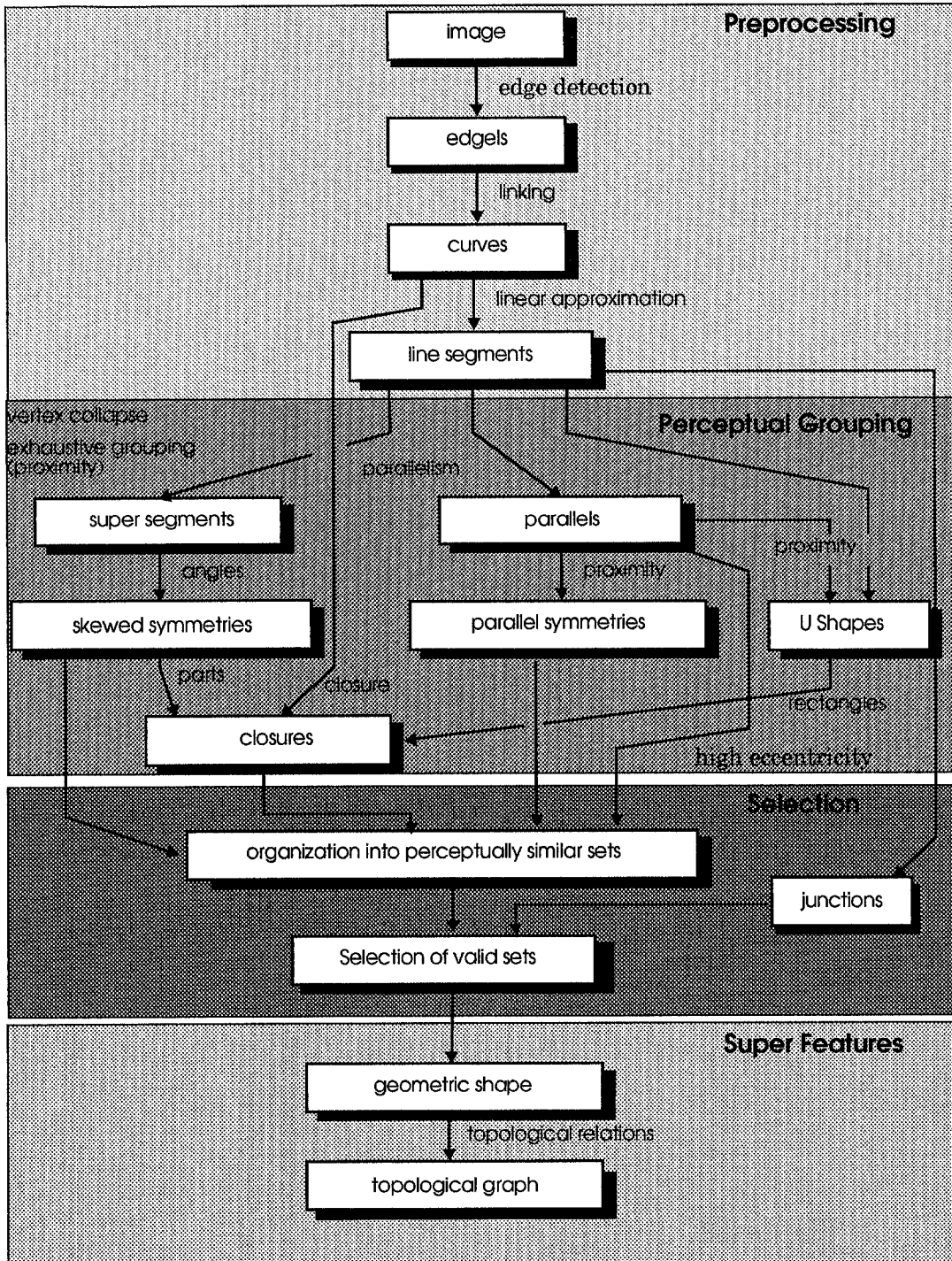


Figure 1. Hierarchy.

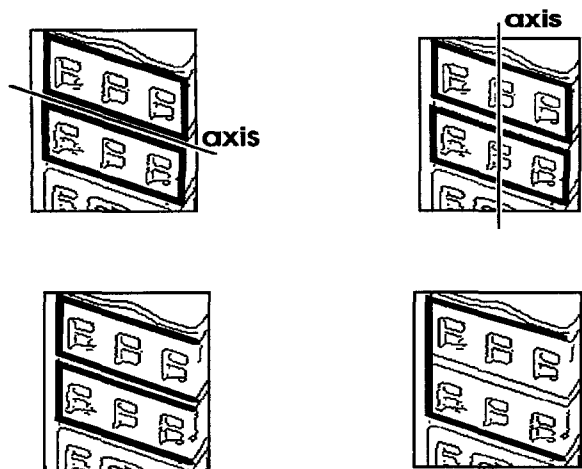


Figure 2. Multiple interpretations.

the same image as the one used in Zerroug and Nevatia (1993), we follow a very different line of reasoning. In particular, we make no assumptions about the kind of objects that we deal with.

2.2. Super Segments

2.2.1. Idea. Since we want to handle occlusion, we do not expect to obtain complete boundaries in our images, but only portions of them. On the other hand, individual segments are too local to be useful as matching primitives. Grouping a fixed number of adjacent segments provides us with one of our basic features, the super segments.

2.2.2. Implementation. The computation of super segments is the same as described in Stein and Medioni (1992a). Connected linear segments form chains of adjacent segments. The segment chains provide the super segments by grouping a fixed number of adjacent segments. (Because of the branching of the polygonal approximations we generate super segments exhaustively by generating them from all possible segment combinations.) We generate super segments of cardinality three to six.

2.3. Parallels

2.3.1. Idea. A *parallel* consists of two linear segments s_1 and s_2 . Both line segments have approximately the same orientation. We require that the two segments overlap, which means that the normal projection of s_1 on s_2 , or the normal projection of s_2 on s_1 not be empty. Furthermore we do not want the aspect ratio of the parallel to reflect elongation, with $ar(p) = (\text{length}(s_1) + \text{length}(s_2)) / \text{distance}(s_1, s_2)$. We require $ar(p) > 0.5$.

2.3.2. Implementation. The acquisition of the parallels is performed in two steps by using proximity indexing.

1. All segments are recorded using the quantized orientation as the key. We use δ as quantization (typically $\delta = 20^\circ$). Using proximity indexing we are guaranteed to find parallels which are at most $\delta/2$ apart and we get some parallels with an enclosed angle between $\delta/2$ and δ .

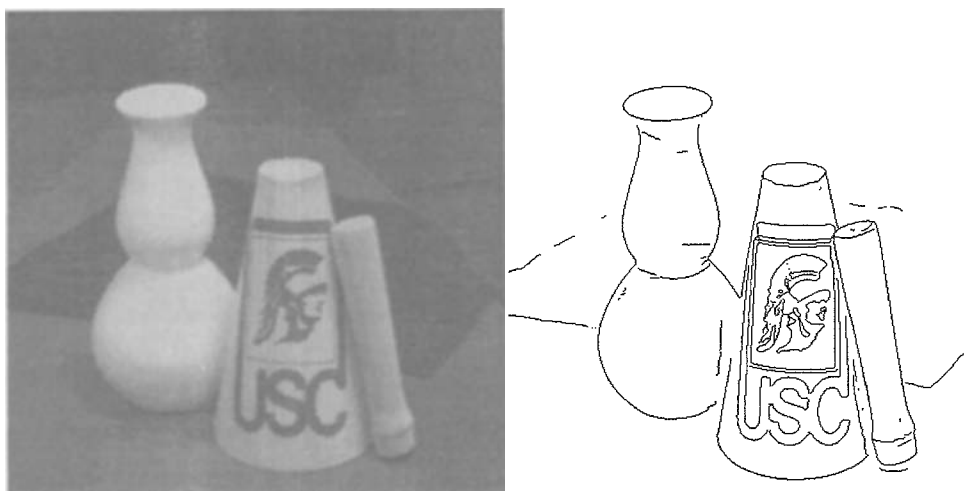


Figure 3. Example of image and detected edges (image—courtesy M. Zerroug).

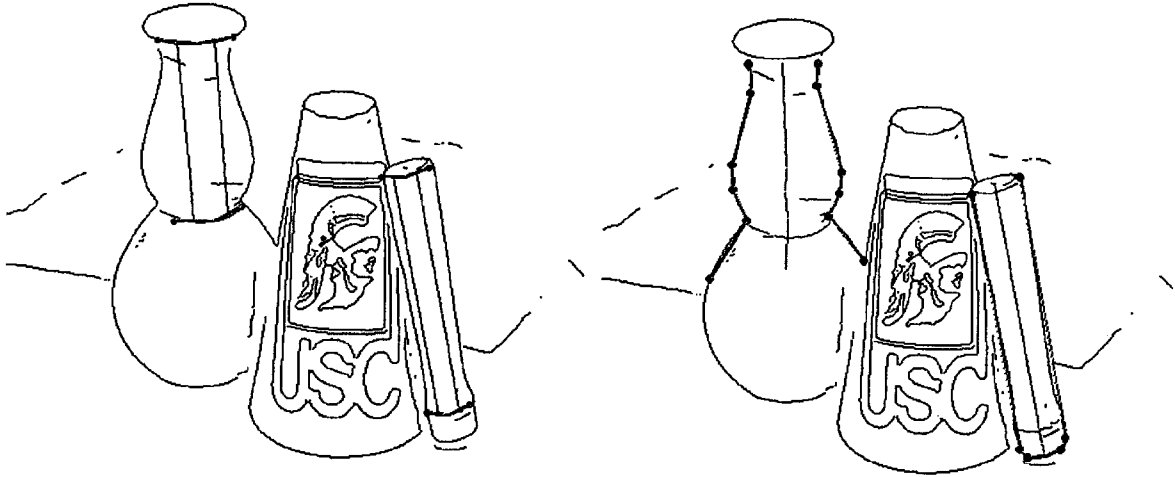


Figure 4. Examples of parallel symmetries (left) and skewed symmetries (right).

2. For every linear segment, the possible candidate parallels are retrieved and verified with respect to aspect ratio and overlap. Segment pairs which meet these constraints generate parallels.

2.4. Symmetries

A symmetry is defined as a one-to-one mapping between the points of two curves, with the symmetry axis defined as the locus of the mid-point of the straight lines joining a point on one curve to its image in the other (Mohan and Nevatia, 1989). Ulupinar and Nevatia (1993) have proposed two specific symmetries, namely skewed and parallel symmetries. Whereas they use them to infer surface orientation, we consider them as a general feature of an object.

2.4.1. Parallel Symmetries.

2.4.1.1. Idea. Given two curves $X_i(s) = (x_i(s), y_i(s))$, for $i = 1, 2$, parametrized by arc length s , and $\theta_i(s) = \arctan((dy_i(s)/ds)/(dx_i(s)/ds))$. $X_1(s)$ and $X_2(s)$ are said to be parallel symmetric (Ulupinar and Nevatia, 1993) if there exists a point-wise correspondence $f(s)$ between them such that $\theta_1(s) = \theta_2(f(s))$ for all values of s for which X_1 and X_2 are defined, and $f(s)$ is a continuous monotonic function. We only consider the special case where $f(s)$ is linear.

2.4.1.2. Implementation. Parallel symmetries are retrieved by finding proximate parallels. We do not use the super segment approach, because we would depend

on the cardinality of the super segments. By using the parallels as the building blocks, we can use proximity indexing to find parallels which share the same vertices. The acquisition of the parallel symmetries is performed in two steps.

1. We record every parallel twice. One time with the sorted list of the vertex coordinates of one side as key, the other time with the sorted list of the vertex coordinates of the other side as key.
2. For every parallel, the possible neighbor parallels are retrieved. The groups of adjacent parallels generate parallel symmetries.

Examples are shown in Fig. 4 (left).

2.4.2. Skewed Symmetries.

2.4.2.1. Idea. In a skewed symmetry, the point-wise correspondence is such that the axis of the symmetry is straight, and the lines of symmetry from a constant angle (not necessarily orthogonal) with the axis of symmetry. Skew symmetry was first proposed by Kanade (1981) and used in the analysis of scenes of polyhedral objects. An example is given in Fig. 5 (left). The detection of skew symmetry for curves was done by Kriegman and Ponce (1990) and Saint-Marc and Medioni (1990), but these methods are quite sensitive to noise. In our system, we are interested in symmetries between line segments. Considering all possible symmetries between line segments as proposed in Herault et al. (1990) is expensive. Our approach is based on finding skew symmetries between super segments. An

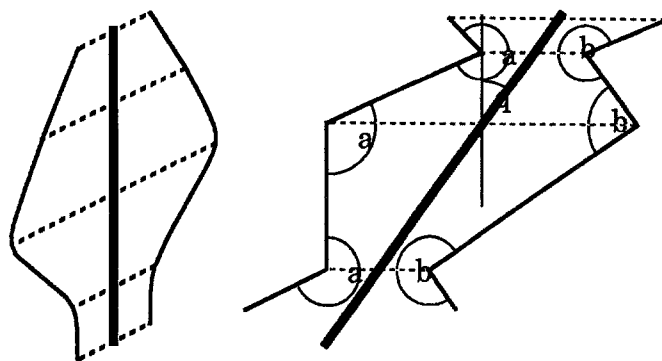


Figure 5. An example of skewed symmetry—with curved and straight lines.

example of skew symmetry for straight line segments is given in Fig. 5 (right). As can be shown (see Appendix of Stein (19...)), we have $\Delta = |\alpha^i - \beta^i| \leq 2\theta$, where θ is the skew angle. This allows us to define a local signature for a super segment, namely the angles, with which we can find possible symmetry candidates by indexing. This avoids the expensive comparison of all pairwise super segments. To test whether two super segments are skew symmetric, we have to test the following:

1. The difference between the corresponding angles must be smaller than $2\theta_{\max}$.
2. The symmetry axis has to be straight.
3. Using proximity indexing allows us to obtain an efficient algorithm.

2.4.2.2. Implementation. Skewed symmetries can be retrieved by finding pairs of super segments with angles of opposite sign. The detection of the skewed symmetries is performed in three steps:

1. We record every super segment using the list of the curvature angles as a key. To allow the guaranteed detection of symmetries with up to skew angle θ_{\max} , we choose as the quantization interval for the super segment angles $2\theta_{\max}$. For every super segment, we use the list of opposite sign curvature angles to retrieve possible symmetry candidates.
2. Super segment pairs which generate symmetry hypotheses have to be checked to see whether the corresponding symmetry axis is straight. We test this by requiring the scatter matrix of the middle points of all corresponding vertices to have a high eccentricity (they lie all along a line).

We show an example of detected symmetries in Fig. 4 (right).

2.5. Closures

Lowe (1987) states: *There is a tendency for curves to be completed so that they form enclosed regions.* Based on this statement, Mohan and Nevatia (1989) developed the idea to close symmetries at their ends to obtain so called ribbons, which form enclosed regions. They use these ribbons to segment images. We want to use closures as features, which do not necessarily have any physical interpretation in the image. At the moment we compute closures from U-Shapes, from closed curves and from skewed symmetries.

2.5.1. Closure from U-Shape

2.5.1.1. Idea. A parallel which is closed at one side by a linear segment is a strong indication that a rectangular structure is at hand where one side could not be detected. We therefore assume that we found a closed contour.

2.5.1.2. Implementation. U-Shapes can be found by indexing over the vertex pairs of parallels and trying to find a segment which forms a U-Shape with the parallel.

1. We record every parallel twice once using the quantized vertices of one “side” of the parallel and then other vertices as keys, where we record the parallel.
2. For every linear segment we use the list of the endpoints to retrieve possible U-Shape candidates.
3. If the angle between the parallel and the segment is $90^\circ \pm 30^\circ$ we generate a new U-Shape.

2.5.2. Closure from Curve. The obvious form of a closure occurs if we have a closed curve. To detect a closure based on a curve we allow the gap between start and end of the curve to be 5% of the arc length of the curve.

2.5.3. Closure from Skewed Symmetry.

2.5.3.1. Idea. We adopt the idea that a segmentation into parts should be done at negative minima of curvature from Rom and Medioni (1993). Such “a part” is used in our system as a closure.

2.5.3.2. Implementation. For every skewed symmetry we traverse the angles of one of its super segments (the other super segment just has the skewed mirror angles). Whenever we encounter a sign change of consecutive angles, we “break” the symmetry at this point and define the symmetry up to this vertex (together with the corresponding vertex of the other super segment) as one part. Applying this step iteratively, we generate alternating convex and concave parts. We use the convex parts to create closures. An example can be seen in Fig. 6.

2.6. Efficient Implementation through Proximity Indexing

Proximity indexing was used to efficiently compute the feature hierarchy described in the previous section. Proximity indexing issues play an important role when we wish to find features with similar attribute values. Traditional search methods which compare every possible pair are very time consuming. Recent vision systems have used indexing. A major problem with indexing is deciding the length of the quantization intervals. Values which are close may fall in different quantization intervals as shown in Fig. 7. Two features match only in the case when they fall into the same interval, which may not always be so. Flynn and Jain (1991) point out, it is essential to have an indexing scheme that preserves proximity in the key values. So far, two strategies based on indexing have been used to deal with this problem: large bucket size and searching of neighboring bins. While large bucket size is based on the hope that “less values will fall into the incorrect bin”, the search of neighboring bins has an exponential complexity with respect to the number of false value matches.

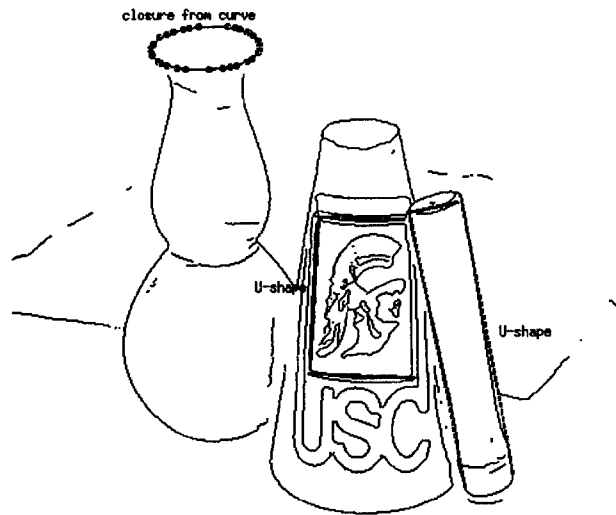


Figure 6. Examples of some closures detected in scene.

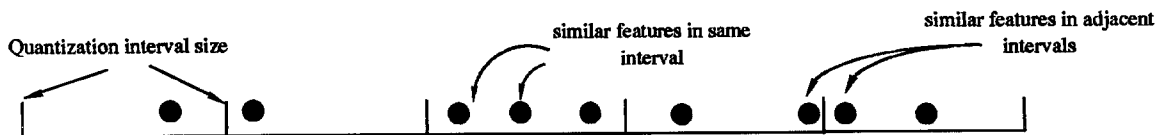


Figure 7. The indexing problem.

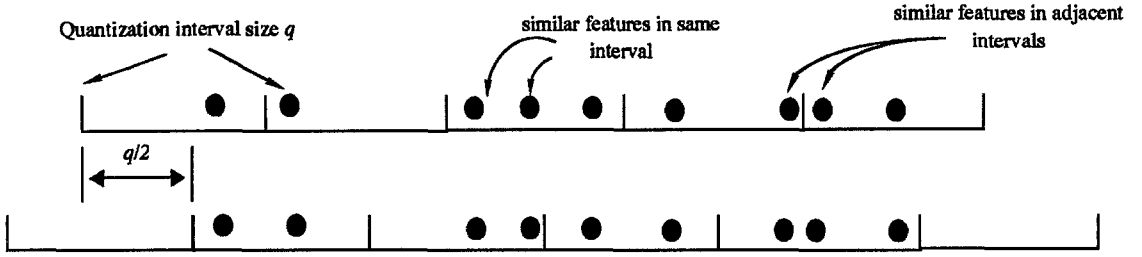


Figure 8. Every value is encoded twice during indexing.

We propose an alternative approach: Suppose we have a set of values which we want to store in buckets of length q . Each value v is assigned a key depending on which interval v falls into. We denote the key corresponding to v as the interval $(\lfloor v \rfloor_q, \lceil v \rceil_q)$. In order to preserve proximity, we encode every feature twice and use indexing on every value separately. Every value is quantized twice as shown in Fig. 8.

1. Under the key

$$(\lfloor v \rfloor_q, \lceil v \rceil_q)$$

2. Under the key

$$\begin{aligned} (\lfloor v \rfloor'_q, \lceil v \rceil'_q) & \quad \text{if } (v - \lfloor v \rfloor_q > \lceil v \rceil_q - v) \\ (\lfloor v \rfloor''_q, \lceil v \rceil''_q) & \quad \text{otherwise} \end{aligned}$$

where

$$\begin{aligned} \lfloor v \rfloor_q &= q \left\lfloor \frac{v}{q} \right\rfloor & \lceil v \rceil_q &= \lfloor v \rfloor_q + q \\ \lfloor v \rfloor'_q &= \lfloor v \rfloor_q + \frac{q}{2} & \lfloor v \rfloor''_q &= \lfloor v \rfloor_q - \frac{q}{2} \\ \lceil v \rceil'_q &= \lceil v \rceil_q + \frac{q}{2} & \lceil v \rceil''_q &= \lceil v \rceil_q - \frac{q}{2} \end{aligned}$$

The stored features for both intervals are retrieved and combined into one set. For all values we get such a feature set. The intersection of all these sets results in the features which are close to f . Such an interlaced quantization is guaranteed to preserve proximity while indexing. Due to the interlaced quantization we are guaranteed to retrieve all features with $|v_{\text{stored}} - v_f| < q/2$, and we get some feature matches with $q/2 < |v_{\text{stored}} - v_f| < q$. The intersection process can be sped up by intersecting the sets in increasing cardinality.

The space complexity of proximity indexing, when compared with traditional indexing, is the same, viz., $O(n)$, where n is the number of features. Looking at

the time complexity, we have to distinguish between the encoding step (building the index table) and the retrieval step (retrieving corresponding features from the table). We assume every feature has as a key a vector k with values v_i with $i = \{1, 2, \dots, p\}$, p is the number of attribute values per feature. Every value is quantized in Q intervals. Furthermore we assume that the features are equally distributed over the table which consists of r buckets. Every bucket has a key and a list of h entries. Therefore, for traditional indexing there are $h = n/r$ entries per bucket, and for proximity indexing $h = 2pn/r$ entries per bucket.

1. The time complexity for encoding with traditional indexing is $O(n)$, whereas the complexity for encoding in proximity indexing depends on sorted or unsorted encoding. As it will become clear in the discussion of the retrieval step, sorted encoding results in a faster retrieval. By sorted encoding we talk about sorted entries in each record. The sorting can be performed by using an arbitrary but stable sorting function (e.g., the creation date of the feature). Using a binary tree for the storage of the entries, the complexity of encoding n features is $O(n \log(n/r))$.
2. The retrieval step requires $O(n)$ for traditional indexing. The retrieval of features for proximity indexing requires the intersection of sets of features. Intersection of two sets of cardinality c has a complexity of $O(c)$ when the sets are sorted. The cost of intersection of two unsorted sets is $O(c \log(c))$ for each retrieval. Therefore, when the features are encoded in a sorted way, the retrieval step has a cost of $O(n^2/r)$, otherwise, in the unsorted case, the complexity is $O(n/r \log(n/r))$ for each retrieval or $O(n^2/r \log(n/r))$ for n retrievals.

The complexities for proximity indexing and that of traditional indexing are listed in the Table 1.

Table 1. Complexities of search and indexing.

	Search	Indexing	Proximity indexing
Preserves proximity	Yes	No	Yes
Encoding complexity	0	$O(n)$	$O(n)$ if entries are unsorted $O(n \log(n/r))$ if entries are sorted
Retrieval complexity	$O(n^2)$	$O(n)$	$O(n^2/r \log(n/r))$ if entries are unsorted $O(n^2/r)$ if entries are sorted

3. Selection of Relevant Groups

The groups extracted from images not only contain perceptually salient features, but also contain many features which do not yield any natural descriptions. Such groups come about when the segments and supersegments give rise to features (symmetries, U-Shapes, etc.) which are geometrically correct, but are less obvious because of other competing groups. Additionally, since each curve (approximated by line segments) is used to exhaustively compute all possible supersegments of various cardinalities, more irrelevant groups are formed by their interaction. Such an exhaustive computation is essential to account for the case when the curve is not present or is broken into smaller curves because of occlusion and noise. The undesired groups besides not having any relevant physical interpretations, also increase the complexity of the representation and matching process (see Section 4).

While the selection process may be helped by purely local heuristics, such as the skewness or orientation of overlapping groups, we prefer to make use of more global constraints. We first aggregate the groups into different sets such that each set contains groups which are perceptually similar. Perceptually similar groups have similar properties, e.g., similar direction of axis, similar segments forming the group, etc. Note that these sets may by themselves be correct or incorrect. Next we reason at the level of the above sets. To decide on the validity of a set, we make use of junctions. The main advantage of this is that if a certain set of features can be verified as not coming from any surface of an object in the image, then the entire set may be discarded. Reasoning at the surface patch level provides a stronger grip on the entire selection process. We now explain the aggregation and selection processes.

3.1. Aggregation of Groups into Sets

We use the properties of the axis to organize the groups into various sets, the objective being that each

set should perceptually provide the same information. Groups which are “similar” have similar axes orientation and their axes are spatially located close together.

1. We first aggregate the groups having similar axes orientations into different sets. We use a quantization angle of 20 degrees to partition the groups. The groups in each set may still be spatially located at different positions.
2. Next we choose each set and further partition it depending upon spatial neighborhoods. This is done by constructing a graph whose nodes represent groups and edges represent spatial closeness. The connected components of the graph give sets of groups having similar orientation and position.
3. Lastly we look at the segments of the groups in each set and further separate out groups which vary markedly in their average distance to the axis.

In Fig. 9 we show examples of four such group sets. Each image displays one group in each set. These sets as such are hypothesized as surfaces. The image shows that Sets 1, 2 and 3 are valid where as Set 4 is invalid. Below we discuss how to select valid sets.

3.1.1. Selection of Sets. Observations show that surfaces of objects in 3D would ideally project to areas bounded by contours in the 2D image. When surfaces are adjacent to each other or occlude one another, they give rise to junctions. Hence the valid sets computed above should be adjacent to other valid sets at the junctions (if present). The invalid sets, in most cases, form junctions in between two other valid sets (though not necessary). The junctions in the image may be because of a variety of reasons, mainly due to occlusion, surface markings or surface-orientation discontinuities on the object. We are not trying to classify the junctions, but rather use them as a tool for verification of the sets.

1. We first compute all the possible junctions formed by the curves in the image. Since the curves in

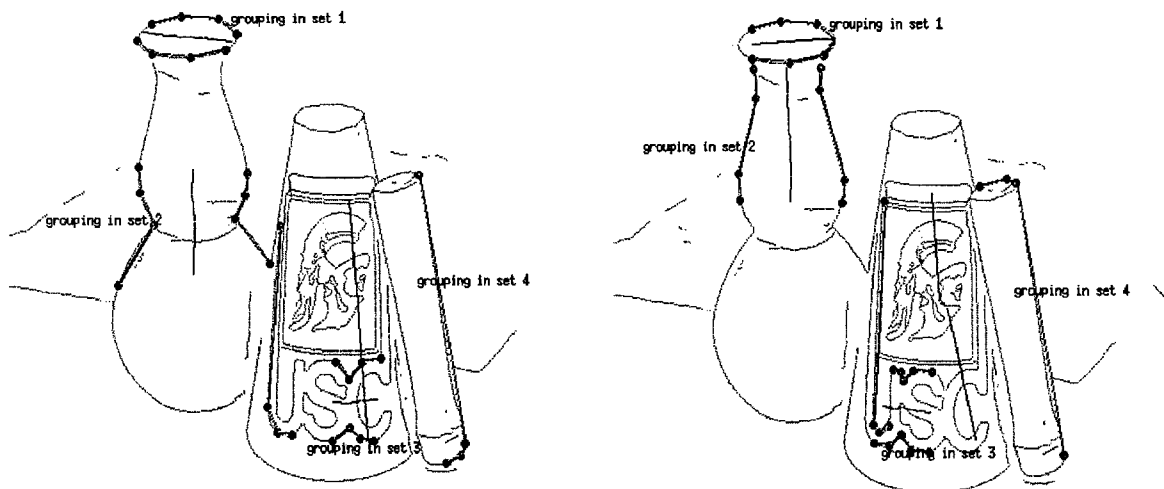


Figure 9. Examples of groups organized into sets of similar perceptual content. Four sets are shown. Each image shows one group of each set, e.g., the two groups of the vase in image fall in one set.

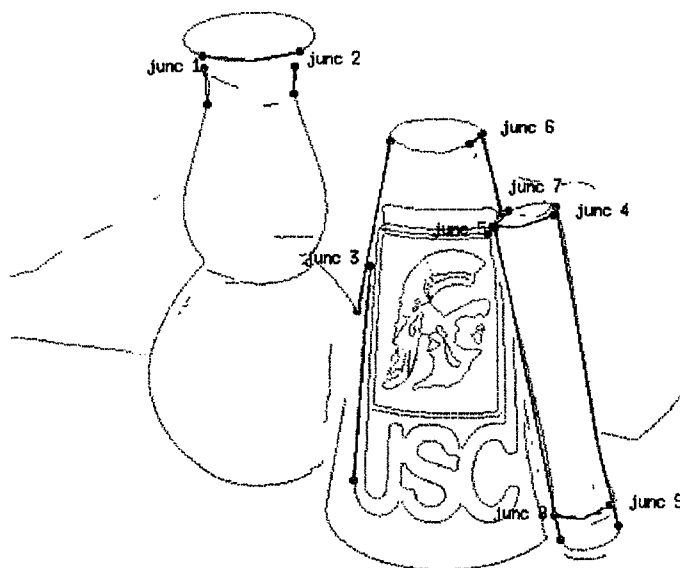


Figure 10. Examples of some junctions detected in image.

the image are approximated by segments, junctions are easily computed by finding segments which are close by and/or intersect. Some computed junctions are shown in Fig. 10.

2. Next we break the segments of the groups in each set into two subsets, each subset containing segments on either side of the axes.
3. If there exist junctions which connect the above mentioned segment subsets, such that one junction connects the segments of one side of the set to another set and a second junction connects the

segments of the other side of the same set to a third set then we label this set as invalid. In Fig. 11, we show some of the valid and invalid groups in the set as a result of this reasoning for the junctions shown in Fig. 10. It can be seen from Fig. 9 that the groups in set 4 are invalid. Two junctions which the segments of this group set form and meet the above reasoning are junctions 3 and 4. Another example of an invalid set comes about because of junctions 6 and 7. On the other hand junctions 1 and 2 are formed between segments of group sets 1 and 2 shown in

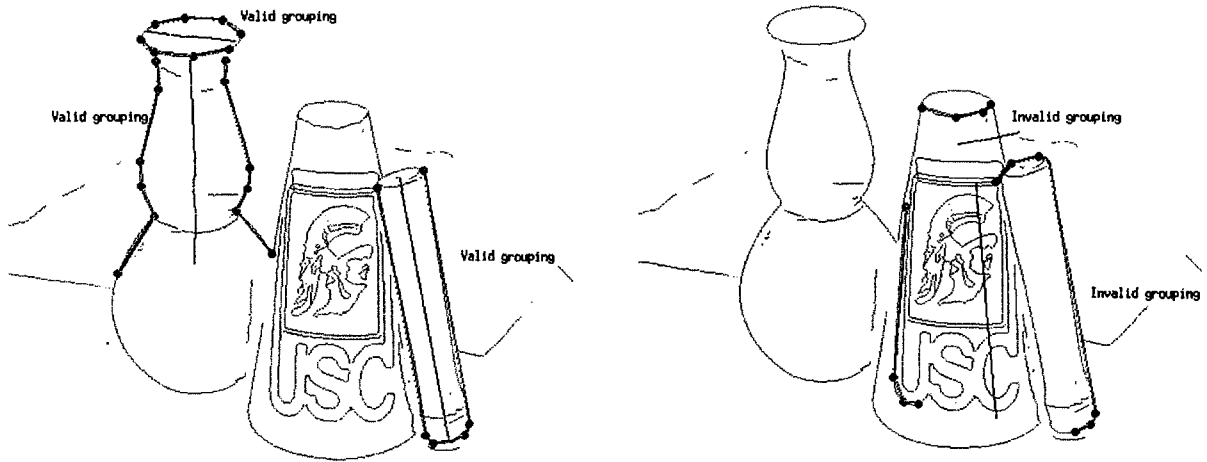


Figure 11. Examples of valid and invalid groups for T-junctions shown above.

Fig. 9. It can be seen that in this case the segments of the junctions are shared by two sets and not three.

At this stage we have the groups organized into sets, each set contains groups which are perceptually similar and can now be used as tokens for recognition. Each valid set may be thought of as coming from some surface or sections of the surface of the object, whose projection appears in the image. Note that each set as such has multiple representations in the form of many groups. We can encode the spatial relationships of these sets in the form of a graph to describe the object. In the following sections we explain how to get to the graph-based representation and how these sets serve the purpose of generic recognition. Note that although the percentage of the irrelevant groups has decreased, they may not necessarily be totally eliminated. However, recognition is achieved by hypotheses voting, which tends to eliminate the effect of irrelevant groups in the scene.

4. Representation and Matching

Given a set of features and the topological relationships between them, a natural representation of this structure is a graph. We first give some definitions:

- A **graph** G is defined by the pair (V, E) ; $V = \{v_0, v_1, \dots, v_n\}$ is the set of vertices, and $E = \{e_0, e_1, \dots, e_m\}$ with $e_i = (v_{i1}, v_{i2})$ is the set of edges.
- A **subgraph** of $G = (V, E)$ is defined as a graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$.

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a bijection $f: V_1 \rightarrow V_2$ such that $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$.
- A **path** P of length k in G is a sequence $\{v_0, v_1, \dots, v_k\}$ of vertices such that $(v_{i-1}, v_i) \in E$ for $i = 1, 2, \dots, k$.
- When the edges of a graph are labeled we talk about a **colored** or **labeled** graph.
- The **adjacency-matrix** representation of a graph G consists of a $|V| \times |V|$ matrix $A = (a_{ij})$ such that $a_{ij} = \text{label}(v_i, v_j)$.
- The adjacency-matrix of **cardinality** n is A^n with $A^0 = I, A^n = A^{n-1} * A$. It represents the set of paths of length n between any two vertices.

Two edge labels are multiplied by an order preserving multiplication. That means: $l_1 * l_2 = l_{12} \neq l_{21}$. The entry $a_{ij} = l_{12} = \text{label}(v_i, v_k) * \text{label}(v_k, v_j)$ represents a path from v_i to v_j through v_k with the labels l_1 and l_2 respectively.

The addition between two labels is not order preserving. The entry $a_{ij} = l_{12} + l_{34}$ represents the fact that there are two paths from v_i to v_j with either the labels l_1 and l_2 , or l_3 and l_4 .

For more on graph theory see (Alavi et al., 1985; Cornil and Gotlieb, 1970; Noltmeier, 1976). For our topological graph, the vertices are the shapes or enclosures of sets of groups, and the edges represent the topological relationship between them. We compute the enclosures or the bounding boxes of the set. This gives us the geometric shape of the set which enables

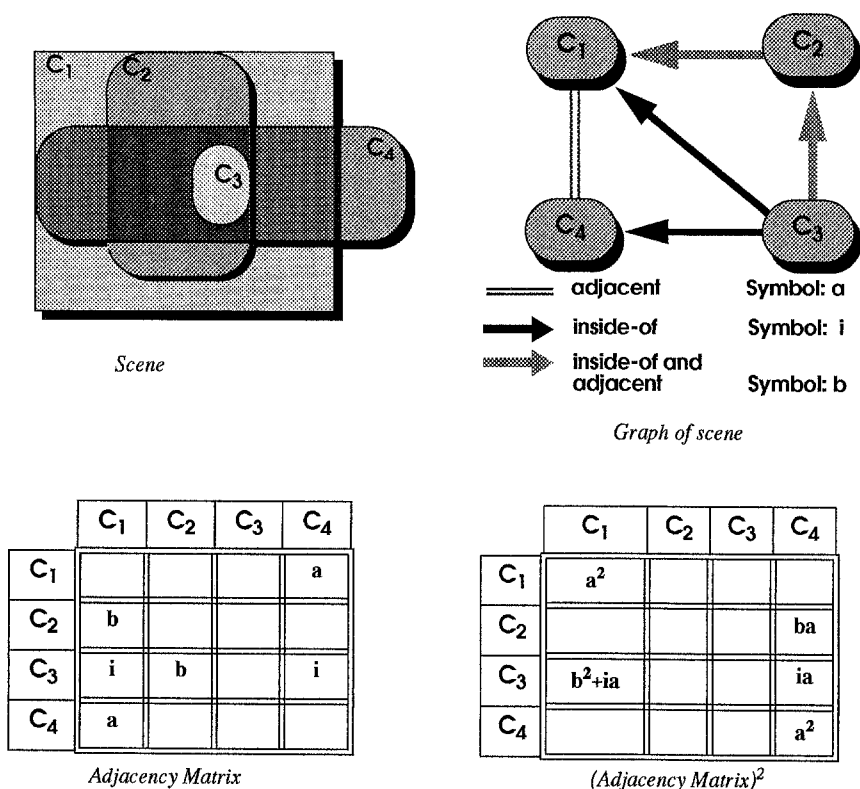


Figure 12. Example scene-1 and its representations.

easy computation of the topological and spatial relationships. In the current implementation we label the edges with only two labels: adjacency and inclusion. The direction of an edge depends on its label. When an edge is labeled with “inclusion”, the edge is directed from the inner set to the outer set. When an edge is labeled with “adjacency”, the edge is undirected. As an example see Fig. 12. We have four sets: C_1 , C_2 , C_3 , and C_4 . C_1 is adjacent to C_4 , C_3 is inside of C_1 , C_2 , and C_4 . In addition C_3 is adjacent to C_2 , C_2 is adjacent to and inside of C_1 . The corresponding graph can be schematically represented as in Fig. 12. Another example is shown in Fig. 13 with the corresponding graph. By representing the scene and the models with graphs, matching becomes a task of subgraph isomorphism. Which subgraphs of the scene are isomorphic to which subgraphs in the models?

In computer vision, graph matching is widely used (see, e.g., Fan, 1990; Lowe, 1987; Nevatia and Price 1992; Parvin and Medioni, 1991). In the worst case, the subgraph isomorphism problem is NP-complete.

Therefore several heuristics were developed to improve the average complexity for specific cases (see, e.g., Corneil and Jotlieb, 1970). Despite all the previous research, we could not find any algorithm which would perform with reasonable complexity for graphs consisting of a hundred vertices or more. Furthermore, we are unaware of theoretical results on subgraph isomorphism in colored graphs. Our goal is to find large subgraph isomorphisms, which are likely to represent detected models in a scene. We believe that we can use structural indexing to find corresponding subgraphs.

Using sets of groups as structural tokens for indexing be very expensive. In our implementation the sets of groups or their enclosures are not labeled and therefore they have little information content. The “color” of the graph mainly resides in the relationships between the sets, i.e., the edges in the graph. We make use of the information of the groups in the sets to separate out consistent hypotheses after the candidate hypotheses have been detected. The idea is to find all paths of length k (corresponding to $k + 1$ connected sets) and

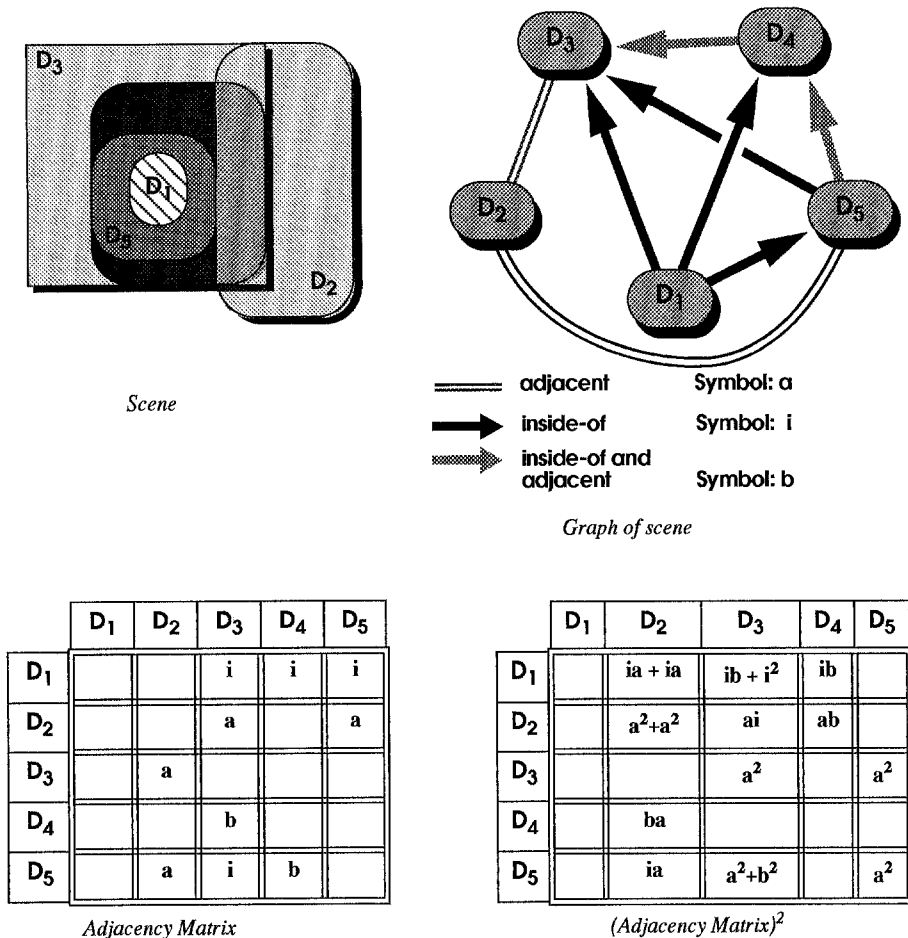


Figure 13. Example scene-2 and its representations.

to cluster these to find the largest consistent group of such paths. The number of such paths of length k in a graph which has $|V| = m$ vertices, and an average branching factor b at each node, is $mb(b - 1)^{(k-2)}$. This corresponds to an upper bound of $O(|V|b^k)$ in the number of paths. On one hand we are interested in using long paths to be as discriminative as possible, on the other hand the number of possible paths in a graph grows exponentially with k . Another consideration for k is the size of the corresponding subgraphs. Choosing a large k can result in not detecting a subgraph which has less vertices than k . In our implementation, we use the path length $k = 2$. This allows us to exploit the discriminative power of three connected sets. At the same time the number of paths has the worst case complexity of $O(|V|b^2)$. The clustering of corresponding paths

enables us to find the corresponding subgraphs with more than 3 vertices.

The computation of the paths is straightforward. As shown in Figs. 12 and 13, the graph can be represented by its adjacency matrix shown below in the figure. The representation of an object works as illustrated in Fig. 14. Every view of a model is processed in the following way:

1. The feature hierarchy is computed.
2. The enclosures of the sets are used to create the topological graph.
3. All paths (in our implementation of length 2) are computed.
4. Every path is encoded and stored in a data base. To encode a path we take the code of all pairs of sets.

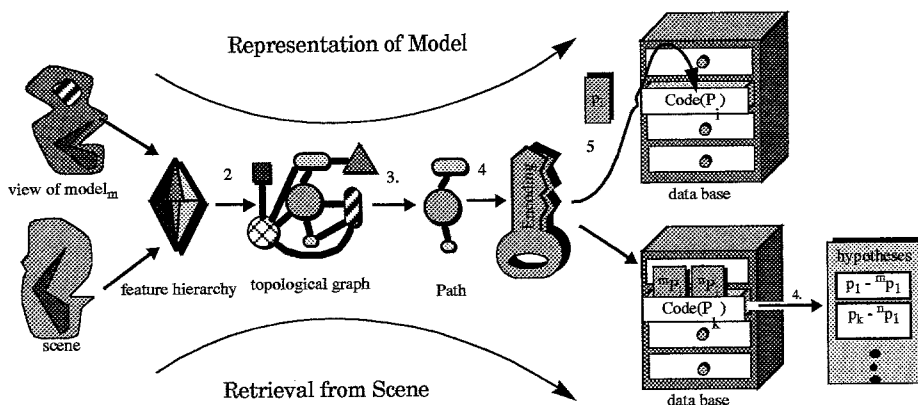


Figure 14. Representation of model and retrieval from scene.

We use the following attributes to encode a pair of sets:

- the label of the connecting edge,
- when the two sets are adjacent, the percentage of common boundary,
- when the two sets are intersecting, the percentage of area intersection.

All numerical values are quantized in a coarse way to allow significant deviations due to viewpoint change and noise. The quantization in our implementation for all numerical values is 20%.

The generation of the hypotheses proceeds in a similar way (see Fig. 14)

1. the feature hierarchy of the scene is computed,
2. the sets are used to create the topological graph,
3. all the paths are computed,
4. every path is encoded,
5. and retrieves from the data base the stored model paths which have the equivalent code.

The retrieved hypotheses are equivalent to subgraph isomorphisms of path length 2 between the different model-views and the scene. So far we have not made use of the groups in each set (node of the graph) because the code used for retrieval included only spatial and topological relationships between the sets. From all the candidate hypothesis we select those for whom there is at least one match between the groups of a node of a scene-path and a node of a model-path.

In the verification step we cluster the hypotheses in order to get larger corresponding subgraphs which are

likely to represent an instance of a model in a scene. Two hypotheses are consistent when the following rules apply:

1. They share at least one corresponding set pair.
2. No contradiction occurs. That means that the combined number of vertices and edges of the two paths in the model-view have to be the same as in the scene.
3. Connectivity has to be preserved. When two vertices are connected in one subgraph they have to be connected with the same label in the corresponding subgraph.

In this case, the combined hypotheses form a new hypothesis. These clusters grow iteratively until no further consistent hypotheses pairs can be found. An example can be seen in the matching between graph 1 and graph 2 of Fig. 12 and Fig. 13. We are matching paths of length 2. For simplicity, a path is only encoded by all its edge labels (maximal 3). For example, the path C_3 to C_1 via C_2 has the code bbi (or b^2i), because the edge from C_3 to C_2 has the label b , the edge from C_2 to C_1 is labelled with b and C_3 to C_1 is labelled with i . It is obvious that loops (closed circuits) of length 2 do not contribute to the structural information in the matching process. This fact is already implicit within an undirected edge. The hypotheses retrieval step of graph 2 leads to the two hypotheses listed in Fig. 15. The verification step tries to cluster these hypotheses. This is done by pairwise checking all hypotheses if the above mentioned consistency rule applies. The consistency check of h_1 and h_2 fails because rule 3 is violated. The connection between C_3 and C_4 has the label i and

Key	Entries	Hypotheses		#
		Path in Graph #1	Path in Graph #2	
b^2	$C_3-C_2-C_1$	$C_3-C_2-C_1$	$D_5-D_4-D_3$	h_1
iai	$C_3-C_4-C_1, C_3-C_1-C_4$			
ba(nil)	$C_2-C_1-C_4$	$C_2-C_1-C_4$	$D_4-D_3-D_2$	h_2

Figure 15. Database after object loaded (left), generated hypotheses (right).

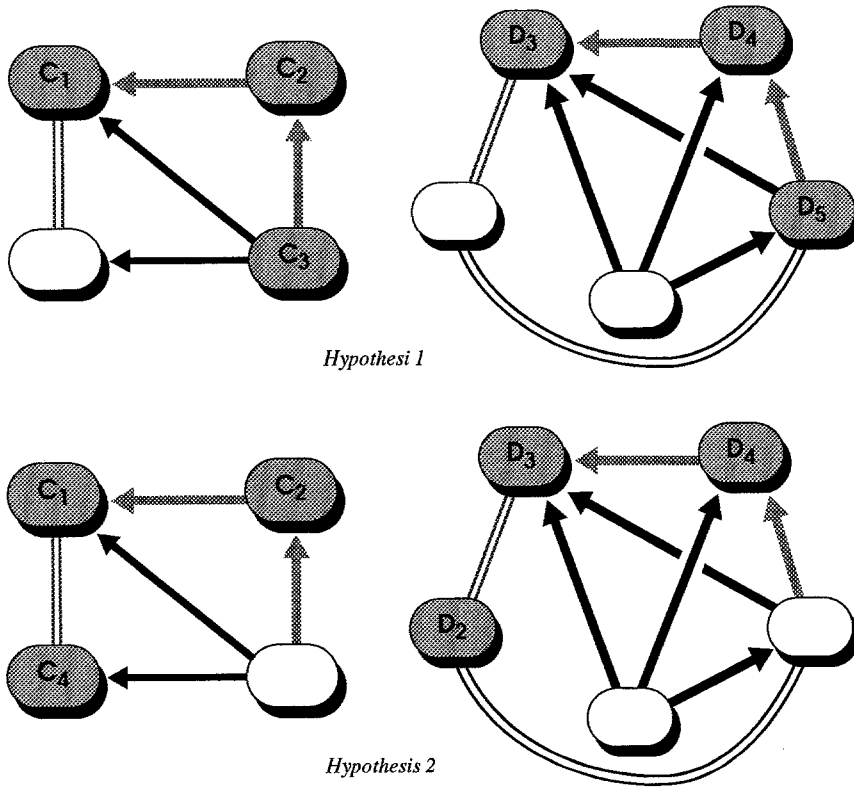


Figure 16. Corresponding subgraph isomorphisms.

the corresponding connection between D_5 and D_2 has the label a. Therefore we find two subgraph isomorphisms as shown in Fig. 16.

Analysis

What would have happened if we would have taken shorter or longer paths as basic matching primitives? Given k as the path length. Then c_k is the number of CA pairs in a path consisting of $k + 1$ vertices, with $c_k = (k + 1)k/2$. In Section 4 we talked about the

attributes which we use to encode a pair of enclosures of sets: the label of the connecting edge, the percentage of common boundary, and the percentage of area intersection. The label can have four different values: nil, adjacent, inside, or adjacent and inside. We further mentioned that we quantize the last two values in five quantizations of 20% each. Hence, the number of different codes to encode a pair of sets is 5 (in case of inclusion) +5 (in case of adjacent) +25 (in case of inclusion and adjacent) +1 (nil) = 36. Let this be a .

Therefore the number of available path codes of length k is

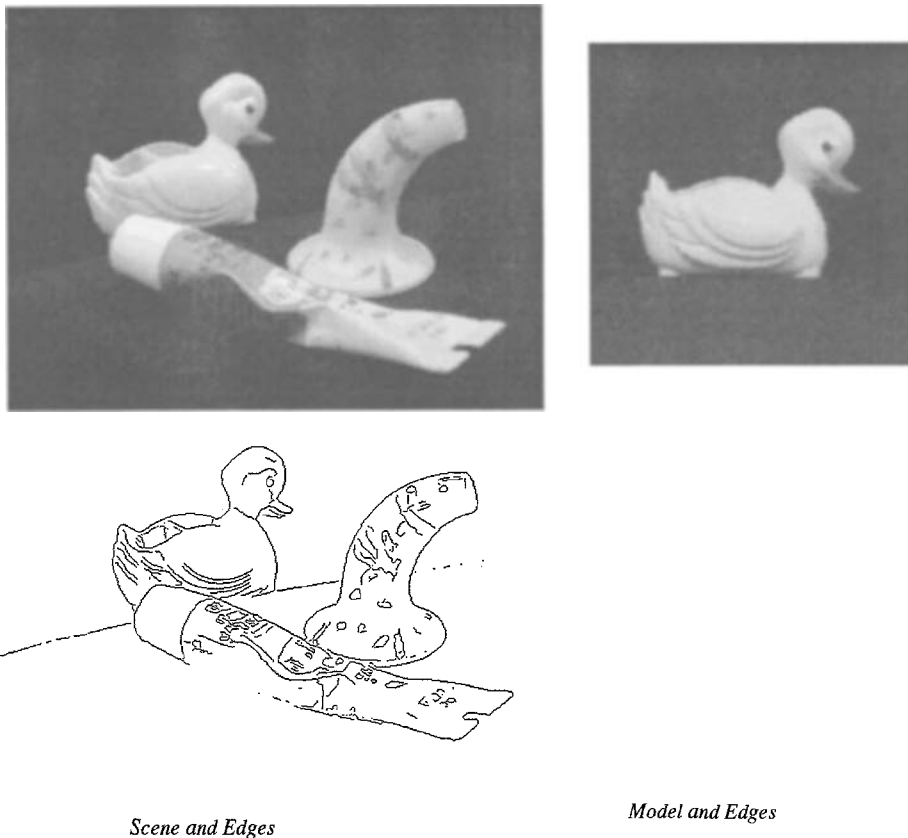
$$D_k = a^{c_k} = a^{(k+1)k/2}.$$

D_k is a measure for the discriminative power of an encoding scheme. The larger D_k , the larger the code alphabet, the more discriminative power a feature has. In our implementation we have $k=2$ and therefore $D_2 = 36^3 = 46656$. The trade-off lies in the generation of the matching primitives versus the generation of the hypotheses with respect to the discriminative power. Taking a short path length results in a low number of paths to generate. For $k=1$, the complexity of the number of paths is $O(|V|b)$, where b is the branching factor. On the other hand the discriminative power of these paths is lower. For $k=1$, $D_1 = 36^1 = 36$. Because the number of paths decreased by one order of magnitude and the discriminative power decreased by two orders of magnitude, the number of generated hypotheses h is in general larger than in our example. Because

the clustering of the hypotheses has a complexity of $O(h^2)$, the matching and verification for $k=1$ is slower than for $k=2$. Taking a long path length results in a large number of paths. For example, for $k=4$, the number of paths is $O(|V|b^4)$, and $D_4 = 36^4$. The number of generated hypotheses will be minimal due to such a high discriminative power, and therefore the final clustering will be very fast. But using paths of length $k=4$ is unlikely to succeed because objects in a scene rarely give rise to such large paths. Furthermore, occlusion reduces the probability of occurrence for a high value of k . The right way to proceed is to increase the value a . This can be done by improving the encoding scheme.

5. Results

In Fig. 17 we show an example of the performance of our current system. The model consisted of a side views of an instance of a duck. In the scene the duck



Scene and Edges

Model and Edges

Figure 17. Example 1: Scene and model.

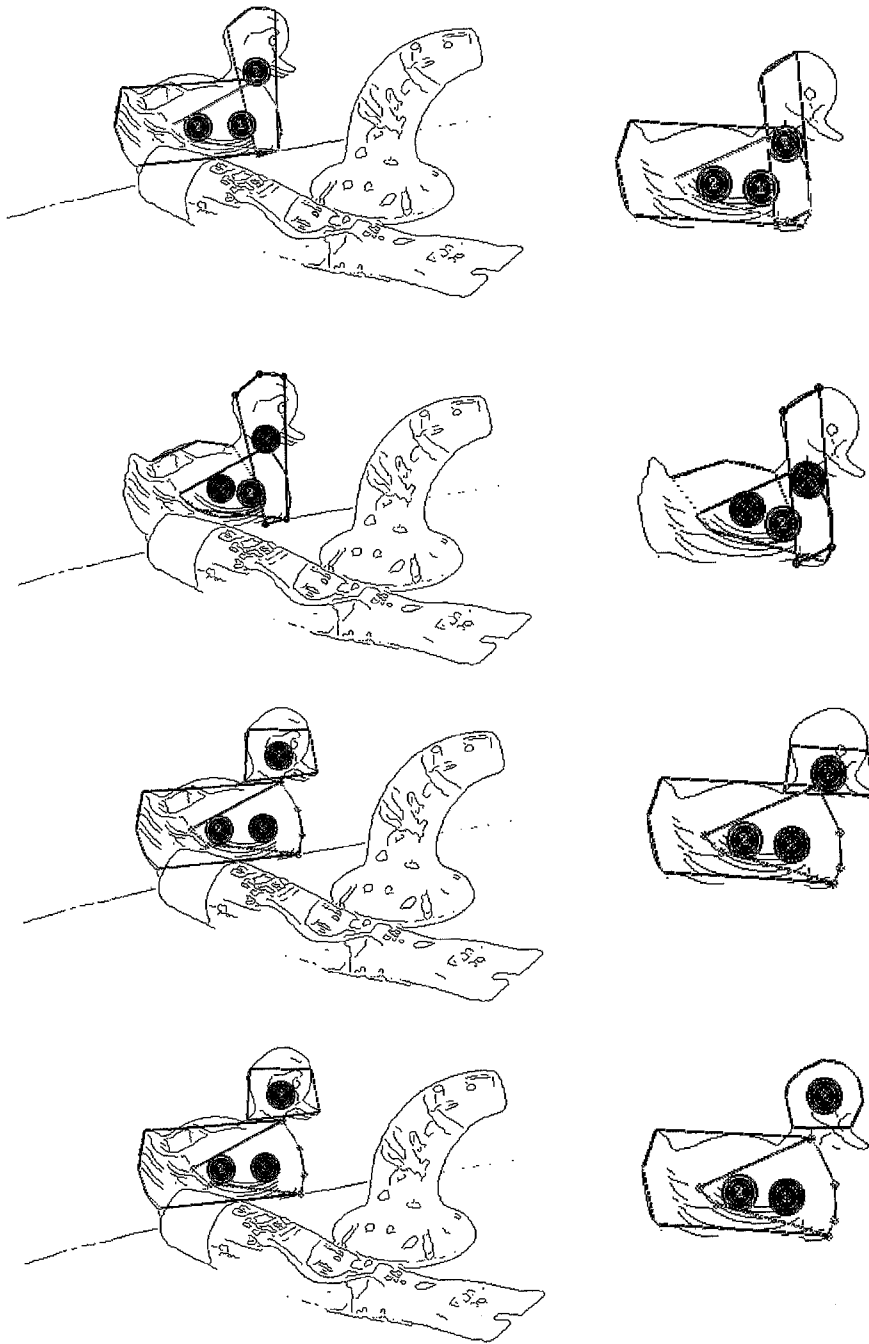


Figure 18. Example 1: Recognition hypothesis 1-4.

is rotated and slightly occluded. In the model, there were 20 curves detected, which resulted in 21 high level groups. These were organized into 9 sets out of which 2 were discarded by the reasoning presented in Section 3. The remaining 7 valid sets contained 6 relevant groups and 1 irrelevant one. The entire hierarchy

took about 13 seconds to compute for the model. The curves in the scene gave rise to 72 groups. These were organized into 24 sets out of which 6 sets could be discarded. The remaining 18 sets contained 12 relevant groups (out of which 5 were of the duck). The entire hierarchy took 136 seconds to compute on a Sun

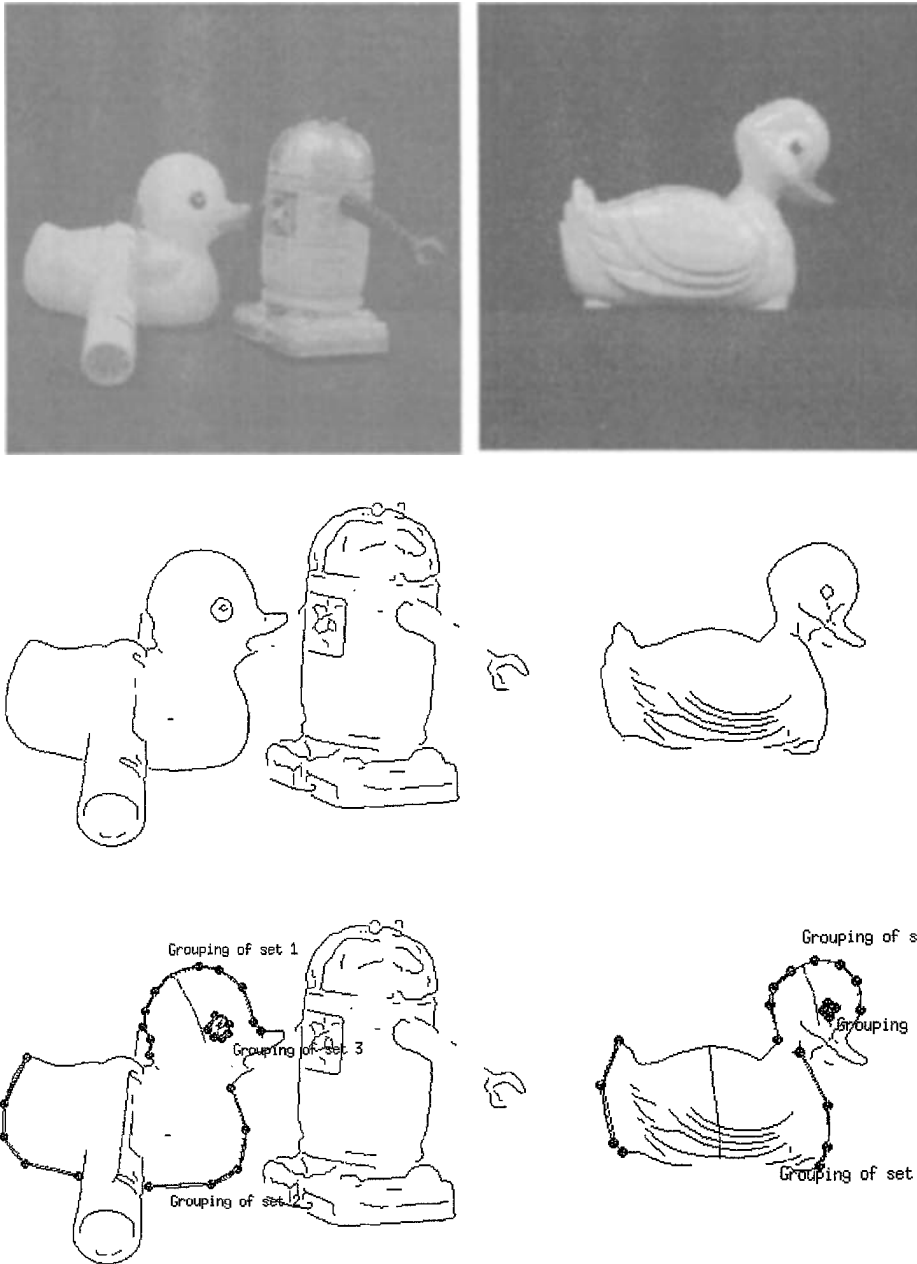


Figure 19. Recognition Example 2: (top) images of scene and model—duck in scene is different from that in model and partially occluded, (middle) edges of scene and model, (bottom) example of matched hypotheses.

Sparc-10. The sets were used to compute a graph. Four of the consistent hypotheses are shown in Fig. 18. The results are summarized below.

In Fig. 20 we show other examples of the performance of our system. The model used for this was the one as in the previous example. In the scenes however,

we had a similar view of another instance of the duck, which was partially occluded and/or rotated. The scenes resulted in 47 groups and 129 groups respectively. These were organized into 21 sets and 19 sets. The sets were used to compute a graph. In each example, we show one of the matched hypotheses. Note

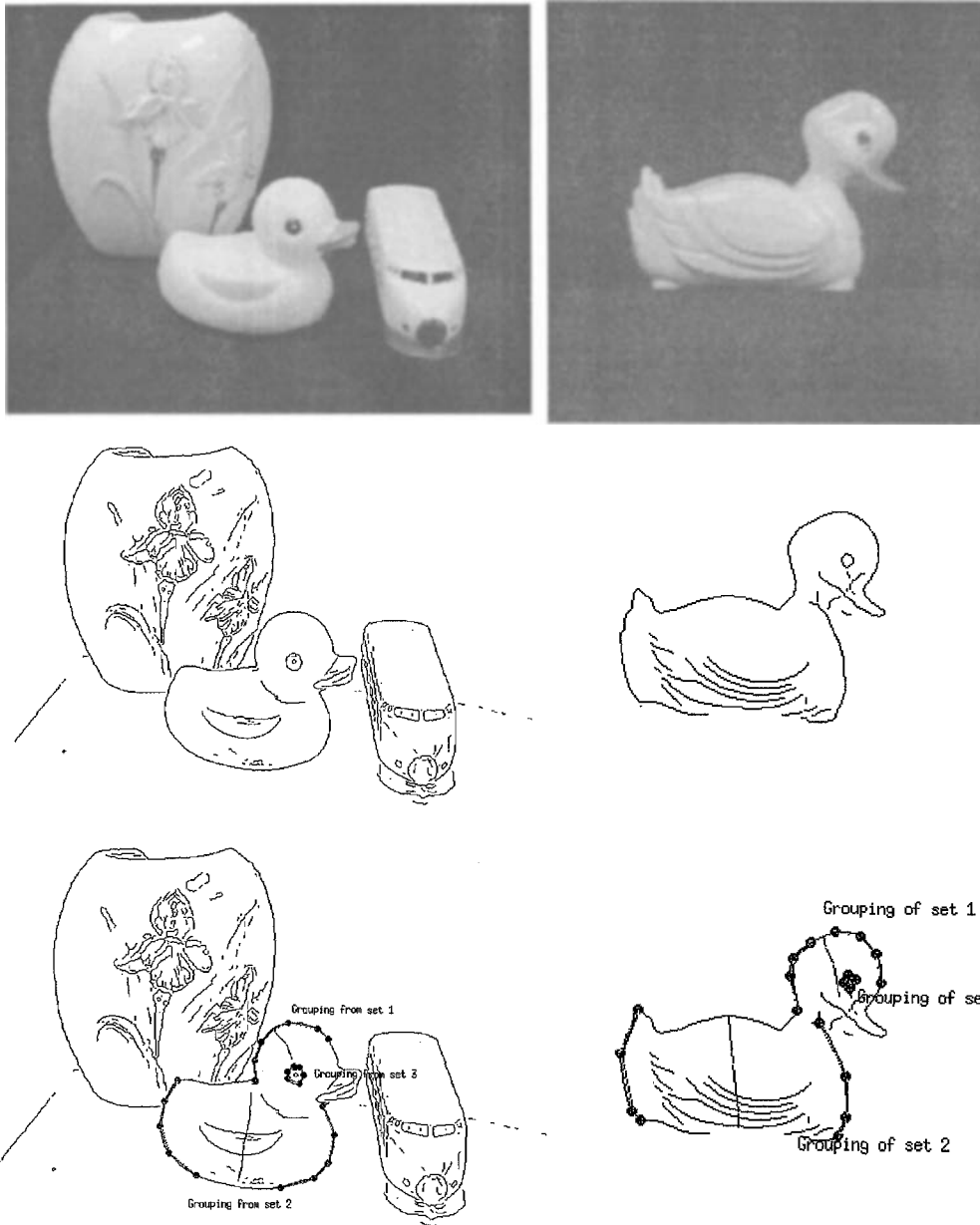


Figure 20. Recognition example 3: (top) images of scene and model—duck in scene is different from duck in model and partially rotated. (middle) edges of scene and model, (bottom) example of matched hypotheses.

that we have shown the groups of maximum cardinality in the corresponding sets.

6. Conclusion

We have developed an approach to use perceptual organization for the purpose of generic object recognition,

and show some promising results. Our perceptual grouping is purely data driven. We first compute all groupings using proximity indexing. We try to resolve ambiguities and try to discard groups which do not necessarily yield any physical interpretation. By using the feature groups for recognition purposes we make effective use of the underlying organization. In our system we emphasize qualitative rather than quantitative

Table 2. Grouping and recognition details of examples.

	Model	Scene 1	Scene 2	Scene 3
Number of groups detected	21	72	47	129
Number of sets	9	24	10	19
Number of sets discarded	2	6	2	4
Number of valid sets	7	18	8	15
Number of nodes in graph	7	18	8	15
Time taken to compute hierarchy	13 sec	136 sec	94 sec	159 sec
Time taken for recognition (after computation of hierarchy)	NA	4 sec	2 sec	5 sec
Number of hypothesis generated (paths of length 2)	NA	10	9	12
Number of consistent hypotheses	NA	7	2	2

tokens and try to achieve recognition using *spatial correspondences* of these tokens. By using multiple representations for each group, we can deal fairly well with occlusion. Scale is also handled well by the way we match these features groups or sets. We perform indexing on the spatial and topological constraints of the sets containing multiple representations of each group. By using a set of different views to represent a model we can deal with *incomplete model descriptions*.

The success of the approach described in this paper is largely dependent on the degree to which the grouping hierarchy can be extracted from the image. When the groupings are not present, the strength of the approach is weakened. By making use of multiple tolerances and various cardinalities to extract the groupings from the model and image, we increase the robustness of our approach.

In our implementation, we are able to recognize one view of a 3D model which appears in the scene with a different orientation, scale and partial occlusion. In order to take care of high orientation differences between the model and scene, we would require multiple views of the model to be initially processed in and stored in the database. Currently we do not make any assumptions on how the multiple views are interrelated, but acknowledge that the representation of a model using multiple views is an interesting and difficult problem and beyond the scope of this paper.

Our future work aims at answering the following questions.

- What happens when the system does not find corresponding high level groups (e.g., due to heavy occlusion or lack of edges)? We want to focus on this point

by developing a multilevel matching, which allows the system to “fall back” on lower level features in order to find correspondences.

- We want to extend the feature hierarchy by including perceptual organization between high level features. So far we use only proximity and closure to generate high level groups. We are investigating the possibility of including symmetry and parallelism.
- We would like to extend the indexing idea directly to the perceptual group sets rather than to the graph matching stage.

Our work and the corresponding results in this paper should demonstrate the viability of this approach.

References

- Alavi, Y., Chartrand, G., Lesniak, L., Lick, D.R., and Wall, C.E. 1985. *Graph Theory with Applications to Algorithms and Computer Science*. John Wiley and Sons.
- Biederman, I. 1987. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147.
- Breuel, T.M. 1989. Adaptive model based indexing. In *Proceedings of the DARPA IUW*, pp. 805–814.
- Brooks, R.A. 1983. Model based three dimensional interpretation of two dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150.
- Canny, J. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698.
- Clemens, D.T. and Jacobs, D.W. 1991. Space and time bounds on indexing 3-D models from 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1007–1017.
- Cornell, D. and Gottlieb, C. 1970. An efficient algorithm for graph isomorphism. *Journal of the ACM*, 17(1):51–64.
- Dickinson, S.J., Pentland, A.P., and Rosenfeld, A. 1992. 3-D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 174–198.

- Ettinger, G.J. 1988. Large hierarchical object recognition using libraries of parameterized model sub-parts. In *Proceedings of IEEE CVPR*, Ann Arbor, Michigan.
- Fan, T.J. 1990. *Describing and Recognizing 3-D Objects Using Surface Properties*. Springer Verlag: New York.
- Flynn, P.J. and Jain, A.K. 1991. 3D object recognition using invariant feature indexing. *IEEE Workshop on Directions in CAD-Based Vision*, Hawaii, pp. 115–123.
- Forsyth, D., Mundy, J.L., Zisserman, A., Coelho, C., Heller, A., and Rothwell, C. 1991. Invariant descriptors for 3-D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 971–991.
- Grimson, W.E.L. 1990. *Object Recognition by Computer—The Role of Geometric Constraints*, MIT Press: Cambridge, MA.
- Herault, L., Horaud, R., Veillon, F., and Niez, J.J. 1990. Symbolic image matching by simulated annealing. In *Proceedings of the British Machine Vision Conference*, University of Oxford, England, pp. 319–324.
- Kanade, T. 1981. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–4460.
- Kriegman, D.J. and Ponce, J. 1990. On recognizing and positioning curve 3-D objects from image contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1127–1137.
- Lamdan, Y., Schwartz, J.T., and Wolfson, H.J. 1988. On recognition of 3-D objects from 2-D images. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1407–1413.
- Lamdan, Y. and Wolfson, H.J. 1988. Geometric Hashing: A general and efficient model-based recognition scheme. In *Proceedings of IEEE ICCV*, Tampa, Florida, pp. 218–249.
- Lowe, D.G. 1987. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395.
- Mohan, R. and Nevatia, R. 1989. Using perceptual organization to extract 3-D structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139.
- Nevatia, R. and Price, K. 1992. Locating structures in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(5):476–484.
- Noltmeier, H. 1976. *Graphentheorie*. de Gruyter.
- Parvin, B. and Medioni, G. 1991. A dynamic system for object description and correspondence. *Proceedings of IEEE CVPR*, Maui, Hawaii, pp. 393–399.
- Rao, K. and Nevatia, R. 1989. Description of complex objects from incomplete and imperfect data. In *Proceedings of the Image Understanding Workshop*, Palo Alto, California, pp. 399–414.
- Roberts, L.G. 1968. *Machine Perception of Three Dimensional Solids*. *Optical and Electro-Optical Information Processing*, pp. 159–197.
- Rom, H. and Medioni, G. 1993. Hierarchical decomposition and axial shape description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 973–981.
- Rothwell, C. 1993. Hierarchical object descriptions using invariants. *Applications of Invariants in Computer Vision II*, Azores, pp. 287–302.
- Saint-Marc, P. and Medioni, G. 1990. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision*, Antibes, France, pp. 604–606.
- Stein, F. Structural indexing for object recognition. Ph.D. Thesis, USC IRIS.
- Stein, F. and Medioni, G. 1992a. Structural indexing: Efficient three dimensional object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 125–145.
- Stein, F. and Medioni, G. 1992b. Structural indexing: Efficient two dimensional object recognition (correspondence). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1198–1204.
- Stein, F., Medioni, G., and Havalдар, P. 1993. Recognizing 3D objects from 2D groupings. *Proceedings of the Workshop on Computer Vision in Space Applications*, Antibes, France, pp. 453–464.
- Ullman, S. and Basri, R. 1991. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 992–1006.
- Ulpinar, F. and Nevatia, R. 1993. Perception of 3-D surfaces from 2-D contours. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 3–18.
- Witkin, A. and Tanenbaum. 1983. On the role of structure in vision. In *Human and Machine Vision*, J. Beck, B. Hope, and A. Rosenfeld (Eds.). Academic Press: New York, pp. 481–543.
- Zerroug, M. and Nevatia, R. 1993. Scene segmentation and volumetric descriptions of SHGCs from a single intensity image. *Image Understanding Workshop*, pp. 905–916.