GRANT
IN-74-CR
38004

A GRAPH THEORETIC APPROACH TO SCENE MATCHING

p.155

by

Heggere S. Ranganath
Laure J. Chipman
Computer Science Department
University of Alabama in Huntsville
Huntsville, AL 35899

(NASA-CR-187817) A GRAPH THEORETIC APPROACH TO SCENE MATCHING Final Progress Report, period ending 1 Aug. 1991 (Alabama Univ.) 155 p
CSCL 20F
G3/74
N91-31947
Unclas
0038004

Final Report
NASA Contract NCC8-16

RESEARCH OBJECTIVE AND APPROACH

A. Space Station Related Applications for Vision Systems

Computer vision systems which can perceive environment through sensors and respond with appropriate action or decision have numerous space station applications. Vision systems can be used to automate routine space station operations thereby relieving crewmen of repetitive tasks. This increases crew time available for more demanding operations requiring human skills. Some of the routine operations which can be performed by vision systems within the space station module are given below:

1) Vision intelligent robots can be used for operations such as locate, fetch, store and adjust.

2) During times the modules are not occupied by crewman, vision systems can be used for "watch dog" monitoring and reporting of unanticipated events. These include loose objects and instruments floating by, ECLSS cabin air anomalies, etc..

3) Some flight experiments such as microgravity crystal growth are difficult to instrument. Vision systems can be used to monitor such experiment, record data and alert crewmen only when necessary.

4) Vision systems are also useful for docking, servicing, assembly and other advanced space station operations. NASA inhouse research shows that providing computer vision capability for orbital maneuvering vehicle (OMV)) offers several advantages: provides independence from docking aids and communication links; eliminates communication delay for vehicle control and reduces operator training cost for remote control.

B. Research Objective

The general objective of the proposed research is to evaluate the potential of expert system approach for the development of computer vision system capable of performing routine tasks within the space station modules.

C Research Approach

The knowledge base contains the descriptions of several flight panels. Knowledge is organized and stored as files: 1) 2-D string file which provides inclusion and left-right-top-bottom relationship among objects. This information is used to direct search for the desired object; 2) Object type attribute file which contains several attributes for each type of object; 3) Feature file which has prominent features of all panels; 4) Scene description file for each panel; 5) Object location file which gives the inclusion relations of objects.

Scene matching techniques are needed for context sensitive object recognition. Context sensitive object recognition which recognizes the object in the context of the scene is more reliable than context free recognition. However, errors due to segmentation make scene matching problem a difficult task. Imperfect segmentation may produce any of the following errors: mismeasured objects, missing objects or relations, merged objects, or extra objects.

This research has produced a robust graph-based scene matching method which is capable of handling problems causes by imperfect segmentation. The approach is very general and may be used in many NASA and other applications. Therefore, the graph theoretic scene matching method is described as a general technique in this report. It is then applied to a specific NASA application (space station) in Chapter VII. The software is developed in Pascal as well as in C.

ABSTRACT

The ability to match two scenes is a fundamental requirement in a variety of computer vision tasks. This dissertation presents a graph theoretic approach to inexact scene matching which is useful in dealing with problems due to imperfect image segmentation. A scene is described by a set of graphs, with nodes representing objects and arcs representing relationships between objects. Each node has a set of values representing various attribute measurements of the object it represents. Each arc has values representing the relations between pairs of objects, such as angle, adjacency, or distance. With this method of scene representation, the task in scene matching is to match two sets of graphs. Because of segmentation errors, variations in camera angle, illumination, and other conditions, an exact match between the sets of observed and stored graphs is usually not possible.

In the approach developed, first the problem is represented as an association graph, in which each node represents a possible mapping of an observed region to a stored object, and each arc represents the compatibility of two mappings. Nodes and arcs have weights indicating the merit of a region-object mapping and the degree of compatibility between two mappings. A match between the two graphs corresponds to a clique, or fully connected subgraph, in the association graph. The task is to find the clique that represents the best match. Fuzzy relaxation is used to update the node weights using the contextual information contained in the arcs and neighboring nodes. This simplifies the evaluation of cliques. A method of handling oversegmentation and undersegmentation problems is also presented. The approach is tested with a set of realistic images which exhibit many types of segmentation errors.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The ability to match two scenes is one of the fundamental requirements in a variety of computer vision tasks including automatic navigation, object location, pictorial databases, and character recognition. The specific task accomplished during scene matching is application or problem dependent. Some of the more frequently encountered tasks are listed below:

1) Image registration: Let Image 1 and Image 2 be the two given images. Assume that the field-of-view of Image 1 is completely contained within the field-of-view of Image 2. Now, the problem of image registration is that of locating the subimage of Image 2 which best matches Image 1.

2) Scene recognition: In scene recognition, the goal is to classify the input image as one of the known images. For example, in character recognition, the goal is to classify the input character as one of a set of known characters.

3) 3-D scene construction: By matching images of a scene obtained from different positions, one can generate 3-D information about the scene. This is known as stereoscopic vision.

4) Object recognition: Scene matching techniques are used for context sensitive object recognition. Context sensitive object recognition which recognizes the object in the context of the scene is more reliable than context-free recognition.

Because of its usefulness in practical applications, scene matching has been a topic of interest for many years. The scene matching methods developed during the last three decades can be classified into three major categories: template matching methods, feature matching methods, and graph theoretic methods.

When the scenes to be matched do not differ in rotation and spatial resolution, template matching methods such as cross correlation and sequential similarity detection algorithms may be used [12]. The major problem with template matching methods is the high computation associated with them. Feature matching methods characterize each image by a pattern or feature vector and then match two images by matching their feature vectors [18]. Feature matching methods can tolerate minor geometric distortions. Many real world problems are not suitable for template and feature matching methods. For example, consider two images of the same scene obtained by sensors from different viewing points. Now the geometric attributes such as size and shape of objects, and distances between objects, will change from image to image. Under these conditions where most template and feature matching methods fail, graph theoretic methods are useful.

1.1 Graph Theoretic Scene Matching Approach

Scene matching is the process of finding a correspondence between regions of an observed image and objects in a stored representation of a scene. This matching process is the final stage in a computer vision system, shown in Figure 1-1. During segmentation (Stage 1) the input image is partitioned into meaningful regions. The region description stage extracts significant attributes of

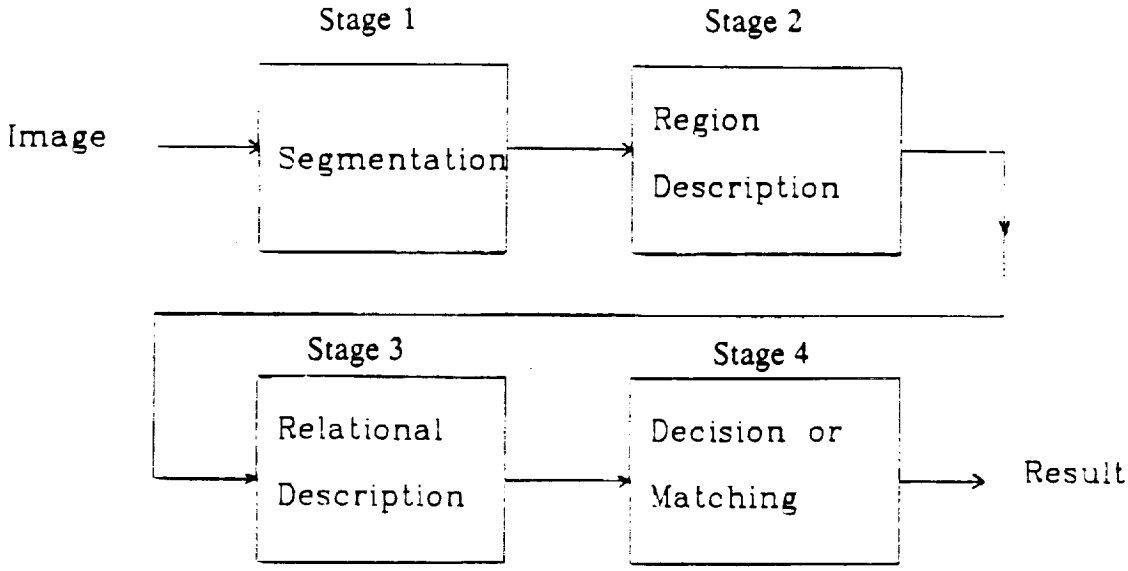


Figure 1-1: General model for computer vision systems.

each region. Relationships that exist among various regions are determined during Stage 3 processing. Matching the input image with the stored scene is accomplished during the last stage.

The idea of matching two scenes based on matching graphical representations of the scenes was first developed in 1975 [1]. A graph-based matching approach has the advantage that it can deal better with inexact matches caused by differences in viewing angles, scaling, or illumination, or by limitations of the segmentation algorithms used. In graph-based matching, regions in a scene are represented by vertices in a graph, and relationships between regions

are represented by arcs. The vertices have associated attribute values, which may include measurements of such properties as intensity, texture, area, or circularity.

A scene can be described by a set of graphs representing various relationships among the objects. As an example of graph-based scene representation, Figures 1-2 through 1-6 show a hypothetical scene and a set of graphs using the relations of adjacency, inclusion, reflectance, and texture. These figures are from Greene [13], who has developed a means of scene knowledge representation.

A perfect match between the graphical representations of an observed scene and a stored scene is an isomorphism between the two graphs. Two graphs are isomorphic if and only if there is a one-to-one mapping of all vertices of the two graphs such that all adjacency relationships are preserved. In the most general case, a matching between two graphical representations of scenes can be a many-to-many mapping of vertices $\mathbf{V}_o$ of the graph of the observed scene to vertices $\mathbf{V}_s$ of the graph of the stored scene. In matching two scenes, the best match is desired. The best match may be defined as the match that minimizes some measurement of differences in attributes of corresponding vertices and relations.

1.2 Problem Statement

Variations and uncertainties in camera angle, scaling, and illumination, and problems due to noise make exact scene matching difficult if not impossible.

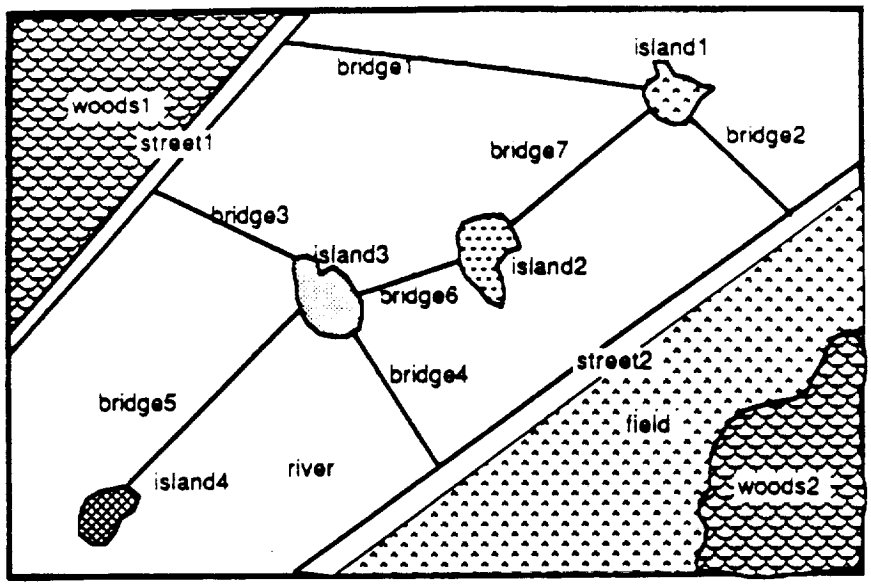


Figure 1-2: A hypothetical river/island scene.

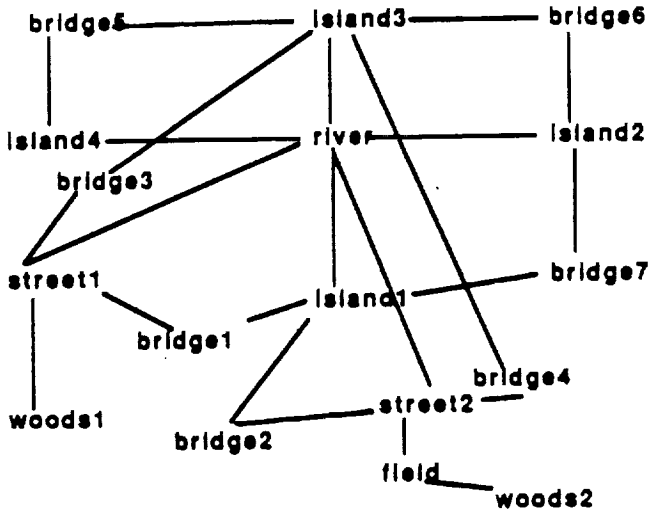


Figure 1-3: Adjacency graph for river/island scene.

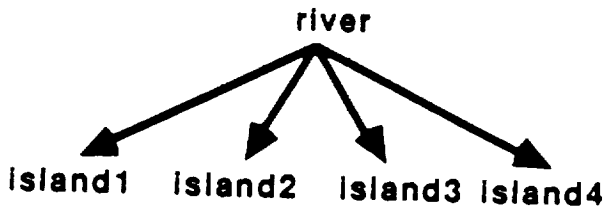


Figure 1-4: Inclusion graph for river/island scene.

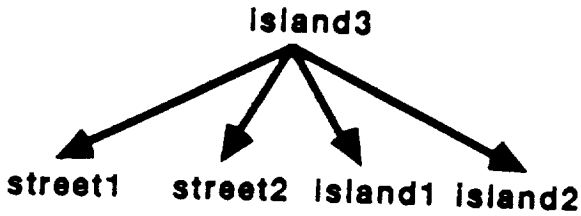


Figure 1-5: Reflectance graph for river/island scene.

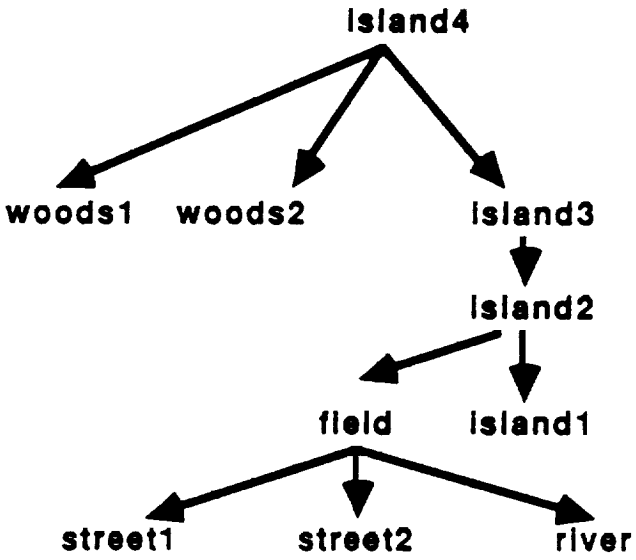


Figure 1-6: Texture graph for river/island scene.

Graph theoretic scene matching methods are able to handle inexact scene matching better than template and feature matching methods.

Previous research, however, tends to assume a perfect segmentation of the input image, meaning that a perfect match to a stored model can be found. Unfortunately, segmentation algorithms are not perfect. Several different types of errors can occur during segmentation. Little work has been done on the intelligent choice of relations and attributes to facilitate graphical matching in the real world of imperfect image segmentations. A shadow can cause an object's boundary to be incorrectly found. An apparent break in an object's boundary can cause the perimeter to be mismeasured. Extraneous marks or shadows may be segmented as objects that do not correspond to any objects in the stored scene representation. Shadows or marks can cause two or more objects to appear to the segmentation algorithm as one large object. An object may not be visible due to glare, shadows, or occlusion. Missing or changed relations are also possible.

In summary, imperfect segmentation may produce any of the following: mismeasured objects, missing objects or relations, merged objects, or extra objects. Any of these can cause a graph matching algorithm to fail unless the attributes and relations used, and the graph matching algorithm used, are intelligently chosen.

The objective of this research is to develop a robust graph-based scene matching approach which is capable of handling problems caused by imperfect segmentation.

The development of a good graph-based scene matching method is a significant contribution to computer vision, in general, and scene matching, in particular. This method can be applied to solve many interesting real world problems. Some of them are given below.

In the task of object location, the input scene is first identified as one of a number of known scenes. Once the scene is identified the goal is to locate a particular object within the scene. If the input scene can be represented symbolically as a set of attributed graphs, this symbolic representation can be matched with the stored scene representation through graph isomorphism. This allows the desired object to be located in the context of the scene, thus making object location more reliable.

In an automatic navigation application, several scenes (e.g. aerial photographs) are stored in memory for path finding purposes. During the actual flight, the system compares the acquired image with those stored in memory to stay on course. In general, the two scenes compared are obtained from different sensor positions, and exact geometric matching is not possible. A graph theoretic approach is highly desirable in such applications.

Pictorial databases constitute another application area for graph-based scene matching [5,17]. In this application, a user may enter, as a database query, a symbolic representation for a scene he wishes to retrieve. This representation must be matched to stored representations to find the correct scene(s). Again, a graph theoretic matching approach is useful, since the use of exact measurements of angles, distances, and object boundaries in a query would be cumbersome. A

perfect match between the query and a stored scene representation is possible here, since there is no image processing involved. The main problem in this application is to index a large number of scenes for quick searching.

In character recognition, the process is to classify an input handwritten character as an instance of a known character. In graph theoretic matching, component lines of the character are represented as vertices of a graph, and their relationships as arcs. This problem differs from the previously mentioned applications, since the differences between stored and observed versions of a character are real, and not simply due to segmentation errors or variations of camera angle.

1.3 Report Overview

Previous research in graph-based scene matching has left several problems unsolved, even when the segmentation process is error-free. The presence of segmentation errors adds several new problems to the existing list. This report builds a complete approach to the problem, from determination of scene representation through the evaluation of match merits.

To clarify the problems associated with segmentation errors, Chapter II presents some typical examples of segmented scenes exhibiting various types of errors. These common segmentation errors and their effects on graphical matching are analyzed. Chapter II also presents previous research into graphical matching techniques and their application to inexact scene matching. On the purely theoretical end of the spectrum, Ullmann and others [3,6,16,29] have devised algorithms for determining graph or subgraph isomorphisms. These

algorithms generally entail some modification to a basic approach consisting of finding a vertex mapping matrix, which indicates all possible mappings of observed to stored vertices, then checking all possibilities by use of a backtracking algorithm. Another approach to graph matching entails the use of association graphs [2,22,32]. The nodes in an association graph are defined over ordered pairs of vertices from the stored graph and the observed graph. An ordered pair is included as a node in the association graph if the two vertices in the ordered pair could map to each other, based on properties of the objects they represent. The mapping of several vertices from the observed graph to a single vertex in the stored graph is allowable, so that regions that were erroneously split by the segmentation algorithm can be mapped to a single object. An arc in the association graph from node A to node B indicates that the mappings represented in A and B are compatible with each other. Matches are then found by determining the largest cliques, or fully connected subgraphs, in the association graph. This approach can be used to find common subgraphs of two graphs.

The association graph method of matching is very promising since it has the potential to deal with any of the problems of imperfect segmentation: missing objects, extra objects, merged or split objects, and mismeasured objects.

The details of the vertex mapping matrix and association graph matching techniques are presented in Chapter II. Then, previous work which was specifically directed at the scene matching problem is presented and evaluated in the context of the segmentation problems presented at the beginning of the

chapter. The balance of the dissertation describes a new approach for graph-based scene matching which is better suited to matching in the presence of segmentation errors.

The first issue to be addressed is the intelligent selection of attributes and relations to be used to represent scenes as attributed graphs. Past research has seldom taken into account limitations of segmentation algorithms used on real-world input scenes. The relations, attributes, and primitives used seem sensible, but are seldom precisely defined, and have not been evaluated in the context of imperfect segmentations. A single set of attributes and relations may not be appropriate for several different types of scenes. For example, in an aerial photograph, the relations of adjacency and inclusion are natural choices, but in a scene consisting of well-separated blobs on a uniform background, these relations would not be useful.

Some obvious attributes that can be used to describe regions are size, intensity, shape, and texture. Attributes such as size and intensity are usually not useful unless they can be scale- or intensity- normalized first. Some attributes, such as perimeter, are quite susceptible to noise or limitations of boundary-finding algorithms. Also, differences in scale may cause disproportionate differences in perimeter measurements.

The relations used to describe a scene depend on the type of scene. Some useful relations have been investigated by Greene [13]. In an aerial photograph with regions that have shared boundaries, adjacency and inclusion are sensible choices. In a scene that can be approximately rotation-normalized, the relations

left-of and above make sense. Other relations, such as larger-than, more-textured-than, or brighter-than, are possible. The use of transitive or intransitive relations is another choice to be made. The use of an intransitive relation can make it more difficult to handle the problem of missing objects. In Chapter III, the selection of attributes and relations to minimize problems due to segmentation errors and to facilitate inexact matching is discussed.

Chapter IV begins the description of a graphical matching technique that allows for inexact matches. The starting point of the work presented here is to assign weights to the nodes and arcs of the association graph, according to how good a node-to-node compatibility is and how good a mapping-to-mapping compatibility is. An algorithm to find the 'best' clique in such a weighted association graph will be presented.

In Chapters V and VI, the use of weighted association graphs for inexact scene matching is developed. In Chapter V, a relaxation algorithm for updating the node weights of the association graph is presented. A simulation is run on several variations of an example scene, including problems of extra and missing objects, as well as mismeasured attributes and relations. The use of binary relations is contrasted with the use of real-valued relations. Another difficult problem is that of oversegmentation and undersegmentation. In Chapter VI, the basic algorithm of Chapter V is expanded to handle these problems. A simulation is run on example scenes that exhibit these problems, using real-valued relations.

Chapter VII describes an application of the matching algorithm developed here to a real-world scene matching problem. A system to demonstrate inexact scene matching for object location was developed for the National Aeronautics and Space Administration (NASA), at Marshall Space Flight Center. Object location would be a necessary capability in a machine vision/robot arm system for use in the space station laboratory module, which could ultimately handle routine tasks such as fetching and storing objects and monitoring experiments. In this system, we incorporate scene representation, matching, and match evaluation techniques developed in this research. Chapter VII includes the results obtained by running the relaxation algorithm on an actual scene of a space shuttle simulator panel which was used as a realistic test image for the object location system. This chapter also includes a discussion of the practical considerations of using this algorithm on a real-world problem of realistic proportions.

Chapter VIII is an evaluation of this work, its limitations and possibilities for future research.

II. PREVIOUS RESEARCH

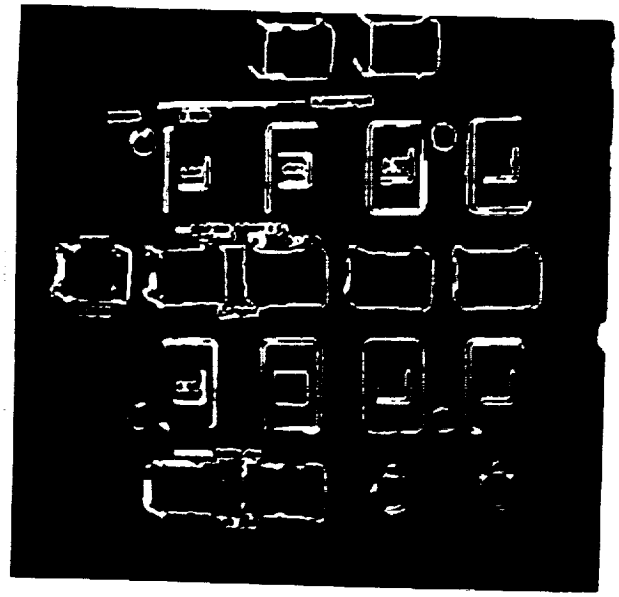
A graph-based or graph-theoretic approach for scene matching was first reported in 1975 [1]. Since then, several researchers have attempted and succeeded in dealing with inexact matches based on topology-like features. In this chapter, the existing graph theoretic scene matching methods are summarized and then analyzed to determine the implications of imperfect segmentation on their performance. Also, the analysis of various types of segmentation errors and their effect on graph theoretic scene matching methods is needed in determining the direction for future research. Different types of image segmentation errors are discussed with illustrations in Section 2.1.

2.1 Segmentation Errors

A robust, reliable, and accurate image segmentation system must form the foundation of every computer vision system. However, in the last three decades, research has not yet produced a truly reliable segmentation system which can handle varying imaging conditions and noise. There are several factors which make image segmentation a difficult problem, and Hung has addressed these problems in detail [15].

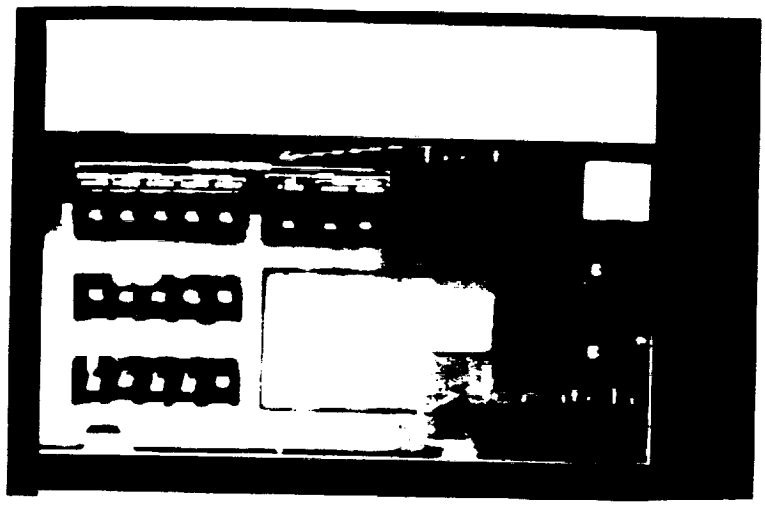
With the help of examples, in this section a few errors which are common to image segmentation are illustrated. Three space shuttle simulator panels before and after segmentation are presented in Figures 2-1 through 2-3. The

(a)

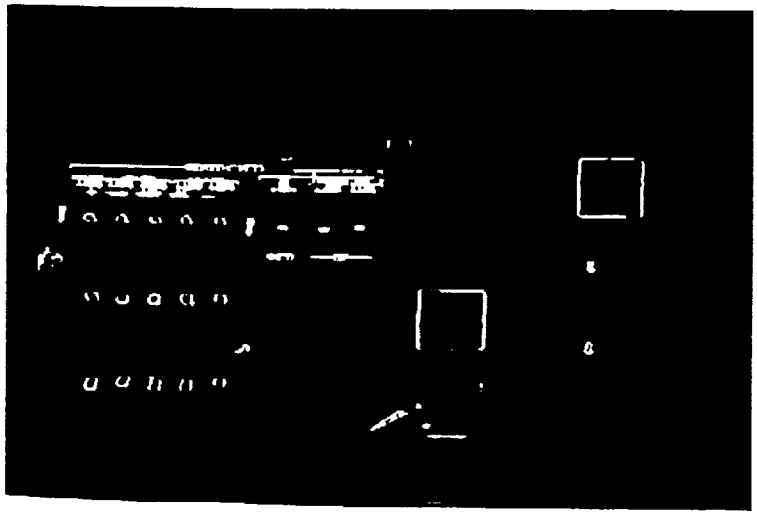


(b)

Figure 2-1: Segmentation errors: merged and missing objects.

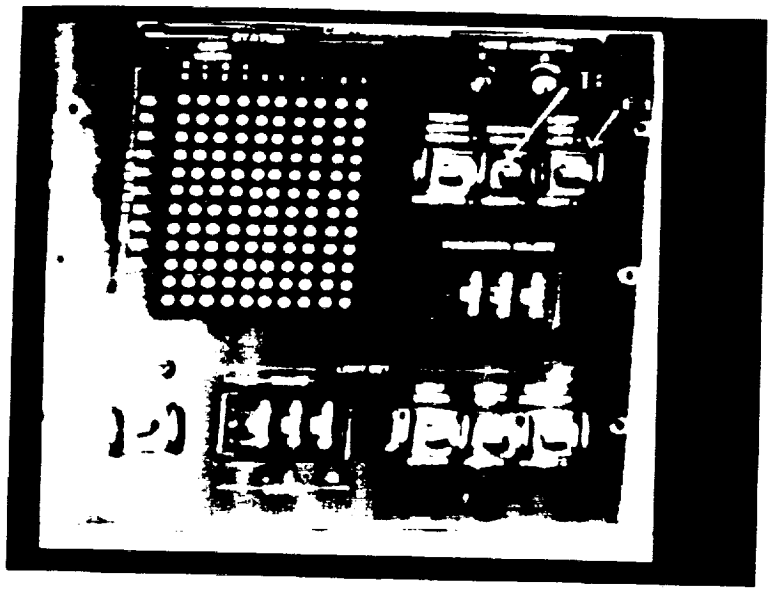


(a)

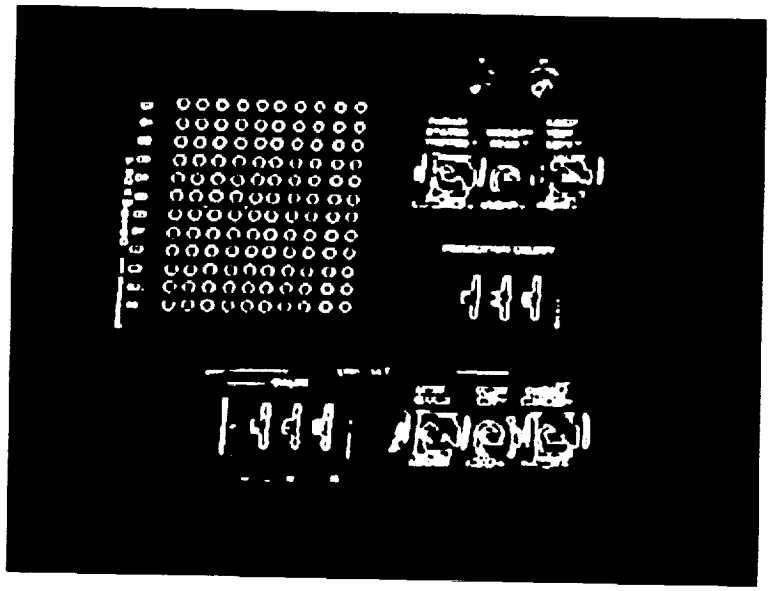


(b)

Figure 2-2: Segmentation errors: extra objects.



(a)



(b)

Figure 2-3: Segmentation errors: split objects and mismeasured boundaries.

digital image captured by a solid state camera is segmented by using segmentation software on a commercially available image processing system (Perceptics). During segmentation, one or more objects may go undetected. Object A in Figure 2-1(a) is not present in the segmented image, Figure 2-1(b). This type of error is normally due to poor contrast between the background and objects. Blurred edges may also be the cause for this type of error.

When the boundary between two objects which are close to each other is not clear, undersegmentation is possible. In an undersegmented image, several objects may merge together to form a single region. In Figure 2-1(a), objects B and C merge to produce a single region in Figure 2-1(b). Similarly, D and E have merged together.

Shadows, glare, and sometimes severe noise may result in extraneous regions in the segmented image which do not correspond to any real object in the scene. In Figure 2-2(b), A is a cluster of extra regions which are due to the text printed on the panel. It is not easy to mask or prevent such regions from appearing in the segmented image. Region B corresponds to a scratch on the panel.

The presence of shadows and noise may also cause oversegmentation in which pixels belonging to an object are partitioned into several disjoint regions. In Figure 2-3(a), object A is split into three regions.

Poor segmentation may yield incorrect values for geometric and intensity attributes used to characterize regions. Object B in Figure 2-3(a) appears much

smaller than its actual size in Figure 2-3(b). This is due to the inability of the boundary detection algorithm to handle the shadow within the bright object. The net effect is a set of distorted geometric attributes.

2.2 Implications for Graphical Matching Methods

In the graph theoretic approach, two scenes are matched by matching their graphical representations. Segmentation errors affect attribute values of nodes as well as the structure of the graph. Various effects that segmentation errors can have on graph representation of scenes are given below.

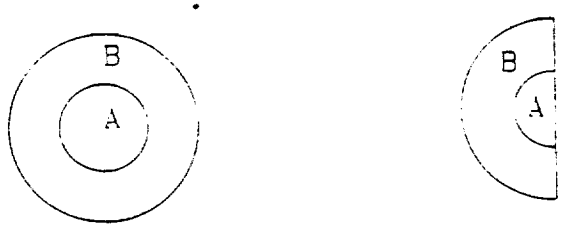
Mismeasured attributes: Since the segmentation algorithms will not find perfect region boundaries, an exact match between regional descriptions of an observed region and a stored object would be a rare occurrence. One must consider that a region may map to a particular object if its regional description is similar enough to that of the object. This implies that the selection of attributes to be used for describing regions is very important, since possible mappings should not be ruled out on the basis of attribute measures that are unreliable because of segmentation errors. As seen in Figures 2-1 and 2-3, a region's perimeter is an example of such an unreliable attribute.

Reversed relations: In the case of relations such as 'adjacent-to, 'left-of,' or 'above,' it may be possible to find exact matches. However, we have seen that segmentation problems may cause errors in relations as well as attributes. For example, positional relations such as 'left-of' or 'above' may be different in the segmented image because of rotation changes or mismeasured boundaries that cause region centroids to be shifted. If the relation 'left-of' is defined based on

horizontal pixel values of object and region centroids, a very slight rotation can cause the relation to be opposite in the observed scene. In other words, selection and definition of relations are critical steps. As far as possible, relations which are less sensitive to segmentation errors must be chosen. When possible, real-valued relations should be used instead of binary relations, so that when seeking a match between observed and stored scenes, one can look for *similarities* in relations rather than exact matches.

Missing objects and relations: Segmentation errors may change the structure of the graphical representation of the scene. When parts of a scene are not visible in the input image, or when there are occluded objects, the graphical representation will have missing nodes. Missing relations among objects result in missing arcs. An example is shown in Figure 2-4. In the stored scene, region B includes region A. Because the scene is only partially visible in the observed image, region B does not include region A.

Extra objects and relations: An unexpected object appearing in the scene corresponds to an extra node. If the extraneous region is not similar to any of the objects in the stored scene, it may be ignored. If it is similar to one or more objects, then the scene matching problem becomes more complicated. An extra relation between two objects in the observed scene corresponds to an extra arc. One common situation that could lead to this problem is an occluded object along with use of an intransitive left-of relation. This situation is shown in Figure 2-5. Here, object B is not visible in the observed scene, so the relation A left-of



Scene **Segmented regions**

Figure 2-4: Occlusion can cause a missing inclusion relation.

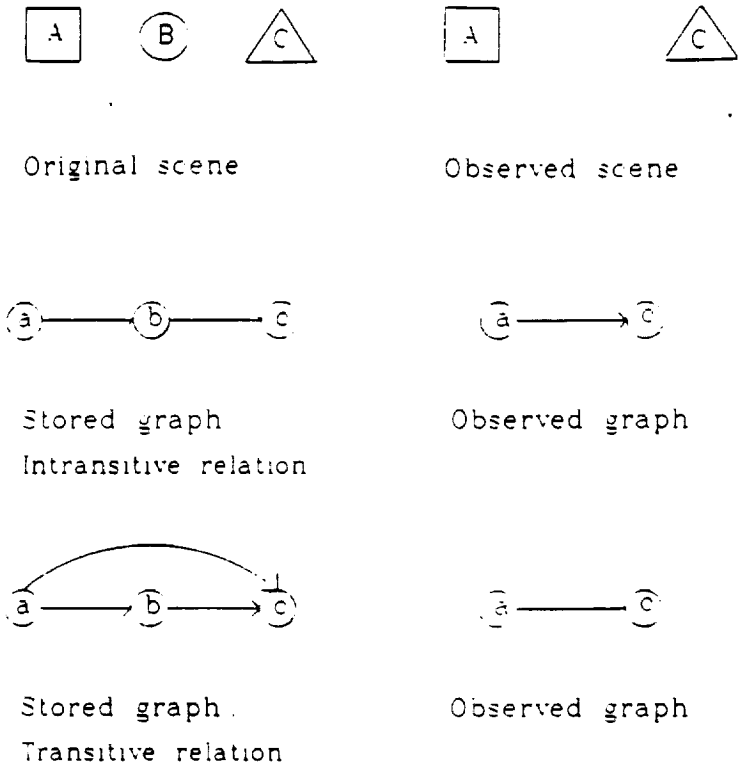


Figure 2-5: Intransitive relation causes extra arc (a,c) in observed graph.

C becomes an extra arc. This problem is eliminated if transitive relations are used. Of course with some relations such as adjacency (shared boundaries), a transitive relation cannot be used.

Split and merged objects: The presence of split or merged objects in the segmented image causes the structure of the observed graph to differ from that of the stored graph. If an object is split into two regions, the observed graph will contain two nodes corresponding to that object rather than one, and may have extra arcs representing the relations between the two nodes. If two objects are merged together, the opposite problem occurs.

In summary, missing nodes, extra nodes, missing arcs, and extra arcs are possible, due to segmentation errors. This may alter the structure of the graph representation of the input image. The graph theoretic matching method must be capable of dealing with the above structural changes. In the rest of this chapter, existing graph-based scene matching methods are presented and evaluated in view of segmentation errors.

2.3 Previous Research in Scene Representation and Matching

Graph-based scene matching methods can broadly be classified into two categories: vertex mapping matrix methods and association graph methods. Both approaches accomplish matching based on the principle of graph isomorphism. There are many terms having to do with graph isomorphism that should be defined, since different authors use these terms in different ways. Shapiro and Haralick provide definitions of many graph-theoretic terms, and some of them are described below [26].

Graph Homomorphism: A graph homomorphism from Graph G to Graph H is a mapping in which all vertices of G map to a subset of the vertices of H such that if Vertex a maps to a' and Vertex b maps to b' ($a' \neq b'$), then any relation that exists between a and b in G also exists between a' and b' in H. Homomorphism need not be a one-to-one mapping. Several vertices in G can collapse into a single node in H.

Graph monomorphism: A graph monomorphism is a homomorphism that is one to one. In other words, each node of G maps to a distinct node of H, while preserving the arc relations of G, although there may be extra nodes or arcs on H that have no counterparts in G. The term **subgraph isomorphism** seems to be a more common term for relational monomorphism, and it is the term used in this dissertation.

Vertex-induced subgraph isomorphism: A vertex-induced subgraph isomorphism is a special case of subgraph isomorphism. If there is a subgraph isomorphism from G to H, and if for all vertices a and b in G mapping to a' and b' in H, respectively, the relations between a and b in G match exactly with those between a' and b' in H, then the isomorphism is a vertex-induced subgraph isomorphism.

Graph isomorphism: A graph isomorphism is a mapping in which each vertex in G maps to a unique vertex in H, and each vertex in H is mapped to by a unique vertex of G. It is a perfect graphical match.

Finding an isomorphism between two graphs is an NP-complete problem, since the number of mappings to try for n nodes is O(n!). The computation is exactly proportional to n! only if all possible node mappings are tried. When the number of mappings can be restricted to a small number (based on graph properties), the problem can be solved in a reasonable amount of time.

2.3.1 Vertex Mapping Matrix Approach

One method for finding graph (or subgraph) isomorphisms involves setting up a matrix called the vertex mapping matrix. The rows of the matrix represent the vertices of the subgraph G, and the columns represent the vertices of the graph H. A value of 1 at position (x,y) in the matrix indicates that vertex x in G could map to vertex y in H. A basic algorithm for finding isomorphisms using the vertex mapping matrix approach is as follows:

1) The initial vertex mapping matrix M^0 is a binary matrix which is formed by comparing the in-degree and out-degree of each vertex in G with all vertices in H. $M^0_{x,y}$ is set to 1 iff the in-degree and out-degree of x in G are less than the in-degree and out-degree of y in H, respectively. Otherwise, $M^0_{x,y}$ is set to 0. If other constraints on nodes are known, they too may be used in narrowing down the possibilities in the vertex mapping matrix.

2) Some of the ones in M^0 are changed to 0 to obtain a matrix M' such that the following are true:

   a) There is exactly one 1 in each row of M'.

b) There is at most one 1 in each column of M'.

A given M^0 matrix may yield many matrices M' to satisfy the above conditions. These matrices may be found by a backtracking procedure.

3) For each M' found in (2), compute matrix C as

$$C = (M'(M'B)^T)^T. \tag{2-1}$$

Now, graph G is isomorphic to a subgraph of H, with the mapping given by M', iff, for all i and j, if $A_{ij} = 1$ then $C_{ij} = 1$. For vertex-induced subgraph isomorphism, C must be equal to A.

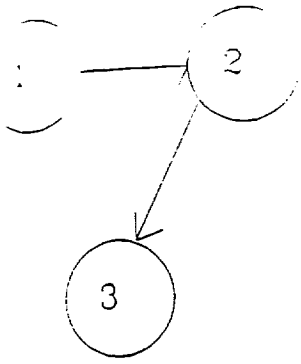
Example:

This example illustrates the vertex mapping matrix method for determining subgraph isomorphism. Two graphs G and H are shown in Figure 2-6. (Note that G is a subgraph of H.) Applying the above procedure to this problem, we obtain

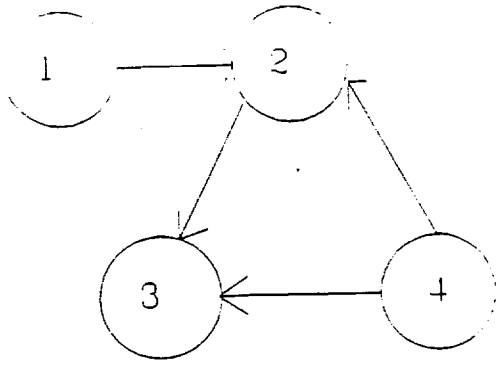
$$M^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 \\ 3 & 0 & 1 & 1 & 0 \end{array}.$$

The backtracking procedure will produce two possibilities for M':

$$M'_1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \end{array} \qquad M'_2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \end{array}.$$



Subgraph G



Graph H

	1	2	3
1	0	1	0
2	0	0	1
3	0	0	0

Adjacency Matrix A

	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

Adjacency Matrix B

Figure 2-6: Two graphs and their adjacency matrices.

Applying Equation 2-1 to these matrices gives:

$$C_1 = \begin{matrix} 0\,1\,0 \\ 0\,0\,1 \\ 0\,0\,0 \end{matrix}, \text{ and } \quad C_2 = \begin{matrix} 0\,1\,1 \\ 0\,0\,1 \\ 0\,0\,0 \end{matrix}.$$

Since $C_1 = A$, the mapping represented by M'_1 is a vertex-induced subgraph isomorphism (i.e. there are no missing arcs among vertices in the subgraph). The matrix C_2 satisfies the broader condition for subgraph isomorphism (if $A_{ij} = 1$ then $C_{ij} = 1$), in which there may be missing arcs in G.

Several modifications to this basic algorithm have been suggested in order to speed up the algorithm [3,6,16,29,30]. The modification proposed by Ullmann [29] consists of reducing the number of ones in M^0 before using the backtracking procedure. If Vertex a in G can map to a' in H, then $M^0_{a,a'}$ is 1. If b and c are neighbors of a in G, then they must map to some b' and c' in H which are neighbors of a'. If this is not true, then $M^0_{a,a'}$ is changed to 0. This ensures that every M' that can be obtained from M^0 is an isomorphism. There is no need to compute the matrix C to check for isomorphism. Ullmann suggests that this procedure be continued until no 1 in the vertex mapping matrix is set to 0 during a complete iteration.

Mittal [16] describes an algorithm for directed graph isomorphism in which properties other than in-degree and out-degree are used to reduce the number of mappings to be tried. He finds the distances between all pairs of vertices in each graph using Floyd's algorithm. The vertices of each graph are then partitioned into several classes such that vertices that are in the same class have the same in-degree and out-degree. Also, vertices in the same class

are all separated from each other by a fixed distance. A vertex from G can map to a vertex from H only if they belong to identical classes. Although many possibilities are eliminated, there may still be backtracking required to determine some remaining mappings. This approach is only for isomorphisms, not for subgraph isomorphisms or other imperfect matches between graphs. So, it is not appropriate for scene matching applications, in which exact isomorphism may be a rare occurrence.

2.3.2 Association Graph Method

Another method for finding graph isomorphisms is the association graph method. Rather than representing a possible mapping by a 1 in the vertex mapping matrix, it is represented by a vertex in the association graph. An arc between two vertices in the association graph indicates that the two mappings represented by the vertices are compatible. In order to find an isomorphism or any subgraph-to-subgraph mapping from the observed to the stored graph, a maximal clique (fully connected subgraph) in the association graph is sought.

In a scene matching application, the nodes represent region-object pairs. Arcs between nodes represent compatibilities between pairs of mappings. Nodes in the association graph exist if a region-object mapping is possible, based on similar local properties of the regions and objects. An arc exists if the relation between the two regions matches the relation between the two corresponding objects. In order to determine a mapping from observed scene to stored scene, the largest maximal clique in the association graph is found.

Example:

Figure 2-7 shows the association graph for the graphs shown in Figure 2-6. Each 1 in M° gives a vertex in the association graph. Two nodes (x, y) and (x', y') are connected by an arc if and only if the following condition holds:

If x is adjacent to x' then y is adjacent to y'.

For isomorphisms, the association graph cannot have arcs between nodes (x, y) and (x, y') or (x, y) and (x', y), since this implies 'collapsing' a pair of nodes in G or H into one node. If we wish to allow for homomorphisms, these arcs are permissible. The association graph in Figure 2-7 contains two largest maximal cliques: {(1,1), (2,2), (3,3)} and {(1,4), (2,2), (3,3)}. The first clique is a vertex induced isomorphism (the solution obtained from M'_1 in Section 2.3.1) and the second clique is a subgraph isomorphism (M'_2).

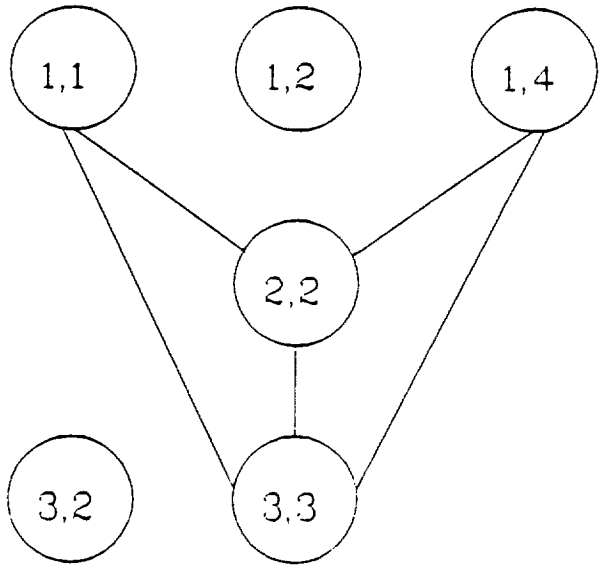


Figure 2-7: Association graph: cliques represent isomorphisms.

An algorithm which may be used for finding maximal cliques is given below:

1) Assume all vertices are in the same clique.

2) Check pairs of vertices; if there is no arc between them, then they must be in separate cliques. Split the potential clique into two parts; one with vertex x and not y, and the other with vertex y and not x. Each part contains all of the other vertices of the original potential clique.

3) Recursively perform (2) on each potential clique. When no more splitting is needed, all the remaining potential cliques are actual cliques. The maximal cliques are the ones that are not subsets of any other cliques.

Figure 2-8 is an illustration of this clique finding algorithm on the example of Figure 2-7. Each group shows the potential cliques at each iteration of the splitting process, separated by semicolons. Maximal cliques are marked by asterisks. The groups that are crossed out are duplicates of groups previously considered or are subsets of maximal cliques previously found.

Iteration 1:
(1,1)(1,2)(1,4)(2,2)(3,2)(3,3)

Iteration 2:
(1,1)(1,4)(2,2)(3,2)(3,3);
(1,2)(1,4)(2,2)(3,2)(3,3)

Iteration 3:
(1,1)(2,2)(3,2)(3,3); (1,4)(2,2)(3,2)(3,3);
(1,2)(2,2)(3,2)(3,3); ~~(1,4)(2,2)(3,2)(3,3)~~

Iteration 4:
*(1,1)(2,2)(3,3); (2,2)(3,2)(3,3);
*(1,4)(2,2)(3,3); ~~(2,2)(3,2)(3,3)~~;
(1,2)(3,2)(3,3); (2,2)(3,2)(3,3)

Iteration 5:
~~(2,2)(3,3)~~; (3,2)(3,3);
(1,2)(3,3); ~~(3,2)(3,3)~~;
~~(2,2)(3,3)~~; ~~(3,2)(3,3)~~

Iteration 6:
*(3,2); ~~(3,3)~~;
*(1,2); ~~(3,3)~~

Figure 2-8: Illustration of a parallel clique finding process.

2.4 Previous Research on Inexact Graph Theoretic Matching

The algorithms described in Section 2.3 are simply intended for matching graphs where exact graphical matches can be found. However, in scene matching applications, an exact match is a rare occurrence. Previous research on inexact scene matching using the graph theoretic approach is summarized in this section.

Tsai and Fu describe an error correcting subgraph matching algorithm [27,28]. In this approach, the observed or input scene is represented as an attributed graph (graph with weights on vertices and arcs). and matched with part of the stored graph by measuring the amount of distortion of vertices and arcs needed to obtain a match. The attributed graph may have missing or

mismeasured nodes and arcs. Missing nodes are due to missing objects, and mismeasured attributes are due to noise, lighting conditions, or geometric distortions. Similarly, missing nodes or deformed nodes may result in missing arcs. Noise may alter the weights associated with an arc. The assumption of their approach is that the pattern deformation probabilities or densities can be determined, so that the conditional probability $P(\omega \mid \omega')$, where ω' is the observed scene and ω is the stored scene, can be computed. In other words, deformation probability densities are used to obtain the maximum likelihood solution. In a problem such as character recognition, it is possible to find these probability densities using a sample set of characters. However, in scene matching, determining the probabilities of an object being missing or an attribute being mismeasured due to a segmentation error is difficult, since the segmentation errors are not predictable. The subgraph error correcting isomorphism approach of Tsai and Fu also allows only for one-to-one mappings. As a result, this approach cannot handle extraneous objects in the input scene or the problems caused by oversegmentation or undersegmentation. Only missing or mismeasured objects or relations are handled.

A similar approach is described by Wong and You [31]. Rather than starting with a prototype and a set of probabilities or densities for all possible deformations, stored scenes are represented as "random graphs." A random graph is a pair R = (W,B), where W is the vertex set and B is the arc set. Each element of W and B is a random variable. Rather than finding a match that maximizes a probability measure, they find a match that minimizes a measure of

entropy. This method is also suitable for applications such as character recognition, where sample patterns can be used to determine probability distributions of the random variables.

Haralick and Shapiro [25,26] describe another approach to inexact graph theoretic scene matching. They assume that the observed scene is a randomly altered version of the stored, prototype, scene. Weighting functions assign weights to vertices in the stored graph. The weight of a vertex indicates the importance or prominence of the corresponding object. Relations (arcs) also have weights assigned to them, to indicate their importance. Weights on vertices sum to 1, as do weights on arcs. An inexact mapping must meet the following conditions:

1) Each observed object must map to the corresponding stored object well enough. The quality of the mapping is determined by a difference measure, and its value must be less than a predefined threshold.

2) The sum of the weights of those stored objects that do not have any observed object mapping to them must be less than a threshold. This ensures that there are not too many important objects missing from the observed scene.

3) The sum of weights of missing relations (arcs) must be less than a threshold.

Haralick and Shapiro agree that defining a "best match" using all three conditions stated above is difficult. A perfect match with very few vertices is no good; neither is a sloppy match involving all the vertices.

Yang, Snyder, and Bilbro [32] describe the use of association graphs for finding inexact matches in scenes that have been oversegmented. They have modified the association graph method by allowing multiple regions to map to the same object. The approach described requires that after many-to-one mappings, the adjacency relations between regions in the observed scene and between corresponding objects in the stored scene match exactly. Adjacency is the only relation used by them.

2.5 Evaluation

An evaluation of the previous research in the area of inexact graph theoretic scene matching has led to the following conclusions:

1) The existing techniques do not explicitly allow for multiple relations to exist between two vertices. The use of several relations among the same set of vertices is sensible in scene matching, since there are many relations among objects in a scene.

2) Proper selection of attributes and relations is critical to the success of the graph theoretic approach. This problem becomes even more important when the segmentation process is not error-free.

3) None of the methods reported in the literature are capable of handling all the various problems caused by imperfect segmentation. Some methods can deal with missing vertices and arcs or with oversegmentation.

4) The probabilistic methods are suitable for problems such as automatic character recognition where various probability density functions can be

determined from sample data.

5) Current applications of association graphs are inadequate since they do not allow for any measures of merit indicating how good a region-to-object mapping is, or how compatible two mappings are. However, this method is more promising than the vertex mapping matrix method, since it allows for matching of common subgraphs, not just subgraph isomorphisms. This is important because a subgraph isomorphism cannot be found when both missing and extra objects occur in the observed scene.

Based on the evaluation presented in this section, a graph-based scene matching method which is capable of handling segmentation errors is developed in the remainder of this c **report.** Problems of attribute and relation selection, and graph representation of scenes, are presented in Chapter III. The improved scene matching method based on association graphs is described in Chapters IV through VI.

III. REPRESENTATION OF SCENES BY MULTIPLE GRAPHS

As mentioned in the previous chapter, the problems addressed in this chapter include graph representation of scenes, and selection of attributes and relations. In developing the method of scene description, we keep in mind that the scene matching process will be attempting to match observed regions to stored objects based on similarities in local attribute measurements. Comparing relations which exist between pairs of regions to the relations between the corresponding pairs of objects will allow the matching procedure to use the scene context to eliminate incorrect mappings.

Much of the existing research does not investigate the appropriateness of various types of relations and attributes to the problems peculiar to scene matching. We describe a set of attributes and relations that should be appropriate for many scene matching applications. The attributes and relations used are chosen to facilitate inexact scene matching by graphical matching techniques. The relations used are transitive when possible, so that an observed graph with missing vertices can be matched directly to a stored graph, by deleting only the missing vertices from the stored graph.

Attributes and relations should be chosen in a way that facilitates inexact matching. The matching should be fault-tolerant, but should be able to determine the correct match, not choosing a false match as the best.

3.1 Graph Representation

Over the years, several methods for representing scenes by graphs have been developed [8,18,19,26,28]. The approach used here is to model a scene by a set of graphs $\{G_1, G_2, \ldots, G_N\}$, all graphs defined on the same set of vertices $\{V_1, V_2, \ldots, V_M\}$. Each object or component in the scene is represented by a vertex. With vertex V_k, we associate an attribute vector $X(V_k) = [x_1(V_k)x_2(V_k)\ldots x_n(V_k)]$. The components of $X(V_k)$ are real-valued measurements taken on the object represented by V_k. For example, $x_1(V_k)$ may be the area and $x_2(V_k)$ may be the circularity measure of V_k. Each graph describes a particular relation among the vertices. For example, G_1 may describe the adjacency relation and G_2 may describe the reflectance relation among the scene objects.

The above approach is similar to the one used by Faugeras and Price [10]. The approach described in this section allows the use of real-valued as well as binary relations, whereas the Faugeras and Price approach allows only binary relations. Real-valued relations are relatively more fault tolerant and therefore suitable for inexact scene matching when the segmentation process is not perfect. The four rules given below are useful in constructing scene model graphs.

Rule 1: If a binary relation R is symmetric and V_i R V_j, draw a directed arc from V_i to V_j, and a directed arc from V_j to V_i. If V_i is not R-related to V_j, leave V_i and V_j unconnected. R is symmetric if and only if V_i R V_j means that V_j R V_i. For example, adjacency is a symmetric relation. An undirected graph may be used to describe a symmetric relation, but since undirected graphs cannot describe asymmetric relations, for the sake of

uniformity, directed graphs are used to specify all types of relations.

Rule 2: If the binary relation R is asymmetric and if it is known that V_i R V_k is always true, then include an arc from V_i to V_k. If there is a possibility of V_i R V_j or V_j R V_i, draw arcs from V_i to V_j and V_j to V_i. Rule 2 ensures that the graph of the observed image remains a subgraph of the model no matter what relation V_i and V_j have in the observed scene. An example would be the relation 'above.' If only slight rotation is expected, for many pairs of objects in the scene we can be sure of the observed 'above' relation. However, if the centroids of two objects have almost the same y coordinate, a very slight rotation can cause the relation 'above' to be reversed. This is shown in Figure 3-1.

Rule 3: Each binary relation must be explicitly shown even if the relation is transitive. If R is a transitive relation, and if V_i R V_j and V_j R V_k, then V_i R V_k. According to Rule 3, the relation V_i R V_k must be shown explicitly. This forces the graph of the observed image to remain a subgraph of the model even if V_j is missing in the observed image due to noise or improper segmentation. (This was illustrated in Figure 2-5.)

Rule 4: In case of a real-valued relation, there must be an arc between every pair of vertices. The absence of the relation between two vertices must be indicated by assigning zero weight to the arcs connecting them.

3.2 Attribute Selection

Primitives used in scene matching applications fall into two categories: regions and line or curve segments. Regions are useful primitives in applications

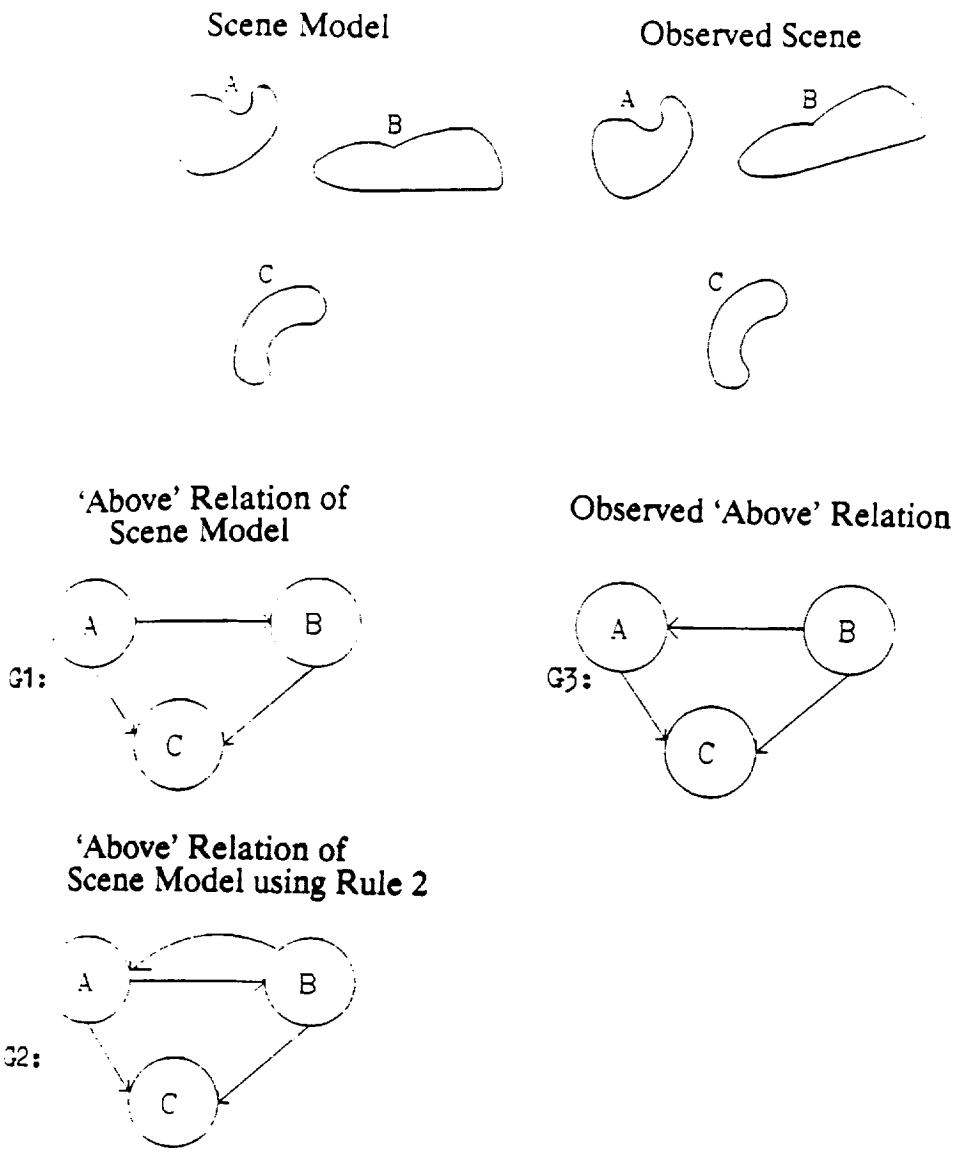


Figure 3-1: With Rule 2, observed graph G3 is a subgraph of stored graph G2.

such as aerial scenes, or when scenes consist of separated objects on a background. Line or curve segments are used in character recognition or in other applications in which scenes can be represented as line drawings. Both types of primitives may be used in the same system.

Attributes that can be used to reliably distinguish among the objects in a scene are useful in narrowing down the set of regions that can map to a particular object. However, there is a tradeoff involved, since the best attributes to use may also be the most difficult to measure because of high computation or special equipment needed. The characteristics of typical objects encountered in scenes, and the types of errors a given segmentation algorithm is prone to make, must be considered in selecting attributes. Any attempt to find one universal set of attributes useful for every application is futile.

Attributes can be classified into two basic types: region-wide attributes and geometric attributes. Region-wide attributes are averages over the entire area of a region, and do not depend on shape. Values of some geometric attributes depend on a region's size, and others depend on shape. Hence, some geometric attributes are sensitive to errors in the detection of a region's boundary. Some examples of each of these types of attributes are given below.

Region-wide Attributes

Intensity: This is the average brightness of the pixels contained within the object boundary. This attribute is sensitive to changes in illumination and sometimes to

changes in sensor position. Because it depends on the boundary, segmentation errors can cause errors in the intensity measure if the region is not of uniform intensity.

Texture: One measure of texture is the variance of the histogram of the region's intensity. As with intensity, boundary detection errors can cause errors in texture measure if the region does not have a uniform texture.

Color: When an image is obtained by a color camera, values for the amount of red, blue, and green at each pixel are available. The color of a region can be defined as a set of coefficients representing the average of the red, blue, and green values respectively over the area of the region.

Range: Range is the distance of the object from the sensor. A range finder is needed to obtain this measurement. It is useful in distinguishing objects from marks that occur on a background, as in the simulator panel scenes shown in Chapter II.

Region-wide attributes are often useful when segmentation errors are present, as they are less sensitive than geometric attributes to errors in the detection of an region's boundary.

Geometric Attributes

Attributes to Describe Curves: Some attributes useful when curve segments are the primitives being used are curve length, length of a line between the curve's

endpoints, total angle change from one end to the other, and symmetry [33]. The main difficulty in using curve segments is to determine where to break a complex curve into primitives. Ordinarily they are broken at inflection points. Noise and segmentation errors can cause discrepancies between observed curves and their stored counterparts.

Perimeter: This is the length of the region's boundary. Perimeter is especially susceptible to segmentation errors, but is invariant to translation and rotation.

Area: The number of pixels contained within a region's boundary is the area. Since it depends on boundary, this attribute is also sensitive to segmentation errors.

Circularity: The circularity of a region can be defined as $4\pi \times \text{area} / \text{perimeter}^2$. This gives a circularity measure of 1 to a circle, $\pi/4$ to a square, and smaller values for rectangles with increasingly uneven side lengths. This attribute is sensitive to boundary detection errors, but is invariant to scale, translation, and rotation.

Length, height: The length (height) of a region in number of pixels in the horizontal (vertical) direction can be useful. However, it is sensitive to changes in rotation.

Extent: The extent of an object can be defined as the product of its length and height. This measure is much less sensitive than the area measure to segmentation errors, although it is sensitive to rotation.

Elongation: The ratio of length to height can be called the elongation of the object. This simple indicator of shape is often useful in distinguishing among different objects. It is invariant to scale, although it is sensitive to rotation.

Minimum bounding rectangle: The area, length, and height of the smallest rectangle that can enclose the region are useful measures, less sensitive than area to segmentation errors, and not sensitive to rotation. However, it is harder to compute than the extent of the region.

Rectangularity: Rectangularity is the ratio of the region's area to the area of its minimum bounding rectangle.

Geometric attributes that depend on the measurement of perimeter are susceptible to segmentation errors. Those that depend on length and height measures are less susceptible to these errors.

The choice of attributes should be tolerant to noise and errors. A pattern vector for each object can be obtained, components of which are relatively insensitive to noise, scaling, and rotation. Differences between pattern vectors of observed regions and stored objects are used to determine which regions could map to which objects, based on local properties alone. If a region's pattern vector does not fall within the allowable distance to any stored object, the region is disregarded. If a region is a possible match for more than one object, the context of the graphical relationships among objects will be used to determine which object it actually represents.

3.3 Relation Selection

It is important to select relations so that inexact matches are facilitated. Relations may either be binary (i.e. the relation either exists or does not) or real-valued. The following sections describe sensible binary relations that may be used for scene descriptions, and real-valued counterparts, which provide an improvement over binary relations in the case of inexact segmentation.

3.3.1 Binary Relations

Relations may be classified into three groups: comparative relations, positional relations, and topological relations. The binary forms of these types of relations are described below.

Comparative Relations: Comparative relations are found by comparing values of attributes of two regions. Any attribute may instead be expressed as a comparative relation. For example, intensity, texture, area, and circularity attributes can be expressed as the relations brighter-than, more-textured-than, larger-than, and more-circular-than, respectively. An advantage of expressing these values as relations is that scaling is not necessary. However, if they are not expressed as attributes, they cannot be used to eliminate possibilities for region-object mappings.

Positional Relations: The relations left-of and above are commonly used to describe the relative positions of two regions [26,31]. However, these relations are seldom precisely defined. There are several possible definitions of left-of and above.

A could be considered left of B if:

1) all y coordinates in object A are less than all y coordinates in object B.
2) some y coordinate of A is less than all y coordinates of B.
3) there is a horizontal line passing through some pixel in A and some pixel in B such that the y coordinate of the A pixel is less than the y coordinate of the B pixel.
4) the y coordinate of the centroid of A is less than the y coordinate of the centroid of B.
5) the centroid of A lies within the left-of quadrant shown in Figure 3-2.

Some of these definitions are inherently transitive, such as the last two. If relations used in graphs to describe scenes are transitive, matching by subgraph isomorphism is facilitated, because the row and column of a missing vertex can be deleted from the stored graph's matrix, and the remaining matrix will be a perfect match. For example, in a left-of relation, if an intransitive relation is used and an intermediate vertex is missing, the observed graph will have an arc that is not present in the stored graph, so the observed graph is not a subgraph of the stored graph.

The use of the 'left-of' and 'above' relations allows a scene to be described in an inexact manner, allowing for matches even when a change in camera angle or a slight rotation occurs. The definition of 'left-of' based on Figure 3-2 allows for scenes that appear quite different to be represented by the same graphs. The fourth definition given above may be more pleasing intuitively, although it too allows for very different scenes to have the same representation. An advantage of the fifth definition is that we may use an

overlapping angle region in which both of the relations 'left-of' and 'above' hold. With this approach, the tolerance for rotation of the scene does not change as the distance between the objects changes.

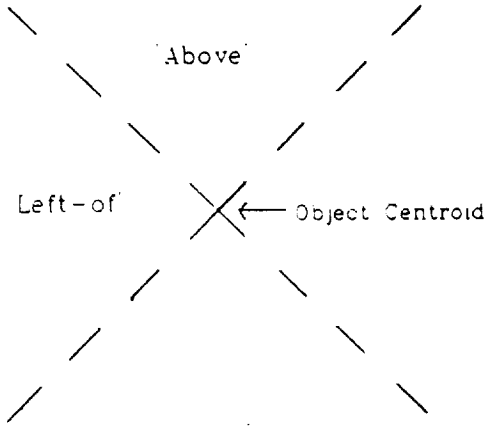


Figure 3-2: A possible definition of relations 'left-of' and 'above.'

A problem with the left-of and above relations is that they are sensitive to rotation. One way to deal with this problem is to allow for overlapping ranges for the two relations. In the stored scene, region A is specified as being left-of and above B. In the observed scene, one or the other will occur, as long as the rotation is not too large. This way, we have a subgraph isomorphism, because the observed scene is missing a relation, but will not have an extra relation. The overlapping of ranges is shown in Figure 3-3. Another useful approach to this problem is explained in the next section.

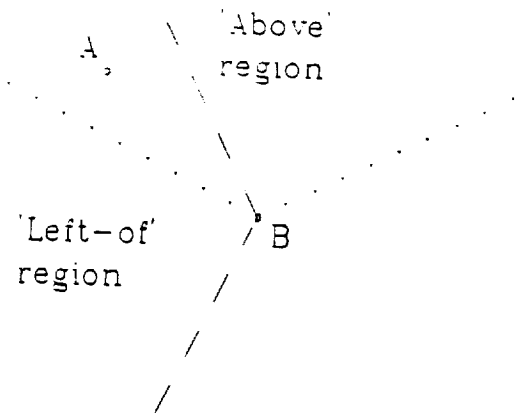


Figure 3-3: 'Left-of' and 'above' relations with overlapping domains.

Topological Relations: The relations of adjacency and inclusion are useful in scenes such as aerial photographs. A boundary pixel of a region is defined as a pixel which has a neighbor belonging to another region. Region A is adjacent to Region B if and only if there are one or more boundary pixels of Region A that have neighbors that are boundary pixels of Region B. Region A is included in Region B if and only if Region A is part of a composite region, S, such that all boundary pixels of S are neighbors of boundary pixels of B. Region A may be the only region in S.

3.3.2 Real-valued Relations

Whenever a binary relation is used, variations in the observed scene can cause the graph representations to vary widely. With binary relations, Region A is either adjacent to B or not; either above B or not; either brighter

than B or not. It is preferable to avoid these either-or relations, since a slight change in rotation or illumination, or a slight segmentation error, can cause these relations to be reversed. For relations that can be measured as real numbers, a real-valued relation may be used.

Figure 3-4(a) shows the disadvantage in an either-or decision about adjacency. Observed regions A and B match objects a and b, but because of a segmentation error, the observed regions are not adjacent to one another.

Figure 3-4(b) shows the same problem with the relation of inclusion. Because of a segmentation error, region B is not included in region A, even though A wraps around B to a large extent.

The difficulty with the left-of and above relations is shown in Figure 3-4(c). Whether the relations are defined in terms of four quadrants or by comparing y coordinate values of the centroids, a slight rotation can cause the binary relation to change.

In Figure 3-4(d), the problem with a binary relation brighter-than is shown. Due to shadows, glare, or changes in illumination, particularly with shiny surfaces, the brighter-than relation between two regions may be reversed.

The use of real-valued relations allows for a more accurate representation of the relations between objects, and a more accurate evaluation of how similar or different two relations are.

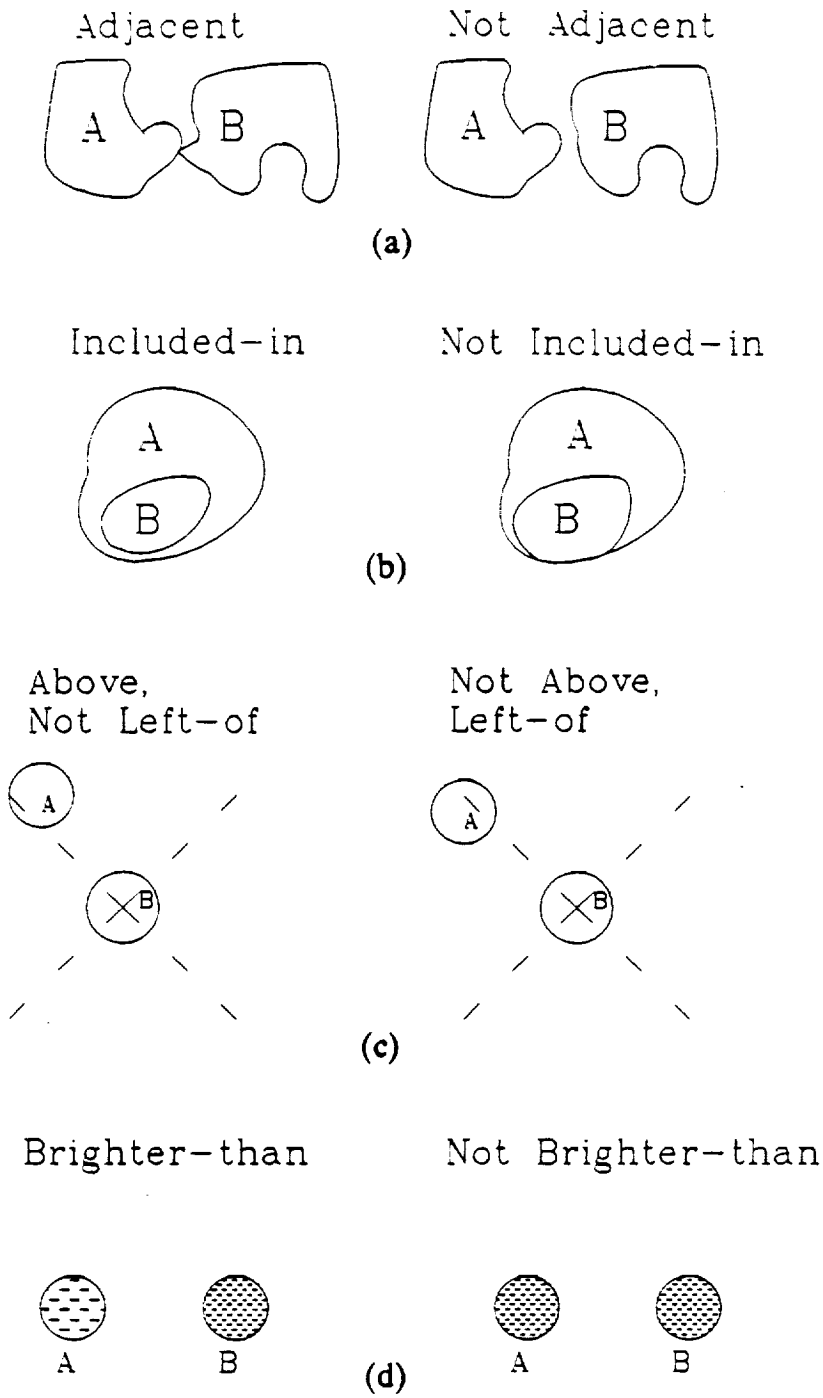


Figure 3-4: Disadvantages of using binary, 'either-or,' relations.

For example, if the adjacency relation is represented as two real-valued arcs, the percentage of each region's boundary that is shared with the other region can be the value of the arc. This is shown in Figure 3-5(a). If the stored relation A adjacent-to B has a value of 1%, and the observed relation A adjacent to B has a value of 0%, the difference is just 1%. This more accurately reflects the difference in adjacency relations between the observed and the stored scene.

This real-valued representation of adjacency as percentage of boundary shared eliminates the need for a separate representation of the inclusion relation. We define a boundary pixel of a region to be any pixel that borders on a pixel that is a part of a different region. By this definition, a region that includes another will have outside and inside boundary pixels. Whenever B adjacent-to A has a value of 100%, it implies that B is included in A, since 100% of its boundary pixels are adjacent to A. In Figure 3-5(b), object B adjacent-to A has a value of 100%, and region B adjacent-to A has a value of 85%, so that there is a difference of 15%, rather than a total mismatch as with the binary inclusion relation.

For representation of positional relations, the angle that a line between object centroids makes with the horizontal may be used. Then, in comparing the observed scene and the stored scene, the angle difference between a pair of regions and a pair of objects will be 0 if a perfect match, and 180 degrees if completely opposite. Figure 3-5(c) illustrates the advantage of using this

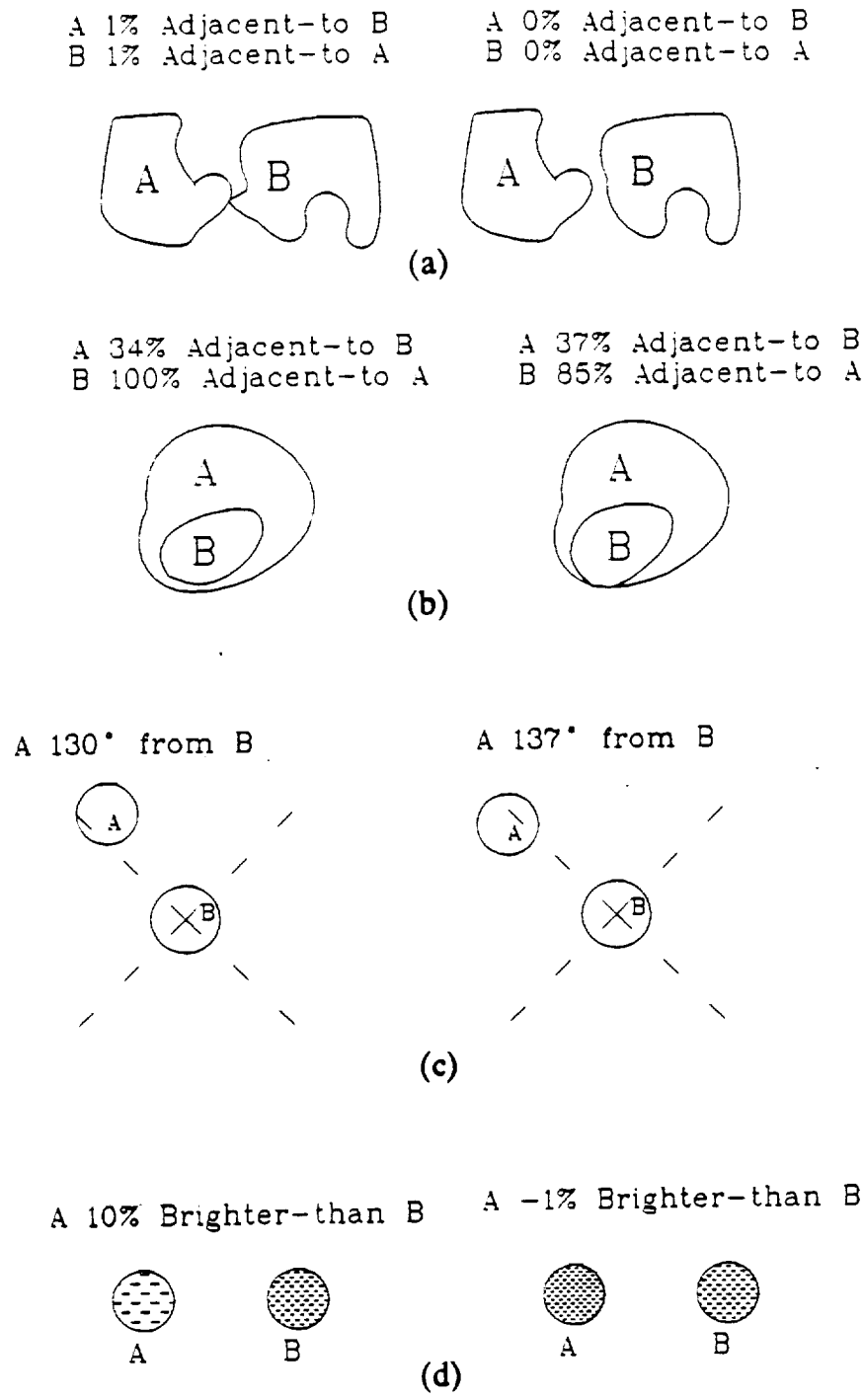


Figure 3-5: Real-valued relations allow similarity to be conveyed.

representation. A slight rotation changes the value of the angle relation from 130 degrees to 137 degrees, rather than changing the binary relations from above and not left-of, to not above and left-of.

Even with a real-valued relation to represent the positional relation between objects, a substantial amount of rotation will cause problems. A scene may be rotated in such a way that a false match of regions to objects may look correct because of the angle measures. So, if substantial rotation is expected, it is better to handle the problem by first finding certain distinguishing features that can be used to rotation-normalize the scene. If only slight rotation is expected, the comparison of angle measures will provide good results.

Another method for handling rotation is to express positional relations as real values, and then to try the matching algorithm on several versions of the scene at different rotations. It would then be expected that the best match would be with the scene that has been rotation-corrected.

For the brightness relation, and other relations between region attributes, as well, it is helpful to put a real value on the relation. For example, the value of the relation A brighter-than B can be expressed as the difference of the intensity of A and the intensity of B. In Figure 3-5(d), this difference is 20% for the stored scene and -1% for the observed scene, the discrepancy possibly due to a shadow over region A. So the difference between the two relations is 21% when a real-valued relation is used, as opposed to a reversal of relations when a binary brightness relation is used.

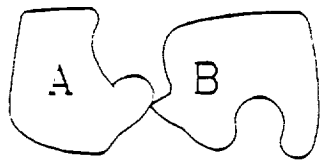
Another useful real-valued relation is distance. This relation is useful is discriminating between pairs of regions that are almost adjacent, when adjacency is described by the real-valued relation explained above, and pairs that are not even close together. For example, in Figure 3-6, the adjacency relations between regions A and B, and between regions A and C, are identical. However, the pair of mappings (RA,OA) and (RA,OB) is more compatible than the pair (RA,OA) and (RC,OB). The relation of distance helps in discriminating which pair of mappings is more compatible. The distance between a pair of stored objects and the distance between a pair of observed regions should be similar if the regions map to the objects.

Another possible way to define the distance relation, for scenes consisting of isolated blobs on a background, is that the distance from A to B is the minimum distance between any boundary pixel of A and any boundary pixel of B such that a line can be drawn between the two pixels without going through another regions. If there are no two pixels that can be joined without going through another region, define the distance as infinity, or the largest real value possible.

This distance measure should, of course, be scaled so that scenes scaled differently will still have similar distance measures between corresponding regions. One way to accomplish the scaling is to express the distance as some proportion of the perimeter, area, or extent of the two regions.

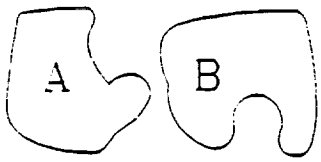
A 0 Distance-from B

A 1% Adjacent-to B
B 1% Adjacent-to A



A 1 Distance-from B

A 0% Adjacent-to B
B 0% Adjacent-to A



A 16 Distance-from C

A 0% Adjacent-to C
C 0% Adjacent-to A

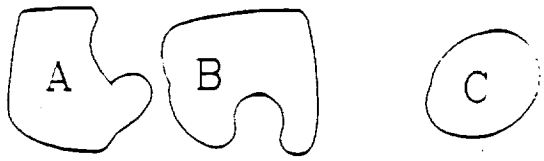


Figure 3-6: A useful real-valued relation: 'distance-from.'

3.3.3 Adaptation of Matching Methods for Scene Matching

The algorithms described in Chapter II were simply intended for the matching of graphs, and are not adapted to the specific problem of scene matching. These standard algorithms are designed to match two graphs, rather than two sets of graphs that would be used for description of scenes.

If the graphs have no weights associated with the arcs, the arc weight in each graph can arbitrarily be assigned to increasing powers of two. For example, arcs in the 'included-in' graph (matrix) may be labeled with '1's, those in the 'left-of' graph with '2's, and those in the 'above' graph with '4's. Then, the adjacency matrices for these graphs may be added together, with the resulting weighted adjacency matrix representing the set of graphs with no loss of information. Isomorphisms matching a set of observed graphs and a set of stored graphs can be found by matching these composite graphs which are obtained by adding the adjacency matrices together. The use of multiple graphs does not add to the complexity of the backtracking algorithm for subgraph isomorphism, although checking the condition for isomorphism between the A and C matrices requires a bit-by-bit comparison of arc weights.

The use of multiple graphs on the same vertex set can help to speed up the search for isomorphisms, by limiting the possibilities to try. In- and out-degree for each vertex in each graph is checked, as well as plausible object matches based on object attributes. Once a vertex mapping matrix is obtained, the problem of finding isomorphisms with multiple graphs on the same vertex set with unlabeled arcs is reduced to matching single graphs with weighted arcs, with no loss of information.

Inexact matching is limited to subgraph isomorphisms in the case of the vertex mapping matrix approach, and common subgraphs, with no splitting or merging of nodes, in the standard association graph method.

The vertex mapping matrix approach can be modified for application to scene matching as follows. Each scene is described by graphs on several relations; for examples left of and above. The observed objects are classified in terms of what object classes they might belong to. The class membership is determined by measurement of several attribute values. Measures of texture and circularity are useful. Allowable ranges for these measurements can be allowed to overlap for different object classes, so each object may be a possible member of more than one object class. This information is used to limit the possible mappings in the M^0 matrix. The relations left of and above are determined by the relationships between centroids of objects. In the stored representation, the ranges of left of and above are allowed to overlap so that there are extra relations in the stored representation. This helps in dealing with small rotations of the scene.

We assume the following to be available: list of stored objects and their types, matrices for several relations on the stored scene, acceptable measurement ranges for each object type, measurements of attributes of observed objects, matrices for the relations on the observed scene.

The scene matching process is as follows:

1) For each observed object, determine to which object types it might belong, based on acceptable measurements of attributes.

2) Form matrix M^c, in which $M^c_{i,j} = 1$ iff object i in the observed scene could be of the object class to which j in the stored scene belongs.

3) Form matrix M^d, the vertex mapping matrix based on degree of vertices. Check degrees for each graph in the set of graphs used. Use both in-degree and out-degree for the directed graphs. The observed node can map to a stored node if the in-degree of the observed node is no greater than the in-degree of the stored node, and the out-degree of the observed node is no greater than the out-degree of the stored node, for each of the graphs describing the scene.

4) Form matrix $M^0 = M^c$ AND M^d.

5) For the observed and stored graphs, obtain composite graphs A and B. For a set of graphs $\{G^0, G^1, \ldots, G^n\}$, set $G^k_{i,j} = 2^k$ iff $G^k_{i,j} = 1$, and set it to 0 otherwise. The composite graph G is the sum of all G^k. G has all the arc information from the set of graphs combined into one graph which can be used in the subgraph matching algorithm.

6) Use backtracking procedure to iterate through possible M' matrices. The Ullmann refinement may or may not be beneficial. If M^0 is very

sparse, the refinement procedure may not be necessary. Compute matrix C and compare with A. An exact match indicates that M' specifies a vertex-induced subgraph isomorphism of A to B.

If $C - A > [0]$, there may or may not be a subgraph isomorphism. For each $A_{i,j}$ and $C_{i,j}$, if for any corresponding pair of bits α and c, the condition ((NOT α) AND c) is true, the matrix C does not represent a subgraph of matrix A. If this condition is false for every corresponding pair of bits for each $A_{i,j}$ and $C_{i,j}$, then C is a subgraph of A.

The vertex mapping matrix approach lends itself best to problems of exact subgraph isomorphism, since various graph properties can be used to eliminate possible mappings. If we wish to allow the possibility of inexact matches between relations, the matrix M^d cannot be used in eliminating possible mappings. Also, occurrences of both extra and missing objects in the observed scene cannot be handled, since the observed scene would not be a subgraph of the stored scene.

Of the two approaches to finding graph isomorphism, the association graph method is better suited to finding inexact matches. The association graph method may also be modified to match two sets of graphs describing scenes. All possibilities for inexact matches can be taken into account. A missing object y will correspond to a situation in which the clique defining the mapping does not contain any node (X, y) for any region X. An extra region X corresponds to the clique not containing any node (X, y) for any object y. These possibilities are handled with no extra modifications to the association

graph method. Problems of oversegmentation and undersegmentation can also be managed by this method. The simplest way to allow for oversegmented regions, e.g. X_1 and X_2, is to say that both regions map to object y if (X_1, y) and (X_2, y) are both in the clique defining the mapping. Likewise, an undersegmented region, X, can map to two or more objects, e.g. y_1 and y_2, if (X, y_1) and (X, y_2) are in the clique. Another way to handle oversegmented or undersegmented regions is to include 'merge nodes' in the association graph. For example, if regions X and Y could both map to object x_1, a node $((X, Y), x_1)$ could be introduced.

The application of the association graph method to the problem of inexact scene matching is described in detail in Chapters IV, V, and VI.

IV. ASSOCIATION GRAPH METHOD IN INEXACT MATCHING

As shown in Chapter II, subgraph isomorphism approaches to scene matching can handle some cases of inexact matching: missing objects and missing relations. In realistic situations, the problems due to imperfect segmentation include extra objects, mismeasured relations, and split and merged objects. A modification of the association graph method for scene matching can provide a better way of dealing with these problems.

In this chapter, we first describe the shortcomings of the basic method of Yang, Snyder, and Bilbro [32], as discussed in Chapter II, and suggest an enhancement to the basic method, which provides a better way of dealing with the inexact matching problems brought about by segmentation errors. The three issues to be decided in order to implement the enhancement are outlined in Section 4.2. The issues brought up in this chapter are addressed in detail in Chapters V and VI, in which a new application of a relaxation algorithm to scene matching is developed.

4.1 Improvements

There are several areas in which the basic association graph approach can be improved. In the basic method, a region-to-object mapping is considered either possible or impossible. Nodes representing all the possible mappings are included in the association graph, but there is no measure of merit associated with the mappings. A value should be assigned to a node to indicate *how* good a mapping is. Likewise, two region-to-object mappings are considered either

compatible or incompatible, based only on equivalence of the binary adjacency relation between objects and between regions. In inexact matching problems, two mappings cannot be considered absolutely compatible or incompatible. As seen in Chapter II, it may not even be wise to consider two regions to be absolutely adjacent or not adjacent. Values should be assigned to relations as well.

Similarly, the evaluation of the largest maximal clique as being the best mapping does not allow for any gray areas in interpretation of compatibility [4,32]. A maximal clique may contain dubious region-object mappings which are just barely considered compatible. Again, the need for weights assigned to nodes and arcs is apparent.

Figure 4-1 shows an example of a basic association graph, and one with weights assigned to vertices and arcs. Any vertex (X,y) indicates the mapping of Region X in the observed scene to Object y in the stored scene. An arc connecting (X_1,y_1) and (X_2,y_2) indicates that these two mappings are compatible. The original method of finding a mapping from observed regions to stored objects was to use the mapping represented by the largest clique in the association graph. In the weighted association graph, the weight on a vertex (X,y) indicates how compatible the mapping of Region X to Object y is, based on attribute measurements of the region. The weight on an edge from (X_1,y_1) to (X_2,y_2) indicates how compatible the two mappings are to each other, based on similarities in the relations between X_1 and X_2, and y_1 and y_2. Based on these weights, it may be determined that a smaller clique, such as {(A,a), (B,b), (C,d)} in Figure 4-1, represents a better mapping than the largest clique.

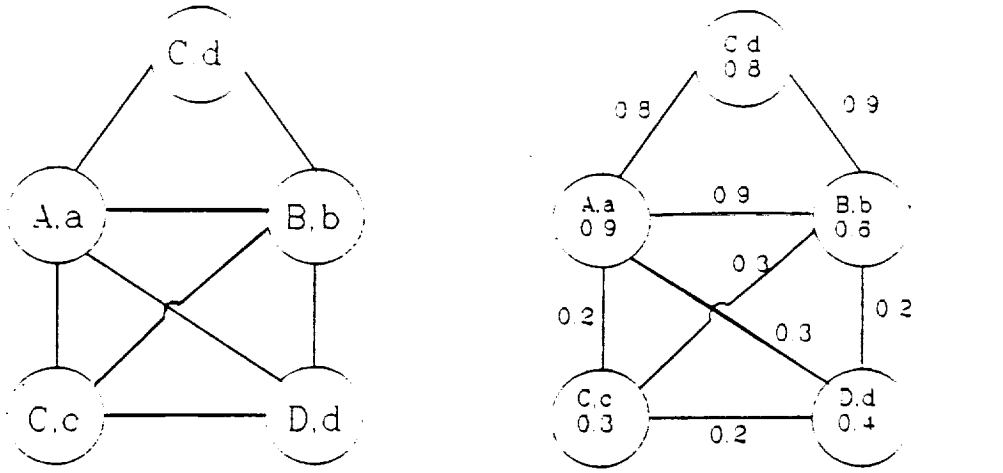


Figure 4-1: A weighted association graph is more useful for inexact scene matching.

The approach used by Yang was designed to handle the problem of oversegmented images, i.e. images in which more than one region may map to a single object. The occurrences of extra regions and missing regions are also handled implicitly by this method. However, the problem of undersegmented regions is not addressed, the assumption being that a segmentation algorithm can be 'tuned' so that either oversegmentation or undersegmentation occurs, but not both. In practice, we may have cases of both problems in the same scene with the same tuning of the segmentation algorithm.

A variation of this basic approach can handle problems of missing and extra objects, and oversegmentation and undersegmentation. Several alternative

methods of dealing with oversegmentation and undersegmentation are explored in Chapter VI. To see that a variation of this approach can handle all of these problems, we can simply look at the result that can be obtained. A clique representing a best mapping can be missing some of the regions of the observed scene, or some of the objects of the stored scene. It can contain two or more nodes that map one region to different objects, or nodes that map more than one region to one object. Alternatively, oversegmentation and undersegmentation may be handled by adding nodes representing multiple mappings to the association graph. The flexibility of the association graph method provides a clear advantage over other graph-based approaches to scene matching.

4.2 Problems to be Addressed

The main problems that need to be addressed in order to extend the method to handle these issues are: 1) the determination of weights for nodes, 2) the determination of weights for arcs, and 3) the evaluation of the result, or finding the 'best' clique in the association graph.

The weights for nodes should represent the closeness of the mapping of a given region to an object. This can be determined by some measure of how similar the region and the object are, based on local properties. The arc weights represent compatibility between two mappings, so they must reflect similarities between region-to-region and object-to-object relations. If we assume there are no split or merged regions, the determination of these weights is fairly straightforward. But, if there is a possibility of split or merged regions, determining the structure of the association graph and the node and arc weights is a more difficult problem. If we wish to maintain the idea of each node

representing the mapping of one region to one object, we must determine how to find weights for arcs between nodes representing the same region mapping to two different objects, or the same object mapping to two different regions. If we abandon the idea of each node representing one object and one region, we need a method of determining node weights on nodes that represent more than one object (region) mapping to one region (object), and arc weights for compatibilities of these nodes with others. The determination of weights for problems without split or merged regions is discussed in Chapter V. The case of split or merged regions is covered in Chapter VI.

The evaluation of the resulting weighted association graph is the most difficult problem. First, we must find cliques that represent candidates for the best mapping. In a large graph, this can be a computationally demanding task. Next, we must evaluate the merits of the cliques found, to determine which is the best. Not only the node weights representing region-object compatibility, but also the arc weights representing compatibility between mappings, must be considered in evaluating the merit of a clique. Do we include a node with a high weight in a clique if all arcs connecting it to the rest of the clique have relatively low weights? Do we include a node with low weight if its arcs have high weights? How do we compare the relative merits of two different weighted cliques, containing different numbers of nodes?

One approach is suggested by Davis [7], who has used association graphs in a different application, boundary matching. His approach is first to use a

discrete relaxation process to prune the association graph to decrease the number of cliques to be found, and then to evaluate the cliques by the following cost formula:

$$C(F) = \sum_{i=1}^{m} M(i, F(i)) + \sum_{i=1}^{m} \sum_{j=1}^{m} S_{ij}(F(i), F(j)) + P(m_T) + P(m_O), \tag{5-1}$$

where $M(i, F(i))$ measures the dissimilarity of object i and region $F(i)$, and $S_{ij}(F(i), F(j))$ is described as "the tension in the spring connecting" i and j if they are represented by $F(i)$ and $F(j)$ in the observed scene. So, this approach assigns costs to nodes and arcs of the association graph, rather than merits. $P(m_T)$ is a penalty for stored objects which are left out of the clique (implying they were not observed in the scene), and $P(m_O)$ is a penalty for observed primitives which are left out (or which do not map to any stored primitives under the mapping represented by the clique). A problem with this approach is that determining the values of these penalties is difficult. Another problem is that every clique, including those that are not maximal, will need to be evaluated to determine the mapping of lowest cost.

It is apparent that the main drawback in using this approach is the difficulty in finding the 'best' mapping by evaluating the cliques found in the association graph. The approach used by Davis attacks the problem by using discrete relaxation to reduce the number of cliques to be found, but his method still entails a complicated evaluation function.

Since the ultimate goal is to decide which region-to-object mappings are correct, it seems that a reasonable goal is to determine a method for incorporating contextual information into the node weights. Node weights could be altered based on weights of arcs and neighboring nodes, so that cliques may be evaluated using only the new node weights, ignoring arcs. The goal should be to decrease the number of cliques to be tried, and also to simplify the measurement of the merit of a clique. The next chapter describes relaxation algorithms, a class of algorithms which will facilitate these goals.

V. RELAXATION ALGORITHM APPLIED TO ASSOCIATION GRAPHS

In general, the classification problem deals with the assignment of a given unit to one of several predefined classes. Relaxation, an iterative parallel approach, has been successfully used to improve classification performance [11,14,23,24]. Let $A_1, A_2, \ldots, A_n$ be a set of n units, and $C_1, C_2, \ldots, C_m$ be the m classes. The relaxation approach assumes that each A_i can map to any of the m classes, and assigns an initial probability for each possible mapping. Therefore, A_i has m probabilities associated with it. Then, at each iteration, these probabilities are updated based on contextual information. This causes one of the probabilities to approach one and others to approach zero. The unit A_i is then assigned to the class determined by the highest probability.

In this chapter, the relaxation approach is combined with the association graph method to obtain a better graph theoretic approach for scene matching. Relaxation algorithms used for classification, in their original forms, are not suitable for scene matching, so significant modification is necessary.

Various types of relaxation algorithms: probabilistic, discrete, and fuzzy, are described in Section 5.1. Section 5.2 describes the application of relaxation algorithms to the scene matching problem. Selection of initial node and arc weights, updating rule, and condition for termination are presented, as well as the interpretation of the final result obtained from the relaxation process. Simulation results which support the theoretical concepts developed are given in Sections 5.3

and 5.4.

5.1 Relaxation Algorithms

As stated earlier, relaxation algorithms are used when each of a set of units $A_1, \ldots, A_n$ is to be assigned to one of the classes $C_1, \ldots, C_m$. In this approach, unit A_i is assigned m initial probabilities $p_{i1}, p_{i2}, \ldots, p_{im}$ where p_{ij} is the probability that A_i belongs to C_j. The assignment of unit A_i to C_j has some degree of compatibility or incompatibility to the assignment of unit A_h to C_k. This compatibility value is denoted by $c(i, j; h, k)$. The probabilities associated with each unit are updated iteratively by taking into account compatibility values and the probabilities of neighboring units. The goal is that the final probabilities obtained will favor one mapping above all others for each unit, and that this mapping will be the correct one. The problems to be addressed in selecting a relaxation algorithm are the computation of initial probabilities, computation of compatibility values, selection of updating rule, and terminating condition.

In the scene matching application, the 'set of units' corresponds to the set of regions in the observed image. The 'classes' correspond to the set of objects in the stored representation. In the discussion below which is specific to scene matching, the terms 'region' and 'object' are used rather than 'unit' and 'class.' The goal of the relaxation process as applied to scene matching differs from the goal as used in applications such as image segmentation. In segmentation, each unit (pixel) can map to one and only one class. Every unit must map to a class. However, in scene matching a region can map to more than one object. Several regions can map to a single object. There can be objects that have no regions

mapping to them and regions that map to no objects. Therefore it is necessary to investigate each relaxation approach to determine its suitability for the inexact scene matching purpose.

Relaxation algorithms can be classified into three basic categories: probabilistic, fuzzy, and discrete. These are described below, and their applicability to scene matching is discussed.

5.1.1 Probabilistic Relaxation

In probabilistic relaxation, with each object A_i, a probability vector $(p_{i1}, p_{i2}, \ldots, p_{im})$ is associated, where p_{ij} is an estimate of the probability that A_i belongs to C_j. If A_h is a neighbor of A_i and p_{hk} is the probability that $A_h \in C_k$, then $c(i, j; h, k)$ indicates the compatibility between the mappings $A_i \in C_j$ and $A_h \in C_k$. Its value is in the range [-1, 1] where -1 indicates total incompatibility, +1 indicates total compatibility, and 0 indicates irrelevancy ("don't care").

Given the initial probabilities and compatibilities between various mappings, one can adjust the probabilities associated with each unit on the basis of contextual information embedded in the probabilities and compatibility coefficients associated with its neighbors. The adjustment process satisfies the following properties:

a) If p_{hk} is high and $c(i, j; h, k)$ is close to 1, p_{ij} should be increased.

b) If p_{hk} is high and $c(i, j; h, k)$ is close to -1, p_{ij} should be decreased.

c) If p_{hk} is low or $c(i, j; h, k)$ is close to 0, then p_{ij} should not change

significantly.

An updating rule suggested by Rosenfeld which has the above properties is given by Equations 5-1 and 5-2 [24]:

$$p_{ij}^{(r+1)} = \frac{p_{ij}^{(r)}(1+q_{ij}^{(r)})}{\sum_{j=1}^{m} p_{ij}^{(r)}(1+q_{ij}^{(r)})} \quad , \tag{5-1}$$

where

$$q_{ij}^{(r)} = \frac{1}{n-1} \sum_{\substack{h=1 \\ h \neq i}}^{n} \left(\sum_{k=1}^{m} c(i,j;h,k)\, p_{hk}^{(r)} \right). \tag{5-2}$$

In Equation 5-2, the products $c(i,j;h,k) \times p_{hk}$ are averaged for each neighbor of node n_{ij}. This average is in the range [-1,1]. In Equation 5-1, the new value for p_{ij} is taken as the old value times 1 plus this average, so that positive averages will increase p_{ij} and negative averages will decrease it. This value is then normalized so that the sum over different classes j for p_{ij} is 1. The iterative procedure is continued until $\sum_{i=1}^{n} \sum_{j=1}^{m} | p_{ij}^{(r+1)} - p_{ij}^{(r)} | < \delta$, where δ is a pre-specified small number.

There are some limitations in applying this approach to scene matching problems. The assumption that each unit maps to one and only one class is not generally valid. One region of the observed image may map to none of the stored objects. It is possible to overcome this limitation by allowing for the existence of a 'null' object to which regions have some probability of mapping. If the node representing the mapping of a region to the 'null' object

ends up with the highest probability for the region, it can be assumed that the region did not map to any of the stored objects. Determining the initial probability that a region will map to 'null' is also problematic.

A region may also map to more than one stored object, in the case of an undersegmented scene. The region may be two or more objects merged together. In this case, a probabilistic relaxation algorithm is not expected to yield satisfactory results.

5.1.2 Fuzzy Relaxation

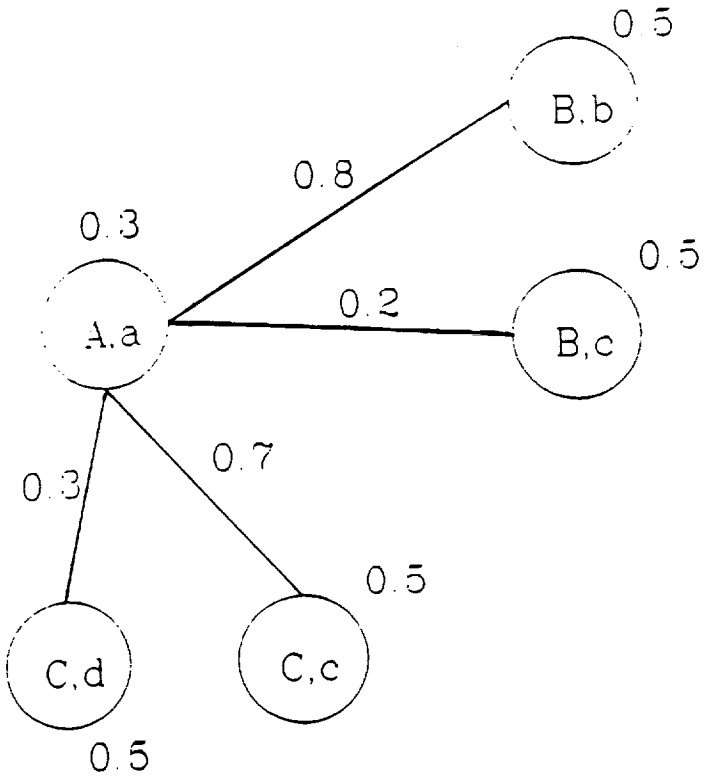
In another approach to relaxation, fuzzy relaxation, for each possible mapping (i.e. $A_i \in C_j$ for all i and j) a weight is assigned. This weight is simply a measure of the closeness of match between unit A_i and class C_j. Weights will all be between 0 and 1, but weights for a given unit need not sum to 1.

The values on arcs, $c(i, j; h, k)$, are values in the range [0,1] indicating the compatibility of mapping unit i to class j with mapping unit h to class k. $c(i, j; i, k)$ is defined as 1 if $j = k$, and 0 otherwise.

During iteration $(r + 1)$, weight $p_{ij}^{(r)}$ is updated by the following rule [24]:

$$p_{ij}^{(r+1)} = \frac{1}{n} \sum_{h=1}^{n} \left[\max_{k=1}^{m} c(i, j; h, k)\, p_{hk}^{(r)} \right]. \tag{5-3}$$

Figure 5-1 shows an example of this updating rule applied in one iteration on node (A, a).



$$p^{1}_{Aa} = [0.3 + 0.8 \times 0.5 - 0.7 \times 0.5] / 3 = 0.35$$

Figure 5-1: Rosenfeld's fuzzy updating rule applied to node (A,a).

The fuzzy relaxation algorithm causes some of the weights to decrease faster than others. After a few iterations, there is no longer a change in the order of probabilities relative to one another. The nodes with highest probabilities are the preferred mappings.

This updating rule displays an undesirable characteristic, as shown in Figure 5-2. In this instance, the initial node weights actually have no effect on the final result. The values of both nodes are driven to 0.5, if the node weights are normalized to add to 1 at each iteration. In general, if all arc weights are the same, the node weights will all stabilize at equal values, regardless of the initial weights. This is not a desirable characteristic in the scene matching application. If all arc weights are the same in an association graph (i.e. the graph is fully connected with all arcs equal), the node weights should not all become identical, as this eliminates all the influence of the initial node weight.

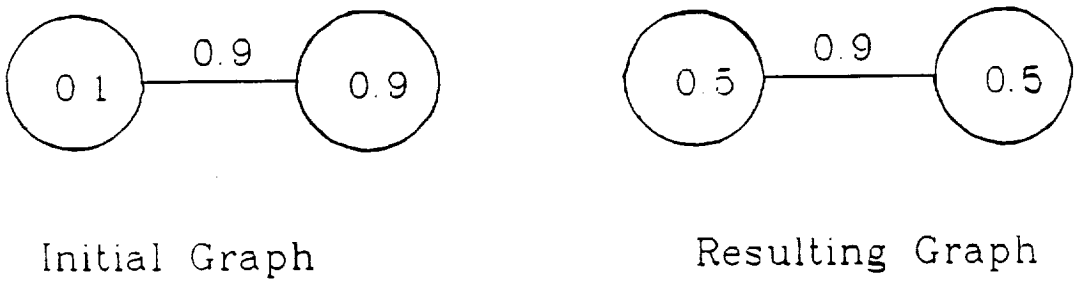


Figure 5-2: Rosenfeld's rule applied to a graph with all arc weights equal.

In scene matching, the fuzzy relaxation approach makes sense, since it allows for mapping a region to none of the stored objects, if all of a region's 'probabilities' are low, or to several stored objects, if several of its 'probabilities' are sufficiently high.

When allowing for mapping of several regions to one object (oversegmentation), there are several problems to be addressed. First, how do we determine the arc weights joining two nodes that map different regions to the same object? Once this is resolved, the problem is to determine if certain nodes should be included in the clique representing the best mapping.

5.1.3 Discrete Relaxation

In discrete relaxation, an assignment of a unit to a class is considered either possible or impossible. Rather than updating the weights on nodes, the relaxation process deletes nodes that are considered impossible, based on supporting values of neighboring nodes.

A usual formulation of the discrete relaxation algorithm is to assign initial probabilities and compatibility values to 1 or 0. The following updating rule, which can also be used with real values for fuzzy relaxation, is used [24]:

$$p_{ij}^{(r+1)} = \min_{h=1}^{n} \left[\max_{k=1}^{m} c(i,j;h,k)\, p_{hk}^{(r)} \right]. \tag{5-4}$$

An approach by Davis [7] (described in more detail in Section 5.2) uses a different idea of "discrete" relaxation. He assigns node and compatibility

values between 0 and 1, to represent degrees of consistency and compatibility. The initial node weights are the values p_{ij} of the local evaluation function that describes the closeness of the observed region A_i to the stored object C_j. Then, a node is deleted if the support for that node,

$$E_{ij} = \sum_{h=1}^{n} \sum_{k=1}^{m} p_{hk}, h \neq i, k \neq j \qquad (5\text{-}5)$$

is less than some threshold. As nodes are deleted, the support for other nodes decreases, and more nodes may be deleted. The process terminates when no nodes are deleted during an iteration. In this process, the node weights are not updated, but values of neighboring nodes are simply used to determine if a node is kept or discarded.

5.2 Application of Relaxation to Scene Matching

In this section, we examine the application of relaxation techniques to the scene matching problem. Whenever relaxation is used, the questions to be resolved are the determination of initial node and arc weights, updating rule, and condition for termination of the algorithm.

In scene matching, we must also determine how to use the result of the relaxation algorithm. Initially, in using weighted association graphs for scene matching, the problem is one of how to select which nodes are included in the clique representing the best match. The nodes included should have high weights, and the arcs connecting those nodes to others should also have high weights. Our goal is to have the relaxation algorithm simplify the selection of the

clique of the association graph that represents the 'best' match by choosing nodes with values above some threshold for inclusion in the clique, without having to consider arc weights.

There have been some applications of relaxation to scene matching problems discussed in the literature. Davis [7] describes a procedure for shape matching using relaxation techniques. In his work, observed angles and template angles of simple closed curves are to be matched. The process used is discrete relaxation, meaning that the goal of the relaxation process is to delete nodes from the association graph, not to change weights and leave all nodes in place. A unique feature of the algorithm presented by Davis is a second step performed after the relaxation on the nodes. A 'line graph' is created, in which the arcs of the original association graph become nodes of the line graph, and nodes of the original become arcs. The same discrete relaxation process is applied to the line graph. Nodes (lines) are deleted, which also reduces the number of cliques in the association graph. The result is a pruned association graph with many of the less likely mappings eliminated. The use of relaxation in this way does not simplify the evaluation of the merit of remaining cliques in the association graph.

Faugeras, Berthod, and Price [9,10,20,21] describe the use of a probabilistic relaxation process in scene matching of aerial images. They consider the observed regions to be the classes of the relaxation algorithm, and the stored objects to be the units. The goal is then to find an observed region that corresponds to each of the stored objects. This handles the problem of extra regions in the image that do not correspond to stored objects. However, each

stored object is assumed to have an observed region in the image that corresponds to it. But as we have seen, missing objects are a common segmentation problem.

5.2.1 Type of Relaxation Algorithm to Use

Probabilistic relaxation has several shortcomings when applied to scene matching. First, there is the assumption that each region must map to one and only one object. That is the basis of assigning node weights representing probabilities. Probabilistic relaxation updates these probabilities, and the result is that the node with highest probability for each region is considered the correct mapping. In order to allow for mapping of a region to none of the stored objects, we must use a 'null' object to which a region may map. Determining the probability that a region maps to this 'null' object as opposed to any existing object is a puzzling problem. For a region to map to more than one object, the best we can do with this approach is to say that if several nodes for a region have similar probabilities which are higher than those of other nodes, we may consider that the region maps to all of the objects represented by those nodes. For example, a region may have nodes with weights of 0.45, 0.45, and 0.1. We may interpret this result as indicating that the first two nodes represent correct mappings of this region to two different objects. A problem with this approach is apparent when we attempt to interpret the final result of the relaxation. We cannot simply favor nodes with the highest values if we are to allow for mapping of a region to multiple objects, since at least one of the probabilities in those cases must be less than 0.5.

Fuzzy relaxation seems to be the most appropriate for use in scene matching. It requires none of the work-arounds needed in probabilistic relaxation. The mapping of one region to multiple objects, or to no object, can be handled with some modification to the algorithm. In the final result, we may interpret nodes of highest value to be the best mappings.

5.2.2 Initial Node Weights

The initial node weights could be set to 1 or 0, indicating whether it is plausible that a region could map to an object based on similarities between attribute measurements. A better method is to determine the weight by a measure of similarity between the region and object. Faugeras and Price [10] describe a sensible approach. They compute a difference rating, $R(u,n)$, where u is the stored representation of an object and n is the observed region.

$$R(u,n) = \sum_{k=1}^{m} |V_{uk} - V_{nk}| W_k S_k, \tag{5-6}$$

where m is the number of attributes that have been measured (color, texture, etc.), V_{uk} and V_{nk} are attribute values. W_k is the feature's weighting (a scaling factor based on the size of the value), and S_k is the feature's strength, or importance. Then, the values are transformed to the range [0,1] by setting $f(u,n) = 1/(R(u,n)+1)$.

W_k was used by Faugeras and Price to handle differences in the scales of various attributes; for example, if the observed region has area 550 and the stored region has area 500, this is a difference of 50 pixels, which is small. But if two angles are compared, a difference of 50 degrees is large.

Scaling the attribute values of the regions before calculating $R(u,n)$ may improve the results. The scaling factor can be determined by comparing the average value of the feature among the stored objects with the average value among the observed regions. This would allow the use of features such as area and intensity, which may have an average value in the observed scene that is higher or lower than expected. So, $f(u,n)$ may be determined as follows:

$$V'_{nk} = \frac{V_{nk}}{\frac{\sum_{u=1}^{q} V_{uk}}{q}} \Big/ \frac{\sum_{n=1}^{p} V_{nk}}{p} \tag{5-7}$$

$$R(u,n) = \sum_{k=1}^{m} |V_{uk} - V'_{nk}| W_k S_k \tag{5-8}$$

$$f(u,n) = \frac{1}{1 + R(u,n)} \tag{5-9}$$

Perfect matches receive a value of 1, and increasingly bad matches have values approaching 0. Weights below some threshold may be set to 0, to reduce the number of possibilities to be tried.

5.2.3 Initial Arc Weights

The value of $c(i,j;h,k)$ is based on similarity of relations between regions A_i and A_h, and objects C_j and C_k. In each graph, if a relation exists between the regions and also between the objects, the value of $c(i,j;h,k)$ is increased. If all relations match, $c(i,j;h,k)$ is 1. If none match, the value is 0.

If real-valued relations are used, arc weights may be determined in the same way as node weights: higher differences in the values of relations lead to lower arc weights.

The arc weights between nodes representing the same region (undersegmentation) or the same object (oversegmentation) must be determined differently.

5.2.4 Updating Rule

A fuzzy relaxation algorithm was run using several example scenes. The first updating rule tried was

$$p_{ij}^{(r+1)} = \frac{1}{n} \sum_{h=1}^{n} \left[\max_{k=1}^{m} c(i,j;h,k)\, p_{hk}^{(r)} \right] \tag{5-10}$$

A scaling step,

$$p_{ij}^{(r+1)} = \frac{p_{ij}^{(r+1)} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{(0)}}{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{(r+1)}} \tag{5-11}$$

is performed, so that it is possible to use the condition of little change between weights at two consecutive iterations as the terminating condition. With this scaling step, the node weights add to the same sum after each iteration, which makes clear which nodes are increasing and which are decreasing.

The rule seemed appropriate for scene matching applications, since weak mappings for other regions should not adversely affect a mapping. Only

the strongest mappings for each region are used to update the probability of a given region-to-object mapping.

However, experimentation showed that with this updating rule, the initial node weights are irrelevant. The averaging of weights in the updating rule eliminates any advantage a node has because of a high initial weight.

A modified rule was tried:

$$p_{ij}^{(r+1)} = \alpha p_{ij}^{(0)} + \frac{1}{n}\sum_{h=1}^{n}\left[\max_{k=1}^{m}\left(p_{hk}^{(r)} c(i,j;h,k)\right)\right]. \qquad (5\text{-}12)$$

This was also followed by a scaling step. Here, the initial node weight influences the value at every iteration, as a sort of 'bonus' for good matches, with α determining the importance of the initial weight. This approach was taken rather than including $p_{ij}^{(0)}$ in the average, so that the importance of the initial weight would not depend on the number of regions.

It is instructive to examine the behavior of the updating rules on a very simple example. Figure 5-3 shows the results of applying Equations 5-10 and 5-11 ($\alpha = 0$), and Equations 5-12 and 5-11 ($\alpha = 0.5$), on an association graph consisting of just two nodes. The node weights are 0.1 and 0.9, and the arc weight is set at 1, 0.1, and 0. At $\alpha = 0$, the initial node weights have no effect on the final outcome. For increasing values of α, the initial node weights have increasing importance in the final result. If the arc weight is zero, the nodes do not affect each other, regardless of the value of α.

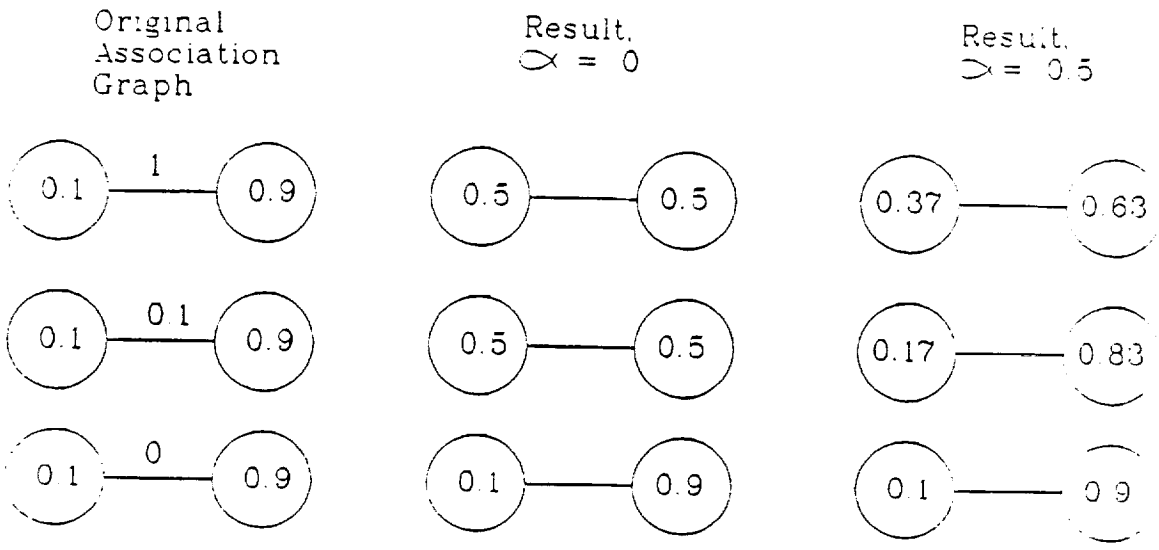


Figure 5-3: Positive α allows initial node weights to affect result.

Another simple example demonstrates how the value of α can affect the order of node weights in the final result. Figure 5-4(a) shows the initial association graphs, which have values on nodes (B,b) and (B,c) reversed. Figure 5-4(b) shows the result of applying Equation 5-10 ($\alpha = 0$) to these graphs. Figure 5-4(c) shows the result of applying Equation 5-12, with $\alpha = 0.15$. With the original rule, the initial node weights are unimportant, but with the modified rule, a high value of α can allow the relaxation process to favor a node that has a high initial weight. Figure 5-5 shows the effect of changing the value of α. With a lower value, the arc weights are given more

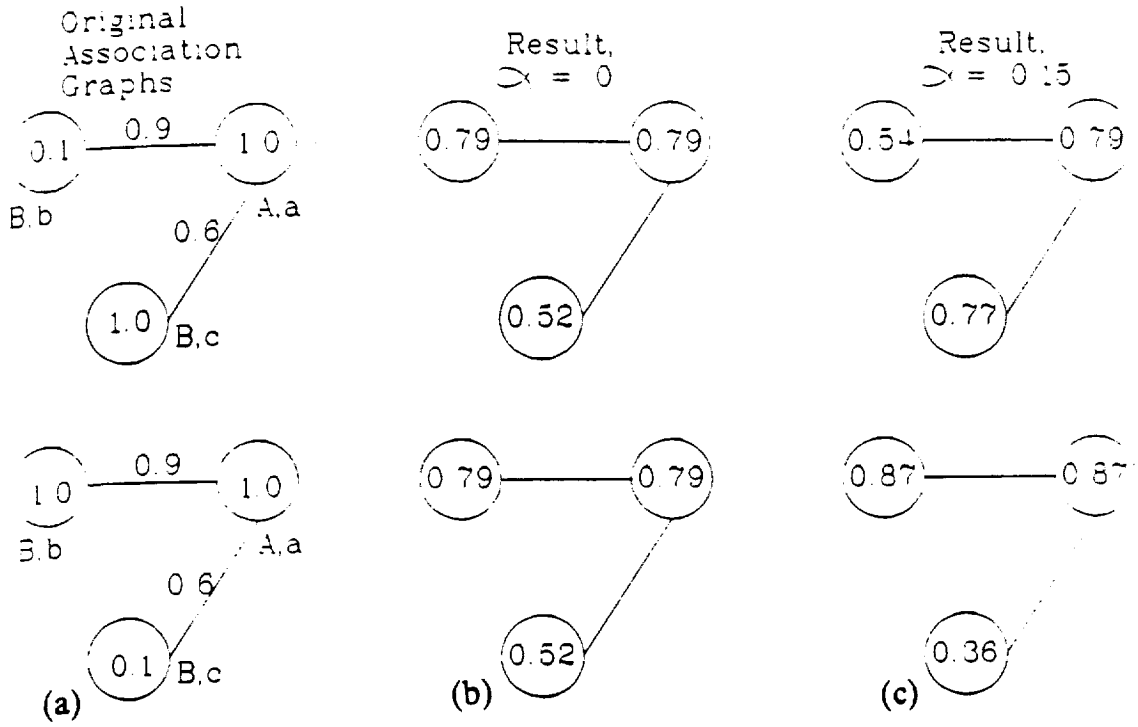


Figure 5-4: With $\alpha = 0$, initial node weights have no effect.

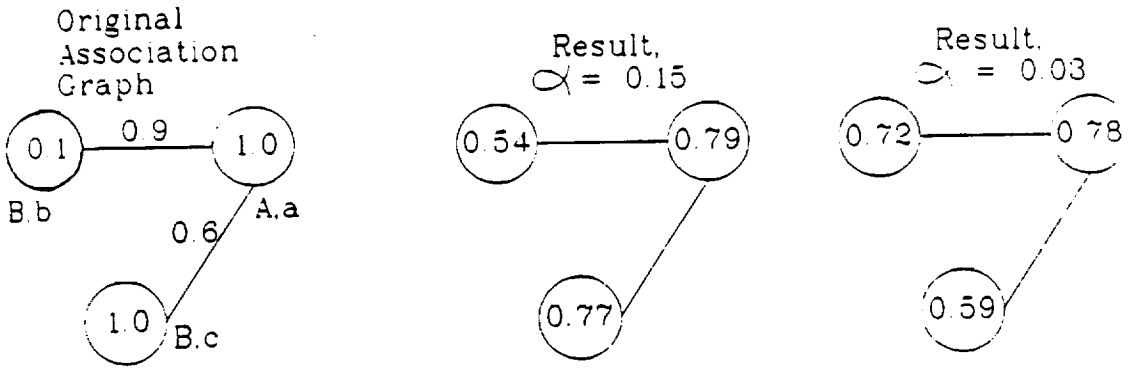


Figure 5-5: Value of α allows importance of node and arc weights to be balanced.

importance, but a higher value gives the node weights more importance. The value of α affects the outcome of the relaxation process, so determining a good value for α is important.

5.2.5 Termination

In fuzzy relaxation, the process changes the weights of each node during each iteration. When scaled so that the sum of node weights is constant, as more iterations are run, the differences between previous and current node weights become smaller and smaller. Running the relaxation algorithm until the differences are below some value is one way to determine when to terminate the algorithm. However, this can be wasteful, since the relative values of the weights may not change after the first few iterations.

The number of iterations needed depends roughly on the radius of the association graph (the maximum length of a shortest path between two vertices). The radius determines how many iterations are needed in order to assure that every node value has affected every other node in the graph. Hence, the radius is the least number of iterations needed. More iterations may be needed, depending on the particular weights on nodes and arcs in the given problem. Because the maximum value of the product of arc weight and node weight for a given region is used as a contribution in updating, it is possible for more iterations to be needed as this maximum may correspond to a different node at a subsequent iteration.

In the examples attempted, the fuzzy relaxation algorithm resulted in probabilities for all desired mappings being above a threshold, and all other mappings being below the threshold, after only a few iterations. After that

point, the probabilities did not change relative order.

5.2.6 Using the Result

The problems remaining after the relaxation algorithm are to choose the proper threshold, above which all region-object mappings will be desirable, and to find and evaluate the cliques.

A hypothetical outcome is shown in Figure 5-6. We are still left with the question of which clique is 'better.' Is it the one with the most nodes, the highest sum of weights, or the highest average of weights? The highest average is not a good measure, since a clique of one node of 0.9 would be considered just as good as one with 5 nodes of 0.9 each. A reasonable way to evaluate these cliques is to first merge the nodes corresponding to the same region or to the same object, averaging their weights. Then, the sum of weights is a meaningful measure of the merit of the clique.

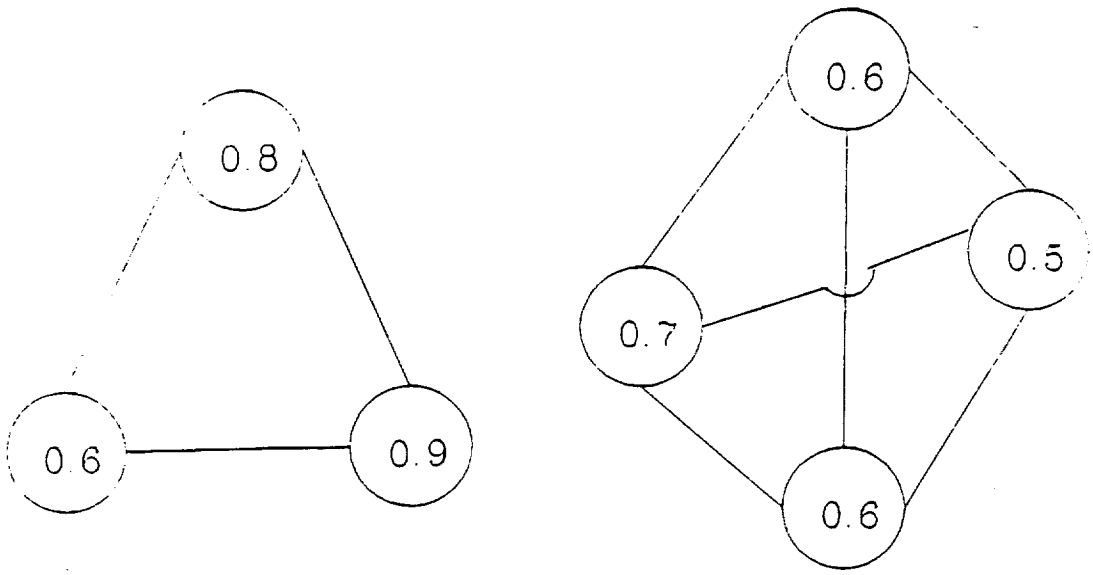


Figure 5-6: Which clique is 'better?'

A simple algorithm to find a 'good' clique which is probably the best, can be used. Select the node with highest weight. Iteratively examine the node with next highest weight to see if it is in the clique with all the previous nodes selected. If so, add it to the clique and continue. If not, examine the node with next highest weight. This procedure will result in finding only one clique, containing the highest-valued node. This clique is likely to be the one with the highest node sum as well. An alternate approach to finding cliques, which insures that the clique with the highest node sum is found, is to use the algorithm outlined in Chapter II and sum the nodes in each of the cliques found to determine the best clique. In spite of increased computation, the later method is preferred.

5.3 Simulation Results - Binary Relations

The fuzzy relaxation process described above was tested on a hypothetical example scene. Attributes of intensity, area, and circularity were used to describe the objects. The node weights were then determined by the following formula:

$$p_{ij}^{(0)} = \frac{K - R(i,j)}{K} \tag{5-13}$$

where K is the number of attributes, and

$$R(i,j) = \sum_{k=1}^{K} |V_{ik} - V_{jk}| W_k . \tag{5-14}$$

V_{ik} is the value of the k^{th} attribute for region i, V_{jk} is the value of the k^{th} attribute for object j, and W_k is a scaling factor which ensures that each term of the sum is in the range [0, 1]. This results in node weights between 0 and 1, with 1 representing a perfect match and 0 representing the worst match possible, given the attributes as measured. The relations 'left-of' and 'above' were defined in terms of four quadrants from an object's centroid: above, left, right, and below, as was shown in Figure 3-2. Because of the nature of these relations, a variation of the method for determining the arc weights $c(i, j; h, k)$ was used. The arc $c(i, j; h, k)$ was set to 1 if the relation between regions i and h, and objects j and k, matched exactly. The value was set to 0 if the relations were reversed, e.g. 'i above h' and 'k above j.' The value was set to 0.5 if the relation was rotated by one quadrant. So, if 'i above h' and 'j left-of k,' the arc weight $c(i, j; h, k)$ was set to 0.5.

The relaxation updating rule used was:

$$p_{ij}^{(r+1)} = \alpha p_{ij}^{(0)} + (1-\alpha)\left[\frac{1}{n}\sum_{h=1}^{n}\left[\max_{k=1}^{m}\left(p_{hk}^{(r)} c(i, j; h, k)\right)\right]\right]. \qquad (5\text{-}15)$$

This rule allows α to range from 0 to 1, with 0 indicating no importance to initial node weights and 1 indicating no importance to arc weights. This is slightly different from Equation 5-12, in which α is in the range $[0, \infty)$.

The following cases of imperfect segmentation were tested:

Case 1 - All model objects have corresponding regions in the observed scene, although there are mismeasured attributes due to segmentation errors, and variations in relations due to a slight rotation.

Case 2 - One object is missing from the observed scene.

Case 3 - The observed scene contains a spurious region that does not map to any of the model objects.

Case 4 - The observed scene is missing an object and also contains a spurious region.

The scene model and the four inexact segmentations are shown in Figure 5-7. The results of the relaxation procedure are shown in Tables 5-1 through 5-4.

In Table 5-1, the final weights obtained by using the updating rule of Equation 5-15 are shown for α set to 0 and to 0.2. In this case, higher values of α were not necessary to obtain a good result. Even though the relations do not match exactly, the relaxation procedure successfully updated the node weights so that the desired mappings came out with high values.

Table 5-2 shows the case of a missing object. In this example, the value of using a non-zero α in the updating rule is demonstrated. If α is set to 0, we cannot decide which of the mappings (R4, O3) and (R4, O5) is better, since the contextual support for each is equal. If α is set to 0.2, the mapping (R4, O5) is favored because it represents a better local match (higher initial node weight) than (R4, O3).

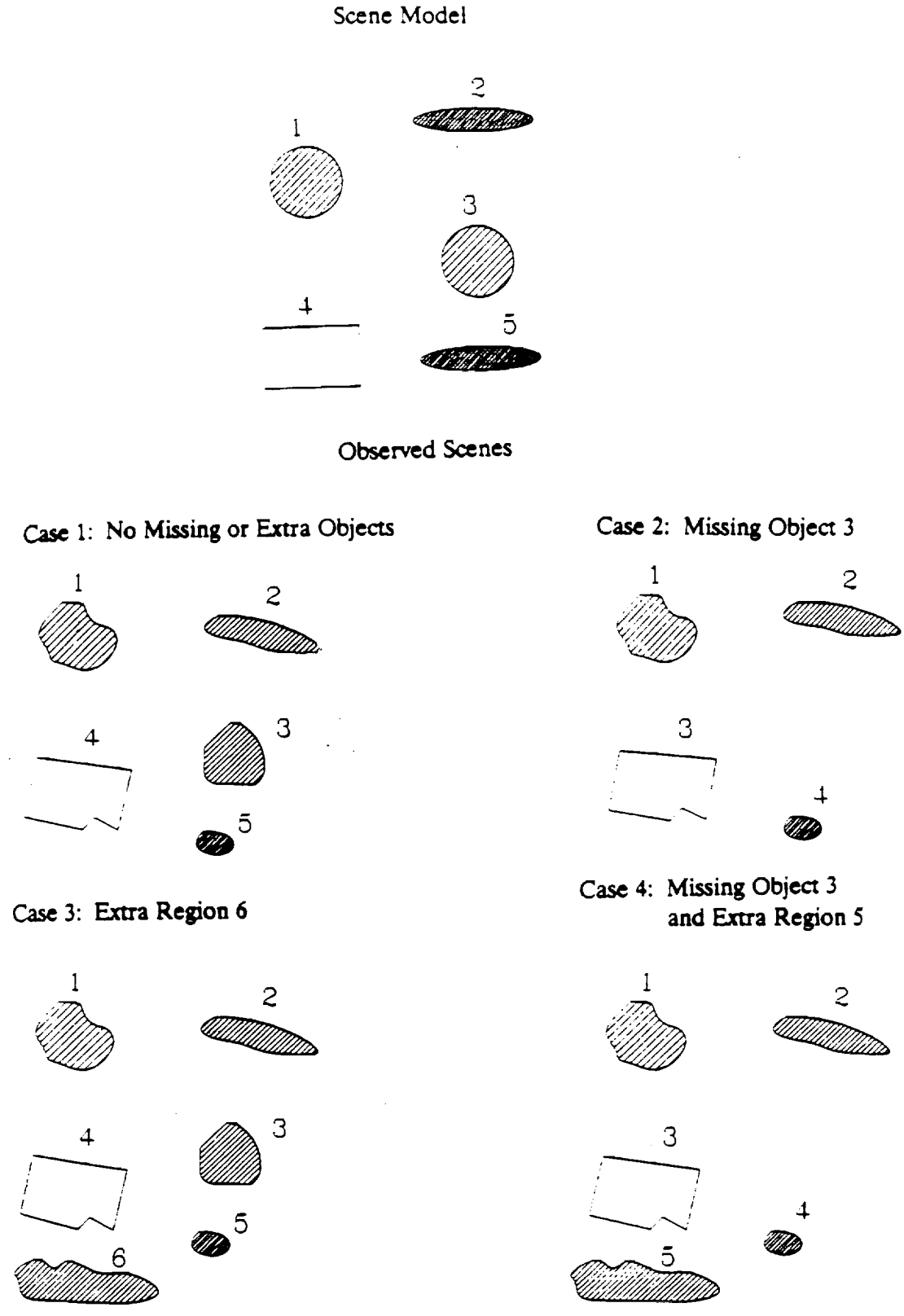


Figure 5-7: Hypothetical scene and four observed scenes with segmentation errors.

Table 5-3 demonstrates the case of an extra region (number 6) in the observed scene. In this example, a non-zero α was not needed. In the result, (R6, O5) does have a relatively high value, but since this mapping is not compatible with the mapping (R5, O5), which has a higher value, the clique containing (R5, O5) would be favored.

Table 5-4 represents the most difficult case, in which there is an extra region and a missing object. The ideal result would be to have no region map to Object 3, and to have Region 5 map to no object. In this table, the importance of choosing an appropriate value for α is shown. If α is set to 0, two incorrect mappings, (R4, O3) and (R5, O5) are favored. (R4, O5) loses to (R4, O3), since the higher initial value of (R4, O5) is ignored. At $\alpha = 0.6$, too much importance is given to the initial node weights, and the contextual support for nodes with relatively low initial weights, such as (R4, O5), does not pull their weights up sufficiently. At $\alpha = 0.2$, the node (R4, O5) is favored, as desired.

C-2

Table 5-1: No missing or extra objects.

ATTRIBUTE MEASUREMENTS

Objects	Intensity	Area	Circularity
1	0.5	1	1
2	0.2	0.6	0.61
3	0.5	1	1
4	0.8	1.7	0.77
5	0.2	0.6	0.6
Regions			
1	0.5	1	0.8
2	0.4	0.62	0.67
3	0.4	0.8	0.8
4	0.8	1.6	0.63
5	0.2	0.2	0.9

RELATIONS

Objects

Left-of

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	1	0	1
5	0	0	0	0	0

Above

	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

Regions

Left-of

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	1	0	1
5	0	0	0	0	0

Above

	1	2	3	4	5
1	0	0	1	1	1
2	0	0	1	0	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

INITIAL WEIGHTS

*(R1,O1)	0.83	(R4,O1)	0.43
(R1,O2)	0.63	(R4,O2)	0.51
(R1,O3)	0.83	(R4,O3)	0.43
(R1,O4)	0.69	*(R4,O4)	0.86
(R1,O5)	0.63	(R4,O5)	0.51
(R2,O1)	0.60	(R5,O1)	0.61
*(R2,O2)	0.86	(R5,O2)	0.67
(R2,O3)	0.60	(R5,O3)	0.61
(R2,O4)	0.51	(R5,O4)	0.31
(R2,O5)	0.86	*(R5,O5)	0.67
(R3,O1)	0.61		
(R3,O2)	0.71		
*(R3,O3)	0.66		
(R3,O4)	0.61		
(R3,O5)	0.71		

* Correct mappings

FINAL WEIGHTS

α:	0	0.2	α:	0	0.2
*(R1,O1)	1.034	0.969	(R4,O1)	-	-
(R1,O2)	0.850	0.795	(R4,O2)	-	-
(R1,O3)	0.717	0.746	(R4,O3)	-	-
(R1,O4)	0.531	0.586	*(R4,O4)	0.938	0.924
(R1,O5)	0.246	0.348	(R4,O5)	-	-
(R2,O1)	-	-	(R5,O1)	0.493	0.520
*(R2,O2)	0.937	0.897	(R5,O2)	0.326	0.422
(R2,O3)	-	-	(R5,O3)	0.889	0.829
(R2,O4)	-	-	(R5,O4)	-	-
(R2,O5)	0.471	0.595	*(R5,O5)	1.034	0.935
(R3,O1)	0.517	0.547			
(R3,O2)	0.666	0.667			
*(R3,O3)	1.034	0.933			
(R3,O4)	0.653	0.635			
(R3,O5)	0.763	0.754			

* Correct mappings

- Initial weight too low to consider

Table 5-2: Missing Object 3.

ATTRIBUTE MEASUREMENTS

Objects	Intensity	Area	Circularity
1	0.5	1	1
2	0.2	0.6	0.61
3	0.5	1	1
4	0.8	1.7	0.77
5	0.2	0.6	0.6
Regions			
1	0.5	1	0.8
2	0.4	0.62	0.67
3	0.8	1.6	0.63
4	0.2	0.2	0.9

RELATIONS

Objects

Left-of

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	1	0	1
5	0	0	0	0	0

Above

	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

Regions

Left-of

	1	2	3	4
1	0	1	0	1
2	0	0	0	0
3	0	1	0	1
4	0	0	0	0

Above

	1	2	3	4
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0
4	0	0	0	0

INITIAL WEIGHTS

*(R1,O1)	0.83	(R3,O1)	0.43
(R1,O2)	0.63	(R3,O2)	0.51
(R1,O3)	0.83	(R3,O3)	0.43
(R1,O4)	0.69	*(R3,O4)	0.86
(R1,O5)	0.63	(R3,O5)	0.51
(R2,O1)	0.60	(R4,O1)	0.61
*(R2,O2)	0.86	(R4,O2)	0.67
(R2,O3)	0.60	(R4,O3)	0.61
(R2,O4)	0.51	(R4,O4)	0.31
(R2,O5)	0.86	*(R4,O5)	0.67

* Correct mappings

FINAL WEIGHTS

α:	0	0.2	α:	0	0.2
*(R1,O1)	1.033	0.965	(R3,O1)	-	-
(R1,O2)	0.803	0.762	(R3,O2)	-	-
(R1,O3)	0.702	0.721	(R3,O3)	-	-
(R1,O4)	0.564	0.616	*(R3,O4)	0.935	0.924
(R1,O5)	0.331	0.415	(R3,O5)	-	-
(R2,O1)	-	-	(R4,O1)	0.331	0.409
*(R2,O2)	0.909	0.873	(R4,O2)	0.538	0.569
(R2,O3)	-	-	(R4,O3)	1.033	0.914
(R2,O4)	-	-	(R4,O4)	-	-
(R2,O5)	0.539	0.651	*(R4,O5)	1.033	0.931

* Correct mappings

- Initial weight too low to consider

Table 5-3: Extra Region 6.

ATTRIBUTE MEASUREMENTS

Objects	Intensity	Area	Circularity
1	0.5	1	1
2	0.2	0.6	0.61
3	0.5	1	1
4	0.8	1.7	0.77
5	0.2	0.6	0.6
Regions			
1	0.5	1	0.8
2	0.4	0.62	0.67
3	0.4	0.8	0.8
4	0.8	1.6	0.63
5	0.2	0.2	0.9
6	0.4	0.8	0.8

RELATIONS

Objects

Left-of

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	1	0	1
5	0	0	0	0	0

Above

	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

Regions

Left-of

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	1	1	0	1	0
5	0	0	0	0	0	0
6	0	0	1	0	1	0

Above

	1	2	3	4	5	6
1	0	0	1	1	1	1
2	0	0	1	0	1	1
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	0	0
6	0	0	0	0	0	0

INITIAL WEIGHTS

*(R1,O1)	0.83	(R4,O1)	0.43
(R1,O2)	0.63	(R4,O2)	0.51
(R1,O3)	0.83	(R4,O3)	0.43
(R1,O4)	0.69	*(R4,O4)	0.86
(R1,O5)	0.63	(R4,O5)	0.51
(R2,O1)	0.60	(R5,O1)	0.61
*(R2,O2)	0.86	(R5,O2)	0.67
(R2,O3)	0.60	(R5,O3)	0.61
(R2,O4)	0.51	(R5,O4)	0.31
(R2,O5)	0.86	*(R5,O5)	0.67
(R3,O1)	0.61	(R6,O1)	0.58
(R3,O2)	0.71	(R6,O2)	0.77
*(R3,O3)	0.66	(R6,O3)	0.58
(R3,O4)	0.61	(R6,O4)	0.58
(R3,O5)	0.71	(R6,O5)	0.77

* Correct mappings

FINAL WEIGHTS

	α: 0	0.2		α: 0	0.2
*(R1,O1)	1.113	1.031	(R4,O1)	-	-
(R1,O2)	0.912	0.846	(R4,O2)	-	-
(R1,O3)	0.821	0.831	(R4,O3)	-	-
(R1,O4)	0.567	0.592	*(R4,O4)	0.939	0.903
(R1,O5)	0.227	0.337	(R4,O5)	-	-
(R2,O1)	-	-	(R5,O1)	0.547	0.562
*(R2,O2)	1.025	0.969	(R5,O2)	0.396	0.477
(R2,O3)	-	-	(R5,O3)	0.919	0.846
(R2,O4)	-	-	(R5,O4)	-	-
(R2,O5)	0.315	0.475	*(R5,O5)	0.991	0.907
(R3,O1)	0.551	0.579	(R6,O1)	-	-
(R3,O2)	0.690	0.690	(R6,O2)	0.362	0.463
*(R3,O3)	1.035	0.931	(R6,O3)	-	-
(R3,O4)	0.650	0.644	(R6,O4)	-	-
(R3,O5)	0.752	0.752	(R6,O5)	0.828	0.804

* Correct mappings

- Initial weight too low to consider

Table 5-4: Extra Region 5, missing Object 3.

ATTRIBUTE MEASUREMENTS

Objects	Intensity	Area	Circularity
1	0.5	1	1
2	0.2	0.6	0.61
3	0.5	1	1
4	0.8	1.7	0.77
5	0.2	0.6	0.6
Regions			
1	0.5	1	0.8
2	0.4	0.62	0.67
3	0.8	1.6	0.63
4	0.2	0.2	0.9
5	0.4	1.2	0.6

RELATIONS

Objects

Left-of

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	1	0	1
5	0	0	0	0	0

Above

	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

Regions

Left-of

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	0	0
3	0	1	0	1	0
4	0	0	0	0	0
5	0	0	0	1	0

Above

	1	2	3	4	5
1	0	0	1	1	1
2	0	0	0	1	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

INITIAL WEIGHTS

*(R1,O1)	0.83	(R4,O1)	0.61
(R1,O2)	0.63	(R4,O2)	0.67
(R1,O3)	0.83	(R4,O3)	0.61
(R1,O4)	0.69	(R4,O4)	0.31
(R1,O5)	0.63	*(R4,O5)	0.67
(R2,O1)	0.60	(R5,O1)	0.58
*(R2,O2)	0.86	(R5,O2)	0.77
(R2,O3)	0.60	(R5,O3)	0.58
(R2,O4)	0.51	(R5,O4)	0.58
(R2,O5)	0.86	(R5,O5)	0.77
(R3,O1)	0.43		
(R3,O2)	0.51		
(R3,O3)	0.43		
*(R3,O4)	0.86		
(R3,O5)	0.51		

* Correct mappings

FINAL WEIGHTS

α:	0	0.2	0.6	α:	0	0.2	0.6
*(R1,O1)	1.148	1.057	0.913	(R4,O1)	0.565	0.577	0.599
(R1,O2)	0.910	0.844	0.721	(R4,O2)	0.445	0.508	0.603
(R1,O3)	0.856	0.845	0.828	(R4,O3)	0.992	0.884	0.713
(R1,O4)	0.440	0.521	0.641	(R4,O4)	-	-	-
(R1,O5)	0.292	0.389	0.536	*(R4,O5)	0.971	0.900	0.766
(R2,O1)	-	-	-	(R5,O1)	-	-	-
*(R2,O2)	1.047	0.981	0.894	(R5,O2)	0.317	0.436	0.631
(R2,O3)	-	-	-	(R5,O3)	-	-	-
(R2,O4)	-	-	-	(R5,O4)	-	-	-
(R2,O5)	0.444	0.565	0.761	(R5,O5)	0.934	0.876	0.799
(R3,O1)	-	-	-				
(R3,O2)	-	-	-				
(R3,O3)	-	-	-				
*(R3,O4)	0.927	0.907	0.886				
(R3,O5)	-	-	-				

* Correct mappings

- Initial weight too low to consider

5.4 Simulation Results: Real-Valued Relations

The simulation presented in this section demonstrates the usefulness of real-valued positional relations. In the simulation of the previous section, since only the binary relations left-of and above were used, only slight rotations of the scene could be handled. With the relative positions of objects represented as angles, a large rotation of the scene can be handled, by running the relaxation process on several rotated versions of the observed scene. The relaxation process applied to the rotation representing the best match should result in the clique with the highest node sum. If only a small rotation is expected to occur, there is no need to run the relaxation process using several rotations.

To demonstrate the use of real-valued positional relations with the scene of the previous example, the relations in the stored and observed scenes shown in Figure 5-8 were represented as angles in degrees that a line from the first object to the second makes with the horizontal.

The weights of arcs in the association graph were determined by the difference of angles in the observed scene and the stored scene. If the angles match exactly, the arc weight equals 1. If the angles differ by 180 degrees, the arc weight is 0. The determination of these arc weights is similar to the determination of node weights, since real values of relations are compared for similarity. The only difference is that in angle relations we must make corrections so that the angle differences are not overstated. (For example, an angle difference of 200 degrees is actually a difference of 160 degrees: no differences can be greater than 180.) In the relaxation process, only the arcs with weights above 0.85 were kept.

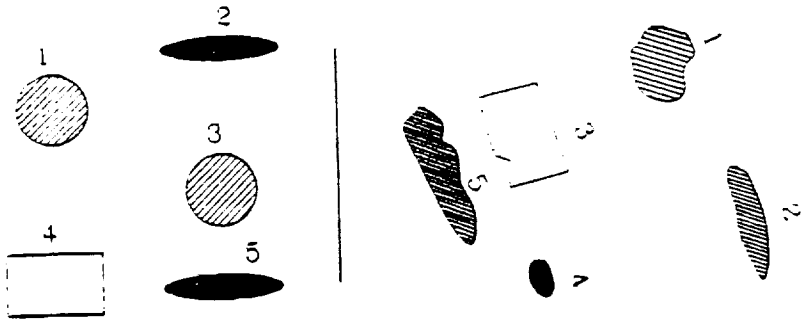


Figure 5-8: A stored scene and rotated observed scene.

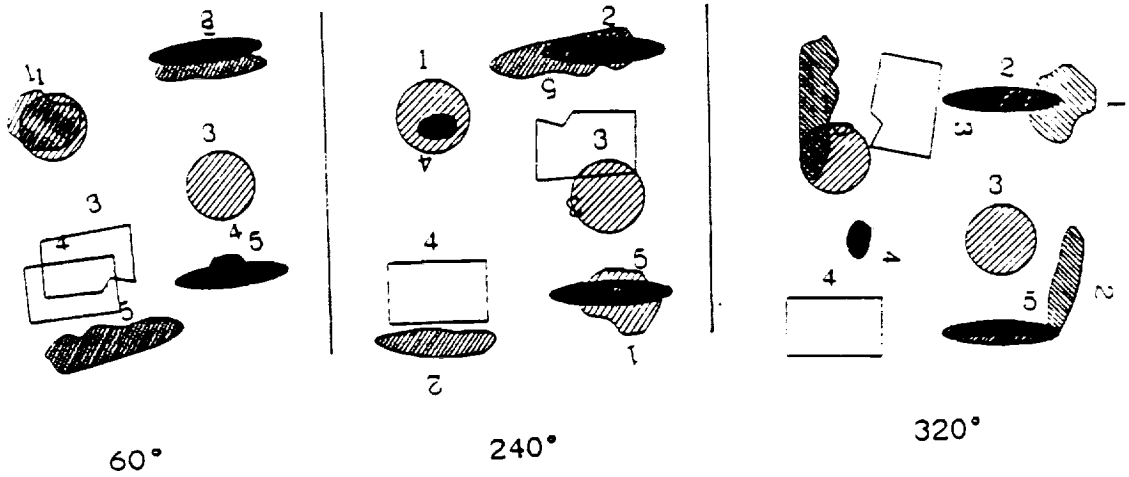


Figure 5-9: Observed scene superimposed on stored scene, three rotations.

The relaxation process was run on the scenes of Figure 5-8, for 18 different rotations, 20 degrees apart, with α set to 0. This simulation shows that the process can be used to rotation-normalize scenes.

The results obtained can be interpreted by finding the clique with the highest node weight sum for each rotation. We find that the rotation that comes closest to matching the original scene is the one with the highest sum. Table 5-5 shows these node weight sums for each rotation. In Table 5-6, the results for rotations of 60, 240, and 320 degrees are shown. These three rotations produced the highest-valued cliques, i.e. the ones with the highest node sums. Figure 5-9 shows the observed scene superimposed on the stored scene at each of the three angles, to show that the favored mappings for each rotation do make sense. It is evident that at a rotation of 60 degrees, the mappings (R1,O1), (R2,O2), (R3,O4), and (R4,O5) are the best. At 240 degrees, the closest mappings are (R1,O5), (R4,O1), and (R5,O2). At 320 degrees, the relaxation process results in (R1,O2), (R2,O5), and (R4,O1) having the highest values.

Table 5-5: Sums of highest cliques, 18 rotations of observed scene.

Angle	Sum	Angle	Sum
0	6.902	180	7.830
20	5.557	200	5.332
40	4.773	220	5.216
60	9.498	240	8.660
80	7.906	260	6.000
100	7.636	280	6.730
120	6.579	300	6.076
140	5.109	320	8.600
160	5.149	340	6.530

Table 5-6: Extra Region 5, missing Object 3.

Stored and observed scenes shown in Figure 5-8.

ATTRIBUTE MEASUREMENTS

Objects	Intensity	Area	Circularity
1	0.5	1	1
2	0.2	0.6	0.61
3	0.5	1	1
4	0.8	1.7	0.77
5	0.2	0.6	0.6
Regions			
1	0.5	1	0.8
2	0.4	0.62	0.67
3	0.8	1.6	0.63
4	0.2	0.2	0.9
5	0.4	1.2	0.6

RELATIONS

Objects

	1	2	3	4	5
1	-	18	335	270	313
2	-	-	270	235	270
3	-	-	-	207	270
4	-	-	-	-	0
5	-	-	-	-	-

Regions

	1	2	3	4	5
1	-	295	205	238	205
2	-	-	155	190	167
3	-	-	-	275	203
4	-	-	-	-	130
5	-	-	-	-	-

INITIAL WEIGHTS

*(R1,O1)	0.83	(R4,O1)	0.61
(R1,O2)	0.63	(R4,O2)	0.67
(R1,O3)	0.83	(R4,O3)	0.61
(R1,O4)	0.69	(R4,O4)	0.31
(R1,O5)	0.63	*(R4,O5)	0.67
(R2,O1)	0.60	(R5,O1)	0.58
*(R2,O2)	0.86	(R5,O2)	0.77
(R2,O3)	0.60	(R5,O3)	0.58
(R2,O4)	0.51	(R5,O4)	0.58
(R2,O5)	0.86	(R5,O5)	0.77
(R3,O1)	0.43		
(R3,O2)	0.51		
(R3,O3)	0.43		
*(R3,O4)	0.86		
(R3,O5)	0.51		

* Correct mappings

FINAL WEIGHTS

Angle:	60	240	320	Angle:	60	240	320
*(R1,O1)	2.415	0.000	0.000	(R4,O1)	0.000	2.862	2.892
(R1,O2)	0.004	0.000	2.903	(R4,O2)	0.000	0.044	0.000
(R1,O3)	0.004	1.496	1.366	(R4,O3)	0.770	0.044	0.074
(R1,O4)	0.004	0.000	0.000	(R4,O4)	-	-	-
(R1,O5)	0.000	2.880	0.050	*(R4,O5)	2.352	0.000	0.074
(R2,O1)	-	-	-	(R5,O1)	-	-	-
*(R2,O2)	2.341	0.000	0.000	(R5,O2)	0.000	2.918	0.000
(R2,O3)	-	-	-	(R5,O3)	-	-	-
(R2,O4)	-	-	-	(R5,O4)	-	-	-
(R2,O5)	0.004	0.044	2.805	(R5,O5)	0.004	0.000	0.000
(R3,O1)	-	-	-				
(R3,O2)	-	-	-				
(R3,O3)	-	-	-				
*(R3,O4)	2.390	0.000	0.050				
(R3,O5)	-	-	-				

* Correct mappings

- Initial weight too low to consider

5.4.1 Discussion

As mentioned previously, using a nonzero value of α allows for a balance between the importance of good local matches of regions to objects, and the importance of good matches between relations. At $\alpha = 0$, the initial node weights are used only to eliminate very poor mappings by thresholding before relaxation begins. After that point, the best match of relations alone determines which mapping is favored, and the initial node weights have no effect. A non-zero value of α tends to hold down the values of nodes with low initial weights, and tends to maintain the values of nodes with high initial weights, thus incorporating local and contextual information into the node weights.

Even if error-free relations are expected, a nonzero α can be useful. In Figure 5-10, an observed and stored scene are shown, the observed scene rotated 180 degrees. The object centroids form a square, so if the observed scene is rotated 0, 90, 180, or 270 degrees, perfect matches can be found among relations. In this scene, if the node weights for all region-object mappings are above the acceptance threshold, the only way to favor the 180 degree rotation is to use a non-zero α, so that the initial node weights influence the result.

Rotation of the observed scene can be handled by performing relaxation at several rotation angles and choosing the best result. At the

rotation angle that matches best, any remaining differences in angles between observed and stored scene are caused by tilt and by errors in segmentation which cause object centroids to be incorrect.

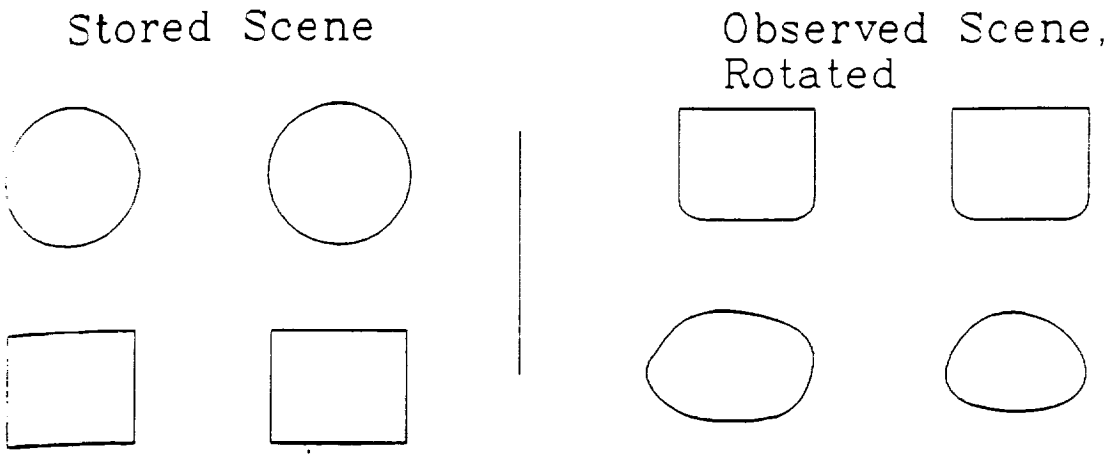


Figure 5-10: Stored and rotated observed scene: node weights are important.

A parallel implementation is needed to make this approach feasible. All rotation angles should be processed in parallel, and ideally all nodes in the association graph should be updated simultaneously in each iteration of the relaxation process.

In this simulation, none of the object attributes were rotation-sensitive. For example, in Figure 5-9, at 320 degrees, even though region 2 and object 2 are not aligned correctly, (R2, O2) has the same initial weight as it does at 60 degrees, the correct rotation. The addition of

shape-based attributes that are rotation-sensitive, and the measurement of these attributes at each attempted rotation, would improve the results by allowing poor shape matches to be discarded before relaxation.

Naturally, if the scene contains symmetries, there will be several rotations at which the relaxation algorithm produces cliques with high values. But if there is no way for anyone to distinguish which of several rotations is the 'correct' one, the relaxation process cannot be expected to succeed either.

VI. HANDLING OVERSEGMENTATION AND UNDERSEGMENTATION

The method described in Chapter V can handle extra regions, missing objects, inexact measurements, and errors in relations. But the problems due to merged or split regions are not addressed.

Section 6.1 describes various ways in which oversegmentation and undersegmentation problems could be handled by the association graph relaxation method. In Section 6.2, the best of these methods is described in more detail. A simulation on a hypothetical scene containing split and merged regions is presented in Section 6.3.

6.1 Approaches to the Problem

One of the reasons for pursuing the association graph approach to scene matching is that it can allow for the mapping of multiple regions to one object, or one region to multiple objects. In this section, various approaches to the problem are discussed in terms of changes in the association graph before the relaxation process is done, and any changes in the relaxation process itself, that are needed in order to handle merged and split regions.

There are two ways of approaching these problems. One is to go back to the segmented image when the possibility of a merged or split object is indicated. The regions in question would then actually be split, or merged, the attributes re-measured, and the graphs of the relations re-built. Then, a revised association graph would be built, and the relaxation process run. The other approach is to

avoid resegmenting the image, and modify the association graph method itself to handle over and undersegmentation. This approach appears more practical and efficient. The simplest possible approach would be to use the relaxation process as described in Chapter V on an association graph that has weighted arcs between nodes that map the same region to two objects, or the same object to two regions. This allows multiple nodes involving the same region or object to belong to the same clique in the association graph. Since this approach has drawbacks, it and other approaches involving modifications of the association graph and the relaxation process are discussed in the next sections.

Obviously, we do not want to try out all possibilities for split or merged regions. Only the regions/objects for which we suspect splitting or merging should be investigated. For a region A, we can suspect that the mappings (A,a) (region A to object a) and (A,b) may both be correct if 1) objects a and b are near one another, 2) each region-wide attribute of A, such as texture, color, or intensity, is close in value to the weighted average of the attribute value of objects a and b, and 3) the area of A is close to the sum of the areas of a and b. Condition 1 can be determined by storing an extra relation , 'next-to,' for each scene. Object a is 'next-to' b if there is a possibility of these two objects being merged together by segmentation. If a and b are adjacent, this possibility clearly exists. Also, if a and b are near each other (although not touching), and have no other objects between them, they could be merged in segmentation. This 'next-to' relation could be the real-valued distance relation discussed in Section 3.3.2.

For oversegmentation, the three conditions above need to be applied to the observed scene. So, we suspect that the mappings (A,a) and (B,a) are both correct if 1) regions A and B are near one another, 2) region-wide attributes are similar, and 3) the area of a is close to the sum of the areas of A and B.

It is possible that either or both of mappings (A,a) and (A,b) may be too unlikely to have nodes in the association graph, and that the mapping (A,(a,b)) should be included in the graph. So, the possibilities for merged or split regions should be determined by comparing region descriptions and 'next-to' relations rather than by looking only at existing nodes in the association graph.

The following sections describe various approaches to handling this problem. The sections are written assuming undersegmentation, or a merged region, but approaches for handling oversegmentation are analogous to these.

6.1.1 Assignment of Arc Weights Only

In the simplest approach, the method presented in Chapter V can be modified by simply allowing for arcs between nodes mapping one region to different objects. This is illustrated in Figure 6-1. Objects a and b in the stored scene are observed as one region, A. The weight z for arc ((A,a),(A,b)) could be determined as if it were an estimated node weight for a node (A,(a,b)) mapping region A to both objects a and b. If the mappings (A,a) and (A,b) are both supported by the context of the scene, they will tend to increase. If not, it is likely that one or the other will increase, but not both. The node weights p and q still represent the similarities between region A and object a, and region A and object b, without taking into account the fact that

the region could be two objects merged together. So, these node weights are likely to be quite low. The arc weights x and y, likewise, are computed based on the assumption that A is not two objects merged together.

If the relaxation updating rule is not modified, a node's weight can only be affected by one contribution from each region. So, for example in Figure 6-1, only the greater of $c(A,a;A,b) \times p_{A,b}$ and $p_{A,a}$ can affect the new weight of node (A,a), making it unlikely that the node (A,b) could affect the weight of (A,a). The final result would be interpreted as in Chapter V, with nodes with weights below a threshold discarded and the remaining cliques evaluated to find the best one. Because an arc is allowed between (A,a) and (A,b), if both nodes have weights above the threshold after relaxation, they can both be members of the highest valued clique.

Although this approach requires few modifications to the original relaxation procedure, it is an unsatisfying solution because the weights on nodes and arcs do not correspond to the reality of the situation, and all arc weights are not calculated in the same way.

6.1.2 Re-estimation of Attributes and Relations

Another approach is to consider region A to be split into two regions, A_1 and A_2, and to estimate new attributes and relations for these regions from the regional and relational descriptions for the original region. This is shown in Figure 6-2. The nodes (A,a) and (A,b) are replaced by (A_1,a) and (A_2,b), to reflect the assumption that region A should be split. Relations between A_1 and A_2 could be assumed to be the same as between objects a and b, so that

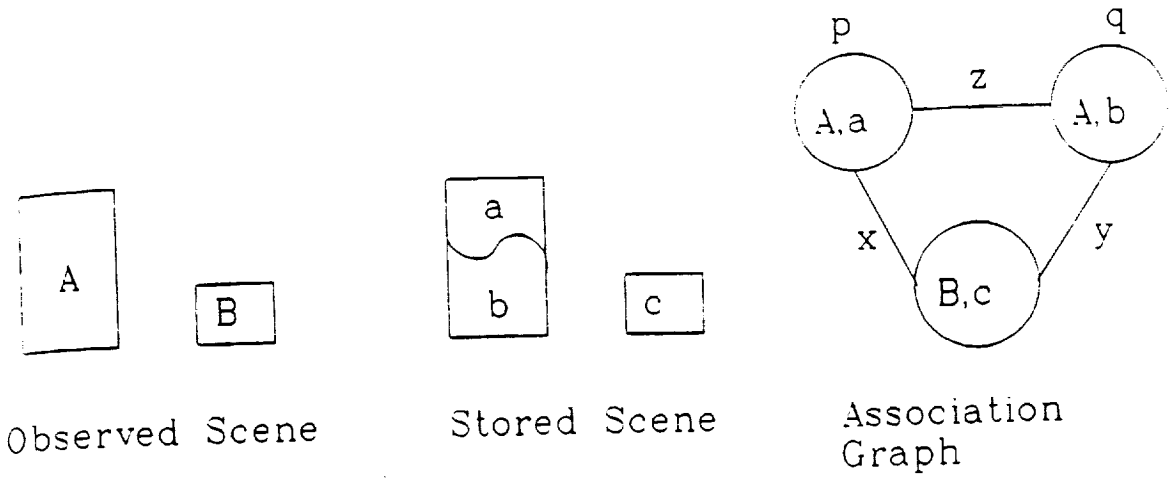


Figure 6-1: Handling merged regions by adding an extra arc (z).

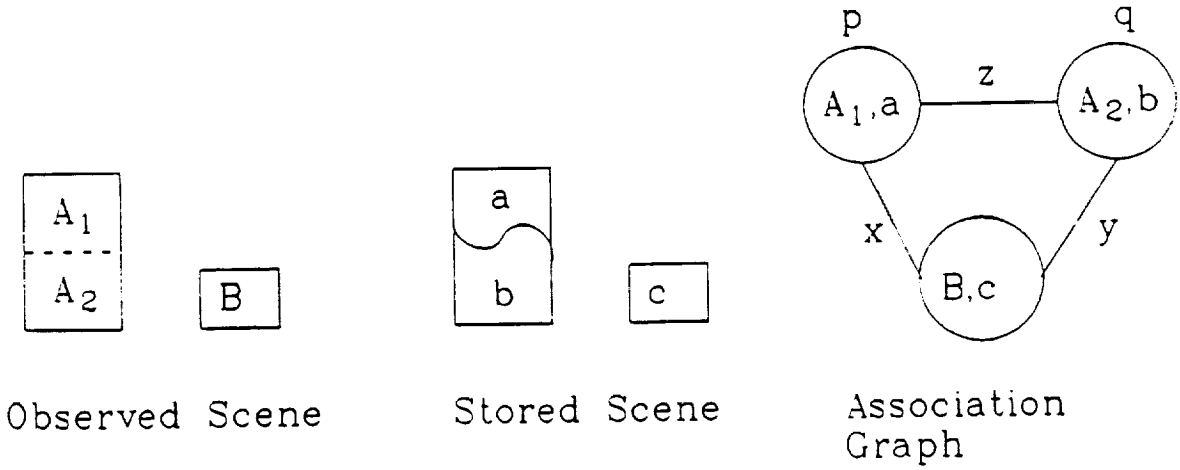


Figure 6-2: Handling merged regions by re-estimating attributes for the two parts.

the value of z would be 1. The values of p and q would be updated to reflect the fact that areas of regions A_1 and A_2 are each estimated to be in the same proportion as objects a and b. This will increase the values of p and q if the region should indeed be split. Values x and y would be recomputed, based on new estimates of the locations of centroids of regions A_1 and A_2.

The original values on nodes (A,a) and (A,b) are lost when this approach is taken. So, we abandon the assumptions that A could map to *only* a, or to *only* b. If one of these possibilities represents a correct mapping, this approach jeopardizes the chance of only one of the nodes ending up with a high weight after the relaxation process is completed.

6.1.3 Adding the Split Nodes to the Original Graph

Rather than replacing the old nodes (A,a) and (A,b) with nodes (A_1,a) and (A_2,b), we could add these new nodes to the original graph, so that we have the association graph shown in Figure 6-3. The weights p and q on the original nodes remain the same, and the weights p' and q' are recomputed as described above. The weights x and y also remain unchanged, while u and w are computed as above. The weight z will be taken as 1, since we assume A to be split in such a way that A_1 and A_2 have the same relations as a and b.

The benefit of this approach is that we continue to allow for the possibility that A maps to object a, or to object b, while adding the possibility that A has been merged. Also, the relaxation rule does not need to be changed. The drawback, of course, is that we have two extra nodes added to the association graph.

6.1.4 Creation of Merge Nodes

Another approach is to create merge nodes such as (A,(a,b)), with weights based on estimated attribute values of the merged objects, and add these merge nodes to the original graph. The resulting association graph is shown in Figure 6-4. The arcs adjacent to these merge nodes in the association graph need to be determined by finding relations of the 'object' (a,b) to other objects in the stored scene, and comparing these relations with those of region A. This could be done by estimating the centroid of the merged object as the midpoint of a line between the centroids of objects a and b. Then, the relations between this new centroid and the centroids of other objects would be found.

The reason for representing the merged region by only one node is that the two new nodes described in the previous section are expected to respond similarly in the relaxation process. Either both should have high values after relaxation, or both should have low values. If only one of the new nodes has a low value, it implies that the region in that node is spurious and also the object in that node is missing in the observed scene, which is not a likely occurrence. So, there is little advantage in representing the mapping of one region to two objects by multiple nodes. It only increases the computation needed.

6.1.5 Partial Re-segmentation of Image

The approaches described so far require us to go back to the regional and relational descriptions of the stored and observed scenes, but not to reconsider the scene itself. The most complicated approach would involve identifying possible merged regions and re-segmenting that portion of the

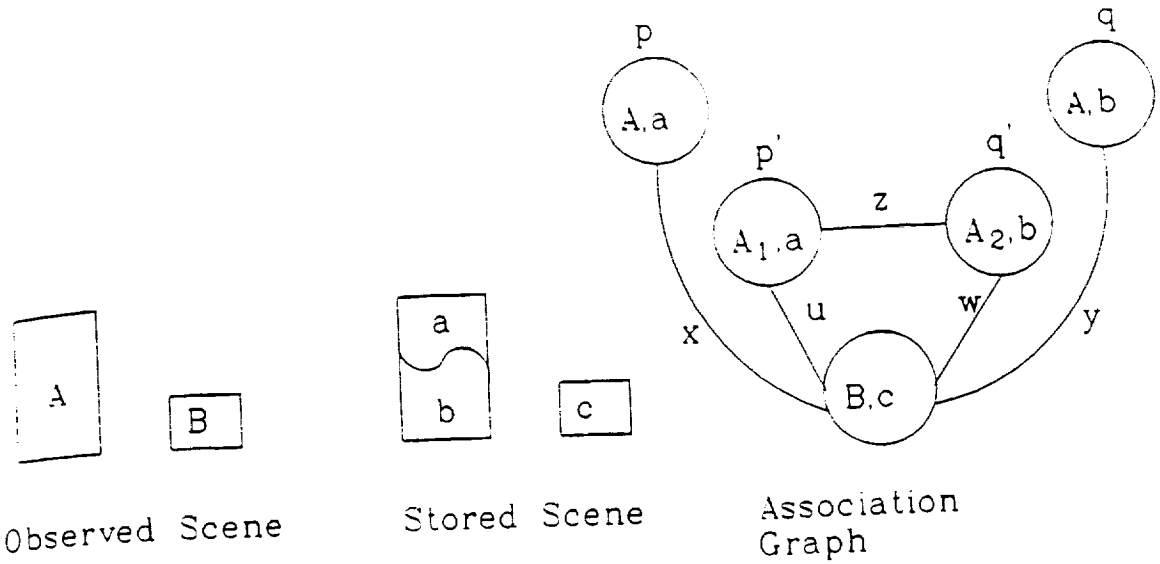


Figure 6-3: Adding re-estimated nodes to the original graph.

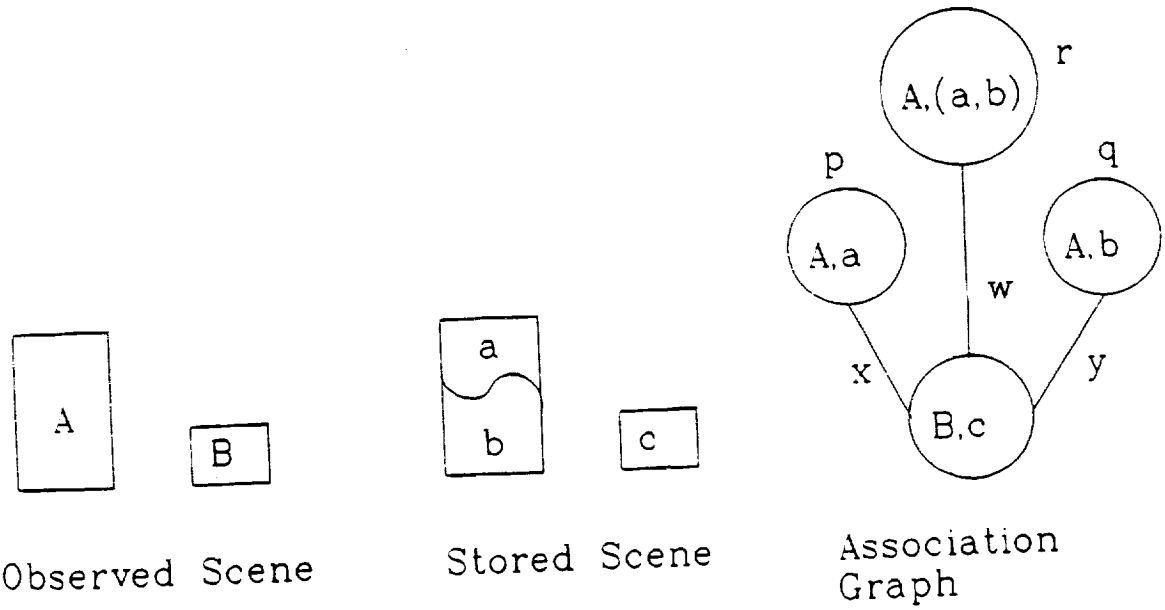


Figure 6-4: Adding a node representing mapping of merged region to two objects.

original image to obtain a segmentation in which the regions are not merged into one. Then, new regional and relational descriptions for the changed regions would need to be created, and a new association graph built.

6.1.6 Evaluation

In evaluating these approaches, there are three goals to keep in mind:

1) All reasonable possibilities for mappings should be retained.
2) The method should not require the addition of more nodes than necessary to the association graph.
3) The node and arc weight computations should make sense.

The example of Figure 6-2 failed to support the first goal, since the mappings of the whole region to each of the objects were no longer considered possible. Nodes (A,a) and (A,b) were not retained in the association graph. The example of Figure 6-3 hinders the second goal, because it requires the addition of two new nodes for each split or merged region. The example of Figure 6-1 supports the first two goals, but not the third. The arc weight between (A,a) and (A,b) is not based on similarities in relations at all, since the region A has no relations to itself. Also, the node weights of (A,a) and (A,b) only reflect the similarity of the whole region A to a, and the whole region A to b. There is nothing in the graph that reflects the similarity of region A with the merged object (a,b), or similarities of region A_1 to object a and region A_2 to object b. So, the method of Figure 6-4 appears to be the best.

Intuitively, it seems that extra nodes should be added (as in Figure 6-3) in the case of undersegmentation, and merge nodes should be added (as in Figure 6-4) in the case of oversegmentation. This would allow the association graph to more accurately reflect the actual situation. However, since there is little advantage to having the extra nodes of Figure 6-3, the method used here creates merge nodes for both the merged regions and the split regions, so that if (A,a) is compatible with (A,b) we create a node (A,(a,b)) representing the merge of objects a and b. Likewise, if (A,a) is compatible with (B,a), we create node ((A,B),a), merging regions A and B. A drawback of this approach is that we add several more nodes and arcs to the association graph, for which we must compute initial weights and update. An advantage is that we still allow for the possibility that the regions in question have *not* been split/merged, by leaving the original weights of nodes (A,a) and (A,b) uncorrupted.

6.2 Procedure for Handling Split and Merged Regions

The procedure used is the last possibility discussed above. We allow for creation of merge nodes which represent merging of observed regions, and merge nodes which represent merging of objects in the stored scene. The procedure is as follows:

1) Compute the initial node weights for the regions and objects.

2) Determine candidate regions/objects for which we will attempt to merge corresponding objects/regions. If all of a region's node weights are below a threshold, this indicates that none of the objects map well to that region. If this is the case, consider pairs of objects. If a pair of objects are adjacent

(or meet some 'closeness' criterion), have similar region-wide attribute values, and have region-wide attribute values that match well with the region having all low node weights, and the merged object does not already exist, then create a merged object, and compute its node weight with the region. If the merged object already exists, simply compute its node weight with the region. Similarly, check node weights for each object. If an object has all low node weights, attempt to create a merged region which will map well to the object.

2a) Process for creating a merged object or region: Calculate attribute values for the merged object or region. For region-wide attributes, the new values are weighted averages of the values for the two objects being merged. For area, the new value is the sum of the areas of the two objects. Add a row and column for the new object to the matrices representing the relations, and estimate values of the new object's relations with other objects. Angle relations are estimated by simply averaging the angle relations of the two constituent objects. The real-valued adjacency relation (the percentage of an object's pixels that border on another object) is computed by determining the perimeter of the merged object (the sum of perimeters minus twice the length of their shared boundary). Then, the percentage of border pixels with other objects is calculated based on the new perimeter. Since the new object is simply added to the list of objects, it can also be considered for merging with other objects, thus allowing for merges of three or more objects.

2b) Compute node weight of the merged object with the region that prompted the attempt to find merged objects. Since the node weights are represented in a matrix with rows corresponding to objects and columns corresponding to regions, if the merged object is being newly added to the list of objects, all the other node weights for the merged object should be initialized to zero.

3) After all plausible merged objects/regions have been created and their initial node weights computed, find the values of the arcs in the association graph. Arcs are not allowed between nodes that include the same object or the same region; e.g., there is no arc between node (R1,(O1,O2)) and (R2,O2). Arcs from merged objects/regions are computed in the same way as other arcs, since the appropriate estimated relation values for the merged objects are included in the relation matrices, just as for any other objects.

4) Perform the relaxation process on the association graph which includes nodes involving merged objects or regions.

A slight change in the relaxation updating rule is necessary. Since two nodes including the same region are defined as incompatible, nodes involving regions that belong to merge nodes have fewer terms possible in the sum in the updating rule, which is an unfair disadvantage. Nodes including only a region that is not involved in a merge node have possible terms in the sum from every

region. To make up for missing terms, a merge compensation factor, b_i is included in the updating rule. The value of $p_{ij}^{(r)}$ is then counted b_i times in the sum. So, the new updating rule is given by:

$$p_{ij}^{(r+1)} = \alpha p_{ij}^{(0)} + (1-\alpha)\left[\frac{1}{n}\left[b_i p_{ij}^{(r)} + \sum_{\substack{h=1 \\ h \neq i}}^{n}\left[\max_{k=1}^{m}(p_{hk}^{(r)} c(i,j;h,k))\right]\right]\right]. \qquad (6\text{-}1)$$

An example of this is shown in Table 6-1. Assuming there are four regions, 1, 2, 3, and (1,2), the table shows that for nodes which map region 1 (or 2) to some object, there can only be terms in the sum in the updating rule from nodes involving regions 1, 2, and 3. However, since nodes involving region 3 could be compatible with nodes involving any of the regions, they have four possible terms. Nodes mapping region (1,2) to any object can have terms from only region 3 and region (1,2). The factors b_i compensate for these missing terms by counting the value of p_{ij} multiple times.

Table 6-1: Missing terms due to merged regions.

p_{ij}	Regions Affecting Sum 1 2 3 (1,2)	b_i
p_{1j}	x x x	2
p_{2j}	x x x	2
p_{3j}	x x x x	1
$p_{(1,2)j}$	x x	3

Example:

Figure 6-5 shows that the relaxation algorithm can use the context of the scene to favor the mapping of a merged region over those of non-merged regions that happen to have higher initial node weights.

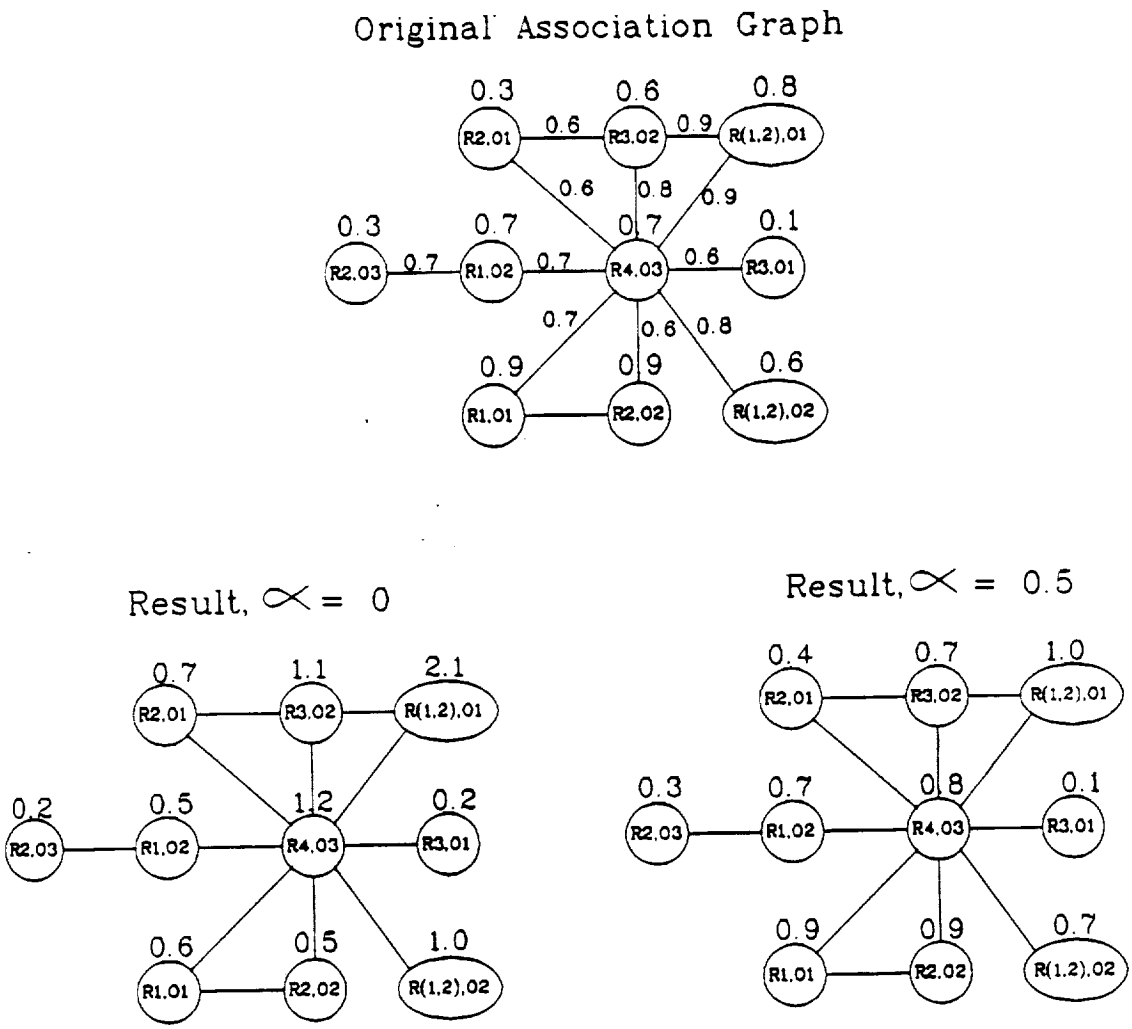


Figure 6-5: Example of relaxation process including a merged region.

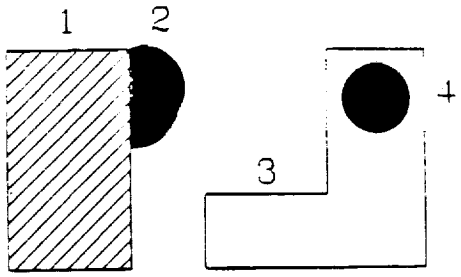
In this example, the weight of (R2,O1) is 0.3, (R1,O1) is 0.9, and (R(1,2),O1) is 0.8. But the weights on arcs connecting (R(1,2),O1) to the other correct nodes ((R4,O3) and (R3,O2)) have higher values than those for (R1,O1), indicating that within the context of the scene, the merged region is a better mapping. At $\alpha = 0$, the result of relaxation is that the clique containing (R(1,2),O1), (R4,O3), and (R3,O2) is the highest-valued. For a very high value of α, the initial node weights carry enough importance that the clique containing (R1,O1), (R2,O2), and (R4,O3) becomes the highest-valued.

6.3 Simulation with Split and Merged Regions

The procedure described above was performed on the hypothetical scene and the four inexact segmentations shown in Figure 6-6. The attributes used for description of regions were intensity, texture, and area. The real-valued relations used were the angle that a line between region centroids makes with the horizontal, and the percentage of region boundary adjacent to another region. For Cases 1 and 4, the intensity value for Object 2 was set to 0.1. For Cases 2 and 3, in which the hypothetical segmentation has merged Objects 1 and 2, the intensity for Object 2 was set to 0.5, a more plausible value, since the intensities of Objects 1 and 2 would likely be similar if the two objects were merged.

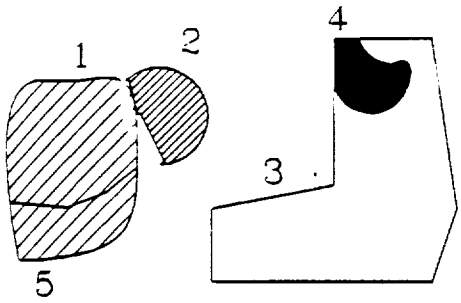
The results for Case 1, in which Object 1 is split into Regions 1 and 5, are shown in Table 6-2. The process correctly mapped Region 6, which includes Regions 1 and 5, to Object 1. Even if the initial values of (R1,O1) and (R6,O1) are switched, the improved context support for (R6,O1) causes that mapping to be favored.

Scene Model

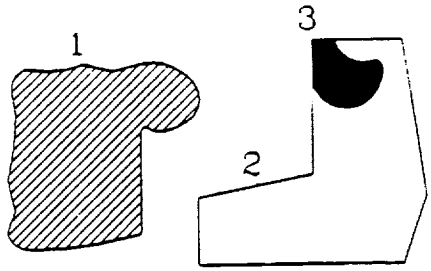


Observed Scenes

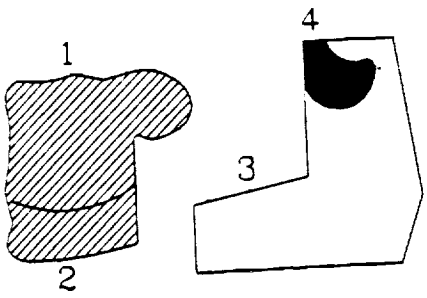
Case 1: Split region.



Case 2: Merged region.



Case 3: Split and merged regions.



Case 4: Three-way split region.

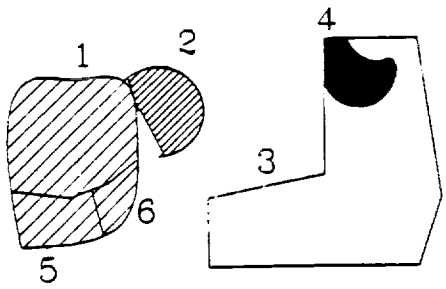


Figure 6-6: Hypothetical scene and observed scenes with split or merged regions.

Table 6-2: Simulation results; Case 1, split Object 1.

ATTRIBUTES

OBJECT	Area	Intensity	Texture
1	1.83	0.7	0.8
2	0.23	0.1	0.5
3	1.82	1.0	0.0
4	0.25	0.1	0.2
REGION			
1	0.98	0.6	0.85
2	0.38	0.2	0.4
3	2.22	0.9	0.05
4	0.28	0.02	0.35
5	0.46	0.65	0.8
6	1.44	0.62	0.83

RELATIONS

OBJECTS

Angle Relation

	1	2	3	4
1	0	37	355	10
2	217	0	335	0
3	175	155	0	65
4	190	180	245	0

REGIONS

Angle Relation

	1	2	3	4	5	6
1	0	25	348	10	270	0
2	205	0	330	7	230	217.5
3	168	150	0	85	187	177.5
4	190	187	265	0	205	197.5
5	90	50	7	25	0	0
6	0	37.5	357.5	17.5	0	0

Adjacency Relation

	1	2	3	4
1	0	0.09	0	0
2	0.24	0	0	0
3	0	0	0	0.2
4	0	0	1	0

Adjacency Relation

	1	2	3	4	5	6
1	0	0.01	0	0	0.28	0
2	0.09	0	0	0	0	0.09
3	0	0	0	0.28	0	0
4	0	0	0.6	0	0	0
5	0.35	0	0	0	0	0
6	0	0.01	0	0	0	0

Initial node weights		Final node weights: Alpha = 0.2	Initial weights of (R1,01), (R6,01) switched
(R1,01)	0.80	0.828	0.857
(R1,02)	0.57	-	-
(R1,03)	0.39	-	-
(R1,04)	0.45	-	-
(R2,01)	0.43	-	-
*(R2,02)	0.90	0.979	0.979
(R2,03)	0.33	-	-
(R2,04)	0.87	0.638	0.639
(R3,01)	0.57	-	-
(R3,02)	0.22	-	-
*(R3,03)	0.88	0.979	0.979
(R3,04)	0.34	-	-
(R4,01)	0.33	-	-
(R4,02)	0.91	0.628	0.629
(R4,03)	0.27	-	-
*(R4,04)	0.91	0.983	0.984
(R5,01)	0.75	0.796	0.797
(R5,02)	0.66	-	-
(R5,03)	0.34	-	-
(R5,04)	0.54	-	-
*(R6,01)	0.89	1.082	1.050
(R6,02)	0.00	-	-
(R6,03)	0.00	-	-
(R6,04)	0.00	-	-

* Desired mappings - Initial weight too low to consider

Table 6-3 shows the results for Case 2, in which Objects 1 and 2 are merged into Region 1. Again, the process correctly mapped Region 1 to Object 5, which includes Objects 1 and 2. The mapping of Region 1 to Object 1 also attained a high value, since Region 1 matches quite well to Object 1 alone, and its centroid is close to the centroid of Object 1, which leads to high arc values for the node (R1,O1).

Table 6-4 contains the results for Case 3, in which Objects 1 and 2 are merged into Region 1, and Object 1 is split into Regions 1 and 2. The mapping desired in this case is Region 5 (Regions 1 and 2) to Object 5 (Objects 1 and 2). At $\alpha = 0.2$, the mapping (R5,O1) is wrongly favored. At $\alpha = 0.25$, the correct mapping (R5,O5) prevails.

Table 6-5 contains the results for Case 4, in which Object 1 has been split into three regions, Regions 1, 5, and 6. This example illustrates the way that more than two regions may be merged into one. Regions are merged pairwise, and the new regions are added to the list of regions, so that they are considered for future merges. In this example, Region 7 is the merge of Regions 1 and 5; Region 8 is the merge of Regions 6 and 1, Region 9 is the merge of Regions 6 and 5; and Region 10 is the merge of Regions 8 (6 and 1) and 5. Of all nodes mapping to Object 1, the node (R10,O1) has the highest weight after relaxation. In each of the examples with $\alpha = 0.25$, the clique with the highest node sum after relaxation represents the correct mapping.

Table 6-3: Simulation results; Case 2, merged Objects 1 and 2.

ATTRIBUTES

OBJECT	Area	Intensity	Texture
1	1.83	0.7	0.8
2	0.23	0.5	0.5
3	1.82	1.0	0.0
4	0.25	0.1	0.2
5	2.06	0.68	0.77
REGION			
1	2.02	0.65	0.8
2	2.22	0.9	0.05
3	0.28	0.02	0.35

RELATIONS

OBJECTS — Angle Relation

	1	2	3	4	5
1	0	37	355	10	0
2	217	0	335	0	0
3	175	155	0	65	165
4	190	180	245	0	185
5	0	0	345	5	0

REGIONS — Angle Relation

	1	2	3
1	0	350	12
2	170	0	85
3	192	265	0

Adjacency Relation

	1	2	3	4	5
1	0	0.09	0	0	0
2	0.24	0	0	0	0
3	0	0	0	0.2	0
4	0	0	1	0	0
5	0	0	0	0	0

Adjacency Relation

	1	2	3
1	0	0	0
2	0	0	0.28
3	0	0.6	0

	Initial node weights	Final node weights Alpha = 0.2	Alpha = 0.25
(R1,01)	0.95	1.009	1.004
(R1,02)	0.53	-	-
(R1,03)	0.53	-	-
(R1,04)	0.28	-	-
Merge object 05=01+02			
*(R1,05)	0.97	1.010	1.006
(R2,01)	0.57	-	-
(R2,02)	0.36	-	-
*(R2,03)	0.88	0.948	0.943
(R2,04)	0.34	-	-
(R2,05)	0.00	-	-
(R3,01)	0.33	-	-
(R3,02)	0.77	0.556	0.575
(R3,03)	0.27	-	-
*(R3,04)	0.91	0.957	0.953
(R3,05)	0.00	-	-

* Desired mappings - Initial weight too low to consider

Table 6-4: Simulation results; Case 3, split Object 1, merged Objects 1 and 2.

ATTRIBUTES

OBJECT	Area	Intensity	Texture
1	1.83	0.7	0.8
2	0.23	0.5	0.5
3	1.82	1.0	0.0
4	0.25	0.1	0.2
5	2.06	0.68	0.77
REGION			
1	1.48	0.65	0.6
2	0.5	0.7	0.7
3	2.22	0.9	0.05
4	0.28	0.02	0.35
5	1.98	0.66	0.63

RELATIONS

OBJECTS

Angle Relation

	1	2	3	4	5
1	0	37	355	10	0
2	217	0	335	0	0
3	175	155	0	65	165
4	190	180	245	0	185
5	0	0	345	5	0

Adjacency Relation

	1	2	3	4	5
1	0	0.09	0	0	0
2	0.24	0	0	0	0
3	0	0	0	0.2	0
4	0	0	1	0	0
5	0	0	0	0	0

REGIONS

Angle Relation

	1	2	3	4	5
1	0	265	346	3	0
2	85	0	5	35	0
3	166	185	0	85	175.5
4	183	215	265	0	199
5	0	0	355.5	19	0

Adjacency Relation

	1	2	3	4	5
1	0	0.21	0	0	0
2	0.33	0	0	0	0
3	0	0	0	0.28	0
4	0	0	0.6	0	0
5	0	0	0	0	0

Initial node weights		Final node weights	
		Alpha = 0.2	Alpha = 0.25
(R1,01)	0.85	0.762	0.772
(R1,02)	0.70	0.715	0.715
(R1,03)	0.59	-	-
(R1,04)	0.45	-	-
Merge object 05=01+02			
(R1,05)	0.83	0.765	0.773
(R2,01)	0.74	0.714	0.717
(R2,02)	0.81	0.716	0.726
(R2,03)	0.40	-	-
(R2,04)	0.56	-	-
(R2,05)	0.71	0.692	0.695
(R3,01)	0.57	-	-
(R3,02)	0.36	-	-
*(R3,03)	0.88	0.922	0.919
(R3,04)	0.34	-	-
(R3,05)	0.00	-	-
(R4,01)	0.33	-	-
(R4,02)	0.77	0.667	0.675
(R4,03)	0.27	-	-
*(R4,04)	0.91	0.918	0.917
(R4,05)	0.00	-	-
Merge region R5=R1+R2			
(R5,01)	0.89	1.067	1.047
(R5,02)	0.60	-	-
(R5,03)	0.00	-	-
(R5,04)	0.00	-	-
*(R5,05)	0.93	1.068	1.052

*Desired mappings - Initial weight too low to consider

Table 6-5: Simulation results; Case 4, three-way split of Object 1.

ATTRIBUTES

OBJECT	Area	Intensity	Texture
1	1.83	0.7	0.8
2	0.23	0.1	0.5
3	1.82	1.0	0.0
4	0.25	0.1	0.2
REGION			
1	0.98	0.6	0.85
2	0.38	0.2	0.4
3	2.22	0.9	0.05
4	0.28	0.02	0.35
5	0.29	0.65	0.8
6	0.17	0.7	0.75
7	1.27	0.61	0.84
8	1.15	0.61	0.84
9	0.46	0.67	0.78
10	1.44	0.62	0.83

RELATIONS

OBJECTS

Angle Relation

	1	2	3	4
1	0	37	355	10
2	217	0	335	0
3	175	155	0	65
4	190	180	245	0

REGIONS

Angle Relation

	1	2	3	4	5	6	7	8	9	10
1	0	25	348	10	265	283	0	0	274	0
2	205	0	330	7	225	240	215	222.5	232.5	223.75
3	168	150	0	85	185	180	176.50	174	182.5	179.5
4	190	187	265	0	205	205	197.50	197.5	205	201.25
5	85	45	5	25	0	25	0	55	0	0
6	103	60	0	25	205	0	154	0	0	0
7	0	35	356.5	17.5	0	334	0	0	0	0
8	0	42.5	354	17.5	235	0	0	0	0	0
9	94	52.5	2.5	25	0	0	0	0	0	0
10	0	43.75	359.5	21.25	0	0	0	0	0	0

Adjacency Relation

	1	2	3	4
1	0	0.09	0	0
2	0.24	0	0	0
3	0	0	0	0.2
4	0	0	1	0

Adjacency Relation

	1	2	3	4	5	6	7	8	9	10
1	0	0.01	0	0	0.16	0.12	0	0	0.28	0
2	0.09	0	0	0	0	0	0.09	0.09	0	0.09
3	0	0	0	0.28	0	0	0	0	0	0
4	0	0	0.60	0	0	0	0	0	0	0
5	0.28	0	0	0	0	0.20	0	0.48	0	0
6	0.32	0	0	0	0.25	0	0.57	0	0	0
7	0	0.01	0	0	0	0.19	0	0	0	0
8	0	0.01	0	0	0.22	0	0	0	0	0
9	0.38	0	0	0	0	0	0	0	0	0
10	0	0.01	0	0	0	0	0	0	0	0

Table 6-5 (continued)

Initial node weights		Final node weights	
		Alpha = 0.2	Alpha = 0.25
(R1,01)	0.80	0.656	0.674
(R1,02)	0.57	-	-
(R1,03)	0.39	-	-
(R1,04)	0.45	-	-
(R2,01)	0.43	-	-
*(R2,02)	0.90	0.920	0.920
(R2,03)	0.33	-	-
(R2,04)	0.87	0.721	0.733
(R3,01)	0.57	-	-
(R3,02)	0.22	-	-
*(R3,03)	0.88	0.945	0.940
(R3,04)	0.34	-	-
(R4,01)	0.33	-	-
(R4,02)	0.91	0.719	0.733
(R4,03)	0.27	-	-
*(R4,04)	0.91	0.942	0.941
(R5,01)	0.73	0.621	0.634
(R5,02)	0.69	-	-
(R5,03)	0.31	-	-
(R5,04)	0.57	-	-
(R6,01)	0.70	0.609	0.621
(R6,02)	0.69	-	-
(R6,03)	0.33	-	-
(R6,04)	0.57	-	-
Merge region R7=R5+R1			
(R7,01)	0.86	0.940	0.934
(R7,02)	0.00	-	-
(R7,03)	0.00	-	-
(R7,04)	0.00	-	-
Merge region R8=R6+R1			
(R8,01)	0.84	0.930	0.923
(R8,02)	0.00	-	-
(R8,03)	0.00	-	-
(R8,04)	0.00	-	-
Merge region R9=R6+R5			
(R9,01)	0.75	0.871	0.859
(R9,02)	0.00	-	-
(R9,03)	0.00	-	-
(R9,04)	0.00	-	-
Merge region R10=R8+R5			
*(R10,01)	0.90	1.179	1.140
(R10,02)	0.00	-	-
(R10,03)	0.00	-	-
(R10,04)	0.00	-	-

* Desired mappings - Initial weight too low to consider

To test the ability of the relaxation process to determine the correct rotation of an observed scene, the scene of Case 1 was processed, with different rotation angles at steps of 20 degrees. Table 6-6 shows the sum of nodes in the highest-valued clique found for each angle.

Table 6-6: Sums of highest cliques, 18 rotations of Case 1.

Angle	Sum	Angle	Sum
0	3.985	180	1.795
20	2.972	200	1.816
40	3.124	220	1.837
60	2.999	240	1.857
80	2.099	260	1.988
100	2.099	280	2.572
120	1.843	300	2.250
140	1.823	320	4.127
160	1.801	340	4.035

This example shows that this process for rotation normalization is not foolproof. There were two rotation angles, 320 and 340 degrees, at which the sums of nodes in the highest clique were greater than at 0 degrees, the correct rotation. These results would seem to indicate that the rotation of 320 degrees yields the best match of the observed scene to the stored model. To understand why this result came about, it is helpful to examine the arcs of the association graphs at rotations of 0 and 320 degrees. At 0 degrees, the incorrect mapping (R4,O2) happens to have high enough compatibility with (R1,O1), (R5,O1), and (R6,O1) that it has arcs in the association graph connecting it with these three nodes. Thus, in the updating formula, the node (R4,O2) is receiving contributions from Regions 1, 5, and 6. So, at 0 degrees rotation, the node

(R4,O2) has a final weight of 0.651. In contrast, at 320 degrees, the node (R4,O2) has no arcs, and at 340 degrees, it has two rather than three. This leads node (R4,O2) to have a lower weight at these rotations, and so the other nodes have relatively higher weights. The initial node weights, and the final weights for rotations of 0, 320, and 340 degrees, are shown in Table 6-7.

Table 6-7: Initial and final node weights, three rotations of Case 1.

Initial Node Weights	Final Node Weights		
	0°	320°	340°
(R1,O1) 0.80	0.828	0.850	0.820
* (R2,O2) 0.90	0.973	0.912	0.975
(R2,O4) 0.87	0.657	0.761	0.693
* (R3,O3) 0.88	0.971	1.024	0.984
(R4,O2) 0.91	0.651	0.310	0.539
* (R4,O4) 0.91	0.977	1.043	1.002
(R5,O1) 0.75	0.794	0.866	0.827
* (R6,O1) 0.89	1.064	1.148	1.074
*Desired mappings			

VII. PROOF OF CONCEPT

The ideas developed here were implemented in a demonstration object location system for NASA. The object of the work was to explore some of the issues involved in implementing a machine vision system to be used in conjunction with a robot arm in the space station laboratory module. Section 7.1 provides background on possible uses of machine vision in the space station environment. The demonstration system developed for the project is described in Section 7.2. Section 7.3 describes the relaxation process applied to an example scene used in the demonstration system.

7.1 Background

There is interest at NASA in investigating the use of vision systems to automate routine space station operations, relieving crewmen of repetitive tasks. Within the space station modules, a vision intelligent robot could be used for operations such as location, fetching, storing, and repairing. Vision systems could be used to monitor experiments, record data, and alert crewmen only when necessary.

Vision systems would also be useful outside the space station, for applications such as orbital docking, servicing, and assembly. The use of vision in docking is particularly beneficial, since it eliminates communication time delay for vehicle control. NASA is also investigating the possibility of using vision

systems for weather prediction by tracking cloud motion. In short, reliable vision systems are needed to automate space station and other advanced NASA operations.

A necessary component of any machine vision system is the ability to locate objects of interest. It is this component that was explored in detail in this research.

7.2 Description of System

The system, developed on a commercially available image processing system (Perceptics), with the relaxation algorithm developed on an IBM PC, demonstrates the object location component of a machine vision system. Given a request to find a particular object, the system returns the location of the object.

The assumptions made in developing the demonstration system were as follows:

1) The robot arm's exact position may be unknown.
2) The interior of the laboratory module is broken into several separate scenes, or panels.
3) All of a panel will appear in the field-of-view at once.
4) There will be changes in scale, translation, and intensity, and slight changes in rotation and tilt.

The steps in the processing are as follows:

1) Identify the panel present in the input scene. This is done by searching for prominent features that distinguish among scenes. Low-level features, such as 'two bright, nearly horizontal lines,' or 'ten nearly vertical lines of high texture' are used.

2) Based on the approximate location of the distinguishing features, find the region of interest on the panel that contains the desired object. This is done to cut down on the number of objects that must be dealt with at one time.

3) Run the segmentation process provided with the Perceptics. This process produces a list of regions found in the image. Each region's location, perimeter, area, length, height, and circularity are listed, along with some other attributes.

4) For objects that lie along rows and columns of objects on the panels, a simplified object location process is performed. This process finds probable locations of objects, based on intersection points of lines of high texture (intensity variation). Then, regions found by the segmentation process are matched with these objects.

5) For objects that cannot be located by the simple process of Step 4, the relaxation process is run on the remaining regions to find the best matches between regions and objects.

7.3 Relaxation Process

The relaxation process described in Chapters V and VI was tested on an actual example of a scene having segmentation errors. The scene, annotated with object numbers, is shown in Figure 7-1.

The attributes used to describe the objects were intensity, texture (variance of intensity), extent (height × width), and elongation (width/height). In addition, the pixel coordinates of the regions were used, in order to rule out mappings of objects to regions which are very far away from their approximate expected location.

The real-valued positional relation of angle between centroids was used. Also, a binary relation, 'nearby,' was defined such that Region A is nearby Region B iff the distance between their centroids is less than a threshold based on the sum of the heights and widths of the two objects. Both relations were used in determining association graph arc values. Additionally, the 'nearby' relation was used in the procedure to merge oversegmented regions: two regions could not be merged unless the relation 'nearby' existed between them.

The practical considerations of running the relaxation procedure on a typical real-world problem became apparent in running this example, since the stored scene had 13 objects and the observed scene contained 50 regions. The example exhibited many problems of mismeasured region boundaries and extra regions. There was also one missing object and several objects that had been split into multiple regions by the segmentation process.

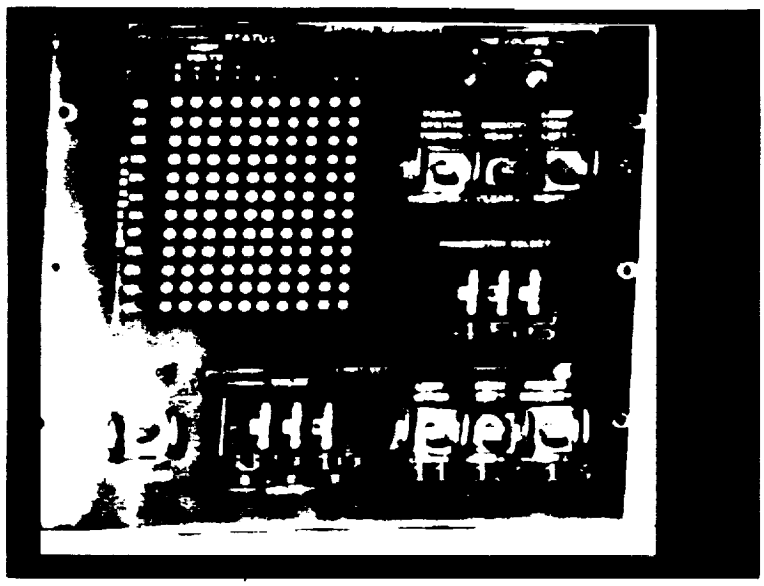


Figure 7-1: Space shuttle simulator panel used as example scene.

Three different ideas were explored with this example. First, using the process described in Chapter V (no merging of regions), two forms of the updating rule were tried. Then, the value of α was tested, at 0 and 0.25. Then, the ideas of Chapter VI were applied to this example, to show how the procedure can handle some real-world examples of over-segmentation.

The results of the various experiments with the relaxation process bear out the previous conclusions regarding attribute and relation selection. Due to practical considerations, the attributes used were not ideal. Because of segmentation errors, the attributes that were based on perimeter were unreliable, and were not used. Attributes of intensity and texture (variance of intensity over a region's area) were of limited usefulness. In the scenes being used, the various objects all had approximately the same intensity and texture. The corresponding observed regions did not have intensity values that matched the objects well, because the segmentation process is based on finding regions of high intensity. The segmentation algorithm tends to outline the brightest parts of the objects, and to go around the shadows or darker parts. The variations in these values could not be used to narrow down mappings of regions to objects, except to help in ruling out the mappings of some extra regions to objects.

7.3.1 Two Forms of Updating Rule

One issue that was brought to light with this example is the form of the updating rule. In the formulation used in previous chapters, a node's weight is updated by the highest contribution (arc weight × node weight) for each *region*. This original updating rule is:

$$p_{ij}^{(r+1)} = \alpha p_{ij}^{(0)} + (1-\alpha)\left[\frac{1}{n}\sum_{h=1}^{n}\left[\max_{k=1}^{m}(p_{hk}^{(r)}c(i,j;h,k))\right]\right]. \quad (6\text{-}1)$$

However, in this example, there are many extra regions. Any contributions to a node's weight from the nodes involving these extra regions are spurious, so it was expected that the result would be negatively affected by this problem. So, a comparison was done between the original updating rule (above) and the following rule:

$$p_{ij}^{(r+1)} = \alpha p_{ij}^{(0)} + (1-\alpha)\left[\frac{1}{m}\sum_{k=1}^{m}\left[\max_{h=1}^{n}(p_{hk}^{(r)}c(i,j;h,k))\right]\right]. \quad (6\text{-}2)$$

The change is that the sum has one term for each *object*, and the maximum is taken over the n *regions* for each given *object*.

Figure 7-2 shows the result of the original rule, Equation 7-1. Regions are labeled with the object number of Figure 7-1 to which they were found to map by the relaxation algorithm. (The oversegmented objects, numbers 1, 3, 11, and 13, were not attempted.) There are several errors in the favored mappings found by the original rule. An extra object in the upper right corner of the picture has been wrongly labeled as Object 2. Also, the wrong region has been labeled as Object 8. There was also an error on Object 12. The region corresponding to the printing on the panel under the switch was labeled as Object 12. Figure 7-3 shows the result of using Equation 7-2 as the updating rule. All the mappings that were found were correct. This result suggests that if many extra regions are expected, given a particular application

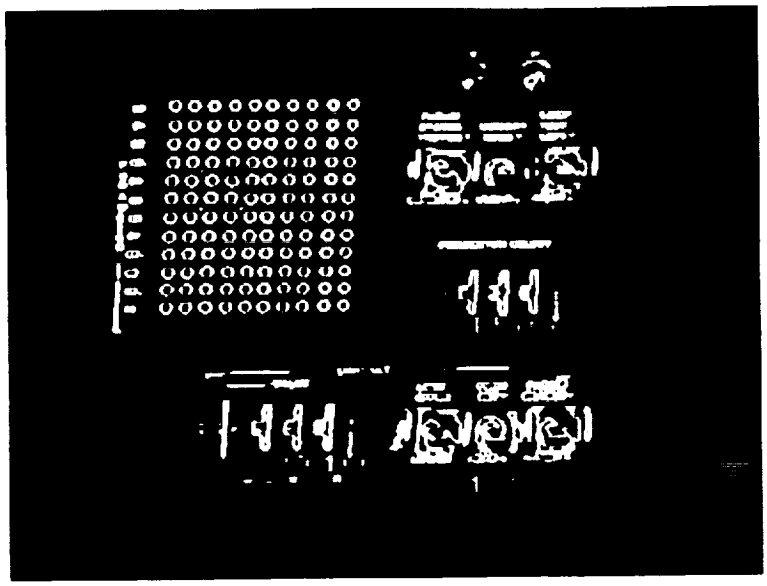


Figure 7-2: Object assignments resulting from using the original updating rule.

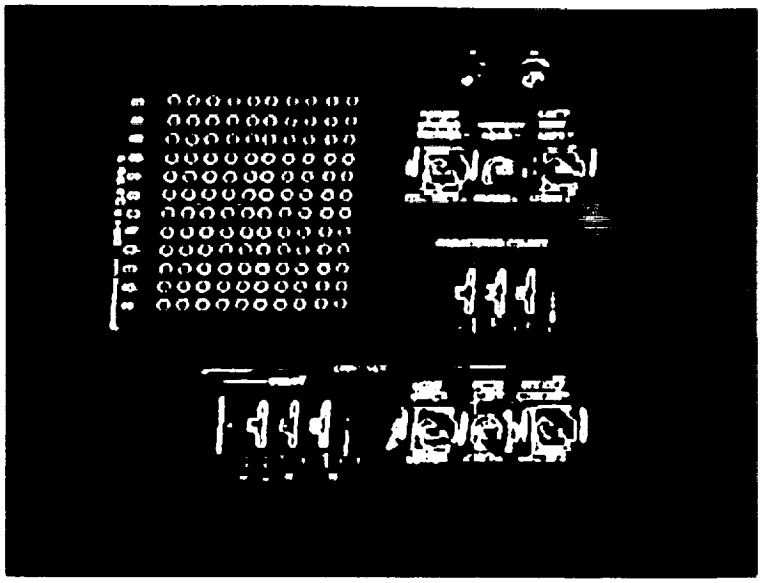


Figure 7-3: Object assignments resulting from using the modified updating rule.

and segmentation algorithm, it is better to use Equation 7-2 as the updating rule. If the segmentation algorithm is expected to miss many objects, the original rule of Equation 7-1 is preferred.

7.3.2 Value of Alpha

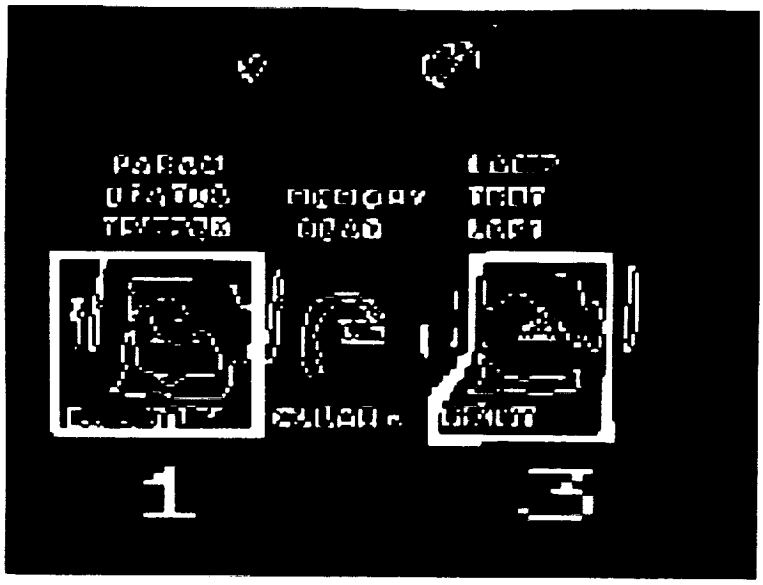
Another experiment explored using two values of α: 0 and 0.25. In this application, the attributes that were readily measured with the system being used were not very effective in distinguishing among the different types of objects or distinguishing the extra objects from the stored objects. So, it was expected that using a lower value of α would provide a better result. The updating rule of Equation 7-2 was used, with α set to 0 and 0.25. (Oversegmented objects, again, were not attempted.) Both led to correct mappings for objects, as shown in Figure 7-3. The sum of the node weights of the favored nodes for each object was 7.866 at $\alpha = 0.25$, and 8.164 at $\alpha = 0$. This indicates that in this application, since the relation values were more reliable than the attribute values, a lower value of α provided a less ambiguous result.

7.3.3 Handling Oversegmentation

This example contained several instances of oversegmented objects. Because of the relatively large number of regions, this posed a practical problem in running the relaxation process on a PC, in terms of memory size and time requirements. Judicious selection of thresholds on the initial node weights and arc values was necessary to keep the number of arcs in the association graph down to a manageable level. In this example, 77 new regions were created by the merging process. The resulting association graph contained 7,248 arcs above the arc weight threshold.

Two regions were considered for merging if an object had all its initial node weights below a threshold, and if the two regions were nearby and had similar region-wide attributes (intensity and texture), and if the node representing the mapping of the merged region to the object had a weight above a threshold. New regions created by this process were added to the list of regions, and later considered for further merges with other regions, thus allowing for merges of more than two regions.

The results of the relaxation process on the oversegmented objects are shown in Figure 7-4. (Results for objects which were not oversegmented were the same as in Figure 7-3.) The merged region that mapped to Object 1 consisted of three regions: the main part of Object 1, the region corresponding to the left bracket of Object 1, and the region corresponding to the line of printing under the object. For Object 3, the top and bottom portions of the object, along with the printing under it, merged to form the region that was favored by the relaxation process. The left bracket and the main part of Object 11 were mapped to that object. The main part of Object 13 merged with the printing above the object to form the region that mapped best to Object 13. In each case, there were some errors, with regions that could legitimately be considered part of an object not included in the favored regions, or extra objects (printing) being included. However, the results are reasonable, as can be seen by examining the original scene of Figure 7-1. In some cases, it is difficult to tell where an object ends and the printing below it begins.



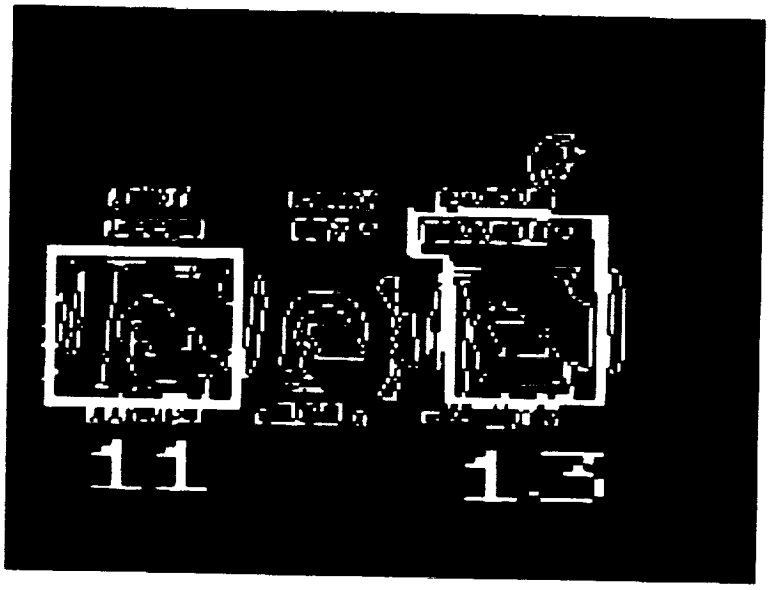


Figure 7-4: Object assignments obtained by merging oversegmented regions.

VIII. CONCLUSION AND RECOMMENDATIONS

8.1 Conclusion

This report describes an approach to inexact scene matching in which we can find the 'best' mapping of observed regions to stored objects, given degrees of closeness in region-object matches and in relation matches. This is important since segmentation errors render exact matches impossible.

The selection of attributes and relations for the description of scenes was discussed in some detail. Real-valued relations have proven to be very helpful in inexact scene matching, because they allow for measurements of the closeness of match between expected and observed relations.

Previous work has demonstrated the use of clique finding within association graphs to model the determination of the 'best' region-to-object mapping. However, that work was limited by the use of 'yes or no' choices in that it was only specified that a region could or could not map to a given object, and that two mappings were or were not compatible with each other. The work presented here extends the use of the clique finding matching approach by weighting the nodes and arcs of the association graph to allow for the quantification of *how* good a mapping is, and *how* compatible two mappings are.

Applying relaxation to the weighted association graph simplifies the selection of the best clique by eliminating nodes that have low values after

relaxation, and by allowing the cliques to be evaluated by examining node weights only and ignoring arc weights, as the relaxation process incorporates the contextual information into the node weights. As mentioned in Chapter IV, relaxation has been used in conjunction with clique finding in association graphs for boundary matching, but in that research, discrete relaxation was used only to reduce the size of the association graph and not to simplify the evaluation of cliques.

Most previous uses of relaxation for scene matching have not been incorporated into the framework of clique finding in association graphs. After the relaxation process, the result is obtained by selecting the highest valued node for each object. This method does not deal with missing or merged objects, and does not handle the possibility that these highest-valued nodes may not all be compatible with one another. Also, previous uses of relaxation for scene matching or similar problems have not used an updating rule that allows for the balancing of importance of initial node weights (local information) and arc weights (contextual information). In scene matching, it is essential to take both the local and contextual information into account when deciding on the best mapping.

A means of handling problems of oversegmentation and undersegmentation has also been presented here. By identifying possible split or merged regions and adding merge nodes to the association graph, mappings of one region to multiple objects, one object to multiple regions, and even multiple regions to multiple objects are possible with little modification to the relaxation process.

8.2 Recommendations

There are many opportunities for future research using this work as a foundation. The major areas for further work are listed below:

1) There is a need for research into the determination of the various thresholds and parameters that must be set properly in order to produce good results. These thresholds and parameters are listed and explained below.

Initial mapping threshold: Nodes with weights below this threshold are dropped from consideration. The threshold should be low enough that it does not eliminate correct mappings with low initial weights due to segmentation errors. But it should be high enough to eliminate unnecessarily processing bad mappings, and not to allow a node with a very low initial weight to be wrongly favored by the relaxation process because it happens to have good contextual support.

Arc weight threshold: Since there are weights on the arcs of the association graph, these values must be thresholded so that only the nodes connected by arcs with values above the threshold are considered compatible. The arc weight threshold should be low enough to retain compatibility between mappings that are indeed compatible but may have a low arc weight due to mismeasured relations. The threshold should be high enough not to assume compatibility between two mappings that are incompatible. In the evaluation of the association graph that results after relaxation, having fewer arcs will result in smaller cliques.

Merge consideration threshold: In determining whether to look for possible regions to merge, the algorithm looks for an object that has all low node weights. The merge consideration threshold is the value below which the node weights are considered to be low. This threshold should be high enough to consider merges even when an initial node weight is somewhat high: it could be falsely high, or context support could be higher for a merged region. The threshold should be low enough so that time is not wasted considering merges for regions/objects that already have good matches. (In the interest of accuracy, this threshold can be set high, so that more possibilities of merged regions/objects are considered.)

Similarity threshold: In order to merge two regions, their similarity (the 'node weight' achieved by comparing their region-wide attributes) must be above this threshold, and the similarity of each region to the object that prompted the merge attempt must be above the threshold. If this threshold is too low, there will be a proliferation of merge nodes, increasing the size of the association graph and the processing required, and increasing the chance of an incorrect merged region/object being wrongly favored by the relaxation process. If the threshold is too high, the process fails to merge regions/objects that should be merged.

The parameter α: As discussed previously, α provides a balance between the importance of the initial node weights (local compatibility) and the association graph arc values (contextual support). If α is too low, a good mapping may have a low node weight after relaxation because of

mismatches in relations due to segmentation errors. If α is too high, the contextual support for a mapping with a low initial weight due to segmentation errors will not pull the node's value up sufficiently.

Final node threshold: After relaxation, any nodes with weights below this threshold are eliminated from consideration. If the threshold is too low, an incorrect node with a very low weight could be included in the clique representing the best mapping. This node should not be thought of as adding to the merit of the mapping. If the threshold is too high, a correct region-object mapping may be eliminated.

In testing the algorithm, when a correct mapping fails to be favored or an incorrect mapping is improperly favored, it is relatively easy to determine what sort of adjustment to thresholds or parameters would improve the result. In applying this algorithm to a particular type of scene with a given segmentation algorithm, a useful enhancement would be a system for tuning these thresholds and parameters by the use of a set of stored scenes and observed scenes exhibiting segmentation errors typical of those expected in actual use. The system would perform the algorithm on the sample scenes. When errors occur, the system would determine what adjustment would be helpful in improving the result. For example, if a node that should have a high weight after relaxation has a low weight, and if its initial weight is high, increasing the value of α could improve the result. If a merged region should have been created to map to a particular object, it could be that the similarity threshold was too high.

2) Another area in which further research is needed is in parallel implementations of the algorithms described here. All of the procedures involved, including measurement of attributes and relations, determination node and arc weights, determination of merge regions and objects, and the relaxation process itself, are computationally intensive. So, any implementation that does not make use of parallel processing is not likely to be practical on problems of typical size.

3) In the handling of split or merged objects, finding candidate regions or objects for merging by analyzing the *initial* node weights before relaxation may not be the best approach. Because contextual information is not considered, many unnecessary merges may be considered, and some necessary ones may be overlooked. A better approach may be to run the relaxation algorithm on the original regions and objects first, assuming that there are no split or merged objects. Then, the *final* node weights may be analyzed to determine if the relaxation procedure should be performed again, on an association graph to which merge nodes have been added.

4) Another area in which there are many possibilities for more research is the use of this matching approach in conjunction with primitives, attributes, and relations different than those used here. The use of image regions as primitives is appropriate for applications in which it is expected that whole regions will usually be visible in the camera's field of view. In applications such as aerial photographs, the use of segments of the curves that constitute the region boundaries may be more appropriate, because often only parts of

regions are within the field of view of the camera. Using generalized relations, which relate *groups* of objects, rather than simply *pairs*, may be useful in handling problems such as rotation.

5) The use of relaxation in conjunction with clique finding in association graphs may have applications in areas that are unrelated to scene matching. This approach may be useful any time a mapping of units to classes is sought, in which compatibility of individual mappings with each other is important and many-to-many mappings are possible.

REFERENCES

[1] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall and R.J. Popplestone, "A versatile system for computer controlled assembly," *Artificial Intelligence*, Vol. 6, No. 2, 1975, pp. 129-396.

[2] H.G. Barrow and R.M. Burstall, "Subgraph isomorphism, matching relational structures and maximal cliques," *Information Processing Letters*, Vol. 4, No. 4, 1976, pp. 83-84.

[3] A.T. Berztiss, "A backtrack procedure for isomorphism of directed graphs," *Journal of the Association for Computing Machinery*, Vol. 20, No. 3, July 1973.

[4] R.C. Bolles, "Robust feature matching through maximal cliques," *Proceedings, Society of Photo-optical Instrumentation Engineers*, Vol. 182, April 1979, pp. 140-149.

[5] S.K. Chang, C.W. Yan, D.C. Dimitroff, and T. Arndt, "An intelligent image database system," *IEEE Transactions on Software Engineering*, Vol. 14, No. 5, 1988, pp. 681-688.

[6] D.G. Corneil and C.C. Gottlieb, "An efficient algorithm for graph isomorphism," *Journal of the Association for Computing Machinery*, Vol. 17, No. 1, Jan. 1970.

[7] L.S. Davis, "Shape matching using relaxation techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 1, January 1979, pp. 60-72.

[8] J.R. Englebrecht and F.M. Wahl, "Polyhedral object recognition using Hough-space features," *Pattern Recognition*, Vol. 21, No. 2, 1988, pp. 155-167.

[9] O.D. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 4, July 1981, pp. 412-424.

[10] O.D. Faugeras and K.E. Price, "Semantic description of aerial images using stochastic labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 6, November 1981, pp. 633-642.

[11] G. Fekete, J.O. Eklundh, and A. Rosenfeld, " Relaxation: Evaluation and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol PAMI-3, No. 4, September 1981, pp. 459-469.

[12] R.C. Gonzalez, and P. Wintz, Digital Image Processing, Second Edition, Addison-Wesley, 1987.

[13] R. Greene, "Scene knowledge representation for expert vision systems," Ph.D. dissertation, University of Alabama in Hunstville, 1989.

[14] R.A. Hummel and Rosenfeld, A., "Relaxation processes for scene labeling," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 10, October 1978, pp. 765-768.

[15] C.C. Hung, "Expert vision system for image segmentation," Ph.D. dissertation, University of Alabama in Hunstville, 1990.

[16] H.B. Mittal, "A fast backtrack algorithm for graph isomorphism," *Information Processing Letters*, Vol. 29, No. 2, 1988, pp. 105-110.

[17] J.A. Orenstein and F.A. Manola, "PROBE spatial data modeling and query processing in an image database application," *IEEE Transactions on Software Engineering*, Vol. 14, No. 5, 1988, pp. 611-629.

[18] T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, 1977.

[19] T. Pavlidis, "Structural descriptions and graph grammars," in S.K. Chang and K.S. Fu, ed., Pictorial Information Systems, Springer-Verlag, 1980, pp. 86-103.

[20] K.E. Price, "Hierarchical matching using relaxation," *Computer Vision, Graphics, and Image Processing*, Vol. 34, No. 1, April 1986, pp. 66-75.

[21] K.E. Price, "Relaxation matching techniques - A comparison," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 5, September 1985, pp. 617-623.

[22] B. Radig, "Image sequence analysis using relational structures," *Pattern Recognition*, Vol. 17, No. 1, 1984, pp. 161-167.

[23] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene labeling by relaxation operations," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-6, No. 6, 1976, pp. 420-433.

[24] A. Rosenfeld and A. Kak, Digital Picture Processing, Second Edition, Volume 2, 1982, pp. 152-184.

[25] L.G. Shapiro and R.M. Haralick, "A metric for comparing relational descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 1, 1985, pp. 90-94.

[26] L.G. Shapiro and R.M. Haralick, "Structural descriptions and inexact matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 5, Sept. 1981, pp. 504-519.

[27] W.S. Tsai and K.S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-9, No. 12, Dec. 1979.

[28] W.H. Tsai and K.S. Fu, "Subgraph error-correcting isomorphisms for syntactic pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 1, January/February 1983, pp. 48-62.

[29] J.R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the Association for Computing Machinery*, Vol. 23, No. 1, January 1976, pp. 31-42.

[30] S.H. Unger, "GIT - A heuristic program for testing pairs of directed line graphs for isomorphism," *Communications of the Association for Computing Machinery*, Vol. 23, No. 1, Jan. 1964.

[31] A.K.C. Wong and M. You, "Entropy and distance measures of random graphs," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, IEEE, 1983, pp. 371-376.

[32] B. Yang, W.E. Snyder, and G.L. Bilbro, "Matching oversegmented 3D images to models using association graphs," *Image and Vision Computing*, Vol. 7, No. 2, 1989, pp. 135-143.

[33] K.C. You and K.S. Fu, "A syntactic approach to shape recognition using attributed grammars," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-9, No. 6, June 1979, pp. 334-345.