EXTRACTION OF GROUPS FOR RECOGNITION*

Parag Havaldar, Gérard Medioni and Fridtjof Stein[1]
Institute for Robotics and Intelligent Systems
University of Southern California, Los Angeles, California 90089-0273
email: havaldar@iris, medioni@iris.usc.edu

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. 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 initial 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. Grimson's book [10] gives a lucid treatment for the *geometric constraints* used in these approaches. 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. However, this approach cannot be extended to more general scenarii containing objects which 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 obviates the 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's theory [2], which states the sufficiency of a limited number of volumetric components (or geons) for the task of recognition. Recovery of parts and their arrangements can help fast recognition of objects even if they are occluded, novel, rotated in depth or extensively degraded.

We therefore have three problems to solve: the 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 first launched by the classic work of L. G. Roberts [19]. Brooks [4]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[14] system takes a bottom-up approach to object-centered recognition. Rothwell in [21] 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.

In section 2 we describe the feature hierarchy computed. As explained in some of our previous work [25], 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. Some of the groups extracted may not yield any natural descriptions. Hence, in section 3, we perform a selection step by organizing the groups into sets which have

* 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

1 Fridtjof Stein is currently with Daimler Benz, Germany

Lecture Notes in Computer Science, Vol. 800
Jan-Olof Eklundh (Ed.)
Computer Vision – ECCV '94
© Springer-Verlag Berlin Heidelberg 1994

similar "perceptual" content making use of junctions to reason about relevant sets. In section 4, we give an outline of our recognition system which uses these sets for recognition. As models we use multiple views of an object. Results are shown in section 5.

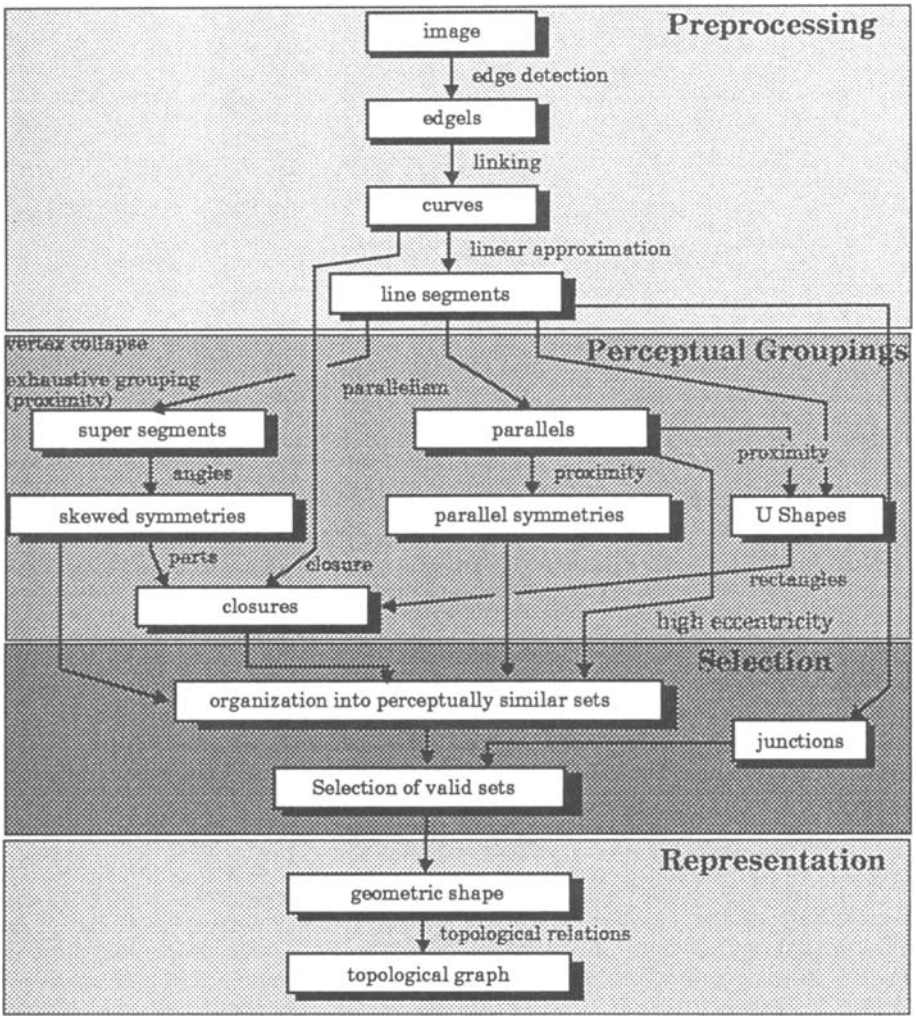


Figure 1 Feature Hierarchy

2 Going from Edgels to Groups

In our previous work [23], the super segment was introduced in two or three dimensions as a feature to represent a piece of a curve. It is based on the assumption that the underlying structure embodies continuity. Here, we propose to go to higher level groups which take into account other grouping criteria besides co-curvilinearity, such as parallelism, closure, and symmetry. In computer vision, many authors have focused on computing perceptual groups (see *e.g.* [11],[14],[19],[12],[20][22],[28]). 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. On

the other hand, we do this in an efficient way by using *proximity indexing*.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 Figure 1.The following sections focus on the details of the perceptual hierarchy.

2.1 Preprocessing

In the *preprocessing* stage we reduce the amount of data: starting from images we first detect edges in the image (using the Canny edge detector [5]). We then 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. 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. Below in Figure 2 we show an example scene and the detected edges. Note that although this is the same image as the one used in Zerroug and Nevatia [29], we follow a very different line of reasoning. In particular, we make no assumptions about the kind of objects we deal with.

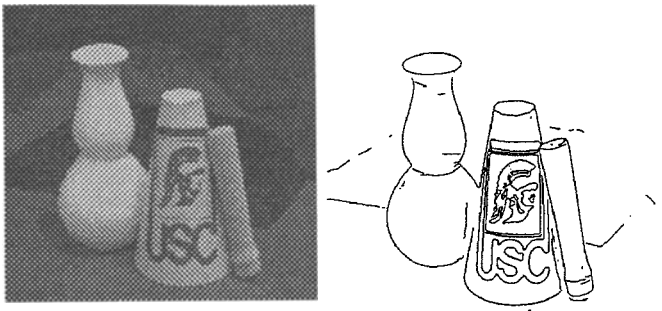


Figure 2 Example of image and detected edges[1]

2.2 Super Segments

Since we want to handle occlusion, we do not expect to obtain complete boundaries in our images, but only portions of them. Grouping a fixed number of adjacent segments provides us with one of our basic features, the super segment. The computation of super segments is the same as described in [23]. Connected linear segments form chains of adjacent segments. We generate super segments from cardinalities3 to 6..

2.3 Parallels

Segments which are parallel within a certain tolerance (δ=20°) are grouped as parallels. For each linear segment, the possible candidate parallels are retrieved and verified with respect to aspect ratio and overlap. Segment pairs which meet theses constraints generate parallels. Using proximity indexing, we are guaranteed to find parallels which are at most $\delta/2$ apart and we get some parallels with angles between $\delta/2$ and δ.

2.4 Symmetries

Symmetries have been used by various authors [15],[28]. We detect two specific symmetries here as features of an object.

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. Examples are shown in Figure 3 (left). Skew symmetry was first proposed by Kanade [11] and its extraction was done by Ponce [12] and Saint-Marc and Medioni [22] but these methods are quite sensitive to noise. We

1 Image - courtesy M.Zerroug

are interested in symmetries between line segments. Our approach is not exhaustive. We use supersegments to detect skew symmetries. Two super segments are skew symmetric, if they satisfy the following: (a) the difference between the corresponding angles must be smaller than $2\theta_{max}$, and (b) the symmetry axis has to be straight. An example is given in Figure 3 (middle).

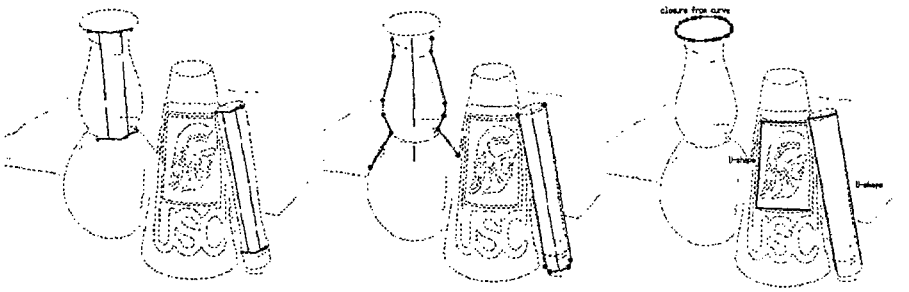


Figure 3 Examples of groups - parallel symmetries (left), skew symmetries (middle) and U-shapes (right)

2.5 Closures

Lowe [14] states: *There is a tendency for curves to be completed so that they form enclosed regions.* Based on this statement, Mohan and Nevatia [15] 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. At the moment we compute closures from U-Shapes, from closed curves and from skewed symmetries. 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. 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.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. We adopt the idea that a segmentation into parts should be done at negative minima of curvature from Rom and Medioni [20]. Such "a part" is used in our system as a closure. Whenever we encounter a sign change of consecutive angles, we "break" the symmetry at this point. Applying this step iteratively, we generate alternating convex and concave parts. We use the convex parts to create closures. Examples of closures can be seen in Figure 3 (right).

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 Figure 4. Two features match only in the case when they fall into the same interval, which may not always be so. Flynn and Jain [9] 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.

We propose an alternative approach: We encode every feature twice and use indexing on every value separately. Every value is quantized twice. 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

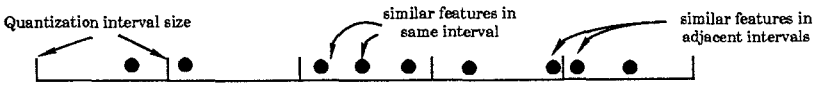


Figure 4 The indexing problem

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. The undesired groups also increase the complexity of the representation and matching process (see section 4). Selecting relevant groups 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. Note that each set may be perceptually correct or incorrect. To pick out relevant groups we make use of junctions and reason at the level of the above sets.

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 such a 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 close together.

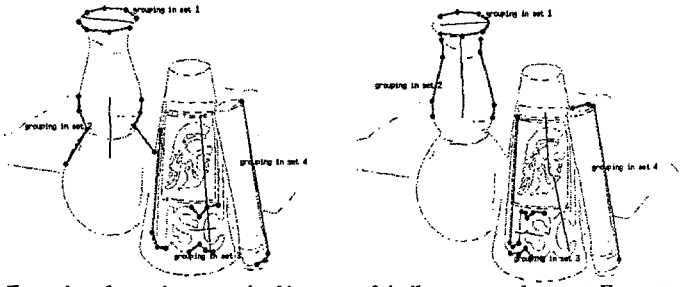
1. We first aggregate the groups having similar axes orientations into different sets. The groups in each set may still be spatially located at different positions.
2. Next we choose each set and further partition it depending upon spatial neighborhoods. This is done by constructing a graph whose nodes 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 Figure 5 (top) we show examples of four such group sets. Each image displays one group in each set. Below we discuss how to select valid sets.

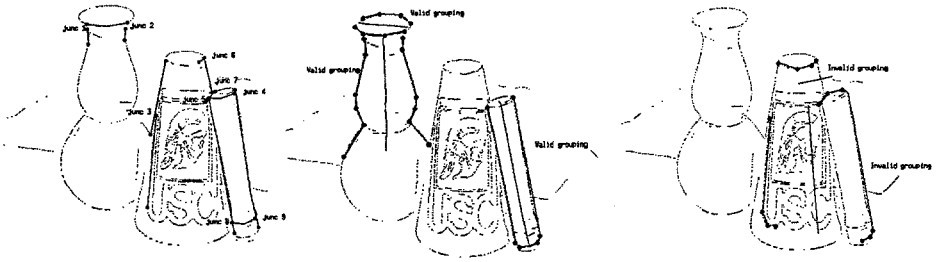
3.2 Selection of sets

We first compute all the junctions in the image and use them to decide the validity of the sets. 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. Some computed junctions are shown in Figure 5 (bottom left). Next we break the segments of the groups in each set into two subsets, each subset containing segments on either side of the axes. 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 Figure 5 (bottom right), we show some of the valid and invalid groups in the set as a result of this reasoning for the junctions shown. It can be seen 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 (top). It can be seen that in this case the segments of the junctions are shared between two sets and not three.



Examples of groupings organized into sets of similar perceptual content. Four sets are shown. Each image shows one group of each set.



Examples of some junctions detected in image Examples of valid and invalid groups for T-junctions shown above

Figure 5 Selection of Sets

At this stage we have the groups organized into sets, each set contains groups which as perceptually similar and can now be used as tokens for recognition. Note that each set yields a multiple representation of a "part" in the form of many groups. We can encode the spatial relationships of these sets to form a graph to describe the object. 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 minimize 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. For basic graph related definitions used here and more on graph theory see [17],[1],[6]. 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 convex approximations (CAs) of the set. This gives us the geometric shape of the set which enables 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. In Figure 6, we show an example of a scene of four CAs and the corresponding graph.

In computer vision, graph matching is widely used (see *e.g.* [16], [8], [19], [18]). 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.* [6]). 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

We need to decide upon structural tokens to perform indexing. Using the sets of groups only would be very expensive. In our implementation the "color" of the graph lies in 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 to cluster these to find the largest consistent group of such paths. In a complete connected graph consisting of $|V|=m$ vertices, there are m-1 outgoing edges at every vertex ($|E| = (|V|-1)!$). Therefore the number of paths of length k is $m(m-1)(m-2)...(m-k)$. This corresponds to an upper bound of $O(|V|^{k+1})$ in the number of paths. A more realistic assumption is that there are only a constant number of outgoing edges at every vertex. This results in an upper bound of $O(|V|^{k})$ 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 $O(|V|^{2})$ (assuming a constant number of outgoing edges at every vertex). The clustering of corresponding paths enables us to find the corresponding subgraphs with more than 3 vertices.

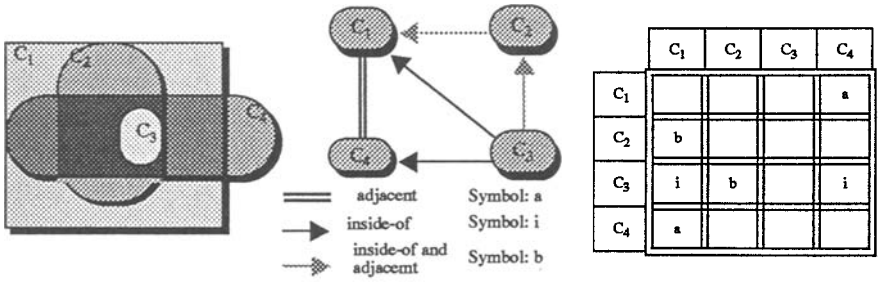


Figure 6 Example of Graph

The computation of the paths is straightforward. The graph can be represented by its adjacency matrix.The representation of an object works as illustrated in Figure 7. 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. Each path is encoded
5. Each path is stored in a data base.

To encode a path we take the code of all pairs of sets. 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 typical quantization in our implementation for all numerical values is 20%.

The generation of the hypotheses proceeds in a similar way (see Figure 7)

1. we perform steps 1 through 4 above, then we
2. retrieve 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. 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.

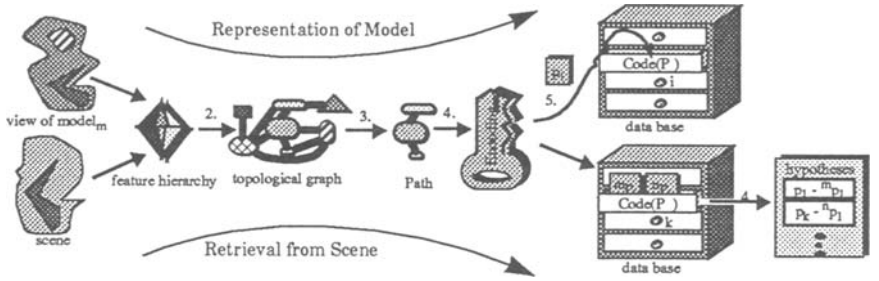


Figure 7 Representation of Model and Scene

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. That means we have a=4*5*5=100 different codes to encode a pair of sets. 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.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 number of paths is $O(|V|)$. On the other hand the discriminative power of these paths is lower. For k=1, $D_1=100^2=10000$. 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 num-

ber of paths is $O(|V|^4)$, and $D_4=100^{10}=10^{20}$. 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 computing $O(|V|^4)$ paths requires a high space complexity which may be prohibitive. The right way to proceed is to increase the value a. This can be done by improving the encoding scheme.

5 Results

In Figure 8 (top) we show an example of the performance of our current system. The model used for this was one view of an instance of a duck. In the scene however, we had a similar view of another instance of the duck, which was partially occluded. The model gave rise to 21 high level groups resulting in 9 sets out of which 2 were discarded by the reasoning presented in Section 3. Among the 7 valid sets contained 6 relevant and 1 was irrelevant. The entire hierarchy took about 13 seconds to compute for the model. The scene gave rise to 47 groups which were organized into 21 sets out of which 5 sets could be discarded. The remaining 16 sets contained 12 relevant sets (out of which 5 were of the duck, and 7 were of the other objects). The sets were used to compute a graph. One of the matched hypothesis is shown. The entire hierarchy took about 2 minutes to compute.

In Figure 8 (bottom) we show another example. In this scene the duck was slightly rotated and from a different viewpoint. The scene resulted in 88 curves, which gave rise to 129 groups. These were organized into 19 sets out of which 4 sets could be discarded. The entire hierarchy took about 6 minutes to compute.

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 try to resolve ambiguities and try to discard groups which not necessarily yield any physical interpretation.Our system emphasizes qualitative rather than quantitative tokens and tries to achieve recognition using *spatial correspondences* of these tokens. By using multiple representations for each group, we can deal fairly well with occlusion and scale.By using a set of different views to represent a model we can deal with *incomplete model descriptions.*

Our future work aims at taking care of cases when the system does not find corresponding high level groups (e.g. due to heavy occlusion)? 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 would like to extend the indexing idea directly to the perceptual group sets, rather then by using their approximations. There are several other features and group strategies which we ignore so far: continuation, texture, saliency etc. Including these features would enrich the descriptive and discriminative power of our feature hierarchy.

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

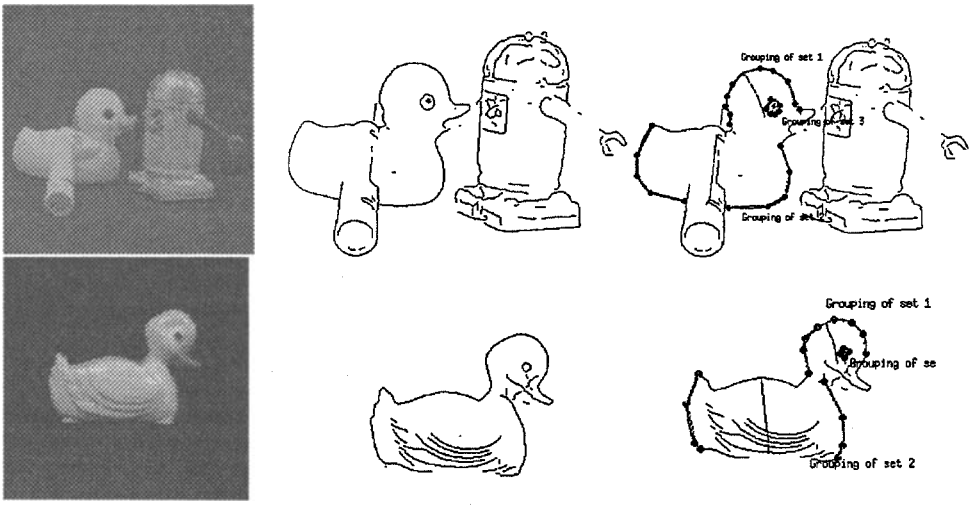
7 References

[1] Y. Alavi, G. Chartrand. L. Lesniak, D.R. lick, and C.E.Wall. *Graph Theory with Applications to Algorithms and Computer Science.* John Wiley and Sons, 1985

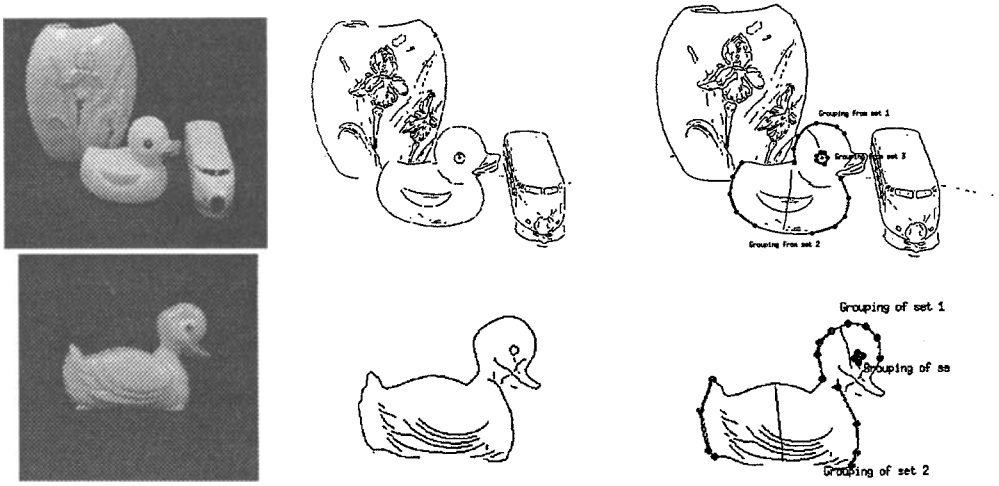
[2] I. Biederman. Recognition-by-Components: *A Theory of Human Image Understanding.* Psychological Review 1987, Vol. 94, No 2, 115-147.

[3] T . M. Breuel, *Adaptive Model Based Indexing*, In Proceedings of the DARPA IUW, pages 805-814, 1989.

[4] R. A. Brooks. *Model based three dimensional interpretation of two dimensional images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(2):140-150, 1983.



Example 1: (left) images of scene and model - duck in scene is different from that in model and partially occluded, (middle) edges of scene and model, (right) example of matched hypotheses.



Example 2: (left) images of scene and model - duck in scene is different from duck in model and partially rotated. (middle) edges of scene and model, (right) example of matched hypotheses.

Figure 8 Recognition examples (1) -top and (2) - bottom

[5] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, 1986, pages 679-698.

[6] D. Corneil and C. Gotlieb. *An efficient algorithm for graph isomorphism.* Journal of the ACM, 17(1):51-64, January 1970

[7] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, *3-D shape recovery using distributed aspect matching.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, pages 174-198.

[8] T.J.Fan. *Describing and Recognizing 3-D Objects Using Surface Properties.* Springer Verlag, New York,1990

[9] P.J.Flynn and A.K.Jain.*3D Object Recognition using invariant feature indexing.* IEEE Workshop on Directions in CAD-Based Vision,115-123,Hawaii, June 1934

[10] W. E. L. Grimson. Object Recognition by Computer - The Role of Geometric Constraints, MIT Press, Cambridge, MA 1990.

[11] T. Kanade. *Recovery of the three-dimensional shape of an object from a single view.* Artificial Intelligence, 17:409-4460, 1981.

[12] D. J. Kriegman and J. Ponce. *On Recognizing and Positioning Curve 3-D Objects from Image Contours.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, pages 1127-1137

[13] Y. Lamdan, J. T. Schwartz and H.J.Wolfson. *On Recognition of 3-D Objects from 2-D Images.* In Proceedings of IEEE International Conference on Robotics and Automation, April, 1988

[14] D.G. Lowe. *Three-dimensional object recognition from single two-dimensional images*, Artificial Intelligence 31, 1987, 355-395.

[15] R. Mohan and R. Nevatia. *Using Perceptual Organization to Extract 3-D Structures*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 11, November 1989, pages 1121-1139.

[16] R.Nevatia and K.Price. *Locating structures in aerial images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 4(5): 476-484, September 1992.

[17] H. Noltmeier. *Graphentheorie.* de Gruyter, 1976.

[18] B. Parvin and G. Medioni. *A Dynamic System for Object Description and Correspondence*, Proceedings of IEEE CVPR, Jun. 1991, Maui, Hawaii, pages 393-399.

[19] L. G. Roberts. Machine Perception of Three Dimensional Solids. Optical and Electro-Optical Information Processing, pages 159-197, 1968.

[20] H. Rom and G. Medioni. *Hierarchical Decomposition and Axial Shape Description*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Oct 1993, pages 973-981

[21] C. Rothwell, *Hierarchical Object Descriptions Using Invariants.* Applications of Invariants in Computer Vision II, pages 287-302, October 1993, Azores

[22] P. Saint-Marc and G. Medioni. *B-spline contour representation and symmetry detection.* In First European Conference on Computer Vision, pages 604-606, Antibes, France, April 1990.

[23] F. Stein and G. Medioni. *Structural Indexing: Efficient Three Dimensional Object Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 125-145, Feb 1992.

[24] F. Stein and G. Medioni. *Structural Indexing: Efficient Two Dimensional Object Recognition (correspondence).* IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1198-1204, Feb 1992.

[25] F. Stein, G. Medioni, and Parag Havaldar.*Recognizing 3D Objects from 2D Groupings*, Proceedings of the Workshop on Computer Vision in Space Applications, pages 453-464, Antibes, France 1993.

[26] A. Witkin and Tanenbaum. On the role of structure in vision. In J.Beck, B.Hope and A.Rosenfeld editors, *Human and Machine Vision*, pages 481-543. Academic Press, New York, 1983.

[27] S. Ullman and R. Basri. *Recognition by linear combinations of models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 992-1006, Oct 1991

[28] F. Ulupinar and R. Nevatia, *Perception of 3-D surfaces from 2-D contours,* In IEEE Transactions on Pattern Ananlysis and Machince Intelligence, pages 3-18, Jan 1993.

[29] M. Zerroug and R. Nevatia, *Scene Segmentation and Volumetric Descriptions of SHGCs from a Single Intensity Image.* Image Understanding Workshop 1993, pages 905-916.