4-1-1985

Image Understanding by Hierarchical Symbolic Representation and Inexact Matching of Attributed Graphs

Mohamed A. Eshera
Purdue University

Image Understanding by Hierarchical Symbolic Representation and Inexact Matching of Attributed Graphs

Mohamed A. Eshera

TR-EE 85-8
April 1985

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

This work was supported by the Office of Naval Research Contract No. ONR-N00014-79-C-0574 and the Air Force Office of Scientific Research through Honeywell Systems and Research Contract No. S49620-83-C-1034def.

IMAGE UNDERSTANDING
BY
HIERARCHICAL SYMBOLIC REPRESENTATION
AND INEXACT MATCHING OF ATTRIBUTED GRAPHS

Mohamed A. Eshera

TR-EE 85-8

April 1985

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

This work was supported by the Office of Naval Research Contract No. ONR-N00014-79-C-0574 and the Air Force Office of Scientific Research through Honeywell Systems and Research Contract No. S49620-83-C-1034def.

BIBLIOGRAPHIC DATA SHEET	1. Report No. TR-EE 85-8	2.	3. Recipient's Accession No.

4. Title and Subtitle
IMAGE UNDERSTANDING BY HIERARCHICAL SYMBOLIC REPRESENTATION AND INEXACT MATCHING OF ATTRIBUTED GRAPHS

5. Report Date
April 1985

6.

7. Author(s)
Mohamed A. Eshera

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Purdue University
School of Electrical Engineering
West Lafayette, IN 47907

10. Project/Task/Work Unit No.

11. Contract/Grant No.
ONR-N00014-79-C-0574
S49620-83-C-1034def

12. Sponsoring Organization Name and Address
Department of the Navy, Office of Naval Research, Arlington, VA 22217
Honeywell Systems and Research, 2600 Ridgway Parkway, NE, Minneapolis, MN 55413

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts
We study the symbolic representation of imagery information by a powerful global representation scheme in the form of Attributed Relational Graph (ARG), and propose new techniques for the extraction of such representation from spatial-domain images, and for performing the task of image understanding through the analysis of the extracted ARG representation.

To achieve practical image understanding tasks, the system needs to comprehend the imagery information in a global form. Therefore, we propose a multi-layer hierarchical scheme for the extraction of global symbolic representation from spatial-domain images. The proposed scheme produces a symbolic mapping of the input data in terms of an output alphabet, whose elements are defined over global subimages. The proposed scheme uses a combination of model-driven and data-driven concepts. The model-driven principle is represented by a graph transducer, which is used to specify the alphabet at each layer in the scheme. A symbolic mapping is driven by the input data to map

17. Key Words and Document Analysis. 17a. Descriptors (continued on next page)

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Unlimited	UNCLASSIFIED	230
	20. Security Class (This Page) UNCLASSIFIED	22. Price

the input local alphabet into the output global alphabet. Through the iterative application of the symbolic transformational mapping at different levels of hierarchy, the system extracts a global representation from the image in the form of attributed relational graphs. Further processing and interpretation of the imagery information can, then, be performed on their ARG representation.

We also propose an efficient approach for calculating a distance measure and finding the best inexact matching configuration between attributed relational graphs. For two ARGs, we define sequences of weighted error-transformations which when performed on one ARG (or a subgraph of it), will produce the other ARG. A distance measure between two ARGs is defined as the weight of the sequence which possesses minimum total-weight. Moreover, this minimum-total weight sequence defines the best inexact matching configuration between the two ARGs. The global minimization over the possible sequences is performed by a dynamic programming technique. The approach shows good results for ARGs of practical sizes.

The proposed system possesses the capability to inference the alphabets of the ARG representation which it uses. In the inference phase, the hierarchical scheme is usually driven by the input data only, which normally consists of images of model objects. It extracts the global alphabet of the ARG representation of the models. The extracted model representation is then used in the multi-layer shceme. We present our experimental results in utilizing the proposed system for locating objects in complex scenes.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

ABSTRACT

Eshera, Mohamed A., Ph.D., Purdue University. May 1985. Image Understanding by Hierarchical Symbolic Representation and Inexact Matching of Attributed Graphs. Major Professor: K. S. Fu.

We study the symbolic representation of imagery information by a powerful global representation scheme in the form of Attributed Relational Graph (ARG), and propose new techniques for the extraction of such representation from spatial-domain images, and for performing the task of image understanding through the analysis of the extracted ARG representation.

To achieve practical image understanding tasks, the system needs to comprehend the imagery information in a global form. Therefore, we propose a multi-layer hierarchical scheme for the extraction of global symbolic representation from spatial-domain images. The proposed scheme produces a symbolic mapping of the input data in terms of an output alphabet, whose elements are defined over global subimages. The proposed scheme uses a combination of model-driven and data-driven concepts. The model-driven principle is represented by a graph transducer, which is used to specify the alphabet at each layer in the scheme. A symbolic mapping is driven by the input data to map the input local alphabet into the output global alphabet. Through the iterative application of the symbolic transformational mapping at different levels of hierarchy, the system extracts a global representation from the image in the form of attributed relational graphs. Further processing and interpretation of the imagery information can, then, be performed on their ARG representation.

We also propose an efficient approach for calculating a distance measure and finding the best inexact matching configuration between attributed relational graphs. For two ARGs, we define sequences of weighted error-transformations which when performed on one ARG (or a subgraph of it), will produce the other ARG. A distance measure between two ARGs is defined as the weight of the sequence which possesses minimum total-weight. Moreover, this minimum-total weight sequence defines the best inexact matching configuration between the two ARGs. The global minimization over the possible sequences is performed by a dynamic programming technique. The approach shows good results for ARGs of practical sizes.

The proposed system possesses the capability to inference the alphabets of the ARG representation which it uses. In the inference phase, the hierarchical scheme is usually driven by the input data only, which normally consist of images of model objects. It extracts the global alphabet of the ARG representation of the models. The extracted model representation is then used in the operation phase of the system to perform the mapping in the multi-layer scheme. We present our experimental results for utilizing the proposed system for locating objects in complex scenes.

CHAPTER I

INTRODUCTION

1.1 General

Throughout ages, human's ambitious to utilize machines in performing varieties of tasks has never stoped or slowed down. Man has always been improving machines to increase their efficiency and expand their capabilities. Intelligence has been regarded as an attribute of human beings, and sometimes of some other living beings but with much lower degrees. Nevertheless, with the rapid development of new generations of digital computers, with their lightening computation speed and immense memory sites, the dream of having intelligent machines is getting closer to reality, if not already real, at least to some extent. During the past two decades, there has been rapidly growing and widely spreading interest in building machines that behave with more and more intelligence, [fuks83b], [fuks83c], [nils80].

The versatile capabilities of intelligent machines will be widely enhanced if equipped with an efficient and powerful computer vision and image understanding system, [kana79], [kana81b], [mats84], [tang80], [wins80]. The mutual interaction, if not overlapping,

between vision and intelligence comes from the need to understand what is seen and to see what is to be understood. In fact, a considerable part of the human brain is dedicated to his vision system. Vision is not merely receiving the falling light on the retina, but it is definitely more into understanding and comprehending what the retina receives. It is considered to be an important, if not the most important way of sensing and perceiving knowledge. Therefore, a major discipline in the area of machine intelligence deals with visual information processing through computer vision and image understanding systems.

An image understanding system considers the image as a quantitative description of a set of objects, [fuks82a], [asad84], [ball82]. A class of objects is a group of objects that share some common properties. The system receives the image data through its imaging devices, e.g., digital cameras, cameras followed by digitizers, or solid-state cameras, etc., [ball82], which sense the light on their receivers and converts it into digital signals in the form of an array of grey scale values. The main task of image understanding is to comprehend, analyze, or "understand", the visual information to produce a useful description of the input image for decision making and other automated tasks and operations, [agga81], [bhan83], [cowi83], [fang83], [fisc83], [fish83], [gonz82], [leeh83], [mart83], [perk78], [perk80], [shap83], [thor83], [trop83].

An image understanding system consists, in general, of two major stages, as shown in Figure 1.1. The first stage concerns mainly with the representation of the visual information of different objects in the image. The objective of this stage is to obtain an efficient, compact, yet, complete and adequate, form of

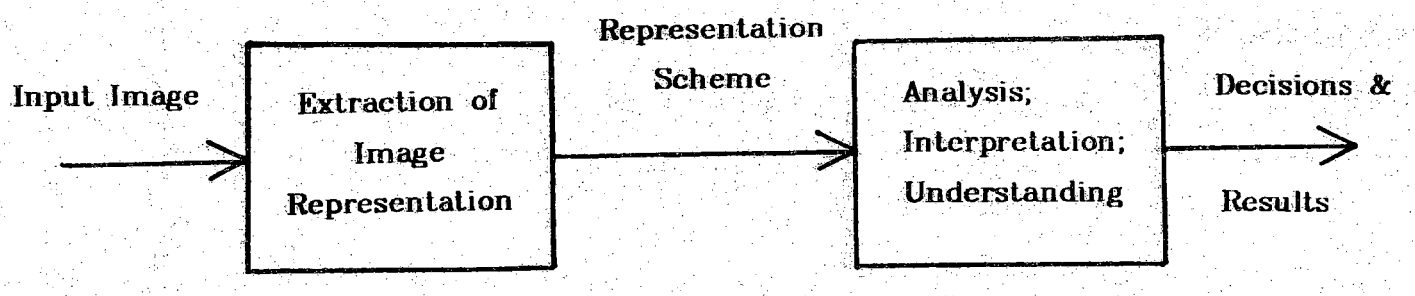


Fig. 1.1 Basic Components of a General Image Understanding System

representation for the visual knowledge, through which digitized images can be stored in the computer memory to facilitate their further analysis. The actual analysis and interpretation of the images are performed, usually on the extracted representations, at the second stage.

1.2 Image Representation

There have been several forms of knowledge representation which have already been used for representing visual information so that it can be stored in the computer memory for later processing and analysis. One of the elementary and most reasonable forms of image representation is the spatial domain, which is actually the representation provided by the image digitizing peripherals. In the spatial domain an image is represented by a two-dimensional array of elements, or pixels, [gonz78], [rose82]. Pixels of the spatial domain represent the local gray values at different locations in a given image. A typical spatial domain image representation is an array of 512 by 512 (or 1024 by 1024) elements defined over a 0-255 gray scale.

The spatial domain representation of images is usually the representation in which most practical vision systems receives their input images. Nevertheless, except for some mainly image processing tasks, this representation is not used as the main form for representing visual information in most computer vision and image understanding systems. It is clear that the spatial domain image representation is very demanding on the memory requirement, considering the fact that in practical images only very

small portions of the image contain some objects of any useful information. Also, the strict locality of information carried by the pixels has required most image understanding and pattern recognition techniques to seek other forms of knowledge representation, usually higher level representation, for storing the images to facilitate their high level analysis. The adequate form of representation usually depends on the task to be performed on the images.

Several hierarchical data structures have been used for image representation on the form of quad-trees, oct-trees, [burt80], [hunt79a], [hunt79b], [jone81], [klin76], [rose80], [rose83], [same80a], [same80b], [same80c], [same82], pyramids, [ichi81], [levi80], [tani76], and cones, [hans80], [uhrl72], [uhrl76], [uhrl78]. The basic idea is to provide several levels of resolution for the image. One of the advantages of this representation is that it provides finer resolution for different parts of the image, only if needed, i.e., for those parts that contain informative details. Therefore, this representation saves on the memory required to store the image and, consequently, it also saves on the computational cost. Another advantage of the hierarchical representation is that it provides several levels of resolution, thus it facilitates performing different tasks on image, in a more efficient manner, since some tasks are easier to perform at certain resolutions rather than others.

Another major approach of representation is to represent objects by vectors of features, e.g., color, size, etc., which can easily be measured from input images, [fuku72], [toui74]. This is the basic form of image representation used in the decision-theoretic

approach to pattern recognition. This approach is mainly concerned with the classification of certain patterns or objects.

A symbolic approach to image analysis and understanding, which has demonstrated usefulness in several applications, is the syntactic or the structural approach, [fuks82a], [eshe83], [fuks80], [fuks82c], [fuks83a], [fuks83c], [pavl77], [rose79a]. Basically, this approach uses symbolic representation for the visual structural information. There have been several forms of visual knowledge representation utilized within this approach itself, namely strings, [fuks82a], trees, or graphs, [eshe84a], [sanf83a], [slat80], as we will discuss in the next chapter. Briefly, in this approach, the structural features of the image, or of the objects in it, are represented as a set of entities, or primitives. These sets are called alphabets of primitives. The structural relation among these features are represented by mutual relations between the primitives of the alphabets through some attachment rules.

A recent approach has been emanating from the syntactic approach by combining the decision theoretic approach into it. In this case, semantic information is incorporated into the syntactic representation on the form of attributed structural representation, [fuks83a], [fuks82b], [pyst78a], [radi84], [tsai80a], [tsai80b], [youk79]. The image features are represented by attributed entities in the alphabet, where the attributes represent some semantic parameter of the structural features. Moreover, the semantic information of the relationships among the image features is represented by the attributes associted with the relations between their corresponding entities. This approach for image representation has shown to provide compact, concise, and powerful representation

that is capable of comprehending all the information contents of the images, as we will demonstrate through the course of this research.

In general, the attributed structural representation of images indicates the tied relation between vision systems and other branches of machine intelligence, such as data-base management systems, expert systems, etc., [chan82], [chan80], [meie83], [tamu84]. It also facilitates the exchange of techniques of analysis and conceptual basis of problem formulation between these different aspects of machine intelligence. For example, attributed relation graphs have demonstrated superior capabilities in handling the proper dimensionality of images and accommodating all their information contents. On the other hand, they deem close relation to some other general forms of knowledge representation in artificial intelligence, e.g., relational database, semantic networks, and frames.

1.3 Processing and Analysis of Visual Information

The actual processing and analysis of information in most image understanding and vision systems, as it is the case of most other knowledge processing systems, usually takes place in the second stage of the system on the extracted representation of images. The choice of a particular form of representation for the image data usually depends on the purpose of the system and the tasks expected to be performed by it. In general, some tasks can be performed on a particular form of representation much more efficiently than on other forms. Moreover, some form of representations do not emphasis the image global information

convenient availability for the analysis techniques, therefore they are usually impractical, if not impossible, to be utilized in most image understanding tasks.

The spatial domain image representation has been used as the major form of image representation in most image processing operations, e.g., filtering, local edge detection, image enhancement, etc. These operations do not need to comprehend the global information in the image, and they are also required to keep the image dimensionality, since their output is usually on the form of processed images, rather than decisions or certain actions. Therefore, the spatial domain representation seems proper for these type of image processing operations, [chen79], [duda72], [diam83], [fros82], [gonz77], [leej83], [naga81], [nevi82], [rose79b], [rose82], [tani77], [tsao81], [vaid82]. Although the spatial domain image representation is the main form of representation in which most machine vision systems receive the input images, but it is very rarely considered to be the adequate form of representation for most image understanding tasks, as we discussed in Section 1.2. Several powerful and global forms of image representation were discussed in that section. The choice of a suitable form of representation often depends on the tasks to be performed on the images.

The attributed structural (or syntactic/semantic) approach to image analysis and understanding has proven to be a powerful approach for several applications of machine vision systems. This approach was originally based on the utilization of the concepts of formal languages and automata theory for image analysis and understanding, [hopc69], [fuks82a], [thom76], [eshe83]. The basic

idea common to most techniques in the structural approach, is the decomposition of visual objects in the image into simple sub-objects. Through a recursive decomposition, complex structural objects can be broken down into sets of simple image features, usually called primitives, which constitute the alphabet of the representation.

In this approach images are represented by relational structures whose entities are defined over an alphabet of image features. Three main types of such relational structures have been used in this approach, namely string, tree, and graph, [ahuj81], [eshe84a], [pavl77], [rose79a]. The relationships among the image features are represented by the relations between their respective entities in the representation, e.g., left-to-right concatenation, father to children relation, and branch between nodes, for the string, tree and graph representations, respectively.

In many techniques which apply the attributed structural approach to image recognition and understanding, formal grammars are used to generate the structural representations of images of the same class, [fuks82a], [pavl77]. An image class is a set of images that share some common properties, usually some structural features. The grammars are used to provide a systematic and concise methodology for generating representations of images of a particular class, which is usually called the language generated by the grammar, [rose79a]. Several types of image grammars have been proposed and utilized for image analysis in this approach. String, tree, and graph grammars have been the main tree types of grammars used in this approach for the main three forms of representations.

An advantage of using formal grammars in this approach is to utilize them in designing systematic recognition algorithms, i.e., parsers. A parser is used to decide whether an unknown image is member of a certain class of images by syntacticly analyzing its structural representation with the language generated by a certain grammar. Parsers have been proposed for most of the grammars used for image analysis, [ahoa72]. However, most of these parsing algorithms require noise-free, or ideal, representation of images, which represent an obstacle in their utilization in real applications.

A major requirement of a practical image understanding system is to be able to handle real-word images which are usually burdened with noise, distortion, and uncertainty. In this case, exact parsing of the extracted object representation with the language representation of a class of objects does not provide an adequate solution to the image recognition problem. Therefore, there have been proposed some error-correcting parsing techniques, which are capable of correcting some errors. The only problem with such error-correcting parsing techniques lies in their high computational complexity, since they mainly test all the combinations that can possibly be used to correct the errors.

Another very useful approach to handle noise and distortion in real world image understanding problems utilizes similarity or distance measures between image representations. In practical problems, the more interesting question is how similar is an object to a prototype of a class of objects, rather than whether two objects are exactly identical. It is intuitive that a good and efficient distance measure between objects (or images) is a main milestone in any intelligent decision making process, a real-life everyday example

is that without a good distance measure between two merchandises an intelligent decision on which is the "smart" buy cannot be reached.

1.4 Research Topic and System Block Diagram

As we discussed briefly in Section 1.1, there are two main phases in most machine vision and image understanding systems. The first phase is concerned with the extraction of an adequate and efficient form of knowledge representation from the image data, while the actual analysis usually is performed in the second phase of the system. Several forms of image representation have been used in image analysis, in general, and in the structural approach in particular.

It is also known that in most practical applications, information sources, and images are no exception, are usually noisy and distorted. Needless to say, the more capable the first stage in any vision system of handling such noise and distortion in real-images, and of comprehending all the image information contents, the easier and more efficient is the analysis of the second stage will be.

For an image understanding system to be useful in practice, it should be able to handle at its input the real-world images themselves, or at most, the output of some simple preprocessing operations. It should comprehend the image information contents and preserve all its useful properties, such as symmetry, closure of

curves, connectivity, etc., throughout the different stages of processing. It should also be able to handle noise, distortion, and uncertainty which almost always exist in real-world images.

In this thesis, we focus our effort on the utilization of Attributed Relations Graphs as powerful tools for visual knowledge representations. For the first stage of the system we present a new hierarchical scheme for the extraction of a global image representation from images. The input to this stage are images as defined over a set of image primitives which are obtained by simple physical measurements on real images, it can be as simple as the gray scale values of the image pixels. The main component of this stage is a hierarchical multi-layer scheme, which receives at its input the result of some very elementary preprocessing operations, e.g., filtering, edge detection and thining. The scheme extracts from the image the informative global features, as defined by the structural alphabet of a hierarchical graph transducer. This transducer is basically a rule-based transformation implemented in bitwise operations. It performs a recursive mapping of the image information contents from the input image primitives into a global output alphabet. The extracted alphabet is then used to produce the image global representation in the form of an Attributed Relational Graph (ARG).

We also propose a new approach for calculating a graph distance measure and inexact matching between two ARG's to enable the system to pursue the image understanding process, and accomplish some tasks, such as locating some objects of interest in the image, etc. Informally, a similarity, or a distance, measure between two images (or subimages) is defined as the maximum number of similar

features that are common between the two images, or the minimum changes that need to be performed on one image in order to produce the other image. The important question in practical applications is not whether two images (subimages or objects) are identical, but rather how similar they are to each other. Thus the noise and distortion in real images can be accommodated by specifying tolerance in the distance between two images. Moreover, an interesting issue is to define and calculate a distance measure between an image and a part of a bigger image, i.e., how similar an object is to a sub-object.

A more detailed block diagram of the proposed image understanding system is shown in Figure 1.2.

1.5 Thesis Organization

The research performed in the course of this thesis has two major contributions to image understanding systems. Each of which lies within one of the two main components of the general system, which is shown in Figure 1.1. The first contribution deals with the extraction of a global symbolic representation in the form of attributed relational graph from real-world images through a new hierarchical scheme which uses a multi-layer graph transducer in mapping local image primitives into an alphabet of global features. This technique, along with some illustrative examples, is presented in Chapter 4.

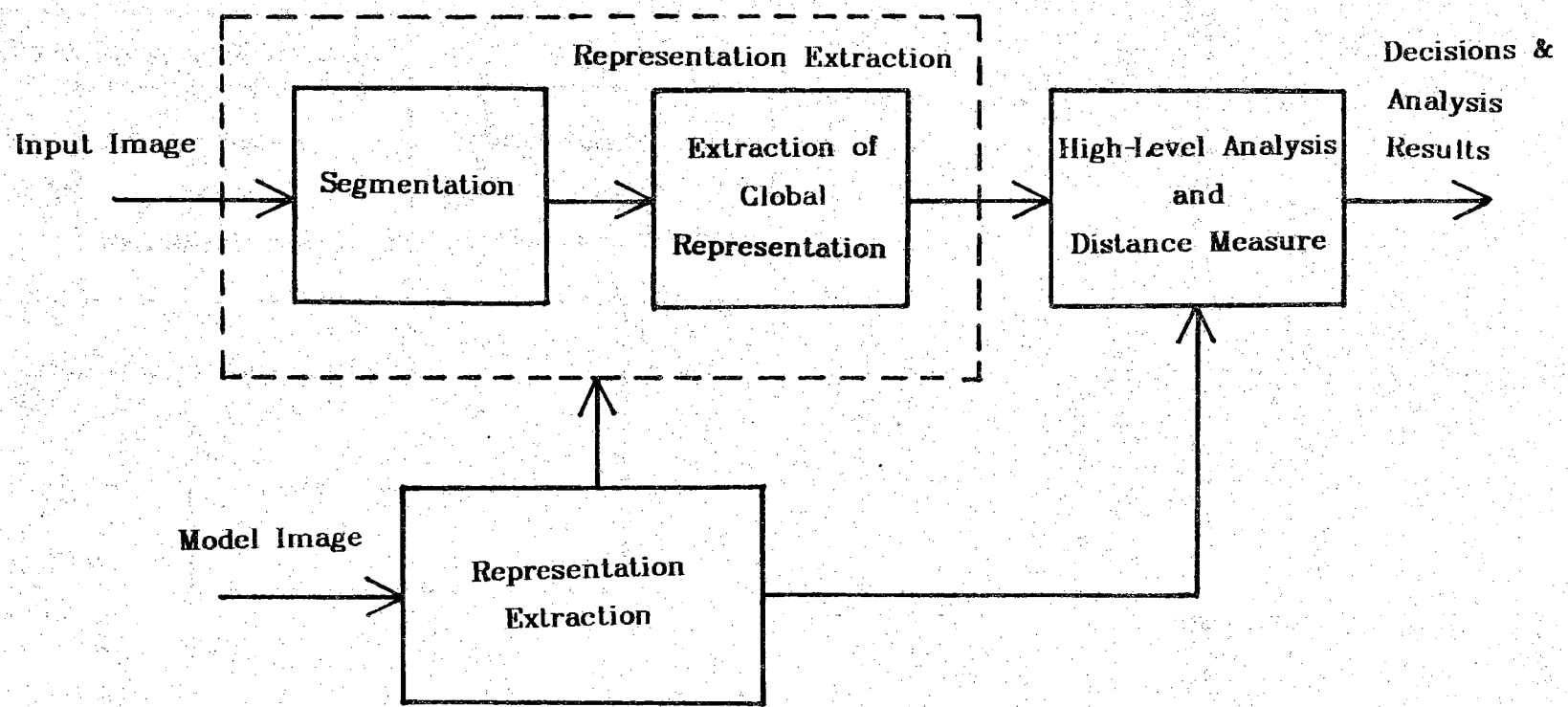


Fig. 1.2 Block Diagram of an Image Understanding System

To set up the background for the hierarchical scheme presented in Chapter 4, we review the related work on the structural techniques for image representation in Chapter 2. In that chapter, we discuss the conceptual bases of the structural approach, the main forms of image representation used in it, compare them, and show some of the advantages and disadvantages of each of them on the others. The powerful capabilities which are offered by ARG are also discussed to demonstrate the need and usefulness of such image representations for image understanding systems. We will also discuss how these approaches are related to other machine-intelligence systems rather than only vision systems.

In Section 1.3, we discussed the importance and the practical need for some distance or similarity measures between images, or between their respective representations. In Chapter 3, we investigate the different techniques for calculating distance measures between image structural representations as a background material to introduce the second major contribution of our research. This contribution deals with a new dynamic programming approach for calculating a global distance measure and finding the best inexact matching configuration, between attributed relational graphs; it is presented in detail in Chapter 5 of this thesis.

In Chapter 6, we present our experimental results and demonstrate the capabilities and the usefulness of the proposed techniques in performing image understanding tasks. The concluding remarks of our research, some problems which are still open for further investigation, and some new directions of machine intelligence, for which our approaches may also be useful, are all presented in Chapter 7.

CHAPTER II

STRUCTURAL AND HIERARCHICAL REPRESENTATION OF VISUAL INFORMATION

2.1 General

The syntactic approach to image analysis was initiated in the early 1960's, and was mainly based on concepts from the theory of formal languages and automata. In formal languages, [hopc69], sentences are described in terms of concatenated phrases which are formed from the concatenation of words. Similarly in image analysis, complex objects can be described in terms of simple sub-objects (or primitives) which are easier to analyze than the original images. Nevertheless, this similarity between formal language analysis and image understanding did not hold for very far. This is because of the difference in dimensionality and the difference in the nature of information contents between formal languages and images, and also due to the problem of noise and distortion that almost always has to be considered in practical applications to image analysis.

Therefore, generalization and extension of the concepts of formal languages and automata theory has been mandated along three major directions in the syntactic approach to image analysis. First, high dimensional forms of representation have been introduced to coup with the dimensionality of images, as we will discuss in Sections 2.2.3 and 2.2.4 as well as in Chapter 4. Second, a mean to incorporate semantic information into image representation has been provided through attributed representations, as will be investigated in Section 2.4 and in Chapter 4. Lastly, error correcting parsing and measures of similarity have been proposed to provide this approach with necessary tools, so it will be capable of handling real-world vision problems, which we discuss in Chapters 3 and 5. Nevertheless, before we discuss these techniques, we briefly review the most commonly used representation forms in the syntactic approach.

2.2 The Syntactic Approach to Image Representation

2.2.1 String Representation

Several of the early techniques in the syntactic approach utilized phrase-structural or string grammars as rule-based systems for imagery knowledge representation, [fuks82a], [kamb79], [eshe83]. An element, or a sentence, of the language generated by a phrase-structural grammar takes the form of one-dimensional string formed by concatenating symbols of the grammar alphabet, i.e., the set of image primitives. These symbols are entities that have two attachment points, namely the head and the tail. A string is an

ordered set of symbols where the left-most symbol is first and the right-most symbol is last. Symbols of the string represent structural features of the image, usually the features of the contour of objects in the image. A class of objects can be represented by a set of strings of some common syntactic properties. Therefore, usually a grammar can be designed to generate the string representation of an image class. Further image analysis tasks can be performed either on the strings themselves, e.g., through distance or similarity measures, or on their grammars, e.g., through building automata or parsers for classification and recognition purposes, [davi76], [davi78], [chia83].

Image representation by means of strings has been widely used in the syntactic approach, and has been very successful and adequate for some applications, [fuks82], [lusy82]. Nevertheless, strings are considered to be one-dimensional encoding of the images which are usually of higher dimensional nature. Therefore, other high dimensional types of representations have been demanded in order to preserve the image information contents and to overcome the elaborate preprocessing which is usually required for extracting the strings from real-world images.

2.2.2 Tree Representation

Another popular form of image representation in the syntactic approach to image analysis utilizes trees as hierarchical structures for image representation, [ball81], [faug83], [sanf83b], [lusy78b], [slat80]. In this case, classes of images are represented by sets of

trees. Usually, every image in the class is represented by a tree in that set.

Unlike the string case, where the only relation between primitives is the one-dimensional left-right concatenation, a tree is formed as a hierarchical structure of nodes. A designated node is considered to be the root of the tree. Every node in the tree is labeled by a ranked symbol that has two parts, the first represents the rank of the node and the second part is a labeling element from the primitive set. Similar to the string case, to perform the task of image analysis and understanding, further analysis is carried on the tree representation of images rather than on the images themselves, [chan79], [pyst78b].

The tree representation of images has been shown to be specially suitable for representing images of hierarchical nature. A wide class of images do not possess that hierarchical nature, therefore the third and the more general form of image representation in the syntactic approach utilizes graphs to represent complex images.

2.2.3 Graph Representation

A more general form of representation for visual information is on the form of graphs, where the basic entities are symbols with more than two attachment points and can be concatenated in an arbitrary way, according to their respective relations. In general, a graph, H, is defined as:

$$H = (N, B)$$

where N is a finite non-empty set of nodes, and B is a set of node pairs, or branches, [bond76], [chan79], [even79], [horo78b], [lova79], [nare81], [neva82], [pavl72]. Graphs are usually used for the representation of collection of objects and the relations among pairs of these objects. In the syntactic approach to machine vision and image analysis, nodes are usually used to represent the local information of sub-images, while the branches are representing the syntactic relationship between different parts of the image.

Several graph languages and grammars have been proposed in this approach as similar to the string case, but on higher dimensional bases. Plex languages are considered the straight forward multi-dimensional generalization of string languages. The primitive alphabet of plex languages consists of N-Attachment Points Entities, commonly called NAPEs, of arbitrary number of attachment points rather than tail and head as in the string case. Sentences of plex languages are generated by the concatenation of NAPEs at their attachment points.

Web grammars have, also, been proposed to provide description of images by means of undirected labeled graphs called webs. If a web grammar has a singleton alphabet, then the labels can be ignored and the web can be identified with its underling graph, in such cases it is usually called graph grammar. Some two-dimensional graph grammars have been introduced for the generation and recognition of ideal images, [bunk81], [eshe80], [nagl83], [schl76], [vigr78], [wong80].

2.3 Hierarchical Data-Structures for Image Representation

2.3.1 General

There have been several hierarchical data structure schemes used for image representation and analysis, [burt81], [hall76], [hans76], [hans78], [hend81], [hong80], [hong82a], [klin73], [klin79], [krus81], [moor81], [tani75], [tani78], [tani80]. Most of these techniques concern with the utilization of those data structures to perform some image processing tasks. The concept of planning was first introduced, [kell71], to speed the edge detection by applying an operator to a cross-resolution digitization and use the result to guide the further processing as where a fine-resolution digitization should be applied to minimize the wasted effort, [mark80]. In the remaining parts of this section we review some of those hierarchical data structures.

2.3.2 Quad-Trees and Oct-Trees

Quad-trees are region representations for images based on the successive subdivision of the image array into quadrants, [rose80], [same80b]. An image block is divided into four blocks. If the block is not homogeneous, then its corresponding node is given four children to represent the four quadrants. This process is repeated as many times as needed until only homogeneous blocks are represented by the leaf nodes of the quad-tree. Theses blocks could be as small as single pixels. Oct-trees follow the same principles as quad-trees except that image blocks are divided into eight sub-blocks instead of four.

The quad-tree image representation has shown to be particularly efficient for several simple image processing operations, such as scaling the images by power of 2, edge detection and enhancement, thresholding, image super-position, component counting, connected regions counting, computation of genus, moments, perimeters, linear transformation of images, etc., [dyer80], [hunt79a], [hunt79b], [jack80], [rana82], [rose83], [same82b], [limi82], [shne8a], [shne8b], [shne8c]. Nevertheless, the disadvantage of the quad-tree image representation, namely that it is not shift-invariant, makes this representation very inefficient and troublesome for many other important image operations such as shape analysis and image understanding in general.

2.3.3 Pyramid and Cone Structures

Pyramid data structures, [tani80], [luca84], [meri84], provide several levels of resolution for the image. A very simple pyramid structure for image representation is the pyramid in which the linear resolution doubles with each successive layer. In this kind of pyramids, the relationship between a level and the level directly beneath it is such that the value of a cell is the average of the values of its four immediate descendant cells. This type is called the M-pyramid and is usually defined as a sequence [M(L), M(L-1), . . . , M(0)] of arrays, where M(L) represents an original image, M(i-1) is a version of M(i) at half the resolution of M(i), and so on, with M(0) is usually a single cell, [ahuj84], [burt80], [dyer78], [dyer79], [ichi81], [levi80], [rose83], [sloa81], [ster81], [tani76].

More generally, pyramid image data representations can be constructed using any desired interval resolution-change factor, or weighted average over neighborhood configurations of any desired size. These neighborhood configurations are allowed to overlap forming what referred to as overlapped pyramids, as opposed to non-overlapped pyramids in which the neighborhood configurations do not overlap, [rose83].

An advantage of using overlapped pyramids, as compared with the non-overlapped pyramids, is that when local feature detectors are applied at various levels of a non-overlapped pyramid, their ability to detect large patterns in the image depends strongly on the positions of these patterns. For example, if we try to detect an object of certain size, it will make difference, when using non-overlapped pyramids, whether the object is located in the center of a subimage or is shared among several subimages. This position dependency could be greatly reduced if we use an overlapped pyramid structure, [dyer77a], [dyer77b].

Several image processing operations have been suggested utilizing pyramid data representations, [anto82], [dyer81a], [hall76], [hong82b], [hong82c], [hong82d], [meri84], [mill84], [piet81], [piet82a], [piet82b], [shne79], [shne80]. An image segmentation technique is presented in [rose79b], which apply a local feature detection operators at each level of the pyramid, or at certain levels only in case if we know the approximate size of the objects. The resulting information extracted by this operator is used to guide the segmentation process. A technique for region extraction and smoothing by block clustering using pyramid data representation is presented in [levi80] and [shne79]. In that work the idea used for

some known image smoothing algorithms is generalized from array data structures into pyramid data structures. For example, they have extended the neighborhood performed upon from being defined on blocks in the same array (in case of array data structures) in order to include neighboring blocks contained in different layers of the pyramid. The smoothing process was shown to be performed faster on the pyramid image data structures in comparison with the array data structures.

Another major hierarchical scheme for image analysis is called "Recognition and Processing Cone". It consists of layered structures of processors which are interconnected in regular pattern within each layer as well as between different layers. Uhr, [uhrl76], proposed the "Recognition Cones" in which the processing within each layer takes a serial form, while the processing over the different layers is conducted in parallel. Uhr's recognition cone consist of a structure of many layers; each layer consists of an array of processors. The size of the layer gets smaller as the layer gets higher in the cone, [uhrl78].

Hanson and Riseman, [hans80], suggested what is called "Processing Cones". A processing cone is basically a parallel array computer which is organized hierarchically into layers of decreasing spatial resolution, so that the information extracted from increasing sizes of image windows can be stored in the cone and further processed. The structure of these processing cones has been described in [hans80]; they bear a great relation to the recognition cones and Kelly's planning algorithms.

In [hans78] and [hans80], the processing cones are used to perform some pre-recognition operations. Some primitive operations for edge enhancement, and data projection using these cones are described in that work.

Image hierarchy is one of the basic concepts in the syntactic approach to computer vision and image understanding, since this approach is usually based on the recursive decomposition of the image into subimages, which are easier to analyze than the original image. Nevertheless, the utilization of hierarchical representation techniques have not received proper attention in this approach. In Chapter 4, we present a new hierarchical scheme for the extraction of global attributed representation from spatial-domain images. It is a rule-based system that utilizes a graph transducer, which we will present in Section 4.2, in performing multi-layer symbolic mapping of the image information contents, from local input alphabet into global output alphabet.

2.4 Syntactic/Semantic Representation of Visual Information

2.4.1 General

The representation of visual information is a very challenging task, since images are considered to be of the richest sources of information, due to the variety of their information contents, e.g., statistical and syntactic, and the generality of their dimensionality. Therefore, some new directions of expansion of the syntactic representation models have been proposed in order to achieve an adequate representation that is capable of efficiently carrying all the

information contents of the imagery data. One form of expansion is through including both the syntactic and semantic image information in the same representation.

In section 2.3, we briefly reviewed the three major types of representation that have been used to represent syntactic information in images, [fuks82b], [tsai80a], [taij80], namely, string, tree, and graph representations. Corresponding to those three types of syntactic (or structural) representation, there have been proposed three types of syntactic/semantic (or attributed structural) representation. The idea is to incorporate semantic information into the syntactic representation in order to form more general attributed structural forms of representation. In this case, attributes are assigned to elements of the primitive alphabets in order to give them more informative meaning. Attributed strings and attributed tree representations have been utilized in some image analysis applications, [brau84], [fuks82b], [shiq82], [bunk84]. However, in this thesis we utilize attributed relational graphs for image representation, since they are the more general and powerful form of representation. In Section 2.4.2 below, we discuss their conceptual bases and in Chapter 4 we discuss the extraction of attributed relational graph representation from the spatial-domain images.

2.4.2 Attributed Relational Graph (ARG) Representation

An Attributed Relational Graph (ARG) is a relational structure which consists of a set of nodes and the relations between these nodes on the form of branches. Both nodes and branches may have

some attributes assigned to them. Usually, nodes are used to represent some objects or parts of objects in the image, while their properties are assigned as attributes to their respective nodes. The relations between two objects are represented by attributed branches between the corresponding nodes. A formal general definition of Attributed Relational Graph (ARG) is given in Definition 2.1.

Definition 2.1 : Formally, an ARG is defined as:

$$G = (N, B, A, E, G_N, G_B)$$

where:

N: $(N=\{n_1, n_2, \ldots, n_{|N|}\})$ is a finite set of nodes, with $|N|$ is the number of nodes in N.

B: $(B=\{b_1, b_2, \ldots, b_{|B|}\})$ is a set of ordered node pairs (or directed branches), i.e., $b=(n_i, n_j)$ for some $1 \le i, j \le |N|$ denotes the branch emanating from node n_i to node n_j, and $|B|$ is the number of branches in B.

A: is an alphabet of node attributes.

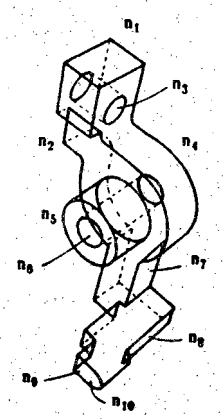
E: is an alphabet of branch attributes.

G_N: is a function (or a set of functions) for generating the node attributes.

G_B: is a function (or a set of functions) for generating the branch attributes.

Figure 2.1 illustrates the representation of a three-dimensional object (an industrial part) by an ARG. In this example , nodes represent three-dimensional primitives, namely Rectangular Blocks (R) and Cylindrical Blocks (C). The attributes of R are the length (l), the width (w), and the height (h) of the rectangular block, while

the attributes of C are the radius (r), and the height (h) of the cylindrical block. There are five relations between primitives in this example. The relation between two primitives, say a and b, is represented by a directed branch between their corresponding nodes n_i and n_j. The relations Inside (I), Above (V), to the Right (T), and Behind (H) with attributes d between two primitives, 'a' and 'b', denote that 'a' is inside, above, to the right of, or behind 'b', respectively, with distance d between their center lines. The Relation Joint (J) denotes that two primitives are joint.



(a) A Machine Part

Fig. 2.1 Example of ARG Representation of a Machine Part

Node Set: $N = \{n_1, n_2, \ldots, n_{10}\}$
Branch Set: $B = \{b_1, b_2, \ldots, b_{16}\}$

Node Attribute Alphabet:

ENTITY	Attributes
Rectangular Block: R	length : ℓ
	width : w
	height : h
Cylindrical Block: C	radius : r
	height : h

$A = \{(R:\ell, w, h), (C:r, h)\}$

G_N:
$n_1 \rightarrow R:\ell_1, w_1, h_1$
$n_2 \rightarrow R:\ell_2, w_2, h_2$
$n_3 \rightarrow C:r_3, h_3$
$n_4 \rightarrow C:r_4, h_4$
$n_5 \rightarrow C:r_5, h_5$
$n_6 \rightarrow C:r_6, h_6$
$n_7 \rightarrow R:\ell_7, w_7, h_7$
$n_8 \rightarrow R:\ell_8, w_8, h_8$
$n_9 \rightarrow R:\ell_9, w_9, h_9$
$n_{10} \rightarrow C:r_{10}, h_{10}$.

Branch Attribute Alphabet:

RELATION	Attributes
Inside: I	distance: d
Joint: J	none
Above: V	distance: d
To the Right of: T	distance: d
Behind: H	distance: d

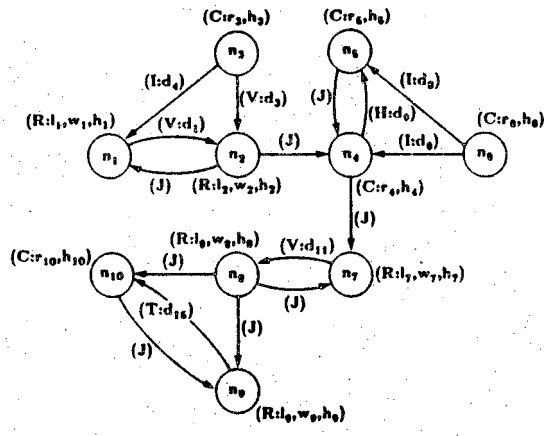
$E = \{(I:d), (J), (V:d), (T:d), (H:d)\}$

G_B:
$b_1 = (n_1, n_2) \rightarrow V:d_1$
$b_3 = (n_3, n_2) \rightarrow V:d_3$
$b_4 = (n_3, n_1) \rightarrow I:d_4$
$b_6 = (n_6, n_4) \rightarrow I:d_6$
$b_8 = (n_6, n_5) \rightarrow I:d_8$
$b_9 = (n_4, n_5) \rightarrow H:d_9$
$b_{11} = (n_7, n_8) \rightarrow V:d_{11}$
$b_{15} = (n_9, n_{10}) \rightarrow T:d_{15}$

$b_2 = (n_2, n_1)$, $b_5 = (n_2, n_4)$, $b_7 = (n_5, n_4)$,
$b_{10} = (n_4, n_7)$, $b_{12} = (n_8, n_7)$, $b_{13} = (n_8, n_{10})$,
$b_{14} = (n_9, n_8)$, $b_{16} = (n_{10}, n_9) \rightarrow J$

(b) Attributed Relational Graph for Object (a)

Fig. 2.1 (Continued)



(c) Graph Representation for ARG of Object (a)

Fig. 2.1 (Continued)

CHAPTER III

DISTANCE MEASURES AND INEXACT MATCHING BETWEEN IMAGE ATTRIBUTED STRUCTURAL REPRESENTATIONS

3.1 General

The need for some informative distance measures is extremely important in image analysis, as in other fields of functional analysis, decision theory and intelligent decision making processes. In most practical applications of computer vision and image understanding, images are usually burdened with noise, distortion, and uncertainity. Images of the same class could be ideal, noisy, or defected. Some of these noise and distortion are random in nature, but some others are controlled noise, and distortion. As we discussed before in Section 1.3, one of the important directions of generalization and expansion of the formal language concepts in the syntactic approach to image analysis is to incorporate the capability to handle noise and distortion in this approach.

To handle cases of random noise and distortion, some extensive preprocessing, e.g., filtering, will usually help in improving the performance of the image analysis techniques. In addition, stochastic languages and stochastic grammars have been proposed in the syntactic approach for modeling the randomness of image classes. However, the more interesting case, with equally wide applications, is the analysis of images with controlled noise and distortion. In this case, extensive preprocessing does not usually help very much, therefore, the need for some similarity or distance measures between images, or their respective representations, seems to be a must, [lusy82], [youk80].

A similarity or distance measure between two objects can be defined as the maximum number of similar features which are common between the two objects, or the minimum changes needed to be performed on one of the objects in order to make it identical to the other object.

In the remaining part of this chapter, we briefly review some related work on distance measures and inexact matching between different types of (usually attributed) structural representation of visual information, namely, strings, trees, and graphs. In Chapter 5, we propose a new dynamic programming approach for calculating a global distance measure and finding the best inexact matching configuration between arbitrary attributed relational graphs.

3.2 String-to-String Distance

Strings have been commonly used for image representation in the syntactic approach. A string is formed by concatenating elements of a set of primitives, where the left-most primitive is considered to be the first and the right-most primitive is the last. For two strings, say X and Y, over a primitive set Σ, we can define the following three transformations, [lusy78a] :

1. Substitution : $\omega_1 a \omega_2 \rightarrow \omega_1 b \omega_2$, $\forall a,b \in \Sigma$, $a \neq b$,
2. Deletion : $\omega_1 a \omega_2 \rightarrow \omega_1 \omega_2$, $\forall a \in \Sigma$,
3. Insertion : $\omega_1 \omega_2 \rightarrow \omega_1 a \omega_2$, $\forall a \in \Sigma$,

where ω_1 and ω_2 are some substrings over Σ. Levenstein, [leve66], defined a metric distance between two strings X and Y, d(X,Y), as the smallest number of transformations required to derive the string Y from the string X.

If some non-negative weights (or costs) are assigned to these three transformations, e.g., w_s, w_d, and w_i as the costs of substitution, deletion, and insertion, respectively, then the distance between two strings X and Y is the cost of the transformation sequence of minimum total cost and which are needed to transform X into Y. Let $Q = \{ q_j \mid j=1, 2, 3, \ldots, J \}$ be the set of all possible sequences of transformations that can be used to derive Y from X. A sequence q_j, $q_j \in Q$, has S_j substitutions, D_j deletions, I_j insertions. The weighted distance between the two strings X and Y is :

$$d^w(X, Y) = \min_{j \in J} \left(w_s . S_j + w_d . D_j + w_i . I_j \right)$$

In order to make the string-to-string distance measure more informative and useful for image analysis applications, Lu and Fu, in [lusy78a], suggested a modified weighted string-to-string distance, in which the costs associated with the substitution, deletion, and insertion transformations are functions of the primitives on which they are applied. In this case, a sequence q_j in the set of all possible sequences of transformations that can be used to derive a string Y from a string X, $q_j \in Q$, has total cost of W_j which is the sum of the costs associated with the transformations in q_j. Thus, the modified weighted distance between X and Y is defined as :

$$D^w (X, Y) = \min_j \{ W_j \}$$

The modified weighted string-to-string distance has been formulated as a shortest path problem between two specific nodes (namely the two opposite corner nodes) in a rectangular acyclic directed lattice, [fuks82a]. Each path in the lattice corresponds to a sequence of transformations that can be used to derive string Y from string X. The different types of error-transformations have been represented by branches in the lattice. The costs of these transformations have been assigned to the corresponding branches. Thus, the optimization aspect of the modified weighted string distance problem is converted into a shortest path problem over an acyclic directed rectangular lattice.

The general shortest path problem over acyclic directed graph is a popular prototype problem in dynamic programming. Wagner and Fischer, [wagn74], proposed an algorithm for the solution of this problem, which was utilized in [lusy78a], for calculating the

modified weighted string-to-string distance. The time complexity of this algorithm is of order ($|X| \cdot |Y|$), where $|X|$ and $|Y|$ are the number of elements in the X and Y strings, respectively.

3.3 Tree-to-Tree Distance

Tree-structures as defined over sets of primitives, have been commonly used for image representation in the syntactic approach. Unlike the string case, where the only relation between primitives is the left-to-right concatenation, a tree is formed as a hierarchical structure of nodes. Every node in the tree is labeled by a ranked symbol that has two parts, the first represents the rank of the node (i.e., the number of the children descending from this node) and the second part is a labeling element from the primitive set, [fuks82a].

For any two trees, say X and Y, over a primitive alphabet Σ, we define some weighted transformations, similar to the string case, which enable us to derive one tree from the other through the repeated application of these transformations. The weighted distance between X and Y, d(X,Y), is the minimum cost sequence of transformations that are needed to derive X from Y. Let x and y be two arbitrary subtrees of X and Y, respectively; then the following three transformations can be defined:

1. Substitution : $\sigma(y) \rightarrow x$, $w_s(x,y)$,
2. Deletion : $\varepsilon(y) \rightarrow \lambda$, $w_d(y)$,
3. Insertion : $\varphi(\lambda) \rightarrow x$, $w_i(x)$,

where λ is a null subtree, and $w_s(x,y)$ is the cost of substituting subtree y by subtree x, $w_d(y)$ is the cost of deleting subtree y, and $w_i(x)$ is the cost of inserting subtree x.

If $Q=\{q_j \mid j = 1, 2, \ldots, J\}$ defines the set of all possible sequences of transformations that can be applied on subtrees from Y in order to derive X, and q_j is an arbitrary sequence in this set, i.e., $q_j \in Q$, then the weighted distance between the strings X and Y is:

$$d(X, Y) = \min_{j \in J} \{ W_j \}$$

where W_j is the total cost of the transformations used in the sequence q_j. The very important issue to be noted here, is that the set Q for the tree case is expected to be very large in comparison with the equivalent set for the string case. This is due to the difference in the way trees and strings are constructed.

Lu, [lusy79], has restricted the range of transformations used to derive the tree X from tree Y, in an attempt to reduce the set of all possible sequences of transformations for deriving X from Y. She imposed three conditions on these transformations, namely :

1. the predecessor-descendant relations of nodes in Y do not change,
2. nodes in Y do not split or merge,
3. the sequence of postfix ordering does not change after applying the sequence of transformations.

In other words, the transposition or rotation between subtrees are prohibited; the transposition of two subtrees is achieved by a number of deletions, insertions, and substitutions. Based on these conditions, Lu proposed an algorithm to compute the tree-to-tree distance using dynamic programming for computing the elements of a distance matrix D(i,j), where i and j are indices of postfix ordering of trees Y and X, respectively. If node x, of postfix ordering i, and node y, of postfix ordering j, are nodes in the trees X and Y, respectively, then D(i,j) is the minimum cost necessary to derive subtree X/x from subtree Y/y. Therefore, the distance between the trees X and Y is:

$$d(X,Y)=D(m,n)$$

where m and n are the number of nodes in trees X and Y, respectively. Lu also showed that, under the above mentioned conditions, the algorithm has time as well as space complexity of O(m.n).

3.4 Similarity Measures and Inexact Matching Between Graphs

3.4.1 General

Graphs have shown to be very useful and powerful tools for image representation. Image representation by means of graphs constitutes the core of a major class of high-dimensional image analysis techniques. Graphs are very adequate to properly handle the high dimensional and/or hierarchical nature of the visual information; they are also topologically invariant. Moreover, we can

have labels or attributes assigned to their nodes and branches on the form of attributed relational graphs, as we discussed in Section 2.4.2.

For the past few decades, the general theoretical aspects of graphs have been studied extensively, [carr79], [bond76]. Unfortunately, several of the problems concerned with graph analysis have been shown to have very high order of computational complexity and are classified as NP problems, [papa82], [gare79], [even79], [read77], [ullm76]. Nevertheless, many of these problems do have some good heuristic solutions with reasonable computational complexity, [horo78a].

Due to the interesting capabilities and descriptive power exhibited by graphs as tools for image representation, operations on graph have been studied by many researchers in the structural approach to image analysis, [chen81a], [chen81b], [chen84], [fuks82], [ghah80a], [ghah80b], [tana77]. Several interesting graph grammars, [bunk81], [vigr78], graph language transformations, and some parsing techniques have been studied, [aul83], [shiq83a], [sanf81]. The problem of finding an informative global graph-to-graph distance measure is very important in most applications utilizing graphs in practical machine vision and image understanding systems, or other knowledge processing systems, so that the system can handle real-world images, or other sources of information, in general, [khan84].

In Chapter 5, we present a new approach for calculating the distance and finding the best inexact matching configuration between two attributed relational graphs, in their general form.

However, as a background material to that approach, in the remaining part of this section, we review the work published in the literature on some related problems.

3.4.2 Review of Related Work on Graph Distance and Inexact Matching

Sanfeliu, [sanf83a], presented a distance measure between two attributed relational graphs. He assumed that the graphs have bijection labeling functions for both the nodes and the branches, i.e., the nodes and branches have unique labels. He also requires three conditions to be fulfilled in order for his approach to be valid, namely:

* Both graphs must be generated by the same "Descriptive Graph Grammar".
* At least one node of one of the graphs (called it the input Graph) must be pre-recognized as a certain node in the other graph (reference graph), otherwise, his approach is invalid.
* At least one label matching configuration between nodes of the two graphs should be feasible from the attribute matching point of view.

If the above three conditions are satisfied, then the technique in [sanf80] considers the nodes of both graphs to form matched pairs, according to their labels. Each pair contains one node from each graph. By forming configurations of matched-pair of nodes over all nodes of both graphs, the distance between two graphs is defined as:

$$\min \{ w_{nr} c_{nr} + w_{ni} c_{ni} + w_{nd} c_{nd} + w_{bi} c_{bi} + w_{bd} c_{bd} \}$$

where the minimization is to be performed over all possible configurations of matched-pairs of nodes; w_{nr}, w_{ni}, w_{nd}, w_{bi}, and w_{bd} are weights of node recognition, node insertion, node deletion, branch insertion, and branch deletion respectively, they sum together to 1; c_{nr}, c_{ni}, c_{nd}, c_{bi} and c_{bd} are costs of node recognition, node insertion, node deletion, branch insertion and branch deletion, respectively.

The shortcoming of Sanfeliu's technique can be summarized in the following:

* The method is only concerned with the distance between two graphs that are generated by the same "Descriptive Graph Grammar", which make it incapable of handling general case of Attributed Relational Graph representation of images.
* The method totally fails, if it is not provided with at least one pre-recognized matched-pair of nodes.
* The attribute information of the graphs are only used in the last step of the method to check whether the two graphs, in a particular matching configuration, fulfill the attributes of one another. If not, the configuration is rejected. Thus the method does not incorporate the attribute information in the actual calculation (or definition) of the distance measure.
* The method considers the insertion and deletion of nodes and branches but does not consider the substitution error-transformations.
* The most serious drawback of Sanfeliu's work is that he handle the optimization part of the problem, which is the major issue in the distance measure, by enumerating all the

possible matching configurations and performing exhaustive search. It is well known that this problem is of very high exponential complexity.

In [sanf83a] and [sanf81], Sanfeliu applied his method on two examples, namely character recognition, and the analysis of muscle tissue patterns. In both cases the size of the graphs were very limited. The preprocessing was done manually and had to be modified in order to obtain graphs of fewer nodes, since the method faced the expected exponential time complexity problem.

An approach for error-correcting parsing of attributed and stochastic tree languages has been proposed in [shiq82]. That approach is based on converting the tree into a pair of related strings, namely the node string and the rank string. By using these two strings, it becomes possible to construct either a minimum distance error-correcting parser (MDECP) or a maximum likelihood error-correcting parser (MLECP) since the problem becomes similar to the string parsing problem. The MDECP is for the deterministic languages while the MLECP is designed for the stochastic case.

In [shiq83], an approach for parsing graph languages via the extraction of their basis graphs has been proposed. That approach focuses on the class of labeled acyclic digraphs where the basis graph is a tree. The approach for error-correcting tree parsing, [shiq82], was extended to cover the parsing of graph languages for some graphs of this class. Four related strings are extracted from the graph, namely a node label string, a rank string, a branch label string, and an index string. The last two strings are used to specify the branch relations among the nodes. Thus the graph parsing task,

for this class of graphs, was converted into a string parsing problem. This approach to the parsing of trees and graphs is a good simplification of the problem, only for that class of graphs.

In [hara79], Haralick and Shapiro investigated the "Consistent Labeling Problem" as a general case for several related problems, e.g., graph homomorphism, subgraph isomorphism, graph coloring, and scene and edge labeling problems. This problem involves a set of objects to be given names, or labels; a set of possible labels for the objects; and a world model of compatibility containing ordered groups of object-label pairs which are compatible. The objective is to find a label for each object such that the resulting set of object-label pairs is consistent with the constraints of the world model. Their approach to this problem is basically a depth-first backtracking search with a look-ahead operator to make the search more efficient.

A poor choice of object-label pair early in the search tree can cause failure of all paths stemming from that choice. To help speed up the search, those paths must be eliminated as early as possible, since they are not contained in any consistent labeling configuration. The look-ahead operator is used in attempting to reduce the world model of compatibility by eliminating the poor choices of object-label pairs. Consequently, that should reduce the size of the backtracking search tree.

As continuation of the above work, Haralick and Shapiro, [hara80a], defined a minimal compatibility model as the compatibility model that contains only the ordered groups of object-label pairs which contribute to a consistent labeling, i.e., the

removal of any one group of object-label pairs will eliminate at least one consistent labeling with respect to that model. Only if we have minimal compatibility model, then the tree search procedure for finding the consistent labeling is very efficient.

The problem of reducing a world model of compatibility into the minimal form, is itself shown by Montanari, [mont74], to be NP-complete. Nevertheless, Haralick and Shapiro, [hara80a], presented supportive argument that although the worst case complexity is exponential but it occurs very rarely and in practical cases the complexity is far from being that bad. They also argued that even that sometimes we can only achieve near minimal (but not minimal) in reducing the compatibility model, but the closer the model is to the minimal, the less work the tree search demands. The reduction of world compatibility model into minimal form is achieved via look-ahead operator. Thus the complexity of consistent labeling problems depends on how much look-ahead work is required to reduce the compatibility models into minimal form.

In [shap81], Shapiro and Haralick investigated the structural description of an object and the concepts of exact and inexact matching of two structural descriptions. They defined a weighted structural description of an object to consist of the weighted descriptions of its primitive parts and the weighted interrelationships among them. They considered the one-way matching as whether a candidate structural description matches a prototype weighted structural description.

An exact matching gives the correspondence from the primitives of the candidate structural description to the primitives of the prototype description. The correspondence from the relations of the

first description to the relations of the second description forms a relational homomorphism. The problem of inexact matching is basically to find the best match for a candidate description over a set of prototype weighted structural descriptions. The best match is usually defined as the match with minimum inexactness. The inexactness is defined by assuming some thresholds for the matching of different primitives; a threshold for the unmatched primitives from the prototype description, and other threshold associated with the homomorphism between the relations of the two descriptions.

The matching of structural descriptions, [shap81], is formulated as a consistent labeling problem, therefore it can be solved by backtracking tree search. The idea of look-ahead operator can be incorporated in order to reduce the search tree. Shapiro and Haralick, [shap81], investigated such tree search with backtracking alone, and with backtracking and look-ahead. They presented their results which showed that looking-ahead searches less number of nodes of the tree than the other method.

Another very interesting work is due to Tsai and Fu, [tsai79b]. They extended the conventional graph isomorphism to include error-correcting capability and apply it on attributed relational graphs for image analysis. The attributed relational graph is considered to consist of an underlying unlabeled graph to represent the global structure of the image. The labels for the nodes and the branches are assumed to specify the local information of the primitives and the relations between them. The labels are considered to have two separate parts, namely a syntactic symbol and a semantic vector.

The error-transformations are defined in local form and called structure-preserved graph deformations, in other words some nodes and branches may be locally corrupted without changing the global topology of the graph. Error probabilities were assigned to both syntactic and semantic primitive and branch deformations. For each entity, i.e., a node or a branch, in the graph they defined a set of entities which represent all the possible deformations of it. The error-correcting isomorphism between two graphs is a one-to-one mapping which maps all the nodes and branches of one of the graphs into the other, based on the sets of deformations of each node or branch.

In this case, both graphs must have the same number of nodes and the same number of branches, thus the error-transformations are on the form of node or branch relabeling but not deletion or insertion. The goodness of the isomorphism could be measured by maximum likelihood, by minimum distance, or by least-square error measure. The problem of finding an error-correcting isomorphism between two attributed relational graphs was handled, in [tsai79b], by an ordered search algorithm which expands a search tree using a heuristic function to assist in reducing the size of the tree.

In [tsai83], Tsai and Fu further extented the concepts of error-correcting of graph isomorphisms, as in [tsai79], for finding subgraph error-correcting isomorphisms. The problem set up for the subgraph case is exactly similar to the former case except for relaxing the constraint requiring that both graphs to have the same number of nodes and the same number of branches. Therefore, the error-transformations on the form of node or branch deletion are

considered, along with relabeling, but not the insertion error of nodes or branches. It is also assumed that the deformations are independent of one another.

Similar to the graph isomorphism case, the problem here is to find a one-to-one mapping which maps all nodes and branches of one of two graphs into a subgraph of the other graph. The remaining nodes and branches of the later graph is mapped to nulls, i.e., deleted. The goodness of the mapping between two graphs can be measured by the likelihood measure of each matched pair of nodes and branches in this mapping. The actual problem is to find the isomorphism of maximum likelihood between two attributed relational graphs. This problem is formulated as an ordered search problem over a state-space tree.

An efficient algorithm for the above problem was proposed, in [tsai83], using some heuristic function and validity conditions in order to contract the search tree and speed up the search. The heuristic function is basically a local function without any look-ahead capability. At each state the function is evaluated on the sets of nodes which are directly connected to the the mapped nodes at this state. Nodes in the search tree is expanded according to the heuristic function. The generated tree contains one path from the root node to a goal node. This path represents the sum of local optimums but not necessarily the required global optimum.

CHAPTER IV

EXTRACTION OF ATTRIBUTED STRUCTURAL REPRESENTATION OF IMAGES BY A HIERARCHICAL GRAPH TRANSDUCER

4.1 General

As we discussed before in Section 1.1, the first phase of an image understanding system concerns, mainly, with the extraction of adequate representation of the images, or the different objects in the images, so that images can be stored in the computer memory in a compact, yet complete, form to facilitate their further analysis. The actual analysis and interpretation of the image are performed in the second phase of the system, usually on the extracted representation. In almost all practical applications, images, like most other information sources, are usually burdened with noise, distortion, and uncertainity. It is clear that the more capable is the first phase of handling the noise and distortion in real data, and of comprehending all the input information contents, the better it is, and the easier and more efficient the later analysis would be.

The most powerful and useful general form of representation utilizes graphs, or more specifically attributed relational graphs, for representing the images. An attributed relational graph is a relational structure consisting of a set of nodes and the relations between these nodes on the form of branches. Both nodes and branches may have attributes associated with them. Usually, nodes are used to represent some objects or parts of objects (or features) in the image, while their characterizing measurements are assigned as attributes to those nodes. Relations between two features are represented by attributed branches between the corresponding nodes.

In this Chapter, we present a new approach for the extraction of attributed relational graph representation from images. The input to this system are images as defined over a set of image primitives which are obtained by simple physical measurements on real-world images. The image primitives can be as simple as the gray scale values of single pixels in the image. The main component of the system is a hierarchical multi-layer scheme for the extraction of a global alphabet from the image. The input to this scheme is the result of some very elementary preprocessing operations, e.g., filtering, edge detection, and edge thinning. A simple block diagram is shown in Figure 4.1. The proposed scheme utilizes a graph transducer to perform a symbolic mapping of the image primitives into the global alphabet. The mapping is performed from one layer to the other in a hierarchical fashion. The scheme extracts the image representation according to some alphabets of global features. The extracted features are used to produce the attributed relational graph representation for the image under consideration.

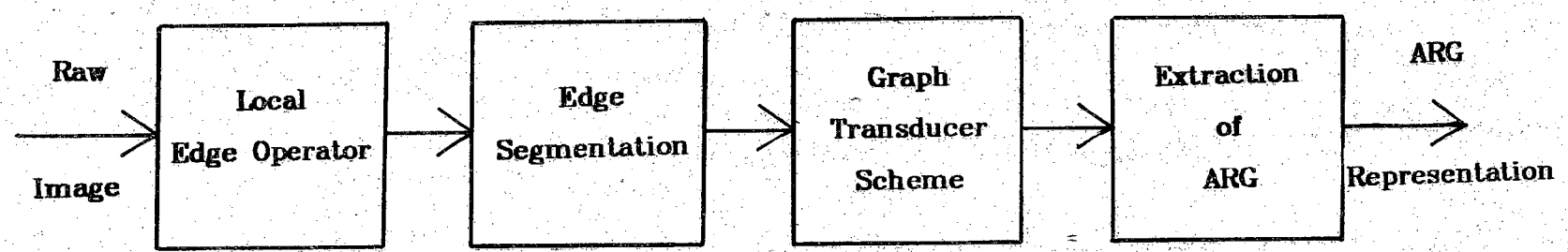


Figure 4.1 Extraction of Image Attributed Structural Representation

In Section 4.2, we present the concepts of the graph transducers which are used in our scheme. The basic concepts of using hierarchical schema in image representation were discussed before in Section 2.3. In Section 4.3, we explore the utilization of these concepts for the extraction of alphabets of global features from spatial-domain images. In Section 4.4, we present the hierarchical graph transducer which is used in the multi-layer scheme for extracting global features. While in Section 4.5, we discuss the extraction of the attributed relational graph representation from the scheme. The preprocessing techniques are not discussed in this thesis, since we only use some standard edge detection and thinning techniques.

4.2 Extraction of Symbolic Representation for Images

4.2.1 General

Graphs are a higher dimensional generalization of other forms of image representation, e.g., strings, trees, etc. The Graph representation of images provides several powerful capabilities which are very useful for image analysis and understanding. Therefore, operations on graphs and their utilization in pattern representation and analysis deserve a great deal of attention, especially in the structural approach to computer vision and image understanding. Several interesting graph grammars [bunk81], [schl76], graph language transformation, and some parsing techniques have been proposed [shiq82]. A very important issue in image analysis by graph concerns with the extraction of the graph representation from real-world images.

In this section, we present a graph transducer for mapping the image information contents from low-level representation into high-level representation. Formal definition of this transducer is presented in Section 4.2.2 along with the algorithm which utilizes this transducer to perform the mapping. We also show some examples for its utilization in performing one-level feature extraction from images.

4.2.2 Graph Transducer for the Extraction of Image Symbolic Representation

The proposed transducer, like almost any other transformation, has a field, domain and range alphabets, and a mapping function. The alphabet over which the input image is defined represents the domain of the transformation. Elements of this input alphabet is defined over some local areas in the image. A more global alphabet is the output alphabet, whose elements are defined over larger areas in the image. The output alphabet represents the range of the transformation. Elements of the output alphabet are composed of some neighboring elements from the input alphabet according to a certain neighborhood configuration, as will be defined by the transducer, e.g., four-nearest neighbors, eight-nearest neighbors, etc. The mapping algorithm of this transducer is basically a rule-based technique which maps the field cells from being defined over the input alphabet (i.e., the set of image primitives) into the output alphabet (i.e., the set of relatively more global features,) according to some neighborhood and compatibility rules, as we will illustrate in this section.

An image can be considered as a field of cells on which we define a function that takes on values in a set of local image primitives. Figure 4.2 shows an example of an image field of cells arranged in an eight-neighbor configuration. In low level image representation, elements of such an alphabet are in the form of some physical measurements taken over local areas in the image, e.g., the color of image pixels, the gray scale value of a single pixel in a gray scale image, or 0 and 1 for binary images. On the other hand, in high level representation, elements of the alphabet are defined over larger subimages and they represent features which are much more global in nature, in comparison with the primitives represented by lower level alphabets. Consequently, the features represented by alphabets of a high level representation are more meaningful and informative than those image primitives represented by alphabets of a lower level representation.

The output of this graph transducer consists of a symbolic representation of the image as defined in terms of an output alphabet, which is relatively more global than the input alphabet. The transformation from the input to the output alphabets is defined by a mapping function, Ω, and will be illustrated by Algorithm 4.I, which is shown below. The formal definition of the transformation as well as the algorithm which performs the mapping are presented later on in this section, but first we discuss the motivation and the utilization of such a transducer in a scheme for the extraction of global symbolic representation from spatial-domain images.

The proposed graph transducer is used to map the input image data as defined over a specified neighborhood configurations of cells in the image field and places the resultant into certain central cells

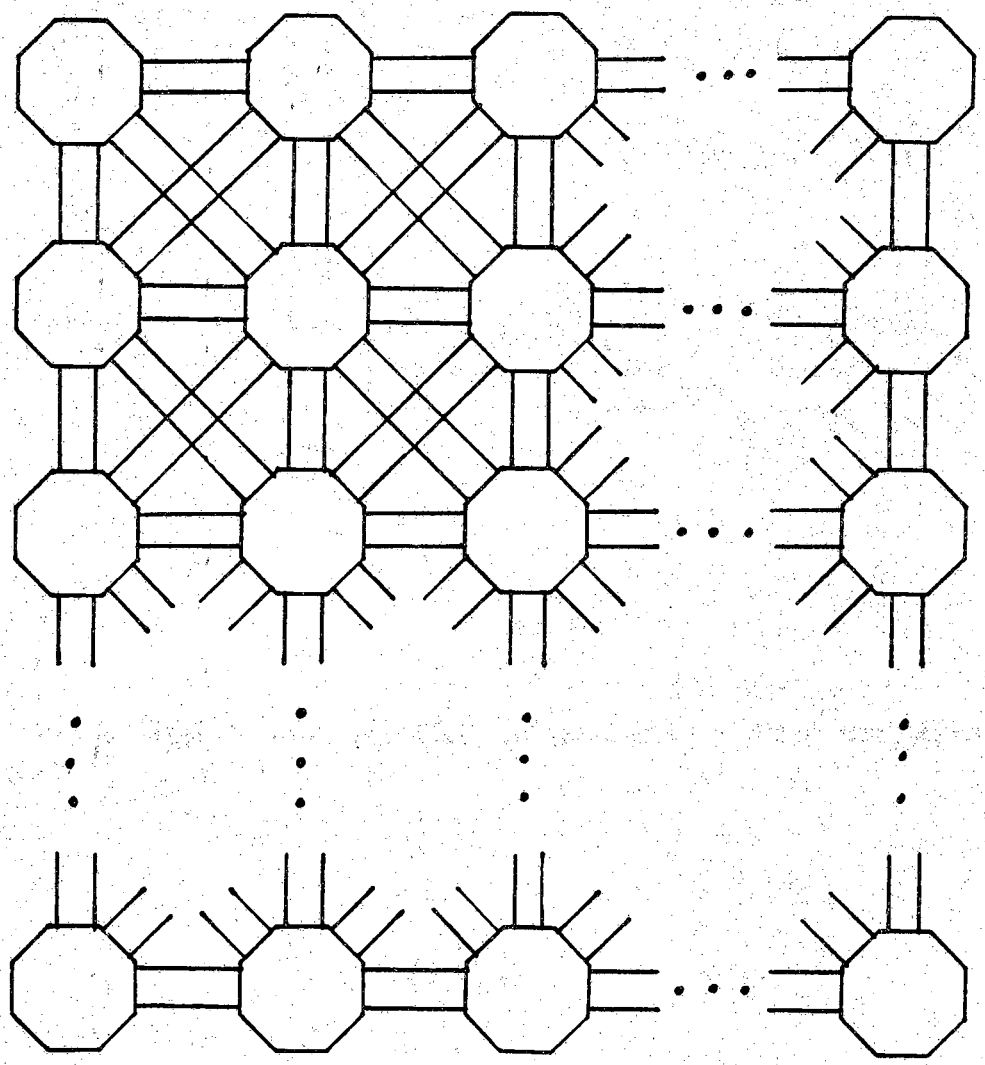


Fig. 4.2 A Graph Transducer Field in an Eight-Neighbor Configuration

of the respective configurations. The number of cells in the neighborhood as well as their topological configuration are both defined by the transducer. This graph transducer can be utilized in a single layer system or in a hierarchical multi-layer scheme.

In a single layer system, the transducer is used to extract some image features, each of which is composed of a set of image primitives as defined by the neighborhood configuration of the graph transducer; an example will be shown below.

In a hierarchical multi-layer scheme, the graph transducer performs the mapping of the image from the set of image primitives into a set of global features. At an arbitrary layer in the scheme, the input alphabet of the graph transducer at this layer is taken as the output alphabet of the layer beneath it, while its output alphabet is fed as input to the transducer at the layer above it. In Section 4.3, we discuss the proposed multi-layer hierarchical scheme in more details.

Formally, we define the graph transducer T of the form :

$$T = \{ \Sigma_I, \Sigma_O, F, R, \Omega, \Phi \}$$

where :

Σ_I is an input alphabet,

Σ_O is an output alphabet,

F is a field of cells, representing the image field,

R is a symmetric predicate defining the adjacency between every two cells in F; let S denotes the set of cell pairs as defined by R,

Ω is a function that defines pairs of alphabet symbols for every pair of adjacent cells in S,

Φ $(\Phi(\sigma_i) \subseteq \Sigma_O)$ is a function that defines a subset of output symbols for every input symbol.

The input alphabet, Σ_I, is chosen as the set of all image primitives in which the digitized input image is presented to the system. It is usually considered to represent some quantized physical measurements taken over local areas in the image, e.g., the color or the gray scale values of image pixels. In some sense, Σ_I is decided very much based on the given input image. On the other hand, the output alphabet, Σ_O, is chosen to represent more global features in the image. Each symbol in this alphabet is composed of a group of image primitives from Σ_I, such that each element in that group occupies a cell in a neighborhood configuration as defined by the predict R of the transducer.

The selection of elements of Σ_O does usually depend on the class of objects sought in the image. If the transducer is designed for the extraction of the features of a certain class of objects, then elements of Σ_O are selected as the informative features of that class, i.e., the discriminative features possessed by that class of objects. On the other hand, if no particular class of objects is under consideration by the system, then elements of Σ_O are selected to represent all features that can be formed by groups of image primitives from Σ_I as defined over the neighborhood configuration of cells specified by the predicate R. The more restricted the set Σ_O is, the faster the mapping would be. In other words, if general pre-knowledge about the class of objects sought by the system is available, then we restrict the alphabet Σ_O to contain only the

informative and discriminative features of this particular object class. In this case, the mapping process will be much faster. If no pre-knowledge about the object class is available, which is usually the case in the inference or learning stage, then the proposed scheme will need to consider a large alphabet of symbols, and then the system will be slower. Nevertheless, this is only fair, since even for human, if no previous knowledge is available, the learning process will be slow.

The field F is a field of cells where the transducer operates and the mapping takes place. It is usually, but not necessarily, taken as M×N array of cells. The interconnections between cells of F are defined by the adjacency predicate R, as will be explained below. Two cells in the field are directly connected if and only if they are defined by the predicate R to be adjacent, i.e., neighbors.

The predicate R is a symmetric predicate which defines the neighborhood configuration of cells in the transducer field F. In other words, it specifies whether two cells in the field are adjacent or direct neighbors of each other. In general, the neighborhood relations among cells of the transducer field can be defined in any arbitrary configuration required, or the best suitable for characterizing a certain class of images. However, in our work here, and also in most practical cases, we restrict the neighborhood relations between field cells into simple, e.g., four-neighbor or eight-neighbor, configuration. We will discuss this issue further in the following two paragraphs.

The main advantage of using a general neighborhood configuration is to enable the transducer to perform the mapping to extract arbitrary complex features, or more global image representation, in a single layer fashion. However, the disadvantages of using such a general neighborhood configuration are mainly due to the increasing complexity of the mapping rules, which are needed to describe the complex features, and also due to the problem arising from interconnecting the field cells in an arbitrary configuration. The interconnection problem is basically a hardware implementation problem. This problem does not actually arise in our software simulation of the system, nevertheless, we take it into consideration so that the proposed transducer will be adequate for hardware implementation.

Therefor, the main reason for using a simple regular neighborhood configuration, as shown in our figures, is to reduce the interconnection networking problem in the single-layer transducer. Yet, the powerful capabilities of our system to extract complex features, or global representation, from images is still achieved through the proposed multi-layer scheme, which will be presented in Section 4.3 and 4.4. In that scheme, global features are recursively decomposed into simpler features. The transducer, then, performs an iterative mapping at different layers, utilizing a simple neighborhood configuration between the field cells and a simple mapping function at each layer, as we will show in an example later on in this section.

The function Ω defines pairs of alphabet symbols for every pair of adjacent cells in the field of the transducer. It is mainly a rule-based function to define the consistency of the features to lie in a

certain neighboring cells in the field in order to form a certain symbol in the output alphabet Σ_O. The function Φ defines a subset of possible output symbols, i.e., subsets of Σ_O, for each input symbol in Σ_I, which represents the possible mappings of that image primitive into those output features. Both functions, Ω and Φ, define a set of rules for Algorithm 4.I to perform the symbolic mapping of the input image into the output representation. The input image is defined over the set of local image primitives represented by the input alphabet, Σ_I, while the output representation is represented by the cells of the transducer field. Cells of the field, after the mapping is performed, are defined over the set of more global features represented by the the output alphabet Σ_O.

Algorithm 4.I operates on the field of the transducer to perform the mapping of an input image representation. It maps the image representation from being defined in terms of symbols of the input alphabet, Σ_I, to be defined in terms of entities from the output alphabet, Σ_O. As we discussed before, elements of Σ_I are basically image primitives defined over local subimages, while elements of Σ_O represent the image features, which are defined over more global subimages. Consequently, the image representation in terms of Σ_O is more informative and more useful for further high level image analysis than the image representation in terms of Σ_I. The algorithm presented below performs this mapping by constructing, in the transducer field, a graph representation of the elements of the output alphabet, Σ_O, as well as the mapping rules which are defined by the two functions Ω and Φ, as shown in Step I in the algorithm. This graph representation is constructed by inserting in every cell in the field a labeled node to represent each symbol of Σ_O. Then, for every two neighboring cells in the field, say cells f_i and f_j, if σ_k is a node in f_i and σ_l is a node in f_j, and if (σ_k,σ_l) is

defined by the function Ω as consistent pair of neighboring cells, then we connect the two nodes σ_k and σ_l by a branch.

The graph configuration constructed in the transducer field actually contains representation of the transducer alphabet as labeled nodes and the consistency rules as branches between nodes in the neighboring cells. This is basically why the proposed technique is called a graph transducer.

When an image is presented to the transducer, the algorithm assigns to every pixel in the input image a cell in the transducer field, (Step II-i). Moreover, it performs the mapping on the input image by executing two major elimination steps on the graph configuration in the transducer field. In the first elimination step, it eliminates from each cell in the field, say cell f_i, all nodes of the set $\{\Sigma_O - \Phi(\sigma_i)\}$, where σ_i is the input symbol of the image pixel p_i to which f_i is assigned, (Step II-ii). The second elimination step is to repeatedly remove from the field all those nodes which have lost all their neighboring nodes from any of the cells adjacent to their own cell, (Step III and IV). Algorithm 4.I is presented later on in this section along with an illustrative experimental example.

In Figure 4.3, we illustrate by a simple example how the different components of the graph transducer are defined. In this example, the input alphabet is defined over single pixels of the image and take a binary value of 1, if the pixel is on the boundary of an object, and a value of 0, otherwise. The cell adjacency predicate (R), for this example, is chosen such that the field cells are configured in the form of eight-nearest neighbors, where the eight surrounding cells are considered to be adjacent to their center

cell, as shown in part (b) of the Figure. A more complete picture of such field configuration was shown in Figure 4.2. The output alphabet is defined over windows of 3×3 pixels, where some of the structural features on such windows are shown in Figure 4.3-c. The functions Ω and Φ are shown in parts (d) and (e) of the same Figure, respectively.

Algorithm 4.I was implemented on a VAX-11/780, on a bitwise manipulation form. The input and output alphabets, the neighborhood predicate, and the mapping functions are shown in Figure 4.3. A run on 64x64 test image is shown in Figures 4.4 through 4.7. The original image is shown in Figure 4.4. Figure 4.5 shows the result of applying the Sobol edge-operator on the image. In Figure 4.6, we show the result of thinning the edges . The thinned edge is then processed by Algorithm 4.I, and the output is shown in Figure 4.7.

ALGORITHM 4.I :

Purpose : To map image information contents from a set of image primitives defined over local subimages into a set of global features defined over larger subimages.

Input : 1. A graph transducer T

2. A low level image representation defined by entities from the input alphabet Σ_I over an input image field of cells P.

Output : A global description of the input image, defined by the output alphabet Σ_O over the transducer field F.

Method :

I. Construction of the graph representation in the transducer field

(i) FOR all $f \in F$ DO {
    FOR all $\sigma_O \in \Sigma_O$ DO {
        Insert a labeled node σ_O in cell f
    }
}

(ii) FOR all $(f_i, f_j) \in S$ DO {
    FOR (σ_k in cell f_i, *.AND.* σ_l in cell f_j) DO {
        IF $(\sigma_k, \sigma_l) \in \Omega(f_i, f_j)$ THEN
            Connect nodes σ_k and σ_l by a branch $\beta(\sigma_k, \sigma_l) \in B$
    }
}

II. Input image into the transducer field

(i) FOR all $p_i \in P$ DO {

Designate a cell $f_i \in F$

}

(ii) FOR all $f_i \in F$ DO {

Eliminate all nodes of the subset $\{\Sigma_O - \Phi(\sigma_i)\}$ from f_i, where $\sigma_i \in \Sigma_I$ is the input symbol in p_i

}

III. FOR all $f \in F$ DO {

FOR all $(f_i, f_j) \in S$ DO {

FOR σ_i in cell f_i DO {

IF (f_i contains more than one node, and there does not exist at least one node σ_j such that $\beta(\sigma_i, \sigma_j) \in B$) THEN

Eliminate node σ_i

}

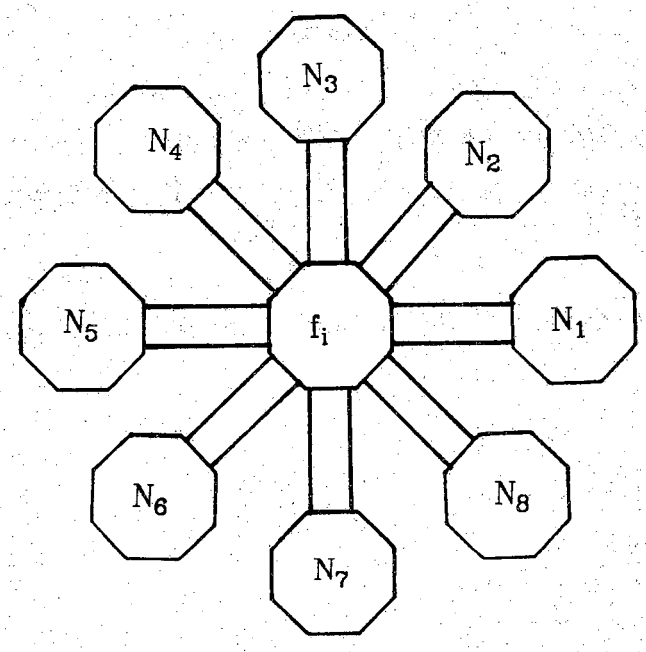
}

}

IV. Repeat Step III until no more nodes can be eliminated from the field F

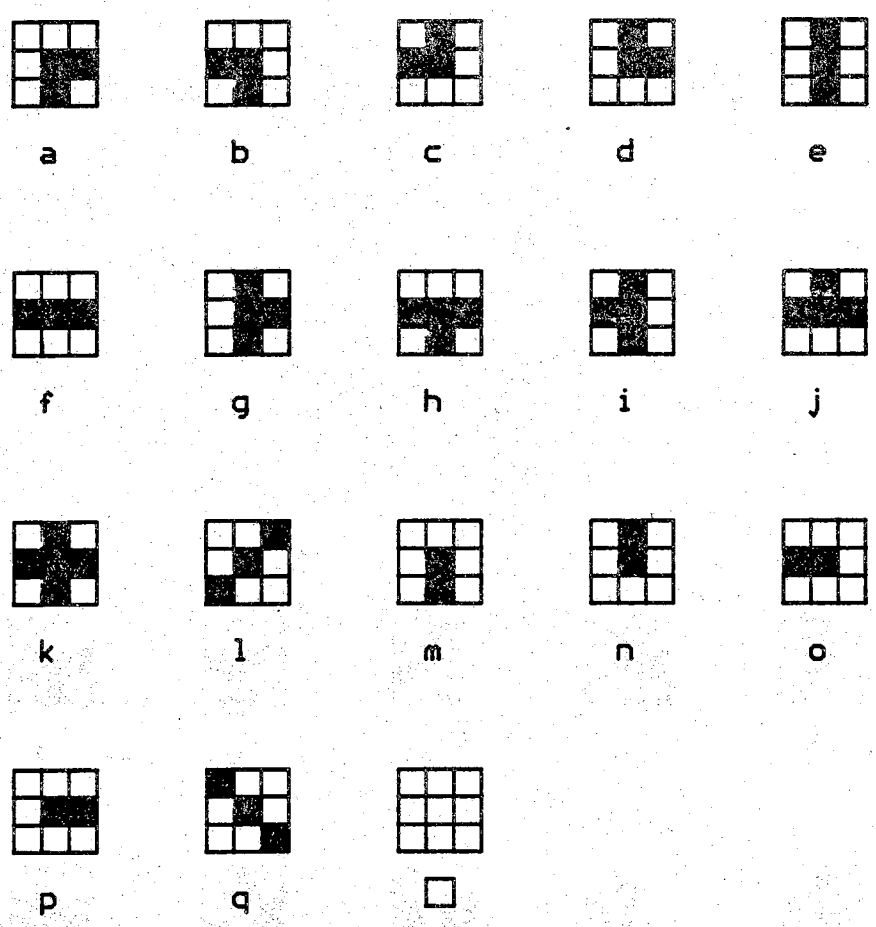
IV. END ALGORITHM 4.I.

(a) The Input Alphabet, $\Sigma_I = \{0, 1\}$



(b) The Neighborhood Configuration as Defined by Predicate R for the Field Cell f_i

Fig. 4.3 An Example of a Graph Transducer



(c) The Output Alphabet, $\Sigma_0 = \{a, b, c, \ldots\}$

Fig. 4.3 (Continued)

N_1 →

N_5 ↓

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	□
a	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	F
b	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
c	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
d	F	T	F	F	F	T	F	T	F	F	F	F	F	F	T	F	F	F
e	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
f	F	T	T	F	F	T	F	T	F	T	T	F	F	F	T	F	F	F
g	F	F	F	F	F	T	F	F	F	F	F	F	F	F	T	F	F	F
h	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	F
i	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
j	F	T	F	F	F	T	F	T	F	F	F	F	F	F	T	F	F	F
k	F	F	F	F	F	T	F	F	F	F	F	F	F	F	T	F	F	F
l	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
m	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
n	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
o	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
p	F	T	T	F	F	T	F	T	T	T	T	F	F	F	T	F	F	F
q	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
□	T	F	F	T	T	F	T	F	F	F	F	T	T	T	F	T	T	T

(d.i) Adjacent Pairs of Output Symbols for the Neighboring Cells N_1 and N_5

Fig. 4.3 (Continued)

N_2 →

N_6 ↓

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	□
a	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	T
b	F	F	F	T	F	F	F	F	F	F	F	T	F	T	F	T	T	T
c	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	T
d	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
e	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	T
f	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	T
g	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
h	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
i	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	T
j	F	F	F	F	T	F	F	F	F	F	F	F	F	F	F	F	F	T
k	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
l	F	F	F	T	F	F	F	F	F	F	F	T	F	T	F	T	T	F
m	F	F	F	T	F	F	F	F	F	F	F	T	F	T	F	T	T	T
n	F	F	T	F	F	T	F	F	F	T	F	F	F	F	T	F	F	T
o	F	F	F	F	F	F	F	F	F	F	F	F	F	T	F	T	T	T
p	T	F	F	F	T	F	T	F	F	F	F	F	T	F	F	F	F	T
q	F	F	F	F	F	F	F	F	F	F	F	F	F	T	F	T	F	T
□	T	T	T	T	T	T	T	T	T	T	T	F	T	T	T	T	T	T

(d.ii) Adjacent Pairs of Output Symbols for the Neighboring Cells N_2 and N_6

Fig. 4.3 (Continued)

N_3 →

N_7 ↓

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	□
a	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
b	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
c	T	F	F	F	T	F	T	F	F	F	F	F	T	F	F	F	F	F
d	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	F
e	T	T	F	F	T	F	T	T	T	F	T	F	T	F	F	F	F	F
f	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
g	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	F
h	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
i	T	F	F	F	T	F	T	F	F	F	F	F	T	F	F	F	F	F
j	F	F	F	F	F	F	F	F	F	F	F	F	T	F	F	F	F	F
k	F	F	F	F	T	F	T	F	F	F	F	F	T	F	F	F	F	F
l	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
m	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
n	T	T	F	F	T	F	T	T	T	F	T	F	T	F	F	F	F	F
o	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
p	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
q	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
□	F	F	T	T	F	T	F	F	F	T	F	T	F	T	T	T	T	T

(d.iii) Adjacent Pairs of Output Symbols for the Neighboring Cells N_3 and N_7

Fig. 4.3 (Continued)

N_4 →

N_8 ↓

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	□
a	F	F	T	F	F	F	F	F	F	F	F	T	F	T	T	F	T	T
b	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	T
c	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
d	F	F	F	T	F	T	F	F	F	T	F	F	F	F	F	T	F	T
e	F	F	F	T	F	T	F	F	F	T	F	F	F	F	F	T	F	T
f	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	T
g	F	F	F	T	F	T	F	F	F	F	F	F	F	F	F	F	F	T
h	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	T
i	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
j	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
k	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
l	F	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T
m	F	T	T	F	F	F	F	F	F	F	F	T	F	T	T	F	T	T
n	F	F	F	T	F	T	F	F	F	T	F	F	F	F	F	F	F	T
o	F	T	F	F	T	F	F	F	T	F	F	F	T	F	F	F	F	T
p	F	F	T	F	F	F	F	F	F	F	F	T	F	T	T	F	T	T
q	F	F	T	F	F	F	F	F	F	F	F	F	F	T	T	F	T	F
□	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	T

(d.iv) Adjacent Pairs of Output Symbols for the Neighboring Cells N_4 and N_8

(d) The Function Ω

Fig. 4.3 (Continued)

	0	1
a	T	F
b	T	F
c	T	F
d	T	F
e	T	F
f	T	F
g	T	F
h	T	F
i	T	F
j	T	F
k	T	F
l	T	F
m	T	F
n	T	F
o	T	F
p	T	F
q	T	F
□	F	T

(e) The Function Φ

Fig. 4.3 (Continued)

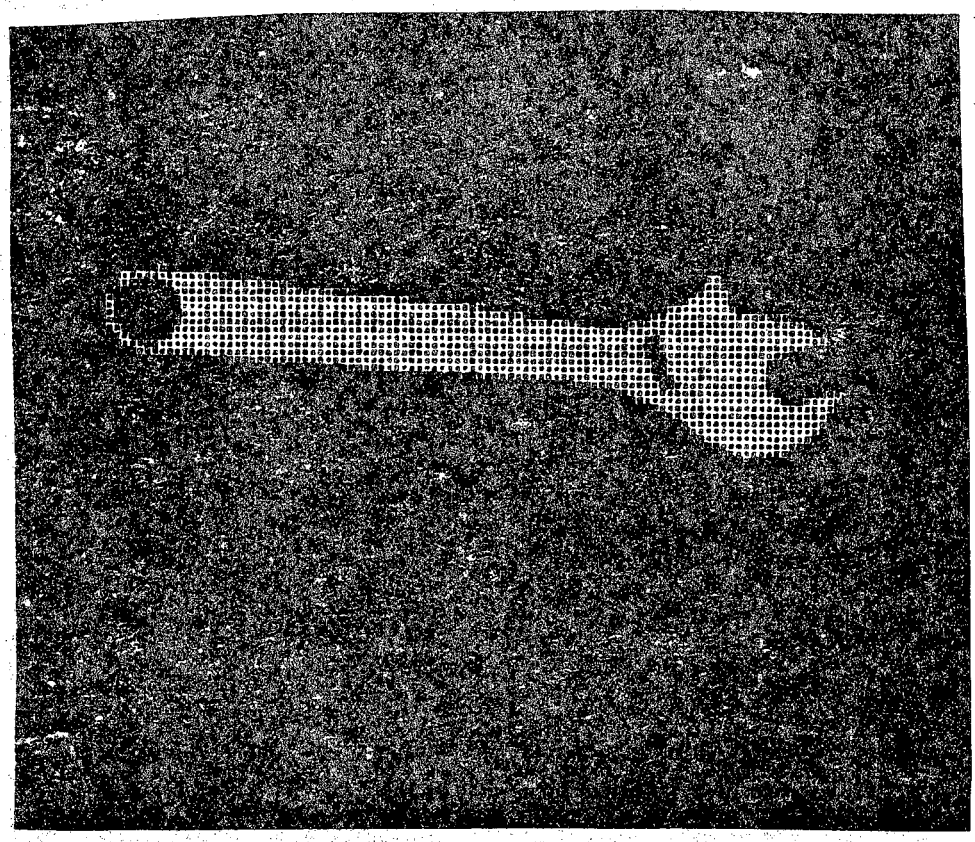


Fig. 4.4 Original 64×64 Gray Scale Image of a Machine Tool

Fig. 4.5 The Result of Sobol's Local Edge-Operator on the Image of Fig. 4.4

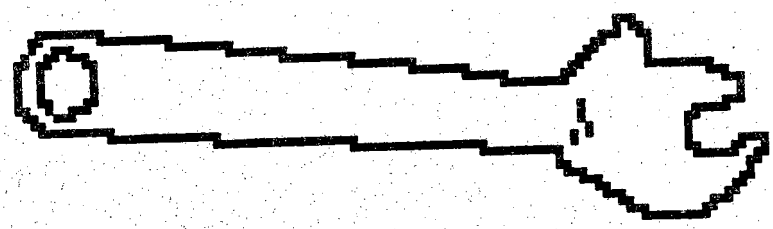


Fig. 4.6 Thinned Edges of the Image of Fig. 4.4

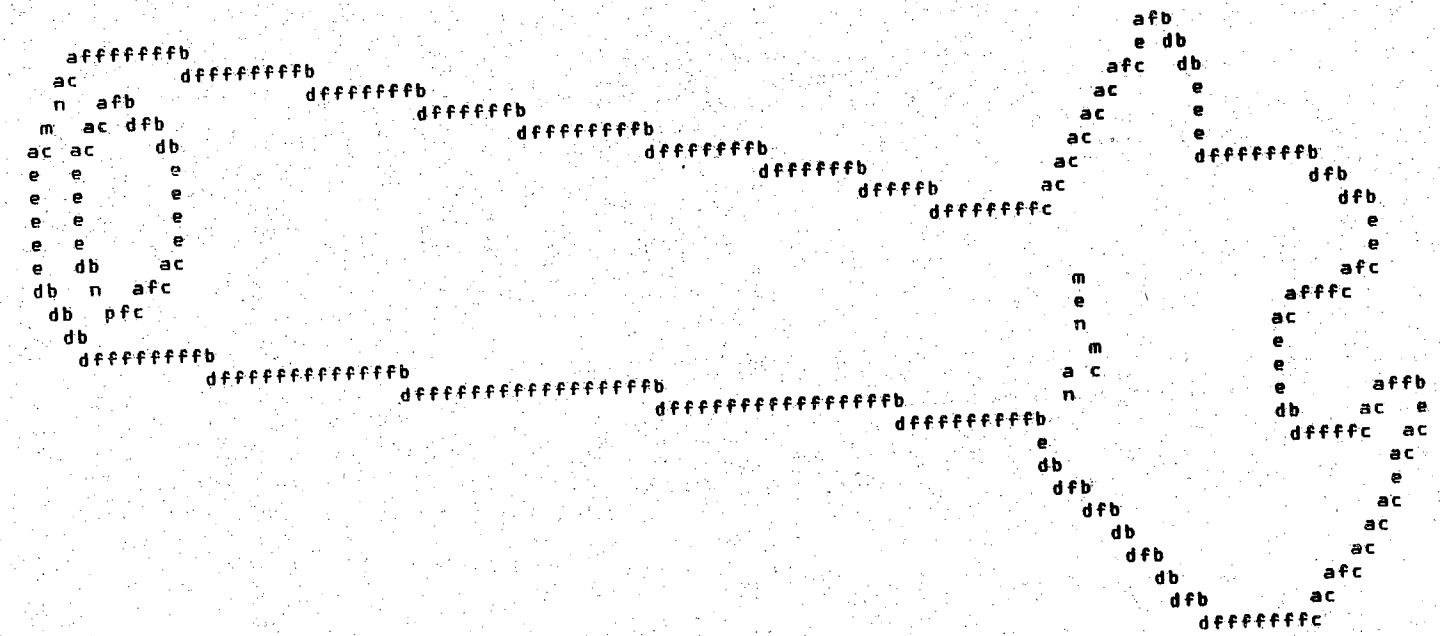


Fig. 4.7 Extracted Symbolic Representation for the Image of Fig. 4.4

4.3 Hierarchical Scheme for the Extraction of Image Global Representation

The hierarchical nature of image data is usually an important property of visual information. One of the basic concepts in the syntactic approach to computer vision and image understanding systems is based on the utilization of this property, through the recursive decomposition of complex images into relatively simple subimages which are easier to analyze than the original images. Nevertheless, the utilization of hierarchical image data structures in the extraction of image structural representations have not received proper attention of research in this field.

An important property of the graph transducer presented in Section 4.2 is the flexibility in choosing both the input and output alphabets. The size of subimages on which these alphabets are defined may be in any arbitrary size, actually it could be as small as one pixel in the image or as big as the whole image.

In this section, we propose a new scheme for the extraction of image global representation, which utilizes the graph transducer, presented in Section 4.2.2, in a hierarchical multi-stage fashion. Basicly, it is a multi-layer parallel scheme that consists of several stages each of them uses the graph transducer to perform a symbolic mapping of image information from a local alphabet into a relatively more global alphabet. In other words, the subimages over which the alphabets are defined grow in size from one layer to the other. In the first layer, the input alphabet (i.e., the set of image primitives) is defined on some very simple subimages which could be as simple as single pixels, e.g., as shown in Section 4.2. At this

stage, the scheme maps those primitive elements into a relatively more global output alphabet. Each element of the output alphabet is defined over a group of neighboring input elements, e.g., four-neighbor or eight-neighbor configurations. An example of this process was shown in Section 4.2.

In general, at an arbitrary layer in the scheme, we perform the mapping of the image information contents from an input alphabet to an output alphabet. The input alphabet of the transducer at any layer in the scheme are the output alphabet of the layer at the preceding lower-level. Elements of the output alphabet are defined as the composition of a group of elements from the input alphabet, therefore they carry relatively more global information. The input symbols that compose an element of the output alphabet occupy some neighboring cells in the field, as defined by the neighborhood predicate (R) of the transducer at this layer. The formulation of output symbols from groups of input symbols is defined by the mapping function (Ω). A more formal description of the proposed scheme is presented below.

We assume that the input image consists of a field of cells which take values over a primitive alphabet and are arranged as an M×N array. A typical example is shown in Figure 4.6, which is a thinned edge image of an object. The objective is to extract the global features from the image. We utilize the graph transducer, of Section 4.2, at different layers in the scheme. Each layer in this scheme consists of a field of cells on which the graph transducer operates. An explicit representation for an example configuration of the field of the multi-layer scheme is shown in Figure 4.8. It can be considered as similar to an overlapping pyramid structure. It has in

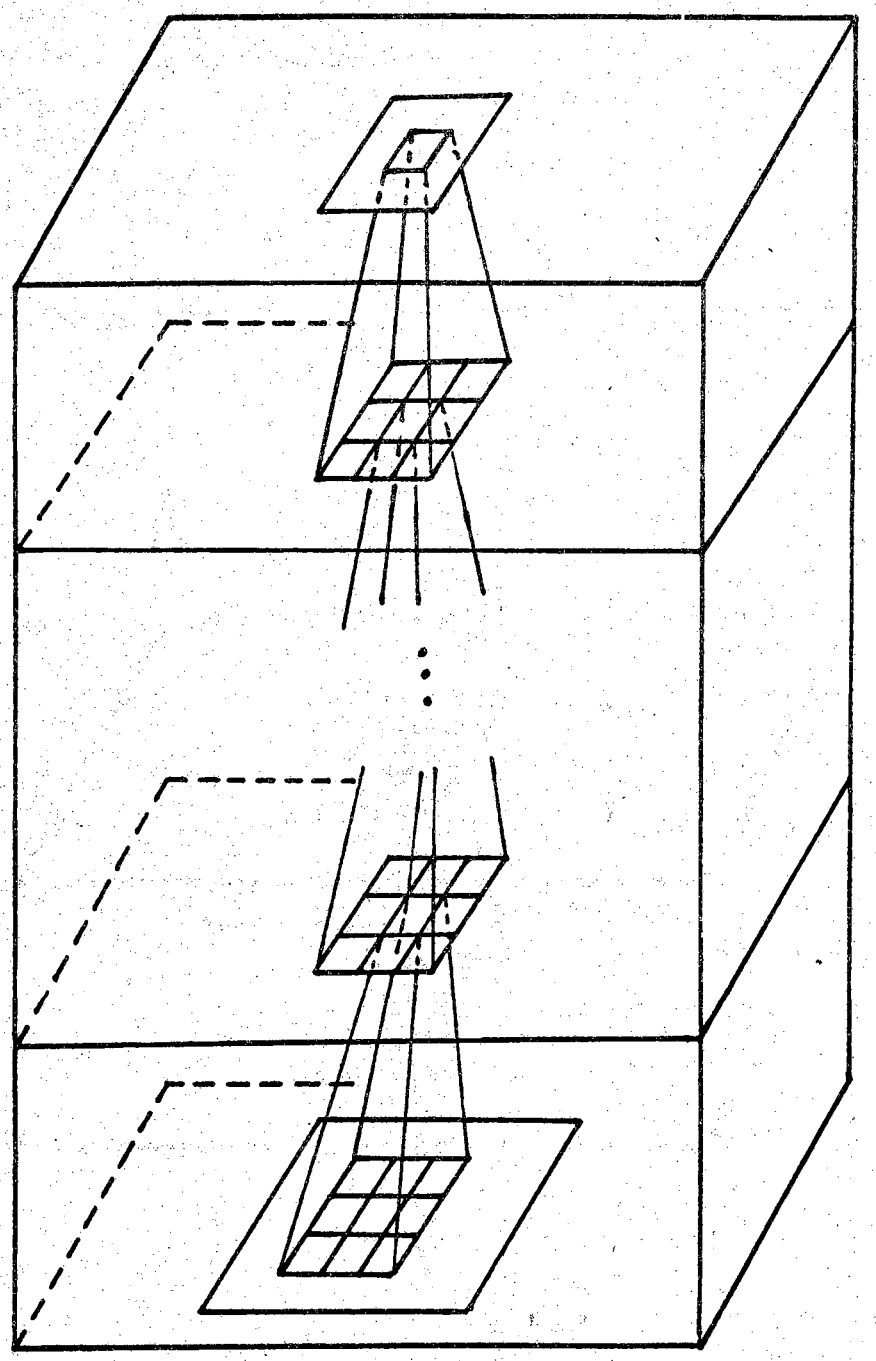


Fig. 4.8 Hierarchical Scheme for the Extraction of Image Global Representation

common with the pyramid structures that the higher the level of a layer, the more global regions its cells cover. On the other hand, it differs from pyramid structures since, in pyramid structures the number of cells in the layers decreases as we move up the pyramid, but the layers of the proposed scheme is assumed to have the same number of cells, such that the scheme would not be sensitive to the location of the extracted features in the image.

The structure shown in Figure 4.8 is explicit in the sense that when the scheme is implemented, only two layers of arrays are needed where the image representation iterates from one array to the other at each layer of processing. Through the graph transducer, each cell in a layer is assigned a symbol from the output alphabet of the transducer. A symbol of an output alphabet is composed from a group of symbols from the input alphabet that reside in a certain neighborhood as defined by the neighborhood predicate of the transducer, e.g., eight-nearest neighbors. The multi-layer graph transducer which is utilized in this scheme is described in Section 4.4.

4.4 Multi-Layer Graph Transducer for the Extraction of Attributed Structural Representation

The idea behind the proposed scheme is essentially based on the simplification of a complex object, or a global feature, by decomposing it into simpler sub-objects, or more local features. The decomposition is made by considering that a complex object consists of a set of simpler sub-objects with certain neighborhood relations between them. In other words, complex objects are decomposed,

recursively, in a top-down fashion into sets of sub-objects (or image features). The input alphabets, the output alphabets, and the mapping performed by the graph transducer which are used at different layers in the proposed scheme are designed according to this recursive decomposition. Figure 4.9 shows the recursive decomposition of a machine tool into simple image features. It is clear that the boundary of any complex object can be decomposed down into a set of global features. These global features are composed of some local features, which are in turn composed of some relatively more local features and finally of some very basic image primitives, e.g., binary image pixels similar to the one shown in Figure 4.5.

The output alphabet Σ_O of the multi-layer graph transducer consists of subsets of feature alphabets, i.e., $\Sigma_O = \{\Sigma_O^1 \cup \Sigma_O^2 \cup \ \ldots \ \cup \Sigma_O^L\}$, each of which denotes the subset of the alphabet at one of the layers of the scheme; Σ_O^l is the subset at the l'th layer. Elements of the alphabet at layer l is formed from groups of elements of the alphabet at layer $l-1$, such that members of that group occupy a certain neighborhood configuration as defined by the predicate R of the graph transducer at this layer.

The input alphabet, Σ_I, which is the input to the first layer of the scheme, consists of a set of symbols in which the input images are presented. This alphabet is usually chosen to be easily measurable. An input image is presented to the scheme as a field of cells that take values over this input alphabet. At the first layer of the scheme, the transducer performs a mapping of the image input symbols into symbols of Σ_O^1, which represents the alphabet of the transformation at the first layer. Each element of Σ_O^1 is

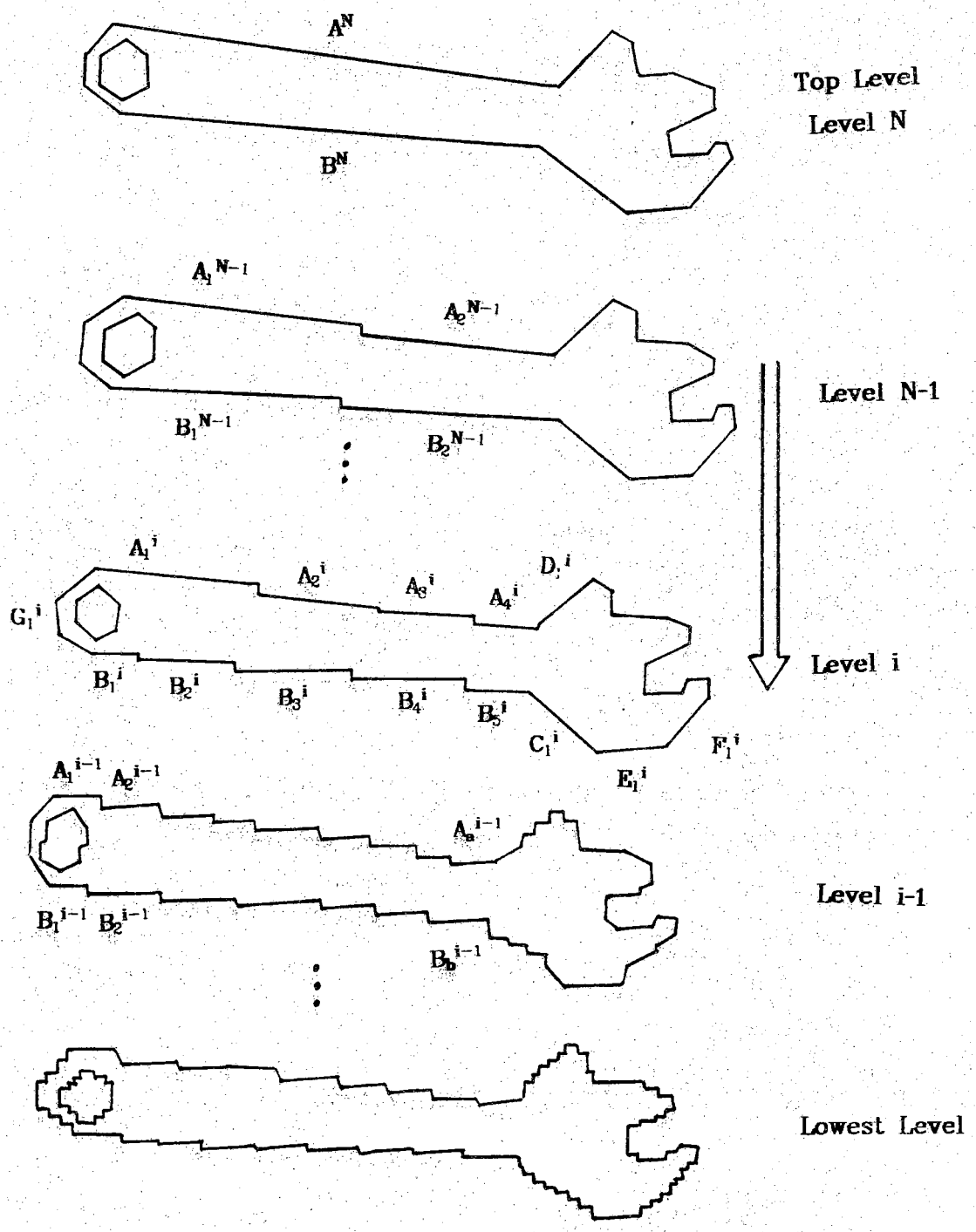


Fig. 4.9 Hierarchical Decomposition of an Object into Alphabets of Symbolic Features and Image Primitives

composed of a group of elements from Σ_I which are arranged according to the neighborhood predicate R at this layer. The image is now represented by the array of cells at this layer of the scheme, which are assigned symbols from the subset of alphabet at this layer, $\Sigma_O{}^1$, as shown by the example in Figure 4.3.

The output of the first layer is taken as input to the second layer of the scheme. In the second layer, L_2, the scheme maps elements of $\Sigma_O{}^1$ into elements of the subset $\Sigma_O{}^2$, which represent the output alphabet of the transducer at this layer. Elements of $\Sigma_O{}^2$ are composed of elements from $\Sigma_O{}^1$, according to the neighborhood predicate R. In other words, relatively more global features are extracted from the field of the first layer into cells of the field of the second layer. The symbolic mapping at this layer is performed in a similar manner as the mapping at the previous layer.

In general, at any layer l of the scheme, the transducer performs a mapping of elements of the output alphabet of the preceding layer, $l-1$, (i.e., subset $\Sigma_O{}^{l-1}$) into the subset $\Sigma_O{}^l$, $\Sigma_O{}^l \subseteq \Sigma_O$, whose elements are formed from those elements of $\Sigma_O{}^{l-1}$ that are lying in a certain neighborhood configuration as defined by the adjacency predicate R at this layer. Thus, the relatively local symbols of the cells of layer $(l-1)$ are mapped into more global features defined over cells of the l'th layer of the scheme.

This hierarchical mapping continues in the scheme with growing size alphabet until it satisfies the size of the global features, which are sought to be extracted from the image, or theoretically until the size of the alphabet covers the whole image. It is clear that the size of the subimages covered by the transducer alphabets grows

very rapidly with the order of the processing layer in the scheme, (in some cases it grows exponentially, as we will show in the experimental results). Also, the objective of this scheme is not to attempt to recognize whole complex objects, but rather to extract the global symbolic representation which can then be represented by an Attributed Relational Graph (ARG) for some further higher level analysis, such as the matching with some prototype object models through defining the distance between images by calculating a distance measure and inexact matching between their respective ARG representations, as we will demonstrate in the comming chapters of this thesis.

The alphabet of the different layers of the transducer, namely, Σ_O^1, Σ_O^2, . . . , and Σ_O^L, are defined to describe the features of a certain class of objects through the decomposition of these objects into sub-objects and the recursive decomposition of global features into local features. Each element in the alphabet of a certain layer represents a feature contained in the subimage covered by the cells of that layer.

Formally, we define the multi-layer graph transducer Γ of the form :

$$\Gamma = \{ \Sigma_I, \Sigma_O, F, R, \Omega, \Phi \}$$

where :

Σ_I is an input alphabet,

Σ_O is an output alphabet, $\Sigma_O = \{\Sigma_O^1 \cup \Sigma_O^2 \cup \ \ldots \ \cup \Sigma_O^L\}$, where L is the number of layers in the scheme,

F is a multi-layer field of cells, $F = F^1 \cup F^2 \cup F^3 \ \ldots \ \cup F^L$,

where F^l , $1 \le l \le L$, is the l'th layer in the field,

R is a symmetric predicate that defines the adjacency between every two cells in F; let S denotes the set of cell pairs as defined by R,

Ω is a function that defines pairs of alphabet symbols for every pair of adjacent cells in S,

Φ $(\Phi(\sigma_i) \subseteq \Sigma_O)$ is a function that defines the correspondence between subsets of $\Sigma_O{}^1$ for every input symbol, and between alphabet symbols of different layers of the scheme.

4.5 Extraction of Attributed Relational Graph Representation

The multi-layer hierarchical scheme, presented in the last section, performs mapping of the image information contents from the input alphabet Σ_I, e.g., the spatial domain representation, into the output alphabet of global features, Σ_O. The output field of the scheme is a two-dimensional array of cells which take values over Σ_O. Nevertheless, the same information contained in that array can be better represented by an Attributed Relational Graph. The attributed nodes of the graph represent the different features extracted from the image, with the parameters of the features as attributes attached to the nodes, e.g., the length of a straight line segment, as reflected by the order of the layer in which it appears, or the length and curvature of a curve segment, etc. The relations between these global features, e.g., their relative positions, are represented by attributed branches between the respective nodes in the ARG representation.

In Fig. 4.10-(a), we show the features extracted from the image of Fig. 4.4, where we use the linear approximation of the edge features of the object as the image features. The length of the line segments represent the attribute of the respective nodes. If two lines are joint or are parallel, the relations Joint (J: ϑ) or Parallel (P: d), respectively, are represented by branches between the respective nodes, where ϑ is the joint angle and d is the distance between the two line segments. Fig. 4.10-(b) shows the relations between features for the image of Fig. 4.4. The ARG representation of the same image is shown in Fig. 4.11. Some more examples will be shown with the experimental results in Chapter 6.

GLOBAL FEATURES EXTRACTED FROM THE TRANSFORMATION FIELD :

feature no.	type	attributes
1	1	65
2	1	12
3	1	66
4	1	14
5	1	8
6	1	14
7	1	6
8	1	8
9	1	8
10	1	12
11	1	12
12	1	12
13	1	10
14	1	10
15	1	8
16	1	8
17	1	6
18	1	4

(a) The Extracted Features

Fig. 4.10 Global Features and Relations for the Image of Fig. 4.4

RELATIONS BETWEEN THE EXTRACTED FEATURES :

relation	type	attributes
1, 2	j	90
2, 3	j	90
3, 4	j	120
4, 5	j	120
5, 6	j	90
6, 7	j	90
7, 8	j	90
8, 9	j	90
9, 10	j	90
10, 11	j	45
11, 12	j	120
12, 13	j	120
14, 15	j	90
15, 16	j	90
16, 17	j	90
18, 9	p	14
1, 3	p	12
1, 13	j	120
2, 14	p	3
6, 8	p	6
8, 10	p	6

(b) The Extracted Relations

Fig. 4.10 (Continued)

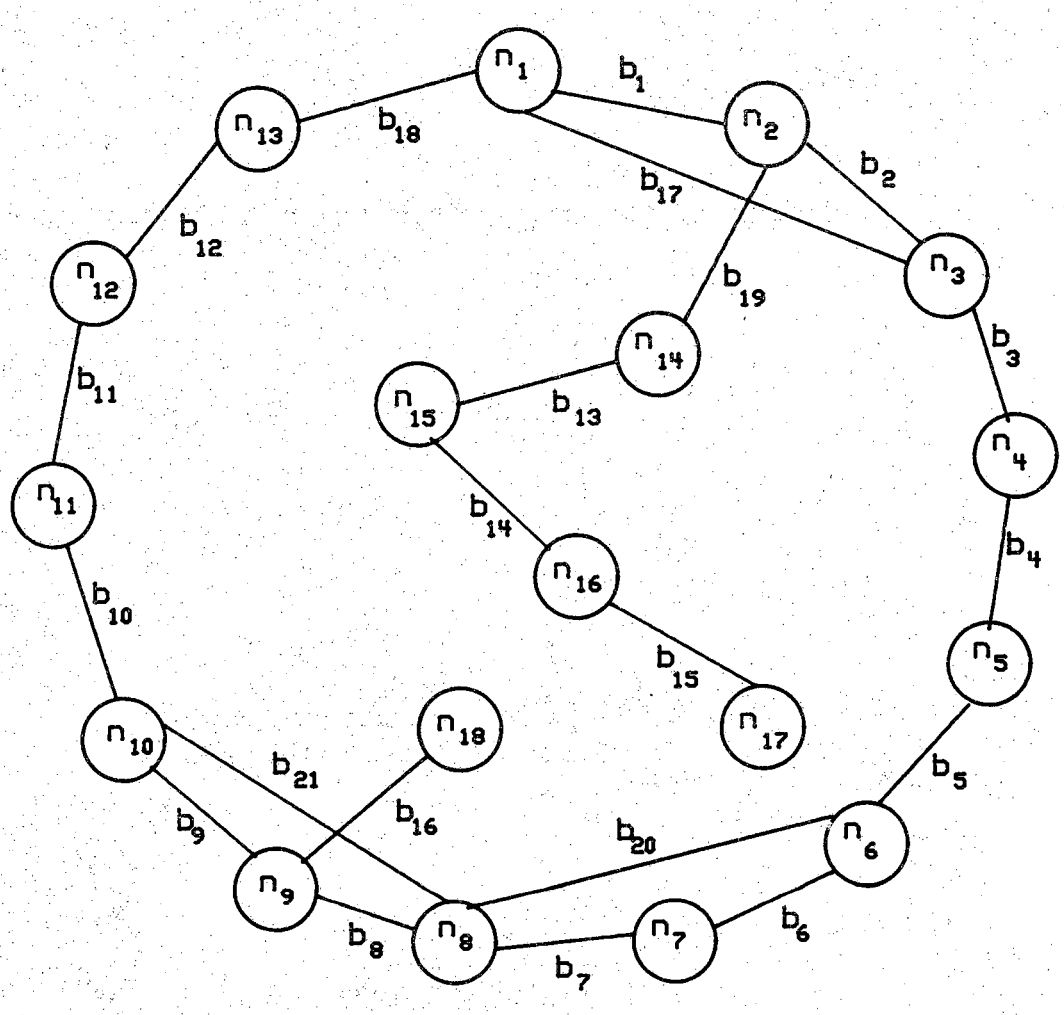


Fig. 4.11 Attributed Relational Graph Representation for the Image of Fig. 4.4

CHAPTER V

A DYNAMIC PROGRAMMING APPROACH FOR DISTANCE MEASURE AND INEXACT MATCHING BETWEEN ATTRIBUTED RELATIONAL GRAPHS

5.1 General

In Chapter 4, we presented a powerful hierarchical scheme for the extraction of Attributed Relational Graph Representation (ARG) from images, which, as we discussed before in Section 1.1, represent the first phase of most computer vision and image understanding systems. On the other hand, the actual analysis and interpretation of the image is usually performed on the extracted representation and it takes place in the second phase of the system.

One of the very important concepts in information processing and analysis is the concept of distance or similarity measures between different entities. In this Chapter, we present a new approach for calculating distance or similarity measures between pairs of attributed relational graphs, in their general form. The graphs may possess arbitrary relative size. The importance of this

problem in computer vision and image understanding arises due to the following three main reasons:

(1) Graphs are very powerful representation tools for several practical applications in image analysis and understanding.

(2) The need for a useful distance measure is very essential in handling any real images.

(3) The results are applicable to several interesting image understanding and pattern recognition problems, such as locating objects in noisy and defected images, error-correcting parsing of ARGs, multi-resolution analysis of images, high-level image segmentation, etc., The approach is also applicable to other application problems in machine intelligence and knowledge representation, where relational structures are chosen as the mean of knowledge representation.

In Section 5.2, we state some formal definitions and terminology based on which we present our formulation to the problem of calculating a distance measure and finding the best inexact matching configuration between pairs of attributed relational graphs. In Section 5.3, a detailed word-description, formal formulation and the underlying concepts of our approach to this problem are given along with formal algorithms for the main skeleton of the approach. The approach, basically, divides the problem into a set of subproblems and constructs a state-space scheme, using subgraphs

to serve as handlers to govern the transition from a state to another, and to define a dominance criterion. In Section 5.4, we analyze the computational complexity of the approach.

5.2 Basic Definitions and Terminology

As we discussed earlier in Chapter 2, the type of graphs which best serves in vision systems is the attributed relational graphs. The formal definition of an Attributed Relational Graph, which, from now on, we may denote as ARG or simply Graph, was given in Section 2.4.2. For the simplicity of notation, we may denote the node and branch attribute vectors by labels, which can take any form generated by the attribute functions. Moreover, we assume that the node as well as the branch attributes are generated by general (many-to-one) functions, i.e., allowing more than one node in the graph to have the same attribute (or label) and more than one branch to have the same attribute (or label). This appears to be an important requirement in graph representation of images in order to reduce the preprocessing which should be performed on the images before and during extracting their graph representation.

For the purpose of representation and for the ease of notation, an ARG is represented by a doublet of the form:

$$G = (N^G, B^G)$$

where,

N^G: denotes the set of attributed-nodes of graph G, with the attribute alphabet A^G and defined as:

$$N^G = \left\{ (i,a) \mid 1 \le i \le |N^G|,\ a \in A^G \right\}$$

B^G: denotes the set of directed attributed-branches of graph G, with attribute alphabet E^G and defined as,

$$B^G = \left\{ (i,j,e) \mid 1 \le i,j \le |N^G|,\ e \in E^G \right\}$$

also, (i,a)$\in N^G$ denotes node i with 'a' as an attribute and (i,j,e)$\in B^G$ denotes the directed branch from node i to node j with 'e' as its attribute.

Definition 5.1 : A BASIC Attributed Relational Graph (BARG or Basic graph) is a graph on the form of one level tree, i.e., it consists of a root node, the branches emanating from it, and the nodes on which these branches terminate.

Although, as we defined before, a BARG is just a special case of ARG, we would like to represent it in a slightly different way. Usually in our work, we define BARGs as subgraphs of an ARG. If G_i is a basic graph in an ARG G, $G_i \subseteq G=(N^G,B^G)$, we denote the BARG G_i as follows:

$$G_i = (r^{G_i}, B^{G_i}, L^{G_i})$$

where:

r^{G_i}: $r^{G_i}=(i,a)\in N^G$ is the root node of G_i,

B^{G_i}: $B^{G_i}=\{(i,j,e) \mid (i,j,e)\in B^G\}$ is the set of branches emanating from r^{G_i}, and

L^{G_i}: $L^{G_i}=\{(j,c)\in N^G \mid (i,j,e)\in B^G\}$ is the set of leaf nodes of G_i.

We adopt the definition of the distance between two ARGs as, in its basic form, the minimum number of changes (or error-transformations) which need to be performed on one graph in order to produce the other graph. These changes are usually in the form of node or branch deletion, insertion, or substitution (i.e., relabeling). In addition, we associate a certain cost with each of these transformations. For example, let w_{ni}, w_{nd}, and w_{ns} be the costs of node insertion, deletion, and relabeling respectively; and let w_{bi}, w_{bd}, and w_{bs} be the costs associated with branch insertion, deletion and relabeling, respectively.

Definition 5.2 : The distance between two ARGs, say U and V, is defined as the cost of the sequence of transformations which possesses minimum total cost and which must be performed on one of the two ARGs, say graph U (or on a subgraph of U), in order to produce the other ARG, i.e., graph V.

In general, the cost of error-transformations may depend on the nodes or the branches on which they are performed (i.e., a function of the attributes of the nodes or the branches). For simplicity of notation in this chapter, we assume that these costs are constant with respect to the same type of transformation. However, our formulation and approach to the problem are designed to handle the case of modified costs, in which costs are functions of the nodes or branches. In this case, these costs will encompass the difference in attributes (or labels) of different nodes or branches.

Definition 5.3 : The Empty Graph (denoted by Λ) is the graph of zero nodes (and no branches), i.e., $\Lambda=(\Phi,\Phi)$.

Definition 5.4 : For an ARG, $G=(N^G,B^G)$, we define $\hat{G}$ as the set of all BARGs contained in G.

Definition 5.5 : The set of core nodes, C_G, of an ARG G is defined as the set of root nodes of the BARGs in $\hat{G}$, i.e., $C_G=\left\{ r^{G_i} \mid (r^{G_i},B^{G_i},L^{G_i})\in\hat{G} \right\}$.

Definition 5.6 : The set of terminal nodes, T_G, of an ARG G is defined as all nodes of G which are not core nodes, i.e., $T_G=N^G-C_G$.

5.3 Distance Calculation and Inexact Matching for Attributed Relational Graphs

5.3.1 General

The calculation of a distance measure between two ARGs involves not only finding a sequence of error transformations for producing one ARG (or a subgraph of it) from the other, but also finding such a sequence which possesses the minimum total cost. Therefore, an essential part of the problem involves optimization over all valid sequences of transformations to find the one with minimum total cost. For the distance measure to be informative, the cost of error tansformations should depend on the nodes and branches on which

these transformations are applied. For any two given ARGs, U and V, the approach for distance calculation between U and V or their subgraphs can be outlined by the three major steps, as shown in Figure 5.1.

First, we decompose U and V into two sets of BARGs denoted as $\hat{U}=\{U_1, U_2, \ldots\ldots, U_M\}$ and $\hat{V}=\{V_1, V_2, \ldots\ldots, V_N\}$, where M and N are the number of nodes in U and V, respectively. The distance between two BARGs is defined as in Section 5.2, since BARGs are just a special case of ARGs. This distance can be easily calculated because of the simple structure of the BARGs.

Second, we reconstruct U and V starting from a pair of empty graphs and iteratively embed into them matched pairs of Basic graphs from the two sets $\{\hat{U} \cup \{\Lambda\}\}$ and $\{\hat{V} \cup \{\Lambda\}\}$, according to some feasibility criterion as we will discuss in Section 5.3.4, until no more pairs can be added. This reconstruction process generates the state-space representation of the problem, in which each state denotes the reconstruction of a pair of subgraphs from U and V.

The Initial State in this representation scheme denotes the starting of the matching process, and the final states are those states denoting the successful completion of reconstructing graph U, graph V, or both. The transition from a state to another state represents the embedding of a pair of matched basic graphs into the already reconstructed subgraphs; the weight on this branch denotes the incremental cost due to this embedding operation. A heuristic criterion is defined to govern the possible next matched pairs of basic graphs which can be embedded at each state in the representation scheme. The concepts and the details of this

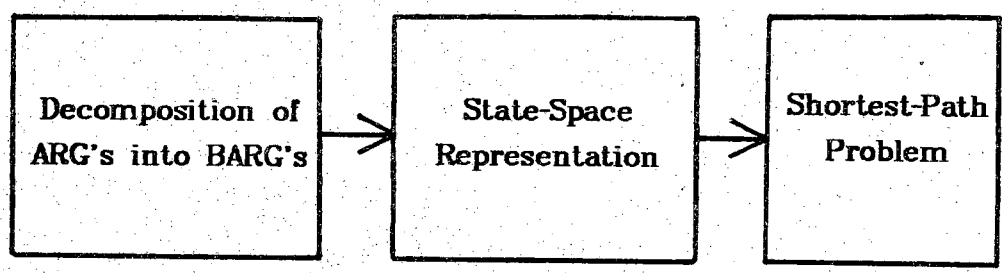


Fig. 5.1 General Flowchart for the Proposed Approach to the Graph Distance and Inexact Matching Problem

criterion are presented in Section 5.3.4. Usually, the incremental cost is not only a function of the nodes and branches on which an error transformation is applied, but it also depends on the matching configuration of the already reconstructed subgraphs. Therefore, it is clear that the resulting state-space representation scheme is, in general, on the form of a Directed Branch-weighted Lattice.

The third and the last step of the approach is to find the Shortest-Path over the Directed Acyclic Branch-weighted Lattice from the initial state to a state in the set of final states. This optimization problem, for the type of lattice that the approach generates, can be solved in linear time by Dynamic Programming. The total weight of the shortest-path denotes the distance measure between the two ARGs, while its final state denotes the best inexact matching configuration between the two graphs.

5.3.2 Decomposition of ARGs into Sets of Basic Graphs (BARGs)

Suppose that two ARGs, U and V, of M and N nodes respectively, are represented by: $U=(N^U,B^U)$, and $V=(N^V,B^V)$; A^U and A^V denote the alphabets of node attributes, and E^U and E^V denote the alphabets of branch attributes for graph U and graph V, respectively. The first main step in the approach is to decompose each of the two graphs, U and V, into a set of basic graphs (BARGs), namely sets $\hat{U} = \{U_1, U_2, \ldots, U_M\}$, and $\hat{V} = \{V_1, V_2, \ldots, V_N\}$, respectively.

In Figure 5.2, we show an example of two ARGs; their corresponding sets of Basic graphs are shown in Figure 5.3. It is clear that the decomposition of graphs into sets of BARGs is unique. It is achieved by Algorithm 5.I, as given below.

ALGORITHM 5.I :

Purpose : To decompose an ARG into a set of Basic ARGs.

Input : An ARG, graph G, represented by

$$N^G = \left\{(i,a) \mid i \in \{1,2,\ldots,|N^G|\},\ a \in A^G\right\} \text{ and}$$

$$B^G = \left\{(i,j,b) \mid i\&j \in \{1,2,\ldots,|N^G|\},\ b \in E^G\right\}.$$

Output : A set $\hat{G} = \{G_1, G_2, \ldots, G_{|N^G|}\}$, such that, $G_i \in \hat{G}$ is a Basic ARG represented as $G_i = (r^{G_i}, B^{G_i}, L^{G_i})$, where:

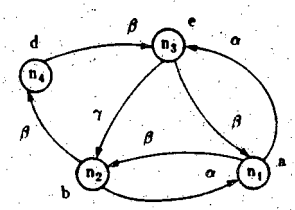
r^{G_i} is the root node of G_i,

B^{G_i} is the set of branches emanating from r^{G_i}, and

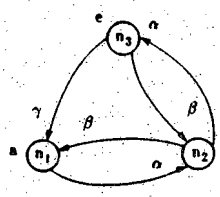
L^{G_i} is the set of leaf nodes of G_i.

Method :

```
FOR i:=1 to |N^G| DO
    BEGIN
        (1) r^Gi ← (i,a)
        (2) B^Gi ← {(i,j,b)|(i,j,b)∈B^G}
        (3) L^Gi ← {(j,c)|(j,c)∈N^G, (i,j,b)∈B^Gi}
        (4) G_i ← (r^Gi,B^Gi,L^Gi)
    END
END ALGORITHM 5.I
```

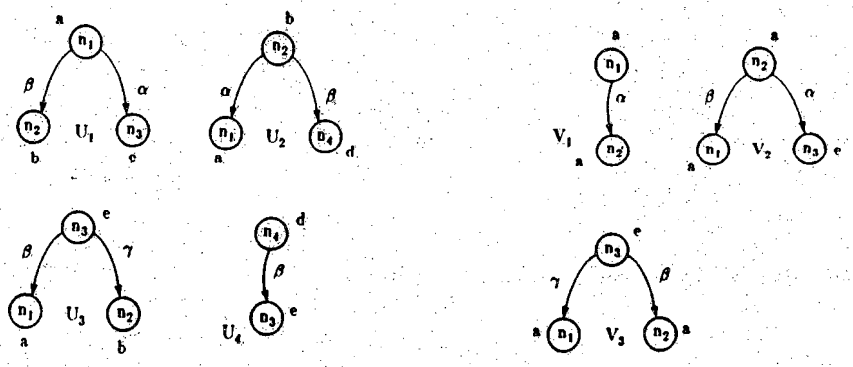


Graph U



Graph V

Fig. 5.2 Two ARG's U and V



$\hat{U} = \{U_1, U_2, U_3, U_4\}$ $\hat{V} = \{V_1, V_2, V_3\}$

Fig. 5.3 The Sets of Basic Graphs for the ARG's Shown in Fig. 5.2

5.3.3 Construction of State-Space Representation

The state-space representation used in this approach is in the form of a Directed Acyclic Labeled Lattice. Such a lattice consists of a set of labeled states, say set $S=\{s_0,s_1,s_2,\ldots,s_L\}$, where $L+1$ is the total number of states, and a set of directed labeled branches R, of the form $R\subseteq\{(s_I,s_J)|s_I\&s_J\in S\}$ where (s_I,s_J) denotes a branch emanating from state s_I to state s_J. For two AGRs U and V, each state in this lattice denotes the reconstruction of a subgraph from graph U and a subgraph from graph V, as well as the matching of their respective basic graphs, according to the feasibility criterion which is shown in Section 5.3.4. For example, a state, s_I, denotes the reconstruction of subgraphs X_I and Y_I from graphs U and V, respectively. Each of X_I and Y_I is composed of a set of BARGs, namely $\hat{X}_I\subseteq\{\hat{U}\cup\{\Lambda\}\}$ and $\hat{Y}_I\subseteq\{\hat{V}\cup\{\Lambda\}\}$. The basic graphs in $\hat{X}_I$ and $\hat{Y}_I$ are matched in pairs in the form (X_i,Y_i), where $X_i\in\hat{X}_I$ and $Y_i\in\hat{Y}_I$. Therefore, each state in the lattice is labeled with a set of matched pairs of basic graphs in the form: $s_I=(X_I,Y_I)=\left\{(X_i,Y_i)\,|\,X_i\in\hat{X}_I,Y_i\in\hat{Y}_I\right\}$. On the other hand, each branch in the lattice, say branch (s_I,s_J) between state $s_I=(X_I,Y_I)$ and state $s_J=(X_J,Y_J)$, is labeled by $w(s_I,s_J)$, which represents the cost of the transition between the two states, i.e., state s_I and state s_J.

In the reconstruction process, the transition from state s_I into state s_J represents the embedding of a pair of BARGs X_k and Y_k into the already reconstructed subgraphs X_I and Y_I in order to produce subgraphs X_J and Y_J, where $X_k\in\left\{\{\hat{U}-\hat{X}_I\}\cup\{\Lambda\}\right\}$, $Y_k\in\left\{\{\hat{V}-\hat{Y}_I\}\cup\{\Lambda\}\right\}$ and $(X_k,Y_k)\neq(\Lambda,\Lambda)$. The new subgraphs X_J and Y_J are composed of the two

sets of BARGs $\hat{X}_J=\hat{X}_I\cup\{X_k\}$ and $\hat{Y}_J=\hat{Y}_I\cup\{Y_k\}$. The embedding of X_k on X_I and Y_k on Y_I is performed in a straight forward fashion according to the configuration of the original ARGs U and V.

The new pair of subgraphs X_J and Y_J is represented in the state-space lattice by a state s_J. The transition from state s_I to state s_J is represented by a branch (s_I,s_J) with an associated cost $w(s_I,s_J)=d_{(X_I,Y_I)}(X_k,Y_k)$, which denotes the incremental distance due to the embedding of X_k and Y_k into subgraphs X_I and Y_I to produce the new subgraphs X_J and Y_J, respectively. This incremental distance is, in general, a function of the matched nodes and branches of X_k and Y_k, as well as of the matching configuration of subgraphs X_I and Y_I. The evaluation of this incremental distance depends on the the application problem in hand, i.e. the practical significance of each node and branch in both ARGs, their attribute alphabets, etc; as it is the case for string and tree distance measures.

A designated state in the lattice is the "Initial State", which is denoted as $s_0\in S$, labeled as (Λ,Λ) and representing the starting point of the graph reconstruction process where both subgraphs are empty. A formal algorithm for reconstructing the graphs and generating the lattice will be given below, and referred to as Algorithm 5.II.

Before presenting Algorithm 5.II, we describe it informally. First, for $U_i\in\hat{U}$ and $V_j\in\hat{V}$ we generate states labeled as $s_I=(X_I,Y_I)$, where, $X_I=U_i$ and $Y_I=V_j$. The weight of a branch (s_0,s_I) from the initial state s_0 to state s_I is $w(s_0,s_I)=\mathrm{d}(U_i,V_j)$, i.e., the distance between the two basic graphs U_i and V_j. In Figure 5.4, we show the state-space representation generated via reconstructing the two

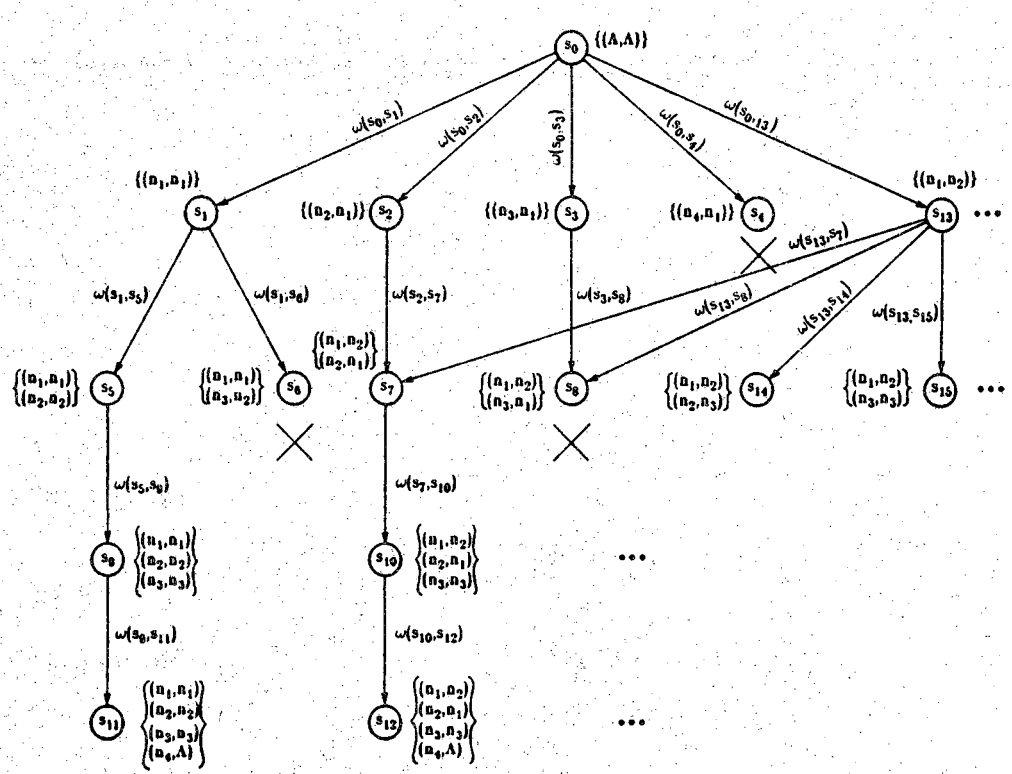


Fig. 5.4 State-Space Representation for the Distance Measure and Inexact Matching between ARG's

ARGs U and V which are shown in Figure 5.2 from the sets of BARGs shown in Figure 5.3. The significance and the information represented by each state are explained below and are shown in Figure 5.5.

At any arbitrary state $s_I=(X_I,Y_I)\in S$, we define the following:

(i) $C(s_I)$: $C(s_I)=\left\{(r^{X_i},r^{Y_i}) \mid X_i\in\hat{X}_I, Y_i\in\hat{Y}_I, (X_i,Y_i)\in(X_I,Y_I)\right\}$ as the set of matched pairs of core nodes in subgraphs X_I and Y_I,

(ii) $C_X(s_I)$: $C_X(s_I)=\left\{r^{X_i} \mid (r^{X_i},r^{Y_i})\in C(s_I)\right\}$ as the set of core nodes of subgraph X_I,

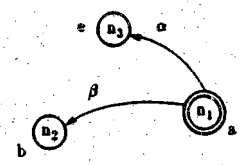
(iii) $C_Y(s_I)$: $C_Y(s_I)=\left\{r^{Y_i} \mid (r^{X_i},r^{Y_i})\in C(s_I)\right\}$ as the set of core nodes of subgraph Y_I,

(iv) $T_X(s_I)$: $T_X(s_I)=\left\{L^{X_i} \mid X_i\in\{N^{X_I}-C_X(s_I)\}\right\}$ as the set of terminal nodes of subgraph X_I,

(v) $T_Y(s_I)$: $T_Y(s_I)=\left\{L^{Y_i} \mid Y_i\in\{N^{Y_I}-C_Y(s_I)\}\right\}$ as the set of terminal nodes of subgraph Y_I.

At any state $s_I=(X_I,Y_I)\in S$, we have the set of core nodes of X_I and the set of core nodes of Y_I in the form of matched pairs, as represented by C(s_I). The approach proceeds by expanding state s_I through the embedding of a new pair of Basic graphs into X_I and Y_I and adding a new pair of matched nodes into C_X and C_Y. This new matched pair is chosen from the terminal-node sets T_X and T_Y

At state s_1:



Subgraph X_1

Subgraph Y_1

$C(s_1) = \{(n_1,n_1)\}$

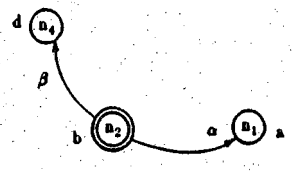
$C_X(s_1) = \{n_1\}$ $T_X(s_1) = \{n_2,n_3\}$

$C_Y(s_1) = \{n_1\}$ $T_Y(s_1) = \{n_2\}$

$P(s_1) = \{(U_2,V_2),(U_3,V_2)\}$

$Q(s_1) = \{(U_2,V_2),(U_3,V_2)\}$

At state s_2:



Subgraph X_2

Subgraph Y_2

$C(s_2) = \{(n_2,n_1)\}$

$C_X(s_2) = \{n_2\}$ $T_X(s_2) = \{n_1,n_4\}$

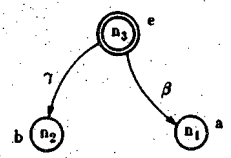
$C_Y(s_2) = \{n_1\}$ $T_Y(s_2) = \{n_2\}$

$P(s_2) = \{(U_1,V_2),(U_4,V_2)\}$

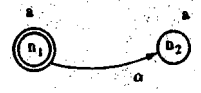
$Q(s_2) = \{(U_1,V_2)\}$

Fig. 5.5 Information Represented by States in the State-Space Representation

At state s_3:



Subgraph X_3



Subgraph Y_3

$C(s_3) = \{(n_3,n_1)\}$

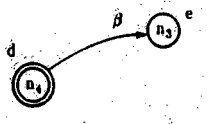
$C_X(s_3) = \{n_3\}$ $T_X(s_3) = \{n_1,n_2\}$

$C_Y(s_3) = \{n_1\}$ $T_Y(s_3) = \{n_2\}$

$P(s_3) = \{(U_1,V_2),(U_2,V_2)\}$

$Q(s_3) = \{(U_1,V_2)\}$

At state s_4:



Subgraph X_4

Subgraph Y_4

$C(s_4) = \{(n_4,n_1)\}$

$C_X(s_4) = \{n_4\}$ $T_X(s_4) = \{n_3\}$

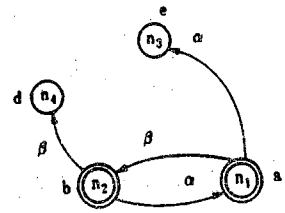
$C_Y(s_4) = \{n_1\}$ $T_Y(s_4) = \{n_2\}$

$P(s_4) = \{(U_3,V_2)\}$

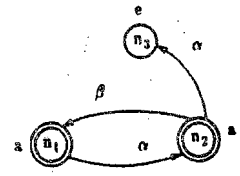
$Q(s_4) = \{\phi\}$

Fig. 5.5 (Continued)

At state s_5:



Subgraph X_5



Subgraph Y_5

$C(s_5) = \{(n_1,n_1),(n_2,n_2)\}$

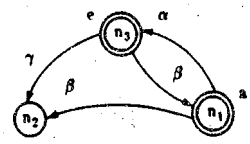
$C_X(s_5) = \{n_1,n_2\}$ $T_X(s_5) = \{n_3,n_4\}$

$C_Y(s_5) = \{n_1,n_2\}$ $T_Y(s_5) = \{n_3\}$

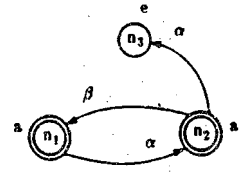
$P(s_5) = \{(U_3,V_3),(U_4,V_3)\}$

$Q(s_5) = \{(U_3,V_3)\}$

At state s_6:



Subgraph X_6



Subgraph Y_6

$C(s_6) = \{(n_1,n_1),(n_3,n_2)\}$

$C_X(s_6) = \{n_1,n_3\}$ $T_x(s_6) = \{n_2\}$

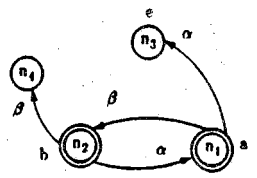
$C_Y(s_6) = \{n_1,n_2\}$ $T_Y(s_6) = \{n_3\}$

$P(s_6) = \{(U_2,V_3)\}$

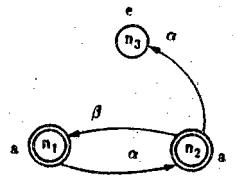
$Q(s_6) = \{\phi\}$

Fig. 5.5 (Continued)

At state s_7:



Subgraph X_7



Subgraph Y_7

$C(s_7) = \{(n_1,n_2),(n_2,n_1)\}$

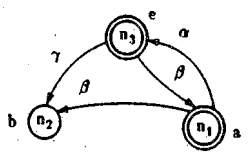
$C_X(s_7) = \{n_1,n_2\}$ $\quad T_X(s_7) = \{n_3,n_4\}$

$C_Y(s_7) = \{n_1,n_2\}$ $\quad T_Y(s_7) = \{n_3\}$

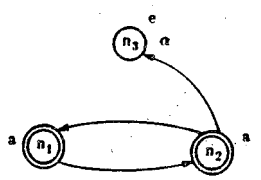
$P(s_7) = \{(U_3,V_3),(U_4,V_3)\}$

$Q(s_7) = \{(U_3,V_3)\}$

At state s_8:



Subgraph X_8



Subgraph Y_8

$C(s_8) = \{(n_1,n_2),(n_3,n_1)\}$

$C_X(s_8) = \{n_1,n_3\}$ $\quad T_X(s_8) = \{n_2\}$

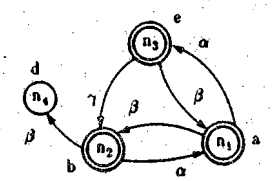
$C_Y(s_8) = \{n_1,n_2\}$ $\quad T_Y(s_8) = \{n_3\}$

$P(s_8) = \{(U_2,V_3)\}$

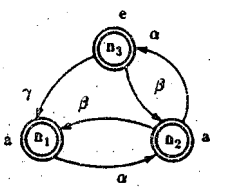
$Q(s_8) = \{\phi\}$

Fig. 5.5 (Continued)

At state s_9:



Subgraph X_9



Subgraph Y_9

$C(s_9) = \{(n_1,n_1),(n_2,n_2),(n_3,n_3)\}$

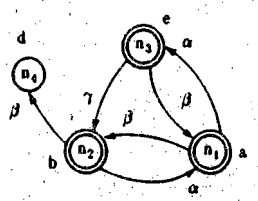
$C_X(s_9) = \{n_1,n_2,n_3\}$ $\quad T_X(s_9) = \{n_4\}$

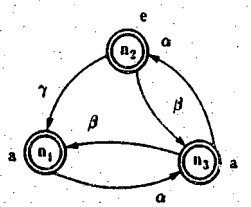
$C_Y(s_9) = \{n_1,n_2,n_3\}$ $\quad T_Y(s_9) = \{\phi\}$

$P(s_9) = \{(U_4,\Lambda)\}$

$Q(s_9) = \{(U_4,\Lambda)\}$

At state s_{10}:





$C(s_{10}) = \{(n_1,n_2),(n_2,n_1),(n_3,n_3)\}$

$C_X(s_{10}) = \{n_1,n_2,n_3\}$ $\quad T_X(s_{10}) = \{n_4\}$

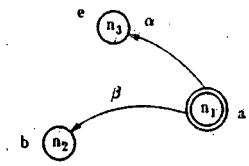
$C_Y(s_{10}) = \{n_1,n_2,n_3\}$ $\quad T_Y(s_{10}) = \{\phi\}$

$P(s_{10}) = \{(U_4,\Lambda)\}$

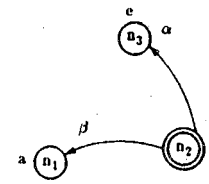
$Q(s_{10}) = \{(U_4,\Lambda)\}$

Fig. 5.5 (Continued)

At state s_{13}:



Subgraph X_{13}



Subgraph Y_{13}

$C(s_{13}) = \{(n_1,n_2)\}$

$C_X(s_{13}) = \{n_1\}$ $\quad T_X(s_{13}) = \{n_2,n_3\}$

$C_Y(s_{13}) = \{n_2\}$ $\quad T_Y(s_{13}) = \{n_1,n_3\}$

$P(s_{13}) = \{(U_2,V_1),(U_2,V_3),(U_3,V_1),(U_3,V_3)\}$

$Q(s_{13}) = \{(U_2,V_1),(U_2,V_3),(U_3,V_1),(U_3,V_3)\}$

Fig. 5.5 (Continued)

according to some heuristic criterion which we call "Feasibility Condition" that has to be satisfied; we will explain this criterion, discuss its justification and validity, and give its algorithm in Section 5.3.4.

At any state $s_I=(X_I,Y_I)$ in the lattice, let t_x and t_y denote two nodes in the terminal-node sets of the subgraphs X_I and Y_I respectively, i.e., $t_x \in T_X(s_I)$ and $t_y \in T_Y(s_I)$. Let X_{t_x} and Y_{t_y} be the two BARGs for which t_x and t_y are the root nodes, i.e., $X_{t_x} = (t_x, B^{X_{t_x}}, L^{X_{t_x}})$, and $Y_{t_y} = (t_y, B^{Y_{t_y}}, L^{Y_{t_y}})$. We formulate a set, $P(s_I)$, as the set of possible pairs of (X_{t_x}, Y_{t_y}), of the form:

$$P(s_I) = \left\{ \{(X_{t_x}, Y_{t_y})\} \cup \{(X_{t_x}, \Lambda)\} \cup \{(\Lambda, Y_{t_y})\} \mid t_x \in T_X(s_I), t_y \in T_Y(s_I) \right\}.$$

$P(s_I)$ denotes the set of candidate pairs of basic graphs to be matched, if they satisfy the feasibility criterion. We augment the set $P(s_I)$ to include $\{(X_{t_x}, \Lambda)\}$ and $\{(\Lambda, Y_{t_y})\}$ in order to encompass the fact that the two graphs U and V are not necessarily of the same size, and thus some nodes may be deleted.

Next, we check the feasibility of pairs $(X_{t_x}, Y_{t_y}) \in P(s_I)$ and let set $Q(s_I)$ be the set of only those pairs which satisfy the "Feasibility Condition", i.e.,

$$Q(s_I) = \left\{ (X_{t_x}, Y_{t_y}) \in P(s_I) \mid FEASIBLE((X_{t_x}, Y_{t_y}), C_X(s_I), C_Y(s_I), T_X(s_I), T_Y(s_I)) \right\}.$$

Then, we expand the state s_I by creating new states, e.g., state s_J, which corresponds to the embedding of a pair of $(X_{t_x}, Y_{t_y}) \in Q(s_I)$ into the already reconstructed subgraphs X_I and Y_I of graphs U and V, respectively. We also label the branch (s_I, s_J) to denote the incremental distance resulting from the embedding of (X_{t_x}, Y_{t_y}) into (X_I, Y_I) to create (X_J, Y_J). The formal algorithm is given below.

ALGORITHM 5.II :

Purpose : To generate the state-space representation for the distance calculation and inexact matching between two ARGs.

Input : Two ARGs U and V, each of which is decomposed into a set of BARGs given as $\hat{U}=\{U_1, U_2, \ldots, U_M\}$ and $\hat{V}=\{V_1, V_2, \ldots, V_N\}$.

Output : A Directed Labeled Lattice of states $S=\{s_0, s_1, \ldots, s_L\}$, where the states are labeled as $s_I=(X_I, Y_I)$, $X_I \subseteq U$, $Y_I \subseteq V$, and the branches are labeled as $w(s_I, s_J)$.

Method :

I. Let $s_0 \leftarrow (\Lambda, \Lambda)$

II. FOR i:=1 to M DO
    FOR j:=1 to N DO
    BEGIN

1. $s_I \leftarrow \{(X_I, Y_I) \mid X_I = U_i \in \widehat{U},\ Y_I = V_j \in \widehat{V}\}$.

2. $S \leftarrow S \cup \{s_I\}$.

3. $w(s_0, s_I) \leftarrow d_{(\Lambda,\Lambda)}(U_i, V_j)$.

4. $s_K \leftarrow s_I$

5. Let $C(s_K)$ denote $\left\{ (r^{X_k}, r^{Y_k}) \mid (X_k, Y_k) \in \{(X_K, Y_K)\} \right\}$,

   $C_X(s_K)$ denote $\left\{ r^{X_k} \mid (r^{X_k}, r^{Y_k}) \in C(s_K) \right\}$,

   $C_Y(s_K)$ denote $\left\{ r^{Y_k} \mid (r^{X_k}, r^{Y_k}) \in C(s_K) \right\}$,

   $T_X(s_K)$ denote $\left\{ L^{X_k} \mid X_k \in \{N^{X_K} - C_X(s_K)\} \right\}$,

   $T_Y(s_K)$ denote $\left\{ L^{Y_k} \mid Y_k \in \{N^{Y_K} - C_Y(s_K)\} \right\}$,

   $P(s_K) = \left\{ \{(X_{t_x}, Y_{t_y})\} \cup \{(X_{t_x}, \Lambda)\} \cup \{(\Lambda, Y_{t_y})\} \mid t_x \in T_X(s_K), t_y \in T_Y(s_K) \right\}$.

6. $Q(s_K) \leftarrow$

   $\left\{ (X_{t_x}, Y_{t_y}) \in P(s_K) \mid FEASIBLE((X_{t_x}, Y_{t_y}), C_X(s_K), C_Y(s_K), T_X(s_K), T_Y(s_K)) \right\}$.

7. For all $(X_l, Y_l) \in Q(s_K)$ DO

   (i) $\widehat{X}_J \leftarrow \widehat{X}_K \cup \{X_l\}$

   (ii) $\widehat{Y}_J \leftarrow \widehat{Y}_K \cup \{Y_l\}$

   (iii) $w(s_K, s_J) \leftarrow d_{(X_K, Y_K)}(X_l, Y_l)$

   (iv) IF NOT$((X_J, Y_J) \in S)$ THEN DO

   $S \leftarrow S \cup \{(X_J, Y_J)\}$

   $s_K \leftarrow s_J$

GO TO STEP (II.5)

8. END FOR

III. END ALGORITHM 5.II

5.3.4 The Feasibility Criterion

At any state (s_I) in the Lattice we view each of the reconstructed subgraphs $X_I \subseteq U$ and $Y_I \subseteq V$ as consisting of two non-overlapping sets of nodes, namely the core-node set and the terminal-node set, i.e., $C_X(s_I)$ and $T_X(s_I)$ for subgraph X_I, and $C_Y(s_I)$ and $T_Y(s_I)$ for subgraph Y_I. The nodes in the core-node sets are matched in pairs forming the set $C(s_I)$, where $c=(r^{X_i}, r^{Y_i}) \in C(s_I)$, as shown in Section 5.3.3.

We define a Feasibility Criterion based on the following idea:

At a state (s_I) along a certain path in the Lattice, we consider the core parts as matched pairs, and thus any new matched pairs to be added to the cores at this state must comply with this fact. In other words, the core parts are already matched in pairs and the approach proceeds to match the rest of the nodes by checking every pair of the terminal nodes, at this state, to determine whether they are a feasible next matched pair, i.e., whether considering them as a matched pair will require any changes to be made on the matching configuration of the core parts at this state. This is done by checking that their

respective neighbor nodes are matched in accordance with the matching configuration of the core parts at the state considered.

If a pair of terminal nodes does not satisfy this criterion, it is called an infeasible pair. If a candidate pair is infeasible, the expansion of the state by adding this pair of terminal nodes into the core parts is disregarded and the remaining candidate pairs are considered.

This criterion is presented below by Algorithm 5.III. In brief, the algorithm checks the leaf nodes of both X_k and Y_k. If a leaf node of X_k is in $C_X(s_I)$ then there has to be a leaf node in Y_k that is matched already with it according to the set $C(s_I)$, and vice versa. Otherwise the pair (X_k, Y_k) is considered infeasible.

ALGORITHM 5.III :

Purpose : To check the feasibility of embedding a pair of Basic graphs (X_k, Y_k) into the already reconstructed subgraphs, $X_I \subseteq U$ and $Y_I \subseteq V$, at state s_I.

Input :

1. The two reconstructed subgraphs of state s_I, i.e., (X_I, Y_I), which is represented by the sets $C(s_I)$, $C_X(s_I)$, $C_Y(s_I)$, $T_X(s_I)$ and $T_Y(s_I)$.

2. A pair of candidate BARGs (X_{t_x}, Y_{t_y}).

Output : True - if the embedding of (X_{t_x}, Y_{t_y}) into (X_I, Y_I) is feasible,
False - otherwise.

Method :

I. Let $L_X \leftarrow L^{X_{t_x}}$,
$L_Y \leftarrow L^{Y_{t_y}}$, and
FEASIBLE $\leftarrow$ TRUE .

II. WHILE L_X is NOT empty DO

1. $l_x \leftarrow$ an arbitrary node in L_X.

2. IF $l_x \in C_X(s_I)$ THEN
    IF *there exists* $l_y \in L_Y$ such that $(l_x, l_y) \in C(s_I)$ THEN
        $L_X \leftarrow L_X - \{l_x\}$,
        $L_Y \leftarrow L_Y - \{l_y\}$,
        GO TO Step II.3
    ELSE
        FEASIBLE $\leftarrow$ FALSE ,
        GO TO Step IV
  ELSE
    IF $l_x \in T_X(s_I)$ THEN
        IF *there exists* $l_y \in L_Y$ such that l_x matches l_y
        THEN
            $L_X \leftarrow L_X - \{l_x\}$,
            $L_Y \leftarrow L_Y - \{l_y\}$,

GO TO Step II.3

ELSE

FEASIBLE ← FALSE ,

GO TO Step IV

ELSE

$L_X \leftarrow L_X - \{l_x\}$,

GO TO Step II.3

3. END while

III. WHILE L_Y is NOT empty DO

1. $l_y \leftarrow$ an arbitrary node in L_Y .
2. IF $l_y \in C_Y(s_I)$ THEN

FEASIBLE ← FALSE ,

GO TO Step IV

ELSE

IF $l_y \in T_Y(s_I)$ THEN

FEASIBLE ← FALSE ,

GO TO Step IV

ELSE

$L_Y \leftarrow L_Y - \{l_y\}$,

GO TO Step III.3

3. END while

IV. END ALGORITHM 5.III.

5.3.5 The Optimization Problem

The problem now is to find the Shortest-Path over the state Lattice, which consists of a set S of states and a set R of directed branches, from the Initial State ($s_0 \in S$) to a Final State ($s_F \in S_F$), where S_F is a set of Final States. Formally, the Shortest-Path Problem over a Acyclic Directed Lattice has a solution by Dynamic Programming which has linear time complexity in the number of states in the lattice. The Dynamic Programming (DP) technique, [dena82], [dena79a], [dena79b], performs an intelligent search over all, but only, the feasible paths in order to find the shortest path. This technique divides the problem into stages at which decisions take place; it finds a recursive relation which moves backward from one stage to the previous stage, i.e., it decomposes the problem into a sequence of smaller problems which can be solved recursively one at a time.

Fortunately, the state lattice generated by our approach is already divided into stages from stage 0 up to stage H. The order of the stage represents the number of matched-pair nodes in the core parts of subgraphs X_I and Y_I. The DP technique starts from the last stage in the lattice and proceeds backward one stage at a time until it reaches the initial state of the lattice. The general subproblem for this problem is :

Suppose we are at some arbitrary state $s_I \in S$, what state should we go to next so that the path from state s_I to a final state will have the shortest possible length?

This state is chosen in the best possible manner; thus it is a state (say s_J) in a path from s_I to a final state, and it achieves the following:

$$D(s_I) = \min_{(s_I,s_J)\in\Omega} \left\{ w(s_I,s_J) + D(s_J) \right\}$$

where Ω is the set of all branches emanating from state s_I, such that (s_I,s_J) is on the path from s_I to a final state . The algorithm for this problem is given below.

ALGORITHM 5.IV :

Purpose : To find the Shortest-Path from an Initial State to a state in a set of Final States, over an Acyclic Lattice.

Input : Acyclic Directed Branch-weighted Lattice represented by :

1. $S=\{s_0,s_1,s_2, \ldots ,s_L\}$: a set of states , divided into H stages of the form $S=\{S^0 \cup S^1 \cup S^2 \cup \ldots S^H\}$, where $S^h \subseteq S$ is the subset consists of all the states at stage h; $S^0=\{s_0\}\in S$ is the Initial State; $S_F \subseteq S$ is a set of Final States.
2. $R\subseteq\{R_{IJ}=(s_I,s_J) \mid s_I \& s_J \in S\}$: a set of directed branches; for all $R_{IJ}\in R$, if $s_I\in S^h$ then $s_J\in S^{h+1}$ for $0\le h<H$.

Output :

* The Total Distance of the Shortest-Path from the Initial State to a Final State, denoted as $D(s_0)$.

* The Final State of the Shortest-Path, denoted as $F(s_0)$.

Method :

I. FOR I:= 0 to L DO

IF $s_I \in S_F$ THEN DO

$D(s_I) \leftarrow 0$,

$F(s_I) \leftarrow s_I$

ELSE DO $D(s_I) \leftarrow \infty$

II. FOR h:=$H-1$ to 0 DO

FOR all $s_I \in S^h$ DO

$D(s_I) \leftarrow \min_{(s_I,s_J)\in R} \{w(s_I,s_J)+D(s_J)\}$

$F(s_I) \leftarrow F(s_J)$

III. END ALGORITHM 5.IV.

5.4 Computational Complexity Analysis

It is a well known fact, unfortunately, that the graph-to-graph distance problem belongs to the class of NP-complete problems. This distance is defined as the cost of the minimum total-cost sequence of error-transformations which should be performed on one graph to produce the other graph. Any conventional search algorithm, e.g., backtracking, will require exponential time. Moreover, several intelligent search techniques, e.g. branch and bound, forward-checking, or look-ahead search, even though their average performance may not be that bad, but their worst case complexity will still be exponential. In this section we investigate the computational complexity of our approach, as presented by the algorithms in Section 5.3.

Suppose that our input are two ARGs, U and V of M and N nodes, respectively. Let δ, a constant, denote the maximum number of branches emanating from any node in U or V. The first part of the approach is to decompose both U and V into Basic ARGs. It is clear that this is done in linear time for each of the two graphs, i.e., has time complexity of order $(M+N)$.

The second part of the approach is to generate the multi-stage state-space lattice which describes the states of reconstructing the decomposed graphs. The initial state of the process composes stage 0 of the lattice. A state s_J in a stage J of the lattice, if the set of matched-pair of core nodes at this state consists of J elements. We expand the state s_J in stage J into a subset of states in stage J+1; each of them denotes the addition of a new pair from the set of terminal nodes at s_J into the core nodes. For the general case of calculating the distance between U and V, the worst case for the number of stages in the lattice is $M+N$. For some more specific cases, such as considering one graph as a reference graph and use the approach to determine the best matching subgraph of the other graph, the number of stages will be less, as we will discuss later in this section.

The subgraph isomorphism problem and its related versions have exponential worst-case complexity. But in practice most subgraph isomorphism algorithms behave far better than that for real applications. In [hara78] and [tsai83], it is assumed that a typical search tree will reduce to just a single line with few branches at each tree level and will contain total number of states of order αn^2, where n is the number of nodes in the graph and α is a constant taking care of the branches in the graph. In our approach we

consider a reasonable assumption, which is justified by the result of our implementation, that each stage of the state-space lattice contains number of states in the order of δMN, where δ, M, and N are as defined before (or δN^2 if $M=N$). Since the worst case for the number of stages is $M+N$, thus the total number of states in the lattice is of the order of $\delta MN(M+N)$.

Now we discuss, in detail, the complexity analysis of the second part of our approach, which generate the state-space lattice. The generation of stage 1, requires time $T_1=(c_1+c_2)M$, where c_1 is a constant for assigning the state label, and c_2 is a constant for calculating the incremental distance between two Basic ARGs, thus it depends on δ^2. At state s_I in stage j, since $|C(s_I)|=j$, thus the sets of core nodes of subgraphs X_I and Y_I contains at most j nodes each, i.e., $|C_X(s_I)|\leq j$ and $|C_Y(s_I)|\leq j$. Therefore, the upper-bound for the number of nodes in the terminal node sets at this state is $j\delta$ for each subgraph, i.e., $|T_X(s_I)|\leq j\delta$ and $|T_Y(s_I)|\leq j\delta$. A more accurate estimate for $|T_X(s_I)|$ and $|T_Y(s_I)|$ can be obtained if we assume that the branches in both U and V are evenly distributed over the nodes, therefore $|T_X(s_I)|\leq j\delta(1-\frac{j}{M})$ and $|T_Y(s_I)|\leq j\delta(1-\frac{j}{N})$. The quantities $j(1-\frac{j}{M})$ and $j(1-\frac{j}{N})$ have upper limits at $j=\frac{M}{2}$ and $j=\frac{N}{2}$, respectively. Therefore, $|T_X(s_I)|\leq\frac{M\delta}{2}$ and $|T_Y(s_I)|\leq\frac{N\delta}{2}$. Thus, the upper limit for the total number of validity check of condition FEASIBLE, at state s_I in the lattice is $\frac{\delta^2 MN}{4}$. If τ is the time required by the feasibility checking algorithm, then the upper limit of time for feasibility checking at this state is $\tau\frac{\delta^2 MN}{4}$.

It can be easily seen that, with proper implementation of the heuristic algorithm FEASIBLE, it needs time $\tau = c_3\delta^2$, where c_3 is a constant time needed to make a variable assignment. Therefore, the worst case time required for feasibility checking at state s_I in the lattice is $c_3\frac{\delta^4 MN}{4}$. Usually only a fraction of the checked pair of Basic ARGs will satisfy the feasibility criterion and will produce states in the next stage of the lattice, thus the time required for feasibility checking represent a dominating factor in the analysis of our approach as we will see. The total time needed, as a worst case estimate, at state s_I in the lattice is:

$$t_2 = c_3\frac{\delta^4 MN}{4} + \delta MN(c_1+c_2+c_4) = O(MN) ,$$

where c_4 is the constant time required to compare two states, all the other parameters are as defined before, and since δ is a constant. The maximum number of states in the lattice is $\delta MN(M+N)$, therefore the upper bound estimate of time needed to generate the lattice is given by:

$$T_2 = (c_1+c_2+c_3\frac{\delta^4}{4}+c_4)\delta M^2N^2(M+N) = O(M^2N^2(M+N)).$$

The third part of our approach is to find the shortest-path over the multi-stage state-space lattice from the designated initial state to a final state. As we discussed before with the presentation of our algorithm, this problem is solved by dynamic programming in linear time. Therefore, it is dominated by the complexity of the second part of the approach as investigated above.

In summary the upper-bound of the overall complexity of our approach for the general case of calculating the distance between any two ARGs U and V is of the order $\delta^5M^2N^2(M+N)$, where M and N

are the number of nodes in U and V, respectively, and δ is a constant representing the maximum number of branches stemming from any node in U or V. In the remaining part of this section we consider some specific version of this problem and investigate how their complexity differ from the general case.

One special version of the problem is to consider V as a prototype ARG of N nodes and U to be deformed version of V. The objective is to find the minimum total cost sequence of error-transformations which must be performed on U to produce V. In this case the maximum number of states in any stage in the lattice will be of order δN^2, the number of stages is N, with all the final states are located in stage N. Therefore, the total number of states in the lattice is of order δN^3, and following similar steps as show above, the upper bound of the overall computational complexity of the approach in this case will be $\delta^5 N^5$ (i.e., $O(N^5)$).

Another special version, we assume that there is a node of U and a node in V which are used as a registration matched-pair, as was assumed by Sanfeliu, [sanf83a]. In this case, this matched-pair will denote the initial state in the lattice and thus the upper-bound on the total number of states in the lattice will be of order N^2 (or $N(M+N)$ for the general case), thus the overall complexity in this case reduces to $\delta^4 N^4$ (i.e., $O(N^4)$).

CHAPTER VI

APPLICATION TO IMAGE UNDERSTANDING AND EXPERIMENTAL RESULTS

6.1 General

The hierarchical scheme presented in Chapter 4 of this thesis, is a powerful scheme for the extraction of high-level (or global) image representation from low-level (or local) input images. It combines both the model-driven and the data-driven concepts. The model-driven principle is basically the top-down decomposition of object models, that are used to configure the graph transducer utilized in the scheme, e.g., to decide the choice of the alphabets, the neighborhood configurations, and the adjacency relations between the primitives. Thus, the model information is represented by the hierarchical graph transformation of the scheme.

On the other hand, the data-driven concept is basically a bottom-up process which is performed in that scheme by the mapping of the the input local alphabets of the bottom layers into the more global alphabets of the upper layers. The scheme is shown to be powerful and very useful for a wide variety of machine

intelligence applications, in which a global knowledge representation is required to facilitate the further processing. As we discussed in Chapter 1, the extraction of an adequate representation is the main objective of the first phase of most knowledge processing systems, to which image understanding systems belong.

Further processing of the information takes place in the second phase of the system. Such processing is usually performed on the extracted representation. In a wide variety of applications, the concept of defining some distance or similarity measures has been shown to be very useful in several decision making processes, in fact it is a very important concept in the fields of decision theory, function analysis, inference procedures, learning, etc.

Some of the very useful forms of knowledge representation utilize relational structures to represent sets of interrelated concepts, evidences, or objects. In Chapter 5, we have presented a new efficient approach for computing a distance measure and finding the best inexact matching configuration between general purpose relational structures of the form of attributed relational graphs. Attributed relational graphs are shown to be a powerful representation tool which combines both the syntactic and the semantic information into sets of attributed nodes and branches.

In this chapter, we discuss the application of the concepts and techniques presented in Chapters 4 and 5 in an image understanding system. The experimental work reported in this chapter consists of two major application experiments. Experiment I concerns with locating objects in a scene composed of complex overlapped objects, while Experiment II deals with target detection in highly noisy and

distorted images, specifically, we apply our techniques on Synthetic Aperture Radar (SAR) images. In the remaining parts of this chapter, we briefly comment on the concepts of our techniques and explain in details how they apply to each of the two experiments.

Figure 6.1 shows the general block diagram of the system which we use in our application experiments. The input data to this system is in the form of raw images defined as arrays of elements (or pixels) which take values over a gray scale. In Section 6.2, we comment briefly on the input image data and the preprocessing performed by the system. Due to some particular characteristics of SAR images, we devote Section 6.2.1 for discussing the nature of SAR images and the preprocessing performed in Experiment II, which deals with the SAR images. The utilization of the multi-layer hierarchical scheme, of Chapter 4, for the extraction of image global representation is discussed in Section 6.3. In the same section, we also present the hierarchical graph transformations which are used in our experiments.

The extraction of an attributed relational graph representation from the output field of the hierarchical scheme is discussed in Section 6.4. In Section 6.5, we comment on the application of the graph distance and inexact matching approach which is proposed in Chapter 5, into our experiments. This is basically done by measuring the distance, or similarity, between the attributed relational graph representations of the input image and of some object models; and extracting the matching subgraphs of minimum distance, or maximum similarity, between the two attributed

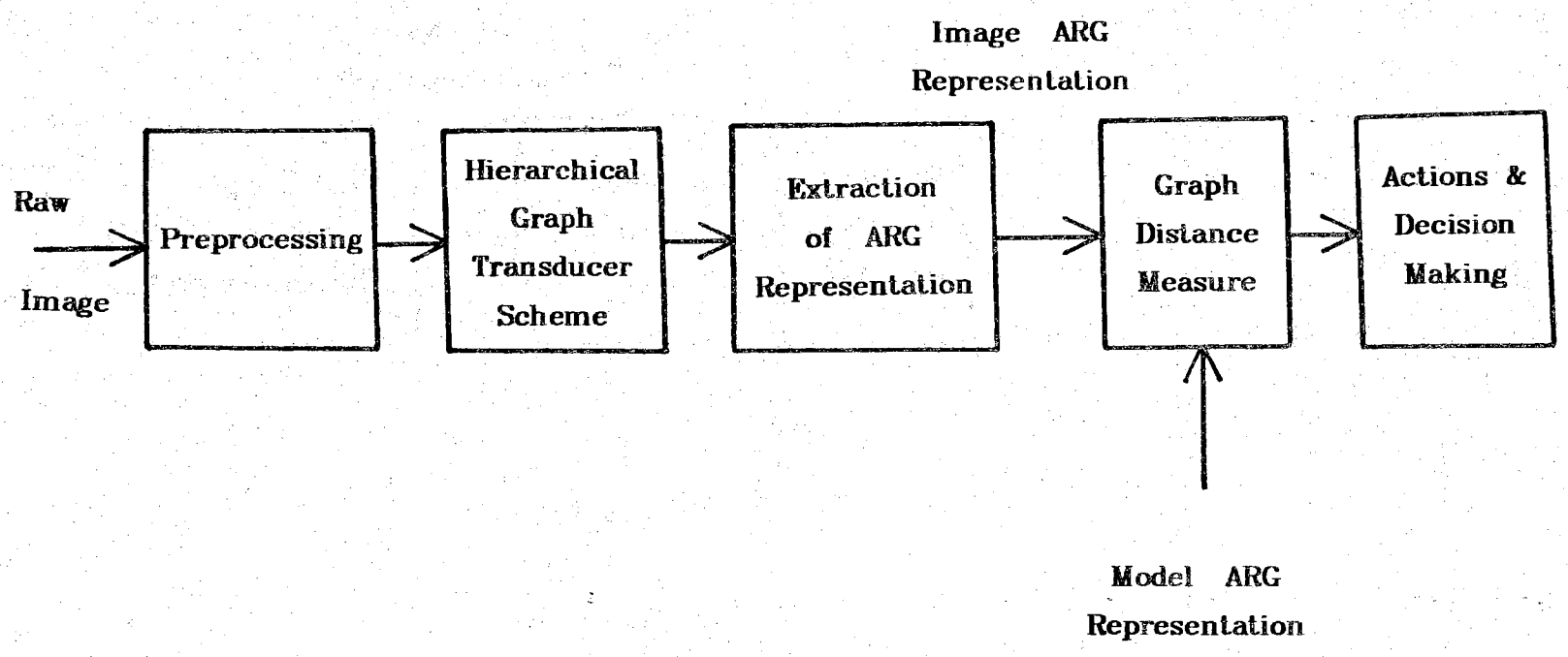


Fig. 6.1 Block Diagram of the Proposed Image Understanding System

relational graphs. In Section 6.6 and 6.7, we present the results of Experiment I, for locating objects in multi-object scene, and Experiment II, for target detection in SAR images, respectively.

6.2 Input Data and Preprocessing

6.2.1 General

Imaging peripherals usually deliver images on the form of two-dimensional digitized arrays for the signal falling on their receivers. This form of raw representation is called the spatial domain representation of images. A typical raw image consists of a 256×256 array of cells (or pixels), that take values over a 0-255 gray scale. The value of each cell represent the darkness of a very local element area in the image. Figures 6.2, 6.3, and 6.4 show examples of typical images which are used in our experiments. Images, as well as most other information sources, are usually burdened with noise, distortion, and uncertainty. Moreover, the data entities in the spatial domain image representation possess a very strict local nature. This is usually the case in most knowledge processing systems, where the input information is usually on the form of local entities, while the analysis and decision must be made based on some global meaningful configuration of these local entities.

The construction of some meaningful global representation from the input imagery data is presented below in Section 6.3. Nevertheless, some very simple preprocessing is needed to prepare the input raw images for the hierarchical scheme used in Section

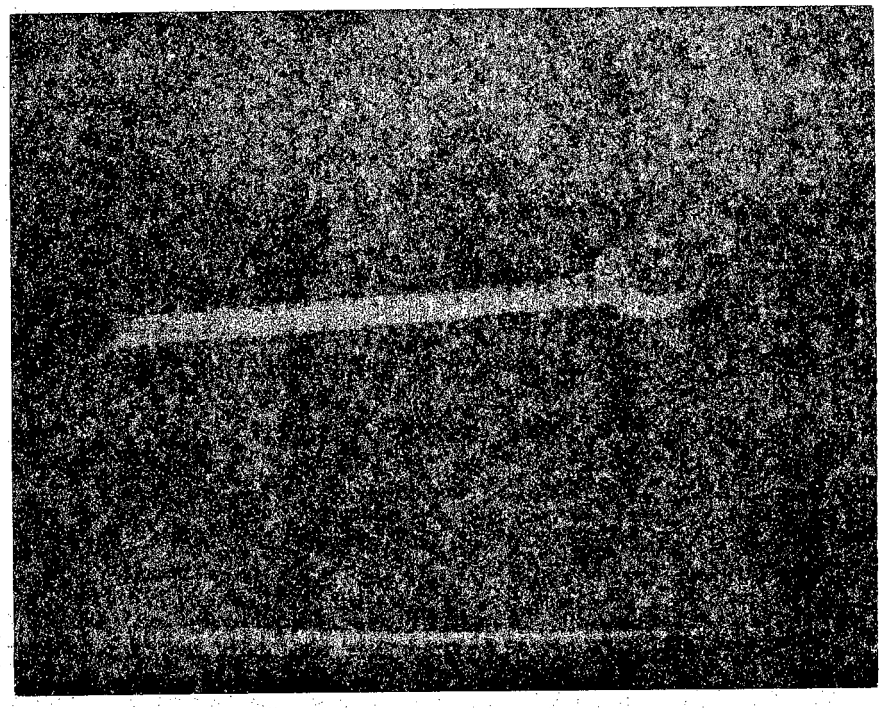


Fig. 6.2 An Image of a Model Object

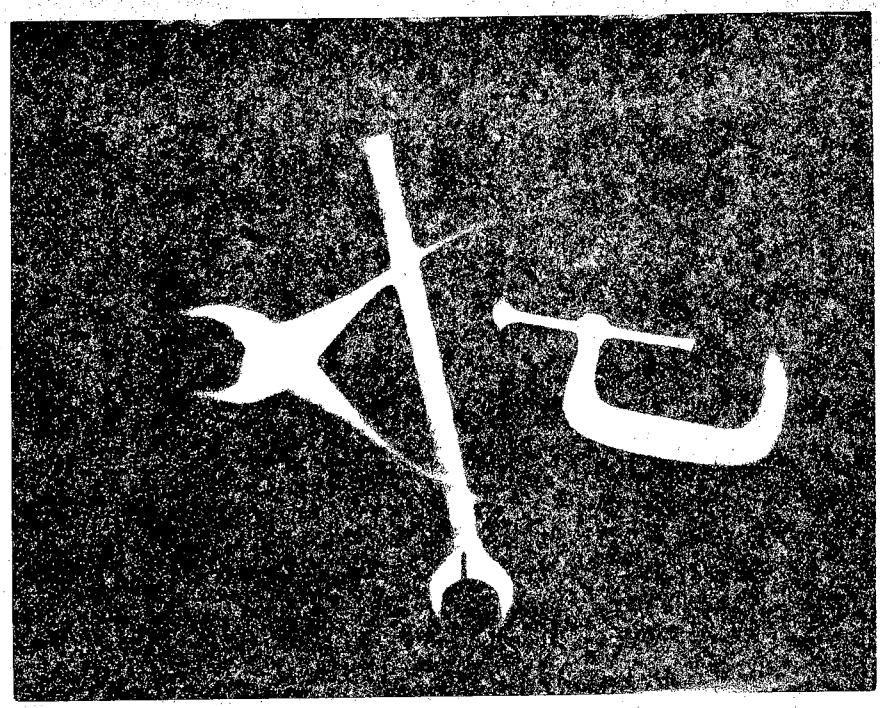


Fig. 6.3 An Image of Overlapping Objects

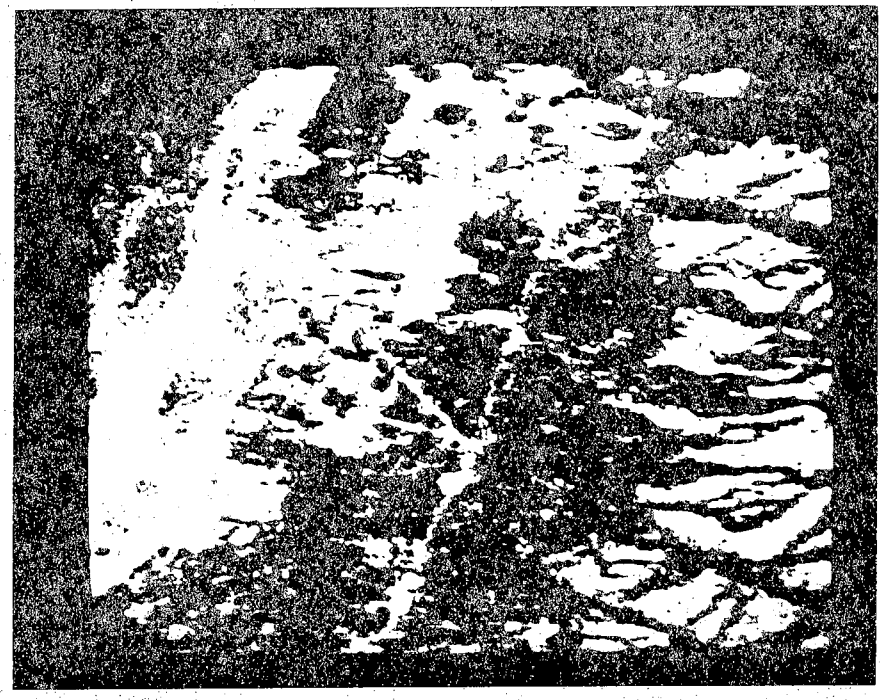


Fig. 6.4 The SAR Gray Scale Image of Santa Barbara Area

6.3. In Section 6.2.2, we present the preprocessing techniques which are utilized in Experiment I. In Section 6.2.3, we discuss the SAR image data and the preprocessing techniques which we propose to handle the noisy nature of SAR images which are used in Experiment II.

6.2.2 Preprocessing of Multi-Object Scenes

In Experiment I, we use very simple preprocessing on the form of a conventional edge operator, namely, the Sobol edge-operator over 3×3 windows. This operator is available on hardware chips and is relatively fast. Figures 6.5 and 6.6 show the result of applying that operator to the images of Figures 6.2 and 6.3, respectively. The edge image is then thresholded and thinned using some simple standard techniques, to extract the contours of the different regions in the images. The results of these operations on the images of Figures 6.5 and 6.6 are shown in Figures 6.7 and 6.8, respectively. The thinned image is now given as input to the hierarchical scheme, which is presented in Chapter 4, as shown in Section 6.3.

6.2.3 Preprocessing of SAR Images

The processing of aerial images, in general, and the Synthetic Aperture Radar (SAR) images in particular, is a challenging task. This is not only due to their low signal-to-noise ratio, but also due to the vast diversion of the shape, the relative size, the nature, and

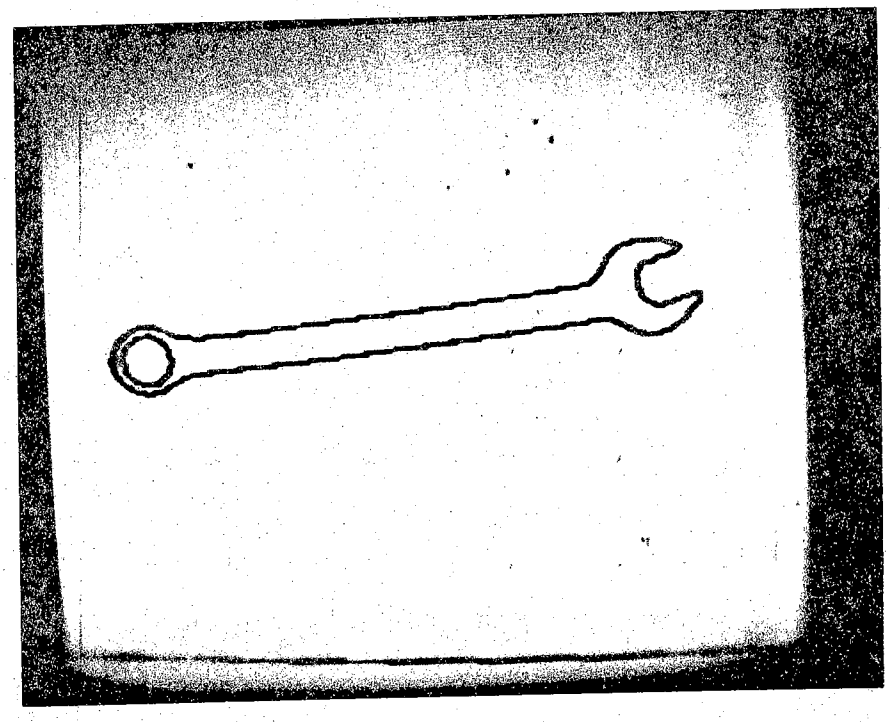


Fig. 6.5 The Result of Applying Sobol Edge-Operator to the Image in Fig. 6.2

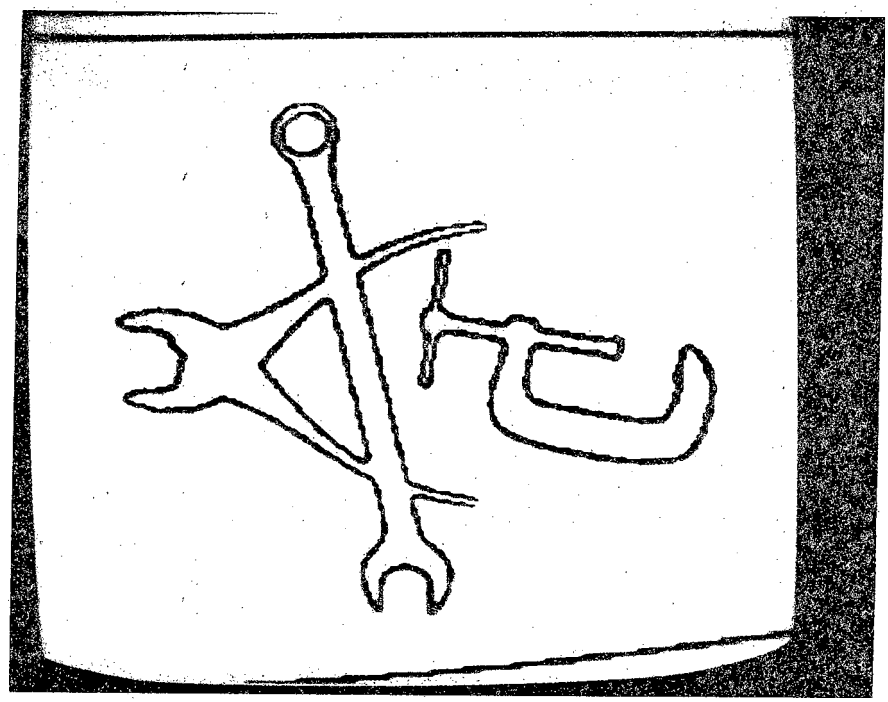


Fig. 6.6 The Result of Applying Sobol Edge-Operator to the Image in Fig. 6.3

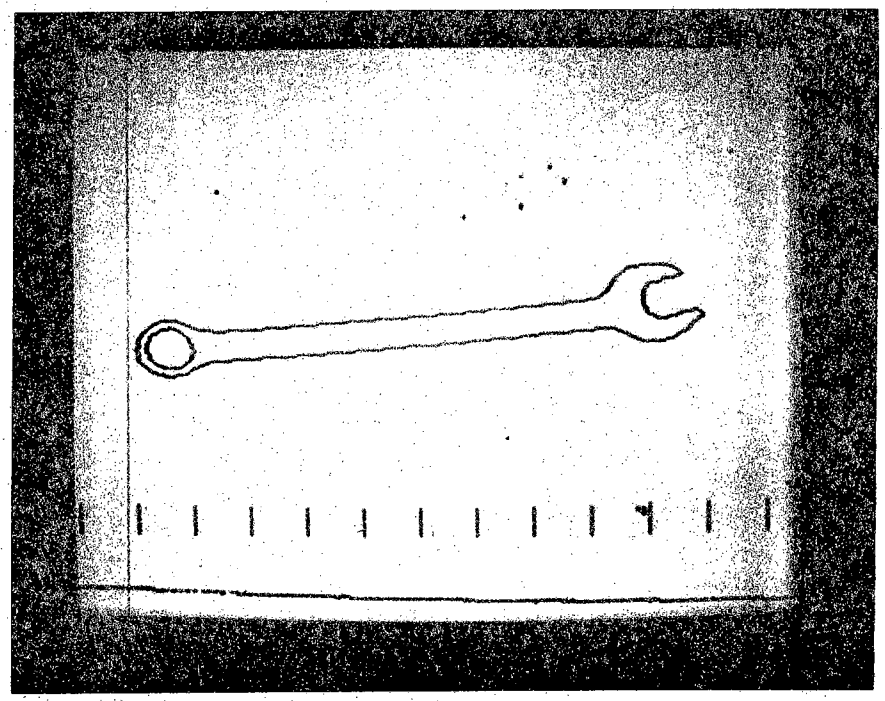


Fig. 6.7 Thinned Edges of the Image in Fig. 6.5

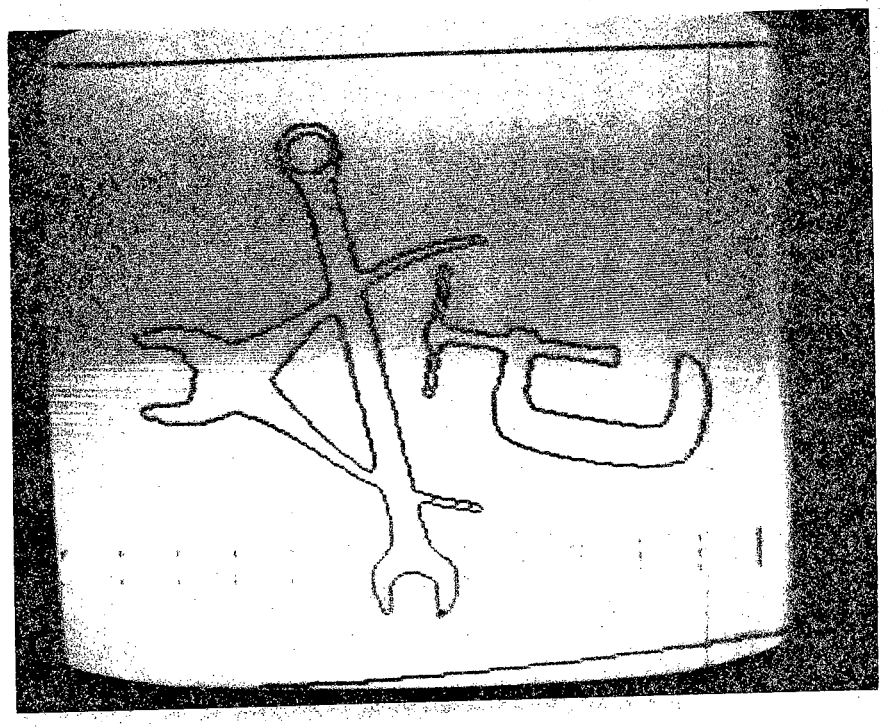


Fig. 6.8 Thinned Edges of the Image in Fig. 6.6

the informative features of the interesting objects in these images, [eshe83], [leej81]. Therefore, a relatively sophisticated preprocessing technique is needed for the analysis of this type of images.

The multi-resolution concept seems very appealing in the processing of aerial images, since one of the major characteristics of these images is the vast diversity of the objects that may exist in them. Some different objects may have the same shape but different sizes or different relative dimensions. A clear example of such a case is the difference between highways and airport runways, where the size and the ratio of width-to-length become major discriminative features. In this experiment, Experiment II, we use a multi-resolution technique in performing preliminary region-based segmentation using split-and-merge approach.

In order to efficiently perform the preliminary segmentation on the SAR images of Figure 6.4, we utilize the multi-resolution technique in a split-and-merge region-based segmentation and combine it with edge-based segmentation. This is basically the form of preprocessing which we utilize in this experiment, as we explain in more details in the remaining part of this section.

We assume that only the approximate size of objects in terms of the area represented by each pixel is known. Starting from the image with a suitable low resolution, we use a simple thresholding, e.g., thresholding over the gray scale or the variance of the subimages, to obtain a low segmentation of the image. The results of this rough segmentation is then used for further segmentation at ne 2 finer levels of resolution. The formal algorithm for this technique is given below.

ALGORITHM 6.I : Multi-Resolution Region-Based Segmentation

Purpose : To perform split-and-merge multi-resolution region-based preliminary segmentation

Input : A gray scale image

Output : Segmented image containing the candidate objects

Method :

I. Obtain a suitable rough resolution image based on the approximate size of the interesting objects.

II. Obtain a segmented image I_1 using some simple criterion, e.g., simple thresholding of the gray scale values.

III. Obtain a finer resolution image.

IV. Obtain a segmented image I_2.

V. Mask I_2 by I_1 in I_3.

VI. From I_2, pick up the pixels which are adjacent to object pixels in I_3 and move them into I_3.

VII. Repeat Step VI, until no more pixels can be added to I_3.

VIII. END ALGORITHM 6.I

The objective of this experiment is to detect some targets of interest in the SAR images. Specifically, in this experiment we aim at detecting airports in the SAR image of the Santa Barbara area.

In general, airports are characterized, from their region-based features as relatively large and uniform flat regions which exhibit high reflectance. From their shape, i.e., edge-based features, they are characterized as consisting of some runways, usually two or more, that take the form of relatively long but not too narrow flat areas.

The results of applying our technique and Algorithm 6.I to the Santa Barbara SAR image of Figure 6.4 are shown in Figures 6.9 through 6.12. The resulting image contains the candidate objects based on the region preliminary segmentation. Then the system focuses the attention on some mask areas around the candidate objects, and concentrates the edge-based segmentation within these areas. Figure 6.13 shows the extracted edges within the focus of attention areas, using the Sobol local edge-operator. It can be easily seen from that figure that the extracted edges of the candidate targets are very noisy and distorted. Nevertheless, we utilize this preprocessing results to perform higher level analysis of the SAR images, where we extract an attributed relational graph representation from the image and perform the inexact matching of minimum global distance between the extracted ARG representation of the image and that ARG representation of the target model as shown in the block diagram in Figure 6.1 and explained in the following sections.

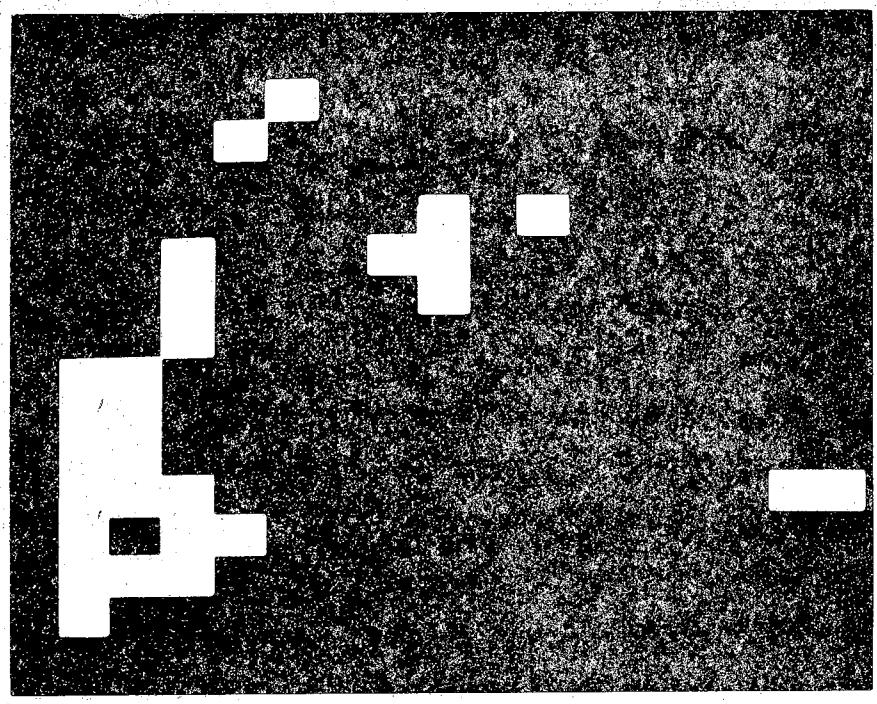


Fig. 6.9 Low Resolution Image of the Santa Barbara SAR Image

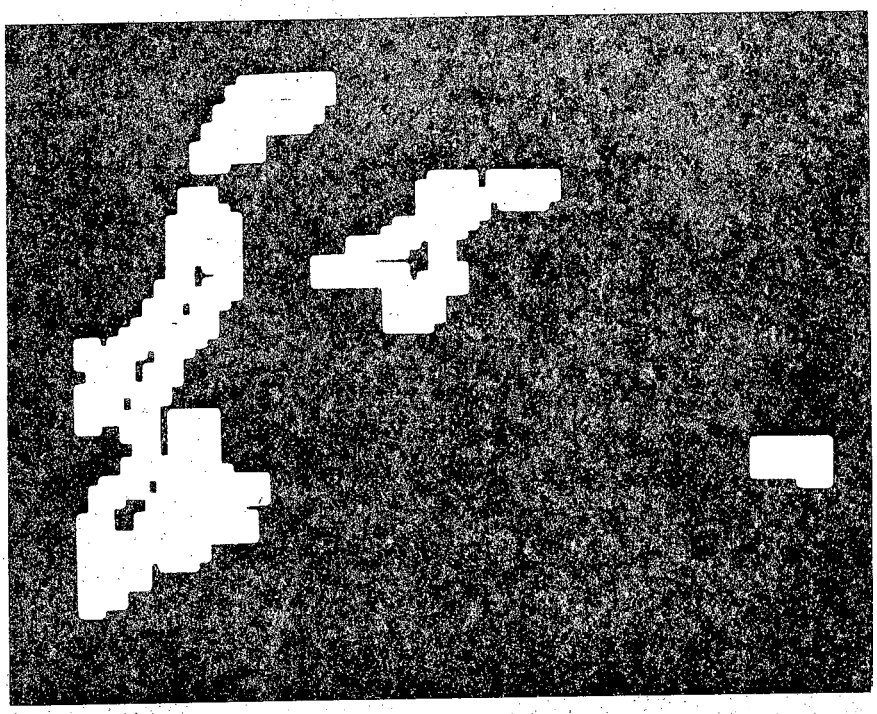


Fig. 6.10 Results of Split-and-Merge for the Image of Fig. 6.9

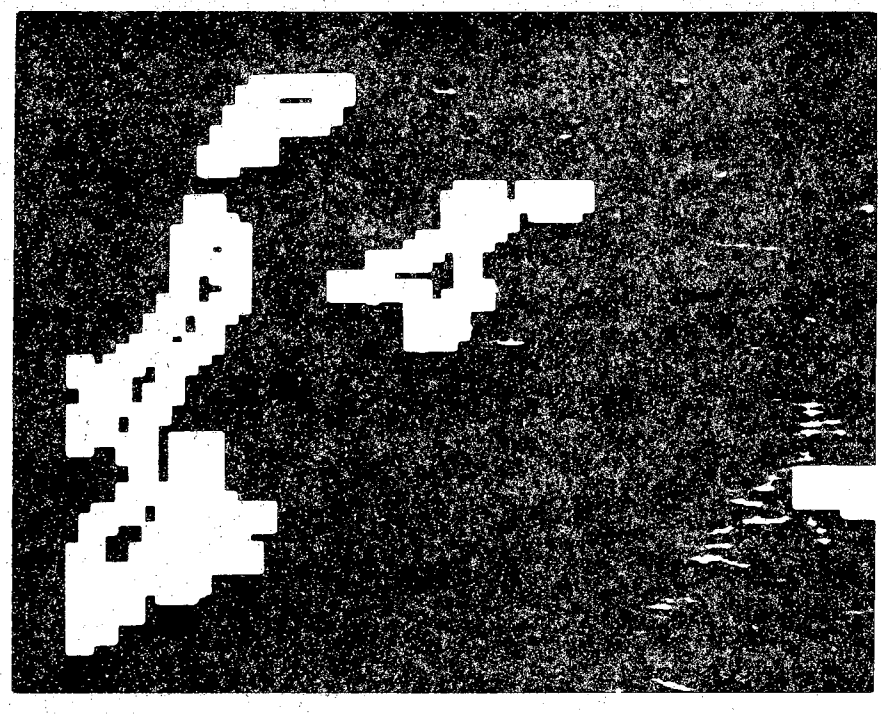


Fig. 6.11 Mask Areas Around Candidate Targets in the Santa Barbara SAR Image

Fig. 6.12 Edge-based Segmentation for Image in Fig. 6.11

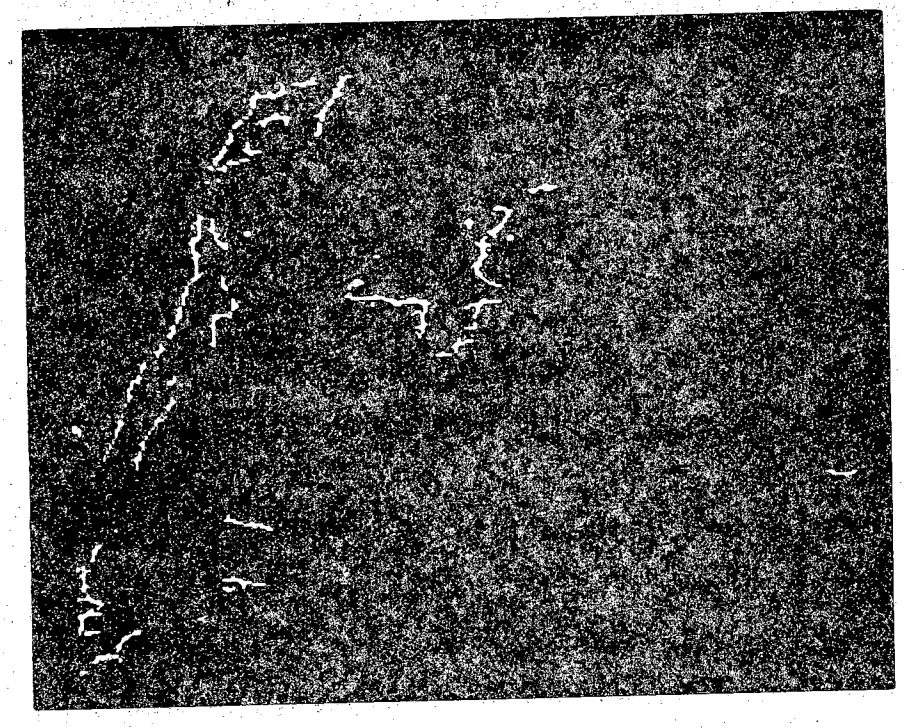


Fig. 6.13 Extracted Edges within the Focus of Attention Areas for the Santa Barbara SAR Image

6.3 The Extraction of Image Global Representation

6.3.1 General

One of the major difficulties in computer vision and image understanding, as well as most knowledge processing, systems is due to the very strict locality of the input image entities, or the input knowledge in general, which carry very little useful information, if taken individually. To achieve meaningful image understanding tasks, the system needs to comprehend the input information in a global form, similarly in other knowledge processing systems, complex relations among the input information entities, (or evidence) are usually required by the decision (or conclusion) making stage in the system.

In this section, we investigate the utilization of the multi-layer hierarchical scheme of Chapter 4 for the extraction of a global attributed symbolic representation from the input images. The input to the scheme is on the form of two-dimensional array of cells which take values over a very local input alphabet, as will be shown in Section 6.3.2. A transformation mapping is driven by the input data and performed over the cells of the scheme field to map these cells from the input local alphabet into a global alphabet.

The output alphabet of the hierarchical transformation at different layers in the scheme is designed according to the decomposition of the candidate complex objects into relatively simpler features (or image primitives). The image alphabets of this experiment are discussed in Section 6.3.3. The neighborhood

configuration among cells of the scheme field, the adjacency predicate between output symbols, and the mapping function of the hierarchical graph transducer are presented in Section 6.3.4. The extraction of ARG representation from the field of the scheme is discussed in Section 6.4 and the results of Experiments I and II are presented in Sections 6.6 and 6.7, respectively.

6.3.2 The Input Alphabet

Images are presented to the multi-layer scheme in the form of two-dimensional arrays of cells which are assigned values from a set of local symbols, which are measured from real scenes by some other available means. The set of input symbols is usually taken as the input alphabet of the hierarchical graph transducer. In both Experiment I and Experiment II, the input to this scheme is in the form of thinned edge image, therefore, we choose the input alphabet to be the simple binary set of black and white image pixels, as was shown in Chapter 4, Figure 4.3-(a).

6.3.3 The Global Output Alphabet

Usually in the structural approach to image analysis and understanding, objects in the images are decomposed into sets of sub-objects, (or features). In turn, complex features are decomposed into sets of simpler features, (or primitives), in a

recursive manner. Therefore, complex objects are decomposed into sets of primitives, which are easily measured by some conventional vision peripherals, or obtained by some simple preprocessing operations from the raw images.

As we discussed in Section 6.3.1, we extract thinned image edges from the raw images through some preprocessing. The output of the preprocessing operations takes the form of two-dimensional binary array of local image pixels. In this section, we present the design of a hierarchical feature alphabet which is used by the multi-layer scheme for the extraction of global image representation. In both experiments, we select to represent complex objects by their contours.

In this case, elements of the output alphabets of the hierarchical graph transformation are defined as line segments. Symbols of the alphabet at low levels in the scheme represent short lines, since the field cells at those levels cover relatively small areas in the image. On the other hand, symbols of the alphabet at higher levels represent relatively longer line segments, since cells of the scheme field at higher levels cover larger areas in the image.

In general, the hierarchical output alphabet is on the form: $\Sigma_O = \{\cup\Sigma_O{}^i \mid 1 \le i \le L\}$, where L is the number of layers in the scheme, and $\Sigma_O{}^i$ is the alphabet at the $i'th$ layer of the scheme. Each element of the alphabet at the $i'th$ layer, $\Sigma_O{}^i$, is composed of a group of elements of the alphabet at the $(i-1)'th$ layer, $\Sigma_O{}^{i-1}$, such that an element of that group occupy a cell in a certain

neighborhood configuration. The neighborhood configurations are defined by the neighborhood predicate R, as will be shown in Section 6.3.4.

In our work here, the output alphabet of the first layer of the transformation, $\Sigma_O{}^1$, is taken to represent the digitized line segments over 3×3-*pixel* windows, as shown in Figure 6.14. While, elements of the alphabet of the second layer, $\Sigma_O{}^2$, are composed of those elements of $\Sigma_O{}^1$ which are laying in a group of neighboring cells as defined by the predicate R of the transducer. Elements of $\Sigma_O{}^2$ represent arbitrary line segments over 7×7-*pixel* $(2^{2+1}-1\times2^{2+1}-1)$ windows, as shown in Figure 6.15. Each 7×7-*pixel* window is formed from a central 3×3-*pixel* window and its contour-surrounding 3×3-*pixel* windows, as shown in that figure. That configuration provides overlapping among low level elements that form the same higher hierarchy element.

Similarly, elements of $\Sigma_O{}^3$ are composed of those elements of $\Sigma_O{}^2$ that are laying in a group of neighboring cells and forming longer lines, that are defined over larger windows, namely 15×15-*pixel* $(2^{3+1}-1\times2^{3+1}-1)$ windows. In general, elements of the alphabet of the *i'th* layer, $\Sigma_O{}^i$, are composed of elements of the alphabet of the $(i-1)$*'th* layer, $\Sigma_O{}^{i-1}$, which lie in cells of the same neighborhood. A symbol $\alpha_i \in \Sigma_O{}^i$ represents, in this case, a line segment that passes through the center cell of a $(2^{i+1}-1\times2^{i+1}-1)$-*pixel* window.

The digitization noise is handled by the proposed approach through specification of the function Ω which defines pairs of alphabet symbols for every pair of adjacent cells in the neighborhood configuration. Moreover, the proposed hierarchical

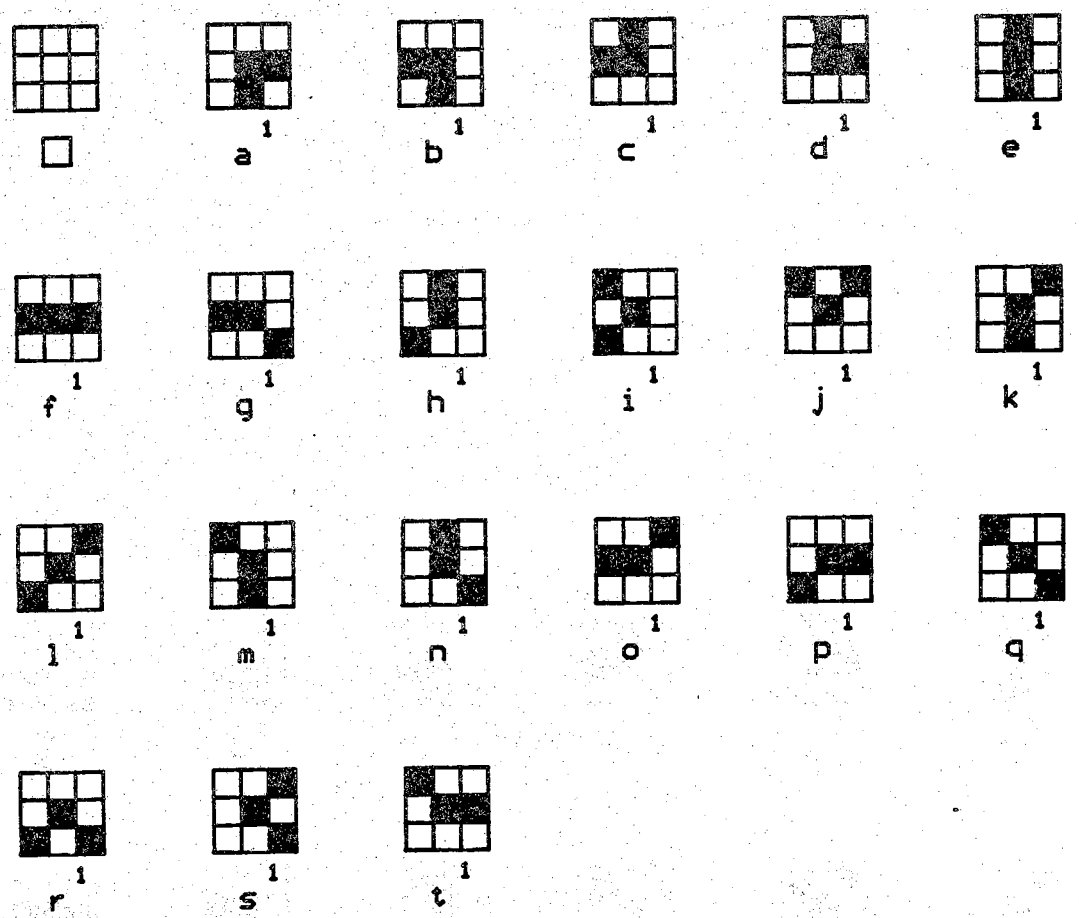


Fig. 6.14 Image Features of $\Sigma_0^1 \subset \Sigma_0$

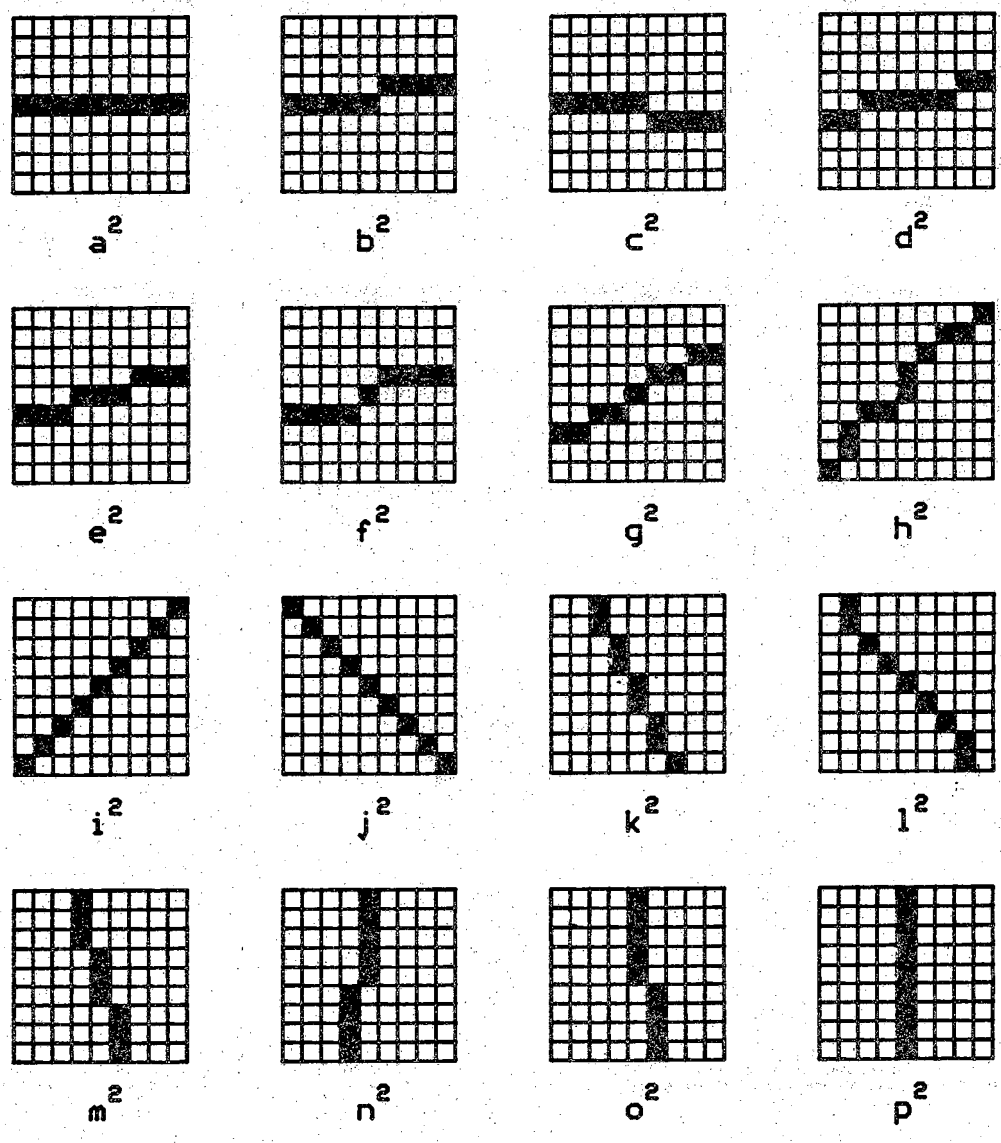
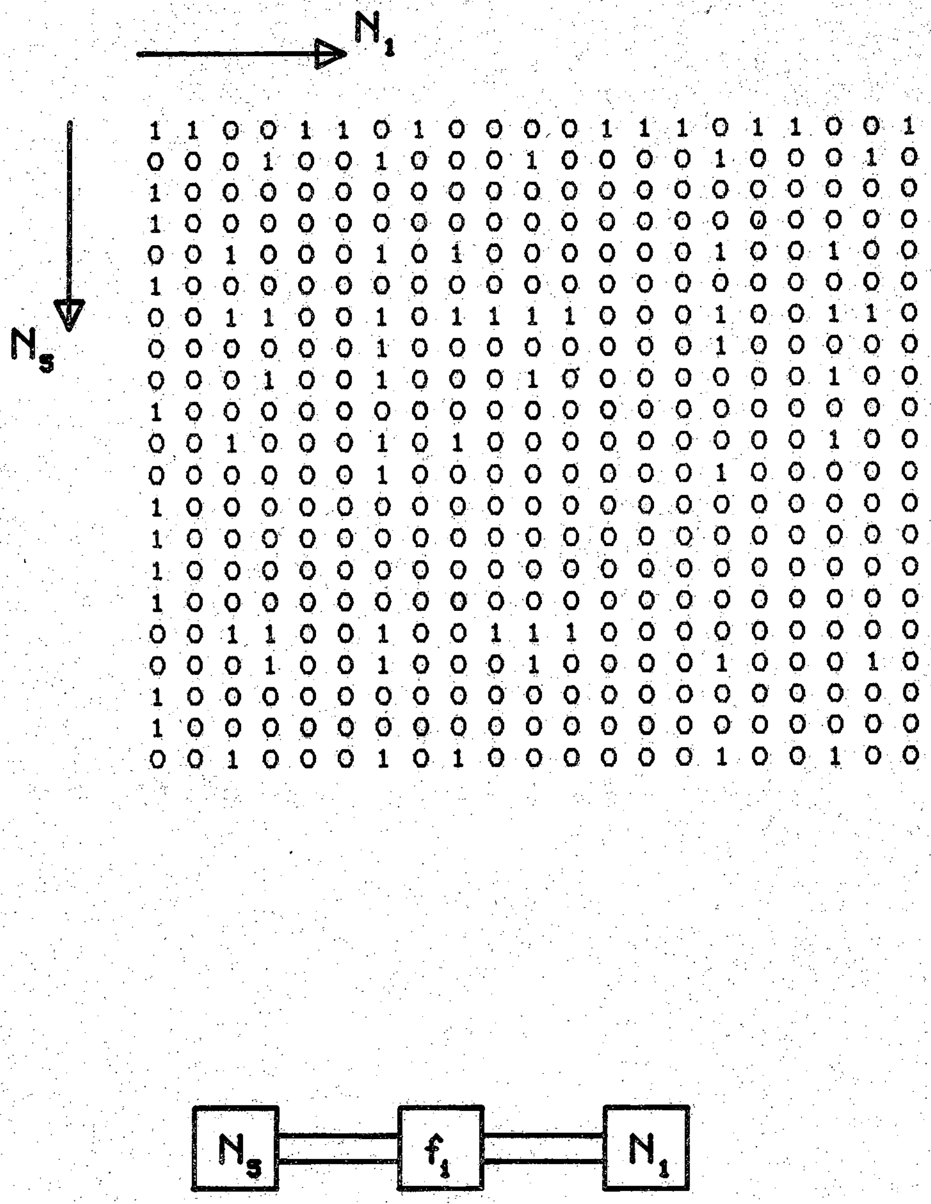


Fig. 6.15 Image Features of $\Sigma_0^2 \subseteq \Sigma_0$

graph transducer can handle other noise and distortion by modifying the transformation mapping, as we will show later in this chapter how the scheme can handle broken line segments in the images. The detailed definition of Ω for both experiments is given in the next section.

6.3.4 The Adjacency Predicate and Mapping Function

The adjacency predicate R is used to define groups of neighboring cells at different layers of the scheme. In these experiments, at the first layer we use the eight-nearest neighbors of each center cell to form nine cells neighborhood configurations. Each of those eight cells is said to be "adjacent" to the center cell and vice versa. The mapping function Ω at this layer defines pairs of possible adjacent output symbols for every pair of adjacent cells, while the function Φ defines the correspondence between input and output symbols. Both Ω and Φ, for these experiments, are defined in Figure 6.16. At this layer, the graph transformation maps the image from the input alphabet, which is the binary set as defined over single pixels, into elements of $\Sigma_O{}^1$, which is shown in Figure 6.14, and defined over 3×3-*pixel* windows. Each cell in this layer of the scheme represents a possible line segment which passes through this cell and lies in its surrounding 3×3-*pixel* window. Thus, the transformation of the first layer maps the scope of the field cells from single pixel into 3×3-*pixel* window.



$$
\begin{array}{ccccccccccccccccccccc}
1&1&0&0&1&1&0&1&0&0&0&0&1&1&1&0&1&1&0&0&1\\
0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&1&0&1&1&1&1&0&0&0&1&0&0&1&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&1&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&1&0&0
\end{array}
$$

(a) The Function Ω for N_1 and N_5 Neighbors

Fig. 6.16 The Mapping Functions Ω and Φ

$$\begin{array}{c|c} & \xrightarrow{\qquad\qquad} N_2 \\ \hline \downarrow N_6 & \begin{matrix}
1&0&0&1&1&0&1&0&0&0&1&0&1&0&1&1&1&0&0&1&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&1&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0\\
0&1&1&0&0&1&0&0&1&1&0&1&0&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&1&0&1&1&1&0&1&0&1&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&1&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0
\end{matrix} \end{array}$$

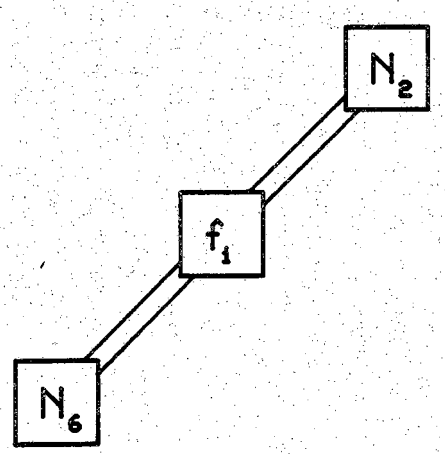


Fig. 6.16 (Continued)

N_3

N_7

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
1	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

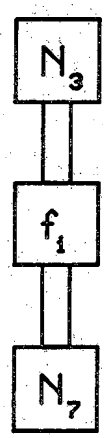
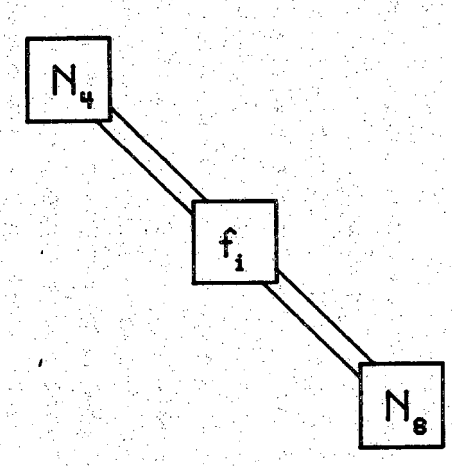


Fig. 6.16 (Continued)

N_4

N_8

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0
1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0



(d) The Function Ω for N_4 and N_8 Neighbors

Fig. 6.16 (Continued)

1	0
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1

(e) The Mapping Function Φ between Σ_I and Σ_O

Fig. 6.16 (Continued)

In the second layer, the adjacency predicate R defines the 16 cells surrounding every 3×3 window of cells as the neighbors of the center cell of that window, forming a neighborhood configuration which has scope of 7×7-*pixel* window, as shown in Figure 6.17. Each of those cells is said to be "adjacent" to the center cell and vice versa. The mapping function Ω at this layer defines pairs of possible adjacent symbols, from Σ_O^2, for every pair of adjacent cells. The function Φ defines the correspondence between symbols of Σ_O^1 and Σ_O^2 for the center cells of every neighborhood configuration. The transformation at this layer maps the image features from Σ_O^1, whose elements are defined over 3×3 windows of image pixels, into Σ_O^2, whose elements are defined over 7×7-*pixel* windows.

In general at the *i'th* layer, the transformation performs mapping of the scope of the field cells from $(2^i-1)\times(2^i-1)$ windows of image pixels into $(2^{i+1}-1)\times(2^{i+1}-1)$ windows of image pixels. The adjacency predicate R within the *i'th* layer defines the 4×2^i cells surrounding every $(2^i-1)\times(2^i-1)$ window as the neighbors of the center cell of that window, where each of these cells is called "adjacent" to the center cell of the window. The transformation at this layer maps the image features from elements of Σ_O^{i-1}, which are defined over $(2^i-1)\times(2^i-1)$ window of pixels into more global symbols of Σ_O^i, which are defined on larger windows of $(2^{i+1}-1)\times(2^{i+1}-1)$ image pixels.

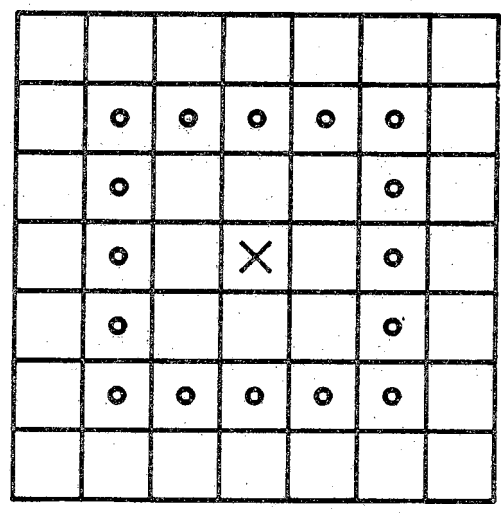


Fig. 6.17 A Neighborhood Configuration of the Second Layer in the Hierarchical Scheme

× : The Center Cell

o : A Neighboring Cell

6.4 The Extraction of ARG Representation

The Multi-layer Hierarchical Scheme, proposed in Chapter 4 and utilized here in this experimental work, performs mapping of the image information contents from the spatial domain into the alphabet Σ_O of hierarchical image symbolic representation, which are digitized line segments of different length and orientation. Elements of the alphabet at layer i, $\Sigma_O{}^i \subseteq \Sigma_O$, which is the subset of the global alphabet at the $i'th$ layer, denotes line segments of different length, while elements of the same layer denote line segments of different orientation. The output field of the scheme is a two-dimensional array of cells which take values over Σ_O. Nevertheless, the same information contained in that array can be better represented by an Attributed Relational Graph (ARG) of the form presented in Section 2.4.2.

The extraction of an ARG from the field of the hierarchical scheme is a straight forward conversion of the image representation from the field of the hierarchical scheme into a graph form. An ARG consists of a set of attributed nodes and a set of attributed branches. The nodes represent different features in the image with attributes representing the properties of the corresponding features. The attributed branches represent the relations between the different features in the image. In Sections 6.6 and 6.7 below, we present the ARG representations which are extracted from both of our experiments, namely, Experiment I which deals with multi-object scenes, and Experiment II which is concerned with the SAR image data, respectively.

6.5 Distance Measure and Inexact Matching between ARGs

In Chapter 5, we presented a new efficient approach for calculating a distance measure between two ARGs, in the general form, and finding the best inexact matching configuration between components of the two ARGs. The best inexact matching between the two ARGs is the matching configuration between components of the ARGs such that the distance measure between the two ARGs possesses a global minimum. The overall distance measure between two ARGs is defined in terms of the incremental distance (or local weights) between their respective components, i.e., between the nodes and the branches of the two ARGs. The local weights between nodes or branches are assigned as functions of the respective features or relations which are represented by those nodes or branches, respectively. It is needless to say that these weights of the local error-transformations of node or branch insertion, deletion, or substitution are basically design parameters that are problem dependent, we will illustrate in Section 6.6 and 6.7 for both of our experiments.

The technique presented in Chapter 5 is utilized for finding the matching configuration of minimum global distance between two attributed relational graphs for locating objects in an overlapped multi-object scene, in Experiment I, and for target detection in SAR images, in Experiment II. In Section 6.4, we commented on the extraction of ARG representation from images. The attributed nodes in the ARGs represent different image features in the images with their attributes representing some properties of these features, such as the length of line segments or the span of curve segments. The branches between nodes in the ARG represent the attributed

relations between the respective features represented by those nodes, as will be seen in Sections 6.6 and 6.7 for both Experiment I and Experiment II, respectively.

6.6 Locating Objects in Multi-Object Scenes

In Experiment I, we choose node and branch alphabets as shown in Figure 6.18. The nodes are chosen to represent straight Line segment (L) with length (l) as an attribute, Arc segment (A) with length (l) and span (d) as attributes, and closed Curves (C) with contour length (l) as an attribute. A branch between two nodes in the ARG represents the relationship between the two features represented by these two nodes. Branches are taken to correspond to Joint relation (J) with attribute as the joint angle (ϑ), Intersection relation (I) with the angle of intersection (ϑ) as an attribute, and the relation between non-joint and non-intersecting, i.e. apart or Facing (F), features with attribute (d) represents the distance between the two center points of the two entities.

The set of image global features extracted from the single object model image of Figure 6.2 is shown in Figure 6.19-(a), while the set of relations between those features are shown in Figure 6.19-(b). The ARG representation of that image is shown in Figure 6.20. Similarly, the set of image global features extracted from the multi-object image of Figure 6.3, and the set of relations between those features of that image are shown in Figure 6.21-(a) and 6.21-(b), respectively. The ARG representation of the multi-object image is shown in Figure 6.22.

Node Attribute Alphabet :

Entity	Attributes
Straight Line Segment : L	Length : l
Arc Segment : A	Length : l Span : d
Closed Curve : C	Contour : l

$$A = \{ (L : l), (A : l, d), (C : l) \}$$

Branch Attribute Alphabet :

Relation	Attributes
Joint : J	Angle : ϑ
Intersect : I	Angle : ϑ
Facing : F	Distance : d

$$E = \{ (J : \vartheta), (I : \vartheta), (F : d) \}$$

Fig. 6.18 Node and Branch Attribute Alphabets

GLOBAL FEATURES EXTRACTED FROM THE TRANSFORMATION FIELD :

feature no.	type	attributes
1	l	144
2	a	54, 18
3	l	136
4	a	33, 24
5	a	39, 18
6	a	36, 27
7	c	15

RELATIONS BETWEEN THE EXTRACTED FEATURES :

relation	type	attributes
1, 2	j	130
2, 3	j	165
3, 4	j	140
4, 5	j	50
5, 6	j	40
6, 1	j	150
1, 3	f	12
2, 7	f	3
7, 5	f	153
4, 6	f	33

Fig. 6.19 Global Features and Relations for the Single-Object Image of Fig. 6.2

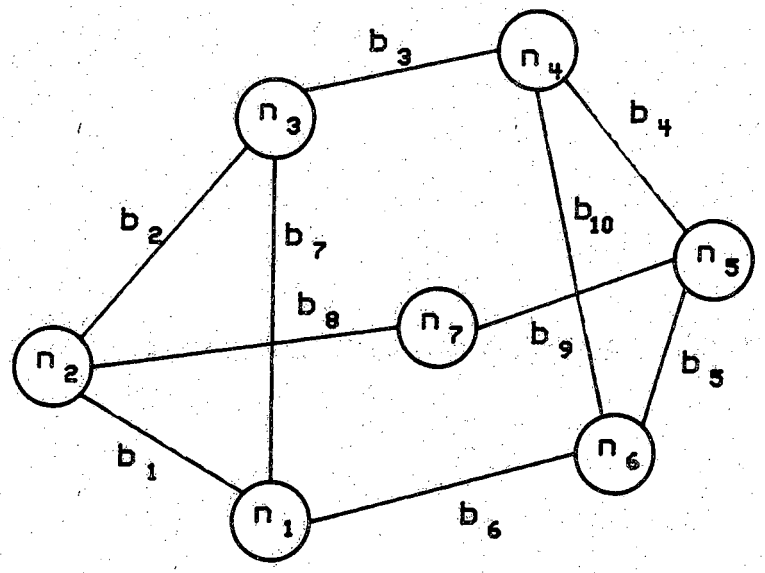


Fig. 6.20 Attributed Relational Graph Representation for the Single-Object Image of Fig. 6.2

GLOBAL FEATURES EXTRACTED FROM THE TRANSFORMATION FIELD :

feature no.	type	attributes
1	l	132
2	a	36, 24
3	a	39, 18
4	a	33, 24
5	l	136
6	a	51, 15
7	c	12
8	a	90, 84
9	a	87, 81
10	a	93, 84
11	a	39, 36
12	a	45, 21
13	a	42, 39
14	a	93, 84
15	a	39, 36
16	l	60
17	l	27
18	l	57
19	l	18
20	a	21, 18
21	a	21, 18
22	l	21
23	a	24, 21
24	a	30, 27
25	l	18
26	l	45
27	l	24

Fig. 6.21 Global Features and Relations for the Multi-Object Image of Fig. 6.3

RELATIONS BETWEEN THE EXTRACTED FEATURES :

relation	type	attributes
1, 2	j	150
2, 3	j	40
3, 4	j	40
4, 5	j	145
5, 10	i	60
10, 11	j	140
11, 13	f	33
13, 12	j	30
12, 11	j	30
9, 10	j	10
1, 14	i	80
1, 20	f	18
20, 21	j	110
21, 1	f	24
21, 22	f	3
22, 23	j	90
23, 14	j	100
24, 18	f	6
18, 17	j	90
18, 23	f	6
17, 16	j	90
16, 15	j	75
15, 27	f	12
27, 26	j	90
26, 16	f	15
26, 25	j	90
25, 18	j	90
1, 8	i	70
8, 14	j	10
1, 10	i	70
1, 6	j	170
1, 5	f	12
5, 6	j	150
2, 4	f	27
3, 7	f	144
7, 6	f	3
8, 5	i	70
5, 14	i	70
5, 9	i	70
9, 8	j	100
19, 20	f	3
19, 17	f	21

Fig. 6.21 (Continued)

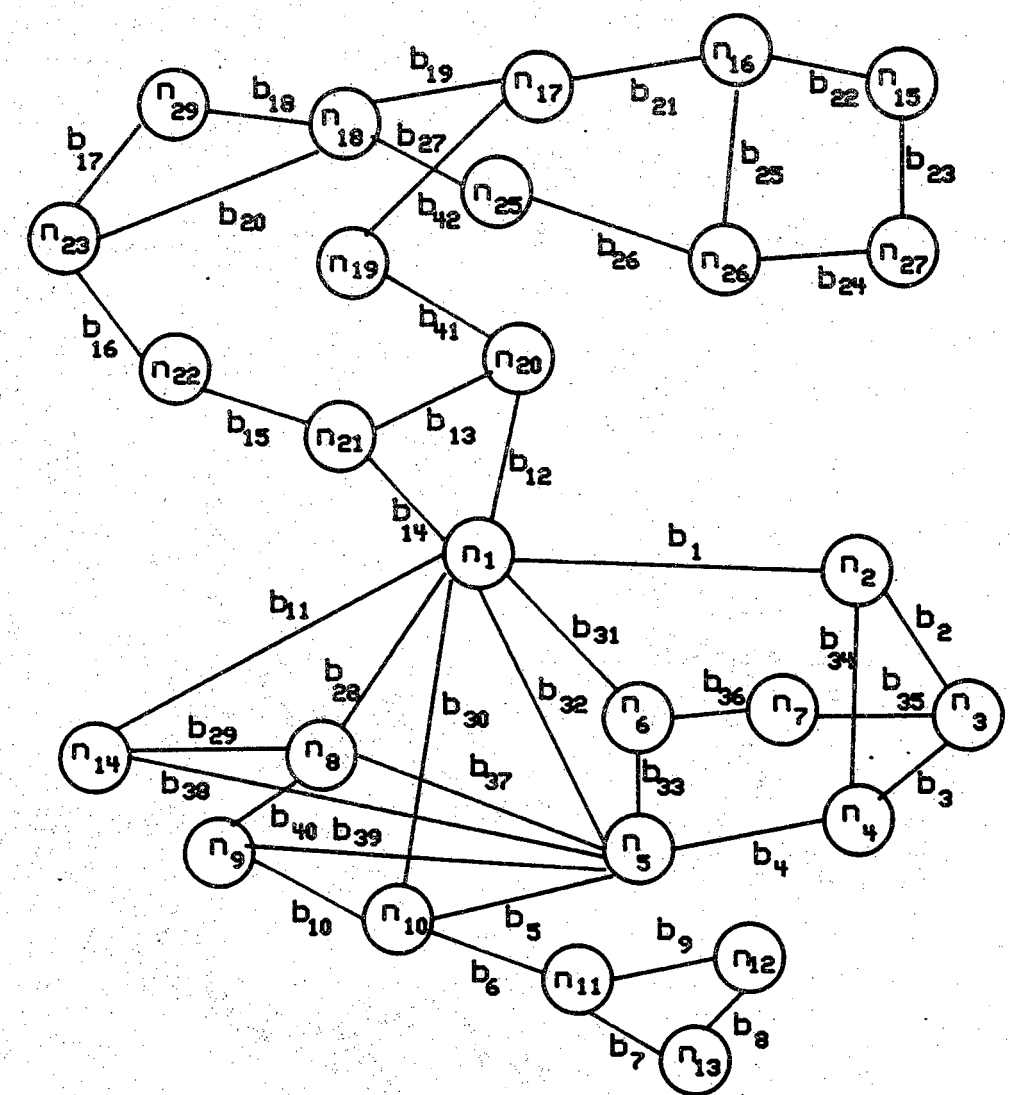


Fig. 6.22 Attributed Relational Graph Representation for the Multi-Object Image of Figure 6.3

Table 6.1 Assigned Weights of Node Insertion and Deletion for Experiment I

Node	Insertion	Deletion
(L : l_1)	l_1	l_1
(A : l_1, d_1)	l_1	l_1
(C : l_1)	l_1	l_1

Table 6.2 Assigned Weights of Branch Insertion and Deletion for Experiment I

Branch	Insertion	Deletion
(J : ϑ_1)	ϑ_1	ϑ_1
(I : ϑ_1)	ϑ_1	ϑ_1
(F : d_1)	d_1	d_1

Table 6.3 Assigned Weights of Node Substitution for Experiment I

	$(L : l_1)$	$(A : l_1, d_1)$	$(C : l_1)$
$(L : l_2)$	$\lvert l_1-l_2\rvert$	$\lvert l_1-l_2\rvert+d_1$	l_1+l_2
$(A : l_2, d_2)$	$\lvert l_1-l_2\rvert+d_2$	$\lvert l_1-l_2\rvert+$ $\lvert d_1-d_2\rvert$	$\lvert l_1-l_2\rvert+d_2$
$(C : l_2)$	l_1+l_2	$\lvert l_1-l_2\rvert+d_1$	$\lvert l_1-l_2\rvert$

Table 6.4 Assigned Weights of Branch Substitution for Experiment I

	$(J : \vartheta_1)$	$(I : \vartheta_1)$	$(F : d_1)$
$(J : \vartheta_2)$	$\lvert\vartheta_1-\vartheta_2\rvert$	$\lvert\vartheta_1-\vartheta_2\rvert$	ϑ_2+d_1
$(I : \vartheta_2)$	$\lvert\vartheta_1-\vartheta_2\rvert$	$\lvert\vartheta_1-\vartheta_2\rvert$	ϑ_2+d_1
$(F : d_2)$	ϑ_1+d_2	ϑ_1+d_2	$\lvert d_1-d_2\rvert$

For this experiment in hand, we choose the weights of insertion and deletion of different nodes and branches of the attributed relational graphs, as shown in Tables 6.1 and 6.2, for the insertion and deletion, respectively. While, Tables 6.3 and 6.4 show the weights of substitution (or relabeling) of nodes and branches, respectively. The physical meaning of these weights of error-transformations can be clearly seen from the physical meaning of the attributes of the different entities of the attributed relational graphs. The approach proposed in Chapter 5 is used to locate the object of Figure 6.2 in the multi-object overlapped scene of Figure 6.3, by calculating the graph distance measure between their respective ARGs, which are shown in Figures 6.20 and 6.22, and finding the best inexact matching configuration which possesses minimum global distance between the two attributed relational graphs. The results of this experiment are shown by the matching configuration shown in Figure 6.23.

6.7 Target Detection in SAR Images

In this experiment, Experiment II, we choose node alphabet to represent arbitrary straight line segments (L), with the length (l) as their attributes. Branches of the ARG representation in this experiment represent the relationships between the line segments which are represented by the nodes. The branches are taken to correspond to the following possible relations: Parallel (P) with attribute as the distance (d) between the two line segments, Joint (J) with attribute as the joint angle (ϑ), and Intersection (I) with the angle of intersection (ϑ) as an attribute.

THE BEST INEXACT MATCHING BETWEEN THE TWO ARGs U and V

NODE-PAIRS

GRAPH V	GRAPH U
n1	n5
n2	n6
n3	n1
n4	n2
n5	n3
n6	n4
n7	n7

BRANCH-PAIRS

GRAPH V	GRAPH U
b1	b33
b2	b31
b3	b1
b4	b2
b5	b3
b6	b4
b7	b32
b8	b36
b9	b35
b10	b34

THE MINIMUM DISTANCE = 84

Fig. 6.23 Matching Configuration and Minimum Distance for the Two ARGs of Figures 6.20 and 6.22

The set of image global features extracted from the SAR image of the Santa Barbara area is shown in Figure 6.24-(a), while the set of relations between these features are shown in Figure 6.24-(b). The ARG representation of that image is shown in Figure 6.25. The sets of attributed features and relations of a model of an airport are shown in Tables 6.5 and 6.6, while its ARG representation is shown in Figure 6.26.

The weights of insertion and deletion of different nodes and branches of the attributed relational graphs, in this experiments are shown in Table 6.7 and Table 6.8, for insertion and deletion, respectively. Tables 6.9 and 6.10 show the weights of substitutions (or relabeling) of nodes and branches, respectively. The technique proposed in Chapter 5 is, then, utilized to find the best inexact matching configuration between the ARG representation of the SAR image and that of a model target airport, and calculating the global distance measure between the two ARGs. The results of this experiment are shown in Figures 6.27 and 6.28.

GLOBAL FEATURES EXTRACTED FROM THE TRANSFORMATION FIELD :

feature no.	type	attributes
1	1	40
2	1	12
3	1	10
4	1	12
5	1	30
6	1	20
7	1	30
8	1	60
9	1	40
10	1	16
11	1	16
12	1	36
13	1	8
14	1	16
15	1	10
16	1	10
17	1	16
18	1	14
19	1	20
20	1	16
21	1	18
22	1	32
23	1	10
24	1	8
25	1	20
26	1	18
27	1	12
28	1	16
29	1	20
30	1	10

Fig. 6.24 Global Features and Relations for the SAR Image of Santa Barbara Area

RELATIONS BETWEEN THE EXTRACTED FEATURES :

relation	type	attributes
1, 2	j	70
2, 3	p	8, 12
3, 4	p	16, 10
1, 5	p	24, 10
1, 6	j	30
1, 7	o	70, 75
7, 8	j	75
7, 10	p	10, 10
8, 9	j	80
8, 11	p	16, 8
9, 12	p	12, 4
12, 13	p	0, 60
13, 14	j	75
14, 15	j	90
15, 16	j	45
16, 17	p	20, 0
16, 18	p	22, 4
17, 18	j	135
12, 19	o	54, 90
13, 19	o	60, 110
19, 20	p	26, 0
11, 21	p	72, 0
21, 22	j	90
22, 23	j	90
23, 24	j	90
22, 24	p	16, 8
25, 26	j	75
26, 27	j	75
27, 28	o	10, 45
28, 29	j	90
29, 30	j	90
21, 29	p	20, 0
21, 30	o	12, 90

Fig. 6.24 (Continued)

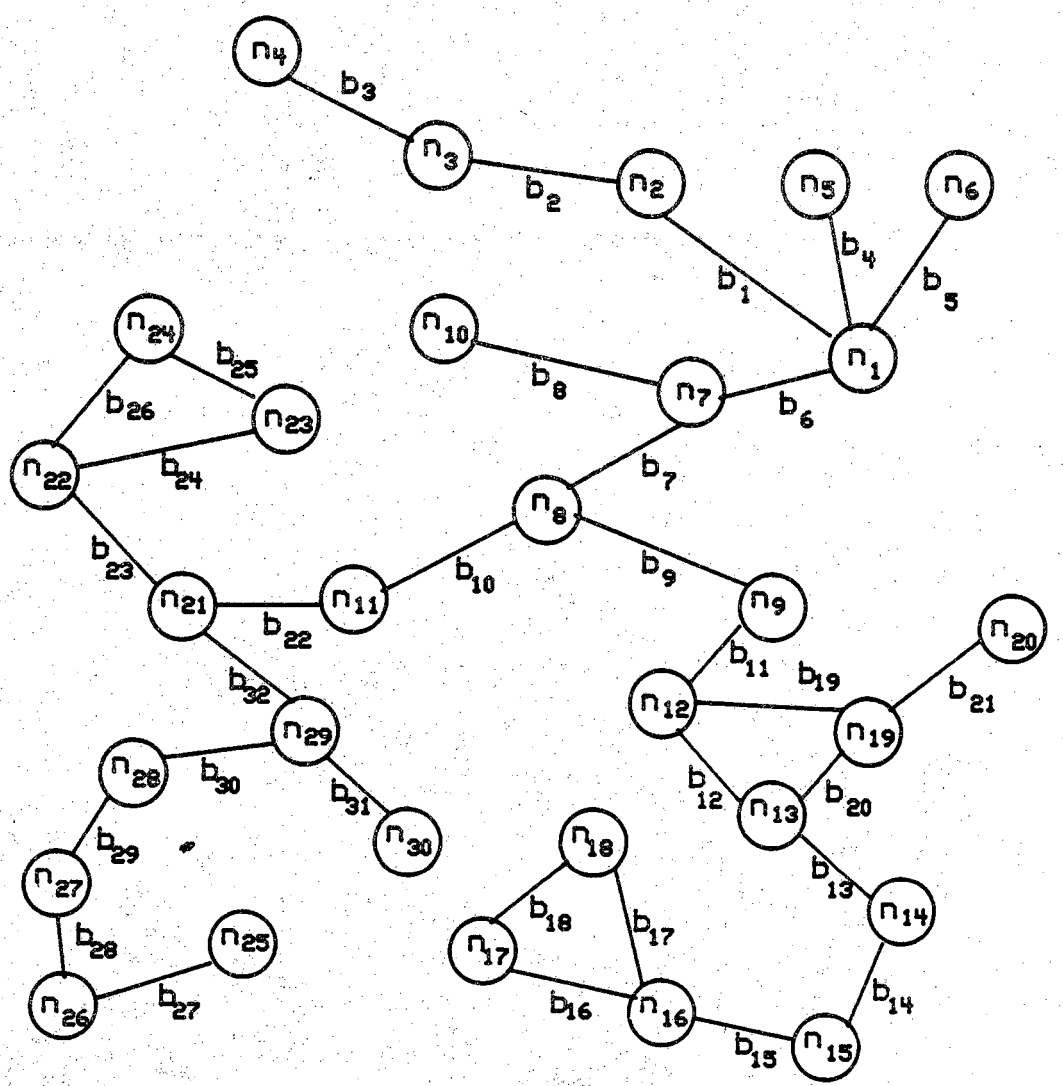


Fig. 6.25 Attributed Relational Graph Representation for SAR Image of Santa Barbara Area

Table 6.5 Attributed Features for a Model of an Airport

Feature No.	Type	Attributes
n_1	1	10
n_2	1	30
n_3	1	30
n_4	1	10
n_5	1	30
n_6	1	10
n_7	1	10
n_8	1	10
n_9	1	20
n_{10}	1	10
n_{11}	1	20
n_{12}	1	30

Table 6.6 Attributed Relations between Image Features for a Model of an Airport

Relation	Type	Attributes
1, 2	j	90
2, 3	j	120
3, 4	j	90
4, 5	j	90
5, 6	j	60
6, 7	j	90
7, 8	j	90
8, 9	j	90
9, 10	j	90
10, 11	j	90
11, 12	j	90
1, 7	p	50, 0
1, 12	j	90
2, 12	p	10, 0
3, 5	p	10, 0
4, 12	p	40, 30
6, 8	p	10, 0
9, 11	p	10, 0

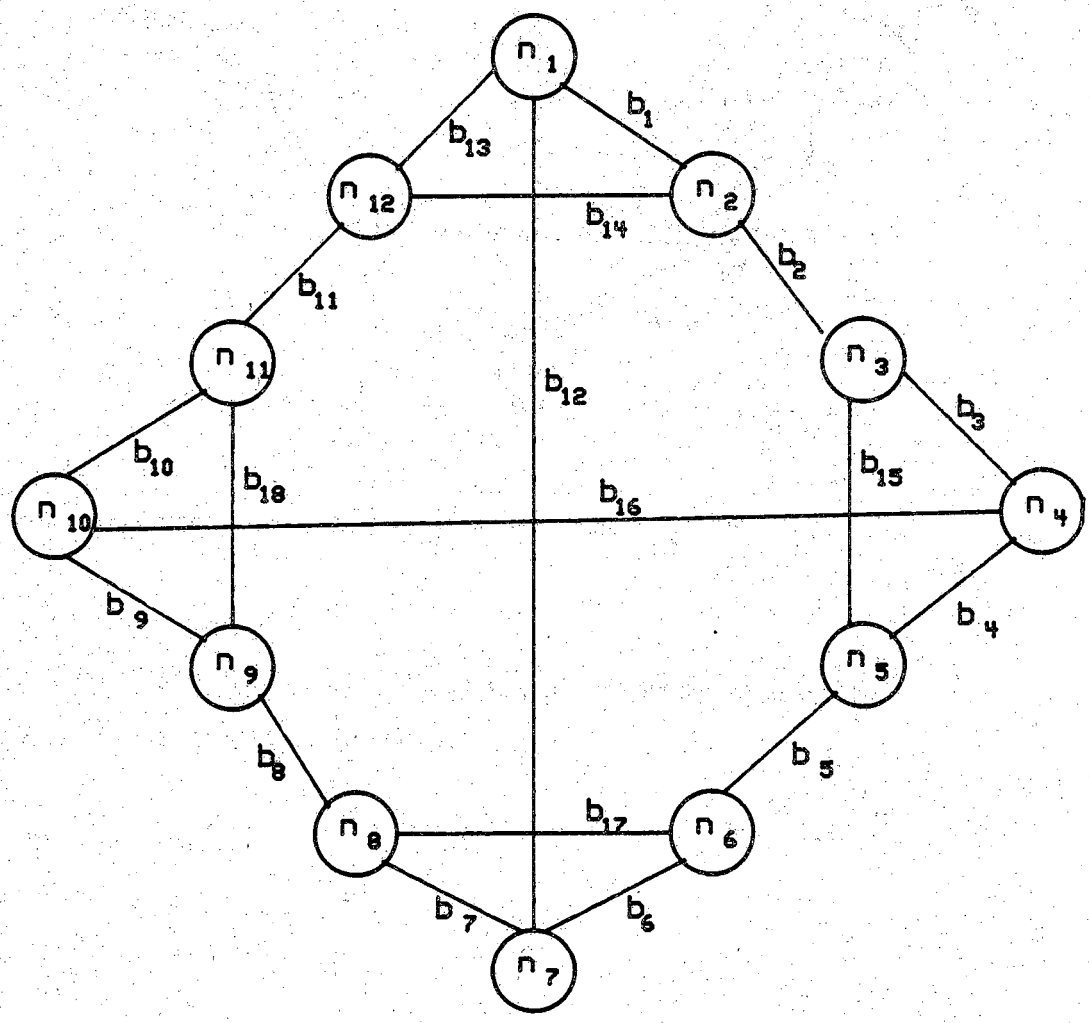


Fig. 6.26 Attributed Relational Graph Representation for a Model of an Airport

Table 6.7 Assigned Weights of Node Insertion and Deletion for Experiment II

Node	Insertion	Deletion
$(L : l_1)$	l_1	l_1

Table 6.8 Assigned Weights of Branch Insertion and Deletion for Experiment II

Branch	Insertion	Deletion
$(J : \vartheta_1)$	ϑ_1	ϑ_1
$(O : d_1, \vartheta_1)$	$d_1 + \vartheta_1$	$d_1 + \vartheta_1$
$(P : d_1, d_2)$	$d_1 + d_2$	$d_1 + d_2$

Table 6.9 Assigned Weights of Node Substitution for Experiment II

	(L : l_1)
(L : l_2)	$\lvert l_1 - l_2 \rvert$

Table 6.10 Assigned Weights of Branch Substitution for Experiment II

	(J : ϑ_1)	(O : d_1, ϑ_1)	(P : d_{11}, d_{12})
(J : ϑ_2)	$\lvert \vartheta_1 - \vartheta_2 \rvert$	$\lvert \vartheta_1 - \vartheta_2 \rvert + d_1$	$\vartheta_2 + d_{11} + d_{12}$
(O : d_2, ϑ_2)	$\lvert \vartheta_1 - \vartheta_2 \rvert + d_2$	$\lvert \vartheta_1 - \vartheta_2 \rvert +$ $\lvert d_1 - d_2 \rvert$	$\lvert 90 - \vartheta_2 \rvert +$ $\lvert d_{11} - d_2 \rvert$
(P : d_{21}, d_{22})	$\vartheta_1 + d_{21} + d_{22}$	$\lvert 90 - \vartheta_1 \rvert +$ $\lvert d_{21} - d_1 \rvert$	$\lvert d_{11} - d_{21} \rvert +$ $\lvert d_{22} - d_{12} \rvert$

THE BEST INEXACT MATCHING BETWEEN THE TWO ARGs U and V

NODE-PAIRS

GRAPH V	GRAPH U
n1	n23
n2	n24
n3	*
n4	*
n5	n25
n6	*
n7	*
n8	n28
n9	n29
n10	n30
n11	n21
n12	n22

BRANCH-PAIRS

GRAPH V	GRAPH U
b1	b25
b2	*
b3	*
b4	*
b5	*
b6	*
b7	*
b8	b30
b9	b31
b10	b33
b11	b23
b12	*
b13	b24
b14	b26
b15	*
b16	*
b17	b29
b18	b32

THE MINIMUM DISTANCE = 219

Fig. 6.27 Matching Configuration and Minimum Distance for the Two ARGs of Figures 6.25 and 6.27

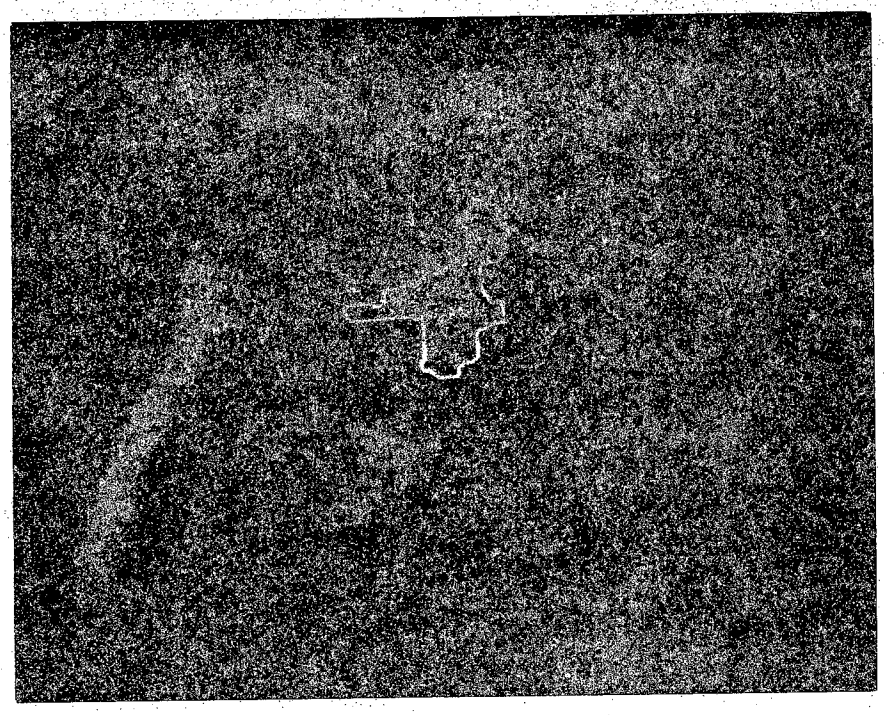


Fig. 6.28 Approximate Location of an Airport in the SAR Image of Fig. 6.4

CHAPTER VII

CONCLUSIONS AND SUGGESTIONS

7.1 Summary of Results and Contributions

Our research contributions in the course of this thesis are concentrated basically in two new techniques. First is the hierarchical graph transducer for the extraction of image global representation in the form of attributed relational graph from raw input images. The second technique is the dynamic programming approach to distance computation between two attributed relational graphs and the extraction of the best inexact matching configuration between the two graphs. In Sections 7.1.1 and 7.1.2, we state some concluding remarks on each of these techniques and on their utilization in image understanding, as investigated in this thesis, as well as in other applications of knowledge representation and machine intelligence in general.

7.1.1 Concluding Remarks on the Hierarchical Graph Transducer

Traditionally, there have been two major areas of research in the field of machine vision and computer image analysis, namely, the "Low-Level" image processing and the "High-Level" image understanding. In the area of image processing, the main emphasis lies on some image processing operations of very local nature. The objective of most image processing techniques is to improve the quality of the images or to extract some local information from the images, e.g., filtering, enhancement, local edge operators, etc. Usually, the media of image representation in this area, not only as input but also throughout the operations, is basically the spatial domain representation. In most cases, the output of an image processing system is a processed image, or a better quality image, but still in the spatial domain representation.

On the other hand, in the area of image understanding, even though input images are usually given in the spatial domain form, but the main media of image representation used in most techniques is usually of more global nature, such as (attributed) strings, trees, or graphs. Such global representations must be extracted from the input spatial domain images. The major task of image understanding is usually performed on the extracted global representation. The output of these techniques is the result of some decision making processes and conclusive decisions about the contents of the images, e.g., locating objects, or recognizing some patterns, etc.

The problem of extracting a global representation from input data is an important problem in the field of machine intelligence and knowledge engineering, in general, and in computer vision and image understanding in particular.

The hierarchical graph transducer described in Chapter 4 is proposed as a systematic technique for extracting general global representation scheme in the form of attributed relational graph from the input images that are given in the spatial domain representation. The proposed technique possesses several advantages which we state them briefly as follows:

* It provides a systematic recursive technique to bridge the gap between low-level image processing and high-level image understanding techniques.

* It utilizes both model-driven and data-driven concepts, as shown in Chapter 4. The model information are used in the design of the different components of the graph transformation used in the transducer, while the symbolic mapping from the input alphabet to the output global alphabet is performed on the input data.

* The proposed transducer operates in an arbitrary configurable field of cells, e.g., two dimensional array of cells, thus it preserves all useful information and properties of the imagery data, e.g., symmetry, closure of curves, connectivity, etc.

* The number of layers used in the transducer is actually very small, since the size of subimages covered by the cells of different layers of the field grows exponentially, as shown in Chapter 4.

* The technique is suitable for hardware implementation from two points of view, both can be seen from the mapping algorithm which is given in Chapter 4. First, the graph transformation mapping can be performed in parallel over all cells of any layer. Second, all mapping operations are modeled as simple bit-wise manipulation. Both aspects are very helpful for high-speed and easy implementable hardware. The details of the hardware aspect of the proposed scheme is beyond the scope of our research in this thesis.

* The main disadvantage of the proposed technique is the large size of the output alphabet, however this only happens when the class of interesting objects is not known. In such cases, the graph transducer will need to include a large set of features in the output alphabet. This is usually the case of learning or inferencing of prototype model representations, where the transducer mapping is mainly driven by the input data only. The large size of output alphabet usually represents a disadvantage of the proposed technique, since it increases the processing time required by hierarchical graph transducer mapping.

7.1.2 Concluding Remarks on the Graph Distance and Inexact Matching Approach

In Chapter 5, we proposed a new approach for calculating a distance measure between two attributed relational graphs, in their general form as defined in Section 2.4.2, as well as finding the best inexact matching configuration between the respective components of the two graphs. Such an inexact matching configuration possesses minimum global distance between the two attributed relational graphs. In this section, we discuss some advantages of our approach to this combinatorial optimization problem.

* The proposed approach handles the problem of distance measure and inexact matching between two attributed relational graphs in general form.

* The state-space representation generated by the approach contains several feasible paths between the initial node and the final nodes, and then the approach utilizes a dynamic programming technique, with linear computational complexity, to search for the global optimum, i.e., the shortest path in the state-space representation from the initial state to any of the final states.

* The approach can handle global as well as local deformations in the attributed relational graphs, as shown in Chapter 5.

* The computational complexity of the proposed approach is $O(M^2N^2(M+N))$, where M and N are the number of nodes in the two attributed relational graphs. This complexity is far better than most approaches for similar problems, [eshe84a].

* The proposed approach performs, in its general version, a two-way matching process.

* The costs of different error-transformations, as defined locally on nodes or branches, can take either deterministic or stochastic form. They are not required to be independent since they are used to calculate the incremental cost, and the global search is performed by the shortest-path dynamic programming algorithm. As we discussed in Section 6.5, the choice of these costs is problem dependent, i.e., they are actually design parameters.

* The approach is amenable to parallel architecture implementation in generating the state-space representation, as well as in performing the shortest-path search. A clear reason for this is the multi-stage nature of the state-space representation, [chia81], [kana81a].

* In the proposed approach, we assumed that the upper bound for the number of directly connected, or adjacent, nodes in either of the two attributed relational graphs is a constant, which is small in comparison with the total number of nodes in the graphs. The case of a complete graph, i.e., a graph in which every node is directly connected to all the other nodes, will cause the computational complexity of the proposed approach to

increase. This is basically the only handicap of the technique. Nevertheless, complete graphs are very seldomly needed in practical applications.

7.2 Suggestions for Future Research

There are several interesting topics of research which emanate from our work in the course of this thesis. In the remaining part of this section, we discuss some of these topics.

(1) The hierarchical graph transducer presented in Chapter 4, uses the model-object information to restrict the alphabet of image features into only those features which are obtained from the recursive decomposition of complex objects into sub-objects and finally into image primitives. The inferencing of such an alphabet of features can be done manually or automatically by a training procedure. The graph transducer, as presented in Chapter 4, is capable of performing the required inference in a straight forward fashion by first considering the set of all image features over the different sub-images. In this case, the mapping is mainly driven by the input data only. The extracted image features are then considered as the required alphabet to be used later in the technique. However, the set of all features might be large and therefore, the technique will be slow in performing the learning task. A faster, or more intelligent, learning technique might be needed to infer the alphabet of image features, which the technique uses during the operation stage.

(2) The inference procedure suggested above can be incorporated with the rest of our system in a more general vision system, as shown in Figure 7.1. The system possesses the learning capability through the inference procedure to build up the alphabet of image features and primitives. It uses attributed relational graphs as the media of image knowledge representation. Both the image feature alphabet and the attributed relational graph representation of the model-objects are extracted from the model images during the learning phase. During the operation phase, the system utilizes the alphabet of image features inferred from the model images to extract the attributed relational graph representation of any given image, as shown in Chapter 4.

(3) The two new approaches proposed in Chapters 4 and 5 of this thesis, are also useful in several other applications of machine intelligence. Each of the proposed techniques provide an efficient solution to an essential problem in the field of machine intelligence. The hierarchical graph transducer provides a systematic methodology for learning and inference of a global knowledge representation from given input data that is defined in terms of local entities (or evidence). This is usually an important issue in artificial intelligence and knowledge engineering that need to be investigated further. Also, the utilization of the dynamic programming approach for distance measure and inexact matching between two general attributed relational graphs in other application areas of machine intelligence need to be studied more, e.g., its usage for defining certainty measures between collective information representation, or for inexact information retrieval from databases, etc.

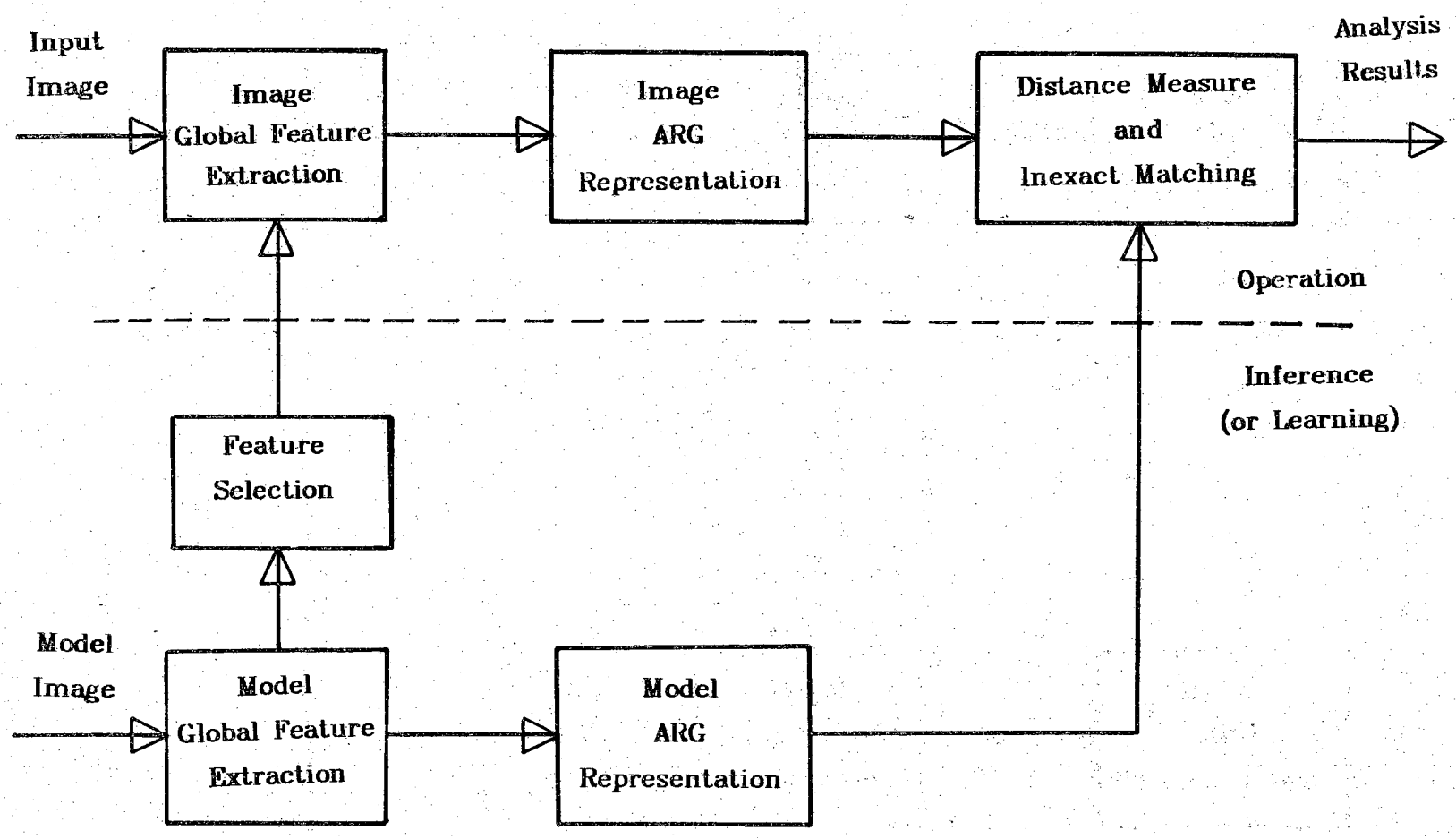


Fig. 7.1 Block Diagram of a More General Image Understanding System

LIST OF REFERENCES

[ahoa72] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation, and Compiling*, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[ahuj81] N. Ahuja, "Approaches to Recursive Image Decomposition," in *Proceedings of the International Conference on Pattern Recognition and Image Processing, PRIP-81*, pp. 75-80, 1981.

[ahuj84] N. Ahuja and S. Swamy, "Multiprocessor Pyramid Architectures for Bottom-Up Image Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 4, pp. 463-475, July 1984.

[anto82] H. J. Antonisse, "Image Segmentation in Pyramids," *Computer Graphics and Image Processing*, vol. 19, 1982.

[asad84] M. Asada, M. Yachida, and S. Tsuji, "Analysis of Three-Dimensional Motions in Blocks World," *Pattern Recognition*, vol. 17, no. 1, pp. 57-71, 1984.

[ball81] D. H. Ballard, "Strip Trees: A Hierarchical Representation for Curves," *Communications of the ACM*, vol. 24, pp. 310-321, 1981.

[ball82] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, 1982.

[bhan83] B. Bhanu, "Recognition of Occluded objects," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, pp. 1136-1138, August 8-12, 1983.

[bond76] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North Holland, 1976.

[brau84] D. J. Braunegg and R. C. Gonzalez, "An Approach to Industrial Computer Vision Using Syntactic/Semantic Learning Techniques," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 1366-1369, Montreal, Canada, July 30 - August 2, 1984.

[bunk81] H. O. Bunke, "Attributed Programmed Graph Grammars as Tool for Image Interpretation," TR-EE, 81-22, Purdue University, 1981.

[bunk84] H. O. Bunke and G. Sagerer, "Use and Representation of Knowledge in Image Understanding Based on Semantic Networks," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 1135-1137, Montreal, Canada, July 30 - August 2, 1984.

[burt80] P. J. Burt, "Tree and Pyramid Structures for Coding Hexagonally Sampled Binary Images," *Computer Graphics and Image Processing*, vol. 14, pp. 271-280, 1980.

[burt81] P. Burt, T. H. Hong, and A. Rosenfeld, "Segmentation and Estimation of Image Region Properties through Cooperative Hierarchical Computation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-11, pp. 802-809, 1981.

[carr79] B. Carre, *Graphs, and Networks*, Clarendon Press, Oxford, 1979.

[chan79] C. L. Chang and J. R. Slagle, "Using Rewriting Rules for Connection Graphs to Prove Theorems," *Artificial Intelligence*, vol. 12, pp. 159-180, 1979.

[chan79] N. S. Chang and K. S. Fu, "Parallel Parsing of Tree Languages for Syntactic Pattern Recognition," *Pattern Recognition*, vol. 11, pp. 213-222, 1979.

[chan80] N. S. Chang and K. S. Fu, "An Integrated Image Analysis and Image Database Management System," TR-EE, 80-20, Purdue University, May 1980.

[chan82] N. S. Chang and K. S. Fu, "A Relational Database System for Images," in *Pictorial Database Systems*,, K. S. Fu and S. K. Cheng (Eds.),, 1982.

[chen79] P. C. Chen and T. Pavlidis, "Segmentation by Texture Using a Co-occurrance Matrix and a Split-and-Merg Algorithm," *Computer Graphics and Image Processing*, vol. 10, pp. 172-182, 1979.

[chen81a] J. K. Cheng and T. S. Huang, "A Subgraph Isomorphism Algorithm Using Resolution," *Pattern Recognition*, vol. 13, no. 5, pp. 371-379, 1981.

[chen81b] J. K. Cheng and T. S. Huang, "Image Recognition by Matching Relational Structures," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 542-547, Dallas, Texas, August, 3-5, 1981.

[chen84] J. K. Cheng and T. S. Huang, "Image Registration by Matching Relational Structures," *Pattern Recognition*, vol. 17, no. 1, pp. 149-159, 1984.

[chia81] Y. Chiang and K. S. Fu, "Parallel Processing for Distance Computation in Syntactic Pattern Recognition," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 337-344, Dallas, Texas, 1981.

[chia83] Y. T. Chiang and K. S. Fu, "VLSI Architectures for Syntactic Pattern Recognition," in *Proceedings of the 1983 CVPR Conference*, Arlington, VA, June 21-23, 1983.

[cowi83] R. I. D. Cowie, "The Viewer's Place in Theories of Vision," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[davi76] L. S. Davis and A. Rosenfeld, "An Application of Relaxation Labeling to Spring-loaded Template Matching," in *Proceedings of Third International Joint Conference on Pattern Recognition*, pp. 591-597, 1976.

[davi78] L. S. Davis and A. Rosenfeld, "Hierarchical Relaxation for Waveform Parsing," in *Computer Vision Systems*, pp. 101-109, Academic Press Inc., 1978.

[dena79a] E. V. Denardo and B. L. Fox, "Shortest-Route Methods: 1. Reaching, Pruning, and Buckets," *Operations Research*, vol. 27, no. 1, January-February 1979.

[dena79b] E. V. Denardo and B. L. Fox, "Shortest-Route Methods: 2. Group Knapsacks, Expanded Networks, and Branch-and-Bound," *Operations Research*, vol. 27, no. 3, May-June 1979.

[dena82] E. V. Denardo, *Dynamic Programming : Theory and Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[diam83] M. D. Diamond, N. Narasimhamurthi, and S. Ganapathy, "Optimization Approaches to the Problem of Edge Linking with a Focus on Parallel Processing," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[duda72] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11-15, January 1972.

[dyer77a] C. R. Dyer, "Cellular Pyramids for Image Analysis, 1," TR-544, Computer Science Technical Report Series, University of Maryland, College Park, May 1977.

[dyer77b] C. R. Dyer, "Cellular Pyramids for Image Analysis, 2," TR-596, Computer Science Technical Report Series, University of Maryland, College Park, November 1977.

[dyer78] C. R. Dyer, "One-Way Bounded Cellular Acceptors," TR-680, Computer Science Technical Report Series, University of Maryland, College Park, July 1978.

[dyer79] C. R. Dyer and A. Rosenfeld, "Thinning Algorithms for Gray-Scale Pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 1, January 1979.

[dyer80] C. R. Dyer, A. Rosenfeld, and H. Samet, "Region Representation: Boundary Codes from Quadtrees," *Communications of the ACM*, vol. 23, pp. 171-179, 1980.

[dyer81a] C. R. Dyer, "A VLSI Pyramid Machine for Hierarchical Parallel Processing," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 381-386, IEEE, Dallas, Texas, 1981.

[dyer81b] C. R. Dyer and A. Rosenfeld, "Parallel Image Processing by Memory-Augmented Cellular Automata," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 1, January 1981.

[eshe80] M. A. Eshera, "The Generation and Recognition of Noisy and Defective Images by Multi-Dimensional Syntactic Technique," M. Sc. Thesis, Drexel University, Philadelphia, Pennsylvania, 1980.

[eshe83] M. A. Eshera, H. S. Don, K. Matsumoto, and K. S. Fu, "A Syntactic Approach for SAR Image Analysis," in *Proceedings of the ONR Workshop on Statistical Image Processing and Graphics*, Luray, VA, May 24-27, 1983.

[eshe84a] M. A. Eshera and K. S. Fu, "A Graph Distance Measure for Image Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 398-408, May/June 1984.

[eshe84b] M. A. Eshera and K. S. Fu, "A Measure of Similarity between Attributed Relational Graphs for Image Analysis," in *Proceedings of Seventh International Conference on Pattern Recognition*, pp. 75-77, Montreal, Canada, July 30-August 2, 1984.

[even79] S. Evens, *Graph Algorithms*, Computer Science Press, 1979.

[fang83] J-Q. Fang and T. S. Huang, "Estimating 3-D Movement of a Rigid Object: Experimental Results," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[faug83] O. D. Faugeras and J. Ponce, "PRISM TREES: A Hierarchical Representation for 3-D Objects," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*,

IJCAI-83, August 8-12, 1983.

[fisc83] M. A. Fischler and H. C. Wolf, "Machine Perception of Linear Structure," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[fish83] R. B. Fisher, "Using Surface and Object Models to Recognize Partially Obscured Objects," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, pp. 989-995, August 8-12, 1983.

[fros82] V. S. Frost, J. A. Stiles, K. S. Shanmugan, and J. C. Holtzman, "A Model for Radar Images and Its Application to Adaptive Digital Filtering for Multiplicative Noise," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 2, pp. 157-166, March 1982.

[fuks80] K. S. Fu, "Recent Development in Pattern Recognition," *IEEE Transactions on Computers*, vol. C-29, no. 10, pp. 845-853, October 1980.

[fuks82a] K. S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[fuks82b] K. S. Fu, "Attributed Grammars for Pattern Recognition - A General (Syntactic-Semantic) Approach," in *Proceedings of the IEEE*, pp. 18-27, 1982.

[fuks82c] K. S. Fu, "Pattern Recognition for Automatic Visual Inspection," in *Computer 15*, (Special Issue on) Robotics and Automation, IEEE, December, 1982.

[fuks83a] K. S. Fu, "A Step Towards Unification of Syntactic and Statistical Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 2, pp. 200-205, March 1983.

[fuks83b] K. S. Fu, "Pictorial Pattern Recognition for Industrial Inspection," in *Pictorial Data Analysis*, R. M. Haralick (Ed.), Springer-Verlag, Berlin Heidelberg, 1983.

[fuks83c] K. S. Fu, "Robot Vision for Machine Part Recognition," in *Proceedings of the SPIE Conference 442 "Robotics and Robot Sensing Systems"*, San Diego, CA, August 25, 1983.

[fuku72] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1972.

[gare79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Bell Telephone Laboratory, Inc., 1979.

[ghah80a] D. E. Ghahraman, A. K. C. Wong, and T. Au, "Graph Optimal Monomorphism Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 4, pp. 181-188, April 1980.

[ghah80b] D. E. Ghahraman, A. K. C. Wong, and T. Au, "Graph Monomorphism Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 4, pp. 189-196, April 1980.

[gonz77] R. C. Gonzalez and P. Wintz, *Image Processing*, Addison-Wesley Publishing Company, Inc., 1977.

[gonz78] R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition: An Introduction*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1978.

[gonz82] R. C. Gonzalez and R. Safabakhsh, "Computer Vision Techniques for Industrial Applications and Robot Control," in *Computer 15*, (Special Issue on) Robotics and Automation, IEEE, December, 1982.

[hall76] E. L. Hall, Lt. J. Rouge, and R. Y. Wong, "Hierarchical Search for Image Matching," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 791-796, Clearwater Beach, Fal., Dec. 1976.

[hans76] A. R. Hanson and E. M. Riseman, "A Progress Report on VISIONS: Representation and Control in the Construction of Visual Modles," COINS Technical Report 76-9, Univ. of Mass., Amherst, July 1976.

[hans78] A. R. Hanson and E. M. Riseman (Eds.), *Computer Vision Systems*, Academic Press, 1978.

[hans80] A. R. Hanson and E. M. Riseman, "Processing Cones: A Computational Structure for Image Analysis," in *Structured Computer Vision*, S. Tanimoto and A. Klinger (Eds.), Academic Press, 1980.

[hara78] R. M. Haralick and J. S. Kartus, "Arrangements, Homomorphisms, and Discrete Relaxation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-8, no. 8, pp. 600-612, August 1978.

[hara79] R. M. Haralick and L. G. Shapiro, "The Consistent Labeling Problem: Part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 173-184, April 1979.

[hara80a] R. M. Haralick and L. G. Shapiro, "The Consistent Labeling Problem: Part II," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 3, pp. 193-203, May 1980.

[hara80b] R. M. Haralick, J. L. Mohammed, and S. W. Zucker, "Compatibilities and the Fixed Points of Arithmetic Relaxation Processes," *Computer Graphics and Image Processing*, no. 13, pp. 242-256, 1980.

[hend81] T. C. Henderson and L. Davis, "Hierarchical Models and Analysis of Shape," *Pattern recognition*, vol. 14, no. 1-6, pp. 197-204, Pergamon Press Ltd., 1981.

[hong80] T. H. Hong, C. R. Dyer, and A. Rosenfeld, "Texture Primitive Extraction Using an Edge-Based Approach," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 10, October 1980.

[hong82a] T. H. Hong, K. A. Narayanan, S. Peleg, A. Rosenfeld, and T. Silberger, "Image Smoothing and Segmentation by Multiresolution Pixel Linking: Further Experiments and Extensions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, 1982.

[hong82b] T. H. Hong, M. Shneier, and A. Rosenfeld, "Border Extraction Using Linked Edge Pyramids," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, pp. 660-668, 1982.

[hong82c] T. H. Hong and M. O. Shneier, "Extracting Compact Objects Using Linked Pyramids," in *Proceedings of Image Understanding Workshop*, pp. 58-71, Palo Alto, CA, 1982.

[hong82d] T. H. Hong and A. Rosenfeld, "Unforced Image Partitioning by Weighted Pyramid Linking," in *Proceedings of Image Understanding Workshop*, pp. 72-78, Palo Alto, CA, 1982.

[hopc69] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1969.

[horo78a] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.

[horo78b] S. L. Horowitz and T. Pavlidis, "Note: A Graph-Theoretic Approach to Picture Processing," *Computer Graphics and Image Processing*, vol. 7, pp. 282-291, Academic Press Inc., 1978.

[hung83] S. H. Y. Hung and T. Kasvand, "Linear Approximation of Quantized Thin Lines," in *Pictorial Data Analysis*, R. M. Haralick (Ed.), Springer-Verlag, Berlin Heidelberg, 1983.

[hunt79a] G. M. Hunter and K. Steiglitz, "Operations on Images Using Quadtrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 145-153, Jan. 1979.

[hunt79b] G. M. Hunter and K. Steiglitz, "Note: Linear Transformations of Pictures Represented by Quadtrees," *Computer Graphics and Image Processing*, vol. 10, pp. 289-296, 1979.

[ichi81] T. Ichikawa, "A Pyramidal Representation of Images and Its Feature Extraction Facility," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 3, pp. 257-264, May 1981.

[jack80] C. L. Jackins and S. L. Tanimoto, "Oct-Trees and Their Use in Representing Three-Dimensional Objects," *Computer Graphics and Image Processing*, vol. 14, pp. 249-270, 1980.

[jone81] L. Jones and S. S. Iyengar, "Representation of A Region As A Forest of Quadtrees," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 57-59, Dallas, Texas, August 3-5, 1981.

[kamb79] Y. Kambayashi, N. Nakatsu, and S. Yajima, "Hierarchical String Pattern Matching Using Dynamic Pattern Matching Machines," in *Proceedings of the IEEE*, pp. 813-818, 1979.

[kana79] L. N. Kanal, "Problem-Solving Models and Search Strategies for Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, April 1979.

[kana81a] L. N. Kanal and V. Kumar, "Parallel Implementations of Structural Analysis Algorithms," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 452-458, Dallas, Texas, 1981.

[kana81b] L. N. Kanal and V. Kumar, "Branch & Bound Formulation for Sequential and Parallel Game Tree Searching: Preliminary Results," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, vol. IJCAI-81, pp. 569-571, Vancouver, B. C., Canada, August 1981.

[kaul83] M. Kaul and E. Koblenz, "Parsing of Graphs in Linear Time," in *Lecture Notes in Computer Science 153 : Graph-Grammars and Their Application to Computer Science*, Second International Workshop, Ehrig, Nagl, and Rozenberg (Eds.), 1983.

[kell71] M. D. Kelly, "Edge Detection by Computer Using Planning," *Machine Intelligence*, vol. 6, Edinburgh University Press, Edinburgh, 1971.

[khan84] N. Khan and R. Jain, "Matching an Imprecise Object Description with Models in a Knowledge Base," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 1131-1134, Montreal, Canada, July 30 - August 2, 1984.

[klin73] A. Klinger, "Data Structures and Pattern Recognition," in *Proceedings of the First International Joint Conference on Pattern Recognition*, pp. 497-498, Wash., D. C., Oct. 30- Nov. 1, 1973.

[klin76] A. Klinger and C. R. Dyer, "Experiments on Picture Representations Using Regular Decomposition," *Computer Graphics and Image Processing*, vol. 5, no. 1, pp. 68-105, Mar. 1976.

[klin79] A. Klinger and M. L. Rhodes, "Organization and Access of Image Data by Areas," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 1, pp. 50-60, 1979.

[krus81] B. Kruse and L. E. Nordell, "Microstructure Representation Through Neighborhood Relations," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, IEEE, Dallas, Texas, 1981.

[leeh83] H. C. Lee and K. S. Fu, "3D Shape from Contour and Selective Confirmation," *Computer Vision, Graphics, and Image Processing*, vol. 22, pp. 177-193, 1983.

[leej81] J. S. Lee, "Speckle Analysis and Smoothing of Synthetic Aperture Radar Images," *Computer Graphics and Image Processing*, vol. 17, pp. 24-32, 1981.

[leej83] J. S. Lee, "A Simple Speckle Smoothing Algorithm for SAR Images," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 1, pp. 85-89, January/February 1983.

[leve66] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics-Doklady*, vol. 10, no. 8, February 1966.

[levi78] M. D. Levine, "A Knowledge-Based Computer Vision System," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (Eds.), Academic Press, NY, 1978.

[levi80] M. D. Levine, "Region Analysis Using a Pyramid Data Structure," in *Structured Computer Vision*, S. Tanimoto and A. Klinger (Eds.), Academic Press, 1980.

[levi82] M. D. Levine and A. M. Nazif, "An Experimental Rule-Based System for Testing Low Level Segmentation Strategies," in *Multicomputers and Image Processing: Algorithms and Programs*, K. Preston and L. Uhr (Eds.), Academic Press, New York, 1982.

[limi82] M. Li, W. I. Grosky, and R. Jain, "Normalized Quadtrees With Respect To Translations," *Computer Vision, Graphics, and Image Processing*, vol. 18, pp. 37-57, 1982.

[lova79] L. Lovasz, "ON the Shannon Capacity of A Graph," *IEEE Transactions on Information Theory*, vol. IT-25, no. 1, pp. 1-7, January 1979.

[luca84] D. Lucas and L. Gibson, "Image Pyramids and Partitions," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 230-233, Montreal, Canada, July 30 - August 2, 1984.

[lusy78a] S. Y. Lu and K. S. Fu, "A Sentence-to-Sentence Clustering Procedure for Pattern Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-8, no. 5, May 1978.

[lusy78b] S. Y. Lu and K. S. Fu, "A Syntactic Approach to Texture Analysis," *Computer Graphics and Image Processing*, vol. 7, pp. 303-330, 1978.

[lusy79] S. Y. Lu, "A Tree-to-Tree Distance and Its Application to Cluster Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, April 1979.

[lusy82] S. Y. Lu, "A String-to-String Correlation Algorithm for Image Skeletonization," in *Proceedings of the International Conference on Pattern Recognition*, pp. 178-180, 1982.

[mark80] P. Marks, "Low-Level Vision Using an Array Processor," *Computer Graphics and Image Processing*, vol. 14, pp. 281-292, 1980.

[mart83] W. N. Martin and J. K. Aggarwal, "Volumetric Descriptions of Objects from Multiple Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 2, pp. 150-158, March 1983.

[mase80] W. J. Masek and M. S. Paterson, "A Faster Algorithm Computing String edit Distances," *Journal of Computer and System Sciences*, vol. 20, pp. 18-31, 1980.

[mats84] T. Matsuyama, "Knowledge Organization and Control Structure in Image Understanding," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 1118-1127, Montreal, Canada, July 30 - August 2, 1984.

[meie83] A. Meier, "A Graph-Relational Approach to Geographic Databases," in *Lecture Notes in Computer Science 153 : Graph-Grammars and Their Application to Computer Science*, pp. 245-254, Second International Workshop, Ehrig, Nagl, and Rozenberg (Eds.), 1983.

[meri84] A. Merigot, B. Zavidovique, P. Garda, and F. Devos, "Pyramidal Algorithms for Image Processing," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 828-832, Montreal, Canada, July 30 - August 2, 1984.

[mill84] R. Miller and Q. F. Stout, "The Pyramid Computer for Image Processing," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 240-242, Montreal, Canada, July 30 - August 2, 1984.

[moor81] R. Moore, "A Multilevel Approach to Pattern Processing," *Pattern Recognition*, vol. 14, no. 1-6, pp. 261-265, 1981.

[naga81] M. Nagata, "Image Processing for Boundary Extraction of Remotely Sensed Data," *Pattern Recognition*, vol. 14, no. 1-6, pp. 275-282, 1981.

[nagl83] M. Nagl, "Bibliography on Graph Rewriting Systems (Graph-Grammars)," in *Lecture Notes in Computer Science 153 : Graph-Grammars and Their Application to Computer Science*, Second International Workshop, Ehrig, Nagl, and Rozenberg (Eds.), 1983.

[nara82] K. A. Narayanan, D. P. O'Leary, and A. Rosenfeld, "Multiresolution Relaxation," *Pattern Recognition*, vol. 15, 1982.

[nare81] P. M. Narendra and M. Goldberg, "A Graph-Theoretic Approach to Image Segmentation," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, vol. IJCAI-81, pp. 248-256, Vancouver, B. C., Canada, August 1981.

[nazi84] A. M. Nazif and M. D. Levine, "Low Level Image Segmentation: An Expert System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 5, pp. 555-577, September 1984.

[neva82] R. Nevatia and K. E. Price, "Locating Structures in Aerial Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 5, pp. 476-484, September 1982.

[nevi82] A. J. Nevins, "Region Extraction from Complex Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 5, pp. 500-511, September 1982.

[nils80] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Company,, Palo Alto, CA., 1980.

[numa84] M. Numao and M. Ishizuka, "A Frame-Like Knowledge Representation System for Computer Vision," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 1128-1130, Montreal, Canada, July 30 - August 2, 1984.

[orou81] J. O'Rourke, "Dynamically Quantized Spaces for Focusing the Hough Transform," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. IJCAI-81, pp. 737-739, 1981.

[pals83] S. K. Pal, R. A. King, and A. A. Hashim, "Image Description and Primitive Extraction Using Fuzzy Sets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 1, pp. 94-100, January/February 1983.

[papa82] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[pavl72] T. Pavlidis, "Representation of Figures by Labeled Graphs," *Pattern Recognition*, vol. 4, pp. 5-17, Pergamon Press, 1972.

[pavl77] T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, New York, 1977.

[pavl78] T. Pavlidis, "Survey: A Review of Algorithms for Shape Analysis," *Computer Graphics and Image Processing*, vol. 7, pp. 243-258, 1978.

[pavl81] T. Pavlidis, "A Flexible Parallel Thinning Algorithm," in *Proceedings of the International Conference on Pattern Recognition and Image Processing, PRIP-81*, pp. 162-167, IEEE, 1981.

[pele80] S. Peleg, "A New Probabilistic Relaxation Scheme," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 4, pp. 362-369, July 1980.

[perk78] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," *IEEE Transactions on Computers*, vol. C-27, no. 2, pp. 126-143, February 1978.

[perk80] W. A. Perkins, "Simplified Model-Based Part Locator," in *Proceedings of the Fifth International Joint Conference on Pattern Recognition*, pp. 260-263, Miami Beach, FL, December 1-4, 1980.

[pers77] E. Personn and K. S. Fu, "Shape Discrimination Using Fourier Descriptors," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 3, pp. 170-179, March 1977.

[piet81] M. Pietikainen and A. Rosenfeld, "Image Segmentation by Texture Using Pyramid Node Linking," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-11, pp. 822-825, 1981.

[piet82a] M. Pietikainen and A. Rosenfeld, "Gray Level Pyramid Linking as an Aid in Texture Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, 1982.

[piet82b] M. Pietikainen, A. Rosenfeld, and I. Walter, "Split-and-Link Algorithms for Image Segmentation," *Pattern Recognition*, vol. 15, 1982.

[pres79] K. Preston, Jr., M. J. B. Duff, S. Levialdi, P. E. Norgren, and J. I. Toriwaki, "Basics of cellular Logic with Some Applications in Medical Image Processing," *Proceedings of the IEEE*, vol. 67, no. 5, May 1979.

[pyst78a] A. Pyster and H. W. Buttelmann, "Semantic-Syntax-Directed Translation," *Information and Control*, vol. 36, pp. 320-361, 1978.

[pyst78b] A. Pyster, "Context-Dependent Tree Automata," *Information and Control*, vol. 38, pp. 81-102, 1978.

[radi84] B. Radig, "Image Sequence Analysis Using Relational Structures," *pattern Recognition*, vol. 17, no. 1, pp. 161-167, 1984.

[rana82] S. Ranade, A. Rosenfeld, and H. Samet, "Shape Approximation Using Quadtrees," *Pattern Recognition*, vol. 15, no. 1, pp. 31-40, 1982.

[read77] R. C. Read and D. G. Corneil, "The Graph Isomorphism Disease," *Journal of Graph Theory*, vol. 1, pp. 339-363, 1977.

[rose76] A. Rosenfeld, R. A. Humel, and S. W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 6, pp. 420-433, June 1976.

[rose79a] A. Rosenfeld, *Picture Languages*, Academic Press, 1979.

[rose79b] A. Rosenfeld and L. Davis, "Image Segmentation and Image Models," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 764-772, May 1979.

[rose80] A. Rosenfeld, "Quadtree and Pyramids for Pattern Recognition and Image Processing," in *Proceedings of the Fifth International Joint Conference on Pattern Recognition*, pp. 802-811, Miami Beach, FL, December 1-4, 1980.

[rose81a] A. Rosenfeld and A. Y. Wu, "Reconfigurable Cellular Computers," in *Proceedings of the IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 217-224, Hot Spring, VA, November 11-13, 1981.

[rose81b] A. Rosenfeld, "Cellular Architectures: From Automata to Hardware," TR-1048, Computer Science Technical Report Series, University of Maryland, College Park, May 1981.

[rose82] A. Rosenfeld and A. Y. Wu, "Parallel Computers for Region-Level Image Processing," *Pattern recognition*, vol. 15, no. 1, pp. 41-50, 1982.

[rose83] A. Rosenfeld, "Quadtrees and Pyramids: Hierarchical Representation of Images," in *Pictorial Data Analysis*, pp. 29-42, R. M. Haralick (Ed.), Springer-Verlag, Berlin Heidelberg, 1983.

[rose84] A. Rosenfeld, "Image Analysis: Problems, Progress, and Prospects," *pattern Recognition*, vol. 17, no. 1, pp. 3-12, 1984.

[rumm84] P. Rummel and W. Beutel, "Workpiece Recognition and Inspection by a Model-Based Scene Analysis System," *Pattern Recognition*, vol. 17, no. 1, pp. 141-148, 1984.

[same80a] H. Samet and A. Rosenfeld, "Quadtree Representations of Binary Images," in *Proceedings of the Fifth International Joint Conference on Pattern Recognition*, pp. 815-818, Miami Beach, FL, December 1-4, 1980.

[same80b] H. Samet, "Note: Region Representation, Quadtrees from Binary Arrays," *Computer Graphics and Image Processing*, vol. 13, pp. 88-93, 1980.

[same80c] H. Samet, "Region Representation: Quadtrees from Boundary Codes," *Communications of the ACM*, vol. 23, pp. 163-170, 1980.

[same82a] H. Samet, "Neighbor Finding Techniques for Images Represented by Quadtrees," *Computer Graphics and Image Processing*, vol. 18, pp. 37-57, 1982.

[same82b]H. Samet, "A Distance Transform for Images Represented by Quadtrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 3, May 1982.

[sanf81] A. Sanfeliu, "An Application of a Distance Measure Between Graphs to the Analysis of Muscle Tissue Patterns," TR-EE, 81-15, Purdue University, May 1981.

[sanf83a] A. Sanfeliu and K. S. Fu, "Distance between Attributed Relational Graphs for Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 3, May/June 1983.

[sanf83b] A. Sanfeliu and K. S. Fu, "Tree-Graph Grammars for Pattern Recognition," in *Lecture Notes in Computer Science 153 : Graph-Grammars and Their Application to Computer Science*, pp. 349-367, Second International Workshop, Ehrig, Nagl, and Rozenberg (Eds.), 1983.

[sanf84] A. Sanfeliu, "A Distance Measure Based on Tree-Graph-Grammars: a Way of Recognizing Hidden and Deformed 3D Complex Objects," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 739-741, Montreal, Canada, July 30 - August 2, 1984.

[shap79] L. G. Shapiro and R. M. Haralick, "Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 1, pp. 10-20, January 1979.

[shap80] L. Shapiro, "A Structural Model of Shape," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 2, March 1980.

[shap81] 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, pp. 504-519, September 1981.

[shap83] L. G. Shapiro, "Computer Vision Systems: Past, Present, and Future," in *Pictorial Data Analysis*, pp. 199-237, R. M. Haralick (Ed.), Springer-Verlag, Berlin Heidelberg, 1983.

[shiq82] Q. Y. Shi and K. S. Fu, "Efficient Error-Correcting Parsing for (Attributed and Stochastic) Tree Grammars," *Information Science*, vol. 26, pp. 159-188, 1982.

[shiq83] Q. Y. Shi and K. S. Fu, "Parsing and Translation of (Attributed) Expansive Graph Languages For Scene Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 5, September, 1983.

[shle76] M. I. Shlezinger, "Syntactic Analysis of Two-Dimensional Visual Signals In the Presence of Noise," *Cybernetics*, vol. 12, no. 4, 1976.

[shne79] M. Shneir, "Using Pyramids to Define Local Thresholds for Blob Detection," TR-808, Computer Vision Lab., University of Maryland, College Park, MD, Sep. 1979.

[shne80] M. Shneir, "Extracting Linear Feature from Images Using Pyramids," TR-855, Computer Vision Lab., University of Maryland, College Park, MD, Jan. 1980.

[shne81a] M. Shneier, "Path-Length Distances for Quadtrees," *Information Sciences*, vol. 23, pp. 49-67, 1981.

[shne81b] M. Shneier, "Calculations of Geometric Properties Using Quadtrees," *Computer Graphics and Image Processing*, vol. 16, pp. 296-302, 1981.

[shne81c] M. Shneier, "Two Hierarchical Linear Feature Representations: Edge Pyramids and Edge Quadtrees," *Computer Graphics and Image Processing*, vol. 17, pp. 211-224, 1981.

[slat80] P. B. Slater, "Preference Trees Based upon Hierarchical Regionalizations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 12, December 1980.

[sloa81] K. R. Sloan Jr, "Dynamically Quantized Pyramids," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI-81*, pp. 714-736, 1981.

[ster81] S. R. Sternberg, "Architectures for Neighborhood Processing," in *Proceedings of the International Conference on Pattern Recognition and Image Processing, PRIP-81*, pp. 374-380, IEEE, 1981.

[taij80] J. W. Tai, "Attributed Parallel Tree Grammars and Automata for Syntactic Pattern Recognition," in *Proceedings of the Fifth International Joint Conference on Pattern Recognition*, pp. 1001-1003, Miami Beach, FL, December 1-4, 1980.

[tamu84] H. Tamura and N. Yokoya, "Image Database Systems: A Survey," *Pattern Recognition*, vol. 17, no. 1, pp. 29-43, 1984.

[tana76] E. Tanaka and T. Kasai, "Synchronization and Substitution Error-Correcting Codes for the Levenshtein Metric," *IEEE Transactions on Information Theory*, vol. IT-22, no. 2, March 1976.

[tana77] J. Tanaka, *A Metric on Graphs and its Applications*, IEEE Japan IP-77-75, Oct. 31, 1977.

[tang80] G. Y. Tang and T. S. Huang, "Using The Creation Machine to Locate Airplanes on Aerial Photos," *Pattern Recognition*, vol. 12, pp. 413-442, 1980.

[tani75] S. Tanimoto and T. Pavlidis, "A Hierarchical Data Structure for Picture Processing," *Computer Graphics and Image Processing*, vol. 4, pp. 104-119, 1975.

[tani76] S. L. Tanimoto, "Pictorial Feature Distortion in a Pyramid," *Computer Graphics and Image Processing*, vol. 5, pp. 333-352, 1976.

[tani77] S. L. Tanimoto and T. Pavlidis, "The Editing of Picture Segmentation Using Local Analysis of Graphs," *Communications of the ACM*, vol. 20, no. 4, pp. 223-229, 1977.

[tani78] S. L. Tanimoto, "Regular Hierarchical Image and Processing Structures in Machine Vision," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (Eds.), Academic Press, 1978.

[tani80] S. Tanimoto and A. Klinger, (Eds.), *Structured Computer Vision: Machine Perception through Hierarchical Computation Structures*, Academic Press, 1980.

[tani84] S. L. Tanimoto, J. P. Crettez, and J. C. Simon, "Alternative Hierarchies for Cellular Logic," in *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 236-239, Montreal, Canada, July 30 - August 2, 1984.

[thom76] R. A. Thompson, "Language Correction Using Probabilistic Grammars," *IEEE Transactions on Computers*, vol. C-25, no. 3, March 1976.

[thor83] C. Thorpe and S. Shafer, "Correspondence in Line Drawing of Multiple Views of Objects," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[trop83] H. Tropf and I. Walter, "An ATN Model for 3-D Recognition of Solids in Single Images," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI-83*, August 8-12, 1983.

[touj74] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, Mass., 1974.

[tsai79a] W. H. Tsai and K. S. Fu, "A Pattern Deformational Model and Bayes Error-Correcting Recognition System," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-9, no. 12, pp. 745-756, December 1979.

[tsai79b] W. H. Tsai and K. S. Fu, "Error-Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-9, no. 12, pp. 757-768, December 1979.

[tsai80a] W. H. Tsai and K. S. Fu, "Attributed Grammar - A Tool for Combining Syntactic and Statistical Approaches to Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 12, pp. 873-885, December 1980.

[tsai80b] W. H. Tsai and K. S. Fu, "A Syntactic-Statistical Approach to Recognition of Industrial Objects," in *Proceedings of the Fifth International Joint Conference on Pattern Recognition*, pp. 251-259, Miami Beach, FL, December 1-4, 1980.

[tsai83] W. H. Tsai and K. S. Fu, "Subgraph Error-Correcting Isomorphism for Syntactic Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 1, pp. 48-67, January/February 1983.

[tsao81] Y. F. Tsao and K. S. Fu, "Parallel Thinning Operations for Digital Binary Images," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 150-155, IEEE, 1981.

[tsot82] J. K. Tsotsos, "Knowledge of the Visual Process: Content, Form, and Use," in *Proceedings of the 6th. International Joint Conference on Pattern Recognition*, pp. 654-669, Munich, Germany, October 19-22, 1982.

[uhrl72] L. Uhr, "Layered "Recognition Cones" Networks That Process, Classify, and Describe," *IEEE Transactions on Computers*, vol. 21, pp. 758-768, 1972.

[uhrl76] L. Uhr, "Recognition Cones: that Perceive and Describe Scenes that Moves and Change over Time," in *Proceedings of the Third International Joint Conference on Pattern Recognition*, pp. 287-293, 1976.

[uhrl78] L. Uhr, "Recognition Cones, and Some Test Results; The Imminent Arrival of Well-Structured Parallel-Serial Computers; Positions, and Positions on Positions," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (Eds.), Academic Press, 1978.

[ullm76] J. R. Ullman, "An Algorithm for Subgraph Isomorphism," *Journal of the ACM*, vol. 23, no. 1, pp. 31-42, January 1976.

[vaid82] P. D. Vaidya, L. G. Shapiro, R. M. Haralick, and G. J. Minden, "Design and Architectural Implications of a Spatial Information System," *IEEE Transactions on Computers*, vol. C-31, no. 10, pp. 1025-1031, October 1982.

[vigr78] P. D. Vigra and C. Ghezzi, "Context-free Graph Grammars," *Information and Control*, vol. 37, no. 2, May 1978.

[voel77] H. B. Voelcker and A. A. G. Requicha, "Geometric Modeling of Mechanical Parts and Processes," *Computer*, vol. 10, no. 12, pp. 48-57, December 1977.

[wagn74] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, pp. 168-173, January 1974.

[wahb83] B. W. Wah, *Class Notes for Course No. EE-695D*, School of Electrical Engineering, Purdue University, Spring 1983.

[wins80] P. H. Winston, "Learning and Reasoning by Analogy," *Communications of the ACM*, vol. 23, no. 12, pp. 689-703, December 1980.

[wong80] A. K. C. Wong and D. E. Ghahraman, "Random Graphs: Structural-Contextual Dichotomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 4, pp. 341-348, July 1980.

[wuay77] A. Wu, "Cellular Graph Acceptors, 2," TR-621, Computer Science Technical Report Series, University of Maryland, College Park, December 1977.

[wuay79a] A. Wu and A. Rosenfeld, "Cellular Graph Automata. I. Basic Concepts, Graph Property Measurement, Closure Properties," *Information and Control*, vol. 42, pp. 305-329, 1979.

[wuay79b] A. Wu and A. Rosenfeld, "Cellular Graph Automata. II. Graph and Subgraph Isomorphism, Graph Structure Recognition," *Information and Control*, vol. 42, pp. 330-353, 1979.

[wuay80] A. Wu and A. Rosenfeld, "Sequential and Cellular Graph Automata," *Information Sciences*, vol. 20, pp. 57-68, 1980.

[wuay81] A. Y. Wu, T. Dubitzki, and A. Rosenfeld, "Parallel Computation of Contour Properties," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 3, May 1981.

[yaum81] M. M. Yau and S. N. Srihari, "Recursive Generation of Hierarchical Data Structures for Multidimensional Digital Images," in *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pp. 42-44, IEEE, 1981.

[youk79] 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, pp. 334-345, June 1979.

[youk80] K. C. You and K. S. Fu, "Distorted Shape Recognition Using Attributed Grammars and Error-Correcting Techniques," *Computer Graphics and Image Processing*, vol. 13, pp. 1-16, 1980.

[zuck76] S. W. Zucker, "Relaxation Labeling and the Reduction of Local Ambiguities," in *Proceedings of the Third International Joint Conference on Pattern Recognition*, San Diego, Nov. 1976.

[zuck77] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An Application of Relaxation labeling to Line and Curve Enhancement," *IEEE Transactions on Computer*, vol. C-26, pp. 394-403, and 922-929, 1977.

[zuck78] S. W. Zucker, E. V. Krishnamurthy, and R. L. Haar, "Relaxation Processes for Scene Labeling: Convergence, Speed, and Stability," *IEEE Transactions on Systems, Man, Cybernetics*, vol. SMC-8, no. 1, pp. 41-48, January 1978.