On the Learning of Concepts from Visual Examples

視覚的な例に基づく概念の学習

by

Andreas Held

Dissertation

Presented to the Faculty of the

Graduate School of Electronic Science and Technology

Shizuoka University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Graduate School of Electronic Science and Technology

Shizuoka University

January, 1995

Alle unsere Erkenntnis hebt von den Sinnen an, geht von da zum Verstande, und endigt bei der Vernunft, über welche nichts Höheres in uns angetroffen wird, den Stoff der Anschauungen zu bearbeiten und unter die höchste Einheit des Denkens zu bringen.

Immanuel Kant, Kritik der reinen Vernunft

TO MY PARENTS AND MEI-YIU

Acknowledgments

The research reported in this thesis was carried out during the academic years 1992 - 1994, at the Graduate School of Science and Technology, Shizuoka University, Japan. I am deeply indebted to the Japanese Ministry of Science and Education, whose generous scholarship I received during those years made this research possible and further enabled me to gain some unique experiences in a culture so different from mine.

I wish to express my gratitude to my supervisor, Prof. Keiichi Abe, for his encouragement and guidance during the course of my thesis. From the first time I met Prof. Abe up to the present I have been impressed by his broad knowledge, encompassing most areas of modern computer science. During many discussions I came to appreciate the support and help he gave me, without which the present work would not have been possible. I also owe much to Dr. Carlo Arcelli, Istituto di Cibernetica, Italy, for his support and encouragement during my first research endeavours, to Prof. Hiromasa Nakatani, Shizuoka University, for many stimulating and sometimes relaxing discussions, and to Prof. Takahira Yamaguchi, Shizuoka University, for advice and suggestions concerning the field of machine learning. Special thanks go to Prof. Yasuhiro Takeuchi, Prof. Yoshifumi Shimodaira, and Prof. Atsuyuki Suzuki for their helpful comments that were greatly appreciated in the preparation of this thesis.

I would also like to thank my fellow students for their help on many occasions. Especially I would like to thank Caihua Wang for many stimulating discussions, Takahiro Sugiyama for his much appreciated technical support, and Jia Jin, who helped in the preparation of Appendix B. Furthermore, I would like to thank the staff of the Department of Computer Science, Shizuoka University, for their help and support on many occasions.

Last but not least, my thanks go to my parents, Elisabeth and Hans Held, for their continuing help and support, and to my wife Mei-Yiu for her patience and understanding during the sometimes long hours I spent at university.

[illegible]

Andreas Held

Graduate School of Electronic
Science and Technology
Shizuoka University
January 1995

Abstract

The learning of concepts from visual input is a field that draws from the areas of both, computer vision and artificial intelligence. The problems encountered in designing such a system are, therefore, manifold. In the present work, some of those problems are addressed, and a system that learns simple model descriptions from sequences of binary images is proposed. The system consists of several independent modules that are grouped together. As the aim is to develop a system that does not rely on any supervision, the input is provided in the form of images, either acquired by a scanner or any other reading device. Four major modules can be distinguished: they are labelling, description, interpretation, and explanation.

Based on some visual input, which has been restricted to binary images, the labelling stage attempts to extract salient components of the input object. Salient components are assumed to be components that give a concise but meaningful description of some object, meaningful in an intuitive sense. Since human intuition cannot generally be captured by a machine, the concept of maximal approximately convex subparts (MACS) is proposed and it is shown that this notion can represent some of the semantics of an object, solely based on its geometrical structure. The decomposition into MACS is then used to derive a description of the relations between MACS. This description is in the form of a directed graph, called relational network, whose links qualitatively describe relations. To obtain such qualitative relations, it is proposed to generalize commonly used predicate relations, as, for instance, used in semantic networks, into so-called generalized predicates. Generalized predicates can be viewed as the "fuzzification" of usual predicates, and they qualitatively describe relations among MACS. Each generalized predicate consists of a sum of scaled Gaussian functions; through generalization, this sum of Gaussians will be adjusted to mirror more general relations and thus provide more general class descriptions.

Once a description of an instance has been obtained, it can be matched with previously acquired concepts or object class descriptions. Since the basic structure of an instance or a concept is that of a graph, this matching corresponds to the subgraph isomorphism problem, which is known to be NP-complete. The feasibility of the matching can be restored, however, by employing a heuristically augmented state-space search. Provided a satisfactory match is found, the new instance and the concept are generalized to mirror the enlarged class. Generalization is carried out in two steps: generalization of the structure of the concept, and generalization of the links and the generalized predicates. Generalizing the structure of the concept is straightforward; however, the relational network is now transformed into a conceptual network. The difference between the relational network and the conceptual network is that the latter has a third dimension along which concept discrimination is carried out. Generalization of the predicates, on the other hand, has to ensure that the definition of the generalized predicates is closed under the used generalization operator. In the present system, an adopted version of MYCIN's generalization operator has been used. As the steps of matching and generalizing correspond to identifying an instance with some recorded descriptions, an interpretation of the input image becomes possible.

Finally, the results obtained from generalization have to be explained. As this explanation should be easily understood by a human operator, a description in near natural language is extracted. Such a description is obtained by first transforming the conceptual network into a simple semantic network. Based on the extracted semantic network, it is then relatively easy to produce near natural language, based on a simple set of production rules.

As stated above, the main task of the system is the acquisition of concepts or of model structures. Using the same mechanisms, however, it is as well possible to recognize new instances based on some concept database. Recognition is not complete, though, as it is only possible to recognize or to describe an instance in terms of the recorded concepts. Hence, the approach might be called recognition by experience. On the other hand, even if it is not possible to completely recognize a new instance, partial recognition based on substructures of the recorded concepts might be possible. Finally, employing the hierarchical abstraction

of concepts that becomes possible in conceptual networks, certain clusterings of instances can be observed. Those clusterings tend to identify common substructures among several instances. It is those substructures or partial concepts that might give valuable hints during recognition.

Contents

CHAPTER 1

Introduction

1.1 Context

Nowadays it is usually taken for granted that empirical knowledge is dependent on, or derived from, perception. Although the role of perception as a basic ability for surviving in our world has never been questioned, this was not always so in the case of learning or acquiring knowledge, as for instance the Greek philosopher Pythagoras taught that thought is superior to sense and intuition is superior to observation. The Pythagorean ideal was the mathematician whose knowledge was supposed to be certain and exact; moreover, it was obtained by mere thinking, without the need of observation. Socrates went even further to say that the body, and thus the senses, is a hindrance in the acquisition of knowledge; the logical consequence of which is that all knowledge is already existing within us, we just have to recollect it [46]. In Greek mathematics as well, the reasoning was deductively from what appeared self-evident, not inductively from what had been observed. Induction did not play any prominent role until the middle ages; for instance, Copernicus based his heliocentric system, which was subsequently refined by Kepler, on observations of the actual movements of the planets. The first detailed description of mathematical induction can be found in Pascal's work [8]. Further successes of induction became apparent in the work by Galileo and Newton later. A somewhat intermediate position was taken by Kant, who, in his "The Critique of Pure Reason," attempted to prove that, although none of our knowledge can transcend experience, it is, nevertheless in part *a priori* and not inferred inductively from experience. However, Kant reasserted the importance of perception or induction by saying that all human knowledge begins with perception, proceeds to understanding, and ends with reason. More recently, the problem of scientific discovery has attracted more attention. For instance, Coleridge wrote that scientific reasoning is the faculty of concluding universal and necessary truths from particular and contingent appearances. On the other hand, Russell and Popper maintained that there is no such thing as the scientific method, and scientists do not make their discoveries by induction or by any other method [34].

Whether or not we are using induction to acquire knowledge, doubtless, Kant's remark that all human knowledge starts with perception is a strong incentive for learning from observation. Observation or perception is so prevalent in our daily lives that very often we are not able to fully appreciate its implications. Learning from observation, as, for instance, professed by Kepler or Newton, proceeds by collecting observations or measurements and then trying to find laws that govern the observations or features that are in common among the observations. Finding laws that could represent a certain set of observations does not only require those observations, but as well a great deal of domain specific knowledge, which we usually acquire during many years in school. The other approach, namely, finding what is common among the observed measurements can be considered a sort of abstraction from data. It has been said that all human knowledge is empirical in nature, hence, knowledge acquisition might have proceeded as abstraction from observation. If we attempt to equip a machine with the ability to learn then we face the same two possibilities, namely, learning using a priori knowledge and learning as abstraction. The former usually has been characterized as *explanation based learning* (EBL), whereas the latter one has been characterized as *similarity based learning* (SBL). Explanation based learning heavily relies on the available domain knowledge and could even be criticized for "answer begging," in that the formulation of the domain knowledge will heavily influence the obtainable results. It appears, therefore, that similarity based learning is more suited to the present state of machine learning, as long as there are no large and general knowledge bases available.

Here I am proposing an approach to similarity based learning that should be more general and much closer to human learning behaviour than previous approaches. The approach is based on the idea of abstraction from visual examples, which are given in the form of images. Hence, the proposed system is a model for the whole process starting from perception and leading to the forming of concepts. However, I do not claim the system to be a model for human knowledge acquisition. On the contrary, it is presumably quite far from human performance, and should rather be considered as an approach closely adapted to the requirements and constraints of machine learning. Any system covering the whole range from perception to concept formation necessarily has to address problems in many different areas. Those problems range from image processing problems, to problems in knowledge representation and generalization. Due to the wide range of the encountered problems, it was not possible to treat them all in the same depth. While some have been solved quite satisfactorily, others were only touched upon, without the possibility of any thorough treatment. In this respect, I hope that the present work might give some hints as to where further research is most needed.

1.2 Historical Overview of Other Research

A mile stones of early research in machine learning is definitely Winston's work [61]. I will try to give a more detailed review of his work in Section 2.2. After Winston, who based his work on visual input in the form of simple line drawings, the field began to separate into symbolic approaches, that is, methods that work on a set of symbols, and numerical approaches, directly working with raw data. Of the two, the former one, namely symbolic learning or reasoning has been prevalent for a long time. In a different direction, a distinction between similarity based methods, attempting to derive a generalization from many examples by analyzing their similarities and differences, and explanation based methods, a knowledge intensive method of examining single examples to derive generalizations based on underlying causal models, becomes possible. It has sometimes been said that the explanation based methods are theoretically better justified and that there is no need to look for similarities across examples. However, as Lebowitz [32] pointed out, it may not always be possible to find a causal explanation, and similarity often implies causality. The objective of the present work is given in the context of similarity extraction from raw data or images, respectively.

We can look briefly at some work done in the area of similarity extraction, after Winston's work, in a chronological order. Haar [19] addresses the problem of the conversion of spatial information into symbolic descriptions. Haar's work did have some influence on the present work, and I will come back to this where appropriate. The actual problem tackled by Haar is the so-called layout problem. That means, for instance, given a room, a set of furniture, and a set of constraints guiding the relation among individual pieces of furniture, the task is to find the best possible placement for the furniture. Haar's main contribution is to introduce some sort of fuzzy concept in order to obtain an adaptive conversion of spatial information into a symbolic description.

Kodratoff and Lemerle-Loisel [31] propose to utilize an optimized recognition tree for the generalization of childlike line drawings and chemical molecules. Kodratoff and Lemerle-Loisel use quantified operators expressing spatial relationships among strokes extracted from an example. A near-miss exists whenever two relations differ only slightly. Each near-miss can be considered as introducing a partition into the set of all relations; near-misses can therefore be represented as the nodes of a tree, whose descendants correspond to the induced partition. Hence, the recognition tree becomes a decision-tree with which new instances can be recognized. The weakest point of Kodratoff and Lemerle-Loisel's work is probably the highly artificial input domain they used. It is questionable whether the approach would not fail in using real examples, due to problems in extracting unambiguous strokes.

Somewhat closer to Winston's work is the approach by Connell and Brady [11]. The input to this system is in the form of two-dimensional shapes, such as tools, planes, or others. Based on Asada and Brady's smoothed local symmetries, a semantic network description of the shapes is initially built. By a clever transformation, the semantic network representation is then transformed into Gray-code, in which semantic distance corresponds to syntactic or Hamming distance. Generalization proceeds by an operation called *ablation*. The idea behind ablation is that if two things belong to the same class then the differences between them must be irrelevant, hence these differences can be removed. In case that ablation would generate an over generalized concept, a disjunctive concept consisting of the initial concept and the new instance is generated instead. Ablation works because the syntax of the representation reflects its semantics in a very simple way. Stress is therefore placed on an ingenious transformation of the semantic network into Gray-code. Connell and Brady's system stresses the concept of form. That is, anything having a flat surface and a part that can be grasped will be considered semantically close to a hammer and could be used for hammering tasks. The system as a whole is very well thought out and appears to be sufficiently general to deal with rather complicated domains.

Segen [49] proposes an approach for the model learning and recognition of non rigid objects. A two-dimensional shape in Segen's approach is represented as a set of local features, such as curvature maxima of the boundary. The local features are then grouped together into a layered graph, showing the interdependencies of the features. A group of shapes whose layered graph representations are not identical can be described by means of a probability model whose outcome is a probabilistic layered graph, where each vertex has associated with it a probability distribution over a set of possible labels. To obtain such a probabilistic graph, the first layered graph is converted into a probabilistic graph by assigning probability values to each vertex. The remaining layered graphs are then used to update the probability graph, leading to the generalized model graph representation. Recognition of new instances can then be done by matching the new instance to the models.

A rather more conventional approach is taken by Dong et al. [12, 13, 14]. Based on the three-dimensional information obtained by trinocular vision, Dong et al. attempt to build a system that learns the structure of objects by abstracting from examples. First, the obtained three-dimensional information is processed so as to obtain a model that describes the surfaces the object is composed of and the lines the surfaces are composed of. The actual model is represented by using a relational description language (RDL), similar to a predicate representation. The main difference between RDL and a predicate representation is that the former allows to represent qualitative information by employing graded predicates, such as small, medium, or large. Operator intervention is then needed to structure the obtained relations; for instance, in the case of learning the concept of a house, the system is taught which surfaces are forming the roof, the walls, and so on. The actual concept refinement takes place by matching the initial concept with an example. Depending on the similarity between the initial concept and the example, the resulting concept description is refined until a prefixed similarity measure is satisfied. The main contributions of this work lie in the fact that real images have been used and in the used relational description language, which allows for a certain flexibility during concept refinement.

Arita et al. [3] propose another system for the acquisition of models from two-dimensional images. The basic idea behind their system is to first segment the image into homogeneous regions. They then introduce a so-called segmentation tree that represents levels of abstraction of the segmentation. The segmentation tree can be obtained by varying the parameters of the segmentation procedure and observing which regions are merged at a certain time. Each node of the segmentation tree carries information on the represented region, such as shape or colour. Generalization of the model proceeds by first finding a match between two segmentation trees. Nodes of the tree that cannot be matched are discarded at that point and the two segmentation trees will have the maximal common structure. Node information is then generalized according to some simple rules. Arita's system is an example of a system that does not rely on a high-level model. Although it appears that such model representations are of advantage in many cases, the problem of how to bridge the gap between the employed model structure and human intuition has not been solved satisfactorily in most cases.

As can be seen from the above descriptions, lately it has become more popular to use model descriptions not based on predicate or semantic networks. I believe, however, that semantic networks are important for the ease of understanding the acquired model. On the other hand, hitherto used semantic network representations appear too rigid and other possibilities of predicate representations should be explored.

1.3 Outline of the Thesis

This thesis is organized as follows. In Chapter 2, I will give a broad overview over the purpose and architecture of the proposed system. To emphasize the background of the present work further, I will also describe two other systems, the one by Winston, because of its historical importance, and one by Ueda and Suzuki, which appears representative for the recent development of vision based model acquisition and recognition schemes. In the next four chapters I will explain in detail the four main components of the proposed system. In Chapter 3, I am going to outline the approach used for finding meaningful decompositions of binary shapes. The chapter is centered on the notions of *approximate convexity*, whose thorough mathematical treatment is given in Appendix B, and *maximal approximately convex subset* MACS. In Chapter 4, I will first explain how to obtain a description of the decomposition into MACS by employing a so-called *relational network*, which is, by itself, a part of the *conceptual network*, used to represent whole concepts or clusters of concepts. The major point of relational networks is the usage of *generalized predicates*, which are designed to represent "fuzzy" relations among segments of the decomposition. In the same chapter, I will also touch the problem of how to match relational networks efficiently. Next, in Chapter 5, the employed procedure for generalizing the model structure is explained. I will show in that chapter that the used generalization scheme does not only bear some resemblances to MYCIN's generalization of certainty factors, but that it has a simple probabilistic interpretation as well. The problem of finding easy to understand explanations of obtained generalizations will be addressed in Chapter 6. That chapter is devoted to the generation of near-natural language explanations based on the transformation of the relational network into a simple kind of semantic network. The performance of the proposed system is illustrated in Chapter 7, where a series of examples is shown. The examples have been chosen so as to give a comprehensive idea of the system. All aspects of the system as outlined in the proceeding chapters are further outlined here by means of some examples. Finally, in Chapter 8, I will summarize the proposed system and try to emphasize the major contributions of this work.

CHAPTER 2

Overview

2.1 Purpose and Architecture of System

2.1.1 Introduction

It is possible to define a general architecture for a system that learns concepts or models from visual examples. Such a general architecture is shown in Fig. 2.1. As can be seen from that figure, a bottom-up and a top-down part can be identified; where they meet, generalization takes part. Based on the generalization results, which actually constitute an interpretation of the scene or image, the refinement of the bottom-up processing becomes possible, which is shown by feedback links. The actual flow of the processing is as follows. First, we have to identify salient components or features in an input image. This step is referred to as labelling or component identification. Based on the found features, a description of the input shape can be derived. Comparing the derived description with previously acquired descriptions gives us an interpretation of the input shape, based on which a generalized description can be obtained. This generalized description will then be stored in the concept data-base for later use. Finally, the system should be able to explain the obtained results in an intuitive and easily understandable way.

The system to be described here is somewhat simpler than the one shown in Fig. 2.1. It could therefore be considered a case-study of a system for learning of concepts. The

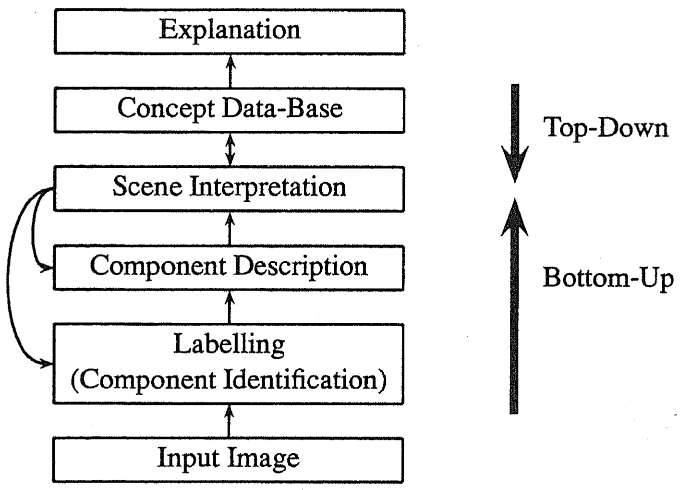


Figure 2.1: A general non-model-based learning system

actual data flow of the system to be described hereafter is shown in Fig. 2.2. What is missing in respect to the general learning system from Fig. 2.1 are the feedback links. That means, at the moment no refinement of the bottom-up results are possible. Furthermore, as will be described later, the input has been restricted to binary images.

Taking into account the above restrictions we can say that the proposed system should be able to acquire and generalize models of simple objects that can be characterized sufficiently by means of their silhouettes. Acquiring a model, as used in the present context, refers to the task of building a model description from a single input image or *instance*. Generalization of a model is achieved by appropriately adopting the model description so as to fit other instances of the same class of objects. Generalizing an instance description yields a concept description, or *concept* for short, which does not refer to a single instance, but to a class of instances. The same system can be used for the recognition of unknown instances. This is achieved by matching the instance against all available models; the model that represents the instance best can be considered as defining the object class the instance belongs to. Strictly speaking, such an approach does only allow for recognition in respect to the acquired model base, that is, in respect to prior knowledge, and does not provide for any exclusion of instances as not belonging to any previously acquired concept. This problem

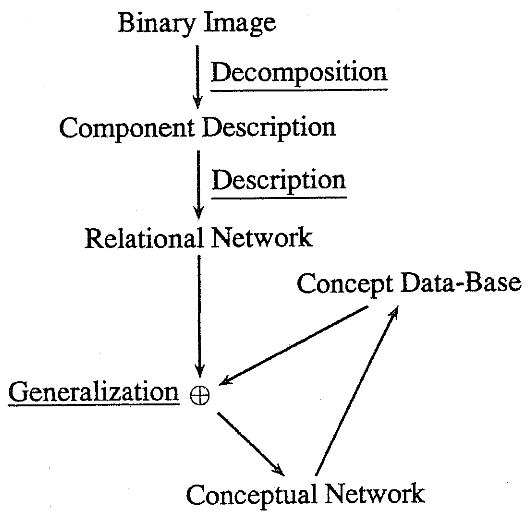


Figure 2.2: Outline of the present system

could be overcome by considering only those instances as recognizable whose similarity to some concept is above a certain threshold. Furthermore, it would be possible to recognize such instances at least partially, by investigating what parts of some concept can be matched to the instance. Hence, a recognition of some constituents of the instance might become possible. This, in turn, might lead to a partial characterization of the instance. Although such an approach appears to be quite straightforward, not much work on the recognition of instances has been done within the framework of the present system.

Let us assume that each concept to be learned or to be recognized can be expressed as a set of primitive building blocks, which are related to each other by some predicates. Conjecturing from human understanding and learning behaviour, it seems reasonable not to duplicate parts of the memory structure that could be used in the recognition of more than one concept. For instance, it is extremely wasteful and not very intuitive to duplicate concepts corresponding to two-engine planes and four-engine planes, as the former can be subsumed by the latter.

What I propose here is a unified framework for model acquisition, concept building and object recognition, based on the same memory structure. The assumption is that the mentioned three tasks are inherently related and intertwined, so they shouldn't artificially be separated and treated differently. The actually proposed memory structure does not claim to be a model of human memory. Quite far from this, it simply appears to be suited for solving the task in question, namely to acquire and recognize instances of simple visual objects by computer.

2.1.2 Structure

I will refer to the structure used to represent all information as *conceptual network.* The conceptual network is a three-dimensional structure, whose z-axis can be viewed as imposing a hierarchy. Three main hierarchical levels can be distinguished. They are from bottom to top: the relational network layer, the partial-concept layer, which can itself consist of multiple layers, and the concept layer. A schematic representation of this structure is given in Fig. 2.3. Depending on the intermediate levels of partial-concepts desired, additional

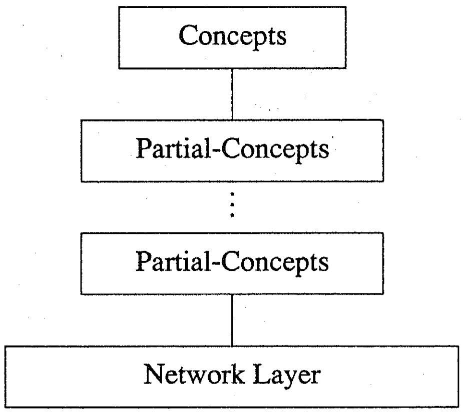


Figure 2.3: Hierarchical structure of conceptual network

layers can be inserted so as to more closely mirror possible multi-level interpretations.

As already mentioned, new instances are presented in the form of binary images, acquired by a scanner. The restriction of using binary images only is rather strict. However, the system is exploratory in nature, and it should be possible to extend it to the domain of gray-scale or range images. The input objects are first segmented into *maximal approximately convex subparts* (MACS) [24] and the relation between those MACS is extracted. Due to the definition of MACS, the actual domain of images that can be processed with this system is further restricted. For instance, only shapes that can be decomposed into a finite union of MACS can be processed. This means, the system cannot deal with circular arcs or the like. Furthermore, shapes that consist of elongated parts will be decomposed better, as such shapes contain more structure to draw information from. The information that can be obtained from the decomposition into MACS will be mirrored in the network layer of the conceptual network. I will refer to this network layer as a *relational network*, as it describes the relation between elements of the decomposition, that is, the relation between MACS. The important point about the relational network is, that its nodes represent MACS, and its links represent relations between elements of the decomposition. Actually, only the relation between elements is mirrored in the relational network; the actual information about the elements on the other hand does not enter explicitly. Furthermore, distinction of concepts is not done at the network level. This means, the network level simply consists of chunks of information, which are not necessarily connected to each other. Each node of the network could actually be viewed as a sensory neuron, although sensing a meaningful entity, namely, a MACS. If a new instance corresponds to some part of the relational network, then the nodes of that network part fire, triggering some nodes in the parental layer. This firing continues to higher levels, until a final concept is chosen. Hence, it is the pattern of matched nodes that allow for the recognition of a new instance. In this respect, the proposed system borrows some ideas from neural networks. For instance, given the three input figures from Fig. 2.4, the relational network as shown in Fig. 2.5 can be derived. In that figure, each link connecting some nodes stands for some relation between elements. For instance, link BC gives the relation between tail-wings and fuselage, link AB stands for the relation between main-wings and fuselage, etc. Although the relational network actually represents three different concepts, namely three different types of planes, only one common chunk

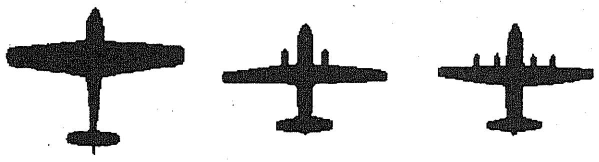


Figure 2.4: Set of input figures

$$\begin{tikzcd}
 & & A \arrow[lld] \arrow[ld] \arrow[d] \arrow[rd] \arrow[rrd] & & \\
F & D & B \arrow[d] & E & G \\
 & & C & &
\end{tikzcd}$$

Figure 2.5: Relational network using inputs from Fig. 2.4

of memory will be used. The distinctions of what nodes belong to what concepts are done on a different level of the conceptual network. Again using the same example, there are three partial concepts, which correspond to the three partial relational networks in Fig. 2.6. Each of those three partial concepts has a node in the partial-concept layer of the conceptual network. That means, nodes A, B, and C are connected to node PLANE1, nodes A, B, C, D, and E are connected to node PLANE2, and nodes A, B, C, D, E, F, and G are connected to node PLANE3. Therefore, the partial concept-layer consists of the three nodes PLANE1, PLANE2, and PLANE3, each of which stands for a valid partial concept. If nodes A, B, and C of the network layer get matched, then all three nodes of the partial-concept layer will receive stimuli. Hence, this conflict must be resolved by communication among the nodes of the partial-concept layer. Finally, all three nodes of the partial-concept layer connect to the node PLANE of the concept layer. The meaning of this is to show that all three partial concepts PLANE1, PLANE2, and PLANE3 are subsumed by the concept PLANE of the concept layer. Therefore, the hierarchical aspect of the conceptual network can be

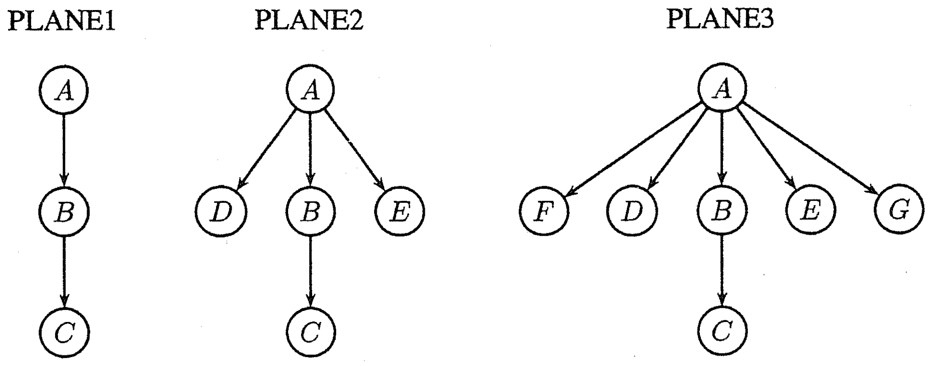


Figure 2.6: Partial concepts

used to implement concept relationships. Furthermore, it can as well be used for denoting building blocks of which a certain concept is made of. The whole conceptual network for the example of the planes in Fig. 2.4 is shown in Fig. 2.7.

As stated before, each matched network node triggers its parental nodes (there can be more than one), sending the match value, that is the confidence in the matching, as a trigger signal. The parental node, in turn sums up the triggering signals or their confidence values respectively. If the summed up signals exceed the threshold of this node, then the node itself fires, in turn triggering its parental nodes. Note that it is important that nodes can inhibit each other. For instance, in our example of the plane, if we match the instance of a plane with two engines against the network in Fig. 2.5, then the nodes PLANE1 and PLANE2 of the partial-concept layer will both be triggered. This can be avoided if the node PLANE2 can inhibit PLANE1. That means, once PLANE2 fires, PLANE1 cannot fire as well, as it is subsumed by PLANE2. Similarly, a general inhibition relation can be defined among the members of any level, except for the network level. The conceptual network will finally respond with a hierarchical ladder of nodes that fired. The top node of this hierarchy states the most general concept to which the instance belongs. Lower levels of the hierarchy give increasingly detailed descriptions, until the lowest level, which shows the generalized relations between elements of the instance.

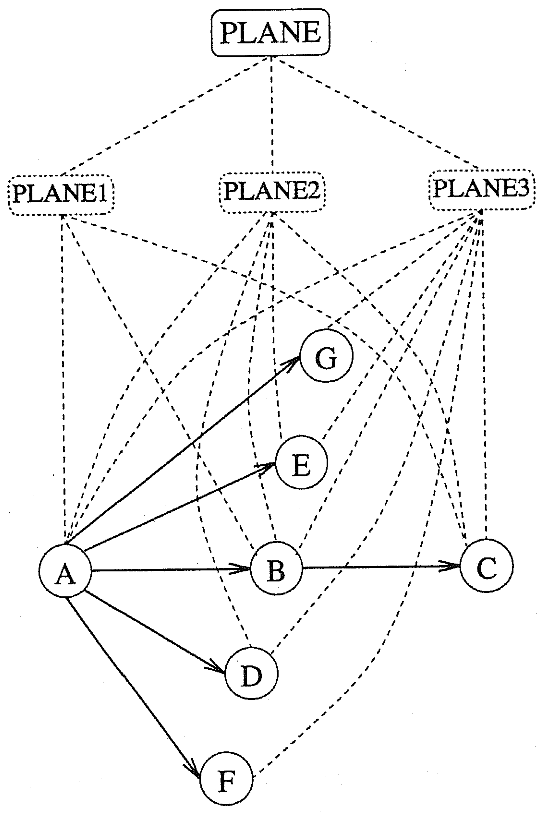


Figure 2.7: Overview of sample conceptual graph

2.1.3 Acquisition and Generalization of the Conceptual Network

If we attempt to generalize two relational networks, then we first have to decide which parts of the two networks correspond to each other. Only if those parts have been identified, we can proceed to generalizing those parts. Such an approach has commonly been described as *learning from examples*. However, several people have claimed that such an approach is not sufficient for learning a concept. For instance, Winston claims that so-called *near-misses*, that is, examples that differ only in small but important points from the positive examples, play a crucial role in learning. I believe, however, that negative examples, and near-misses in particular, do not only place too much stress on choosing an appropriate learning sequence, but are unnatural from the human point of view as well. Rather than discriminating an object due to its negative features, it appears that humans search for the most appropriate concept to match the object to. Hence, we have what we might call a *complete world* paradigm.

For instance, if we are faced with an object we have never seen before, then we usually try to identify it in terms of something previously encountered, that is, something within our experience. Hence, it appears that primarily positive concepts are used for identification. Therefore, as long as our world, or our experience, is not complete, we are apt to misclassify objects. In other words, we can only recognize objects in relation to our world. Now, if we transpose this paradigm to machine learning, then it means that we have to learn a multitude of different concepts at the same time. Therefore, learning in this context corresponds to the acquisition of new models or the generalization of already acquired models. Furthermore, since every object presented to such a system is a real object, it makes sense to acquire them all as separate, positive concepts and thus, each concept has the set of all other concepts as negative examples. It appears that this is the logical generalization of the idea of negative examples (see Fig. 2.8).

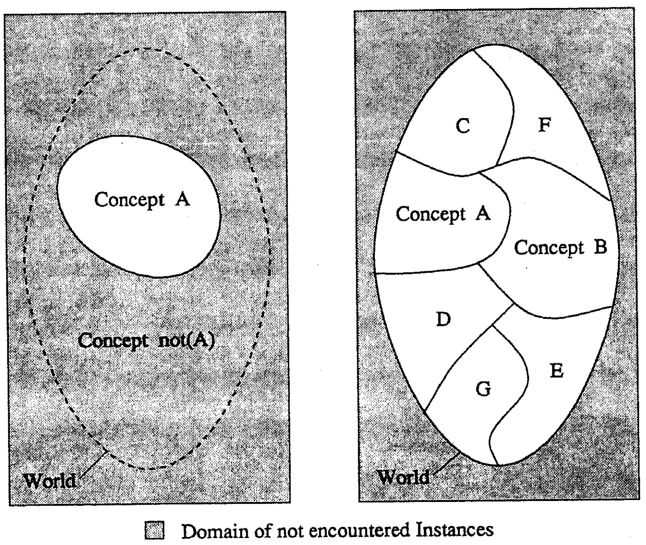


Figure 2.8: Partial world vs. complete world paradigm

We start with an empty conceptual network. The first instance that is acquired will be taken as is and used as the network layer of the conceptual network. The concept layer obtains one node, which is connected to all nodes of the network layer and which carries the name of the concept or class to which the instance belongs. From the second instance onwards, we have to match a new instance with the already existing network layer. Once a valid matching between the relational network and the new instance has been found, the two have to be generalized into a new relational network. This is done by merging those nodes and links that can be merged, and by attaching all the other nodes of the instance that could not be merged to the relational network. The generalization then propagates up through the hierarchy. This propagation is achieved by identifying groups of nodes of the network layer that constitute two or more concepts or partial concepts.

Nodes that cannot be matched are simply attached to the relational network of the network layer, causing the network to grow gradually. If the newly acquired instance is an instance of a previously acquired concept, then all the additional nodes that might appear are connected to that previous concept. If the new instance subsumes or is subsumed by a previous concept, then additional layers have to be added to the conceptual network. Hence, multiple possibilities do not have to be encoded in the network layer, but will be dealt with according to the structure of the upper layers of the conceptual network. This helps to greatly simplify and disambiguate the network layer. If there is no match between the instance and the relational network possible, then the new instance will simply be added as a separate chunk to the relational network layer, without any connections to other nodes of the same layer. In a later stage, this chunk might become connected to the remaining network. However, such a connection is not vital for the functioning of the system.

On the other hand, we might notice that the discriminative power of the relational network is not sufficient to discriminate between two concepts. In that case, it is necessary to split at least one node. To that aim, each node has to be checked whether it could be subjected to splitting. Splitting proceeds by duplicating the node and separating at least one relational predicate connecting to the node. Conflict resolution in the case of splitting is a difficult problem which is not yet completely solved. Splitting might also be applicable in the case where two concepts overlap and one of them gets overly generalized. In that case, the excessive generalization needed for one concept might not be justified for the other concept, and the two concepts should be separated.

It should be noted at this point that the proposed scheme is applicable to both, supervised and unsupervised learning. This means, learning can proceed by giving an example together with the class the example belongs to, or by independent abstraction from a given set of data.

2.1.4 Recognition

Similar to the acquisition of a new instance, recognition starts with matching the instance against the network layer. As stated before, each matched node of the network layer triggers its parental nodes. This is achieved by sending the parental nodes the value obtained as a confidence value of the matching. This confidence value is given as the similarity between two relation nodes, one from the instance and one from the relational network. Each node in the upper layers of the conceptual network accumulates the trigger signals it receives. Once this accumulated value exceeds a node-specific value, the node itself fires. When a node fires, two things happen: first, its parental nodes receive a signal, which consists of the accumulated trigger signals from the lower layer. Second, if there are any inhibitory links connecting the firing node to other nodes, then those other nodes will be inhibited. This mechanism ensures that not two nodes that stand for similar concepts fire at the same time. The triggering of nodes proceeds up the network, until a node without any parental nodes is reached. At that point, the propagation stops and the hierarchy of fired nodes is returned as the response of the system. Hence, a hierarchical recognition of the instance in question becomes possible. If the instance to be recognized is too disjoint from any concept seen before, then the nodes of the partial-concept layers will not be triggered enough to fire themselves, hence the propagation dies out and an empty answer is returned. An empty answer means that the instance cannot be recognized with the present structure of the conceptual network. Even in such a case, it might be possible, however, to recognize at least parts of the instance and thus to give some valuable hints on the structure of the new instance.

Although some examples of recognition are shown in Chapter 7, the scheme for recognition is still in a conceptual stage. The details have not yet been worked out fully and the experimentation is still insufficient to draw conclusions as to the practical usefulness regarding recognition.

2.2 Review of some Learning Schemes

In order to further illustrate the background of the present work, I will briefly reviews two seminal works on concept acquisition and learning, both within the context of computer vision. The most important work in this field is without doubt the one by Winston [61], which I will attempt to cover in the next section. Although Winston's work is somewhat outdated by now, it is still important enough to warrant some closer studies. More recent works in the same field can be found, here I will concentrate on a rather interesting approach by Ueda and Suzuki [58]. Although Ueda and Suzuki's work is related to the system I am describing here in so far as they as well chose binary images as input, the actual approach chosen for representation and generalization is quite different. It is, however, the difference in approach that make Ueda and Suzuki's work worthwhile as a comparison for the present system.

2.2.1 Learning Structural Descriptions from Examples

Winston in his seminal paper [61] attempted to address the following four important points:

- How do we recognize examples of various concepts?

- How do we learn to make such recognitions?

- How can machines do these things?

- How important is careful teaching?

In order to shed some light on those points, Winston developed a computer program that became one of the most famous examples in the area of machine learning. Winston's program works in the domain of three-dimensional structures made of bricks, wedges, and other simple objects. A description of a scene in terms of such simple objects is obtained from a line drawing, employing an early form of Waltz filtering. Once a description of two scenes is obtained, a matching program relates the two scenes together and thus differences in the descriptions can be found and be described themselves.

To obtain a description of a scene, each object is thought of in terms of relationships to other objects. All descriptions of relationships, together with the description of scenes that use those relationships, can be stored in one homogeneous network. The starting point for a description is a line drawing. By means of an early form of Waltz filtering, relations like IN-FRONT-OF, ABOVE, SUPPORTED-BY, A-KIND-OF, and HAS-PROPERTY-OF can be extracted, and a sort of semantic network can be built. When a scene has more than a few objects it is sometimes useful to find a more appropriate description by grouping the objects into individual objects that can be described and related to each other. Grouping is implemented as a two-part process. First, an initial grouping is conjectured from three or more objects that have some properties in common. This is followed by a criticism and revision stage during which those objects are excluded whose relation to the group is weak compared with the average.

Winston describes two different approaches to grouping. The first treats so-called *sequences*, that is, a chain of objects linked together by a SUPPORTED-BY or IN-FRONT-OF relation. Criticism steps in where objects tied together by a chain of relations should not be grouped together because of some other factors. This can be done by checking other relations, like size or position. The second approach to grouping uses common relations and properties. All candidates for group membership must be related to one or more particular objects in the same way. For instance, the four legs of a table are all related to the board by a SUPPORTED-BY relation. They are, therefore, candidates for grouping. Again, the criticism stage attempts to exclude all those objects that are only weakly bound to the present group. Other possibilities for grouping would, for instance, involve checking which objects fit together, as in the case of a jigsaw puzzle, or checking more global relationships.

Once a group has been found, all the parts are gathered under a node that has specifically been created in order to represent the group as a conceptual unit. The meaning of membership in this group is expressed by means of a *typical-member* node. This typical-member node describes the properties and relations that most of the group members share. Hence, each group again can be thought of as an individual concept and the initial hierarchy of the description can be deepened.

Before similarities and differences between two descriptions can be found, it is necessary to determine which parts of the descriptions correspond to each other. This is achieved by matching pairs of nodes, one from each description, which have the same function in their networks. Pairs of nodes that are found to correspond to each other are linked together, and all those linked pairs form the so-called skeleton. Hence, the skeleton is a copy of the structure that appears in both networks. Besides the skeleton, a complete comparison description contains so-called comparison notes, or c-notes for short. A c-note describes the kind of concept the two nodes from a linked pair are pointing to. For instance, if a pair of corresponding objects from two scenes share some features, then an intersection c-note is extended from the skeleton concept corresponding to the linked pair to a new concept describing what the objects have in common. There are a number of defined c-notes, each of them expressing some special relation between the two nodes.

According to Winston, *near misses*, that is, negative instances that differ only in one but important aspect from positive instances, play an important role in learning. This is so because small differences allow the machine to refine certain parts of its current concept. In that respect near misses are used to convey particular properties and ideas rather directly.

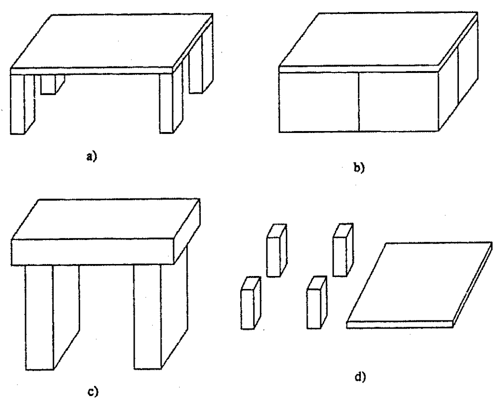
The machine's model building starts with a description of some example of the concept to be learned. This description itself is the first model of the concept. Subsequent samples are either examples of the concept to be learned, or near misses.

The simplest case occurs if there is only one difference between the current model and the new example or near miss. In the case of only one difference there are several possibilities to account for this difference. The way chosen by Winston's program is to find a meta-class to which both of the differing entities belong. For instance, a brick and a wedge both belong to the class of objects or prisms. Hence, they could be replaced by a pointer to object or prism. If the comparison is between a model and a near miss, then the difference gives rise to a MUST or MUST-NOT pointer. Instead of only one difference, it is more common that there will be a number of differences. If the comparison is between a model and a near miss then any of the comparison notes might be the key to a proper generalization of the model. Winston's model builder, in such a case, produces a tree of possible interpretations that are ranked so that the most promising hypothesis can be pursued first. For this, ranking is done by level. This means, the comparison note that is nearest to the origin is ranked first and is then transformed as if it were the only difference. The other interpretations are only used in case that the initial choice of the significant comparison note leads to a subsequent contradiction. In such a case the model builder closes the current branch of the interpretation for further exploration and backtracks up the tree, attempting to develop different interpretations.

As an example of a model building task, consider Figs. 2.9 and 2.10. Fig. 2.9 shows the instance of a table in a) and three near misses in b), c), and d), respectively. The model description of the table, as obtained by Winston's system is shown in Fig. 2.10. All the illustrations are taken from [61].

2.2.2 Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching

Ueda and Suzuki [58] propose an approach for the learning of visual models from real shape samples belonging to the same class. Their approach is based on the generalization of multiscale convex/concave structures, their targets are therefore two-dimensional contour shapes, and proceeds without any a priori knowledge of the class of images to be processed.



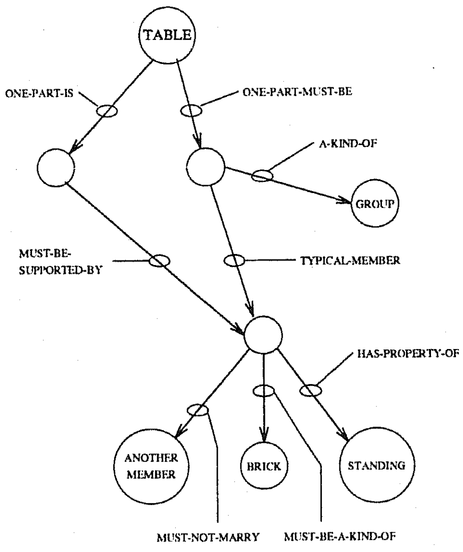


Figure 2.9: Instance (a) and near misses (b,c,d) for a table

Figure 2.10: Derived model description for a table

Ueda and Suzuki claim that graph based learning methods have many disadvantages, like high complexity, and problems arising from describing shapes as parts and relationships among parts. They, therefore, propose a scheme that should overcome those problems, and which allows for an easy comprehension of the derived models by humans. The major idea behind the approach is that shape generalization corresponds to *simplification* of shape structures, which is equivalent to the dropping condition rule commonly used in inductive reasoning. By simplification they do not mean an approximation of shapes, but rather the extraction of common convex/concave structures present within instances of the class.

The system consists of the following four procedures: multiscale representation, multiscale convex/concave structure matching, integration of matching, and model generation. I will describe all of them briefly.

2.2.2.1 Multiscale Representation

A series of smoothed shape contours and their inflection points can be obtained by using a curvature scale space filtering technique. All points on a shape contour can be expressed as a periodic function

$$C(t) = (x(t), y(t)), \tag{2.1}$$

where the variable t denotes contour length. $X(t, \sigma)$ is defined as the convolution of $x(t)$ by a Gaussian kernel $g(t, \sigma)$

$$X(t, \sigma) = \int_{-\infty}^{+\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-u)^2/2\sigma^2} du. \tag{2.2}$$

By computing $Y(t, \sigma)$ in a similar manner, and noting that curvature can be calculated as

$$K(t, \sigma) = \frac{X'Y'' - X''Y'}{(X'^2 + Y'^2)^{3/2}} \quad , \tag{2.3}$$

different levels of curvature of the smoothed curve can be expressed.

The scale space image is a binary image whose 1-pixels represent inflection points on a shape contour curve, corresponding to certain values of t and σ. Every two inflection points thus border a convex or concave segment of the curve. Using discrete values for the scale variable σ results in the creation of incomplete scale space images that make tracking rather difficult. Ueda and Suzuki propose to solve this problem by checking for a consistent replacement of convex/concave segments as the scale factor increases. Let $P^{(k+1)}$ and $P^{(k)}$ represent two inflection point sets associated with consecutive discrete scale factors $\sigma^{(k+1)}$ and $\sigma^{(k)}$ ($\sigma^{(k+1)} > \sigma^{(k)}$).

$$\begin{aligned} P^{(k+1)} &= \{p_i^{(k+1)} \mid i = 1, 2, \ldots, N^{(k+1)}\}, \\ P^{(k)} &= \{p_j^{(k)} \mid j = 1, 2, \ldots, N^{(k)}\}. \end{aligned} \tag{2.4}$$

Let $d(i, j)$ denote the Euclidean distance between $p_i^{(k+1)}$ and $p_j^{(k)}$. The goal is to find a mapping J such that the total distance measure defined as

$$D(P^{(k+1)}, P^{(k)}) = \min_{\{J(i)\}} \sum_{i=1}^{N^{(k+1)}} d(i, J(i)) \tag{2.5}$$

is minimized over J, under the condition $J(1) \leq J(2) \leq \ldots \leq J(N^{(k+1)})$. The above minimization problem can be solved efficiently by means of a dynamic programming approach.

2.2.2.2 Multiscale Convex/Concave Structure Matching

The robust matching of multiscale convex/concave structures is a major point in Ueda and Suzuki's work. Their matching algorithm searches for the corresponding segment pairs from the finest scale to coarser scales by replacing odd consecutive segments at the finest scale with a segment at a coarser scale. Two convex/concave segment sequences at scales $\sigma^{(h)}$ and $\sigma^{(k)}$ can be written as

$$\begin{aligned} A^{(h)} &= \{a_1^{(h)}, \ldots, a_{N^{(h)}}^{(h)}\} \\ B^{(k)} &= \{b_1^{(k)}, \ldots, b_{M^{(h)}}^{(k)}\}. \end{aligned} \quad (2.6)$$

$N^{(h)}$ and $M^{(k)}$ are the number of segments, and $a_i^{(h)}$ stands for either the convex or concave segment between the two consecutive inflection points $p_i^{(h)}$ and $p_{i+1}^{(h)}$. The matching is formally performed between the finest scale segment sequences $A^{(0)}$ and $B^{(0)}$. However, if consecutive segments at the finest scale can be replaced with one coarser scale segment, then the replaced segment is used for the matching, as, for instance, shown in Fig. 2.11. The

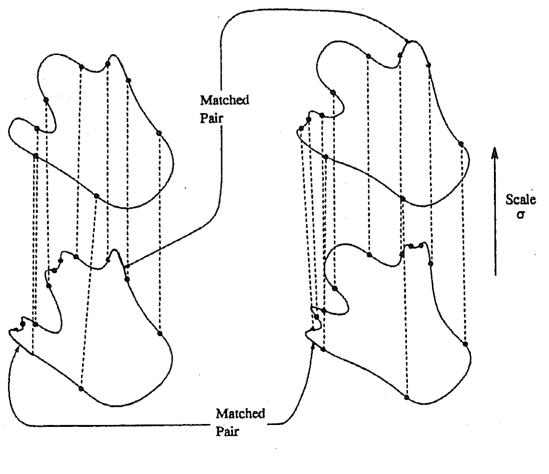


Figure 2.11: Matching strategy of multiscale segment matching (from [58])

best match is equivalent to the best segment correspondence such that the sum of multiscale segment dissimilarities is minimized. The multiscale segment dissimilarity ψ is defined as

$$\begin{aligned} &\psi\left(\boldsymbol{a}(i-2n \mid i), \boldsymbol{b}(j-2m \mid j)\right) = \\ &\qquad \delta(a_{i'}^{(h)}, b_{j'}^{(k)}) + \rho_A\left(\boldsymbol{a}(i-2n \mid i) \rightarrow a_{i'}^{(h)}\right) + \rho_B\left(\boldsymbol{b}(j-2m \mid j) \rightarrow b_{j'}^{(k)}\right), \end{aligned} \quad (2.7)$$

where $a(i-2n \mid i)$ denotes a sequence of $(2n+1)$ consecutive segments of A, $\rightarrow$ denotes the replacement operator, and $a(i-2n \mid i) \rightarrow a_{i'}^{(h)}$ stands for the replacement of the sequence $a(i-2n \mid i)$ by the segment $a_{i'}^{(h)}$. Furthermore, δ is the one-to-one segment dissimilarity, and ρ_A and ρ_B represent the costs of the respective replacements.

The one-to-one segment dissimilarity δ is defined as

$$\delta(a_i^{(h)}, b_j^{(k)}) = \frac{\mid \theta_i^{(h)} - \theta_j^{(k)} \mid}{\theta_i^{(h)} + \theta_j^{(k)}} \left| \frac{l_i^{(h)}}{L_I^{(h)}} - \frac{l_j^{(k)}}{L_J^{(k)}} \right|, \tag{2.8}$$

where $\theta_i^{(h)}$ equals the total rotation angle of the tangent vector along $a_i^{(h)}$, $l_i^{(h)}$ is the segment length of $a_i^{(h)}$, and $L_I^{(h)}$ corresponds to the total length of all $a_i^{(h)}$. δ is set to ∞ whenever the polarities of $a_i^{(h)}$ and $b_j^{(k)}$ are different. This means that a convex segment cannot be matched to a concave segment.

The cost for replacing a sequence of segments with one segment at a coarser scale is defined as

$$\rho_A \left(a(i-2n \mid i) \rightarrow a_{i'}^{(h)} \right) = \lambda \sum_{s=i-(2n-1)}^{i} \frac{\mid \theta_{s-1} - \theta_s \mid}{\theta_{s-1} + \theta_s} \frac{\mid l_{s-1} - l_s \mid}{L_I}. \tag{2.9}$$

ρ_B is defined similarly. The weighting factor λ is usually set to 1.

The total segment dissimilarity is now given as

$$\Psi(A,B) = \min \sum_{w=1}^{W} \psi \left(a(i_{w-1}+1 \mid i_w), b(j_{w-1}+1 \mid j_w) \right), \tag{2.10}$$

where W corresponds to the number of matched pairs. Although the number of all possible combinations of segment correspondences increases exponentially with increasing $N^{(h)}$ and $M^{(k)}$, an effective solution of the minimization problem can again be obtained by means of a dynamic programming approach.

2.2.2.3 Integration of Results

The segment correspondences found from the multiscale segment matching operation can be integrated into one convex/concave structure for all samples. The integration consists of three steps

1. For every sample shape, construct a partially ordered set from the corresponding segment results.
2. For every sample shape, find a maximal family of sets (MFS) that consists of maximal elements in the partially ordered set.
3. For all the MSF's, check the consistency among them.

The integration is not just a minimization of the number of convex/concave segments, but rather the simplest convex/concave structures among the corresponding segment match results are extracted.

The optimal correspondences from the multiscale segment match can be represented by using the finest scale. In the case of N samples, matching is performed $N(N-1)/2$ times, hence each sample has $N-1$ sets of correspondence results. Let A_i be the set whose elements are the finest scale segments of the ith sample shape S_i. Assuming that $n_{i,j}$ segment correspondences are obtained by matching the two shapes S_i and S_j, then both A_i and A_j are partitioned into $n_{i,j}$ correspondence units. Then, the partitioned segment family of sets (PSFS) of A_i by A_j is defined as

$$\mathcal{P}_{i,j} = \{a^j_{i,w} \mid w = 1, \ldots, n_{i,j}\}. \tag{2.11}$$

Considering N different sample shapes, $N-1$ PSFS are obtained for each sample. The union of these PSFS is given as

$$Q_i = \bigcup_{1 \le j \le N, j \ne i} \mathcal{P}_{i,j}, \tag{2.12}$$

which can be shown to be a partially ordered set. Therefore, a maximal element in the partially ordered set Q_i is an element $q \in Q_i$, such that $q \subseteq q'$ for no $q' \in Q_i$. As there can exist multiple maximal elements in Q_i, the maximal family of sets (MFS) in the partially ordered set Q_i can be obtained. The MSF of Q_i will be denoted as Q_i^*. If an MFS does not consist of pairwise disjoint sets, there does not exist a structure common to the given samples and generalization becomes impossible. Furthermore, only if there exists a one-to-one correspondence between any two MSF's Q_i^* and Q_j^*, is it possible to define a common structure between the samples.

2.2.2.4 Model Generation

Each element in the obtained MFS Q_i^* corresponds to an optimal scale segment in the multiscale representation of S_i. As curvature scale space filtering results in a shrinking of boundary size, position gaps are filled by a linear interpolation method, and N generalized shapes $G_1, \ldots, G_N$ can be obtained by processing all the N sample shapes.

Each generalized shape G_i can be approximated as a polygon, which is represented by an ordered list of vertices

$$G_i = \{(x_{i,t}, y_{i,t}) \mid t = 1, \ldots, T_i\}, \quad i = 1, \ldots, N, \tag{2.13}$$

where $(x_{i,t}, y_{i,t})$ are the (x, y) coordinates of the tth segment, and T_i is the number of vertices of the ith generalized shape. All generalized shapes can be normalized such that they have the same number of vertices $\overline{T}$. This is achieved by setting the vertices at distance $L_i/\overline{T}$, where L_i is the contour length of the ith generalized shape. If we denote the normalized generalized shapes as

$$G_i' = \{(x_{i,t}', y_{i,t}') \mid t = 1, \ldots, \overline{T}\}, \quad i = 1, \ldots, N, \tag{2.14}$$

and the desired visual model M as

$$M = \{(X_t, Y_t) \mid t = 1, \ldots, \overline{T}\}, \tag{2.15}$$

then M can be obtained by using the following interpolation

$$\begin{aligned} X_t &= \left(\sum_{i=1}^{N} \alpha_i x_{i,t}'\right) / \sum_{i=1}^{N} \alpha_i \\ Y_t &= \left(\sum_{i=1}^{N} \alpha_i y_{i,t}'\right) / \sum_{i=1}^{N} \alpha_i . \end{aligned} \tag{2.16}$$

The weight parameters α_i are usually set constant.

2.2.2.5 Conclusions

Ueda and Suzuki use some examples to further illustrate the performance of their algorithm. They show how a modified dissimilarity measure could be used to recognize an unknown sample given several model classes. The approach as proposed by Ueda and Suzuki is novel and features many good ideas. The authors claim that their work is the first attempt to shape generalization that uses the image domain directly, rather than the symbolic domain.

2.3 Context of this Research

In several respects the two systems outlined in Section 2.2 represent two extremes of one problem. As Ueda and Suzuki point out, their approach is strictly restricted to the image domain, whereas Winston's approach, and indeed many others, is working in the symbolic domain. Both approaches appear to have advantages and disadvantages.

Similar to Ueda and Suzuki's work, the input to our system consists of binary images only. Binary images are in so far a restricted domain as all their information is concentrated on the contour. Ueda and Suzuki choose to employ this feature directly, by using a contour-based approach. Their method, as explained in the previous section, is based on a multiscale representation of the boundary of a shape. In contrast, the approach as proposed here, attempts to first recover region information based on the contour. However, such recovery is, in general, an ill-posed problem, and we cannot expect to obtain unique results in any case. It appears, nevertheless, that such an interpretation in terms of regions, rather than contours, is often more intuitive and captures more of the functionality of the input shape. I will explain the recovery of region information in depth in Section 3.4.

Most similarities can possibly be found with Haar's work [19], who used parameter intervals to obtain a fuzzy representation of relations. Our concept of generalized predicates can be understood as a logical conclusion of Haar's fuzzy interpretation. However, both Haar's work and the present work take their incentive from the work on semantic networks, as for instance Winston's work. Semantic networks appear to be easily understood by humans; whether they coincide with human memory structures is another question that cannot be answered here. However, semantic networks do have several shortcomings, as outlined before. The present work can therefore be understood as an attempt to find a structure for representation and generalization of visual objects that is more adaptable and more stable than semantic networks.

CHAPTER 3

Decomposition of Binary Shapes

3.1 Purpose and Overview

The task I am addressing in this chapter concerns the description of a shape by decomposing it into meaningful parts. As image intensity is not trivially correlated with geometrical structure, I restrict the task to binary images, that means, that all the available information is assumed to be concentrated along the boundary or silhouette of the object in question. There have been several approaches to obtaining a meaningful decomposition in this restricted domain. Usually, the basic constituents are defined to be convex sets, or more restrictive, simple geometrical figures like squares, rectangles, or circles. Some of the first work was done by restating the problem in terms of polygonal decomposition into convex sets [29, 48]. Shapiro and Haralick [52] and Shapiro [51] recast the problem as a graph clustering problem and, instead of looking for perfect convexity, were looking for intuitively pleasing parts. Even more work has been done by using the skeleton as a starting point. For instance, Ito et al. [27] or Ibaraki et al. [26] use both, contour and skeleton information explicitly to guide their decomposition into strokes and loops. Arcelli and Serino [2] decompose a shape using the information obtained by propagating distance labels into the interior of the shape. Another interesting approach, based on mathematical morphology, was proposed by Pitas and Venetsanopoulos [43, 44]. Although they claim that the choice of simple geometric objects as basic constituents is intuitively used by humans, the results they obtain are quite far from intuition and seem rather to be applicable for coding tasks.

Here we are trying to overcome several shortcomings that shape decomposition schemes suffered from. In terms of Pavlidis' taxonomy [42], we propose an interior, information losing, space domain transform, using the skeleton representation as a starting point. Similar to Pitas and Venetsanopoulos' work, our approach is based on concepts of mathematical morphology; however, as basic constituents of the decomposition we accept anything that is approximately convex, a term that will be defined by itself. We overcome the restrictions of mathematical morphology concerning rotation invariance by proposing a new operator, the so-called elongation operator.

The remainder of the chapter is organized as follows. First, I will give a simple introduction to the methods of mathematical morphology, on which the proposed decomposition approach is based upon. Next, I will introduce the concept of approximate convexity, which plays an important role in the proposed decomposition scheme. Although only the application of approximate convexity in two dimensions will be used here, the definition is given in n-dimensional Euclidean space. Finally, I will outline the actual decomposition of binary shapes, together with giving a series of examples for illustrating the performance of the proposed approach.

3.2 Mathematical Morphology

3.2.1 Introduction

The word *morphology* refers to the study of form and structure (from the Greek words *morphe* meaning form and *logia* to speak) and is in this sense used in areas as various as biology, geography, and linguistics. In image processing, the term mathematical morphology refers to a particular discipline concerned with the analysis of the structure of materials as for instance in mineralogy, petrography, or cytology. The origin of mathematical morphology is generally traced to the year 1964, when G. Matheron from the École des Mines at Fontaînebleau, Paris, was asked to investigate the relationships between the geometry of porous media and their permeabilities. The original notions used in that studies, however, date much further back and are due to integral geometry, especially to H. Minkowski and H. Hadwiger.

The morphological approach, as developed by Matheron [33] and Serra [50] at Fontaînebleau, is generally based on the analysis of a two-valued image (binary image) in terms of some predetermined geometric shape, known as the *structuring element.* This two-dimensional approach was first generalized to gray-level images by applying the same operations to cross-sections of the images. The real generalization of structuring elements to *structuring functions* in $2\frac{1}{2}$ dimensions, together with the theoretical connection to fuzzy models, is, however, mainly due to Sternberg [56].

Mathematical morphology is an approach for the analysis of structure, based on set-theoretic concepts. It has three aspects: an algebraic one, dealing with image transformations derived from set-theoretical operations; a probabilistic one, dealing with models of random sets applicable to the selection of small samples of materials; and an integral geometrical one, dealing with image functionals. This section mainly addresses the first aspect: the algebraic study of a body of image transformations based on operations similar to those of set theory, which are in general nonlinear.

Why then do we need set theoretical concepts for the analysis of images? It is generally agreed that humans can identify a scene from a two-tone drawing of it, where only simple elements like contours, bars, shadows, etc. can be seen. As such images are Boolean in nature, any further processing of them cannot be linear, but must be related to Boolean algebra. For example, if object X is behind object Y, then in the drawing one can see the contour of X minus Y (in the set-theoretical sense). Or the shadow of a union of objects will be the union of their shadows. These thoughts illustrate certain properties of images and set-theoretic concepts and explain why image transformations can be based on the Boolean algebra of set operations.

3.2.2 Binary Morphology

Binary mathematical morphology is expressed using the language of set theory. Sets in morphology represent the shape of an object given by all black and white pixels in the image. Sets in two dimensions denote foreground regions; in three dimensions a set may denote time-varying binary images or binary solids.

The primary morphological operations are dilation and erosion. From dilation and erosion, the morphological filter operations opening and closing can be composed. Especially the latter two filter operations are widely used in shape representation or decomposition.

3.2.3 Binary Dilation

Dilation is the transformation that combines two sets by vector addition of set elements. In the mathematical literature this operation is often called *Minkowski addition*, after the German mathematician Minkowski who first proposed it [37]. If A and B are two sets with elements $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_m)$ respectively, then the dilation of A by B is the set of all possible vector sums of pairs of elements, one coming from A and one coming from B. Formally, the dilation of A by B is denoted by $A \oplus B$ and is defined as

$$A \oplus B = \{c = a + b \mid a \in A, b \in B\} \tag{3.1}$$

Although dilation is commutative, i.e., $A \oplus B = B \oplus A$, in practice, the two sets A and B are not thought of symmetrically. The first set A of the dilation $A \oplus B$ is usually associated with the image undergoing morphological processing, and the second set B is referred to as the *structuring element*, that is, the shape that acts on A to produce $A \oplus B$.

To characterize dilation more properly, a notation for the translation A_t of a set A by a vector t is needed:

$$A_t = \{c = a + t \mid a \in A\} \tag{3.2}$$

Thus, the dilation can be represented as the union of translates of the image, namely

$$A \oplus B = \bigcup_{b \in B} A_b \tag{3.3}$$

An example of the dilation of a set by a square structuring element is shown in Fig. 3.1, where $+$ denotes the origin of the structuring element.

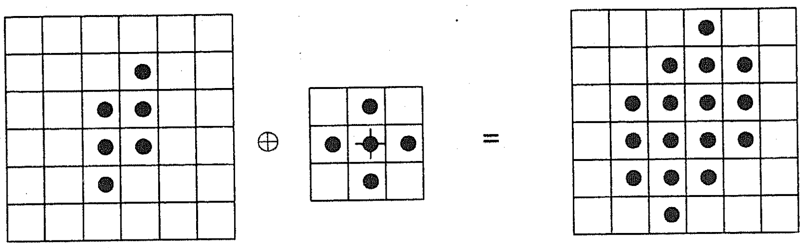


Figure 3.1: Dilation by a diamond structuring element

Because dilation is commutative,

$$A \oplus B = \bigcup_{a \in A} B_a \tag{3.4}$$

As addition is associative, so is dilation, namely

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \tag{3.5}$$

This property of the dilation is called *chain rule* or *iterative rule* and is of significant practical importance. By using a straightforward implementation, the dilation of an $N \times N$ image by a $K \times K$ square structuring element can be accomplished in K^2 operations. However, a structuring element of that form is highly decomposable, that is, the structuring element itself can be expressed as a dilation of two or more simpler structuring elements. Hence, if we decompose the $K \times K$ square structuring element into $B_1 \oplus B_2$, where B_1 and B_2 are $K \times 1$ and $1 \times K$ structuring elements, respectively, then according to Eq. 3.5, the complexity of the dilation is of order $2K$, which is an immense improvement from the initial value of K^2. Fig. 3.2 illustrates the decomposition of a 5×5 structuring element. Of course, each of the resulting column and row structuring elements could be decomposed

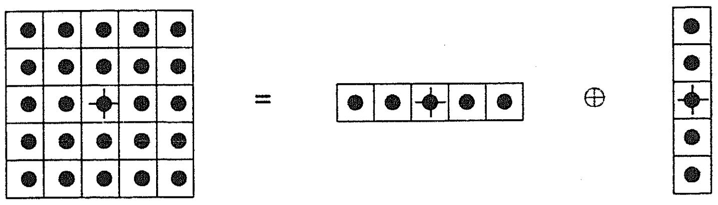


Figure 3.2: Decomposition of a square structuring element

further, yielding an even better performance.

Other important properties of the dilation include:

- Translation invariance:

$$A \oplus B_t = (A \oplus B)_t \tag{3.6}$$

- Distribution over union:

$$(A \cup B) \oplus C = (A \oplus C) \cup (B \oplus C) \tag{3.7}$$

- Extensivity:

$$A \oplus B \supseteq A \quad \text{if and only if } O \in B, \text{ where } O \text{ stands for the origin} \tag{3.8}$$

- Increasing:

$$\text{If } A \subseteq B \text{ then } A \oplus K \subseteq B \oplus K \tag{3.9}$$

3.2.4 Binary Erosion

Erosion is the morphological dual to dilation. Sometimes, erosion expressed in a slightly different form is referred to as Minkowski subtraction, and as such was first proposed by Hadwiger [20]. If A and B are two sets with elements $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_m)$ respectively, then the erosion of A by B is the set of all elements c for which $c + b \in A$ for every $b \in B$. The erosion of A by B is denoted by $A \ominus B$ and is defined as

$$A \ominus B = \{c \mid c + b \in A, \forall b \in B\}. \tag{3.10}$$

An example of the erosion is shown in Fig. 3.3.

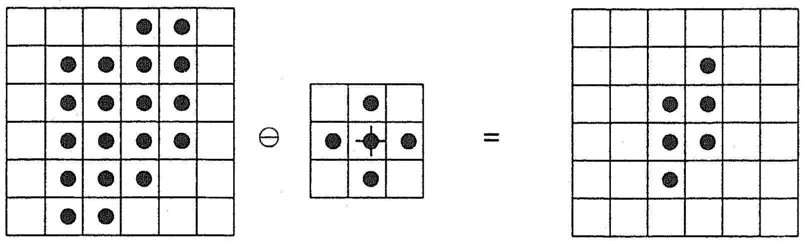


Figure 3.3: Erosion by a diamond structuring element

Again employing the notion of a translated set, erosion can be expressed as follows

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{3.11}$$

In other words, erosion is the intersection of all negative translates of the image.

As stated before, erosion is the dual of dilation. Therefore, every erosion can be expressed as a dilation or vice versa:

$$(A \ominus B)^C = A^C \oplus \breve{B} \tag{3.12}$$

where A^C is the complement of A, i.e., $A^C = \{x \notin A\}$, and $\breve{B}$ is the reflection of B, i.e., $\breve{B} = \{x = -b | b \in B\}$. With respect to structuring element decomposition, a chain-rule for erosion holds

$$A \ominus (B \oplus C) = (A \ominus B) \ominus C \tag{3.13}$$

That means, a large erosion can be computed more efficiently by a number of smaller erosions.

Again, some of the basic properties of the erosion include:

- Translation invariance:

$$A \ominus B_t = (A \ominus B)_t \tag{3.14}$$

- Distribution over intersection:

$$(A \cap B) \ominus C = (A \ominus C) \cap (B \ominus C) \tag{3.15}$$

- Anti-extensivity:

$$A \ominus B \subseteq A \quad \text{if and only if } O \in B \tag{3.16}$$

- Increasing:

$$\text{If } A \subseteq B \text{ then } A \ominus K \subseteq B \ominus K \tag{3.17}$$

However, erosion is not commutative, i.e.,

$$A \ominus B \neq B \ominus A \tag{3.18}$$

Although erosion and dilation are dual, this does not mean that they undo each other. In general the following inequality holds

$$(A \oplus B) \ominus B \neq A, \tag{3.19}$$

but

$$(A \oplus B) \ominus B \subseteq A. \tag{3.20}$$

3.2.5 Opening and Closing

Very often, dilations and erosions are employed in pairs, either dilation of an image followed by an erosion, or vice versa. In either case, the aim of successively applied dilations and erosions is the elimination of specific image details smaller than the structuring element, an operation that is especially useful for removing noise in an input image.

The opening of an image B by a structuring element K is denoted by $B \circ K$ and defined as

$$B \circ K = (B \ominus K) \oplus K. \tag{3.21}$$

The closing, on the other hand, is denoted by $B \bullet K$ and defined as

$$B \bullet K = (B \oplus K) \ominus K. \tag{3.22}$$

If $B \circ K = B$, then B is open with respect to K. On the other hand, if $B \bullet K = B$, then B is closed with respect to K. However, morphological openings and closings have no relation with topologically open or closed sets!

The functionality of opening and closing is closely related to the specification of a filter by its bandwidth. Morphologically filtering an image by an opening or closing operation corresponds to the ideal, non realizable bandpass filters of conventional linear filtering. Once an image is filtered by an ideal bandpass filter, clearly, further application of the same filter does not alter the result. This characteristic of a transformation is called idempotence. That is

$$\begin{aligned} (B \circ K) \circ K &= B \circ K \\ (B \bullet K) \bullet K &= B \bullet K. \end{aligned} \tag{3.23}$$

The opening of a set B is the domain swept out by all translates of the structuring element K that are included in the set B, namely

$$B \circ K = \{x \mid x, t \in B, x \in K_t \text{ and } K_t \subseteq B\} = \bigcup_{K_t \subseteq B} K_t. \tag{3.24}$$

Hence, the opening of a set by a disk structuring element, that is, a circular structuring element, smoothes the contour, breaks narrow isthmuses and eliminates small islands and sharp peaks or capes.

There is a duality between opening and closing. What opening does to the object, closing does to the background. The following relation holds

$$(B \circ K)^C = B^C \bullet \check{K}. \tag{3.25}$$

Therefore, the closing can be characterized as follows

$$B \bullet K = \{x \mid x \in \check{K}_t \Rightarrow \check{K}_t \cap B \neq \emptyset\} = \bigcap_{\{t \mid \check{K}_t \cap B \neq \emptyset\}} \check{K}_t^C. \tag{3.26}$$

Fig. 3.4 shows examples of the opening and closing of a shape. As can be seen from Figs. a) and c), the opening removes narrow promontories, and the closing fills narrow bays and small holes. The used octagonal structuring element can be obtained as the dilation of a square structuring element by a diamond structuring element.

As already mentioned before, both opening and closing are idempotent transformations. Besides idempotence, the following properties are of interest:

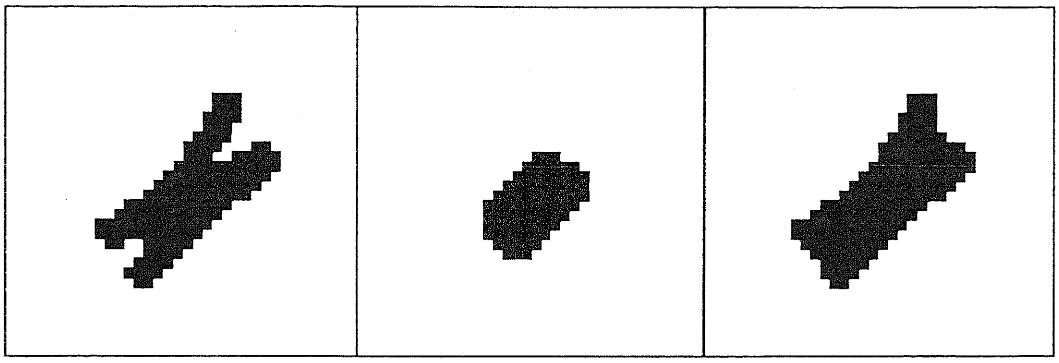


Figure 3.4: Opening and closing with an octagonal structuring element of size 2

- Translation invariance:

  It follows directly from Eq. 3.6 and 3.14 that openings and closings are translation invariant. In addition, they are invariant to the translation of the structuring element

$$A \circ B_t = A \circ B$$
$$A \bullet B_t = A \bullet B. \tag{3.27}$$

- Anti-extensivity of opening:

$$A \circ B \subseteq A. \tag{3.28}$$

- Extensivity of closing:

$$A \bullet B \supseteq A. \tag{3.29}$$

- Increasing:

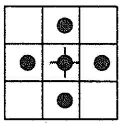
  Follows from Eq. 3.9 and 3.17.

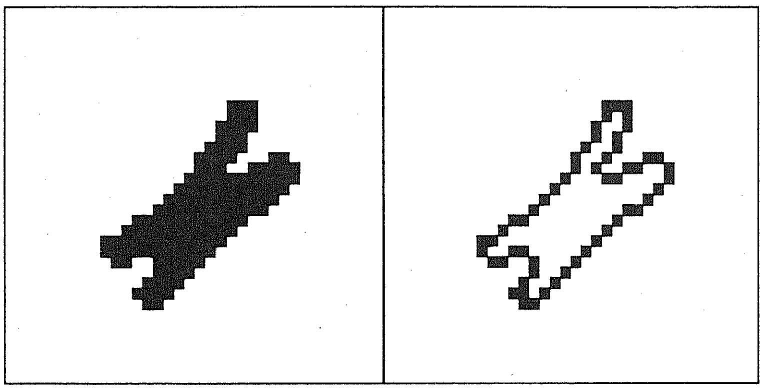
3.2.6 Boundary extraction

The boundary B of a binary object A can easily be obtained by an erosion followed by a subtraction. This means, we have to subtract from the object an eroded version of itself. The remainder should then be the boundary. Writing this down formally, gives

$$B(A) = A \setminus (A \ominus K_1), \tag{3.30}$$

where $\setminus$ stands for the set subtraction, i.e., $A \setminus B = A \cap B^C$, and K_1 is the diamond structuring element of size 1, i.e.,

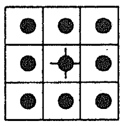




a) Object b) Extracted boundary

Figure 3.5: Extracted boundary of a binary object

Note that eroding by a diamond structuring element of size 1 peels off an 8-connected layer from the object, hence the resulting boundary is 8-connected as well, as shown in Fig. 3.5. If the 4-connected boundary should be needed, then a square structuring element of size 1 has to be used instead, e.g.,



3.2.7 Convex hull

The following proposition concerning the convex hull $C(A)$ of an object A is due to Serra [50]:

Proposition 1 *If K is a compact convex set with non-empty interior and if K admits finite curvature at each point of its boundary, then for every closed set A,*

$$C(A) = \lim_{\lambda \to \infty} A \bullet \lambda K.$$

This means, by closing an object by a structuring element of infinite size, the convex hull of the object can be obtained.

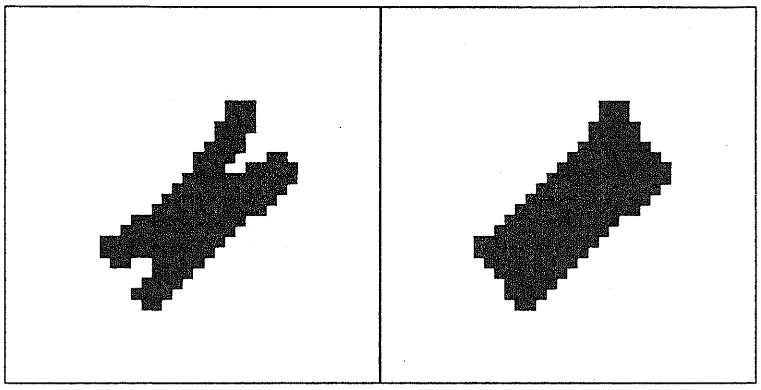


Figure 3.6: Convex hull as obtained by a hexadecagonal structuring element

As Proposition 1 is using the Euclidean space as its domain, there appear two problems when trying to transpose it to the discrete plane, i.e., $\mathbb{Z}^2$. Namely, how to implement a discrete structuring element having finite curvature everywhere, and how to realize $\lambda \to \infty$. Hence, using Matheron's proposition we will only be able to obtain a pseudo convex hull, rather than the perfect convex hull. The question that remains to be solved is how accurate the pseudo convex hull will be. In order to approximately achieve the criterion of finite curvature, we need a structuring element that is a close approximation to the disk. A hexadecagon, that is, a regular polygon with 16 vertices, is the best approximation to the disk that can be implemented iteratively by 2×3 and 3×2 templates, where the origin is placed in one of the corners of the template. An example of using a hexadecagonal structuring element is given in Fig. 3.6. For determining the necessary size r of the structuring element we can analyze the worst-case behaviour of the pseudo convex hull. Using the general results from Appendix A for the case of the hexadecagon (with number of vertices $n = 16$), we obtain for the size r of the hexadecagon and the depth d_{cav} of the maximal nondetectable concavity the following values

$$\begin{aligned} r &\geq \frac{D_{max}}{0.77} \\ d_{cav} &\leq \frac{D_{max}}{10.05}, \end{aligned}$$

where D_{max} denotes the maximal diameter of the set X in question.

However, a much smaller structuring element will do in most practical cases. Values ranging

$$\frac{D_{max}}{3} \leq r \leq \frac{D_{max}}{2}$$

have been tested and found to yield useful results in many cases.

3.2.8 Elongation

Elongation can be understood as a dilation using a vector as structuring element. In other words, elongation is a dilation where the direction of the propagation has been fixed. Elongation of a set X by a structuring element S, which is defined to be a unit vector parallel to the main axis of X, is defined as

$$\widetilde{X}^{\lambda} = X \oplus \lambda S \;\; (\lambda \geq 0) \tag{3.31}$$

The elongation operation is in this form due to Hadwiger [20] and it has been employed in integral geometry. However, as an operation of mathematical morphology, elongation has not been used so far. I will show a possible implementation of elongation within the framework of mathematical morphology in Section 3.4.1.3.

3.3 Approximate Convexity

3.3.1 Introduction

The concept of convexity plays an important role in various fields like computational geometry, computer graphics, image processing, etc. Convexity and its implications are well understood, thanks to its sound definitions from mathematics and geometry. Despite the mathematical and geometrical importance of convexity, convex sets are rarely encountered in nature. For many applications, particularly in image decomposition or recognition, it seems appropriate to relax the notion of convexity and to deal with sets that are almost convex or approximately convex. Such a notion of approximate convexity does not only allow to actively disregard effects of distortion and noise, but it also captures some perceptual notions about fuzziness of the appearance of shapes. Both of those points are important if we attempt to build a truly versatile vision system.

What we need for capturing this idea of approximate convexity is a measure that expresses how much a given shape differs from or resembles its convex hull. Ideally but not necessarily, such a measure should be obtained as a scalar in the range $[0, 1]$. In the literature not much work along those lines can be found. In the next section, I will mention three different approaches together with one new approach. Next, I will attempt to give a complete theoretical treatment of approximate convexity in n-dimensional Euclidean space. I will show that it is possible to find a solution that is valid in Euclidean space of arbitrary dimensionality. Finally, I will mention some practical considerations on how to actually measure approximate convexity in discrete 2 or 3-dimensional spaces.

3.3.2 Convexity Measures

Scanning the literature, it is possible to find some papers that deal explicitly or implicitly with the problem of measuring approximate convexity. The first such approach is due to Sklansky [54] and is centered on the idea of *convex deficiency*. Sklansky actually defines two different measures: an area measure S_a and a depth measure S_d. The area measure S_a simply corresponds to the ratio of the area of the concavity to the area of the convex hull. The depth measure S_d is given as the sum of weighted ratios of the depths of concavities to the width of the convex hull. Further work includes Stern's *polygonal entropy* [55] and Boxer's *Deviation from Convexity* [7], which, however, is subsumed by Sklansky's measure S_d, which can be viewed as a normalized version of Boxer's deviation measure.

A slightly different approach to the measurement of nonconvexity was first proposed in [23]. This work, which is the foundation for the present work, takes its ideas from geometrical probability. Unlike the other approaches that are defined over the family of polygons, this approach is defined using the language of set theory. Although this is not of practical importance, it allows us to obtain mathematically simple but strong results, which can be easily generalized.

In order to define approximate convexity, it is of advantage to start with a proper definition of convexity. There are several definitions possible, here we choose the following one:

Definition 1 *A set of points K is called convex if for each pair of points $A \in K$, $B \in K$ it is true that $\overline{AB} \subset K$, where $\overline{AB}$ is the line segment connecting A with B.*

It can easily be seen that this definition is as such valid in Euclidean space of any dimensionality. Taking the above definition as a starting point, in the case of two dimensions, approximate convexity can be defined as follows:

Definition 2 *The degree of approximate convexity of a set X is defined to be the probability that an arbitrary line segment that intersects X is convex, i.e., produces only one intersection.*

Again, this definition can be transposed to Euclidean spaces of arbitrary dimensionality without any changes. As I will show in the next section, based on the above definition, it is possible to derive an equation that keeps its generality for n-dimensions, and that yields computationally simple results for the important cases of two and three-dimensional spaces.

3.3.3 Derivation

3.3.3.1 Definitions

In what follows, let us denote the set in question by X, and its convex hull as $C(X)$. Further we define the residue of $C(X)$ minus X

$$R(X) = C(X) \setminus X, \tag{3.32}$$

that is, the set of concavities. Hence, $X \cup R(X) = C(X)$. Similarly we need the residue that can be defined using the boundaries of $C(X)$ and $R(X)$

$$Q(X) = B(C(X)) \cap B(R(X)), \tag{3.33}$$

where $B(X)$ denotes the boundary of the set X. In what follows, $Q(X)$ will sometimes be called the cover of $R(X)$, as it can be visualized as covering the hole created by the concavity. Note that if X has a certain dimension n, then $Q(X)$ will have dimensionality $n - 1$.

In the next subsection I will try to obtain the most general result possible, based on the derivations in Appendix B. We can start by redefining approximate convexity in the more general setting:

Definition 3 *The degree of approximate convexity of a set X^n is defined to be the probability that the intersection of an arbitrary set L^r ($r < n$) with X^n is convex.*

where the superscript denotes the dimensionality of the set. It was shown in [23] that the consideration of interactions between concavities introduces complications that make the problem almost intractable, although doing so does not change the result too much. Therefore, here as well, I will neglect any interactions between concavities and consider only the simple case.

According to the law of large numbers, we can obtain probabilities by observing and counting outcomes of experiments. In the case of Definition 3, this means, that we can count how many of the intersections of the probing set L^r with the probed set X^n are

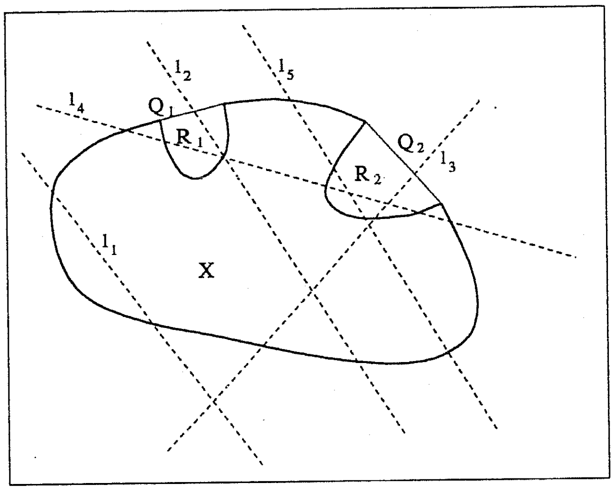


Figure 3.7: Random lines intersecting a set X: lines l_1, l_2, and l_3 are convex, whereas lines l_4 and l_5 are nonconvex

convex. However, as for instance the number of lines that intersect a given curve is infinite, we have to employ concepts from integral geometry, in particular the concept of a *measure* of a line or a hyper-plane. Therefore, instead of actually counting hyper-planes, we can work with the measures of those planes.

Instead of using the measure of the sets that give a convex intersection, it is actually easier to use the measure of all those sets L^r whose intersection with X is not convex (see Fig. 3.7). Let us denote the measure of all the sets L^r that intersect X by $m(L^r \mid L^r \cap C(X) \neq \emptyset)$. The measure of the sets L^r that intersect $R(X)$ can then be written as $m(L^r \mid L^r \cap R(X) \neq \emptyset)$; clearly, all the nonconvex intersections must be included in this measure. Among all the sets L^r intersecting $R(X)$, only those sets whose members are intersecting the associated cover of $R(X)$, namely $Q(X)$ are convex, hence the measure of the nonconvex set is given by $m(L^r \mid L^r \cap R(X) \neq \emptyset) - m(L^r \mid L^r \cap Q(X) \neq \emptyset)$[1]. The degree of approximate convexity can now easily be obtained by putting together all those fragments, namely

$$cv^{n[r]}(X) = 1 - \frac{m(L^r \mid L^r \cap R(X) \neq \emptyset) - m(L^r \mid L^r \cap Q(X) \neq \emptyset)}{m(L^r \mid L^r \cap C(X) \neq \emptyset)}. \tag{3.34}$$

[1]As stated before, here we are neglecting the fact that all those sets L^r which intersect more than one concavity are counted multiply. For an example of such a L^r see line l_4 in Fig. 3.7

where $cv^{n[r]}$ denotes the approximate convexity of an n-dimensional set X when using an r-dimensional set L^r for probing. If we rephrase the above equation, then we have that the value for approximate convexity is one minus the probability that a set that intersects X is nonconvex.

3.3.3.2 Approximate Convexity in N-Dimensions

As we have seen in Section 3.3.3.1, we need to calculate the measure for three different sets, which can actually be divided into two groups. To the first group belong the convex hull of X, namely $C(X)$, and the concavities $R(X)$. Both sets are proper n-dimensional sets in E^n. The case is somewhat different for the third set $Q(X)$, which is a boundary set, that is, a $(n-1)$ dimensional set in E^n.

As already stated, we want to calculate the degree of approximate convexity for the set X as given in Eq. 3.34. Some of the steps that lead to those results can be found in Appendix B. The two measures $m(L^r \mid L^r \cap R(X) \neq \emptyset)$ and $m(L^r \mid L^r \cap C(X) \neq \emptyset)$ can readily be calculated from Eq. B.17. For the calculation of $m(L^r \mid L^r \cap Q(X) \neq \emptyset)$, however, we have to employ Eq. B.19, in connection with Eq. B.17. Therefore, we can calculate $cv^{n[r]}(X)$ as follows

$$cv^{n[r]}(X) = 1 - \frac{W_r(R(X)) - \frac{\binom{n-1}{r-1}}{\binom{n}{r}} \frac{O_{r+1}}{O_r} W_{r-1}(Q(X))}{W_r(C(X))}, \tag{3.35}$$

which is the most general solution possible. The notation of the above equation is according to Appendix B.

Intuitively, it appears best to use lines as the probing set L^r, that is, $r = 1$. Furthermore, in that case Eq. 3.35 can be given a particularly simple and concise form. We obtain

$$cv^{n[1]} = 1 - \frac{F(R(X)) - 2V(Q(X))}{F(C(X))}, \tag{3.36}$$

where $F = nW_1$ is the area of the hyper-surface, and $V = W_0$ is the volume of the set in question.

3.3.4 Practical Considerations

The main problem that remains to be solved for an application of Eq. 3.36 in practice is how to estimate hyper-surface and volume of sets, particularly in 2D and 3D. For a digitized two-dimensional shape we orient ourselves at the thorough work done by Dorst [15]. If we assume that the contour of a shape consists only of straight lines and circular arcs then Dorst gives several estimators of different complexity and accuracy. I will here briefly describe one such estimator, which is simple to calculate and still yields remarkably accurate results.

The estimator we use here employs different weights for horizontal or vertical and for diagonal links, which for instance can be obtained from the chain-coded boundary. If we call the number of horizontal or vertical links as n_e and the number of diagonal links as n_o, then the length estimator $L_K(n_e, n_o)$ that minimizes the expected error is given as [15]:

$$L_K(n_e, n_o) = 0.948 n_e + 1.343 n_o \tag{3.37}$$

As can be seen from Eq. 3.36, we need to estimate three different entities: the perimeter of the convex hull of X, the perimeter of the concavities, and the length of the cover. The perimeter of the convex hull can readily be estimated from the boundary of the convex hull. For the perimeter of the concavities, however, we cannot simply use the perimeter of the set $R(X) \setminus X$, but we have to obtain the length of the boundary of pixels adjoining $R(X)$ in X, in order to get a coherent estimate even for small concavities. Therefore we are using the set $(R(X) \oplus D_1) \cap C(X)$, where D_1 is the diamond structuring element of size 1. The cover length is then obtained from the set $B((R(X) \oplus D_1) \cap C(X)) \cap B(C(X))$. In practice, it is advisable to disregard small concavities of one pixel size. This can easily be done by a filtering of $R(X)$.

To see why it is necessary to dilate the concavities for obtaining a coherent boundary length estimate, consider the example in Fig. 3.8. Clearly, for small concavities no meaningful boundary can be extracted. We can circumvent the problem by considering that the original shape X is closed and, therefore, the concavities are open (closed and open in the topological sense). Therefore, the boundary of the concavities is equal to the boundary

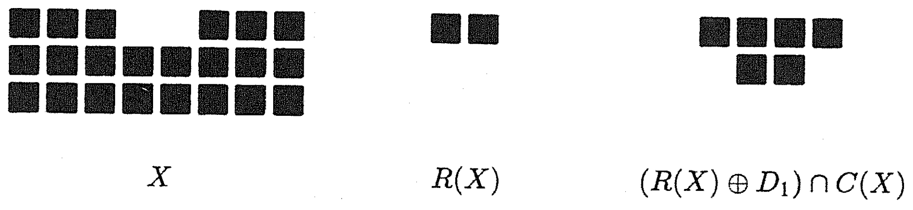


Figure 3.8: Finding the boundary of a concavity

of the original shape X adjacent to the concavities.

Approximate convexity is a rather vague concept and thus is not easily analyzed analytically. If we employ psycho-physical experiments for verifying concepts like approximate convexity then there remains always a doubt as to how far the subjects were influenced by the phrasing of the question and how strong individual differences of the subjects are. Notwithstanding such problems, I attempted to find some comparative evaluation among a set of measures, as outlined below, and with qualitative impressions as obtained from sixteen human subjects. Fig. 3.9 shows some shapes that were used for a comparative

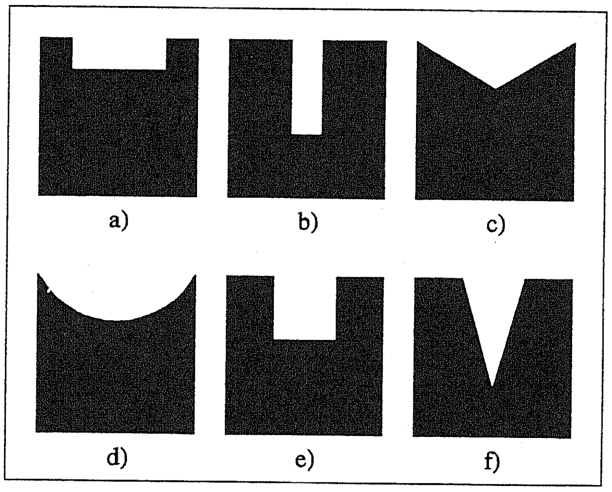


Figure 3.9: Shapes used for evaluation

characterization. The values for Sklansky's two measures S_a and S_d, Boxer's measure pd, Stern's polygonal entropy pe, and the proposed degree of approximate convexity cv are given in Table 3.1. For purposes of reference, the commonly used measure of compactness

P^2/A, where P is the perimeter and A is the area, is also included. In the case of the

Figure	S_a	S_d	pd	pe	cv	P^2/A
a)	0.88	0.8	8.5	0.94	0.9	21.89
b)	0.88	0.41	25.5	0.67	0.7	30.47
c)	0.85	0.7	13	0.97	0.96	20.47
d)	0.78	0.7	13	0.97	0.94	22.8
e)	0.84	0.6	17	0.81	0.8	27.2
f)	0.86	0.3	30	0.68	0.74	29.64

Table 3.1: Values for shapes in Fig. 3.9

human subjects, as the question for a quantitative degree of approximate convexity is too demanding, only an ordering of the six shapes from Fig. 3.9 according to their "convexity" was obtained, averaged over the sixteen subjects. The so obtained ranking is shown in Table 3.2, together with the rankings as obtained from Table 3.1.

The correlation between the human ordering and the orderings for each measure can then be expressed in terms of a distance function d, which is defined as

$$d = \sum_{i=0}^{n} \mid \mathrm{ord}_{\mathrm{measure}}(i) - \mathrm{ord}_{\mathrm{human}}(i) \mid, \tag{3.38}$$

where ord(i) is the ordinal number of the ith shape, either in respect to the human results or the measure in question. The distance d for the set of used measures is also shown in Table 3.2. The results ask for some explanations. As stated earlier, Boxer's *deviation from convexity* is subsumed by Sklansky's measure S_d, this fact is mirrored nicely in the equivalent ordering for the two measures. Although the measures S_d, *pd*, *pe*, *cv*, and the measure of compactness have comparable scores, it is important to keep in mind the several shortcomings they all suffer from. As there exist no objective evaluation criteria to decide which of the measures to use, it is finally left to each prospective applier of such a measure to select the one that suits his or her needs best.

The case in 3D is more complicated than the above calculation, because estimating the area of three-dimensional digitized surfaces is more complicated. However, if we follow the thorough work by Mullikin and Verbeek [38], then it is as well possible to find a simple

Figure	S_a	S_d	pd	pe	cv	P^2/A	Human
a)	1	1	1	3	3	2	1
b)	1	5	5	6	6	6	6
c)	4	2	2	1	1	1	2
d)	6	2	2	1	2	3	3
e)	5	4	4	4	4	4	4
f)	3	6	6	5	5	5	5
Distance d	13	3	3	5	4	2	—

Table 3.2: Obtained rankings and evaluation (see text)

and sufficiently accurate estimate. Unlike the 2D case, in 3D three different situations have to be distinguished. This distinction is done according to the number of adjacent sides of a voxel that are exposed to the background. Neglecting the cases of lines or planes, only configurations of voxels with one, two, or three adjacent sides exposed to the background are possible. This corresponds roughly to voxels in planar areas, voxels in edges, and corner voxels. If we denote the number of those three types by n_1, n_2, and n_3, respectively, then the optimal estimator according to [38] is given as

$$F(n_1, n_2, n_3) = 0.894n_1 + 1.3409n_2 + 1.5879n_3 \tag{3.39}$$

The actual derivation of the three entities used in Eq. 3.36 proceeds in an analogous way to the 2D case. An example of the kind of results obtainable in three-dimensional space is given in Fig. 3.10.

3.3.5 Conclusions

In this section I proposed a new *degree of approximate convexity* and showed that this degree has a natural extension into Euclidean spaces of arbitrary dimensionality. I showed that the initial definition of approximate convexity, if some simplifying facts are assumed, resolves to simple measurements of hyper-surface and volume. Furthermore, I outlined how such measurements could be carried out in discrete spaces of dimensionality two and three, that is, in spaces that are generally of interest. Due to its generality and its simplicity, the

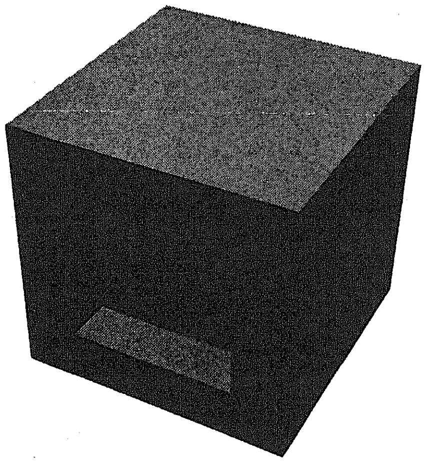


Figure 3.10: Approximate convexity in 3D; $cv = 0.946$

proposed degree of approximate convexity could be a useful tool in many shape description tasks.

3.4 Decomposition

3.4.1 Fundamental Notions

3.4.1.1 Maximal Approximately Convex Subparts (MACS's)

The simple and intuitive mathematical notion of the *convex ring* $\mathscr{G}$ is due to Hadwiger [20]. It is obtained by constructing the class stable under finite unions of convex sets. Therefore by definition

$$X \in \mathscr{G} \Leftrightarrow X = \bigcup_{i=1}^{N} Y_i \qquad Y_i \text{ is convex,} \tag{3.40}$$

that is, a set belongs to the convex ring if and only if it can be decomposed into a finite union of convex sets. Obviously, the convex ring is closed under set union and intersection, i.e.,

$$A, B \in \mathscr{G} \Rightarrow A \cup B, A \cap B \in \mathscr{G}.$$

This makes the convex ring the smallest possible ring over the class of convex shapes. The convex ring further has the advantage that it allows to extend concepts of integral geometry to nonconvex sets. However, here the convex ring is only used as a basic idea for defining the new concept of a *Maximal Approximately Convex Subpart* (MACS). Namely, reversing the above definition of the convex ring, we can say that a shape can be decomposed into a finite union of convex sets if and only if it is a member of the convex ring. The convex ring, therefore nicely summarizes the domain of the proposed approach. Now, we can actually define the notion of MACS as:

Definition 4 *Let X be a bounded subset of $\mathbb{Z}^2$, i.e., of the plane of integers. A MACS M of the set X at a given location $x \in X$ is the maximally expanded, approximately convex subset, such that $M \subseteq X$ and $x \in M$.*

The idea behind approximate convexity, as mentioned in the above definition, is that human beings can actively disregard certain concavities, as long as doing so aids the general understanding of the shape. Therefore, approximate convexity is a necessary concept in shape decomposition, although, so far it has mainly been used in an empirical sense [2, 51]. In Section 3.3 I showed how approximate convexity could be defined and implemented, one important step in the present approach.

3.4.1.2 Skeleton

In many decomposition schemes, the skeleton representation of a shape has been used as a starting point [27, 41]. This choice appears reasonable, as the skeleton already went through a certain kind of smoothing, that is, fine details of the contour are usually not preserved on the skeleton. The question of what skeleton should be used for a certain application has, as yet, not been solved satisfactory and hence any choice has a certain randomness attributed to it. In our case, two different approaches are used. Previous work [27] suggested that the skeleton approach by Fischler and Barrett [17] is probably the best suited one. Reasons for this choice include, among others, the generality of that approach, for instance, any digital distance can be used, and adjustable sensitivity through the detection of so-called critical points on the boundary. For the detection of those critical points the method described in [25] is used. This method has been shown to be rather stable under rotation and scaling, thus making the skeleton rather stable as well.

The second approach, the one by Jang and Chin [28], is more closely related to mathematical morphology. In comparison with Fischler and Barrett's approach, Jang and Chin's method seems to yield somewhat better results for big input shapes. On the other hand, for small shapes Fischler and Barrett's approach achieves a better centering of the medial line. For the examples in Section 3.4.3, if not otherwise stated, Fischler and Barrett's approach has been used.

3.4.1.3 Elongation and Fattening

Elongation, or the expansion of a given set along a predefined axis, is a very intuitive idea and there have been several implementations proposed so far. For instance, Azumatani and Abe [4] use an elongation procedure to connect broken contour lines in maps. More recently, Yamada et al. [63] proposed a method called *Multi Angled Parallelism* (MAP), where twisting operations are used to propagate edges along a given direction. However, both approaches suffer from certain shortcomings. For instance, it is not clear how to extend Azumatani's approach to a more general setting. In the case of MAP, a newly defined picture plane that a priori specifies the possible resolution of directions is necessary.

In contrast to other approaches, the method proposed here is based directly on Hadwiger's definition, as given in Section 3.2. As the definition implies, a candidate for elongation has to have a distinct main axis, which, in our case, can a priori be obtained by extracting approximately straight parts of the skeleton. Hence, the object to be elongated is obtained by dilating the just mentioned skeleton segments according to some width label.

Given an approximately straight skeleton segment seg_i we extract its starting point (x_a, y_a) and its end point (x_e, y_e). Then, we extract all the pixels on seg_i, and apply a least-square fit to obtain the equation of the straight chord that fits the segment best, i.e.,

$$y = m\,x + c.$$

The case where $m \rightarrow \infty$, that is, a vertical line, can easily be accommodated by rotating the coordinate system. Using the above equation, we can extrapolate the initial skeleton segment beyond its original dimensions. This can be done as follows. If $| \, m \, | \leq 1$, we have to ensure that $x_a < x_e$. Then the coordinates on the chord at integer intervals are given as

$$\left.\begin{aligned} x_k &= x_a + k \\ y_k &= \lfloor m\, x_k + c + 0.5 \rfloor \end{aligned}\right\} k = 0, 1, 2, \ldots \tag{3.41}$$

At each position k on the chord, we can calculate an x and a y vector i_x, i_y, respectively, such that $i_x, i_y \in \{-1, 0, 1\}$ and i is the nearest discrete approximation to the chord at position k, hence

$$\left.\begin{aligned} {i_x}^{(k)} &= 1 \\ {i_y}^{(k)} &= \lfloor m\, x_k + c - y_{k-1} + 0.5 \rfloor \end{aligned}\right\} k = 1, 2, \ldots \tag{3.42}$$

For the case where $| \, m \, | > 1$, ${i_y}^{(k)}$ becomes 1 and ${i_x}^{(k)}$ can be calculated accordingly. Now, given the minimum width label ω_{min} on seg_i, we derive the following sets

$$\begin{aligned} B_a &= \{x_a, y_a\} \oplus \omega_{min}\, B \\ B_e &= \{x_e, y_e\} \oplus \omega_{min}\, B \\ S_i &= seg_i \oplus \omega_{min}\, B \end{aligned} \tag{3.43}$$

Using these initial definitions, the actual elongation proceeds according to the NS chart in Fig. 3.11, where $\mathcal{L}$ stands for the area.

For the break condition it is necessary to know by how much the segment S_i would ideally grow with each step. For the case of an octagonal structuring element B the growth ΔS can be calculated as

$$\begin{aligned} \Delta S &= 2\,\omega_{min} + 1 && \textit{if } {i_x}^{(k)} = 0 \textit{ or } {i_y}^{(k)} = 0 \\ &= 2\,\omega_{min} + 1 + 2\lfloor \omega_{min}/2 \rfloor && \textit{otherwise.} \end{aligned} \tag{3.44}$$

Clearly, the achievable resolution for elongation is not a priori fixed, as, for instance, in the case of MAP where the number of possible directions is directly encoded in the data structure, but depends solely on the size of the object.

Forward pass

```
$X_0 = B_e$
    $X_i = \left(X_{i-1}\ {}_{i^{(k)}}\right) \cap F$
    $S_i = S_{i-1} \cup X_i$
until $(\mathcal{L}(S_i \setminus S_{i-1}) < \Delta S)$
    and $(\mathcal{L}(S_{i-1} \setminus S_{i-2}) < \Delta S)$
$S = S_{i-2}$
```

Backward pass

```
$X_0 = B_a$
    $X_i = \left(X_{i-1}\ {}_{-i^{(k)}}\right) \cap F$
    $S_i = S_{i-1} \cup X_i$
until $(\mathcal{L}(S_i \setminus S_{i-1}) < \Delta S)$
    and $(\mathcal{L}(S_{i-1} \setminus S_{i-2}) < \Delta S)$
$S = S_{i-2}$
```

Figure 3.11: Algorithm for directional dilation

A concept that is very closely related to elongation is *fattening*. The term fattening simply refers to the dilation of a set by two anti-parallel vectors normal to the main axis of the set. Although the definitions of elongation and fattening are almost identical, the same is not true for the implementation. Obviously, in the case of fattening it is not possible to define a disk, having the length of the set as its diameter, and which is completely contained in the set. Therefore, instead of propagating a disk, we simply propagate the whole set along the two anti-parallel vectors, which can easily be obtained from Eq. 3.41.

3.4.2 Algorithm

The general overview of the algorithm for decomposing a given binary shape into MACS's is given in the flowchart in Fig. 3.12. In this section, I will explain each step separately, as they can be implemented as modules that receive some input and hand on some output.

To further illustrate the algorithm, we can look at an actual example, which allows to directly see the results of each step separately. The chosen input, together with its skeleton representation is shown in Fig. 3.13.

3.4.2.1 Skeleton Decomposition

Starting from a binary object, hereafter referred to as F, we select a set of conveniently located seeds $S = \{S_0, S_1, \ldots\} \subset F$, whose members S_i are assumed to be convex. Using

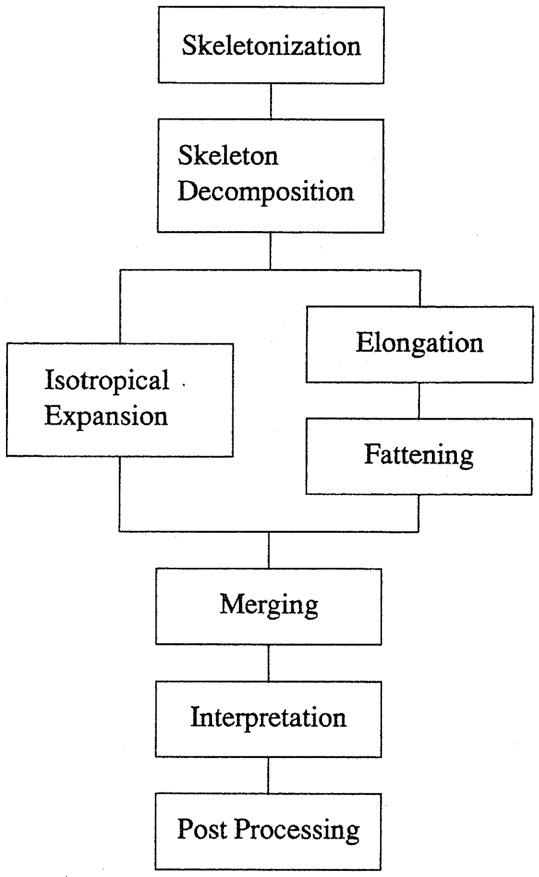


Figure 3.12: Flowchart of the decomposition

the skeleton representation of F, as explained in Section 3.4.1.2, we first prune the skeleton by removing branches that are shorter than a given threshold t_b and then cut the pruned skeleton at its branching points. How to select the mentioned threshold t_b is a general problem of skeleton pruning; here we empirically set it to 4, that means we only want to remove rather short branches of the skeleton. Next, we further investigate each branch, using a dominant point detection scheme [25] adopted for open curves, and decompose the branches into straight parts by cutting them at detected dominant points. Finally, we examine all the width labels in one straight part of the skeleton; if the standard deviation of the labels is above a certain threshold then we determine a cut point, using a method by

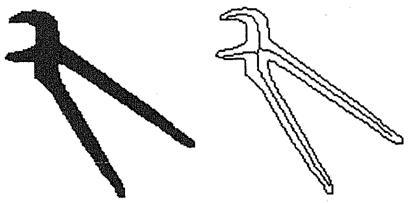


Figure 3.13: Sample input shape with its skeleton

Otsu [40] for the thresholding of bimodal histograms. Left and right of the cut point, we have two smaller segments, each having a minimum width label. Instead of simply cutting the skeleton segment at the found cut point, we retain the whole segment, relabelled with the smaller width label, as well as reproducing the part having the bigger width label, which then will overlap the segment with the smaller width label. This procedure allows for a more accurate elongation, as explained in the next section.

For each of the so obtained skeleton segments seg_i we extract the minimum width label ω_{min} and dilate the segment by an octagonal structuring element B of size ω_{min}, that is

$$S = \{S_i \mid S_i = seg_i \oplus \omega_{min} B\}. \tag{3.45}$$

Therefore, the set S is a priori contained in F and a priori fulfills our convexity constraint.

The resulting set of expanded skeleton segments for the input shape from Fig. 3.13 is shown in Fig. 3.14.

3.4.2.2 Elongation

In the next step, we apply the elongation procedure, as outlined in Section 3.4.1.3, to all those subparts S_i that meet the elongation constraint. The elongation constraint simply states that the segment has to consist of a distinct line segment, in practice, we adopted the convention that it has to be longer than three pixels. Only then can we guarantee that the directional information is meaningful and can be used for elongation. Thus we are obtaining the maximally elongated parts L_i with respect to the initial seeds S_i., i.e.,

$$L_i = \widetilde{S_i}^{\lambda}. \tag{3.46}$$

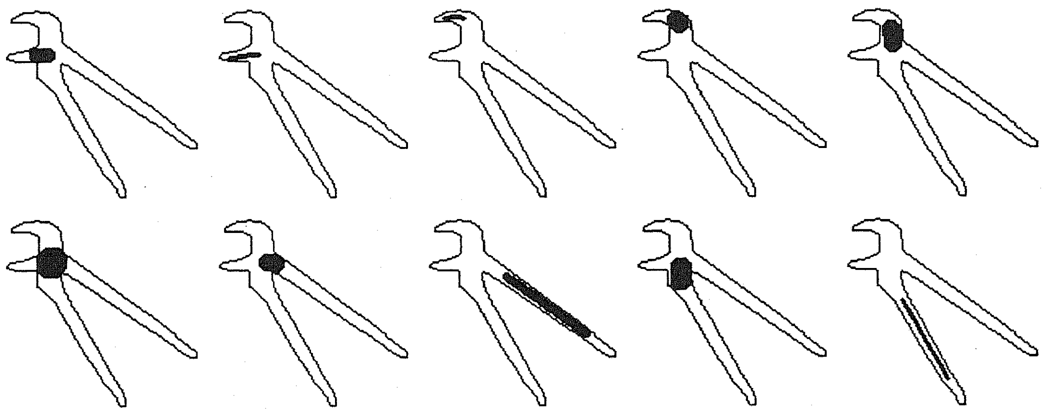


Figure 3.14: Expanded skeleton segments

The choice for the parameter λ, that is, the length of the vector with which to elongate, will in general depend on the application. Although an a priori choice is imaginable, more likely, as in our case, λ will be determined adaptively. That means, when the elongation propagates beyond the original object, the procedure stops.

In order to avoid unnecessary computation during the following steps, it is checked whether there are any redundant segments present, that is, segments, whether they were elongated or not, which are fully contained in any other segment. If yes, they are deleted, thus leaving the final set L of elongated subparts

$$L = \{L_i \mid L_i = \widetilde{S_i}^{\lambda} \; or \; L_i = S_i, \; L_i \not\subset L_j \, for \; any \; i \neq j\}. \qquad (3.47)$$

Elongation in the case of the pair of tongs produces the result as shown in Fig. 3.15. Especially noteworthy is how the two handle segments have been propagated across the whole length.

3.4.2.3 Fattening and Isotropical Expansion

To all the segments that did not meet the initial elongation constraint as outlined in the previous subsection, we apply an isotropical expansion. This is done by dilating those segments L_i with an octagon of monotonously increasing size τ, i.e., τB, and intersect the dilate with the original object F (conditional dilation). After intersecting, we test

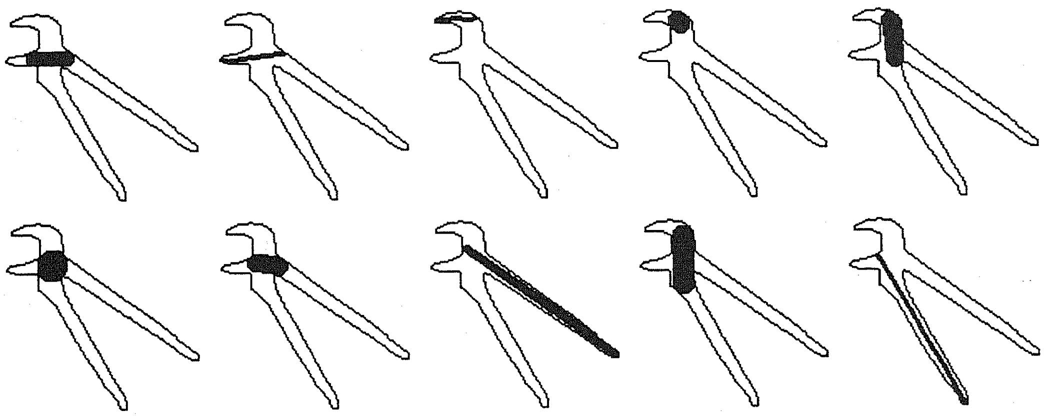


Figure 3.15: Elongated segments

whether the result is still approximately convex, as outlined in Section 3.3. The threshold for approximate convexity was obtained empirically by comparing the resulting expansions with human intuition. In most of the experiments shown in Section 3.4.3, the value is set to 0.99, however, depending on the domain of application, this threshold might have to be adjusted. If the measure of approximate convexity of the expanded segment is below that threshold, the last expansion step is discarded and the expansion of that segment stops. If it is equal or bigger, expansion continues with the next bigger octagon. This can be formalized as

$$C_i = (L_i \oplus \tau B) \cap F.$$

All the segments that passed through at least one elongation step, are candidates for fattening, rather than isotropical expansion. The idea behind this distinction is that segments that have been elongated have exhausted their expansibility along the main-axis. Further isotropical expansion, however, would disregard this and lead to a new expansion along that axis, a fact that can often lead to the appearance of undesirable concavities. As outlined in Section 3.4.1.3, we dilate the segment by two anti-parallel vectors of length one at the same time, hence the segment will be expanded symmetrically to the main axis. After one fattening step, we check for the approximate convexity of the resulting segment in the same way as outlined above. If the check fails, the last fattening step is discarded, otherwise we continue. Fattening will be denoted by $\widetilde{L_i}^{\perp\lambda}$ where $\perp\lambda$ stands for the unit vector normal to the main axis of the set L_i.

After carrying out either fattening or isotropical expansion on all segments, again redundant segments are discarded, using the same method as after the elongation step. The whole expansion can thus be summarized as

$$\begin{aligned} C = \ & \{C_i \mid \ C_i = \widetilde{L_i}^{\perp\lambda} \cap F \textit{ or } C_i = (L_i \oplus \tau B) \cap F, \\ & \qquad C(C_i) \approx C_i, \ \ C_i \not\subset C_j \textit{ for any } i \neq j\}. \end{aligned} \tag{3.48}$$

The example in Fig. 3.16 nicely outlines the effects of expansion and fattening. If we again concentrate on the handles, we can see that they now occupy their whole width.

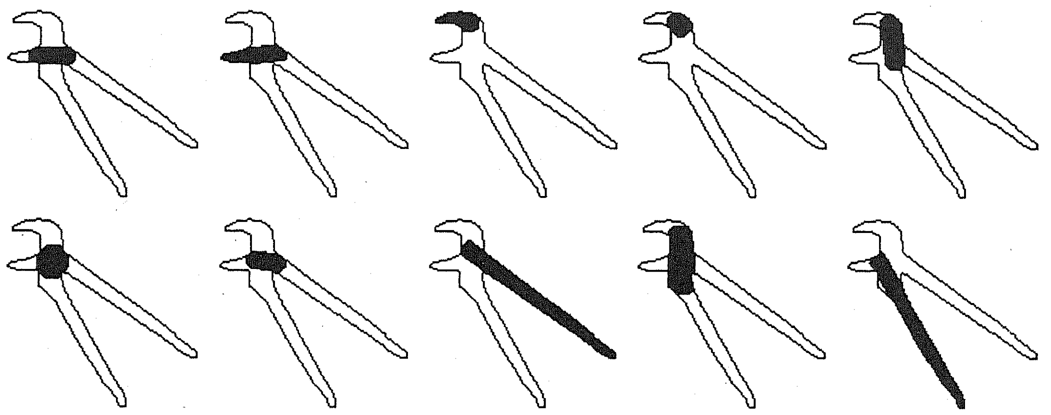


Figure 3.16: Expanded and fattened segments

3.4.2.4 Merging

In the next step the geometrical relations between the members of C are investigated. This is done by first constructing a relation graph r, which specifies which segments overlap at least partially:

$$\begin{aligned} r_{ij} &= 1 \ \ \textit{if } C_i \cap C_j \neq \emptyset \quad (\textit{including } i = j) \\ r_{ij} &= 0 \ \ \textit{otherwise}. \end{aligned} \tag{3.49}$$

According to this relation graph r, we try to merge segments according to a best-fit principle. For each segment C_i, we find all segments C_j such that $r_{ij} = 1$. We tentatively merge all those pairs C_i, C_j and check for approximate convexity. As at that stage the conditions should be a little bit looser, the threshold for approximate convexity is set to 0.98. Again, this threshold was obtained empirically and could be adjusted for different domains of application. Among all pairs C_i, C_j the one with the highest value of approximate convexity is found. If this value is equal to or above threshold, the two segments are merged, segment C_j is discarded, and segment C_i is replaced by the merge result. This procedure is repeated as long as there are any pairs C_i, C_j whose value of approximate convexity permits a merging. The result of the merging can be summarized as follows

$$M = \{M_i \mid M_i = C(C_i \cup C_j) \cap F,\ C(C_i \cup C_j) \approx C_i \cup C_j,\ r_{ij} = 1\}. \qquad (3.50)$$

Merging is usually the step that allows to remove most redundant segments. This fact is shown in Fig. 3.17, which defines the major parts of the pair of tongs.

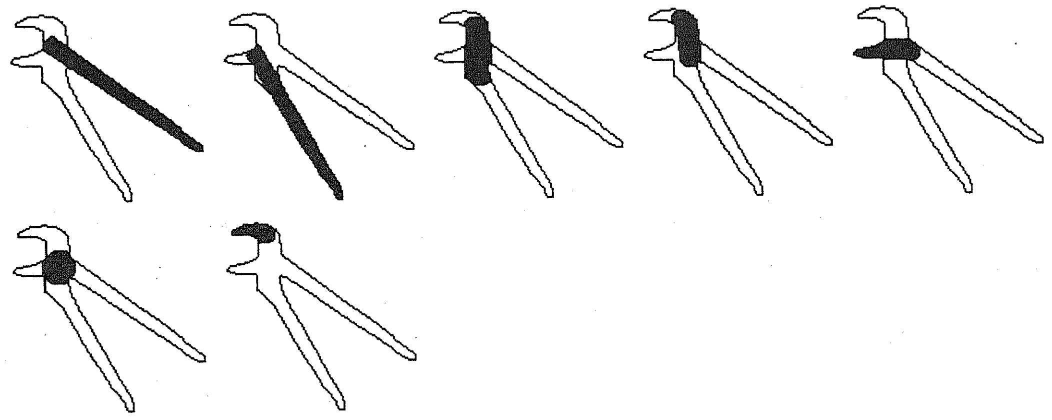


Figure 3.17: Merged segments

3.4.2.5 Interpretation

The set M of MACS's that are obtained at this stage still contains, in general, some redundancies. The decomposition can now be cleaned by assuming that the information content is constant for all MACS's defined over the same image plane. Hence,

Definition 5 *Among all possible combinations of MACS's covering the initial object completely, the best decomposition is the one with the least number of MACS's.*

In practice, however, we have to relax one aspect of the above definition, namely, that the initial object F should be covered completely. Actually it might happen that such a covering will not be possible for any combination of MACS's. This is for instance the case when small details on the contour already have been missed by the skeleton. To avoid a breakdown in such a case, we introduce a reference set F', given as

$$F' = \bigcup \{M_i \mid M_i \in M\} \tag{3.51}$$

and then substitute F' for F in the above definition, ensuring thus that there will always be a solution for the interpretation. In addition, all the pixels lying on the boundary of F, i.e., $B(F)$, are disregarded. This allows to compensate for inaccuracies of the skeleton representation in parts of the shape with even width.

The interpreted decomposition of F in terms of MACS's, optimized with respect to the above definition, is therefore given by

$$D = \{M_i \mid M_i \in M,\ \mathcal{L}\left((F \ominus B_1) \cap (F' \setminus \cup M_i)\right) = 0,\ \mathcal{N}(\cup M_i) = \mathit{minimum}\}. \tag{3.52}$$

where $\mathcal{N}(\cdot)$ stands for the number of MACS's used for the covering, $\mathcal{L}$ denotes the area, and B_1 stands for the unit disk, or for the diamond structuring element, as shown on page 42, in the digital case.

Interpretation, in the case of our example of the pair of tongs, is able to remove all redundant segments, leaving the decomposition as shown in Fig. 3.18.

3.4.2.6 Post Processing

It is possible to reduce the number of produced segments further by introducing a significance factor for each segment. The significance factor of a segment is a quantitative measure of the importance of a segment in respect to the union of all other segments. Namely,

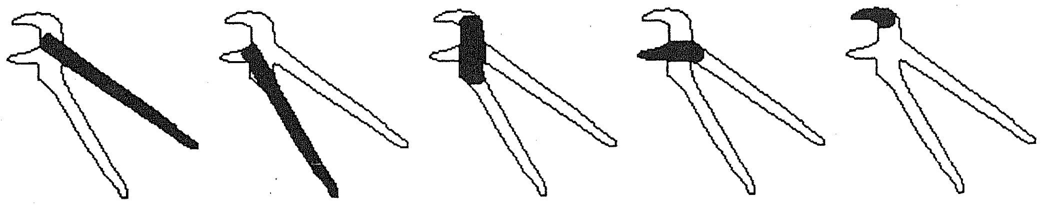


Figure 3.18: Segments after interpretation

Definition 6 *The significance factor of a segment is the ratio of the area covered by this segment only, divided by the area of the segment.*

Hence, the significance factor sf_k for a certain segment k can be calculated as

$$sf_k = \frac{\mathcal{L}(M_k \setminus (M_k \cap (\cup M_i)))}{\mathcal{L}M_k} \quad \forall i \neq k. \tag{3.53}$$

The post processing procedure is as follows. First the significance factors for all segments are calculated and the segments are ordered according to their significance factors. If the value of the minimal significance factor is below a certain threshold, then the corresponding segment is removed and the whole procedure is repeated. If the minimum significance factor is not below the threshold, then the post processing procedure stops, returning the remaining segments, together with their updated significance factors. For all examples shown, a preset threshold of 0.05 has been used. The meaning of this threshold is that a segment should at least contribute with 5 % of its area to the present decomposition, otherwise it is not considered significant. Although no exhaustive experiments with this threshold have been carried out, it appears conservative enough to be acceptable.

In the case of our example, post processing does not have any effect, leaving the decomposition from Fig. 3.18 as the final result. The decomposition results are intuitive and capture the idea of the used tool well. To further underline the performance of the algorithm, in the next section I will show some more extensive experiments.

3.4.3 Examples

The decomposition of a shape into convex subparts is, in general, not unique (see [20] for a proof). Therefore, there is no tool for the analytical evaluation of such decompositions and we have to resort to intuitive evaluation. To aid such an intuitive evaluation, I have chosen examples that a human observer would usually decompose in a unique way, called *ideal decomposition*, against which the results of the algorithm can then be evaluated. The examples shown in Figs. 3.19 – 3.24 have been digitized on a 128×128 pixel picture plane. The occlusion scene from Fig. 3.25 has a size of 200×200 pixels. Each figure shows the input (top left), the skeleton representation (top right), and the final decomposition (bottom). The results of the decomposition are shown in Table 3.3. The last row of this table indicates the processing time in seconds. The algorithm was coded in C, compiled with the optimization flag, and executed on a SUN S-4/EC.

Figure	Skeleton segments	After elongation	After expansion	After merging	Inter-pretation	Final decomp.	Ideal decomp.	Proc. time [sec]
Cross	13	11	9	3	2	2	2	27.32
Screwdriver	4	4	2	2	2	2	2	6.57
Tongs	15	14	14	8	5	5	5	41.35
Plane I	14	14	13	4	4	4	4	42.48
Plane II	14	11	10	5	5	5	5	23.03
Occlusion I	25	23	21	10	7	7	7	70.48
Occlusion II	52	52	51	28	16	13	13	839.73

Table 3.3: Decomposition results

The first example in Fig. 3.19 shows the synthetic example of a cross, whose skeleton has been decomposed into 13 initial segments. This example especially highlights the merge stage, during which 6 segments can be removed. The final decomposition nicely shows the two bars that are the basic constituents of the cross. Next are some real examples of tools, starting with the driver in Fig. 3.20. In the case of the driver, the skeleton decomposition is rather simple and yields only 4 segments, two of which can be removed subsequently, yielding the final decomposition into blade and handle. Fig. 3.21 shows a pair of tongs, with an initial decomposition into 15 segments. The final decomposition very nicely captures the general structure of the tool, with the two handles, the two front parts, and the spring used to open the pair of tongs. The next two examples in Fig. 3.22 and Fig. 3.23, show the

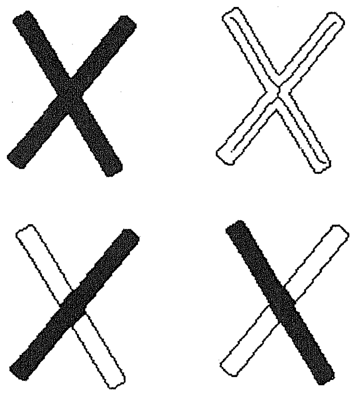


Figure 3.19: Rotated cross

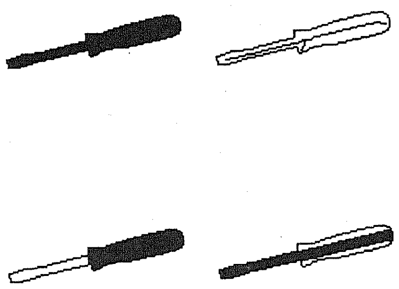


Figure 3.20: Screwdriver

decomposition of the silhouettes of planes. The numbers of the final segments correspond in both cases to the ideal decomposition, and the shape of the segments in Fig. 3.22 are close to ideal. In Fig. 3.23, which shows a two-engine plane, all the meaningful parts are extracted and can be identified as wings, fuselage, tail-wings, and the two engines. In Fig. 3.24 we applied the algorithm to an occlusion scene of a driver and a pair of tongs, starting with a skeleton decomposition into 25 segments. The final decomposition into 7 segments again corresponds nicely to the ideal decomposition, as all the elements of the driver and the pair of tongs are identified. However, the shape of the third segment is not captured perfectly. This problem can be traced to the used approximation to the circle

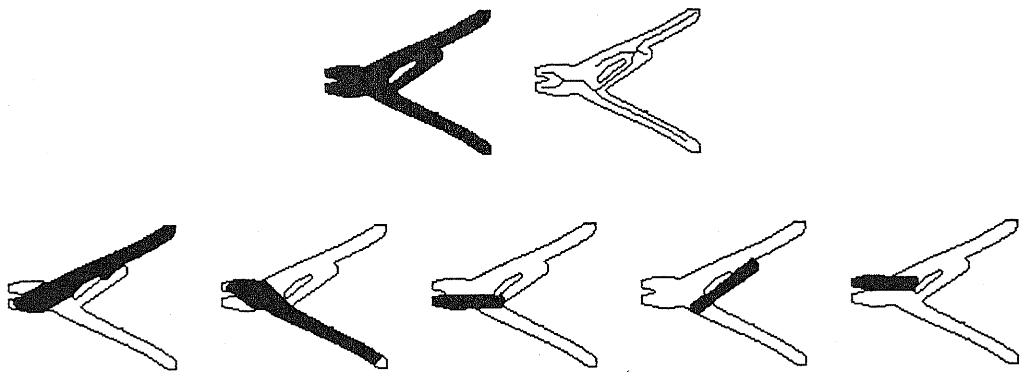


Figure 3.21: Pair of tongs

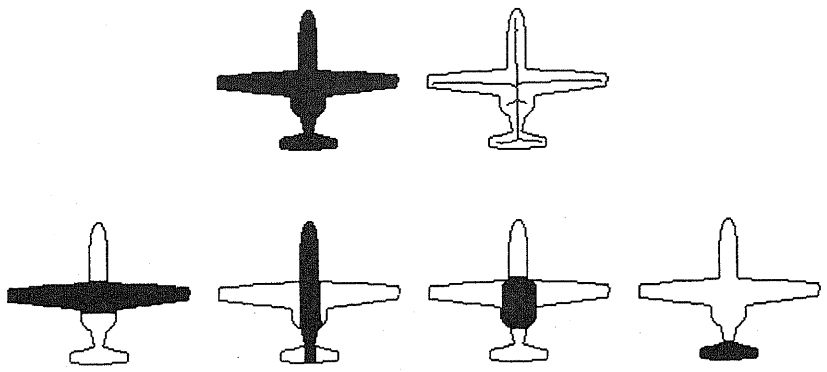


Figure 3.22: Plane I

while calculating the convex hull, and could be remedied by using the exact convex hull. Finally, in Fig. 3.25 we show an example of a more complicated occlusion scene. Initially, the skeleton has been segmented into 52 parts. If the number of segments is getting rather large, then the need for the interpretation stage and the post processing becomes clear. In this example 12 unnecessary segments were removed only during interpretation, while post processing allowed to remove three more segments. The obtained segments do not completely correspond to the ideal decomposition, although the number of segments is equivalent. This is because the head of the uppermost hammer has been segmented into two parts, and the upper screwdriver has been over merged. It is arguable whether the constraint

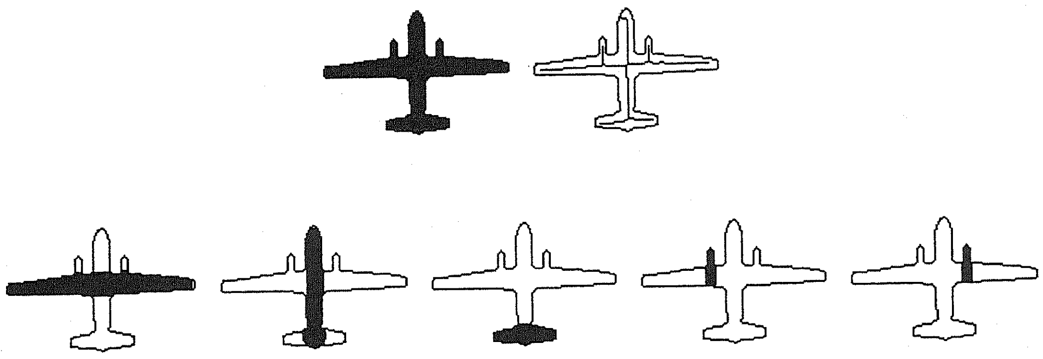


Figure 3.23: Plane II

of approximate convexity is too strong in the first case. Alternatively, a further stage for the extraction of "convex arcs", which could be defined as a convex shape with a bent skeleton, could be implemented. The over merging of the screwdriver could again be remedied by employing the exact convex hull.

Considering processing time, we can see from Table 3.3, that processing gets rather slow for complicated scenes. One reason for this is the work spent on calculating convex hulls. For instance in the case of Fig. 3.25, 356 convex hulls have to be computed. In general, some speedup should be possible by an optimized coding of the algorithm, a point that has mainly been neglected so far. Particularly, by using dedicated hardware for morphological operations, it should be possible to bring down the processing time to more favourable values. To gain further increases in speed, the possibility of pruning candidate segments during the first stages of the decomposition, e.g., after skeletonization, should be investigated. However, there is a trade-off between such a pruning and the stability of the decomposition, which might result in the loss of some meaningful segments.

3.4.4 Conclusions

I proposed the concept of MACS's into which a binary shape can be decomposed, and showed how this concept can be related to integral geometry. I used the theoretical notion

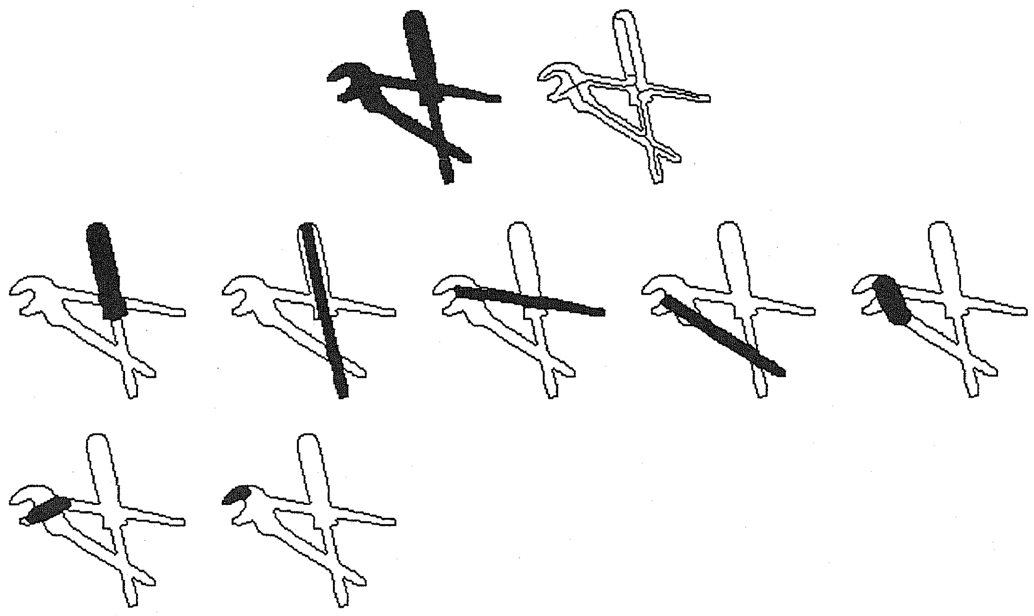


Figure 3.24: Occlusion scene I

of MACS's as a centerpiece for a novel shape decomposition and description scheme, using concepts of mathematical morphology to obtain an algorithm that is open to evaluation and to implementation in hardware. To overcome the problem of rotation invariance, I proposed a new elongation operator, which allowed to implement the *Law of Good Continuation* of Gestalt psychology directly. Comparing the present work with results from other recent shape decomposition approaches, for instance, Pitas and Venetsanopoulos [44], shows clearly that our results are much closer to human intuition. Keeping in mind that our approach is purely bottom-up, the used examples show clearly that all the available information has been used in an intuitive and coherent way. Therefore, given no prior knowledge about the expected input, the notion of MACS's seems to be usable as a major element in shape decomposition.

In order to obtain a more generally useful algorithm, it would be necessary to employ parallel or dedicated hardware, as already mentioned. This could quite easily be done if we consider that the work on each segment, as initially obtained from the skeleton

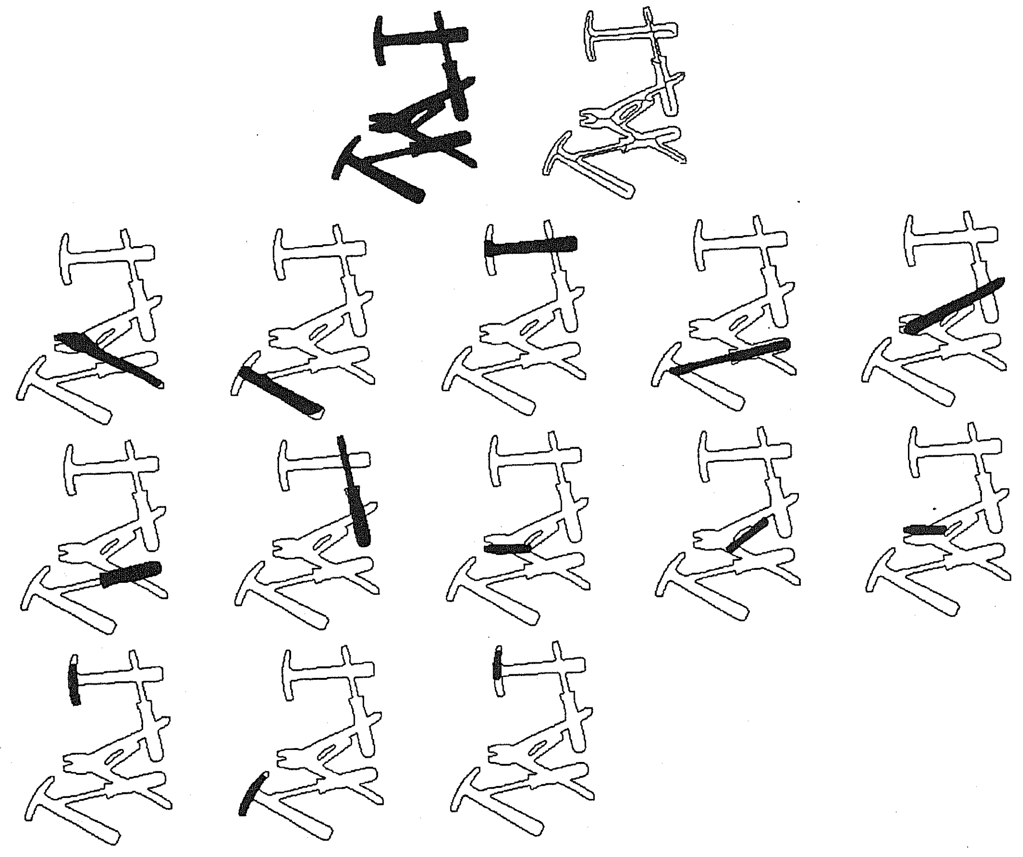


Figure 3.25: Occlusion scene II

decomposition, could be carried out in isolation. Only in the case of the merging stage, a slightly more elaborate scheme would be needed. The removing of redundancies as done between two consecutive processing stages could then be used as a synchronization step. In this manner a significant speedup should be achievable.

As the present algorithm is mainly based on ideas of mathematical morphology, it should also be possible to extend it to the three-dimensional case. Work on three-dimensional skeletons can be found in the literature and the extension of elongation, expansion, etc., to three dimensions is straightforward. However, in this case it is of special importance to use parallel or dedicated hardware, as the implementation of three-dimensional morphological algorithms on a normal workstation is at best very ineffective.

For a general shape decomposition scheme, however, the notion of MACS's alone does not seem to be sufficient. Additional elements, as for instance loops or loop segments, corresponding to the above mentioned "convex arcs", have to be added to the process so as to yield a more general and versatile approach.

CHAPTER 4

Matching

4.1 Purpose and Overview

As already explained in Chapter 2, the basic structure used for acquisition and generalization of concepts is that of a hierarchical graph. The actual relations among parts of the decomposition are stored in the network layer. The way these relations interact so as to form multiple concepts is mirrored in several layers that connect the relations hierarchically. Hence, we can speak of the conceptual network as a three-dimensional graph structure.

In this Chapter I will address two main points. The first is how to build the conceptual network from the result of the decomposition. In other words, in Sections 4.2 and 4.3, I will first address how an instance can be translated into a conceptual network. Once the conceptual network of an instance has been obtained, generalization proceeds by matching the conceptual networks of two or more instances. Hence, I will address the problem of graph matching or subgraph isomorphisms. First, in Section 4.4, I will briefly review some approaches towards weighted graph matching, giving the connection to the present application. Finally, in Section 4.5, I will outline the matching approach that has actually been used in the present system.

4.2 Relational Network

The description of binary shapes into Maximal Approximately Convex Subparts (MACS) has been described in detail in Section 3.4. As result of the decomposition we obtain a set of strokes or blobs that is the starting point for obtaining the relational network layer of the conceptual network. In the first step, for each of those blobs or MACS's, a table is created which contains all the important information about the structure and the geometry of the segment. In the second step, the actual relational network is built by using the above mentioned MACS information tables.

4.2.1 Describing MACS's

In order to exhaustively describe a MACS and its relations to other MACS's, the following information is extracted for each MACS:

- MACS number
- Numbers of MACS's with which intersection is nonempty
- Area
- Centre of gravity
- Length
- Width
- Direction of main axis
- Significance factor

Let us assume that each MACS is given as a binary image. Then most of the used relations can be extracted by using moments, e.g. [5]. Generally speaking, the moment of order $(p+q)$ is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y), \tag{4.1}$$

where $f(x,y)$ is the image intensity at a point (x,y). For instance, in the case of binary images $f = 1$ or $f = 0$. Therefore, the area of a binary image is given by m_{00}. The centre of gravity $(\overline{x}, \overline{y})$ can then be calculated as

$$\begin{aligned} \overline{x} &= \frac{m_{10}}{m_{00}} \\ \overline{y} &= \frac{m_{01}}{m_{00}}. \end{aligned} \tag{4.2}$$

Based on the calculation of the centre of gravity, so-called central-moments are defined as

$$\mu_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q f(x,y). \tag{4.3}$$

Using these central moments, the direction of the main axis of the segment can be calculated as follows

$$\phi = \frac{1}{2} \tan^{-1} \left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right). \tag{4.4}$$

The length of the MACS can now easily be obtained by drawing a line with direction ϕ through $(\overline{x}, \overline{y})$ and intersecting it with the MACS. The width can then be approximately obtained by dividing the area of the MACS by its length. The set c_k of all MACS that have a nonempty intersection with the MACS k is given as

$$c_k = \{ i \mid \text{MACS}_k \cap \text{MACS}_i \neq \emptyset,\ k \neq i \}. \tag{4.5}$$

Finally, the meaning and calculation of the significance factor have been explained in Section 3.4. To avoid extreme mismatches, however, the significance factor is not used as is, but is transformed into the range 0.5...0.8, as otherwise the fluctuations of the significance factors would distort the resulting matching too much.

For instance, in the case of the driver from Fig. 3.20, which was decomposed into handle and blade, the following information for the two segments can be extracted.

```
Segment 1 ####################    Segment 2 ####################
  intersect : 2;                    intersect : 1;
  area      : 802                   area      : 708
  gravity-x : 88                    gravity-x : 66
```

```
    gravity-y : 59                     gravity-y : 64
    length    : 58                     length    : 106
    width     : 13                     width     : 6
    direc     : 12                     direc     : 12
    signif    : 665                    signif    : 643
##############################    ##############################
```

For the sake of simplicity, all entries have been rounded to integers. The direction is given in degrees, within $\pm 90°$, and the confidence factor is in parts of thousand. The segments are ordered according to area; segment 1 corresponds to the handle and segment 2 corresponds to the blade of the screw driver.

Based on this description, each MACS can be approximated by a rectangle with direction ϕ and centre at $(\overline{x}, \overline{y})$. For the case of the screwdriver, the approximation as shown in Fig. 4.1 can be obtained. Whenever a reference to the decomposition results will

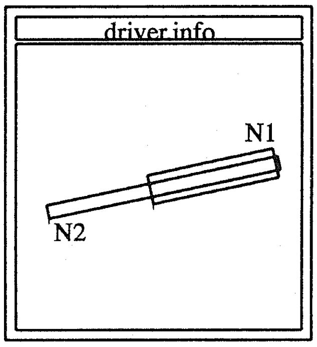


Figure 4.1: Approximated decomposition of a screw driver

be necessary, hereafter, I will be using the above approximative representation.

4.3 Describing Relations between MACS's

As its name already implies, the relational network mainly represents relations between segments of the decomposition, that is, between MACS's. Structurally, nodes of the relational network correspond to MACS's, while links correspond to relation between

MACS's. All of the information in a relational network, except structural information, is concentrated in its links. Links are connecting those MACS's, which intersect each other. Therefore, only the relations between intersecting MACS pairs will be used for constraining the concept structure, while relations between non intersecting MACS will not be used. The main purpose of using relations instead of the actual segment information is that relations can easily be modified to be rotational and translational invariant. This means, we do not want to distinguish between an object and its rotated or translated version. The choice of appropriate relations or relational predicates has to be guided by those invariances. Furthermore, relational predicates should be sufficiently adequate and relevant to the object domain. Beyond those constraints, the choice of relational predicates is up to a certain degree arbitrary. For the present study four different predicates are used. They are a length predicate, a width predicate, a direction predicate, and an eccentricity predicate. The length predicates describe the ratios of the lengths of two segments. Similarly, the width predicates describe the ratios of the widths. Further, the direction predicates give the difference in angles of the main axes, and the eccentricity predicates give a normalized measure for the distance of the centers of gravity. Formally, we have the following relations between two MACS's n_1 and n_2

$$\begin{array}{lll}
\text{Length} & t_1 & = \log(l_2/l_1) \\
\text{Width} & t_2 & = \log(w_2/w_1) \\
\text{Direction} & t_3 & = \mid \phi_1 - \phi_2 \mid \\
\text{Eccentricity} & t_4 & = \dfrac{\left(\sqrt{(x_3-x_1)^2+(y_3-y_1)^2}+\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}\right)\cos\left(\frac{t_3}{2}\right)}{\max(l_1,l_2)}
\end{array}$$

where l_1, l_2 are the lengths, w_1, w_2 are the widths, ϕ_1, ϕ_2 are the directions of the main axes, measured from the x-axis, and (x_1, y_1), (x_2, y_2) are the centers of gravity of any two intersecting MACS, and (x_3, y_3) is the point where the two main axes intersect. The definition of eccentricity corresponds to the distance of two parallel lines through the centers of gravity, in direction of half the angle between the two segments. If we invert directionality, that is, exchange n_1 and n_2, then the length and the width relation will change their signs, while the direction and the eccentricity relations remain unchanged. The logarithms in the case of length and width relations are employed in order to restrict the domain of t_1 and t_2; the domain of t_3, i.e., the direction relation is restricted to $-90° \dots 90°$. Each relation between two MACS's is expressed by the above four relational predicates, each of which, in turn, is a simple number. Furthermore, due to the definition of the length and the width predicate, a relation between two MACS, and thus a link of the relational network, is directed. We must account for this directionality when matching two relational networks.

The system I am describing here should finally be able to learn and to recognize objects solely based on information as obtained from instances of the object. Hence, we do not want to have any assumptions of ideality concerning instances used for learning. This means that the relational predicates as defined above might be rather inaccurate for certain, probably distorted instances. Therefore, I propose to *fuzzify* the relational predicates into generalized predicates. Each generalized predicate does not consist of a number, but rather of a function that can be considered a scaled version of a probability distribution. A generalized predicate is obtained by applying a point-spread function of the form of a Gaussian to the relational predicates. Doing this, the generalized predicates can be written as

$$f_i(z) = \frac{a_i}{\sqrt{2\pi}\sigma_i} \exp^{-\frac{1}{2}\frac{(z-t_i)^2}{\sigma_i^2}} \qquad i \in 1, \dots 4. \tag{4.6}$$

In other words, the relational predicates $t_1, \dots, t_4$ become the expectations of the Gaussian distributions. It is important to note that the above equation does not represent a Gaussian normal distribution, but rather a scaled version of Gaussian or normal distribution. We can interpret the scale factor a_i as mirroring some sort of confidence we have in the expectations $t_1, \dots, t_4$. Therefore, we can use the significance factor as obtained from the decomposition as scaling factor of the Gaussian normal distribution. Finally, the standard deviation σ_i is retained as a system variable. The proper selection of σ_i involves a tradeoff between accuracy and speed. The smaller σ_i is, the more the search-space for possible matchings can be reduced, thus probably eliminating valuable candidates for matching.

Although initially each generalized predicate contains only one Gaussian normal distribution, this might not be true anymore during generalization of several instances.

Depending on the used generalization rules, as for instance the rules to be shown in Chapter 5, Gaussians might get added. Therefore we need to further generalize the notion of generalized predicates so as to accommodate generalization as well. Doing so leads to the following structure of the generalized predicates

$$f_i(z) = \sum_j \frac{a_{ij}}{\sqrt{2\pi}\sigma_{ij}} \exp^{-\frac{1}{2}\frac{(z-t_{ij})^2}{\sigma_{ij}^2}} \qquad i \in 1, \ldots 4, \tag{4.7}$$

where the summation is over all Gaussians that belong to the generalized predicate in question. Now, the case where $j = 1$ corresponds to Eq. 4.6, such that $a_{i1} = a_i$, $\sigma_{i1} = \sigma_i$ and $t_{i1} = t_1$. The calculations for the case where $j \neq 1$ will be shown in Chapter 5.

Once the generalized predicates have been obtained, we can proceed to the building of the relational network. Each link of the relational network contains the above mentioned generalized predicates. In addition each link has a so-called *confidence value*, or *cval* for short. The confidence value will be used during generalization, initially, it is set to one. Nodes in the relational network correspond to segments of the decomposition, however, without carrying any information.

Finally, the conceptual network consists of the derived relational network layer and one entry in the concept layer. The entry in the concept layer stands for the name of the concept the instance belongs to, and it is connected to all nodes of the relational network.

4.3.1 Example

To illustrate the above ideas somewhat more in detail, let us look at how a real example will be described in a conceptual network. Let us assume that the instance from Fig. 4.2 is a new instance for the concept of a plane. The decomposition of the input object is given in Fig. 4.3. The structure of the resulting conceptual network is then shown in Fig. 4.4 and the generalized predicates showing the relation of N2 (fuselage) with respect to N1 (main wings) are shown in Fig. 4.5.

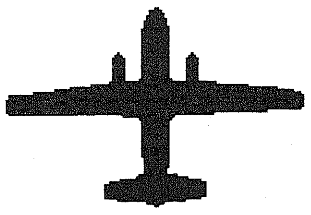


Figure 4.2: Input object

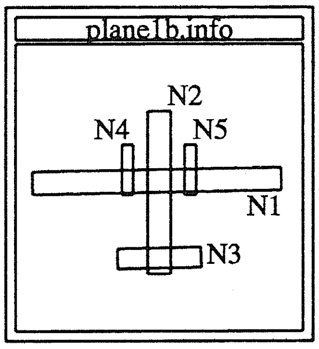


Figure 4.3: Decomposition

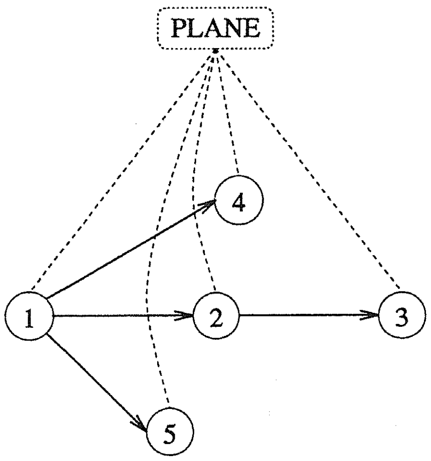


Figure 4.4: Resulting conceptual network

4.4 Review of some Graph Matching Methods

Finding subgraph isomorphisms or matching graphs is a computationally expensive task. According to Garey and Johnson [18], finding subgraphs of G isomorphic to H (subgraph isomorphism) is an NP-complete problem. However, the problem can be solved in polynomial time if G is a forest and H is a tree. The problem of graph isomorphisms, on the other hand, has still neither been proved to be NP-complete nor to have polynomial complexity. An ordinary graph in graph theory can be considered a weighted graph with binary weights. Therefore, the weighted graph matching problem includes the graph isomorphism problem.

In what follows, I will review some approaches to graph matching, starting with

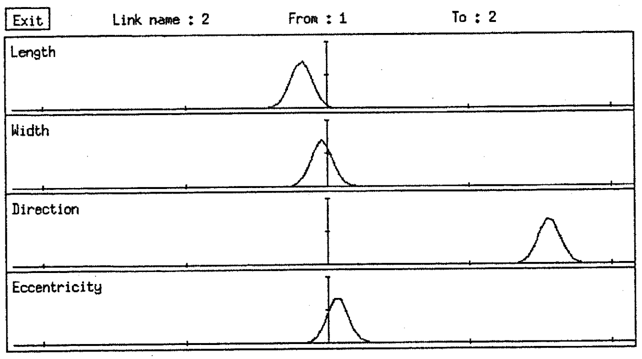


Figure 4.5: Generalized predicates of link 1 – 2 from Fig. 4.4

graph isomorphisms and then moving on to subgraph isomorphisms. Of those two it is only the latter that has direct implications to our case. However, the former could be used as a theoretical starting point for the latter, and extensions to the former might still make it a viable option at a later stage of this project.

4.4.1 Graph Isomorphisms

A weighted graph is a tuple of the form $G = (V, g)$, where $V = v_1, \ldots, v_n$ are the vertices of the graph and g is a weighting function that gives a non-negative value to each arc (v_i, v_j) of the graph. If the weighting function is symmetric, i.e., $g(v_i, v_j) = g(v_j, v_i)$ then the graph G is called undirected. A weighted graph can be represented by its adjacency matrix. The adjacency matrix A_G of the graph G is defined as

$$A_G = [a_{ij}] \quad \begin{cases} a_{ij} = w(v_i, v_j) & i \neq j \\ a_{ij} = 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

When G is undirected A_G becomes a symmetric matrix.

The problem of matching two weighted graphs $G = (V, x)$ and $H = (W, y)$ of the same number of vertices n consists of finding a one-to-one correspondence between V and W which makes G and H as similar as possible, similar in the following sense. Employing an $n \times n$ permutation matrix P, the weighted graph matching problem can be formulated as finding a P which minimizes

$$J(P) = \| A_G - PA_H P^T \|^2 \tag{4.9}$$

There are several approaches for finding P. Umeyama [60] proposed to use the eigendecompositions of the adjacency matrices in order to find a nearly optimal matching. In general, it is difficult to find the permutation matrix P directly. Therefore, Umeyama proposed to extend the domain of J to the set of orthogonal matrices. The set of orthogonal matrices (Q) that minimize $J(Q)$ can be obtained in closed form by using eigendecompositions of the adjacency matrices A_G and A_H. A nearly optimal permutation matrix can then be determined by using those orthogonal matrices as clue. In the case where G and H are really isomorphic, the problem reduces to a bipartite maximum weighted matching that can be solved in $O(n^3)$. Even if G and H are not isomorphic, it can be expected that the solution will be very close to the optimal permutation matrix. The solution can be improved by using hill-climbing or relaxation methods.

Another approach to the same problem is taken by Almohamad and Duffuaa [1], who propose to find the permutation matrix by means of linear programming. In order to apply a linear programming approach, the problem has first to be transformed into a linear one. This is done by noting that the minimization problem from Eq. 4.9 is equivalent to minimizing

$$R = A_G P - PA_H. \tag{4.10}$$

The matrices $R = \{r_{ij}\}$ and $P = \{p_{ij}\}$ can be partitioned by columns such that

$$\begin{aligned} \text{VEC}(R) &= (r_{11}, \ldots, r_{n1}, r_{12}, \ldots, r_{n2}, \ldots, \\ &\quad r_{1n}, \ldots, r_{nn})^T \qquad (4.11) \\ \text{VEC}(P) &= (p_{11}, \ldots, p_{n1}, p_{12}, \ldots, p_{n2}, \ldots, \\ &\quad p_{1n}, \ldots, p_{nn})^T, \qquad (4.12) \end{aligned}$$

then, Eq. 4.10 can be written in the form

$$\text{VEC}(R) = A_{GH} \text{VEC}(P), \tag{4.13}$$

where A_{GH} is an $n^2 \times n^2$ constant matrix obtained from the weights of graphs G and H. Therefore, the weighted graph matching problem can be written as

$$\min_P \| \mathrm{VEC}(R) \|^2 = \min_P \| A_{GH}\mathrm{VEC}(P) \|^2, \tag{4.14}$$

which is a linear problem. In a first step, the problem consists of finding a real basic solution $P' = \mathrm{VEC}(P)$ that minimizes

$$\| A_{GH}P' \|^2 \quad \text{under } P' \geq 0. \tag{4.15}$$

Since $A_{GH}P' = 0$ might not have a feasible solution, two sets of real positive goal variables $S = \{S_i\}$ and $T = \{T_i\}$ are introduced such that

$$A_{GH}P' + S - T = 0. \tag{4.16}$$

Then, the linear optimization problem becomes

$$\begin{aligned} &\min_{P',S,T} \textstyle\sum_{i=1}^{n^2} (S_i + T_i) \\ &\qquad \text{such that} \qquad A_{GH}P' + S - T = 0 \\ &\qquad\qquad\qquad\qquad P' \geq 0, S \geq 0, T \geq 0. \end{aligned} \tag{4.17}$$

Adding further constraints on the solution P' allows for an efficient treatment by using the simplex method. Almohamad and Duffuaa point out that their approach has complexity $O(n^6)$, which is, though still polynomial, worse than Umeyama's approach. Empirical tests, however, showed that the approach based on linear programming produces superior matching results and is more general than the approach based on eigendecompositions.

4.4.2 Subgraph Isomorphisms

The problem that is closer to us is the problem of finding subgraph isomorphisms. Subgraph isomorphisms can be determined by brute-force enumeration. Here, I will first outline such an enumeration technique, which actually is a depth-first tree search algorithm. This algorithm was given by Ullmann [59], together with further improvements that lead to finding subgraph isomorphisms based on relaxation techniques.

We consider two non weighted graphs $G = (V)$ and $H = (W)$. The problem is to find all isomorphisms between G and subgraphs of H. Again, the two adjacency matrices are given by $A_G = [a_{ij}]$ of size $n \times n$, and $A_H = [b_{kl}]$ of size $m \times m$, respectively. Introducing a permutation matrix P of size $n \times m$, we have

$$J(P) = \| A_G - P(PA_H)^T \|^2. \tag{4.18}$$

At the start of the enumeration algorithm we construct a $n \times m$ element matrix $P^0 = \{p^0_{ij}\}$ where we set $p_{ij} = 0$ if there is any a priori reason that the jth vertex of H could not correspond to the ith vertex of G. The algorithm works by generating all possible permutation matrices P such that for every element p_{ij} of P, $(p_{ij} = 1) \Rightarrow (p^0_{ij} = 1)$. Matrices P are generated by systematically changing to 0 all but one of the 1's in each row of P^0. In the search tree, the terminal nodes correspond to distinct matrices P.

To reduce the amount of computation needed, Ullmann proposed a procedure which he called *refinement procedure* [59], but which actually corresponds to relaxation labelling as pointed out by Kitchen and Rosenfeld [30]. First, let us consider a matrix P' that is associated with some non-terminal node in the search tree. Any subgraph isomorphism corresponds to a particular matrix P. An isomorphism is said to be an isomorphism under P' if its terminal node in the search tree is a successor of the node with which P' is associated. As seen above, the 0's in P' forbid correspondences between vertices of G and H. From the definition of subgraph isomorphism it is necessary that if v_i corresponds to w_j, then for each v_x adjacent to v_i, there must exist a w_y that is adjacent to w_j, such that w_y corresponds to v_x in the isomorphism. In other words, if v_i corresponds to w_j then

$$\forall x((a_{ix} = 1) \Rightarrow \exists y(p_{xy} b_{yi} = 1)). \tag{4.19}$$

The refinement procedure checks each 1 in P' to find whether the above condition is satisfied. For any $p_{ij} = 1$ such that the condition is not satisfied, $p_{ij} = 1$ is changed to $p_{ij} = 0$. The refinement procedure applies the above condition in turn to each 1 in P', until during one iteration no occurrences of 1's can be changed. Therefore, a necessary and sufficient condition for subgraph isomorphisms is that the refinement procedure leaves P'

unchanged. Empirical tests carried out by Ullmann using random graphs showed a time complexity of roughly $O(n^3)$. However, Ullmann points out that even a poor algorithm for isomorphisms may work well with random graphs.

The first attempt to characterize weighted subgraph isomorphisms as a state-space search was given by Tsai and Fu [57]. The idea turned out to be a fruitful one and subsequently several papers dealing with the state-space matching of graphs appeared [9, 16], all of which are rather similar in their approach. Tsai and Fu noted that if we are trying to match deformed graphs, then it is possible to use the a priori deformation probabilities for guiding the matching. According to Tsai and Fu, a state is described by a collection M of 2-tuples, each of which denotes a pair of matched nodes $v_i \in V$ and $w_j \in W$. Hence, a state in the search space gives that part of the matching that has been found until a certain moment. The initial state is defined as $M = \emptyset$. By expanding of a certain state M, a new state M' is generated. Expansion of the state $M = \{(v_1, w_1), \ldots, (v_n, w_n)\}$ is defined by adding a new pair $(v_i, w_j) \in V \times W$, such that $v_i \in V \setminus \{v_1, \ldots, v_n\}$ and $w_j \in \setminus\{w_1, \ldots w_n\}$, and (v_i, w_j) must be valid in respect to state M. Valid refers to the fact that embedding (v_i, w_j) into M does not contradict any previously matched pairs. By expanding M, the partial matching has been augmented by one element. A state M that exhausts all vertices $w \in W$ is a goal state and thus a valid subgraph isomorphism. In the case of weighted or attributed graphs, it is possible to define a cost for the expansion of a state, which will be the cost associated with the matching of (v_i, w_j). Tsai and Fu propose to use ordered search or an A* algorithm [6]. The successor of a state is obtained by finding all possible expansions of the state. Then the search algorithm uses an evaluation function to order states for expansion. For a state M_i, an evaluation function of the form

$$g(M_i) = g_1(M_i) + g_2(M_i), \tag{4.20}$$

can be used. Here, $g_1(M_i)$ represents the minimum total path cost from the start state M_0 to M_i, and $g_2(M_i)$ is a lower bound estimate for the cost of the optimal path from M_i to the goal state M_n. As has been proved, if g_2 is a consistent lower bound estimate for the remaining path cost, this approach is guaranteed to give the optimal solution, with fewer states expanded than without using look-ahead information.

4.5 Matching of Relational Networks

The matching approach used in the present system is largely based on the approach by Eshera and Fu [16]. Eshera and Fu's method is a state-space method and is very closely related to the method discussed in the previous section. As we have seen before, the efficiency of the state-space approach comes from the fact that we can use heuristics for the ordering of the search space. An essential point is therefore the derivation of an appropriate cost function for matching parts of the graph. As only links of the relational network do carry information, the cost function has to be derived for matching links or relations. In what follows, I will first describe how to derive this cost function in the case of the present system. Subsequently, I will outline the used matching scheme in some detail.

4.5.1 Cost Functions and Link Similarities

As has been seen in Section 4.2, the information contained in a relational network is concentrated in its links. Therefore, the matching of relational networks proceeds via the matching of its links. Qualitatively, the cost for matching two links should be smaller the more similar the two links are. Therefore, the cost should be 0 for matching a link with itself, and it should be maximal for matching two absolutely dissimilar links. Hence, the cost for matching two links is inversely proportional to the similarity of the two links. In a first step, I will therefore proceed to the explanation of link similarity.

As shown in Section 4.2, a link of the relational network represents the relation between two segments of the initial description. As such, each link carries a confidence value and a set of four generalized predicates in the form

$$f_i(z) = \sum_j \frac{a_{ij}}{\sqrt{2\pi}\sigma_{ij}} \exp^{-\frac{1}{2}\frac{(z-t_{ij})^2}{\sigma_{ij}^2}} \qquad i \in 1, \dots 4. \tag{4.21}$$

Clearly, two links are most similar if and only if they are the same. This means, all their generalized predicates have to be identical. Therefore, link similarity has to be defined over the set of the generalized predicates. We start with the definition of similarity between generalized predicates. Again, generalized predicates are most similar if and only if they are identical. A simple and intuitive solution for the problem of similarities of generalized predicates can be obtained if we remember that a generalized predicate can be considered a probability distribution (up to a certain scaling factor) of a random variable. Now, we can assume that the two generalized predicates to be compared are statistically independent. Actually, this is quite a reasonable assumption, as the two generalized predicates, in general, will come from two separate instances. Obviously, the joint distribution of two random variables can be calculated as the pointwise multiplication of their respective distributions. Hence, we can use the joint distribution as a measure for the similarity of the two generalized predicates (again up to some scaling factor). Furthermore, as we would like to obtain a scalar value for the similarity, we could use the value of the probability of the expectation of the joint distribution. If the joint distribution has a mirror symmetry around the expectation, which is given in the case of a Gaussian distribution and the multiplication of two Gaussians, we can use the maximum value of the joint distribution instead. Note that in the case of summed Gaussians, using the maximum value might not be correct anymore, however, we retain it as a simplification. Therefore

Definition 7 *The similarity s between two generalized predicates f_i and f_j is defined as*

$$s(f_i, f_j) = \max_{domain(f_i \cdot f_j)} f_i \cdot f_j \tag{4.22}$$

where $\cdot$ denotes the pointwise multiplication.

Given the similarity between generalized predicates, we can define the similarity between sets of generalized predicates, that is, links. Again we use the statistical interpretation of generalized predicates and of their similarities. We saw that the similarity s of two generalized predicates is given as the maximum of their joint distributions. Hence s is a probability as well. Again assuming statistical independence between the generalized predicates of a link, we can again use the joint probability as a measure for the link similarity

r. Namely,

Definition 8 *The link similarity r of two links b_u and b_v is defined as*

$$r(b_u, b_v) = \Pi_i s(f_{ui}, f_{vi}) \quad i = 1, \ldots, 4. \tag{4.23}$$

Actually, as can be seen from the definition of the generalized predicates in Section 4.2, the assumption of statistical independence between generalized predicates of a node is not really fulfilled; for instance, the eccentricity and the angle predicate are somewhat related. However the assumption of statistical independence was considered a justifiable simplification.

Finally, given the link similarity r we have to calculate the cost for matching two links. Per definition, the initial values for the generalized predicates, and therefore for the link similarities, are in the range $0 \ldots 1$. As should become clear from the generalization of generalized predicates in Section 5.2, this range is preserved through generalization. Therefore, link similarity as well will be in the range $0 \ldots 1$, and we have

Definition 9 *The cost for the matching of two links b_u and b_v is defined as*

$$cost(b_u, b_v) = 1 - r(b_u, b_v). \tag{4.24}$$

In other words, the cost for the matching of two links is the complement of their similarity.

4.5.2 Matching

The actual matching of relational networks closely follows the approach by Eshera and Fu [16]. Since the two relational networks to be matched are not of same size, the approaches for graph isomorphisms, as described in Section 4.4 are not applicable. In addition, since we cannot be sure that the smaller of the two relational networks can be matched completely, the task is even more formidable than subgraph isomorphism, maybe we could speak of subgraph sub-isomorphisms. It appears that for such tasks the method by Eshera and Fu does rather well.

Eshera and Fu explicitly address the problem of matching attributed relational graphs or ARG's for short, that is, a class of graphs that are especially adequate for image representation tasks. Formally, an ARG is defined as

$$G = \langle N, B \rangle \tag{4.25}$$

where $N = \{n_1, \ldots, n_{|N|}\}$ is a finite set of nodes, and $B = \{b_1, \ldots, b_{|B|}\}$ is a set of node pairs or branches, i.e., $b = (n_i, n_j)$ for some $1 \leq i, j \leq | N |$ denotes the branch connecting node n_i with node n_j. Both, nodes and branches on an ARG carry attributes. Note that no restrictions at all are placed on the nature of the attributes, they could be symbolic, numerical, or, as in our case, a set of functions. Based on the definition of an ARG, it is possible to define a *Basic Attributed Relational Graph* or BARG as follows

Definition 10 *A BARG is defined as a graph in the form of a one-level tree, i.e., it consists of a root node, the branches emanating from the root, and the nodes on which these branches terminate.*

A BARG G_i is given as a tuple of the form

$$G_i = \langle r_i, B_i, L_i \rangle \tag{4.26}$$

where r_i is the root node of the BARG G_i, B_i is the set of branches emanating from r_i, and L_i is the set of leaf nodes where the branches terminate. To make this definition more transparent we can use the example decomposition of a plane in Fig. 4.6, together with its relational network in Fig. 4.7. To transform a relational network into an ARG it is sufficient to extract connected parts of the relational network, which might actually consist of multiple, disconnected subgraphs. In the example shown in Fig. 4.7, the ARG corresponds to the relational network. Transferring the ARG into a set of BARG's gives the structures as shown in Fig. 4.8. Actually, Eshera and Fu propose an approach for calculating the distance between two graphs. However, matching and distance calculation are the same in that the minimum distance between two graphs defines the optimal matching between them.

Matching an ARG U with another ARG V means we have to find a sequence of transformations for converting V into U, with minimal cost. Hence it is essential to

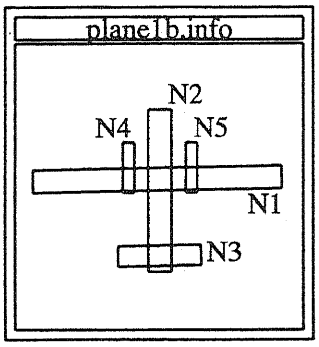


Figure 4.6: Decomposition of a plane

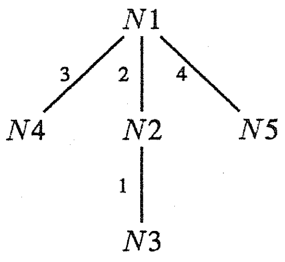


Figure 4.7: Relational network for Fig. 4.6

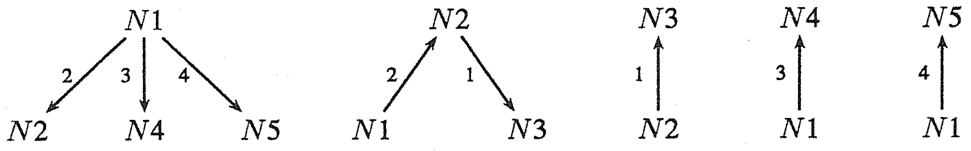


Figure 4.8: Set of BARG's obtained from Fig. 4.7

optimize over all valid sequences of transformations. Without loss of generality we assume that $| U |>| V |$. We start by defining the empty graph $\Lambda = (\Phi, \Phi)$ as the graph of no nodes and no branches. Further we decompose two ARG's U and V into their sets of BARG's $\hat{U} = \{U_1, \ldots, U_m\}$ and $\hat{V} = \{V_1, \ldots, V_n\}$, respectively, where m and n are the numbers of nodes in U and V. The matching itself employs a state-space representation as described in Section 4.4. The state-space representation used is in the form of a directed acyclic labelled lattice, which consists of a set of states $S = \{s_0, s_1, \ldots, s_L\}$, where $L + 1$ is the total number of states, and a set of directed branches, $R(s_I, s_J)$ which connect two states s_I and s_J. For two ARG's U and V, each state in the lattice corresponds to a partial reconstruction of the two graphs U and V, as well as to the matching of their BARG's. The initial state s_0 corresponds to the empty graph Λ, starting from which possible states, corresponding to partial matchings, are expanded. The transition from one state to the other represents the embedding of a new pair of matched BARG's into the reconstructed subgraphs; the weights on the branches stand for the incremental cost due to the matching of the new pair of BARG's. At each state s_I, we have to determine the best successor s_{I+1}

of s_I. First, only those states are expanded that yield possible reconstructions of the ARG's U and V. Possible reconstructions means that the new state should not contradict any previous matchings, in respect to the structures of both, U and V. Furthermore, we discard all those expansions whose cost is above a certain threshold, which is set so as to exclude all those matches whose similarity implies that the corresponding Gaussian distributions are further apart than 4σ, assuming an amplitude of 0.65 for the Gaussians. This step allows to exclude some completely unfeasible matches and therefore to speed up the matching. Finally, among the remaining possible expansions, we choose the one that is heuristically most promising. The used criterion transforms the search problem to a heuristic search [6] where that state s_{I+1} match that minimizes the cost function

$$g(s_{I+1}) = g_1(s_{I+1}) + g_2(s_{I+1}), \tag{4.27}$$

is chosen as the successor of state s_I. In the above equation, $g_1(s_{I+1})$ are the summed costs of all the matchings on this path, namely

$$g_1(s_{I+1}) = \sum_{i=1}^{i=I+1} \text{cost}(s_{i-1} \rightarrow s_i), \tag{4.28}$$

where $\text{cost}(s_{i-1} \rightarrow s_i)$ denotes the cost associated with the state transition from state s_{i-1} to state s_i. Further, $g_2(s_{I+1})$ is an estimate of the cost for completing the matching, starting from state s_{I+1}. Assuming $|U|>|V|$, the look-ahead cost can be calculated as

$$g_2(s_{I+1}) = \alpha(|V| - I), \tag{4.29}$$

where α is an empirically determined weight, presently set to 1.1, and I denotes the number of matches obtained so far. The expansion of the path stops if no more states can be expanded, either because V has been reconstructed completely, or because no further reconstructions are possible. In the former case the algorithm stops, returning the path from Λ to the final state. In the latter case, the path is stored as one possible solution, and we go on to continue expansion for a fixed number of times. Doing so will possibly give a number of feasible matches, ordered according to their cost function. If this is the case, the best matching is retained, however, the remaining matchings are used to detect unstable matches.

An unstable match means that one node $v_i \in V$ could be matched to two different nodes $u_j, u_k \in U$, without altering the resulting cost too much. Since such unstable matchings can be interpreted as not having sufficient information for being matched, we opt to remove them from the resulting matching.

In order to give a clearer illustration of the used matching scheme, we can study its performance by using a simple example. Let us try to match the plane from Fig. 4.6, which we will call PLANE1, with the one from Fig. 4.9, which will be denoted by PLANE2. The ARG and the set of resulting BARG's for PLANE2 are given in Figs. 4.11 and 4.12,

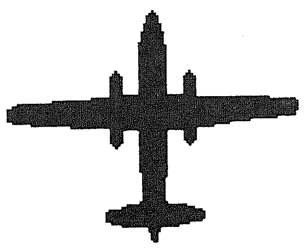


Figure 4.9: Input object

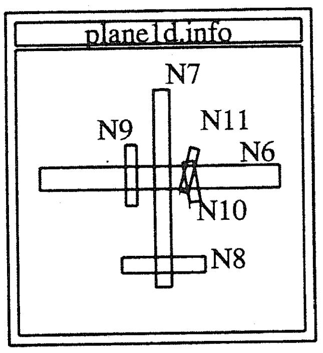


Figure 4.10: Decomposition

respectively. Expanding $s_0 = \Lambda$ proceeds as follows. As there is cost 0 associated with s_0 the cost function does not use any look ahead information and is reduced to best-first at this stage. This means among the possible matches $b^U \times b^V$, where b^U and b^V denote the branches in ARG's U and V, respectively, we select the one with minimal cost. This

$$\begin{tikzcd}
 & N6 \arrow[dl, "8"'] \arrow[d, "7"] \arrow[dr, "9"] \arrow[drr, "10"] & & \\
N9 & N7 \arrow[d, "6"'] & N10 \arrow[r, "5"'] & N11 \\
 & N8 & &
\end{tikzcd}$$

Figure 4.11: Relational network for Fig. 4.9

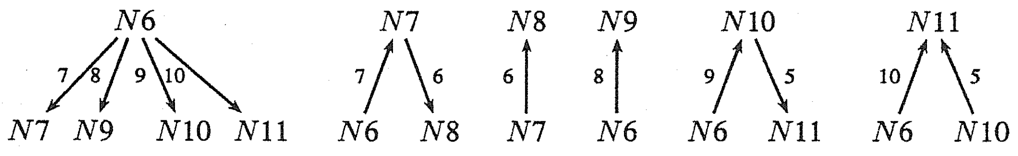


Figure 4.12: Set of BARG's obtained from Fig. 4.11

is illustrated in Fig. 4.13, where the paths are ordered so that the one with least cost is on the left. For purposes of illustration, beneath each expanded node $s_1 \ldots$, the matched nodes of the partially reconstructed ARG are listed. Expanding the path with minimum

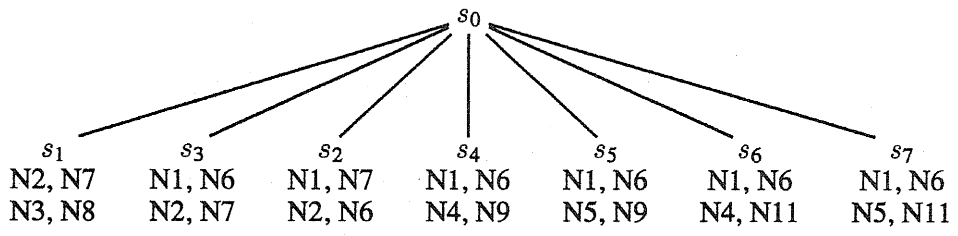


Figure 4.13: Expanding the first level of the state-space

cost, that is, the leftmost path, terminating at state s_1, and again ordering the result yields the state-space as shown in Fig. 4.14. Again, we can expand, obtaining the results as shown in Fig. 4.15. The final expansion, as shown in Fig. 4.16, cannot be expanded any further, it is a final state. State s_{13} is a terminal state, that is, it cannot be expanded any further. Hence, the algorithm stops and returns the solution, namely s_{13}, which is representing a possible path from s_0 to a terminal state. As can be seen from Fig. 4.16, $s_{13} = \{(N2, N7), (N3, N8), (N1, N6), (N4, N9), (N5, N10)\}$. Hence, the resulting matching will be of the form $M = \{(N2, N7), (N3, N8), (N1, N6), (N4, N9), (N5, N10)\}$, corresponding to matching the fuselages, the main-wings, the tail-wings, and the two engines, as can be seen from Figs 4.6 and 4.10.

Given the segment matching M, sometimes it might be possible to update the matched links. To see why this could be possible, let us have a look at the two ARG's in

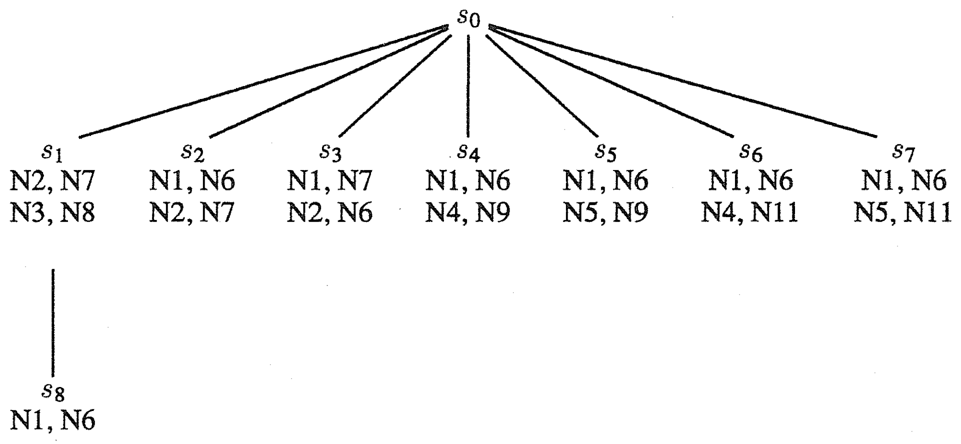


Figure 4.14: Expanding the second level of the state-space

Fig. 4.17. Furthermore, let us assume that the matching algorithm returned the following maximal matching

$$M = \{(N1, N5), (N2, N6), (N3, N7), (N4, N8)\},$$

which was obtained by matching the following links

$$M_l = \{(1,6), (2,7), (4,9)(5,10)\}.$$

Based on M, we can augment the set of matched links M_l by all those matches that are possible according to M, namely

$$M_l = \{(1,6), (2,7), (3,8)(4,9), (5,10)\}$$

This means, one additional link match can be found.

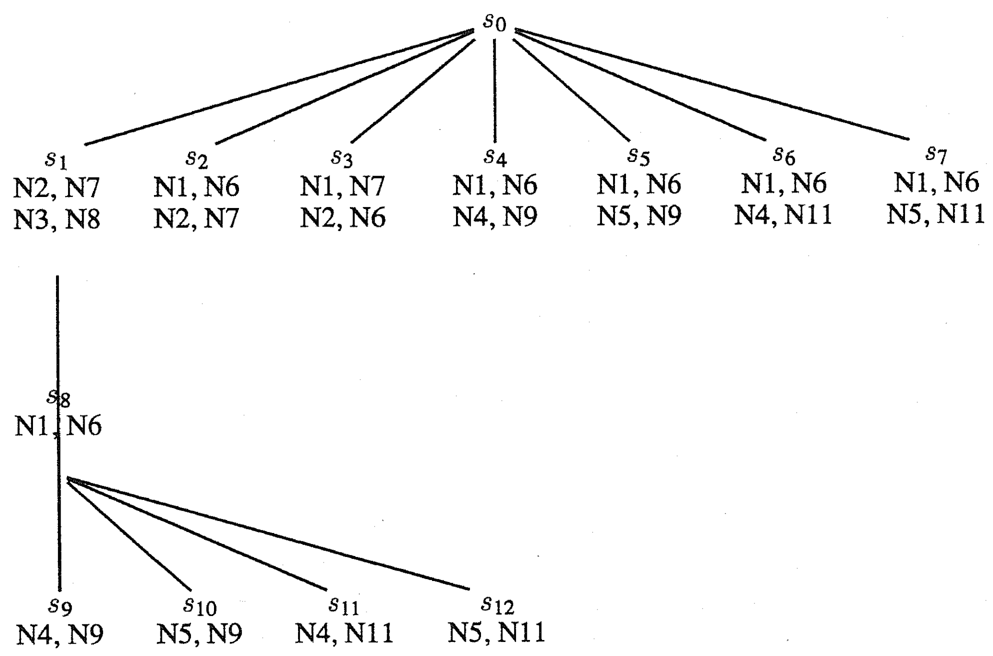


Figure 4.15: Expanding the third level of the state-space

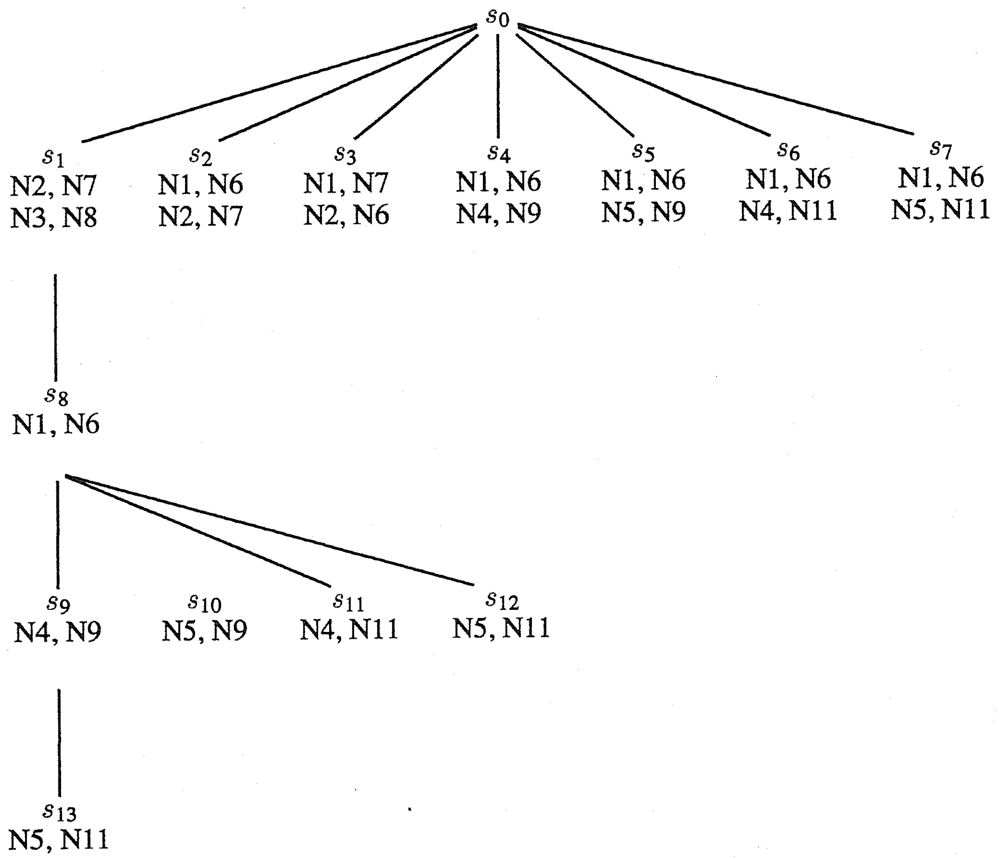


Figure 4.16: Expanding the fourth level of the state-space

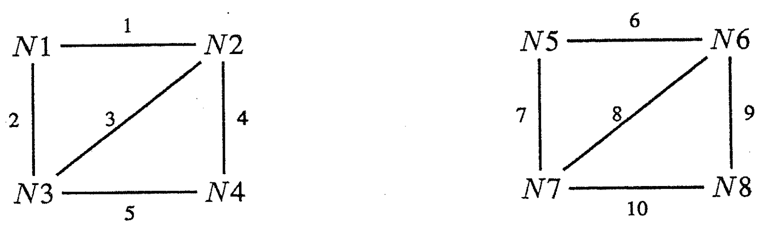


Figure 4.17: Two ARG's

CHAPTER 5

Generalization and Specialization

5.1 Purpose and Overview

Once a new instance has been matched with an existing concept, the instance and the concept have to be generalized, so as to yield a new concept, which can represent the new instance as well. The simplest case of generalization is given if there is no concept present yet. In such a case, no matching can be obtained either. As the instance is the only one of its kind that was seen by the system, it makes sense to directly promote the instance to the level of concept. This is particularly easy, as there is no structural difference between a concept and an instance. As soon as we attempt to generalize an instance with a nonempty concept, however, things look different. Such a generalization takes part on three levels, roughly corresponding to level of detail or abstraction. The lowest level is the level of single relations or links of the relational network. In the next level, the structure of the relational network has to be adjusted. Finally, the higher levels of the conceptual network have to be generalized as well. It is here that concept abstraction and grouping takes part. In the next section, I will treat the generalization of links of the relational network, leaving structural questions to Section 5.3. Finally, in Section 5.4, I will treat the opposite of generalization, namely specialization.

5.2 Generalization of Network Links

All the nodes and links of the relational network that appeared in the matching M, as explained in Section 4.5, have to be generalized. This means, each matched pair of nodes and links will be replaced by one new node or link, respectively. As the nodes of the relational network do not carry any information, their generalization is straightforward; the more difficult generalization will take part among the links of the relational network. In what follows, I will denote any generalization by the symbol $\vdash$; for instance, the relation that link 4 and link 9 are generalized into link 15 will be denoted as $(4, 9) \vdash 15$. As seen in Section 4.2, a link of the relational network has a confidence value $cval$ and a set of four generalized predicates, which numerically define the relation the link stands for. While generalizing, we have to treat the different constituents of a link differently. To make things easier to understand, I will denote the link from the concept by the subscript C, the corresponding link from the instance will be denoted by the subscript I, and the link containing the generalized information will be denoted by the subscript new. This leads to the following approach for generalizing links:

Step 1: Add the confidence values:

$$cval_{new} = cval_C + cval_I. \tag{5.1}$$

Step 2: For each pair of corresponding generalized predicates f_{iC} and f_{iI}:

$$\begin{aligned} f_{i\ new} &= f_{i\,C} + \varepsilon\, f_{i\,I}(1 - f_{i\,C}) \quad \text{if } cval_C > cval_I \\ f_{i\ new} &= f_{i\,I} + \varepsilon\, f_{i\,C}(1 - f_{i\,I}) \quad otherwise. \end{aligned} \tag{5.2}$$

where the weight ε is defined as

$$\varepsilon = e^{-\frac{1}{2}\frac{(cval_I - cval_C)^2}{\sigma_\varepsilon^2}}, \tag{5.3}$$

following the arguments to be given shortly.

The second step of the above procedure introduces the product of two Gaussians. Strictly speaking, the definition of generalized predicates as the sum of Gaussian distributions (see Section 4.2) is therefore not valid anymore. However, I will show in Appendix C, that the product of two Gaussian distributions can itself be sufficiently approximated by another Gaussian distribution. Therefore, the definition for generalized predicates retains its validity.

Looking at the case where $cval_C = cval_I$ and replacing f_{iI} by $\max(f_{iI})$ and f_{iC} by $\max(f_{iC})$, the above generalization rule (step 2) corresponds to the generalization rule for certainty factors as used in MYCIN. It is possible to find a probabilistic interpretation for this generalization rule, in doing so we rely on work by Heckerman [22]. Let us first consider Bayes' theorem for updating the probability of a hypothesis H, given some evidence E:

$$p(H \mid E) = \frac{p(E \mid H)p(H)}{p(E)}. \tag{5.4}$$

The equation for the negation of the hypothesis, $\neg H$, is:

$$p(\neg H \mid E) = \frac{p(E \mid \neg H)p(\neg H)}{p(E)}. \tag{5.5}$$

If we divide those two equations, we get:

$$O(H \mid E) = \frac{p(E \mid H)}{p(E \mid \neg H)} O(H), \tag{5.6}$$

where $O(X)$ are the odds of some event X. The ratio $p(E \mid H)/p(E \mid \neg H)$ in the above equation is called a *likelihood ratio* and we can write it as $\lambda(H, E)$.

Now, if there are two items of evidence, E_1 and E_2, which are assumed to be statistically independent under the hypotheses H and $\neg H$, Bayes' theorem can be written as:

$$\frac{p(H \mid E_1 E_2)}{p(\neg H \mid E_1 E_2)} = \frac{p(E_1 E_2 \mid H)}{p(E_1 E_2 \mid \neg H)} \frac{p(H)}{p(\neg H)} = \frac{p(E_1 \mid H)}{p(E_1 \mid \neg H)} \frac{p(E_2 \mid H)}{p(E_2 \mid \neg H)} \frac{p(H)}{p(\neg H)}. \tag{5.7}$$

From the definition of λ, it is evident that

$$\lambda(H, E_1 E_2) = \lambda(H, E_1)\lambda(H, E_2). \tag{5.8}$$

The range of λ is $0 \ldots \infty$, however, the range of a generalized predicate is $0 \ldots 1$. Therefore, we have to apply a mapping for adjusting the range of λ. For instance, we can consider the following mapping function:

$$F(x) = \frac{x-1}{x} \quad \text{however, } x \geq 1. \tag{5.9}$$

Therefore, we have

$$f_i = F(\lambda(H, E_i)) = \frac{\lambda(H, E_i) - 1}{\lambda(H, E_i)} \quad \lambda \geq 1. \tag{5.10}$$

The relation $\lambda \geq 1$ is always true, since we are only considering evidence that is in favour of the new hypothesis H. Using Eq. 5.8, we obtain

$$f_{i\,new} = \frac{\lambda(H, E_C)\lambda(H, E_I) - 1}{\lambda(H, E_C)\lambda(H, E_I)}. \tag{5.11}$$

After a series of transformations, we finally get

$$f_{i\,new} = \frac{\lambda(H, E_C) - 1}{\lambda(H, E_C)} + \frac{\lambda(H, E_I) - 1}{\lambda(H, E_I)} - \frac{\lambda(H, E_C) - 1}{\lambda(H, E_C)} \cdot \frac{\lambda(H, E_I) - 1}{\lambda(H, E_I)}. \tag{5.12}$$

This is clearly equivalent to our generalization rule, using the transformation function $F(x)$. Therefore, we can say that our generalized predicates, together with their generalizations do have a probabilistic interpretation. Namely, a generalized predicate is a pointwise mapping of the likelihood ratio of a hypothesis H, and the generalization of generalized predicates corresponds to the pointwise multiplication of their likelihood ratios, or likelihood functions in our case.

Now, for understanding the role of $cval_C$ and $cval_I$, we can look at human learning behaviour. In human learning, the ordering of instances representing the concept to be learned is usually important. In other words, instances presented early leave a more persistent impression, whereas instances presented later will not affect our ideas about the concept too much. This is particularly visible in the way we form preconceptions or prejudices, based on some isolated incidents. This means that at a later stage of learning more instances are needed to give a similar impression as a single instance in the beginning of the learning sequence. In other words, the importance of an instance is inversely proportional to its location in the learning sequence. However, this is rather a poor model of human learning behaviour, as it does not take into account factors like forgetting or changing of

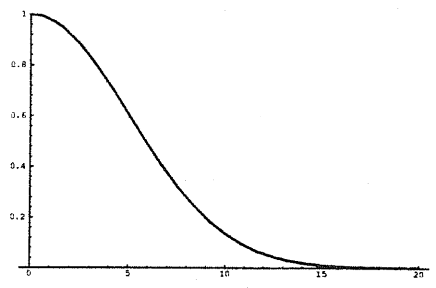
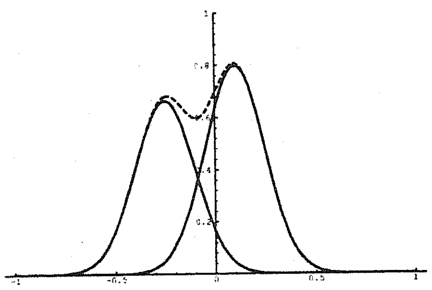


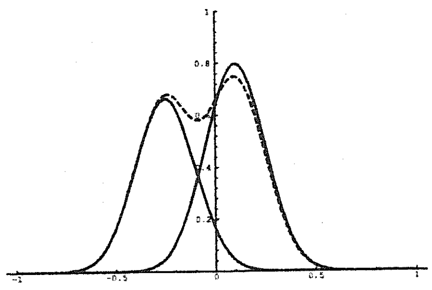
Figure 5.1: The weight ε as a function of $|\, cval_I - cval_C \,|$ ($\sigma_\varepsilon = 5$)

opinions. To come back to our application, the confidence value *cval* can be considered as giving us the location of the instance in the learning sequence. We have seen above that during generalization, confidence values are being added. Hence the confidence value of a particular link of the relational network tells us how many times it has been generalized, or in other words, with how many instances it has been generalized. To achieve the aforementioned effect, the weight ε, which grades the instance according to its importance in respect to the concept, is introduced. The graph of ε as a function of $|\, cval_I - cval_C \,|$ is shown in Fig. 5.1. The symmetry in Eq. 5.2 is only necessary if we attempt to generalize two concepts. Only in such a case we have $cval_I \neq 1$. In Fig. 5.2, the generalization of two generalized predicates, with changing confidence values is shown. As becomes clear from that figure, as learning proceeds, the influence of new instances gradually decreases.

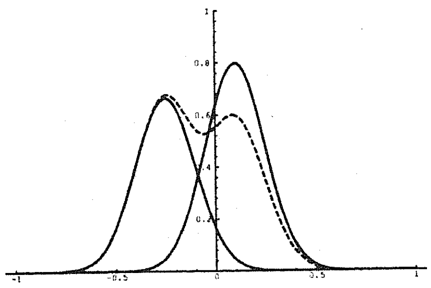
As can be seen from Eq. 5.2, during generalization a Gaussian will be replaced by the sum of three Gaussians. This means, after a few generalization steps, each generalized predicate can consist of a rather large number of Gaussians, which will slow down matching and generalization considerably. To avoid such a break down in performance, each generalized predicate that consists of more than 20 terms will be approximated by a new sum of Gaussians containing at most 12 terms. The actual approximation is explained in detail in Appendix D.



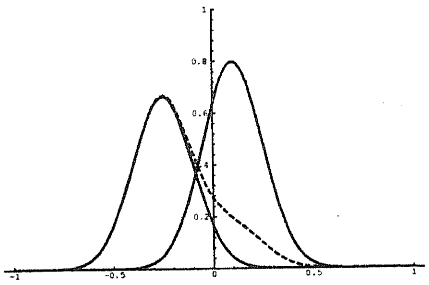
a) $cval_C = 1, cval_I = 1$



b) $cval_C = 3, cval_I = 1$



c) $cval_C = 6, cval_I = 1$



d) $cval_C = 10, cval_I = 1$

Figure 5.2: Generalization of generalized predicates (generalization result shown as dashed line), $\sigma_\varepsilon = 5$

In Section 4.2 we were speaking about interpreting the generalized predicates as probability distributions. As we have seen in that section, generalized predicates have function values in the range $0 \ldots 1$. In order to keep the probabilistic interpretation of the matching of generalized predicates, it is essential that this range be preserved during generalization. From the definition of ε, we have $\varepsilon \leq 1$. Now, per definition, we have that $f_{iC}, f_{iI} \leq 1$, thus $\varepsilon f_{iI} < 1$. Therefore, $f_{iC} + \varepsilon f_{iI}(1 - f_{iC}) \leq 1$.

5.3 Structural Generalization

Let us assume that we obtained a matching M between an instance and a concept. The structural generalization of the concept and the instance, denoted by $U, V \vdash W$, can then be obtained as follows. First, we replace all nodes and links of the concept U that are covered by the matching M by new nodes and links in W. Then, for each node $v_i \in V$ such that $u_k, v_i \vdash w_k$, we extract all nodes $\{v_j\}$ that connect to v_i but are not covered by the matching M. The nodes $\{v_j\}$ are then connected to the node $w_k \in W$. This is repeated for all nodes $v_i \in M$. If there are any nodes and links in U that are not covered by this treatment, then they must be disconnected from the generalized part and they can be included in W as is. The example in Fig. 5.3 illustrates the approach. Given the matching set $M = \{(N2, N10), (N4, N11), (N5, N12)\}$, node $N13$ from the instance V is not covered. Therefore, by extracting the nodes that are connected to node $N12$ and that are not covered by M, that is, node $N13$, and attaching this node to $N5$ in W, since $N5, N12 \vdash N5$, the structure can be fully recovered.

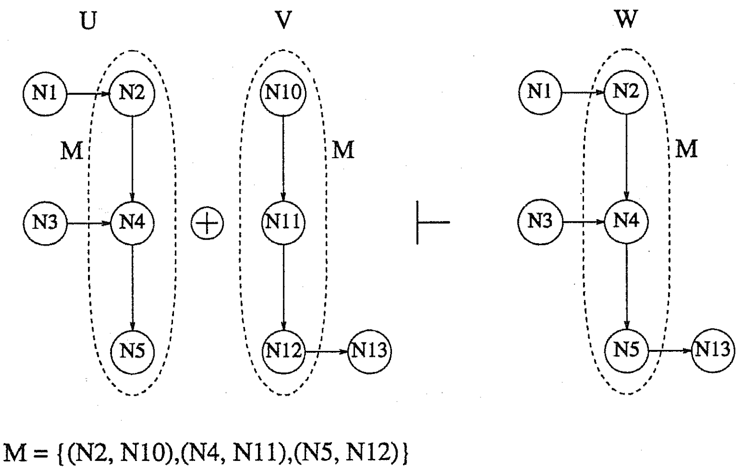


Figure 5.3: Structural generalization

If the conceptual network and the new instance represent the same concept, then the whole of W will be connected to this concept. This will add a third dimension to the conceptual network, this dimension will cover hierarchical abstractions. An example of the

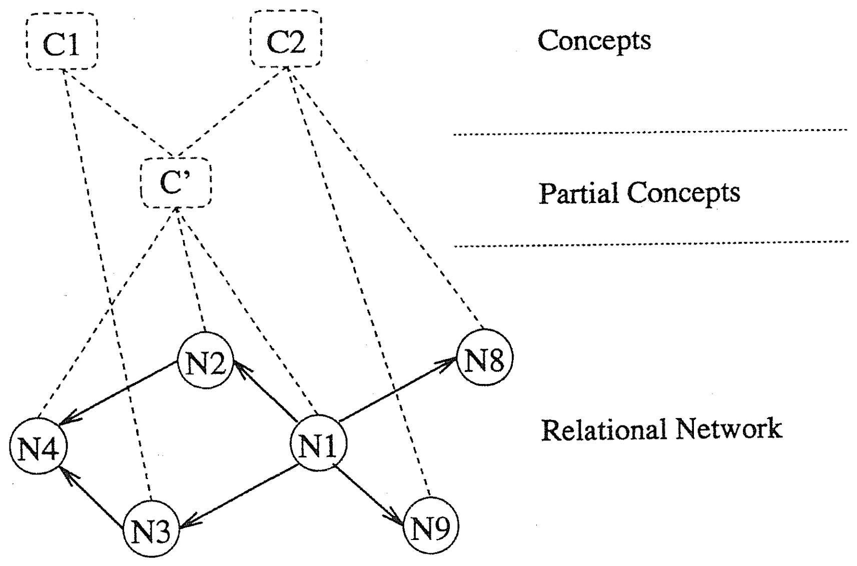


Figure 5.4: Resulting conceptual network

simplest form of conceptual network is given in Fig. 4.4. The case is different, however, if the conceptual network and the new instance stand for different concepts. As outlined in Section 2.1, the two concepts might still share some parts of the relational network. Therefore, it becomes possible to derive intermediate abstractions, which correspond to partial concepts, that are shared by two or more concepts. For instance, assume that there are two different concepts, say C_1 and C_2, which were partially matched, yielding a matching set M. Based on that matching set, the part of the relational network shared by the two concepts can be generalized as shown above, let us call this part W'. Now, instead of connecting all the generalized nodes to both concepts, we can create an intermediate concept C', whose descendants are the nodes in W' and which connects to C_1 and C_2 up the hierarchy. An example of such an intermediate concept is given in Fig. 5.4. Continuing the generalization with another instance might reveal that C' can be divided into two partial concepts C_1' and C_2', and so on. Carrying the idea further, something like conceptual clustering [35] becomes possible. This means, concepts can be clustered according to the partial concepts they share. An example of such conceptual clustering will be given in Chapter 7.

5.4 Specialization

Assume that we learned the description of a concept, say TOOL1, and now we want to learn another concept, say TOOL2. The scenario described here most naturally arises from supervised learning, that is, we are teaching the system what concept a new instance belongs to. It is imaginable that the matching of TOOL2 with the conceptual network representing TOOL1 reveals a complete match between the two. This means, that the discriminative power of the conceptual network does not suffice, it has been over-generalized. It is important in such a case that some generalizations can be reverted, so as to obtain two disjoint concepts. Two concepts are disjoint if they differ in at least one node of the relational network. Since we do not have sufficient information to actually revert some steps in the generalization process, a different approach has to be taken. We opt to duplicate one node of the relational network by actively splitting one link that connects to that node. The way of doing this is as follows. First, concept U and instance V have been generalized to the new concept W, such that $|U|=|V|=|W|=|M|=N$. That means, all nodes of the concept and the instance have been matched. Now, each generalized predicate f_{ij} of each link $w_j \in W$ is searched for an optimal threshold t_{ij}, at which to split f_{ij}. The threshold is obtained by assuming that a generalized predicate has a bimodal characteristic. For this task, Otsu's method [40] for the thresholding of bimodal histograms recommends itself. Otsu's method will be explained briefly in Appendix E. For splitting a link, it is sufficient to split one generalized predicate. For each generalized predicate $f_{ij} \in w_j, w_j \in W$, we define a cut point t_{ij}, according to Otsu's method. Based on the cut point t_{ij}, the generalized predicate f_{ij} can be split into two generalized predicates f'_{ij} and f''_{ij}. The choice of which generalized predicate to split corresponds to finding the best dichotomy for any f_{ij}. This can, for instance, be done by the following optimizations

$$J(f_{ij}) = \max_{i,j} \max_{domain(f_{ij})} |g_{ij} \cdot f'_{ij} - g_{ij} \cdot f''_{ij}|, \quad g_{ij} \in u_j, u_j \in U, \tag{5.13}$$

The f_{ij} that maximizes J will then be the candidate for splitting. To achieve a proper specialization, the node that is pointed at by w_j is duplicated, and so are all the links connecting to that node, except for w_j. The splitting of w_j into w'_j and w''_j is done by first duplicating all its generalized predicates, except for f_{ij}, i.e., the one that has to be split. Now, f_{ij} is discarded and replaced by f'_{ij} in w'_j and f''_{ij} in w''_j.

To illustrate the ideas developed here, we can look at an example to see how splitting is carried out in practice. To this aim, consider the two input shapes as given in Fig. 5.5.

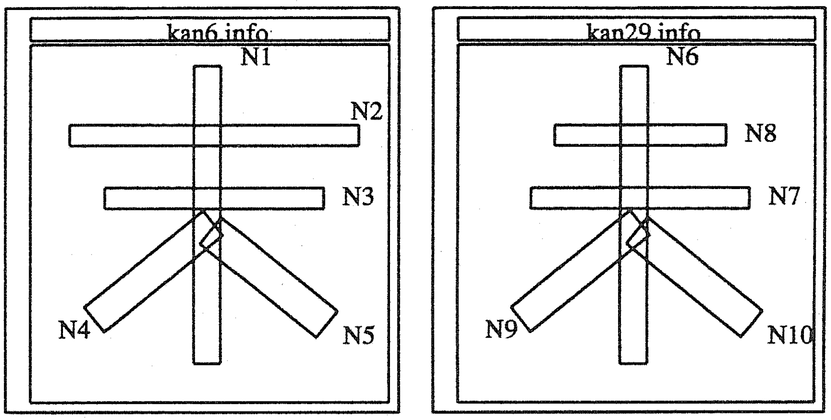


Figure 5.5: Two input shapes

The generalization of the two shapes proceeds according to Fig. 5.6, and reveals a complete match between them. Since the two shapes from Fig. 5.5 are acquired as two disjoint concepts, the complete matching makes it necessary to find some specialization that allows for a successful discrimination of the two concepts. The differences necessary for specialization are found according to the splitting scheme outlined above and in Appendix E. An evaluation of the links reveals that the length predicate of link 4 is best suited for splitting. As a result, node N2 is split into nodes N2 and N11, and link 4 is split into links 4 and 15. The effect of this split on the resulting links is shown in Fig. 5.7. According to the splitting of link 4 and node N2, the resulting conceptual network can be updated. This is shown in Fig. 5.8.

Node 1 and 6 ⊢ Node 1
Node 2 and 8 ⊢ Node 2
Node 3 and 7 ⊢ Node 3
Node 4 and 9 ⊢ Node 4
Node 5 and 10 ⊢ Node 5
Link 1 and 8 ⊢ Link 1
Link 2 and 9 ⊢ Link 2
Link 3 and 10 ⊢ Link 3
Link 4 and 12 ⊢ Link 4
Link 5 and 11 ⊢ Link 5
Link 6 and 13 ⊢ Link 6
Link 7 and 14 ⊢ Link 7

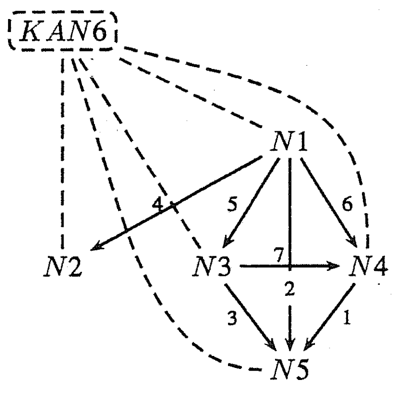


Figure 5.6: Generalization of the two shapes from Fig. 5.5

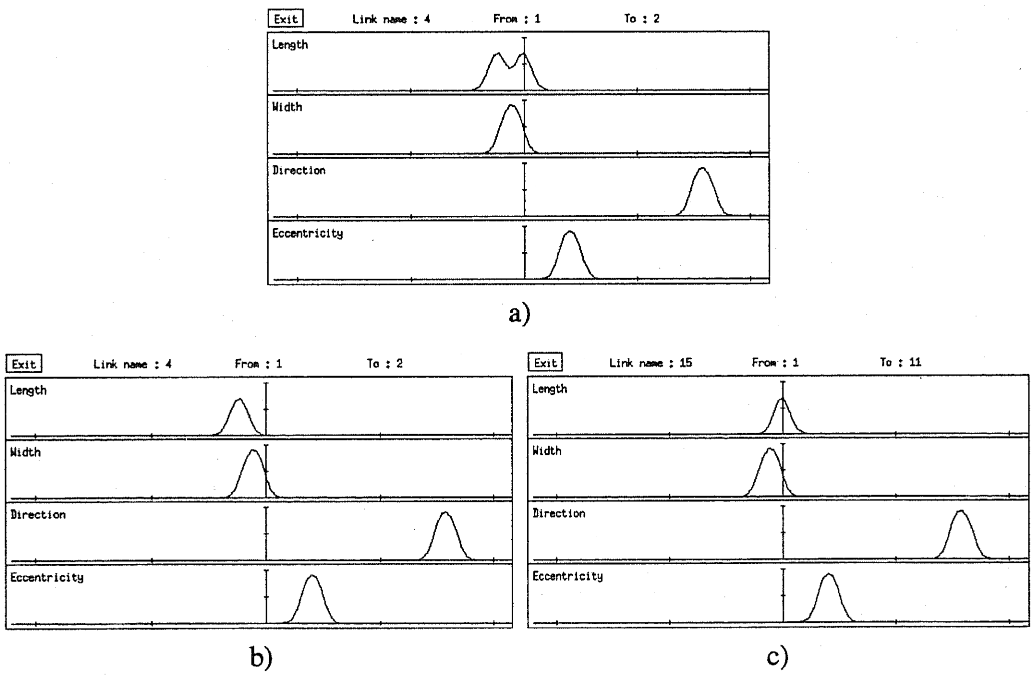


Figure 5.7: Specialization of a link: a) link before splitting, b) and c) resulting links after splitting length predicate

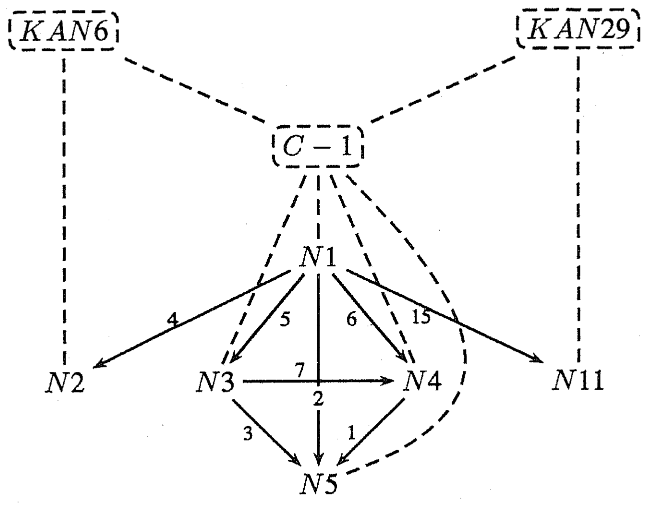


Figure 5.8: Resulting conceptual network after specialization

CHAPTER 6

Description

6.1 Purpose and Overview

In general, results of systems that can learn are not easily visualized by the human operator. Ueda and Suzuki [58] claim that graph models do not allow for a visualization at all, therefore, they propose that a completely different structure of the learning scheme be used. Although their criticism of graph models is warranted, graphs appear powerful enough to justify some further inquiries into how they could be visualized best.

The main purpose of this chapter is to derive and explain an approach for the visualization of relational networks. By visualization I do not mean to obtain any sort of image of the generalization results, rather anything that is easily understood by humans is appropriate. What is understood well by most people are explanations in natural language. Therefore, I will dwell on an approach that transfers relational networks into simple natural language. There are many problems associated with deriving explanations from some structure. For instance, what level of detail should be chosen for an optimal explanation? Furthermore, what aspects of the structure appear most interesting and important? It appears that such problems cannot easily be solved in a general way, and I do not claim to have them solved here. Ideally, any explanation should be interactive, first only providing the general structure. Parts of the structure that appear interesting to the human operator should then be explored in greater detail, always interactively checking back with the operator the direction in which to proceed.

The approach presented here is still on a relatively modest level. Only one, relatively coarse, description using natural language can be obtained at present, and no possibilities for interactive exploration of concepts have been added. Still, in most cases reasonable results can be obtained, as will further be shown in Chapter 7. The approach is loosely based on the work by Simmons and Slocum [53], who propose to generate English sentences from a form of semantic network. In the next section, I will first describe how to construct a semantic network from a relational network, by extracting various predicates. In Section 6.3 I will then outline how the extracted semantic network can be transformed into natural language, that is, English.

6.2 Creating Semantic Networks

The creation of semantic networks from relational networks starts by extracting a certain set of predicates that describe the most salient relations in the relational network. First, we assume that the relational network to be explained is connected. If this is not the case, then the method is applied separately to each connected part of the network. At present, six different predicates are extracted, three describing size relations and three describing relative orientation:

EQL-SIZ	Similar size, i.e., similar length and similar width
EQL-LEN	Similar length
EQL-WID	Similar width
DIR-PAR	Relative orientation approx. parallel
DIR-RIG	Relative orientation approx. at a right angle
DIR-VAR	Relative orientation variable

The first predicate, namely EQL-SIZ, can easily be extracted once the other two size predicates are given. I will describe this part later. The remaining five predicates are found by correlating links of the relational network that are connected to each other.

The approach for extracting the predicates EQL-LEN, EQL-WID, and DIR-PAR is the same, they only differ in the generalized predicate used. The extraction of those three predicates is based on the idea that equality should show up in generalized predicates, this means, two MACS m_i and m_j have the same characteristic, if there is another MACS m_k such that rel$(m_i, m_k) \approx$ rel(m_j, m_k), where rel(m_a, m_b) denotes the relation between MACS m_a and m_b. We again employ the interpretation of a generalized predicate as a probability distribution. As we have seen in Section 4.2, the multiplication of two generalized predicates f_i and f_j can be viewed as the joint probability distribution of their respective random variables. If we denote the maximum values of f_i and f_j by $\hat{f}_i$ and $\hat{f}_j$ respectively, then the maximum value of the joint probability distribution is given by $\hat{f}_j \cdot \hat{f}_i$ if and only if $\hat{f}_j$ and $\hat{f}_i$ occur at the same location. In practice, however, it is very unlikely to find two maxima in exactly the same position. In addition, we would like to extract relations that are similar, not exactly equal. Therefore, two generalized predicates f_i and f_j are judged equal if

$$\max(f_i \cdot f_j) > \alpha \hat{f}_i \cdot \hat{f}_j, \tag{6.1}$$

where the factor α is treated as a system parameter. In all applications $\alpha = 0.6$ has been fixed. Therefore, by using Eq. 6.1 together with the appropriate generalized predicates, predicates of type EQL-LEN, EQL-WID, and DIR-PAR can be extracted.

The approach for extracting predicates of type DIR-RIG, i.e., showing pairs of MACS that are approximately at a right angle to each other, is only slightly different. Instead of finding pairs of generalized predicates that are similar to each other, similar means in respect to Eq. 6.1, we find generalized predicates that are similar to an ideal generalized predicate representing a right angle relation. Such an ideal generalized predicate can easily be constructed by placing a Gaussian of amplitude $\hat{a}$ and standard deviation σ, where σ is the same as used in setting the initial Gaussian distributions as shown in Section 4.2, at 90°

or $\pi/2$. Again, if we take this ideal generalized predicate as f_i and $\hat{a}$ as $\hat{f}_i$, then, by using Eq. 6.1 we can extract those relations that represent directional changes of approximately $90°$.

The case is somewhat more complicated if we attempt to find MACS that appear to have variable direction in respect with each other. In general, generalized predicates representing variable directions will be spread over relatively a wide range, in comparison with a pure Gaussian distribution. This suggests to employ the ratio of the standard deviation of an approximate Gaussian, which could have been obtained through generalization of Gaussians with approximately the same location, to the width of the generalized predicate in question. The first question arising here is how to find the standard deviation of the control Gaussian. We saw in Section 5.2 that the result of the generalization $f_i, f_j \vdash g_i$ can be approximated by

$$g_i = \sum a_i' N(\mu_{ij}, \sigma_{ij}^2), \tag{6.2}$$

and the confidence value $cval_i$ tells us how often generalization took place. Therefore, the average amplitude $\overline{a}$ of a Gaussian is given by $\overline{a} = \hat{a}/\ cval$. In the case of a true Gaussian distribution g, the width of g at $\overline{a}e^{-\frac{1}{2}}$ is 2σ. In general, $k\sigma$ is the width of g at $\overline{a}e^{-\frac{k^2}{8}}$. Now, let us assume that g_i, that is, the result of the generalization, can approximately be obtained by a scaling of g, that is,

$$g_i = mg, \qquad \text{where } m = \hat{a}_i/\overline{a} = cval\,. \tag{6.3}$$

Therefore, putting the width of g_i at $\hat{a}_i e^{-\frac{l_i^2}{8}}$ equivalent to the width of g at $\overline{a}e^{-\frac{k^2}{8}}$, we have

$$l_i = 2\sqrt{-2\ln\left[\frac{\overline{a}}{\hat{a}_i}e^{-\frac{k^2}{8}}\right]} = 2\sqrt{-2\ln\left[\frac{1}{cval}e^{-\frac{k^2}{8}}\right]}. \tag{6.4}$$

Hence, l_i gives us an estimate on how the width, and thus the standard deviation, should change in the ideal case. By measuring the actual width of a generalized predicate f_i at a certain height $\hat{a}_i e^{-\frac{k^2}{8}}$, let us call that width $h_i\sigma$, and comparing it with $l_i\sigma$, we can get an idea on how far the generalized predicate was spread out during generalization. Whenever

$$\frac{h_i}{l_i} > \beta \tag{6.5}$$

is fulfilled, we judge the direction between the two MACS's as variable. Again, β is a system parameter that has been set to 2 for all the examples shown hereafter.

Finally, the predicate EQL-SIZ can be obtained by applying the following rule to the semantic network:

```
(RULE S1
  (IF (EQL-LEN SEG1 SEG2)
      (EQL-WID SEG1 SEG2))
  (THEN (EQL-SIZ SEG1 SEG2)))
```

where SEG1 and SEG2 refer to the two MACS's that are connected by some relation.

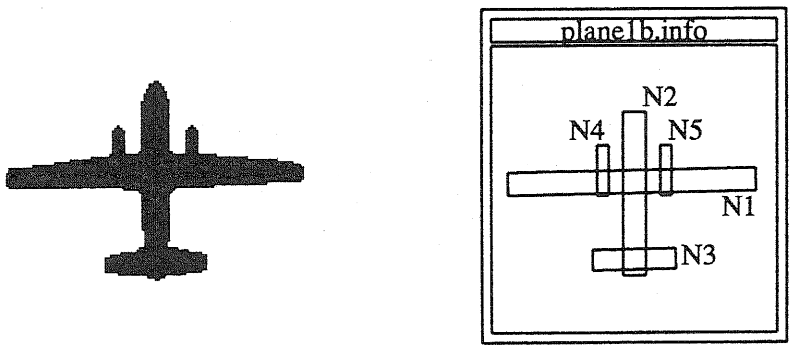
It is possible to group the MACS's according to the derived predicates. Three kinds of groupings are used at present: equal size groupings, parallel direction groupings, and fixed direction groupings. First, let us look at the case of equal size groupings and parallel direction groupings. At that stage, the semantic network is given as a list of binary predicates, e.g.,

```
(EQL-SIZ SEG3 SEG5)
(DIR-VAR SEG3 SEG7)
(EQL-LEN SEG5 SEG7)
        ⋮
```

Groups are defined as clusters of segments that share one or more characteristics. In the case of the equal size grouping, the characteristic is that all segments are of equal size. Such groups can easily be found by interpreting the semantic network as a graph. Any group sharing a certain characteristic can then be found as a maximum clique in respect to that characteristic. Since the number of predicates is rather small, the resulting graph is small as well, and we opt to find cliques by exhaustive search. If there are several cliques that involve the same segment, then the clique with the largest number of segments only is retained. As for the case of fixed direction groupings, we can consider the variable direction predicates as introducing a dichotomy into the set of all MACS's. By dichotomy I mean that they possibly divide the set of all MACS's into two or more subsets, the relation between subsets being that of variable direction, while no variable directions can be found within a subset.

As the different groupings might be inconsistent, they are given a hierarchical ordering. That means, fixed direction groupings are preferred to equal size groupings, which, in turn, are preferred to parallel direction groupings.

At this stage, the transformation of a relational network into a simple semantic network is complete. For purposes of illustration, let us look at the example in Fig. 6.1. Several relations appear conspicuously: the two engines are of approximately the same



a) Input image b) Decomposition

Figure 6.1: Example image

size, the fuselage and the wings form a right angle, etc. The derived semantic network, according to the procedure outlined above, is shown in Fig. 6.2.

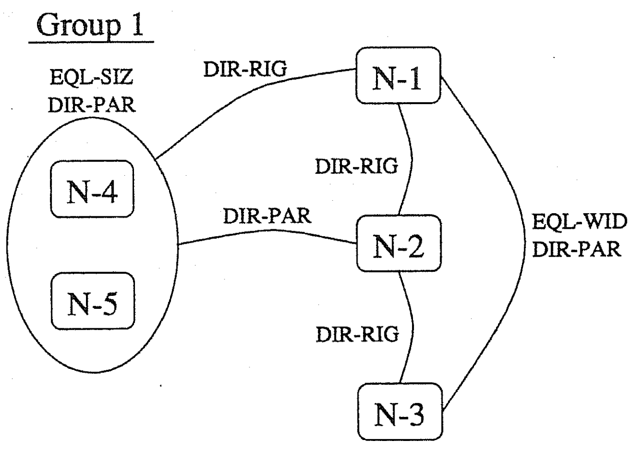


Figure 6.2: Semantic network for plane in Fig. 6.1

6.3 Descriptions in Natural Language

As mentioned earlier, the transformation of the obtained semantic network into English is strongly based on the work by Simmons and Slocum [53]. In contrast to their work, however, our application is much more restricted in scope. It is therefore possible to obtain reasonable results with a very small grammar and dictionary. Simmons and Slocum propose to represent a grammar as a state transition network. For instance, consider the following grammar

$$\begin{aligned}
&NP \rightarrow (DET) + (ADJ^*) + N + (PP^*)\\
&PP \rightarrow PREP + NP\\
&S \rightarrow NP + (AUX) + VP\\
&S \rightarrow AUX + NP + VP\\
&VP \rightarrow V + (NP) + (PP^*)
\end{aligned}$$

which is a context-free phrase structure grammar where parentheses indicate optionality and the asterisk indicates one or more occurrences. The state network representing this grammar is given in Fig 6.3, where nodes correspond to states and links denote possible transitions.

In general, the elements of a semantic network are concepts and relations. In our application, the concepts are implicitly given as the segments of the decomposition or MACS's. Hence only relations are relevant for the discourse generation. While travelling through the state network representing our grammar, state transitions can be constrained by the information contained in the semantic network. With each state transition, a daemon is associated. The daemons can roughly be grouped into three categories: transition daemons, grammar daemons, and terminal daemons. The duties fulfilled by those daemons are manifold, among others, the transition daemons decide their own applicability and provide information on which segments of the description we are dealing with at the moment. The grammar daemons decide whether we are about to start a subject phrase or an object phrase, and the terminal daemons will produce actual output of some phrases as parts of the discourse. To be more specific, the information contained in the semantic network will be

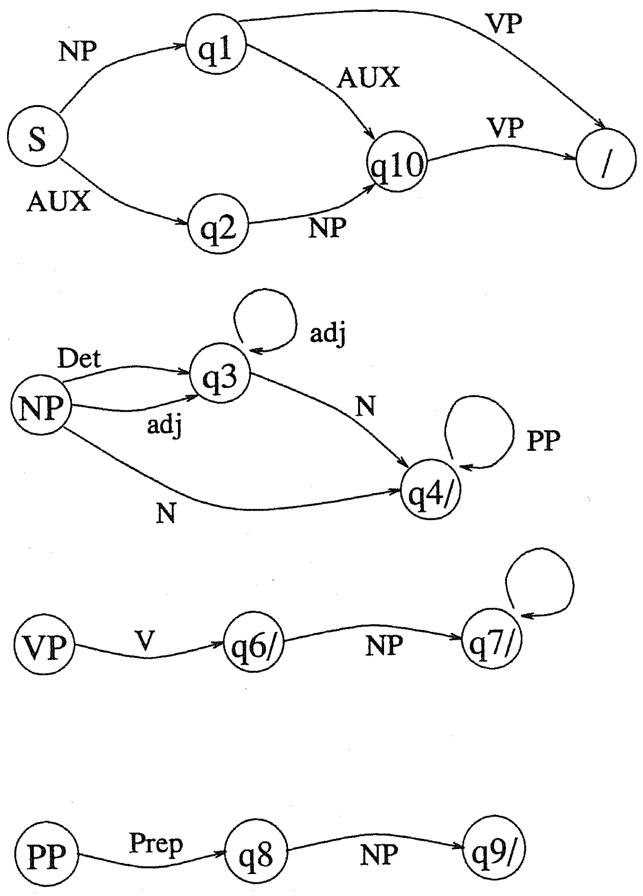


Figure 6.3: Recursive transition network, from [53]

primarily exploited by the transition daemons, while the actual output is generated by the terminal daemons.

While Simmons and Slocum approach the problem of discourse generation from a strictly grammatical point of view, this does not appear to be very fruitful in our case. I therefore opted to use semantic inference, rather than grammatical inference. The approach thus reduces to putting together patterns and fragments of sentences that sufficiently describe the domain in question. The complete grammar used in the present approach is given in Appendix F. Note that the lexicon is directly built into the grammar, it is therefore sufficient to just decide whether to use singular or plural, instead of having to look for the appropriate word as well.

Let us again look at the example from Fig. 6.1, with its semantic network in Fig. 6.2. As can be seen from Appendix F, the description starts with a variable group, a size group, or a rest description. The semantic network in Fig. 6.2 shows one group whose elements have equal size and are parallel in respect with each other. Hence, the daemon for variable groups is not applicable, but the one for equal size groups is. The explanation for an equal size group starts with a prologue and then each group is explained in respect to the characteristics its members share. Next, the segments that do not belong to any group are introduced, again with a prologue, before each segment is treated separately in how it relates to the other segments. The resulting natural language output is given in Fig. 6.4. The above

```
There is 1 group.
Group G-1 consists of elements N-5, and N-4 which are
  of same size and parallel to each other.
Elements N-1, N-3, and N-2 do not belong to any group.
Element N-2 is parallel to G-1.
Element N-3 is parallel to N-1 and of same width
  as N-1.
Element N-1 is at a right angle to N-2, and G-1.
```

Figure 6.4: Natural language explanation of plane from Fig. 6.1

example, though somewhat simplistic, shows the workings of the employed approach quite clearly. Further examples, notably of more complex concepts, will be given in Chapter 7.

The purpose for implementing the description stage were twofold. First, an explanation of the obtained results in natural language appears to be necessary, as outlined above. Second, I attempted to show that with a very small effort, many systems could be rendered much more user-friendly. Only a system that can explain itself and that can somehow communicate with the user will have any practical success. Although the present implementation of the explanation stage is still far from actually reaching the goal of communicating with the user, I believe that it is a step in the right direction.

CHAPTER 7

Applications

This chapter is divided into three main sections, outlining three different aspects of the system. The major parts of the explanations so far have concentrated on the aspect of acquiring models from some visual input. This aspect will be described by some examples in Section 7.1. However, the basic outline of the system, as explained before, can be used for different tasks as well. The first such task is the one of conceptual clustering. Conceptual clustering has been defined by Michalski [35] as the formation of classes of objects that are describable by a concept from a predefined concept class. In this respect, it is not entirely adequate to use conceptual clustering to describe the results of the hierarchical abstraction as obtained by the present system, and I will, therefore, refer to it as hierarchical clustering. Nevertheless, as is shown in Section 7.2, some grouping of sets of instances, according to sub-concepts shared by those instances, can be observed. The final application concerns the problem of instance recognition, and is outlined in Section 7.3.

With all examples, I took precautions to ensure the uniformity of the approach. This means, as far as possible the same set of parameters were used for all examples. The main difficulties encountered with this approach were found during the decomposition of instances (see Section 3.4). As was pointed out in that section, the used skeletonization might fail with bigger shapes. This was indeed the case for all the examples using Kanji characters. For those examples, the skeletonization approach by Jang and Chin [28] has been used. In addition, as some of those shapes show transitions from straight segments to slightly curved segments, the threshold for approximate convexity had to be relaxed. In the case of the Kanji characters, it was set to 0.96 for expansion and 0.94 for merging, as contrasted to 0.99 and 0.98 for all the other examples. As for matching and generalization, the same set of parameters were used for all examples. Those parameters include the standard deviations of the generalized predicates, which were set to $\sigma = 0.075$ for all generalized predicates, and the threshold for unfeasible matches, which was made to correspond to 4σ, again for all cases.

7.1 Model Acquisition

Model acquisition has been described as the main purpose of the present system, and it, therefore, warrants some further illustration. I choose three different sets of instances to outline the performance of the system.

7.1.1 Plane Model I

7.1.1.1 Input Images

For the sequence of input images in Figs 7.1–7.11, the original image is shown on the left, to the right of it the decomposition results and the relational network.

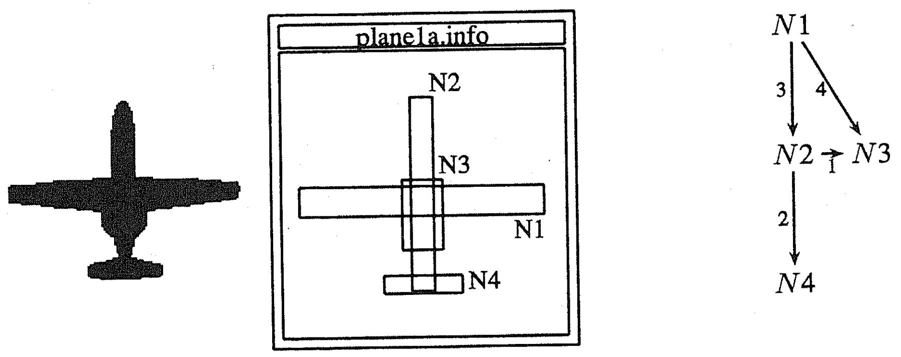


Figure 7.1: Example I: Input 1

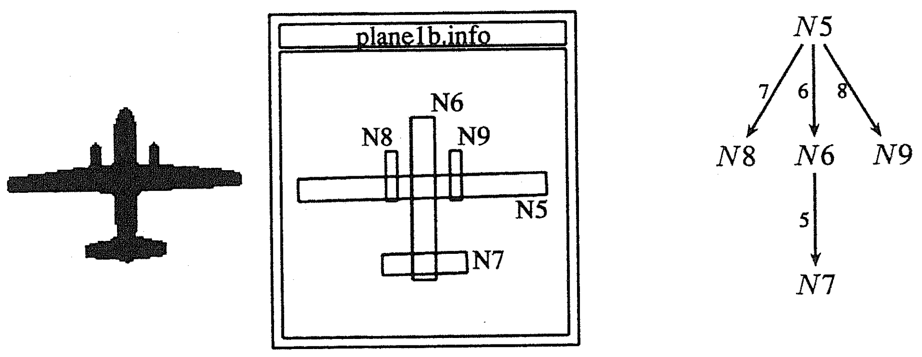


Figure 7.2: Example I: Input 2

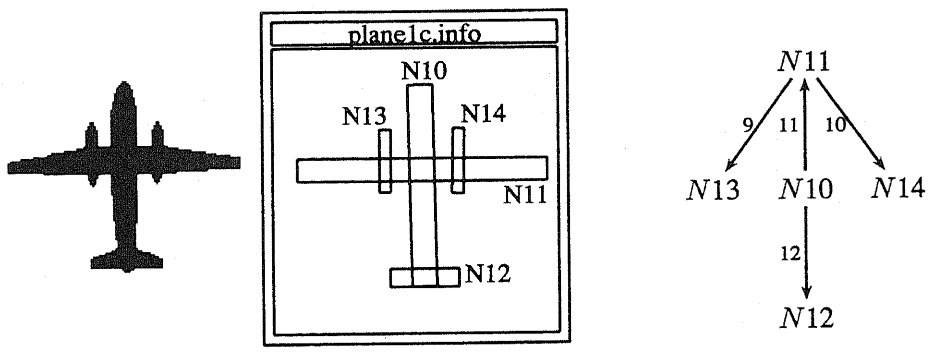


Figure 7.3: Example I: Input 3

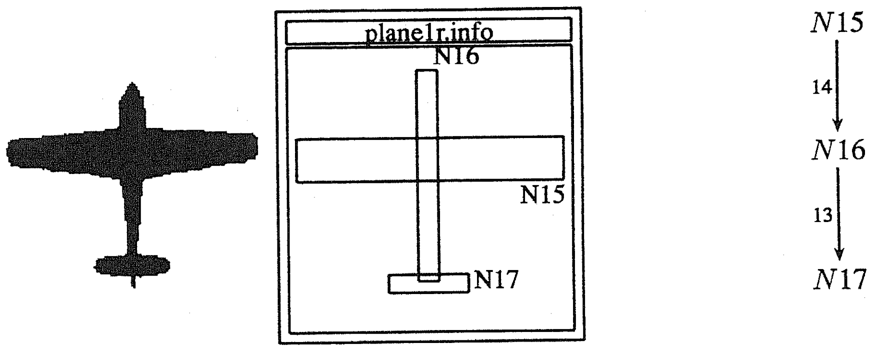
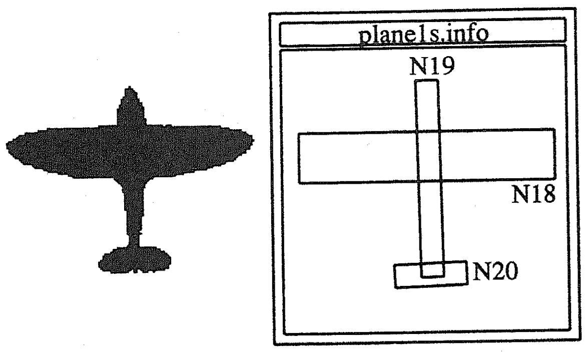
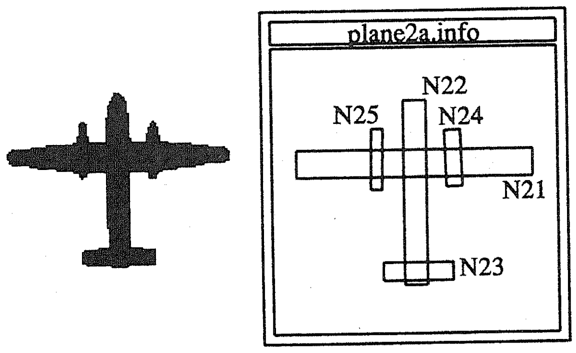


Figure 7.4: Example I: Input 4



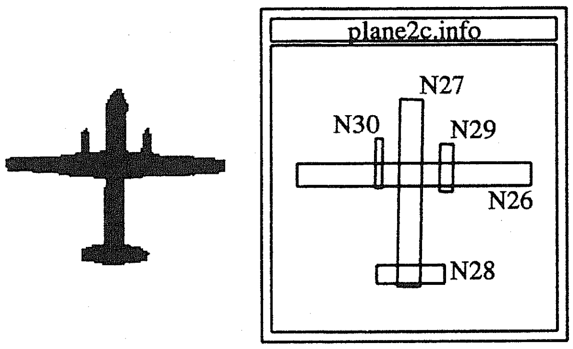
$$\begin{tikzcd}
N18 \arrow[d, "16"'] \\
N19 \arrow[d, "15"'] \\
N20
\end{tikzcd}$$

Figure 7.5: Example I: Input 5



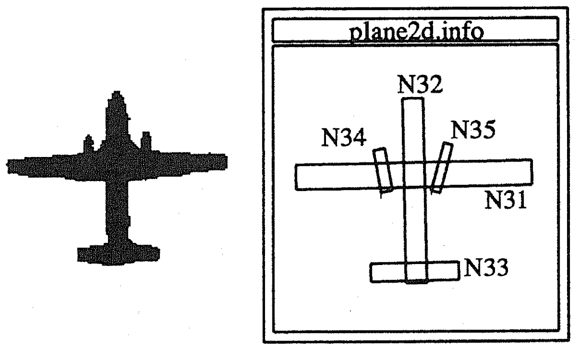
$$\begin{tikzcd}
 & N21 \arrow[dl, "20"'] \arrow[d, "18"] \arrow[dr, "19"] & \\
N25 & N22 \arrow[d, "17"'] & N24 \\
 & N23 &
\end{tikzcd}$$

Figure 7.6: Example I: Input 6



$$\begin{tikzcd}
& N26 \arrow[dl, "24"'] \arrow[d, "22"] \arrow[dr, "23"] & \\
N30 & N27 \arrow[d, "21"] & N29 \\
& N28 &
\end{tikzcd}$$

Figure 7.7: Example I: Input 7



$$\begin{tikzcd}
& N31 \arrow[dl, "27"'] \arrow[d, "26"] \arrow[dr, "28"] & \\
N34 & N32 \arrow[d, "25"] & N35 \\
& N33 &
\end{tikzcd}$$

Figure 7.8: Example I: Input 8

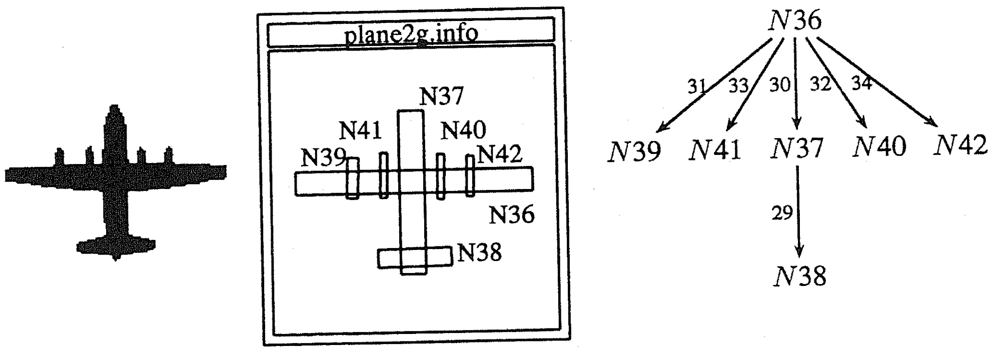


Figure 7.9: Example I: Input 9

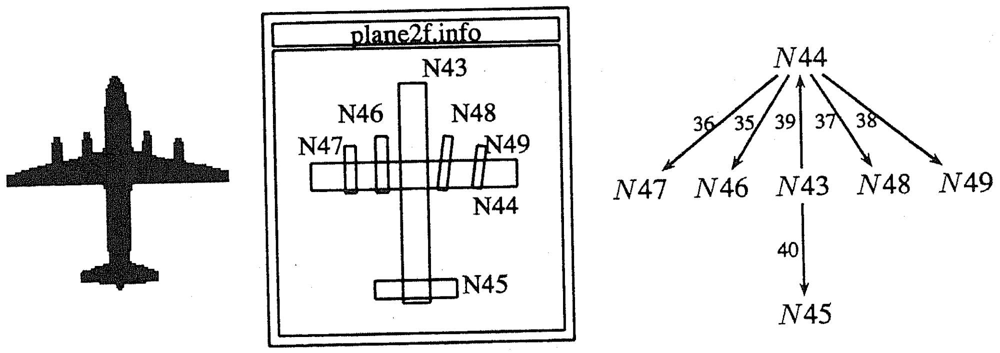


Figure 7.10: Example I: Input 10

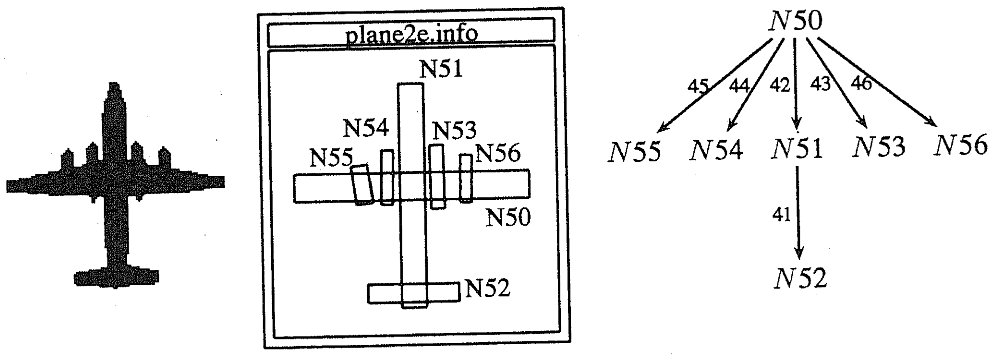


Figure 7.11: Example I: Input 11

7.1.1.2 Generalization

Generalization proceeds sequentially, according to the input sequence in the previous section. As there are 11 input images and the first input is automatically promoted to a concept, generalization is done in 10 steps. While the intermediate steps have been skipped, the list of matched nodes and links is shown in Fig. 7.12, while with the resulting topological generalization is shown in Fig. 7.13.

Node 1, 5, 11, 15, 18, 21, 26, 31, 36, 44, and 50 $\vdash$ Node 1

Node 2, 6, 10, 16, 19, 22, 27, 32, 37, 43, and 51 $\vdash$ Node 2

Node 4, 7, 12, 17, 20, 23, 28, 33, 38, 45, and 52 $\vdash$ Node 4

Node 8, 14, 24, 29, 34, 41, 46, and 53 $\vdash$ Node 8

Node 9, 13, 25, 30, 35, 40, 48, and 54 $\vdash$ Node 9

Node 39, 47, and 56 $\vdash$ Node 39

Node 42, 49, and 55 $\vdash$ Node 42

Link 2, 5, 12, 13, 15, 17, 21, 25, 29, 40, and 41 $\vdash$ Link 2

Link 3, 6, 11, 14, 16, 18, 22, 26, 30, 39, and 42 $\vdash$ Link 3

Link 7, 10, 19, 23, 27, 33, 35, and 43 $\vdash$ Link 7

Link 8, 9, 20, 24, 28, 32, 37, and 44 $\vdash$ Link 8

Link 31, 36, and 46 $\vdash$ Link 31

Link 34, 38, and 45 $\vdash$ Link 34

Figure 7.12: Example I: Matched nodes and links

To add some further explanation, node N1 corresponds to the main-wing, while N2 and N4 stand for the fuselage and the tail-wings, respectively. The remaining nodes all correspond to engines, either connected to the main-wing or to the fuselage.

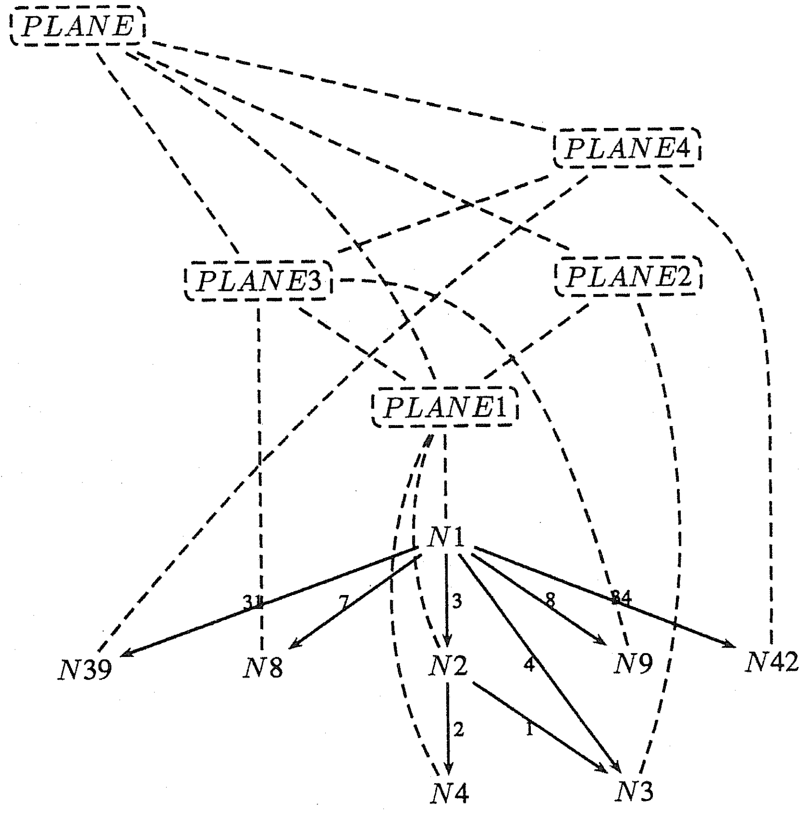


Figure 7.13: Example I: Structural generalization

As becomes clear from the above generalization results, besides the concept of a plane, four subconcepts, each referring to one valid type of plane, have been identified. The hierarchical grouping among those subconcepts, identifies the subconcept PLANE1 as the least common factor. In other words, for an instance to be recognized as a plane, it should at least consist of segments N1, N2, and N4, or of the main-wings, the fuselage and the tail-wings.

7.1.1.3 Generalized Predicates

In Fig. 7.14, a graphic representation of the generalized predicates is shown. Especially noteworthy is the difference between links 1 and 4 on one hand and the other links on the other hand. As can be seen from the list of matched nodes and links, both, link 1 and 4 have never been matched. They, therefore, remained unchanged during the whole generalization process. The generalized predicates of the other links have not only grown in amplitude, but some of them have been spread out or even have a distinct bimodal characteristic.

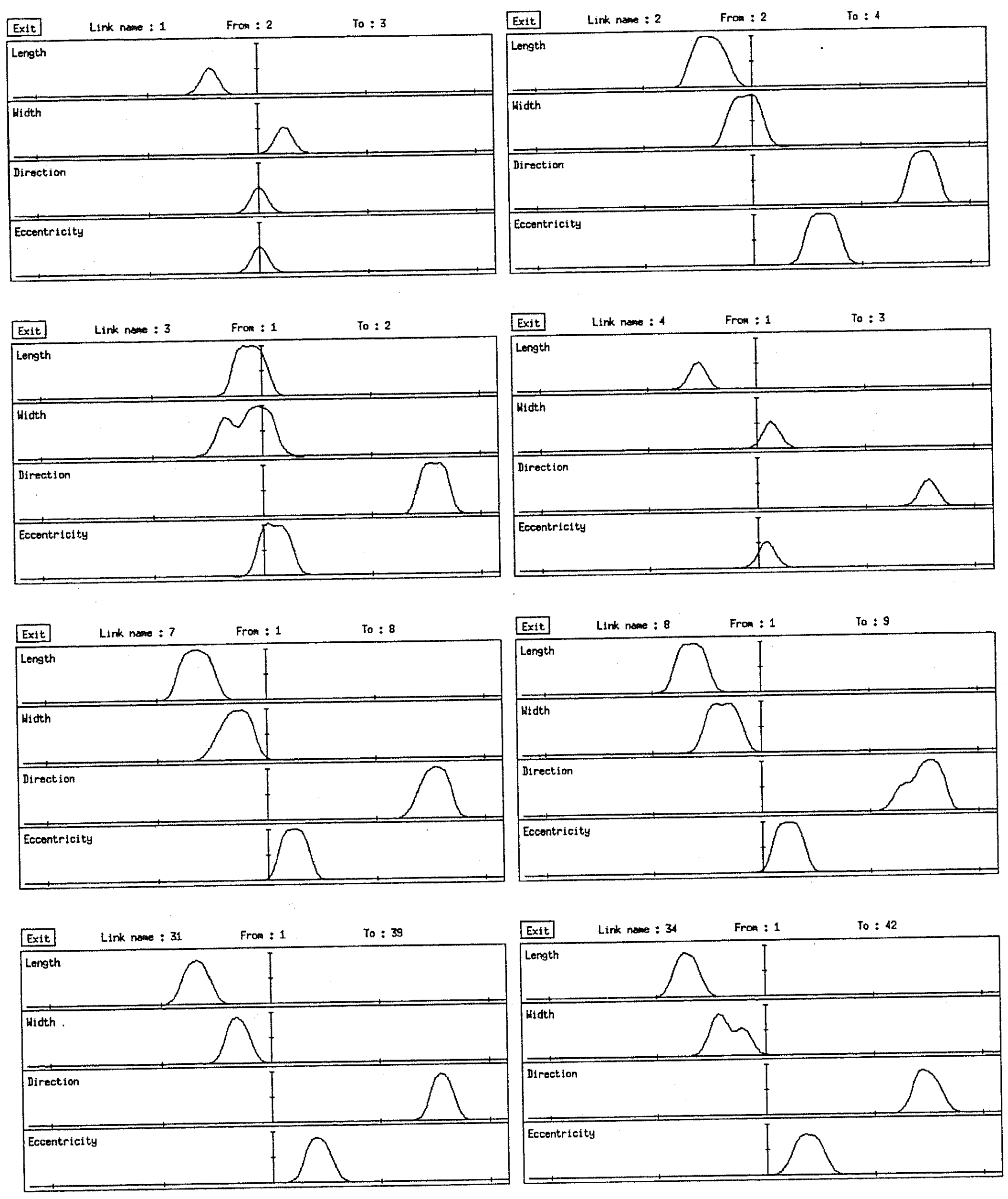


Figure 7.14: Example I: Generalized predicates

7.1.1.4 Explanations

The verbal description of the generalization results, in Fig. 7.15, reveals that the four engines can be grouped together. They do not only share the feature of being of equal size, but they are found to be parallel as well. Furthermore, the main-wings are found to be parallel to the tail-wings.

```
There is 1 group.
Group G-1 consists of elements N-8, N-39, N-42, and N-9
  which are of same size and parallel to each other.
Elements N-3, N-2, N-4, and N-1 do not belong
  to any group.
Element N-1 is parallel to N-4 and of same width
  as N-4.
Element N-4 is at a right angle to N-2 and of same
  length as N-3.
Element N-2 is parallel to N-3, and G-1.
Element N-3 is parallel to G-1 and of same length as G-1.
```

Figure 7.15: Example I: Verbal description

The semantic network representation that was used for obtaining the above explanation is shown schematically in Fig. 7.16. Certain relations that were extracted for the semantic network, as for instance the DIR-RIG relation between the main-wings and the four engines, have not been used for the verbal description. This is done in order to keep the description simple and easy to understand.

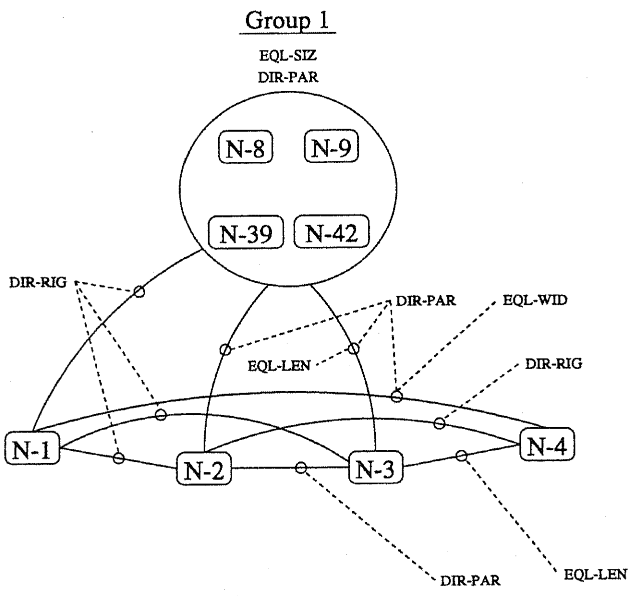


Figure 7.16: Example I: Semantic network

7.1.2 Plane Model II

The second set of instances shows different types of planes. Although only four instances are used, shown in Figs. 7.17–7.20, the generalization appears difficult, due to the widely changing features of the instances.

7.1.2.1 Input Images

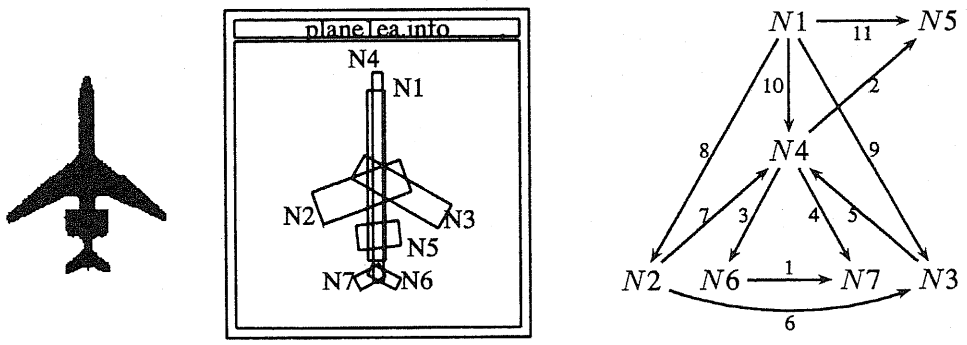


Figure 7.17: Example II: Input 1

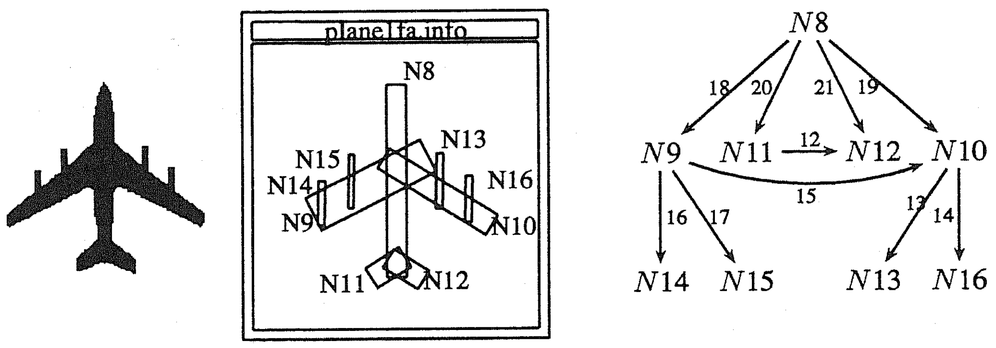


Figure 7.18: Example II: Input 2

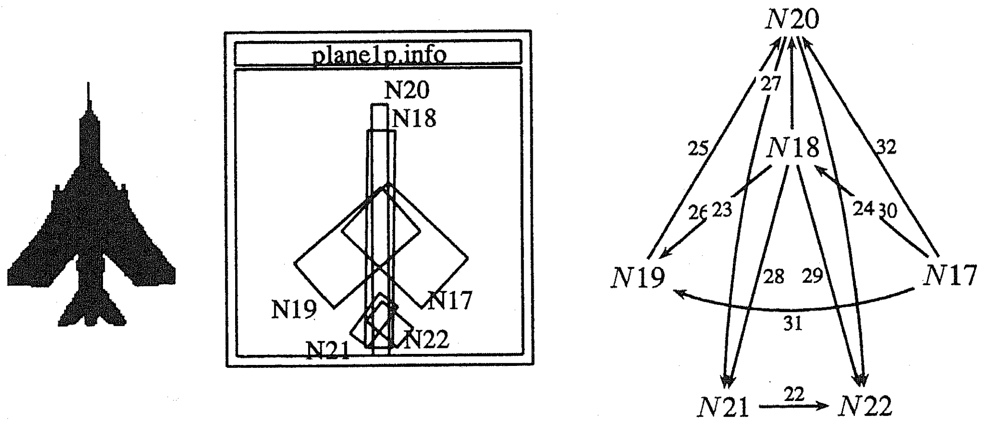


Figure 7.19: Example II: Input 3

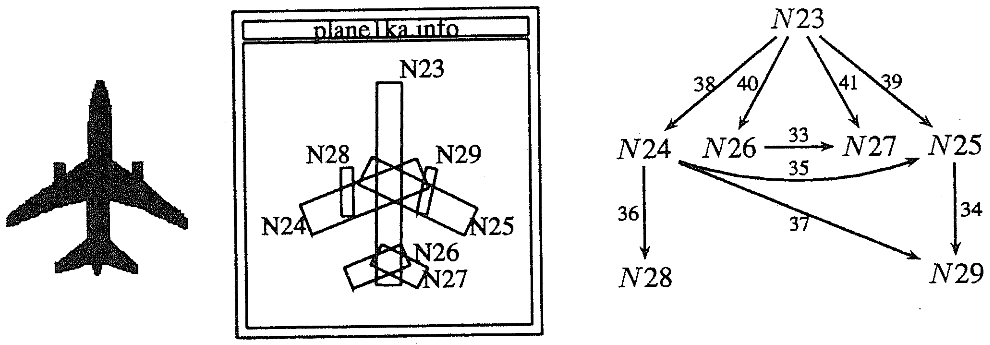


Figure 7.20: Example II: Input 4

7.1.2.2 Generalization

Generalization proceeds in three steps, according to Fig. 7.21, leading to the final conceptual network as shown below. Tracing the matched nodes back to the input images reveals that N1 and N4 correspond to the fuselage, N2 and N3 correspond to the main-wings, and so on. Again, it is possible to identify some derived partial concepts, corresponding to valid representations for a plane. To keep the representation simple, those partial concepts have been skipped here.

Node 1 and 18 ⊢ Node 1

Node 2, 9, 19, and 25 ⊢ Node 2

Node 3, 10, 17, and 24 ⊢ Node 3

Node 4, 8, 20, and 23 ⊢ Node 4

Node 6, 11, 22, and 26 ⊢ Node 6

Node 7, 12, 21, and 27 ⊢ Node 7

Node 13 and 28 ⊢ Node 13

Node 16 and 29 ⊢ Node 16

Link 1, 12, 22, and 33 ⊢ Link 1

Link 3, 20, 24, and 40 ⊢ Link 3

Link 4, 21, 23, and 41 ⊢ Link 4

Link 5, 19, 32, and 38 ⊢ Link 5

Link 6, 15, 31, and 35 ⊢ Link 6

Link 7, 18, 25, and 39 ⊢ Link 7

Link 8 and 26 ⊢ Link 8

Link 9 and 30 ⊢ Link 9

Link 10 and 27 ⊢ Link 10

Link 14 and 37 ⊢ Link 14

Link 13 and 36 ⊢ Link 13

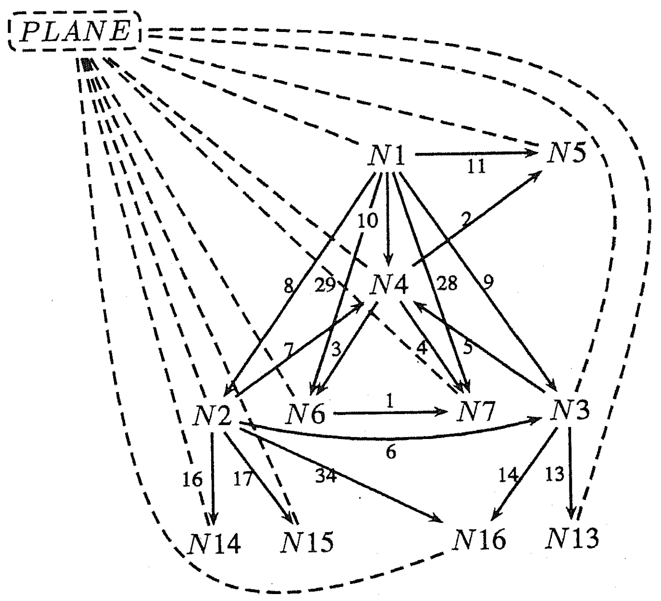


Figure 7.21: Example II: Generalization

Although the resulting conceptual network appears to be the correct generalization, there is one problem that can be found by tracing the list of matched nodes. Namely, nodes N28 and N29, that is, the two engines of the fourth instance, have been matched to nodes N13 and N16, respectively. Looking at the second instance reveals that those two nodes are both attached to the right main-wing and, hence, the symmetry has not been recognized. The reason for this mismatch is the following. As can be seen from the relational network of the fourth instance, nodes N24 and N29 share a connection. This connection amounts to saying that both engines can be attached to the same main-wing. In addition, the angles and the width relations between N24 on one hand and N28 and N29 on the other hand resemble the situation of the right main-wing of the second instance, this mismatch occurs. The actual reason can therefore be traced back to the decomposition of the fourth instance.

7.1.2.3 Generalized Predicates

As the resulting conceptual network is quite complex, here I only show some generalized predicates in Fig. 7.22. They include the relation between the tail-wings (link 1), between the main-wings (link 6), and the relations between the fuselage and the two main-wings (link 8 and 9, respectively). In all of those links, the direction predicate has either been spread out or shows a bimodal characteristic. This shows the widely varying angles of the main-wings and the tail-wings of the instances.

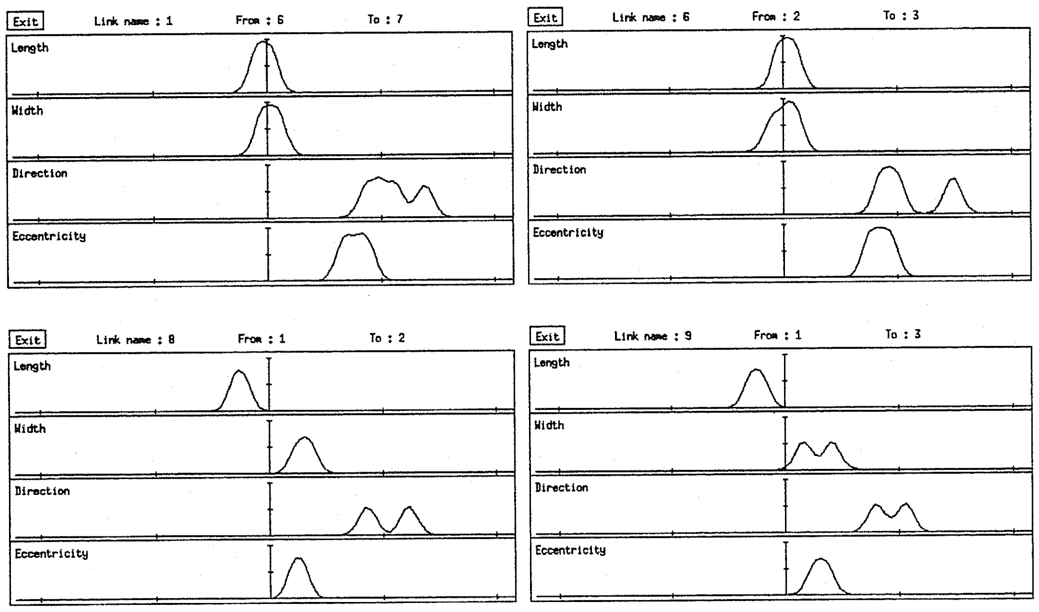


Figure 7.22: Example II: Generalized predicates

7.1.2.4 Explanations

The verbal description for this example, shown in Fig. 7.23, is not able to capture any grouping. This can be understood by noticing the widely changing instances, which do not allow for common features to emerge strongly enough. Nevertheless, the relations of particular importance, like the direction relations, have been captured. Furthermore, the widely spread direction predicates have now been translated into variable directions. Although this is not correct from a semantic point of view, the generalized predicates clearly show such a trend.

```
Element N-1 is parallel to N-4 and of same width
  as N-16, N-6, and N-7.
Element N-7 is parallel to N-3 and of same length
  as N-6.
Element N-6 is at variable angle to N-4, and N-7.
Element N-2 is at variable angle to N-1, N-4, and N-3.
Element N-4 is parallel to N-13 and of same width
  as N-15.
Element N-5 is at a right angle to N-4, and N-1.
Element N-3 is at variable angle to N-1, and N-16.
Element N-13 is of same size as N-16.
Element N-14 is parallel to N-15 and of same length
  as N-16.
```

Figure 7.23: Example II: Verbal description

The semantic network representation for this example is rather complicated and badly structured, as shown in Fig. 7.24. From this representation it is quite difficult to see what relations are important, a fact that is also mirrored in the above description. Nevertheless, the important relations have been captured.

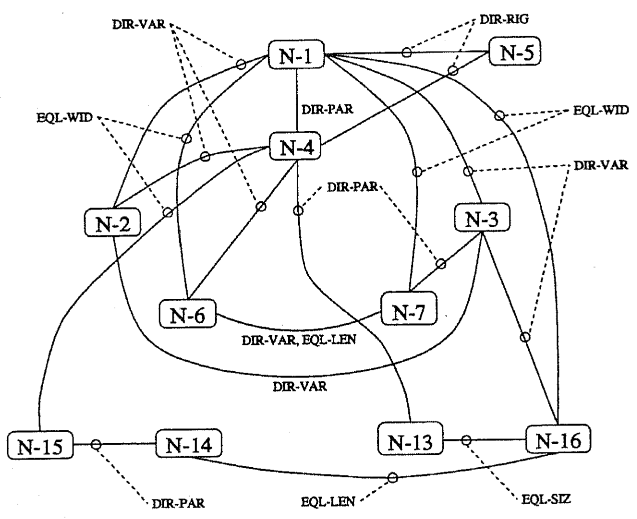
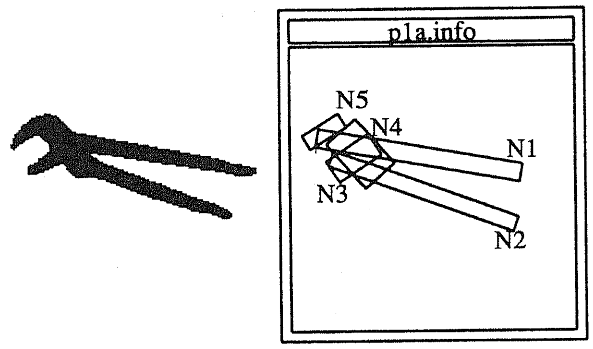


Figure 7.24: Example II: Semantic network

7.1.3 Tool Model

The final example for model acquisition shows the generalization of a dynamically changing shape. The sequence of the pair of tongs in Figs. 7.25–7.28 shows an increasing opening angle, together with changes in direction and size.

7.1.3.1 Input Images



$$\begin{tikzcd}
N5 & \\
N1 \arrow[u, "8"] \arrow[r, "6"] \arrow[d, "5"'] \arrow[dr, "7"] & N3 \arrow[ul, "2"'] \arrow[d, "1"] \\
N2 \arrow[ur, "3"] \arrow[r, "4"'] & N4
\end{tikzcd}$$

Figure 7.25: Example III: Input 1

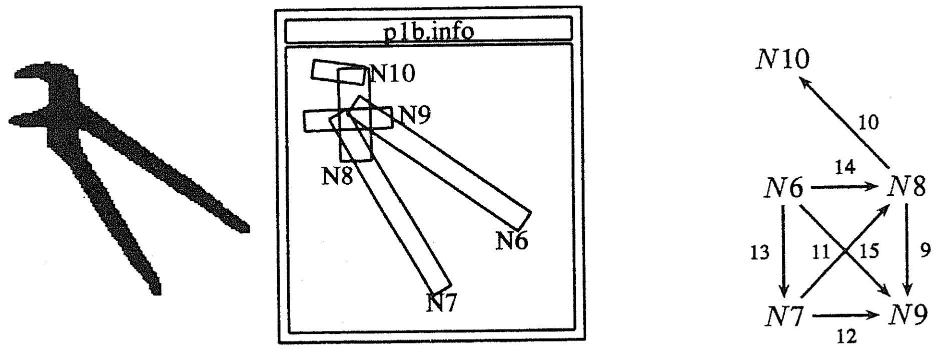


Figure 7.26: Example III: Input 2

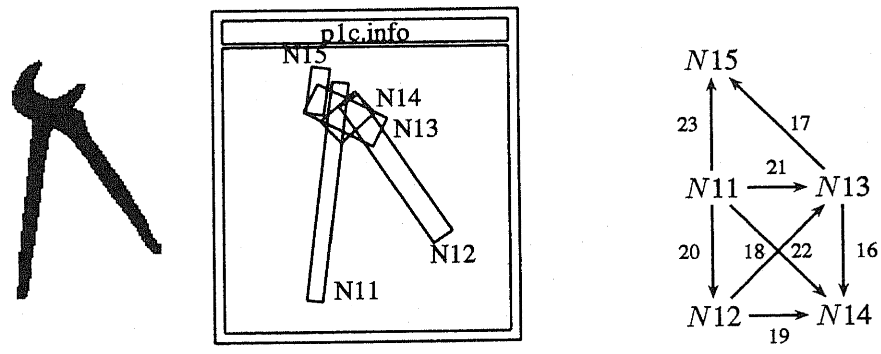


Figure 7.27: Example III: Input 3

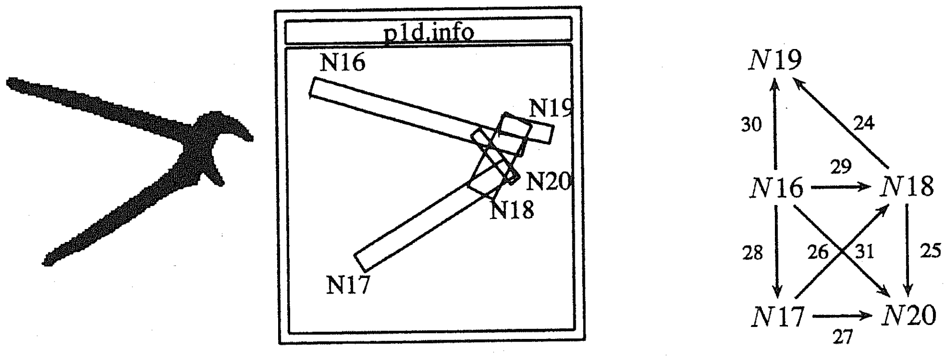


Figure 7.28: Example III: Input 4

7.1.3.2 Generalization

The generalization results in Fig. 7.29 reveal that all instances have been matched completely. It is therefore not possible to identify any partial concepts in the conceptual network.

Node 1, 6, 11, and 16 ⊢ Node 1

Node 2, 7, 12, and 17 ⊢ Node 2

Node 3, 8, 13, and 18 ⊢ Node 3

Node 4, 9, 14, and 20 ⊢ Node 4

Node 5, 10, 15, and 19 ⊢ Node 5

Link 1, 9, 16, and 25 ⊢ Link 1

Link 2, 10, 17, and 24 ⊢ Link 2

Link 3, 11, 18, and 26 ⊢ Link 3

Link 4, 12, 19, and 27 ⊢ Link 4

Link 5, 13, 20, and 28 ⊢ Link 5

Link 6, 14, 21, and 29 ⊢ Link 6

Link 7, 15, 22, and 31 ⊢ Link 7

Link 8, 23, and 30 ⊢ Link 8

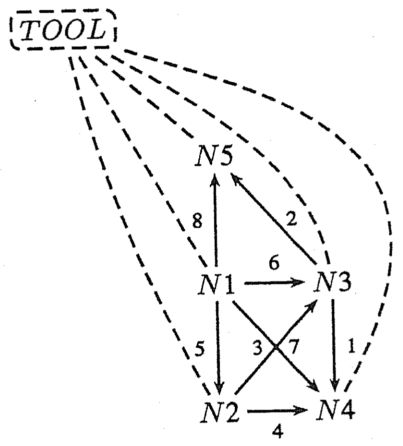


Figure 7.29: Example III: Generalization

7.1.3.3 Generalized Predicates

More interesting than the structural generalization is the generalization of the generalized predicates, as shown in Fig. 7.30. Especially noteworthy is the spreading out of the direction predicates of links 5 and 6. Going back to the instances shows that those links correspond to the relations between the handles and between the upper handle and the pivot, respectively. Clearly, this mirrors the change of direction as caused by opening the pair of tongs. On the other hand, the direction predicates of links 2 and 3 have hardly changed, implying that the direction between segments N2, N3, and N5 is fixed.

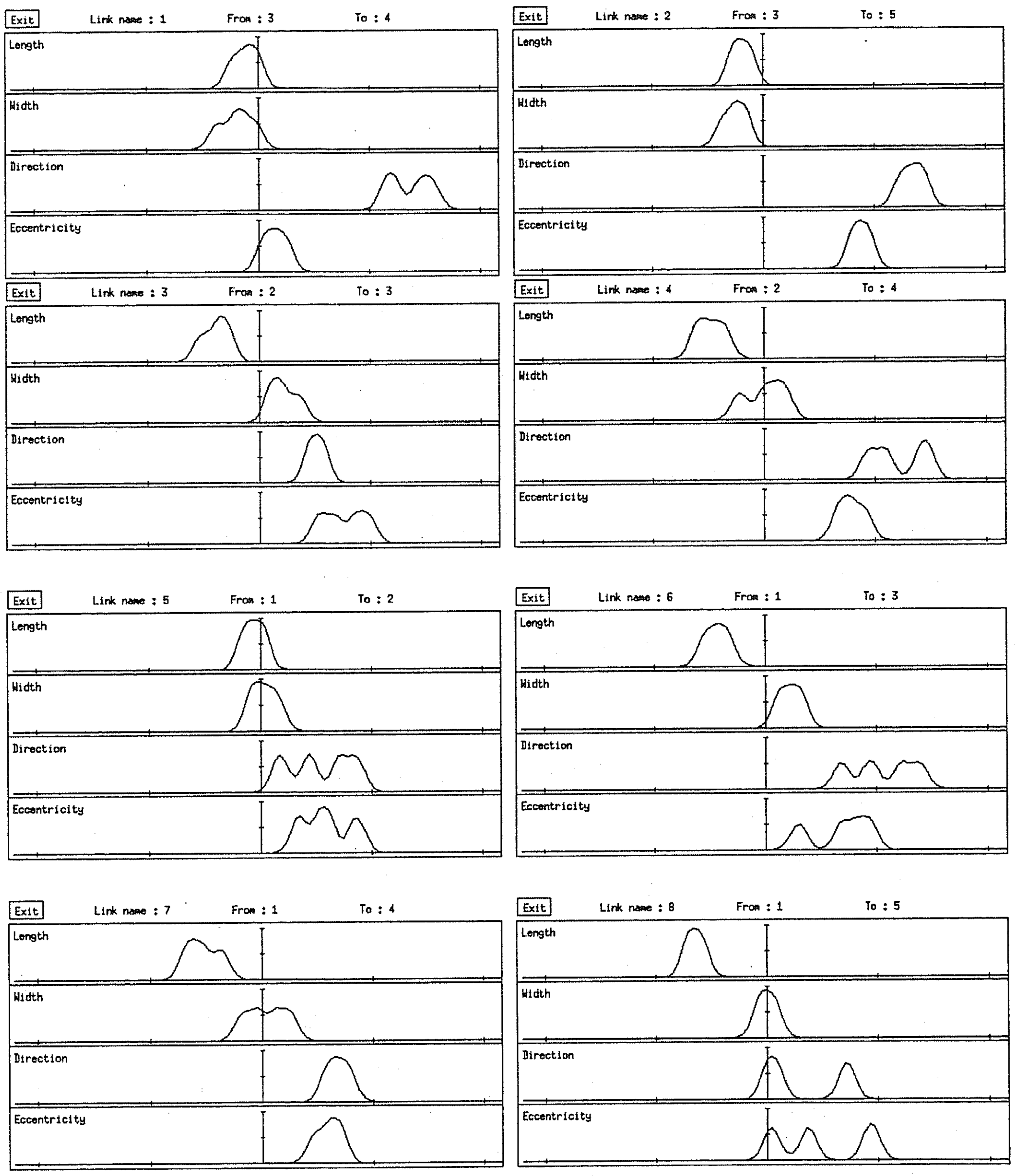


Figure 7.30: Example III: Generalized predicates

7.1.3.4 Explanations

In this case, illustrated in Fig. 7.31, the variable direction grouping that was found is more than adequate, although it did not perfectly identify the group consisting of elements N-2, N-3, and N-5. It might be arguable whether the constraints for recognizing variability are too strict in this case.

```
There are 2 groups which have variable angle
  with respect to each other.
Group G-1 consists of elements N-2, and N-3.
Element N-2 is of same size as N-1.
Element N-3 is at a right angle to N-5 and
  of same width as N-4.
Group G-2 consists of elements N-4, and N-1.
Element N-4 is parallel to N-5 and of same
  size as N-5.
Element N-1 is parallel to N-5 and of same
  width as N-5, and N-4.
Element N-5 does not belong to any group.
Element N-5 is of same width as N-2.
```

Figure 7.31: Example III: Verbal description

Due to the missed attributing of segment N5 to group 1, the derived semantic network in Fig. 7.32 shows a symmetry that cannot be found in the instances. However, this symmetry is only structural, and looking at the actual predicates reveals some of the real structure of the pair of tongs.

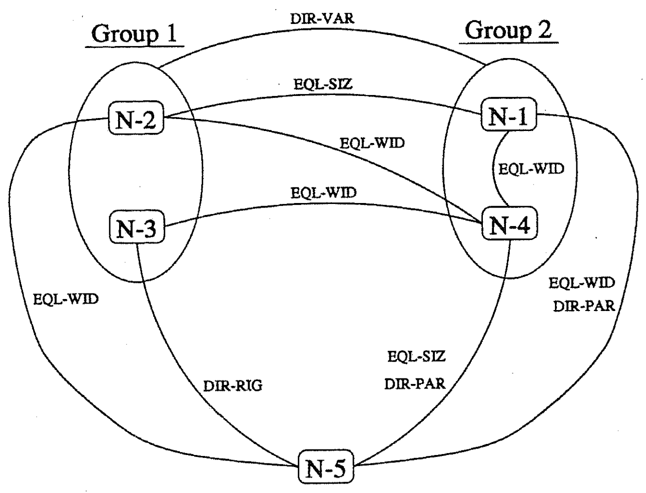


Figure 7.32: Example III: Semantic network

7.1.4 Discussion

The three examples presented in the context of model acquisition outline the basic performance of the proposed system. We can try to analyze those examples from the four viewpoints of decomposition, matching, generalization, and explanation, which correspond to the four main stages of the system. Decomposition appears to yield good results in almost all cases. While no important details have been missed, almost no superfluous segments were extracted either. The biggest problem with decomposition appears to be that in some cases the shape of the original detail was not captured well. This is, for instance, shown in the extraction of some of the engines of the planes, whose direction and width vary too much. The result of the matching appears reasonable as well. The only mismatch detected was the matching of the engines as explained in the second example. Of course, such a mismatch could be overcome by employing symmetry constraints. However, how to extract and deal with symmetries has not been addressed and remains and open problem. As for generalization, we can restrict the discussion to the generalization of the predicates, as topological generalization is a direct outcome of the matching. It might be desirable to introduce some further generalization operators on generalized predicates. For instance, in the case of the direction relation between the handles of the pair of tongs, the generalized predicate shows a trimodal characteristic. In accordance with generally used generalization heuristics, it might be of advantage to smooth out those modes. Finally, although explanation is helpful in some cases, generally, its implementation seems too weak to give reasonable results, especially for more complicated scenes with few instances. Besides, the transformation of the relational network into a semantic network representation relies on some preset thresholds, for instance for the extraction of variable direction predicates (see Section 6.2). It seems that those thresholds are rather difficult to set in a general context.

Although several problems are remaining, the obtained results are accurate and reliable. Especially, the second example emphasizes the actual possibilities of the approach. It seems that at least for the acquisition of models from binary images, the chosen approach gives the expected results.

7.2 Hierarchical Clustering

Hierarchical clustering refers to the abstraction that becomes possible during generalization (see Section 5.3). Clearly, such a clustering cannot be unique, but it can still be used to identify common structures among instances. The used set of instances, which are shown in Fig. 7.33, have all been acquired from a real Gothic font through scaling and rasterization. The resulting hierarchical abstraction is outlined in Section 7.2.2. Finally, in Section 7.2.3, I will show some resulting partial concepts that are derived as building blocks of the instances.

7.2.1 Input Images

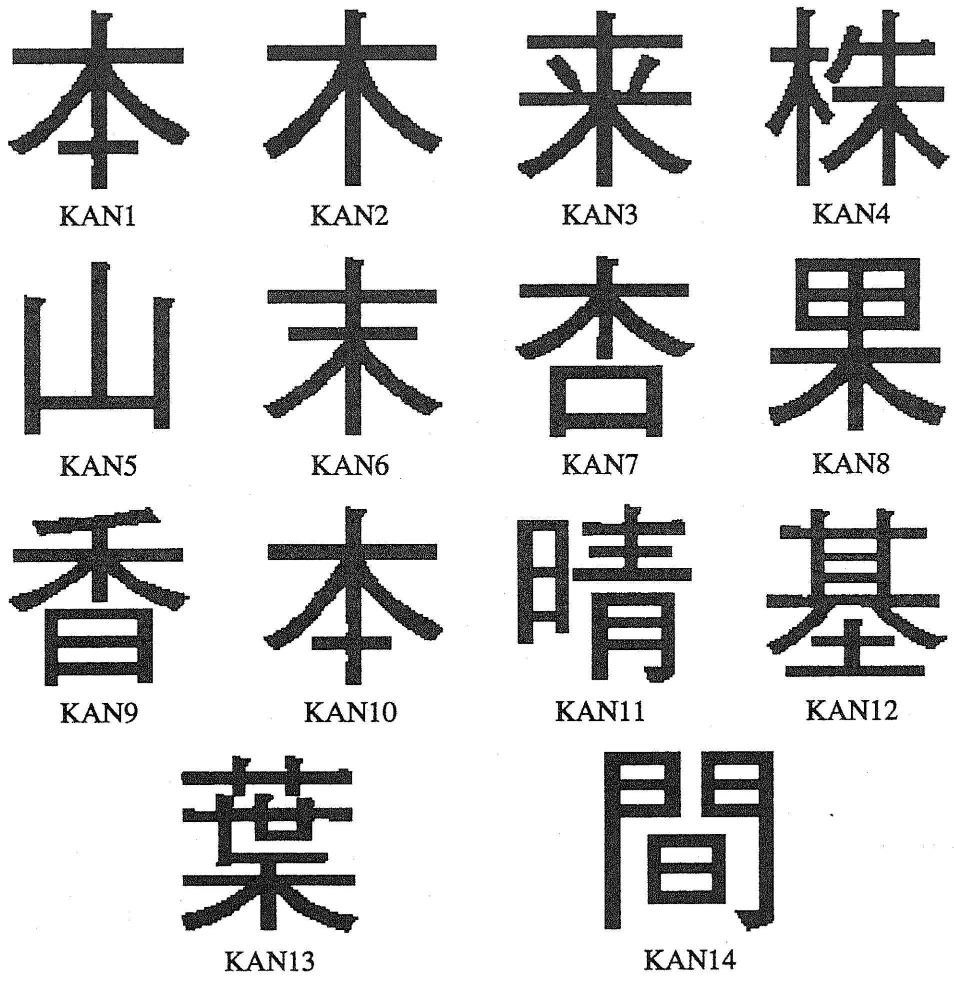


Figure 7.33: Hierarchical clustering: Input shapes

7.2.2 Generalization

The actual ordering of the input sequence is according to Section 7.2.1. In the generalized representation in Fig. 7.34, the instances have been reordered in order to clarify the obtained results. The framed entries refer to the instances, whereas the other entries refer to derived partial concepts. For the sake of simplicity, the relational network layer, together with all connections to it, has been left out.

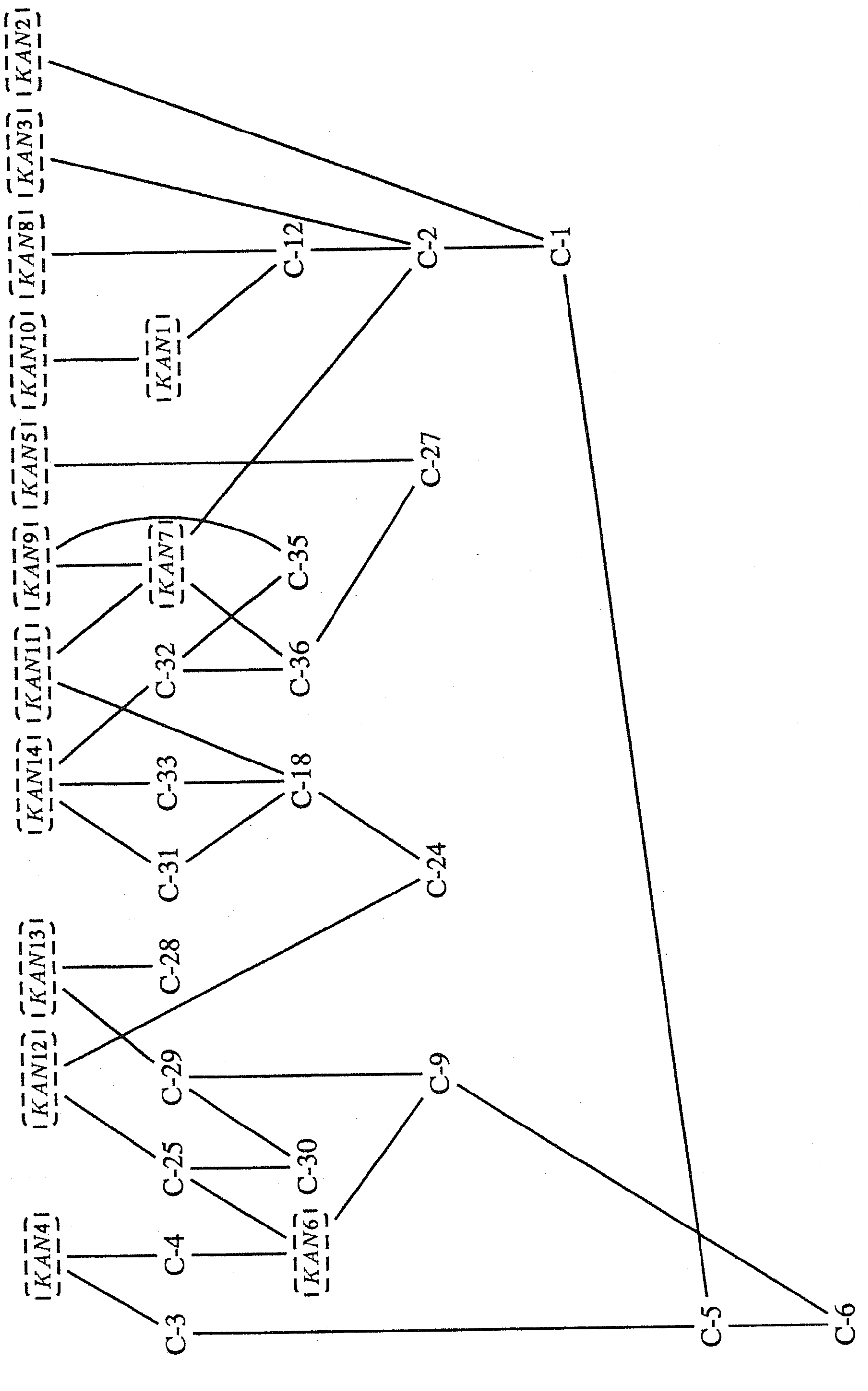


Figure 7.34: Hierarchical clustering: Generalization

7.2.3 Partial Concepts

It is possible to identify a number of terminal nodes, or terminal partial concepts, in the preceding hierarchical generalization. Those nodes can be considered as expressing the basic building blocks of the instances they represent. The terminal partial concepts are shown in Fig. 7.35. As a simplification, those parts of the instances that first corresponded to the partial concepts are shown here. As some subsequent generalization took place, the actual relations are more generally valid than the shown examples imply.

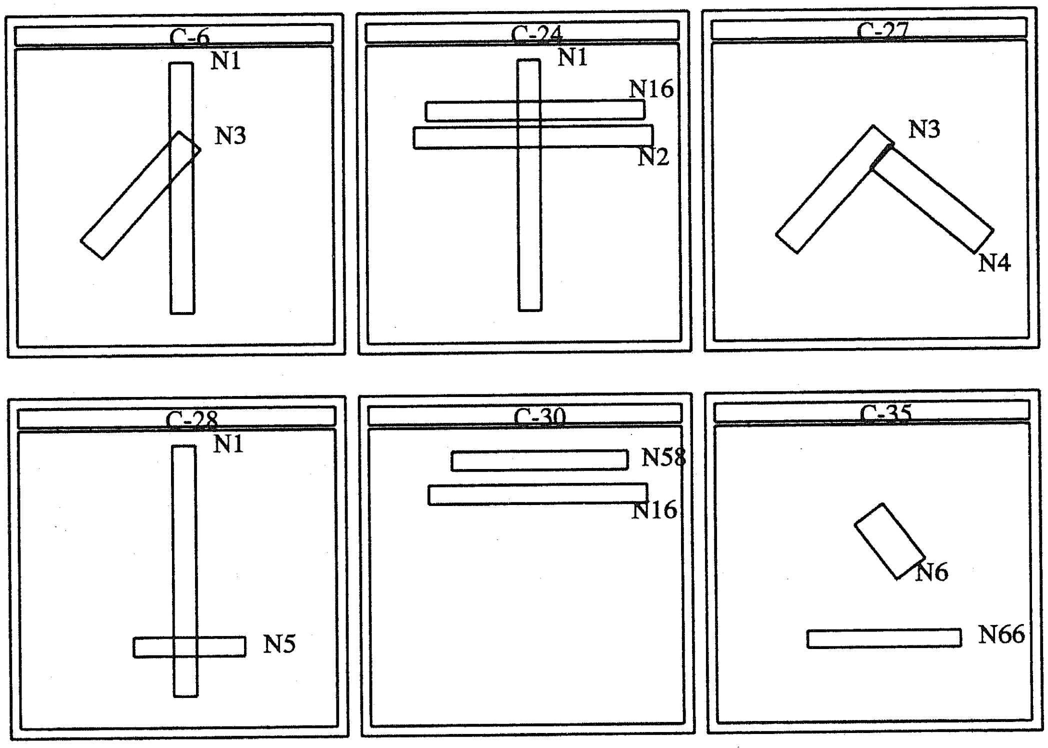


Figure 7.35: Hierarchical clustering: Terminal concepts

The partial concepts in Fig. 7.36 are not terminal nodes of the hierarchical clustering. Nevertheless, they express some important aspects of the represented instances.

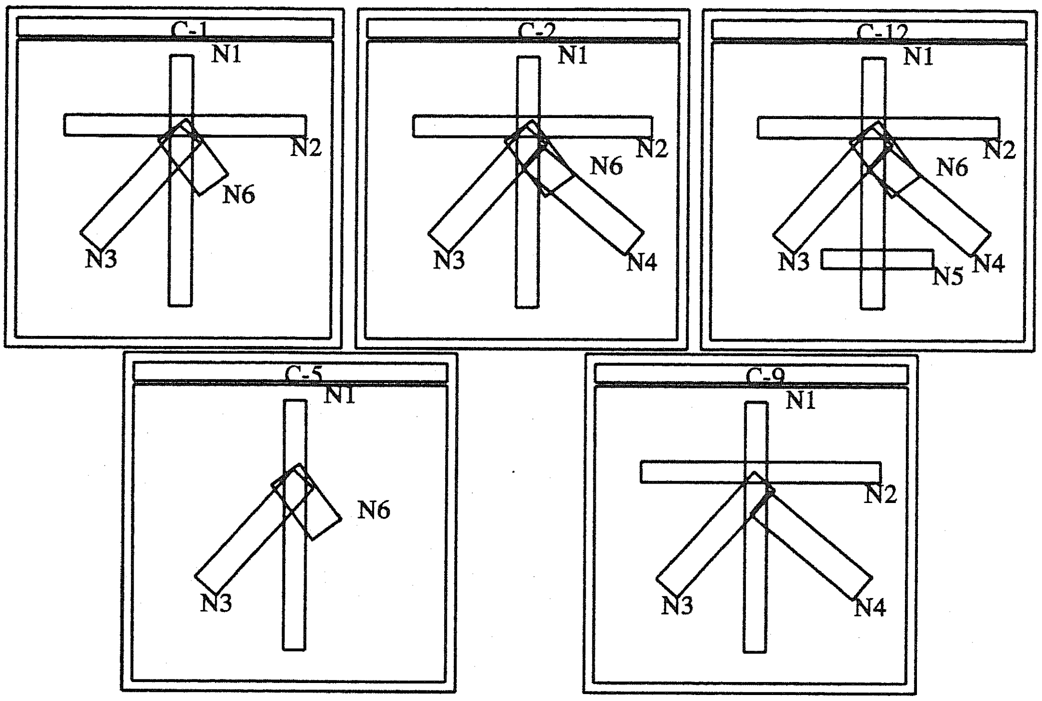


Figure 7.36: Hierarchical clustering: Partial concepts

One of the basic and meaningful building blocks is C-1, which contributes to instances KAN1, KAN2, KAN3, KAN7, KAN8, KAN9, and KAN10. The illustration of C-1 in Fig. 7.36 is actually somewhat misleading, as segment N6 has consistently been merged with longer segments. The real partial concept C-1 would therefore look similar to C-9. It is C-9 which forms the basic building block for concepts KAN4, KAN6, KAN12, and K13. Clearly, partial concepts C-1 and C-6 intuitively represent the same basic structure. The representation of KAN12 by C-9 deserves some more comments. C-9 is found to stand for the upper part of KAN12, a correspondence that is rather counter intuitive. However, the vertical stroke of C-9 was matched to the right vertical stroke of KAN12, leading to this correspondence. A geometrically nice relation can be found between KAN7 and KAN9, the latter being made up of KAN7 and C-35, however, this relation is not intuitive from the ordinary theory of Kanji configuration. The relation between KAN5 and partial concept C-27 can be understood by considering the relational invariance of the employed representation. A problem seems to manifest itself by the correspondence of C-12 to KAN8. As only relations between pairs of segments are considered, segment N5 from partial instance C-12 gets matched to the middle one of the three horizontal segments in the upper part of KAN8. Therefore, although locally the matching is consistent, global consistency is not completely ensured. Finally, instances KAN1 and KAN10 are actually the same character. This can be seen in the fact that KAN1 fully represents KAN10.

7.2.4 Discussion

Hierarchical clustering differs from conceptual clustering in that instances are represented in terms of building blocks. Whereas in conceptual clustering instances are grouped in respect to their similarities with some predefined class, in hierarchical clustering, a grouping is obtained by analyzing each instance in terms of its decomposition. As the main purpose of the present system is the learning of concepts, hierarchical clustering can at most be regarded as a by-product, and it should be viewed in that context. First, hierarchical clustering is not stable under the ordering of the instances. This means, if we shuffle the instances, the resulting hierarchy might not be the same anymore. This instability is closely related to the non-uniqueness of the clustering. Nevertheless, the clustering might be useful for recognizing new instances based on partial characteristics they share with some concept.

Although most of the clusterings in Fig. 7.34 make sense from a geometrical point of view, they are by no means a model for human character learning. Anybody somewhat familiar with Kanji characters could not do much with the clustering in Fig. 7.34. For instance, the instantiation of KAN12 by KAN6 seems to be far fetched. On the other hand, most of the clusterings, not all though, make sense from a geometrical point of view. This is because the system does not rely on any knowledge about the processed shapes, hence, there is nothing but geometry that can guide the clustering.

7.3 Recognition

Recognition as done here is based on the complete world paradigm, as explained in Section 2.1. Complete world paradigm means that we attempt an interpretation of the image or scene, rather than recognition. In any case, interpretation is based on previous knowledge, or, in our case, on previously acquired concepts. For illustrating the aspect of recognition, two different concept data-bases have been used, the resulting examples are shown in Sections 7.3.1 and 7.3.2, respectively.

Recognition is very similar to matching, as has been described in Chapter 4. Therefore, the part of the data-base that matches the instance best, can be viewed as standing for the concept the instance belongs to. If no satisfactory match can be found, then we either explain the instance in terms of partial concepts, that is, concepts that have been instantiated partially only, or the instance is judged to be non-recognizable or to belong to some concept that has not been learned yet. For a complicated image, it is possible to group the interpretations so as to find an image interpretation that is globally optimal. Globally optimal means, that the interpretation is able to explain the biggest number of segments. This interpretation has been termed the most-likely image interpretation.

7.3.1 Tools

The first example uses a simple data-base consisting of four tools only, as shown in Fig. 7.37. The resulting conceptual network for the database is shown in Fig. 7.38. Two experiments are shown in Sections 7.3.1.2 and 7.3.1.3, both consisting of an occlusion scene of different complexity.

7.3.1.1 Data Base

The input shapes, together with the resulting data-base used for recognition is shown below. As there were insufficient similarities found during matching, the relational network consists of three, disjoint parts. The hammer has been matched to the pivot and the upper jaw of TOOL2. This is, admittedly, not a very intuitive correspondence, however, it appears reasonable from a structural point of view.

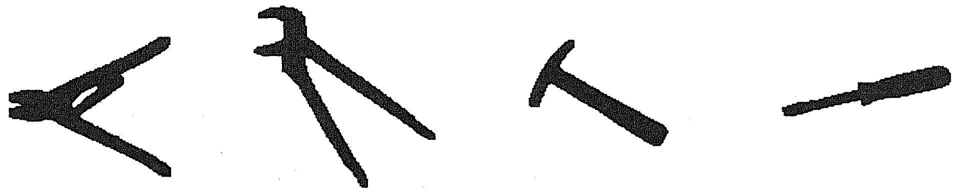


Figure 7.37: Recognition: Input shapes

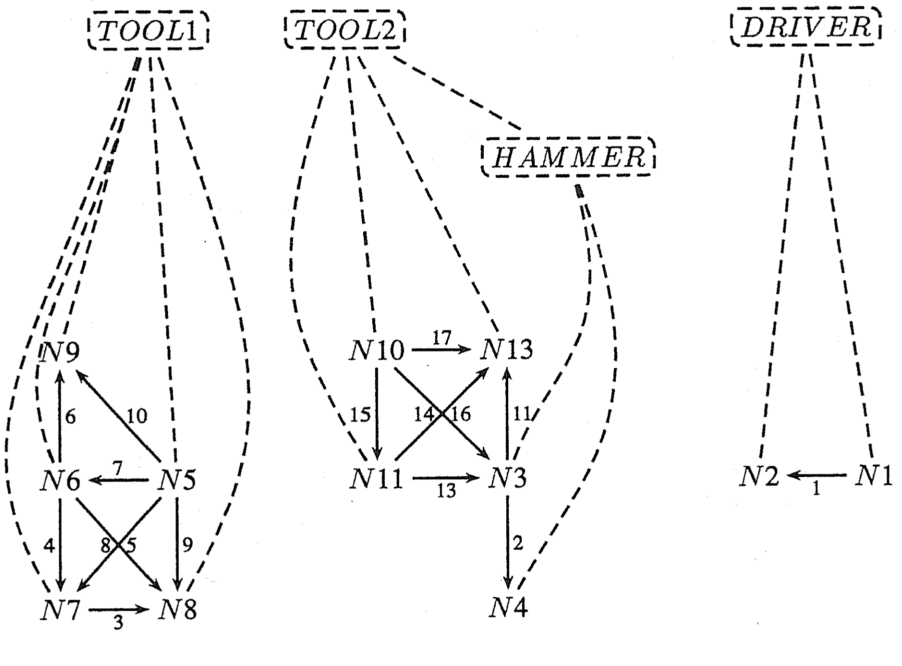
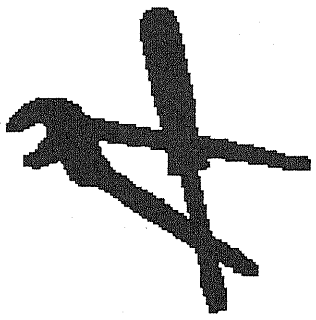
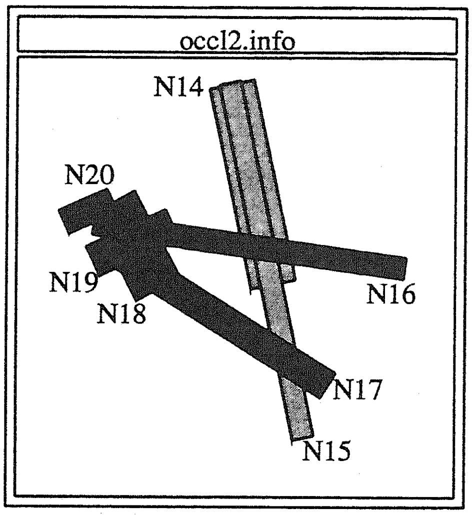


Figure 7.38: Recognition: Generalization

7.3.1.2 Simple Occlusion Scene

In Fig. 7.39, the input shape used is shown on the left, whereas the decomposition, overlaid with the most likely image interpretation, is shown on the right.





Explanations:

▨ DRIVER

■ TOOL2

Figure 7.39: Recognition: Example 1

The actual output of the recognition stage is shown in Fig. 7.40. The most likely interpretation is the one shown above in different shades of gray. Besides this interpretation, different ones are possible as well. However, those interpretations are only locally optimal and do not optimize the interpretation of the scene. Besides the instantiated concept, the nodes of the instance that are instantiated are listed, together with the cost of matching that concept to the instance. The numerical value at the end of each line corresponds to the cost of the matching. Therefore, small values indicate a better match.

```
The following interpretation of the scene is most likely:
    Concept: DRIVER, Inst. nodes: (14 15), Val: 13.085
    Concept: TOOL2, Inst. nodes: (16 17 19 18 20), Val: 12.678

The following interpretations are also possible:

  Fully instantiated concepts:
    Concept: HAMMER, Inst. nodes: (18 20), Val: 13.082

  Partially instantiated concepts:
    Concept: TOOL1, Inst. nodes: (19 15 17), Val: 12.999
```

Figure 7.40: Recognition: Output for example 1

7.3.1.3 Complicated Occlusion Scene

The next example in Fig. 7.41 shows another, more complicated occlusion scene, similar to the one that was used when explaining the performance of the decomposition, in Section 3.4. Although the scene is quite complicated, humans would not have any problems in recognizing the basic tools.

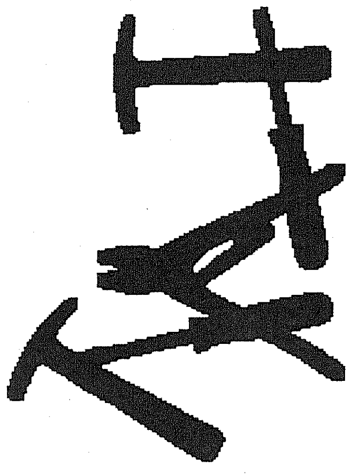
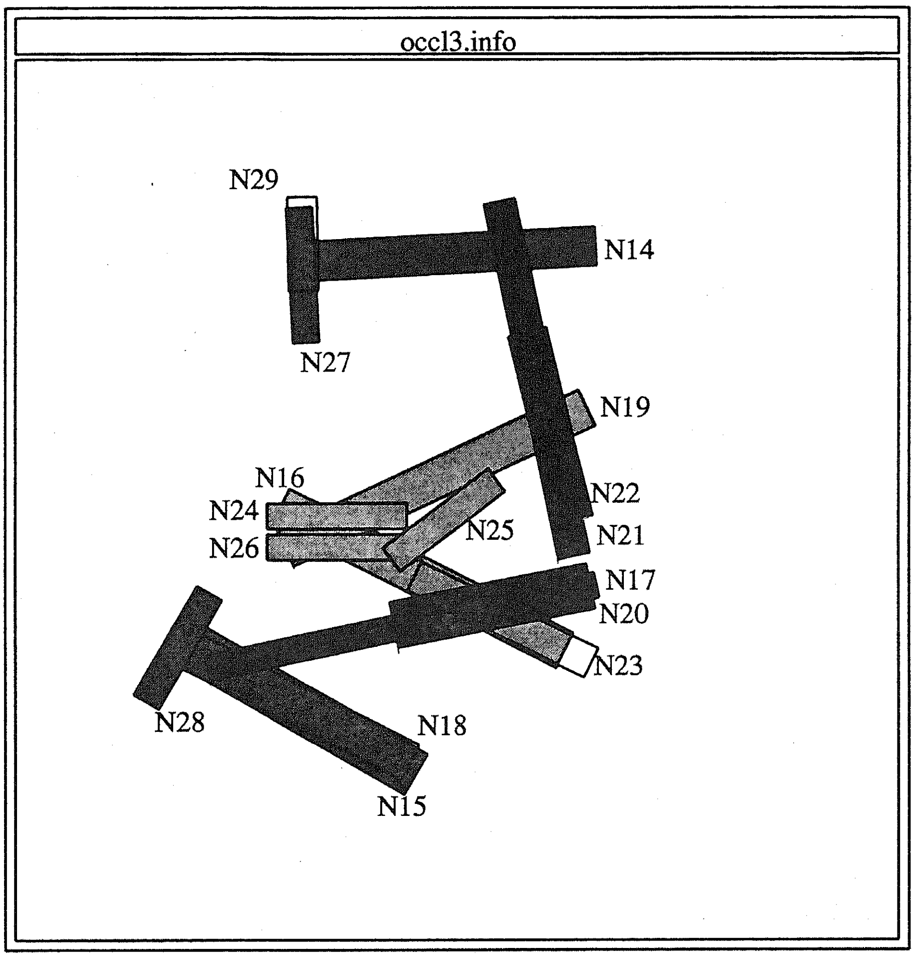


Figure 7.41: Recognition: Example 2

Again, the decomposition has been overlaid with the recognition results in different shades of gray. This is shown in Fig. 7.42.



Explanations:	▨	TOOL1
	■	DRIVER
	■	HAMMER

Figure 7.42: Recognition: Results for example 2

The actual output from which the above description has been derived is given below, in Fig. 7.43. Again, besides the most likely interpretation, several other interpretations have been found. They correspond, for instance, to a second hammer represented by segments

N18 and N28, or, a rather difficult to explain, partial appearance of TOOL2 at nodes N15, N17, N23, and N28.

```
The following interpretation of the scene is most likely:
    Concept: TOOL1, Inst. nodes: (26 24 16 19 25), Val: 27.740
    Concept: HAMMER, Inst. nodes: (14 27), Val: 28.481
    Concept: HAMMER, Inst. nodes: (15 28), Val: 28.484
    Concept: DRIVER, Inst. nodes: (20 17), Val: 28.498
    Concept: DRIVER, Inst. nodes: (22 21), Val: 28.499

The following interpretations are also possible:


  Fully instantiated concepts:
    Concept: HAMMER, Inst. nodes: (18 28), Val: 28.487
    Concept: HAMMER, Inst. nodes: (21 14), Val: 28.494
    Concept: HAMMER, Inst. nodes: (19 22), Val: 28.498
    Concept: HAMMER, Inst. nodes: (14 29), Val: 28.499


  Partially instantiated concepts:
    Concept: TOOL2, Inst. nodes: (23 17 15 28), Val: 28.284
    Concept: TOOL2, Inst. nodes: (23 17 18 28), Val: 28.287
```

Figure 7.43: Recognition: Output for example 2

7.3.2 Kanji characters

The second example uses a data-base of six Japanese Kanji characters, as shown in Fig. 7.44. The characters used for the data-base can be considered ideal. This means, they have been created by hand so as to represent the basic structure of the character, while leaving out all decorations or unneeded details. In addition, the data-base consists of characters that are commonly used as building blocks for other, more complicated characters. The aim of this recognition experiment is, therefore, to recognize any Kanji character in terms of its building blocks.

7.3.2.1 Data Base

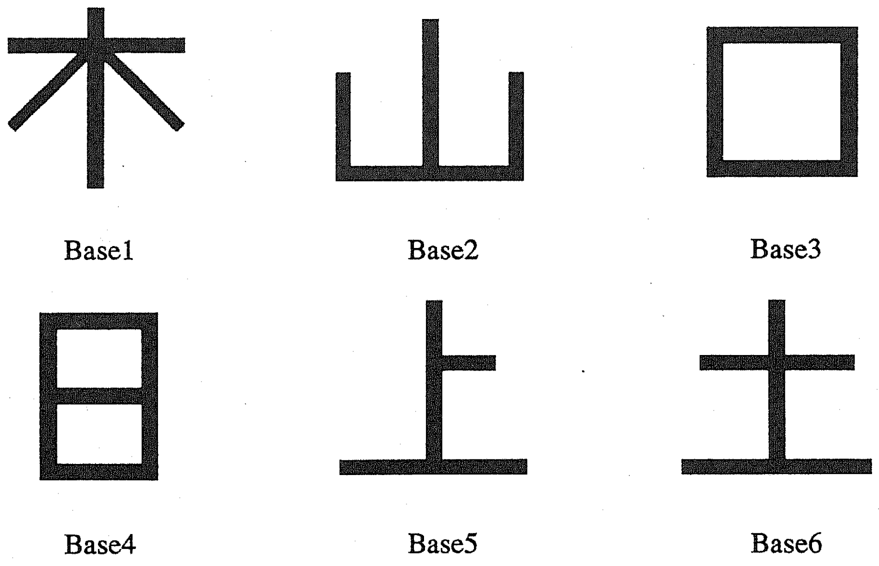


Figure 7.44: Kanji recognition: Input shapes

The resulting conceptual network is shown in Fig. 7.45. Evidently, concept discrimination cannot be done in the relational network anymore, but has to be done in the concept layer.

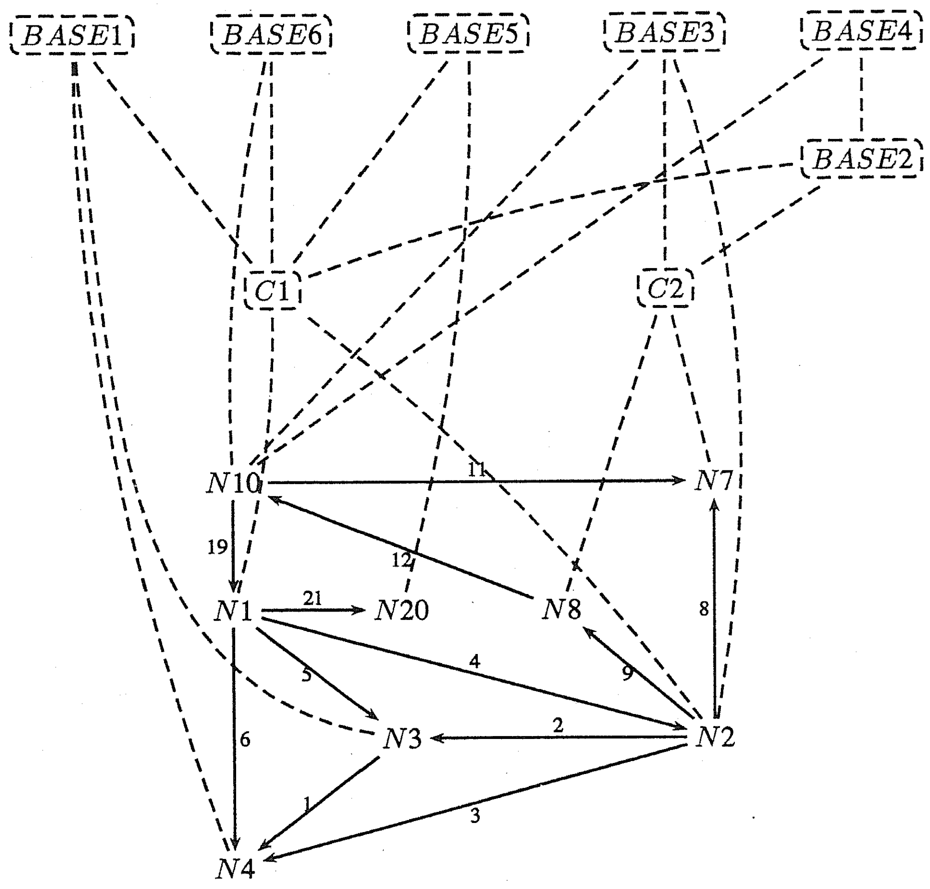


Figure 7.45: Kanji recognition: Conceptual network

7.3.2.2 Recognition

The instances used for recognition were all obtained from a real Gothic font, through scaling and rasterization, and they correspond to some of the instances used in hierarchical clustering. Due to the reality of the font, many small details and especially the slightly curved strokes pose some problems. Especially the curvature of the strokes is a difficult problem for the decomposition stage. The recognition results are shown in Figs. 7.46–7.53.

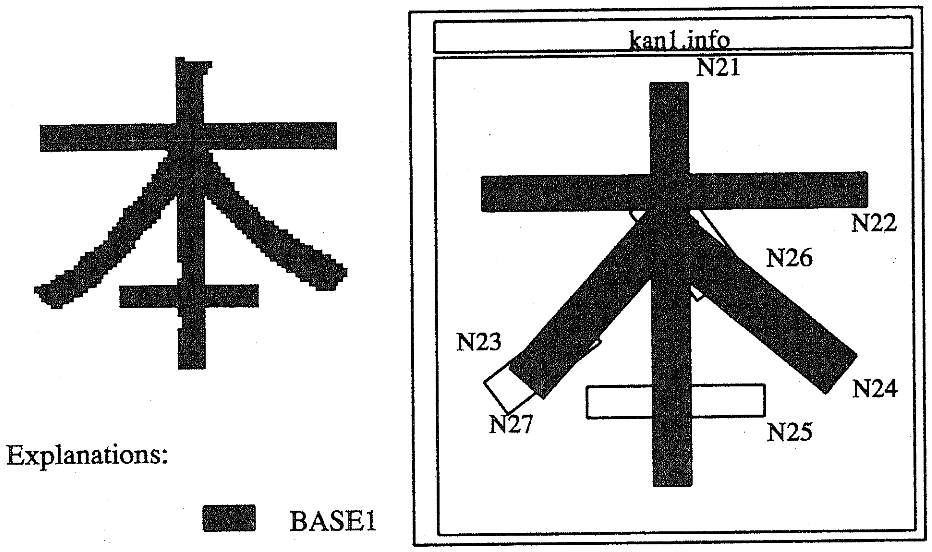


Figure 7.46: Kanji recognition: Example 1

For the input shown in Fig. 7.46, very nicely, an instantiation of BASE1 has been found. This instantiation, together with an analysis of the remaining segments, should lead to a complete recognition of the instance.

```
The following interpretation of the scene is most likely:
    Concept: BASE1, Inst. nodes: (23 24 21 22), Val: 12.408

The following interpretations are also possible:

  Fully instantiated concepts:
    Concept: BASE5, Inst. nodes: (25 21 22), Val: 12.807

  Partially instantiated concepts:
    Concept: BASE1, Inst. nodes: (26 24 21 22), Val: 12.524
    Concept: BASE4, Inst. nodes: (26 23 24), Val: 12.962
```

Figure 7.47: Kanji recognition: Output for example 1

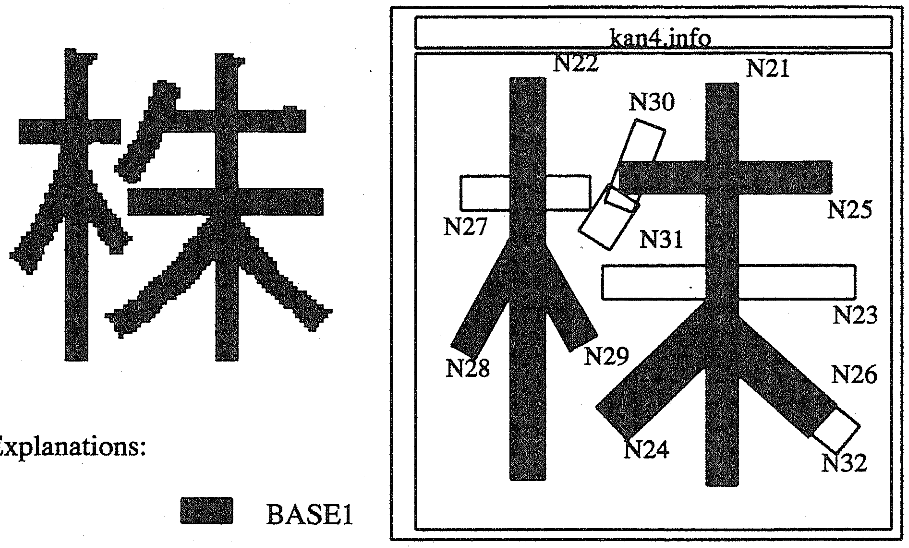


Figure 7.48: Kanji recognition: Example 2

In this case, Fig. 7.48, two instances of BASE1 have been found, although one only partially. Looking at the data-base makes clear that the instantiation of segment N27 is quite difficult, it is simply too short. Maybe it would be of advantage to consider the left part as an altogether different concept.

```
The following interpretation of the scene is most likely:
    Concept: BASE1, Inst. nodes: (26 24 25 21), Val: 11.547
    Concept: BASE1, Inst. nodes: (28 22 29), Val: 4.200

The following interpretations are also possible:

  Partially instantiated concepts:
    Concept: BASE1, Inst. nodes: (26 24 25 21), Val: 11.547
    Concept: BASE1, Inst. nodes: (24 23 21 26), Val: 11.684
    Concept: BASE4, Inst. nodes: (25 21 23), Val: 11.883
```

Figure 7.49: Kanji recognition: Output for example 2

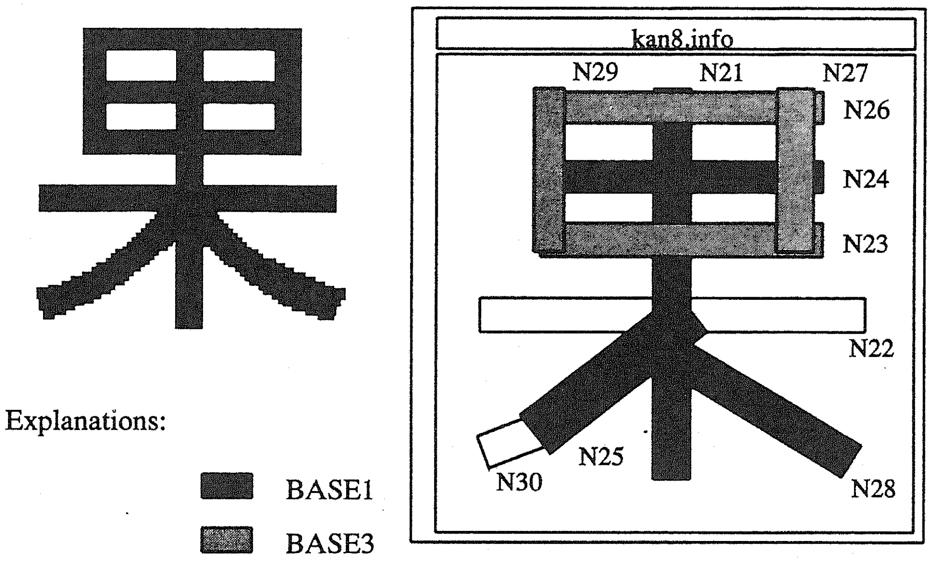


Figure 7.50: Kanji recognition: Example 3

While the two instantiations in Fig. 7.50 are correct, it seems that a different interpretation in terms of BASE1 and BASE4 would be more intuitive. However, again this was not possible because segment N22 is too long to give a viable alternative to segment N24 for instantiating BASE1. As can be seen from Fig. 7.51 segment N22 does not appear in any instantiation of BASE1, it has therefore been pruned during the match stage.

```
The following interpretation of the scene is most likely:
    Concept: BASE1, Inst. nodes: (28 25 24 21), Val: 17.166
    Concept: BASE3, Inst. nodes: (23 27 26 29), Val: 17.071

The following interpretations are also possible:

  Fully instantiated concepts:
    Concept: BASE4, Inst. nodes: (27 23 21 26 29), Val: 16.791
    Concept: BASE6, Inst. nodes: (27 24 21), Val: 17.279
    Concept: BASE6, Inst. nodes: (22 21 24), Val: 17.313

  Partially instantiated concepts:
    Concept: BASE4, Inst. nodes: (27 24 21 26 29), Val: 17.009
    Concept: BASE4, Inst. nodes: (26 23 29 24 21), Val: 17.069
    Concept: BASE4, Inst. nodes: (23 26 27 24 21), Val: 17.074
    Concept: BASE1, Inst. nodes: (28 25 26 21), Val: 17.205
```

Figure 7.51: Kanji recognition: Output for example 3

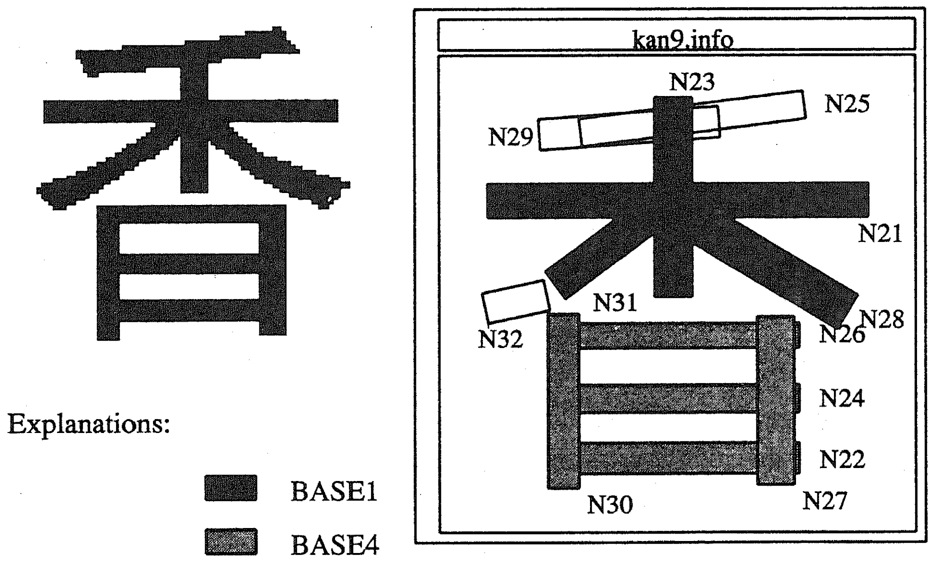


Figure 7.52: Kanji recognition: Example 4

The final example in Fig. 7.48 is maybe the most successful one. Both, BASE1 and BASE4 have been identified correctly, and the remaining segments could either be treated as noise, or they could be used for further analysis of the instance.

```
The following interpretation of the scene is most likely:
    Concept: BASE1, Inst. nodes: (23 28 21 31), Val: 10.689
    Concept: BASE4, Inst. nodes: (27 26 22 24 30), Val: 5.919

The following interpretations are also possible:


  Fully instantiated concepts:
    Concept: BASE6, Inst. nodes: (21 23 25), Val: 10.715


  Partially instantiated concepts:
    Concept: BASE2, Inst. nodes: (29 25 23), Val: 10.739
    Concept: BASE1, Inst. nodes: (31 25 23), Val: 10.758
    Concept: BASE4, Inst. nodes: (21 29 23), Val: 10.789
```

Figure 7.53: Kanji recognition: Output for example 4

7.3.3 Discussion

Although the used examples are rather simple, I think they outline the possibilities for recognition with the present system. As in the case of hierarchical clustering, recognition was not the main purpose for the design of the system. Rather, because of its similarity with the implemented matching scheme, straightforward recognitions become possible. One major problem with the examples shown is that the used concept data-base is too small. Therefore, an accurate implementation of the complete world paradigm was not really possible. An interpretation of instances that have not actual relation to the used concept data-base is, therefore, meaningless. Even though, the obtained results are useful, as in practical applications a system would never be confronted with objects from many different domains.

Obviously, the presented system does not represent an alternative to dedicated schemes for Kanji recognition. However, it should be kept in mind that the system was developed with the aim of concept learning. The above examples should therefore be taken as an illustration of the general purpose performance of the presented system.

CHAPTER 8

Conclusions

In this thesis I proposed a system that learns descriptions and concepts from visual input. Visual input has been restricted to binary images, a strong limitation in some sense, but still reasonable if we consider how many objects can be sufficiently characterized by their silhouettes. Binary images are first decomposed into MACS, that is, maximal approximately convex subparts, whose interrelation is then extracted and described. Interrelations among MACS are translated into relational networks, whose basic building blocks are generalized predicates. Whereas the structure of the relational network mirrors the topological or qualitative aspect of the input object, the generalized predicates mirror the quantitative aspects. The actual learning consists of the generalization of two or more relational networks into a conceptual network, where the conceptual network can be understood as giving a class description, covering all the acquired instances. Finally, based on the obtained conceptual network, a very simple explanation in natural language becomes available. This explanation serves a twofold purpose: first, it allows for a better evaluation of the obtained results, and second, the gap between man and machine can be partially bridged.

Lately, research on inductive learning has concentrated on so-called *constructive induction systems* [36], which include mechanisms for generating new, more relevant descriptors, and for removing the less relevant descriptors. In comparison, *selective induction systems* are utilizing a fixed, a priori defined representation space. The present work does not fully fit into either of the two categories. Basically, it is a selective induction approach, as the representation space, that is the set of possible generalized predicates is fixed. On the other hand, the extraction and construction of partial concepts during generalization is characteristic of constructive induction.

Since the present system is analyzing its input data, especially the interrelation among features or MACS in our case, according to Wnek and Michalski [62], it can be further characterized as being a data-driven system that learns concepts from examples (*concept acquisition*). This means, based on specific instances of some concept, the system attempts to induce general descriptions for the concepts in question. The instances, or observations, are characterizations of some objects, whereas the induced concept description, or hypothesis, can be viewed as a concept recognition rule. In other words, if an object satisfies the concept description, then it represents the given concept. Since no negative instances are being used for deriving the concept description, we can characterize the present system as one learning *characteristic descriptions*, rather than a system that learns *discriminant descriptions*. Discrimination power derives from the acquisition of multiple concepts, which will then act as negative concept descriptions. The major danger in learning from positive instances only, as opposed to Winston's system utilizing *near-misses*, is that there is no limit to which a description can be generalized. This problem was dealt with in the present work by introducing the monotonically decreasing weight ϵ in Eq. 5.2. That weight will cause the generalization of generalized predicates to approach a stable state after a certain number of generalization steps.

Another important characterization of data-driven inductive learning systems is based on Mitchell's *version space* [10]. In version space, concepts are partially ordered according to their generality. In the present system, however, no such partial ordering can be found, thus making version space not applicable. Why this is so can again be understood by looking at Eq. 5.2. Clearly, the generalization of two generalized predicates does not completely subsume the two generalized predicates, therefore, no partial ordering can be defined in general.

One weakness of the proposed system is that no set of plausible hypotheses, that is, candidate concepts, is maintained, and we are restricted to updating a single hypothesis. The advantages of this approach are considerable savings in memory and processing time. On the other hand, it is not possible to develop hypotheses simultaneously and to see which one finally represents the concept best. Having only one hypothesis corresponds to having a well structured, one-dimensional hypothesis space, or in other words, to having a strong bias. As has been pointed out by Rendell [45] having strong bias allows for fast learning, while it might exclude some possible hypotheses. Another problem arises when the used bias is not appropriate. This problem has not been addressed in the present work, and it arises when the used features are not appropriate for the concept acquisition. In such a case, the need for constructive induction becomes evident.

Several contributions made in the preceding chapters deserve some more emphasis. The most important three are approximate convexity, the decomposition into MACS, and the notion of generalized predicates. Each of those contributions can be considered in isolation and they could be used in other systems.

1. The idea of approximate convexity, as outlined in Section 3.3, relies on a definition that is very intuitive. Although other definitions, leading to different measures of approximate convexity, might be possible, similar to changing the axioms of Euclidean geometry, which leads to non-Euclidean geometries, the derivations are mathematically sound and general. As opposed to other measures, the proposed measure of approximate convexity yields simple estimators for the important cases of two and three-dimensional spaces. Although the actual estimation in spaces of higher dimensionality has not been treated here, the results presented reduce this question to the measuring of hypervolumes and hypersurfaces.

2. Heavily relying on the results of approximate convexity in two-dimensions is the decomposition of binary objects into MACS, as explained in Section 3.4. The results of that decomposition are meaningful in many cases, as has been illustrated with some examples. The main problem at the moment, however, appears to lie in the

implemented skeletonization approaches. As has been shown in Chapter 6, depending on the class of input shapes, a different skeletonization scheme might be called for. As a thorough comparison by Haralick et al. [21] showed, a skeletonization approach by Ogniewicz [39] is among the best in respect to resistance towards noise and rotational invariance. An implementation of that approach might improve the stability of the decomposition. Furthermore, an extension of the approach for three-dimensional objects should be possible.

3. The idea that is the centerpiece of the proposed learning scheme is the generalized predicate. Generalized predicates allow for quantitative generalizations in an intuitive way, unlike, for instance, the graded predicates proposed by Gang et al. [14]. Based on the idea of generalized predicates, relational networks and conceptual networks were introduced, the former representing the structure of an instance, whereas the latter can be understood as a generalization of semantic networks. Although these two network structures have not been developed fully yet, further extensions based on generalized predicates appear possible.

Future work based on the ideas presented here, seems possible and necessary. The most urgent extension might be to enlarge the used domain. Although working with binary images showed some possibilities of generalized predicates and conceptual networks, only if those notions are applied to real world objects, can their validity be verified. Real world objects might mean three-dimensional objects, recovered from range-data or from gray-scale or colour images. Even at present, there are some remaining problems that ought to be addressed.

1. Both, matching and generalization are based on an optimization. Although optimization is an approach that appears close to human problem solving, the problem of how to distinguish between a good optimization and a bad optimization has not been solved. In other words, how do we recognize an instance or a part of an instance that has no counterpart in the existing concept. At present, the question has been solved

by introducing a threshold for avoiding infeasible matches. Although this appears to work for the shown examples, it cannot really be considered a general solution.

2. The employed approach of representing only relations among segments of the decomposition that at least partially overlap cannot be considered sufficient in general. Often, the constraints available are not really enough for guiding the matching. For instance, the matching of the engines of the various planes shown in Chapter 7, is only restricted by their relations to the main-wings. The relations of the engines to the fuselage or to the other engines would add constraints that could be used for a more meaningful matching. On the other hand, introducing a complete graph structure for the representation of an instance would severely impede the performance of the system. What relations to choose, and whether those relations should be grouped hierarchically, are some more questions that should be considered in further detail.

3. Another problem I would like to mention here is how to find a better generalization function for generalized predicates. Although the function proposed in Section 5.2 is intuitive and has been used in previous research, certain points deserve some more attention. For instance, it is a valid assumption that the generalization of instances with widely varying generalized predicates should spread out the resulting generalized predicate. However, even if we generalize the same predicate over and over, the result will be a generalized predicate with increased standard deviation. Intuitively, the standard deviation should decrease in such a case, mirroring the growing confidence in the repeatedly found generalized predicate. How to incorporate this, together with keeping the definition of generalized predicates closed under the generalization operator is another open question.

4. Finally, a more complete system would be possible by incorporating some sources of domain knowledge. If this domain knowledge is available in the form of a conceptual network, then it could be directly used for matching and generalization. To incorporate domain knowledge in the decomposition, however, some changes to

the present system would be necessary. For instance, at the moment there are no possibilities of driving the decomposition based on some given constraints. To allow for such a guided decomposition, some feedback links, prompting the decomposition towards the right direction while penalizing deviations from it, would be necessary. This point as well might be a worthwhile extension of the present system.

Although the above list is by no means complete, it mentions the most urgent problems encountered with the system. Addressing some of them in a future system might lead to a more versatile and generally useful system than the one described here.

APPENDIX A

Convex Hull

The necessary size of the discrete structuring element used for the computation of the convex hull, together with the maximal depth of non-detectable concavities, can easily be calculated by using the geometrical structure of the structuring element. First, let us assume that the structuring element is a regular polygon with n vertices, designed to approximate the disk as closely as possible. If we connect each vertex of the structuring element with the origin O, i.e., the centre of the polygon, then we can define a set of n lines, let us call them $l^{(k)}$, $k = 1, \ldots, n$, normal to the lines emanating from the centre and each passing through one vertex of the polygon. Now, we can consider the projection of the set X, of which we want to obtain the convex hull, on the kth line, namely $X \mid l^{(k)}$. Clearly the length of this projection, that is, $L(X \mid l^{(k)})$, specifies the maximal dimension of the set X in direction of the line $l^{(k)}$. The maximum dimension of the set X is therefore given as the maximum of the length of all projections on the lines $l^{(k)}$, i.e.,

$$\max_k \left(L(X \mid l^{(k)}) \right).$$

Now, we can again return to the structuring element and look at its properties as a regular polygon. We can choose three consecutive vertices a_{k-1}, a_k, a_{k+1}, such that the projection of X on the normal through a_k is maximal for all possible projections. If we consider $L(X \mid l^{(k)})$ as fixed, then the maximal possible width of a concavity is $d(a_{k-1}, a_{k+1})$

(imagine a crescent-like figure as shown in Fig. A.1), where $d(a_{k-1}, a_{k+1})$ denotes the Euclidean distance between points a_{k-1} and a_{k+1}. Thus, in order to fill the concavity by a closing operation, we need

$$d(a_{k-1}, a_{k+1}) \geq \max_k \left(L(X \mid l^{(k)})\right), \tag{A.1}$$

as otherwise a part of the concavity reappears during the erosion stage of the closing. Further, if we consider the triangle a_k, O, a_{k+1} then the centre angle is given as $2\pi/n$. Therefore, as $d(O, a_k) = d(O, a_{k+1}) = r$, we obtain

$$\begin{aligned} \max_k \left(L(X \mid l^{(k)})\right) &\leq 2r \sin\left(\frac{2\pi}{n}\right) \\ \Rightarrow r &\geq \frac{\max_k \left(L(X \mid l^{(k)})\right)}{2\sin(2\pi/n)}. \end{aligned} \tag{A.2}$$

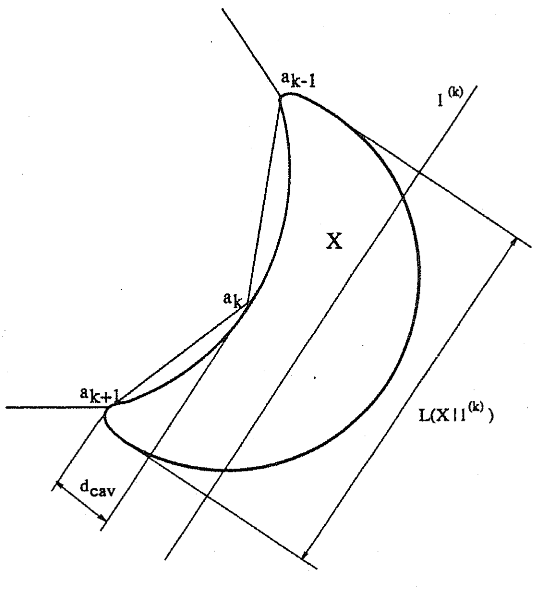


Figure A.1: Maximal possible concavity

The depth d_{cav} of the maximal non-detectable concavity is given as the height in the triangle a_{k-1}, a_k, a_{k+1}. Therefore,

$$d_{cav} \leq \frac{\max_k \left(L(X \mid l^{(k)})\right)}{2\tan\left(\pi\left(\frac{n-2}{2n}\right)\right)}. \tag{A.3}$$

APPENDIX B

Approximate Convexity

Here, I will try to give a very simplified derivation of the equations used in Section 3.3. For a more thorough treatment, the reader is referred to the books by Hadwiger [20] and Santaló [47].

Let us denote by $x = (x_1, x_2, \ldots, x_n)$ the coordinates of an n-dimensional space. The measure $m(X)$ of a set of points X is defined as the integral over the set of a differential form $\omega = f(x_1, x_2, \ldots, x_n) dx_1 \wedge dx_2 \wedge \ldots \wedge dx_n$ (provided that integral exists in the Lebesgue sense), where the function $f(x_1, x_2, \ldots, x_n)$ is chosen so that the measure $m(X)$ is invariant under the group of motions in n-space.

A motion in Euclidean n-space is an affine transformation of E^n onto itself such that distances are preserved. The motions form a subgroup $\mathscr{M}$ of the group of affine transformations, defined by

$$x' = Ax + B \ , \quad AA^t = I, \tag{B.1}$$

where I denotes the identity matrix of dimension n, and, therefore, A is an orthogonal matrix.

Clearly, $A = (a_{ij})$ and $B = (b_i)$ are $n \times n$ and $n \times 1$ matrices respectively. Hence, if we set $A = (a_{ij})$, $B = (b_i)$, $A^{-1} = (\alpha_{ij})$, then we obtain the covectors ω_{ij} and ω_i:

$$\omega_{ij} = \sum_{h=1}^{n} \alpha_{ih} da_{hj}$$

$$\omega_i = \sum_{h=1}^{n} \alpha_{ih} db_n. \tag{B.2}$$

Consider a moving frame $(p; e_1, \ldots, e_n)$ composed of a point p and n independent unit vectors e_i. Using the relations $e_i \cdot e_j = \delta_{ij}$, $de_i \cdot e_j + e_i \cdot de_j = 0$, we have

$$\begin{aligned} \omega_i &= dp \cdot e_i \\ \omega_{ji} &= de_i \cdot e_j \\ \omega_{ji} + \omega_{ij} &= 0. \end{aligned} \tag{B.3}$$

Now, we want to define a density for r-dimensional planes L^r in E^n, invariant under the group of motions. If we denote the group of motions in E^n by $\mathscr{M}$, we can further denote by $\mathscr{H}^r$ the group of all motions that leave a fixed r-dimensional plane L_0^r invariant. Clearly, there is a one-to-one correspondence between the r-planes of E^n and the elements of the homogeneous space $\mathscr{M} \setminus \mathscr{H}^r$, which is the set of left cosets $g\mathscr{H}^r$, $g \in \mathscr{M}$. This means that each plane L^r can be obtained from L_0^r by some motion. Therefore, the problem of finding an invariant density for sets of r-planes is equivalent to that of finding an invariant density on $\mathscr{M} \setminus \mathscr{H}^r$.

It can be proven (see [47] for details) that

$$d(\mathscr{M} \setminus \mathscr{H}^k) = \omega_1 \wedge \ldots \wedge \omega_k, \tag{B.4}$$

is invariant under $\mathscr{M}$, and up to a constant factor, is the unique form with this property. In terms of our r-plane in E^n, the invariant density for $\mathscr{M} \setminus \mathscr{H}^r$ is therefore given by

$$dL^r = \bigwedge_{i,\beta} \omega_{i\beta} \bigwedge_{\alpha} \omega_\alpha, \tag{B.5}$$

with $\alpha, \beta = r+1, \ldots, n; i = 1, \ldots, r$.

Let us next consider a q-dimensional plane $L^q{}_0$, fixed in E^n. Now, we are looking for the density of r-planes L^r $(r > q)$ that contain $L^q{}_0$. With similar reasoning as above, the density for r-planes about $L^q{}_0$ becomes

$$dL^{r[q]} = \bigwedge_{h,i} \omega_{hi} \quad (h = r+1, \ldots, n;\ i = q+1, \ldots, r). \tag{B.6}$$

Let us next try to compute the measure of all r-planes L^r about $L^q{}_0$. To this aim, we first consider the case where $q = 0$, that is, the group of rotations about the origin p of the moving frame $(p; e_i)$. By rotation about p, the end points of the vectors e_i move on the unit sphere U_{n-1} centered at p. The scalar product $de_i \cdot e_j = \omega_{ji}$ is equal to the arc element on U_{n-1} at the endpoint of e_i in the direction of e_j, so that, denoting by du_{n-1} the $(n-1)$-dimensional volume element of U_{n-1} at the end point of e_1, we have

$$du_{n-1} = \omega_{21} \wedge \ldots \wedge \omega_{n1}. \tag{B.7}$$

Therefore

$$du_{n-r} = \omega_{r+1,r} \wedge \ldots \wedge \omega_{n,r}. \tag{B.8}$$

If du_{r-1} denotes the volume element of the unit $(r-1)$-dimensional sphere in $L^{r[0]}$, then after some transformation, we obtain

$$dL^{r[0]} \wedge du_{r-1} = dL^{r-1[0]}_{(n-1)} \wedge du_{n-1}. \tag{B.9}$$

By successive exterior multiplications and by integrating we get the total measure of r-planes of E^n through a fixed point

$$\int dL^{r[0]} \wedge du_{r-1} \wedge \ldots \wedge du_1 = \int du_{n-1} \wedge \ldots \wedge du_{n-r}. \tag{B.10}$$

Hence,

$$m(G_{r,n-r}) = m(G_{n-r,r}) = \int_{G_{r,n-r}} dL^{r[0]} = \frac{O_{n-1}O_{n-2}\cdots O_{n-r}}{O_{r-1}O_{r-2}\cdots O_1 O_0}, \tag{B.11}$$

where O_i is the surface area of the n-dimensional unit sphere[1].

We have seen in Eq. B.5, that dL^r is given in terms of a point p and r orthonormal vectors e_i contained in L^r. If we set $d\sigma_{n-r} = \omega_{r+1} \wedge \omega_{r+2} \wedge \ldots \wedge \omega_n$, then we get

$$dL^r = d\sigma_{n-r} \wedge dL^{r[0]} \tag{B.12}$$

such that $d\sigma_{n-r}$ is the volume element of $L^{n-r[0]}$ at p, as $\omega_{r+h} = dp \cdot e_{r+h}$ is the arc element of $L^{n-r[0]}$ at p in direction e_{r+h}.

[1]The surface area of the n-dimensional unit sphere is given as $O_n = \frac{2\pi^{(n+1)/2}}{\Gamma(\frac{n+1}{2})}$.

Let K be a convex set and let O be a fixed point in E^n. We consider all the $(n-r)$-dimensional planes $L^{n-r[O]}$ through O and we let K'^{n-r} be the orthogonal projection of K onto $L^{n-r[o]}$. In other words, K'^{n-r} denotes the convex set of all intersection points of $L^{n-r[o]}$ with the r-planes L^r perpendicular to $L^{n-r[o]}$ through each point of K. Now, the mean value of the projected volumes $V(K'^{n-r})$ becomes

$$E(V(K'^{n-r})) = \frac{\int_{G_{n-r,r}} V(K'^{n-r}) dL^{r[O]}}{m(G_{n-r,r})}. \tag{B.13}$$

The relation

$$I_r(K) = \int_{G_{n-r,r}} V(K'^{n-r}) dL^{r[O]}, \tag{B.14}$$

is an important characteristic in integral geometry. $I_r(K)$ can be expressed in terms of the quermassintegrals $W_r(K)$ as introduced by Minkowski. We have

$$W_r(K) = \frac{(n-r)O_{r-1}\cdots O_0}{nO_{n-2}\cdots O_{n-r-1}} I_r(K). \tag{B.15}$$

In general, the following equalities hold for quermassintegrals

$$\begin{aligned} W_0 &= V \\ nW_1 &= F \\ nW_n &= O_{n-1}, \end{aligned} \tag{B.16}$$

where V is the volume, F is the surface, and O_{n-1} is the surface of the $n-1$-dimensional unit ball.

Integrating Eq. B.12 and substituting Eq. B.15 gives for the measure of the r-planes

$$\begin{aligned} m(L^r, L^r \cap K \neq \emptyset) &= \int_{L^r \cap K \neq \emptyset} dL^r = \int d\sigma^{n-r} \wedge dL^{r[0]} \\ &= \frac{nO_{n-2}\cdots O_{n-r-1}}{(n-r)O_{r-1}\cdots O_0} W_r(K). \end{aligned} \tag{B.17}$$

Let K^q be a convex body in the q-plane $L^q \subset E^n$. Further, let $W_r^{(q)}(K^q)$, $(r = 0, 1, \ldots, q)$ be the quermassintegrals of K^q as a convex body of L^q. If we consider K^q as a flattened convex body of E^n $(n > q)$, the quermassintegrals $W_r^{(n)}(K^q)$ can be evaluated by considering first the quermassintegrals of the parallel convex body[2] ${K_\rho}^q$ and then taking the limit as $\rho \to 0$. The actual derivation and proof has been done by Hadwiger [20], here we will only give the results. As Hadwiger showed, if we assume that $q = n - 1$, then we have

$$W_r^{(n)}(K^{n-1}) = \frac{r}{n}\frac{O_{r+1}}{O_r} W_{r-1}^{(n-1)}(K^{n-1}). \tag{B.18}$$

Now, we can recursively apply Eq. B.18 to obtain the quermassintegrals for any convex body K^q of dimensionality q, such that $q < n$:

$$W_r^{(n)}(K^q) = \frac{\binom{q}{r-n+q}}{\binom{n}{r}} \frac{O_{r+1}}{O_{r-n+q+1}} W_{r-n+q}^{(q)}(K^q). \tag{B.19}$$

[2]The parallel body K_ρ in the distance ρ of a convex set K is the union of all solid spheres of radius ρ, the centers of which are points of K.

APPENDIX C

Approximation of the Product of two Gaussians

A scaled Gaussian normal distribution is, in its general form, given as

$$G(x) = \frac{a}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2}\frac{(x-\mu_0)^2}{\sigma^2}}. \tag{C.1}$$

When multiplying two scaled Gaussians, we face the problem of approximating the result by a single of scaled Gaussians. That is,

$$G_1(x) \cdot G_2(x) \approx G_3(x).$$

Writing out the above equation gives

$$\frac{a_1 a_2}{2\pi\sigma_1\sigma_2} e^{\frac{-1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{-1}{2}\frac{(x-\mu_2)^2}{\sigma_2^2}} \approx \frac{a_3}{\sqrt{2\pi}\sigma_3} e^{\frac{-1}{2}\frac{(x-\mu_3)^2}{\sigma_3^2}}. \tag{C.2}$$

The purpose of this appendix is to give an estimate for the three parameters μ_3, a_3, σ_3. We approach the problem by first noting that by setting the derivative of $G_1(x) \cdot G_2(x)$ to zero and solving for x, we can obtain μ_3.

$$\frac{d}{dx}\frac{a_1 a_2}{2\pi\sigma_1\sigma_2} e^{-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{1}{2}\frac{(x-\mu_2)^2}{\sigma_2^2}} = \frac{a_1 a_2}{2\pi\sigma_1\sigma_2} e^{\frac{-1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{-1}{2}\frac{(x-\mu_2)^2}{\sigma_2^2}} \left(-\frac{x-\mu_1}{\sigma_1^2} - \frac{x-\mu_2}{\sigma_2^2}\right) = 0.$$

The solutions where either a_1, a_2, σ_1, or σ_2 are zero can be discarded as meaningless. Therefore, we have

$$\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(x-\mu_2)^2}{\sigma_1^2} = 0.$$

This gives the following solution

$$\mu_3 = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}. \tag{C.3}$$

Now, given the solution for x_3, we can calculate σ_3. To this aim, we note that the coefficient of the two multiplied Gaussians in Eq. C.2 is approximately equal to the coefficient of the derived Gaussian G_3. Therefore, we can divide by the common coefficient, apply the logarithm and solve for σ_3 by a least square fit over the exponent. That is,

$$\frac{d}{d\sigma_3}\int\left[-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{1}{2}\frac{(x-\mu_2)^2}{\sigma_2^2} + \frac{1}{2}\frac{(x-\mu_3)^2}{\sigma_3^2}\right]^2 dx = 0. \tag{C.4}$$

The integral in Eq. C.4 can not be solved generally, as the two polynomials diverge towards $\pm\infty$. However, integrating over a wide enough range, for instance from -10 to 10 should yield acceptable results. Differentiating with respect to σ_3 and integrating for $x = -10 \ldots 10$ gives the following equation

$$\sigma_3 = \sigma_1\sigma_2\sqrt{\frac{3(2000+\mu_3^2(200+\mu_3^2))}{\sigma_2^2(6000+100\mu_1^2+400\mu_1\mu_3+100\mu_3^2+3\mu_2^2\mu_3^2)+\sigma_1^2(6000+100\mu_2^2+400\mu_2\mu_3+100\mu_3^2+3\mu_2^2\mu_3^2)}}. \tag{C.5}$$

Although the above equation appears quite involved, its evaluation is simple and straightforward.

Finally, we need to find an estimate for a_3. Again, given the values for μ_3 and σ_3, this is straightforward

$$a^3 = G_1(\mu_3)G_2(\mu_3)\sigma_3\sqrt{2\pi}. \tag{C.6}$$

The validity of this approximation can be confirmed by looking at a simple example. Fig. C.1 shows two Gaussians, together with their product, which is shown by a dashed line. The product of the two Gaussians is shown overlayed with the above approximation in Fig. C.2. The error of the approximation, that is, $\mid G_1 \cdot G_2 - G_3 \mid$, is shown in Fig. C.3. As can be verified easily, the approximation error is within the bounds of numerical instabilities and is therefore negligible.

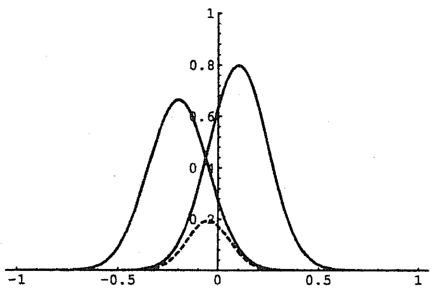


Figure C.1: Two Gaussians and their product

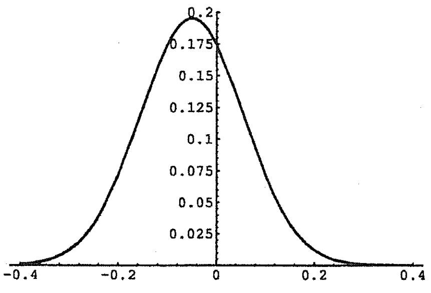


Figure C.2: Product of Gaussians overlayed with approximation

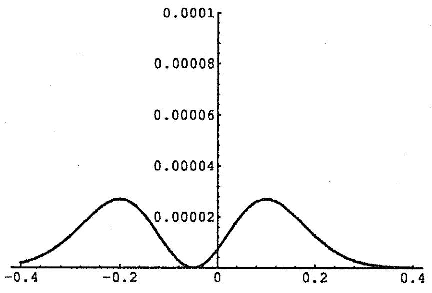


Figure C.3: Approximation error

APPENDIX D

Approximation of a Sum of Gaussians

Given a sum of Gaussian normal distributions as for instance obtained during the generalization of generalized predicates, we face the problem of finding a sufficiently close approximation of this sum in terms of a sum of Gaussians containing fewer terms than the initial sum. That is,

$$\begin{aligned} G(x) &= \sum_i^h \frac{a_i}{\sqrt{2\pi}\sigma_i} e^{\frac{-1}{2}\frac{(x-\mu_i)^2}{\sigma_i^2}} \\ G\text{approx}(x) &= \sum_i^k \frac{a_i}{\sqrt{2\pi}\sigma_i} e^{\frac{-1}{2}\frac{(x-\mu_i)^2}{\sigma_i^2}} \qquad \text{for some integer } k < h \\ G\text{approx}(x) - \Delta &\leq G(x) \leq G\text{approx}(x) + \Delta. \end{aligned}$$

This should not just be done by discarding summands that are deemed to be insignificant. But rather we would like to find an approximation that minimizes the error Δ. Since a direct least square fit yields non-linear equations that are very difficult to solve, we apply an iterative scheme, that approximates one Gaussian after the other, using the results from Appendix C.

We proceed as follows. First, let us assume that the maximal component of the sum of Gaussians can be found around the maximum of the sum. Therefore,

$$\mu_0 = x \quad \text{such that } G(x) \text{ is maximal.} \tag{D.1}$$

Again, we solve for σ by a least square fit over the exponent. To this aim, we assume that $G(x)$ has the form of a Gaussian. Therefore, we can write the equation for the least square fit as follows

$$\sum \frac{d}{d\sigma_0}\left[\ln(G(x)/G(\mu_0)) + \frac{1}{2}\frac{(x-\mu_0)^2}{\sigma_0^2}\right]^2 = 0. \tag{D.2}$$

Solving for σ_0^2 gives

$$\sigma_0^2 = -\frac{\sum(x-\mu_0)^2}{2\sum \ln(G(x)/G(\mu_0)}. \tag{D.3}$$

Finally, we can calculate the scaling factor as

$$a_0 = G(\mu_0)\sqrt{2\pi}\sigma_0. \tag{D.4}$$

Hence, we obtain the first term of the approximation as

$$G_0(x) = \frac{a_0}{\sqrt{2\pi}\sigma_0} e^{\frac{-1}{2}\frac{(x-\mu_0)^2}{\sigma_0^2}}. \tag{D.5}$$

Using this first approximation term, we can calculate a new function, namely

$$G'(x) = G(x) - G_0(x). \tag{D.6}$$

If we now apply the same procedure as above to $G'(x)$, we successively obtain the lesser terms of the approximation, namely $G_1(x), G_2(x), \ldots G_i(x)$. This procedure can be repeated either until a previously fixed number of approximation terms has been calculated, or until the remaining function $G'(x)$ is below a certain threshold Δ.

Again, we can look at an example in order to verify the above procedure empirically. Fig. D.1 shows a sum of Gaussians with 25 terms (solid line), overlayed with their approximation using 7 terms (dashed line). No significant difference can be seen. This is further stressed by plotting the approximation error, that is, $\mid G(x) - G_{\text{approx}}(x) \mid$ in Fig. D.2. The approximation appears to be sufficiently accurate. However, to improve accuracy even further, in the actual implementation of the system, approximations with 12 terms have been used.

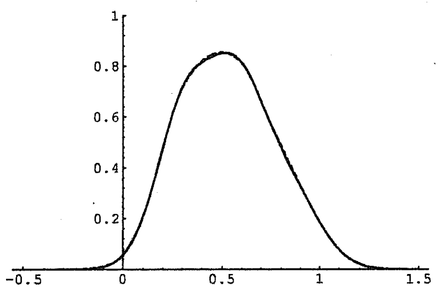


Figure D.1: Sum of Gaussians overlayed with their approximation

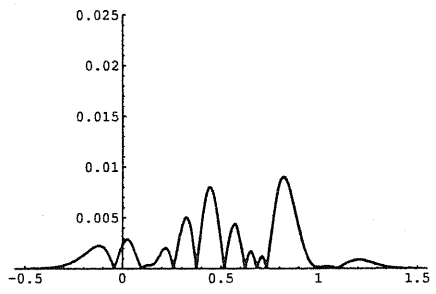


Figure D.2: Approximation error

APPENDIX E

Splitting of Generalized Predicates

An unsupervised approach for the splitting of bimodal histograms has been proposed by Otsu [40]. First, we assume a discrete, normalized histogram, which represents a probability distribution such that

$$\sum p_i = 1, \tag{E.1}$$

with the range $i = 1, \ldots, L$. Suppose that we divide the distribution into two classes by thresholding it at some level k. C_0 denotes the class with $i \leq k$, and C_1 represents the class with $i > k$. The class probabilities ω_0 and ω_1, together with the means μ_0 and μ_1 are given by

$$\omega_0 = \sum_{i=1}^{k} p_i = \omega(k) \tag{E.2}$$

$$\omega_1 = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{E.3}$$

$$\mu_0 = \sum_{i=1}^{k} \frac{ip_i}{w_0} = \frac{\mu(k)}{\omega(k)} \tag{E.4}$$

$$\mu_1 = \sum_{i=k+1}^{L} \frac{ip_i}{w_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)}, \tag{E.5}$$

where

$$\mu(k) = \sum_{i=1}^{k} ip_i \tag{E.6}$$

$$\mu_T = \mu(L) = \sum_{i=1}^{L} i p_i. \tag{E.7}$$

$\omega(k)$ and $\mu(k)$ are the 0th and the 1st order moments of the distribution up to the kth level, and μ_T is the mean of the original distribution. For any choice of k, we have

$$\omega_0 \mu_0 + \omega_1 \mu_1 = \mu_T, \qquad \omega_0 + \omega_1 = 1. \tag{E.8}$$

Finally, the variances for the two classes are given by

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 \frac{p_i}{\omega_0} \tag{E.9}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 \frac{p_i}{\omega_1}. \tag{E.10}$$

In order to find the best threshold k, the criterion measure, given by

$$\eta = \frac{\sigma_B^2}{\sigma_T^2}, \tag{E.11}$$

where

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{E.12}$$

$$\sigma_T^2 = \sum_{i=1}^{L} (i - \mu_t)^2 p_i, \tag{E.13}$$

must be minimized. Since σ_T^2 does not depend on k, the optimal threshold k^* that minimizes η, maximizes σ_B^2. Therefore, the optimal threshold k^* is given as

$$\sigma_B^2(k^*) = \max_{1 \le k \le L} \sigma_B^2(k), \tag{E.14}$$

where

$$\sigma_B^2 = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]}. \tag{E.15}$$

Since both, $\omega(k)$ and $\sigma(k)$ can be updated cumulatively, Eq. E.14 corresponds to a simple linear search, which is very effective in practice.

APPENDIX F

Grammar used for Description

F.1 Transitions

```
(Description    →   (VariableGroupDescription   RemainingSegments)
                    (SizeGroupDescription   RemainingSegments)
                    (RemainingDescription))
(VariableGroupDescription
                →   (VariablePrologue   VariableGroup))
(VariableGroup
                →   (VariableGroup   VariableGroup)
                    (VariableGroupContents   VariableGroupMembers))
(VariableGroupContents
                →   (SubjectPhrase   Group   Name   Consist   ObjectPhrase   Identifier   Dot))
(VariableGroupMembers
                →   (VariableGroupMembers   VariableGroupMembers)
                    (VariableMemberDecsription))
(VariableMemberDescription
                →  (SubjectPhrase  Identifier  Verb2  ObjectPhrase  ModifierPhrase  Dot))

(SizeGroupDescription     →  (SizePrologue  SizeGroup))

(SizeGroup      →  (SizeGroup  SizeGroup)
                   (SizeMemberDescription))

(SizeMemberDescription
                →  (SizeGroupContents  Pronoun  Verb2  SizeGroupRelation  Dot))

(SizeGroupContents
                →  (SubjectPhrase  Group  Name  Consist  ObjectPhrase  Identifier))

(SizeGroupRelation        →  (SizeModifier  Conjunction  ParallelModifier)
                             (SizeModifier)
                             (ParallelModifier))

(ParallelModifier         →  (ParallelAngle  Rel1  StructureModifier))

(RemainingSegments        →  (SegmentDescription  RemainingDescription))

(RemainingDescription
                →  (RemainingDescription  RemainingDescription)
                   (SingleSegment))

(SingleSegment
                →  (SubjectPhrase  Identifier  Verb2  ObjectPhrase  Modifier  Dot))

(SegmentDescription
                →  (SubjectPhrase  Identifier  NonMembership  ObjectPhrase  Group  Dot))

(Modifier                 →  (AngleModifier  Conjunction  SizeModifier)
                             (AngleModifier)
                             (SizeModifier))

(AngleModifier            →  (AngleDirection  Rel1  Name)
                             (AngleDirection))
```

```
(AngleDirection            →  (ParallelDirection)
                              (NormalDirection)
                              (VariableDirection))
(ParallelDirection         →  (ParallelAngle))
(NormalDirection           →  (Rel2   Article   RightAngle))
(VariableDirection         →  (Rel2   VariableAngle))
(SizeModifier              →  (SizeRelation   Rel3   Name)
                              (SizeRelation))
(SizeRelation              →  (SizeEqual)
                              (LengthEqual)
                              (WidthEqual))
(SizeEqual                 →  (SameSize))
(LengthEqual               →  (SameLength))
(WidthEqual                →  (SameWidth))
(Identifier                →  (SegmentIdentifier)
                              (GroupIdentifier))
(SegmentIdentifier         →  (Segment   Name))
(GroupIdentifier           →  (Group   Name))
(VariablePrologue          →  (SubjectPhrase   Intro   VariableCharacteristic   Dot))
(SizePrologue              →  (SubjectPhrase   Intro   Dot))
(Intro                     →  (Existential   Verb2   Number   Group))
(VariableCharacteristic    →  (Pronoun   Verb1   VariableDirection   Rel4   Rel1   Rel5))
```

F.2 Terminals

```
(Group              →   (TERM   "group"   "groups"))
(Segment            →   (TERM   "segment"   "segments"))
(ParallelAngle      →   (TERM   "parallel"   NIL))
(RightAngle         →   (TERM   "right angle"   NIL))
(VariableAngle      →   (TERM   "variable angle"   NIL))
(SameSize           →   (TERM   "of same size"   NIL))
(SameLength         →   (TERM   "of same length"   NIL))
(SameWidth          →   (TERM   "of same width"   NIL))
(NonMembership      →   (TERM   "does not belong to any"   "do not belong to any"))
(Existential        →   (TERM   "there"   NIL))
(Consist            →   (TERM   "consists of"   NIL))
(Article            →   (TERM   "a"   NIL))
(Rel1               →   (TERM   "to"   NIL))
(Rel2               →   (TERM   "at"   NIL))
(Rel3               →   (TERM   "as"   NIL))
(Rel4               →   (TERM   "in respect"   NIL))
(Rel5               →   (TERM   "each other"   NIL))
(Pronoun            →   (TERM   "which"   NIL))
(Conjunction        →   (TERM   "and"   NIL))
(Verb1              →   (TERM   "has"   "have"))
```

(Verb2	→	(TERM "is" "are"))
(Name	→	(TERM NIL NIL))
(Number	→	(TERM NIL NIL))
(Dot	→	(TERM "." NIL))

F.3 Grammar Daemons

(SubjectPhrase	→	(GRAM T))
(ObjectPhrase	→	(GRAM NIL))

Bibliography

[1] H.A. Almohamad and S.O. Duffuaa, "A Linear Programming Approach for the Weighted Graph Matching Problem", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-15, no.5, pp. 522–525, 19893.

[2] C. Arcelli and L. Serino, "Finding Near-Convex Shapes in Complex Patterns", in H. Bunke, (Ed.), *Advances in Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992.

[3] D. Arita, N. Tsuruta, R. Taniguchi, and M. Amamiya, "A Model Acquisition Method for Object Recognition by Pictorial Examples", *Technical Report of IEICE*, vol. PRU93-141, pp. 41 -48, 1994, [In Japanese].

[4] Y. Azumatani and K. Abe, "A Method for Connecting Contour Lines Using the Expansion Operation", in *Proc. of Spring National Conv. of IECEJ*, p. 1689, 1985, [In Japanese].

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

[6] A. Barr and E.A. Feigenbaum, (Eds.), *The Handbook of Artificial Intelligence, Vol. I*, Addison-Wesley, Reading, Massachusetts, 1981.

[7] L. Boxer, "Computing Deviations from Convexity in Polygons", *Pattern Recognition Letters*, vol. 14, pp. 163–167, 1993.

[8] C.B. Boyer and U.C. Merzbach, *A History of Mathematics*, John Wiley & Sons, New York, 1989.

[9] H. Bunke and G. Allermann, "Inexact Graph Matching for Structural Pattern Recognition", *Pattern Recognition Letters*, vol. 1, no.4, pp. 245–253, 1983.

[10] P.R. Cohen and E.A. Feigenbaum, (Eds.), *The Handbook of Artificial Intelligence, Vol. III*, Addison-Wesley, Reading, Massachusetts, 1982.

[11] J.H. Connell and M. Brady, "Generating and Generalizing Models of Visual Objects", *Artificial Intelligence*, vol. 31, pp. 159–183, 1987.

[12] G. Dong and M. Yachida, "Learning Structural Concept from 3-D Quantitative Information of Objects Obtained woith Stereo Vision", *Technical Report of IEICE*, vol. PRU94-41, pp. 17–24, 1994.

[13] G. Dong, T. Yamaguchi, and M. Yachida, "Learning Shape Concept from 3-D Spatial Structure of Objects", to appear in *Proc. PRICAI'94*, 1994.

[14] G. Dong, T. Yamaguchi, Y. Yagi, and M Yachida, "Shape Concept Learning from Examples and Explanation", *Computer Vision*, vol. 87-8, pp. 57–64, 1994, [In Japanese].

[15] L. Dorst and A.W.M. Smeulders, "Length Estimators for Digitized Contours", *Computer Vision, Graphics, and Image Processing*, vol. 40, pp. 311–333, 1987.

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

[17] M.A. Fischler and P. Barrett, "An Iconic Transform for Sketch Completion and Shape Abstraction", *Computer, Graphics, and Image Processing*, vol. 13, pp. 334–360, 1980.

[18] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freman, San Francisco, 1979.

[19] R.L. Haar, "Sketching: Estimating Object Positions from Relational Descriptions", *Computer, Graphics, and Image Processing*, vol. 19, pp. 227–247, 1982.

[20] H. Hadwiger, *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie*, Springer, Berlin, 1957.

[21] R.M. Haralick, M.Y. Jaishimha, and D. Dori, "Quantitative Performance Evaluation of Thinning Algorithms in the Presence of Noise", in C. Arcelli, L.P. Cordella, and G. Sanniti di Baja, (Eds.), *Aspects of Visual Form Processing*, pp. 261–286, World Scientific, Singapore, 1994, Proc. *2nd Int. Workshop on Visual Form*, Capri, Italy.

[22] D. Heckerman, "Probabilistic Interpretations for MYCIN's Certainty Factors", in L.N. Kanal and J.F. Lemmer, (Eds.), *Uncertainty in Artificial Intelligence*, pp. 167–196, Elsevier Science Publishers, North-Holland, 1986.

[23] A. Held and K. Abe, "On Approximate-Convexity", *Pattern Recognition Letters*, vol. 15, no.6, pp. 611–618, 1994, also in *Technical Report of IEICE*, PRU92-53.

[24] A. Held and K. Abe, "On the Decomposition of Binary Shapes into Meaningful Parts", *Pattern Recognition*, vol. 27, no.5, pp. 637–647, 1994.

[25] A. Held, K. Abe, and C. Arcelli, "Towards a Hierarchical Contour Description via Dominant Point Detection", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-24, no.6, pp. 942–949, 1994.

[26] T. Ibaraki, K. Abe, and C. Arcelli, "Decomposition of Two-Dimensional Shapes into Meaningful Parts", in *Proc. of Asian Conference on Computer Vision, ACCV*, Osaka, Japan, 1993.

[27] Y. Ito, K. Abe, and C. Arcelli, "Region Organization in Two-Dimensional Shapes", in V. Cantoni et al., (Ed.), *Progress in Image Analysis and Processing*, World Scientific, Singapore, 1990.

[28] B.K. Jang and R.T. Chin, "Analysis of Thinning Algorithms Using Mathematical Morphology", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-12, no.6, pp. 541–551, June 1990.

[29] H.S. Kim and K.H. Park, "Shape Decomposition Based on Perceptual Structure", in L.G. Shapiro, (Ed.), *Computer Vision and Image Processing*, Academic Press, New York, 1992.

[30] L. Kitchen and A. Rosenfeld, "Discrete Relaxation for Matching Relational Structures", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-9, no.12, pp. 869–874, 1979.

[31] Y. Kodratoff and R. Lemerle-Loisel, "Learning Complex Structural Descriptions from Examples", *Computer Vision, Graphics, and Image Processing*, vol. 27, pp. 266–290, 1984.

[32] M. Lebowitz, "The Utility of Similarity-Based Learning in a World Needing Explanation", in Y. Kodratoff and R.S. Michalski, (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. III*, pp. 399–433, Morgan Kaufmann, Los Altos, 1990.

[33] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, 1975.

[34] P.B. Medawar, *The Limits of Science*, Oxford University Press, Oxford, 1984.

[35] R.S. Michalski, "Learning from Observation: Conceptual Clustering", in R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 331–363, Springer, New York, 1984.

[36] R.S. Michalski, "A Theory and Methodology of Inductive Learning", in R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 83–133, Springer, New York, 1984.

[37] H. Minkowski, "Volumen und Oberfläche", *Math. Annal.*, vol. 57, pp. 447–495, 1903.

[38] J.C. Mullikin and P.W. Verbeek, “Surface Area Estimation of Digitized Planes”, *Bioimaging*, vol. 1, no.1, pp. 6–16, 1993.

[39] R. Ogniewicz and M. Ilg, “Voronoi Skeletons: Theory and Applications”, in *Proc. Conf. on Comp. Vision and Patt. Recog., CVPR 92*, pp. 63–69, 1992, Champaign, USA.

[40] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms”, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-9, no.1, pp. 62–66, January 1979.

[41] S.K. Parui and A. Datta, “A Parallel Algorithm for Decomposition of Binary Objects through Skeletonization”, *Pattern Recognition Letters*, vol. 12, no.4, pp. 235–240, April 1991.

[42] T. Pavlidis, “A Review of Algorithms for Shape Analysis”, *Computer, Graphics, and Image Processing*, vol. 7, no.2, pp. 243–258, April 1978.

[43] I. Pitas and A.N. Venetsanopoulos, “Morphological Shape Decomposition”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-12, no.1, pp. 38–45, January 1990.

[44] I. Pitas and A.N. Venetsanopoulos, “Morphological Shape Representation”, *Pattern Recognition*, vol. 25, no.6, pp. 555–565, 1992.

[45] L. Rendell, “Similarity-Based Learning and its Extensions”, *Comput. Intell.*, vol. 3, pp. 241 - 266, 1987.

[46] B. Russell, *History of Western Philosophy*, George Allen & Unwin Ltd, 1961.

[47] L.A. Santaló, *Integral Geometry and Geometric Probability*, Addison-Wesley, Reading, 1976.

[48] B. Schachter, “Decomposition of Polygons into Convex Sets”, *IEEE Trans. on Computers*, vol. C-27, no.11, pp. 1078–1082, November 1978.

[49] J. Segen, “Model Learning and Recognition of Nonrigid Objects”, in *Proc. CVPR 89*, pp. 597–602, 1989, San Diego, USA.

[50] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[51] L.G. Shapiro, “A Structural Model of Shape”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 111–126, 1980.

[52] L.G. Shapiro and R.M. Haralick, “Decomposition of two-dimensional shapes by graph-theoretic clustering”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 10–20, 1979.

[53] R. Simmons and J. Slocum, “Generating English Discourse from Semantic Networks”, *Comm. of the ACM*, vol. 15, no.10, pp. 891–905, 1972.

[54] J. Sklansky, “Measuring Concavity on a Rectangular Mosaic”, *IEEE Trans. on Computers*, vol. C-21, no.12, pp. 1355–1364, 1972.

[55] H.I. Stern, “Polygonal Entropy: A Convexity Measure”, *Pattern Recognition Letters*, vol. 10, pp. 229–235, 1989.

[56] S.R. Sternberg, “Grayscale Morphology”, *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 333–355, 1986.

[57] W.H. Tsai and K.S. Fu, “Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis”, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-9, no.12, pp. 757–768, 1979.

[58] N. Ueda and S. Suzuki, “Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-15, no.4, pp. 337–352, 1993.

[59] J.R. Ullmann, “An Algorithm for Subgraph Isomorphism”, *Journal of Assoc. Computing Machinery*, vol. 23, no.1, pp. 31–42, 1976.

[60] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-10, no.5, pp. 695–703, 1988.

[61] P.H. Winston, "Learning Structural Descriptions from Examples", in P.H. Winston, (Ed.), *The Psychology of Computer Vision*, pp. 157–209, McGraw-Hill, New York, 1975.

[62] J. Wnek and R.S. Michalski, "Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments", *Machine Learning*, vol. 14, pp. 139 - 168, 1994.

[63] H. Yamada, K. Yamamoto, S. Matsui, T. Saito, and S. Muraki, "MAP: Multi-Angled Parallelism for Feature Extraction from Topographical Maps", in *Proc. of IEEE Int. Conf. on Image Proc., ICIP'89*, pp. 83–87, 1989, Singapore.

List of Publications

[1] A. Held and K. Abe, "On the Decomposition of Binary Shapes into Maximal Approximately Convex Subparts", in H. Bunke, (Ed.), *Advances in Structural and Syntactic Pattern Recognition*, pp. 227–236, World Scientific, Singapore, 1992.

[2] A. Held and K. Abe, "Learning of Structural Concepts based on Similarities and Dissimilarities", in *Proc. Asian Conf. on Computer Vision, ACCV93*, pp. 688–691, 1993, Osaka, Japan.

[3] A. Held and K. Abe, "Learning Structures from Raw Images", *Reports of the Graduate School of Electronic Science and Technology, Shizuoka University*, vol. 15, pp. 83–88, 1994.

[4] A. Held and K. Abe, "On the Decomposition of Binary Shapes into Meaningful Parts", *Pattern Recognition*, vol. 27, no.5, pp. 637–647, 1994.

[5] A. Held and K. Abe, "On Approximate-Convexity", *Pattern Recognition Letters*, vol. 15, no.6, pp. 611–618, 1994, also in *Technical Report of IEICE*, PRU92-53.

[6] A. Held and K. Abe, "Learning Model Structures from Binary Images", *Technical Report of IEICE*, vol. PRU94-2, pp. 9–16, 1994.

[7] A. Held and K. Abe, "Model Acquisition by Learning Visual Structures", in *Proc. 8th Annual Conf. of JSAI*, pp. 183–186, 1994, Tokyo, Japan.

[8] A. Held and K. Abe, "Learning Model Structures from Images", *IEICE Transactions on Information and Systems*, vol. E77-D, no.11, pp. 1281–1290, 1994.

[9] A. Held, J. Jia, and K. Abe, "Approximate Convexity in N-Dimensions", in C. Arcelli, L.P. Cordella, and G. Sanniti di Baja, (Eds.), *Aspects of Visual Form Processing*, pp. 251–260, World Scientific, Singapore, 1994, Proc. *2nd Int. Workshop on Visual Form*, Capri, Italy.

[10] A. Held and K. Abe, "Learning Structural Descriptions from Images", to appear in Proc. *Workshop on Structural and Syntactic Pattern Recognition*, SSPR '94, Nahariya, Israel, 1994.