A New Neural Network Based Approach to Position and Scale Invariant Pattern Recognition

Emmanouel Christopher Mertzanis

Submitted for the degree of Doctor of Philosophy

University of York
Department of Computer Science
October 1992

To Barbara

ACKNOWLEDGEMENTS

The author would like to acknowledge the important contribution of Dr. James Austin, Dr. Ian Benest and Dr. A. Wood in the completion of the present thesis. Their advice, and helpful research guidelines were vital to the final completion of my thesis as were their useful hints during our numerous discussions of the past three years. The author wishes also to acknowledge the help of Davor Dukic who, through his image manipulation graphical tool (the DrawingBoard), made the generation of most of the enclosed complicated graphs a reality. His comments along the various stages of my thesis have been particularly helpful, as was the help of George Bolt in the design of ADAM neural network software. Last, and more important, I would like to thank my parents for their financial and moral support during my studies in York. Without their continuous support and encouragement the present thesis would have been impossible to achieve.

Declaration

Parts of the following thesis chapters have been presented in the publications listed below:

[1] **E.C. Mertzanis** , "Image Data Bases", Neural Networks for Automatic Target Recognition, Research Conference at Wang Institute of Boston University, Boston University Press, May 11-13, 36, 1990.

[2] **E.C. Mertzanis** , "Normalised Octree/Quadtree Structures for Efficient Position Invariant Pattern Recognition", Neural Networks for Vision and Image Processing, Research Conference at Wang Institute of Boston University, Boston University Press, May 10-12, 45, 1991.

[3] **E.C. Mertzanis** , "Quadtrees for Neural Network based Position Invariant Pattern Recognition", NCM'91 : Applications of Neural Networks, Centre of Neural Networks, King's College, London, U.K., October 1, 2-3 , 1991.

[4] **E.C. Mertzanis** , "Hierarchical Volume Data Structures for position independent Pattern Recognition", International Conference on Artificial Neural Networks, San Diego, California (USA), May 29-31, 230-242, 1991.

[5] **E.C. Mertzanis** , "Hierarchical Volume Data Structures for position independent Pattern Recognition", AMSE REVIEW, Vol. 22, No. 1, 53-64, 1992.

[6] **E.C. Mertzanis** , "Normalised Hierarchical Data Structures for Automatic Target Recognition", IEE Second International Conference on Artificial Neural Networks, Bournemouth International Centre, U.K., 18-20 November, Vol. 349, 229-233, 1992.

[7] **E.C. Mertzanis, Davor A. Dukic, Ian D. Benest, James Austin** , "A Neural Network Based Position Invariant Pattern Recognition System with a Constrained Hypermedia Style User-Interface", IEE Fourth International Conference on Image Processing and its Applications, Maastricht, The Netherlands, 7-9 April, 1992.

[8] **E.C. Mertzanis, James Austin** , "Linear Quadtrees for Neural Network based Position Invariant Pattern Recognition", "Series in Neural Networks", Springer Verlag, 1992 (to be published)

"But if I'm content with a little,
Enough is as good as a feast"

Isaac Bickerstaffe

Contents.

Chapter 9. Conclusions.

Abstract

This thesis describes a new technique, through which the invariant three dimensional object classification problem can be effectively solved for both uncluttered and noisy scenes. The technique is based on a new algorithmic approach that modifies the known concept of intersecting three 3D object pseudo-volumes in order to create a unique invariant three dimensional object volume description. The thesis shows that the proposed algorithm for volume intersection together with a neural network classifier, can perform effectively in both uncluttered and noisy environments. In particular, a single picture of each aircraft is required, in order to achieve an error-free aircraft classification in a variety of noisy and uncluttered environments.

The thesis additionally presents the results of research in using a standard relational database system for an exact and error-free pattern matching operation. The pattern matching was applied on compressed aircraft pictures that are stored within the relational database system. In that context, the wider problem of partial pattern matching is also investigated, and it is concluded that although an exact and error-free pattern matching operation on compressed aircraft images can be achieved, with a standard relational database environment, it takes an unsatisfactory long processing time to be completed.

The thesis also reviews the standard shape description techniques for two and three dimensional feature extraction. The reviewed techniques for shape analysis are presented in the context of a position and scale invariant pattern recognition operation. These methods include the moments invariants and the normalised Fourier descriptors. It is concluded that in the context of the invariant aircraft identification approach, the above methods possess a number of highly desirable properties. They demonstrate a high classification success ratio, and are both computationally efficient and simple to implement. However, both methods suffer from information redundancy, information loss, data suppression and require a large number of a priori information for a successful object identification. Because the limitations of both techniques are considered unreliable for the general three dimensional pattern recognition problem.

Finally, the thesis proposes a constraint hypermedia style user-interface, through which the pattern recognition process can be implemented. The aim is to present a user-interface through which a significant number of images (more than a hundred) can be simply and efficiently introduced into the system, can then be effectively processed using a variety of image processing techniques, and can finally generate suitable patterns for training and testing the underlying neural network architecture. Furthermore, within the proposed user-interface all results regarding a specific experiment can be kept together, and additional performance charts can also be added in order to demonstrate graphically the overall system's classification behaviour.

CHAPTER 1

Introduction

"Those who wish to succeed must ask the right preliminary questions"

Aristotle

1.1 Background

As the speed, capability, and economic advantages of modern signal processing devices continue to increase, there is simultaneously an increase in efforts aimed at developing sophisticated real time automatic systems capable of emulating human abilities. This thesis concerns one of the most obvious of these, namely *vision*. Energy from our three dimensional world is converted into two dimensional entities called images by either an electronic sensor or our visual system, such as a television camera. The way in which these images are processed to take actions or form conclusions is a subject of continuous research by people from various disciplines. The processing algorithms are generally far from exact or complete. Lacking this detailed knowledge, the difficulty encountered when attempting to develop autonomous systems that could emulate this processing is in many cases substantial. It appears to be a formidable and challenging task to automatically manipulate and encode years of human experience in effectively processing visual images from a three dimensional world.

This thesis aims at investigating the problem of simulating human vision in the specific context of recognising three dimensional objects. The same limitations that are encountered in real world, regarding the recognition of

objects independent of their positions, relative sizes, and cluttered environments are also investigated in the thesis. Time considerations, an all important property of human observers, are also taken into consideration and alternative architectures are presented that offer computational responses close to human performance. Of course other tasks also exist that involve the processing of images by computer where the objective is not to emulate human behaviour, but rather to extract or display more (subjectively) useful information from the image. Examples are the enhancement of images degraded by noise and the splicing of low resolution images to achieve a higher resolution composite. This first task is also addressed in the thesis.

Interest in computer vision stems from two principal application areas: improvement of pictorial information for human interpretation, and processing of scene data for autonomous machine perception. The early 1920s Bartlane system, used for transmitting cable pictures, could code images in 5 distinct brightness levels. This capability was increased to 15 levels in 1929. During this period, the reproduction process was also improved considerably by the introduction of a system for developing a film plate using light beams that were subsequently modulated by a coded picture tape. Although over the next 35 years significant improvements were made regarding processing methods for transmitted digital pictures, it took the combined introduction of digital computer mainframes and the processing time restrictions of the space program to bring into attention the potential of computer vision concepts. Work on using computer techniques for enhancing images from a space probe began at the Jet Propulsion Laboratory in Passadena, California, in 1964, when pictures of the moon transmitted by Ranger 7 were processed in order to correct image distortion inherent in the on-board television camera. This type of project served as the basis for the creation of a number of improved methods that were used in the enhancement and restoration of images from familiar programs such as the Surveyor missions to the moon, the Mariner series of flyby missions to Mars, and the Apollo manned flights to the moon.

From 1964 to the present day, the field of computer vision has experienced a considerable and vigorous growth. In addition to applications in the space program, computer vision methods are used today in a variety of problems that, although often unrelated, share though a common need for methods capable of enhancing pictorial information for human interpretation and analysis. In areas such as medicine, doctors are assisted by computer procedures that can enhance the contrast of a picture or code its intensity values into colour for a visually improved interpretation of biomedical images. Similar techniques are used by geographers in studying pollution patterns from aerial and satellite imagery. Image enhancement and restoration procedures have been used to process degraded images revealing unrecovable objects or improving experimental results too expensive to duplicate. In physics and related fields, images of experiments in such areas as high energy plasmas and electron microscopy are often enhanced by computer vision techniques. Similar successful applications of computer vision and image processing concepts can be found in astronomy, biology, nuclear medicine, law enforcement, defence, and industrial applications.

The second major area of application of computer vision techniques, mentioned at the beginning of this section, is in problems dealing with machine perception. In this case, interest is focused on procedures for extracting image information in a form suitable for computer processing. This information has little resemblance to visual features used by humans in interpreting the content of an image. Examples of the type of information used in machine perception are statistical moments, Fourier transform coefficients, and multi-dimensional distance measures. This thesis describes research work undertaken in the area of machine perception.

1.2 The difficulty in describing a scene.

The extensive processing capabilities of human visual systems point to the fact that much of the cognitive processes are buried in the subconscious.

The difficulty of describing the exact operation of subconscious functions presents a significant problem in developing algorithms that are capable of emulating the human visual behaviour. A simple example that can visually illustrate this problem at the lowest level is presented below. A test image of an easily recognised object (an aircraft) is shown in Figure 1.1a. A plot of the image intensity function (histogram), that is, the image array of intensities comprising this image, is shown in Figure 1.1b.

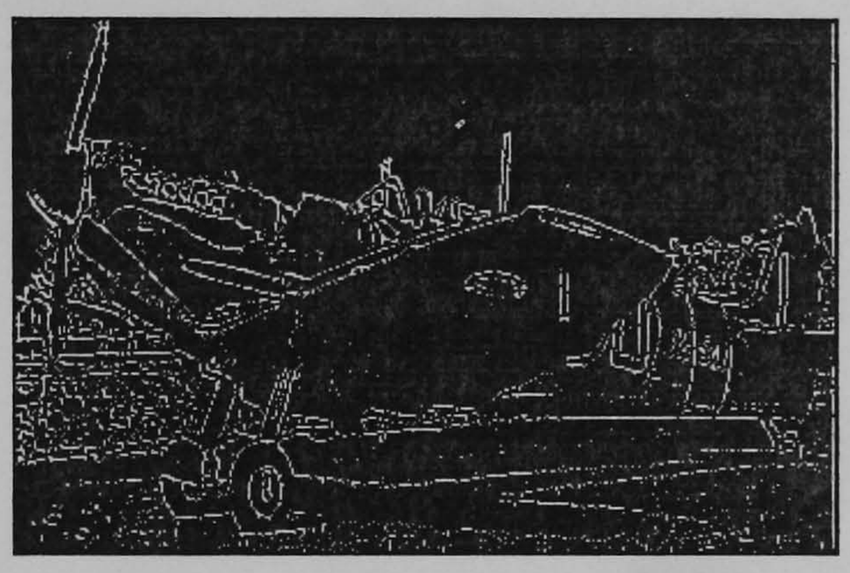


Figure 1.1a. An aircraft.

To human observers the intensities plot conveys little visual information regarding the object contained in a scene. At a lower level, if a plot of contours of constant image intensity can be drawn, the formulated topographic image map will represent a form of low level image pre-processing and will reflect information associated to areas of the image that are similar in intensity. The crude outline of an aircraft could then be evident. Finally, Figure 1.1c shows a pre-processing version of the image of Figure 1.1a where edges have been extracted using computer vision techniques. From the pre-processed version of the image in Figure 1.1c, it can be seen that the important shape information content of the image has been retained with a substantial reduction in the amount of raw data. Hopefully, this simple example reveals the complexities involved in developing image understanding systems and demonstrates that an image processing task is

often regarded as trivial until the algorithm must be coded. Unfortunately, there are no pre-defined data types in any current programming language, such as "aircraft", "box", "house", "robot arm", that can straightforwardly be associated to the previously mentioned simple objects. In fact, a detailed and in depth research and experimentation is always required in order to efficiently code such entities and provide a unified model that could be invariant to any possible rotational, positional and size changes of the object.

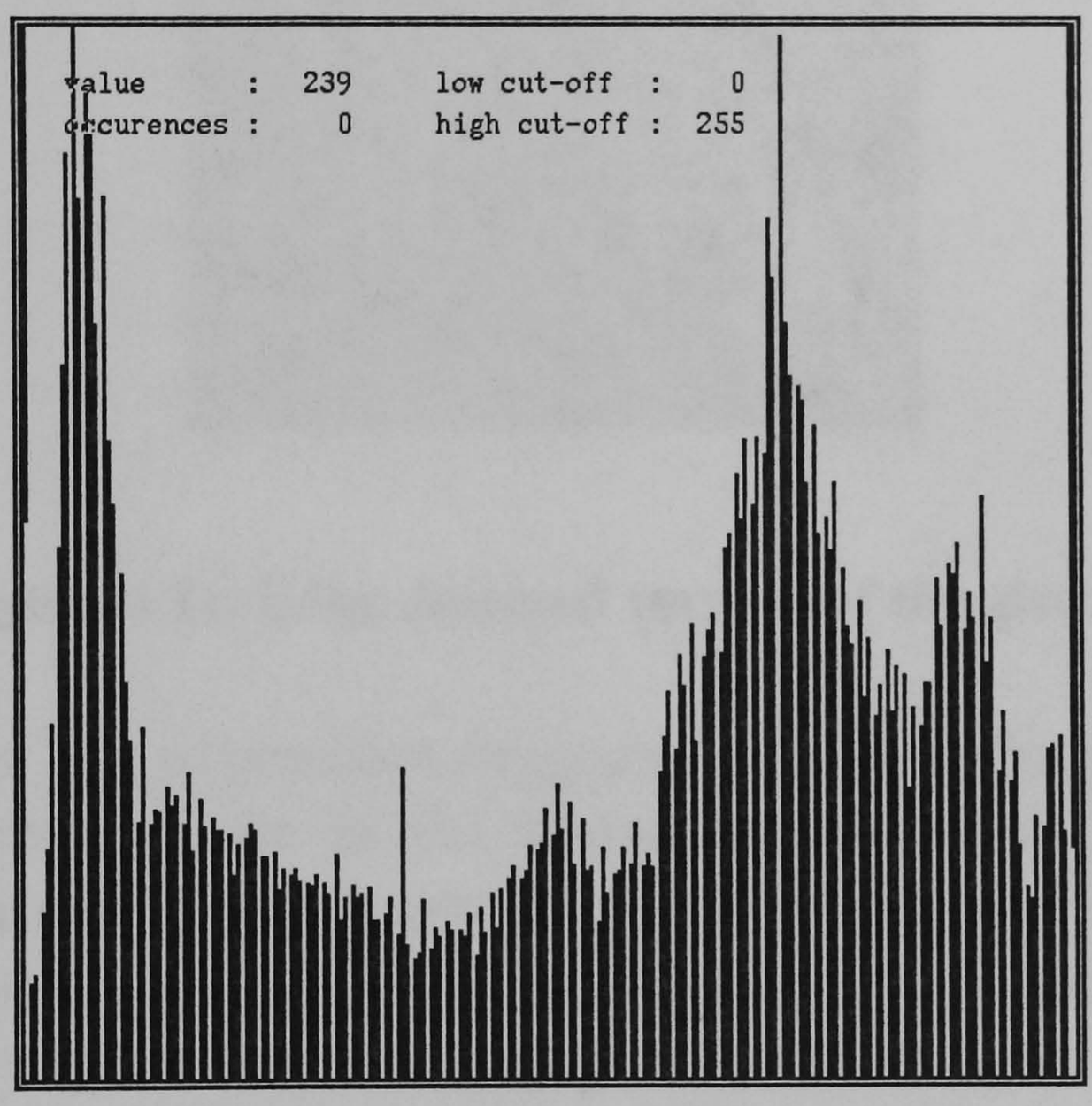


Figure 1.1b. A 2D plot of the image intensity values.

In the same way that the computer system does not "see" high level entities, but rather only takes as input a spatially distributed array of intensities or numbers, a human observer presented with image data that are displayed in a numerical form, often has a nearly impossible time trying to infer anything about the image content. Figure 1.1b simply proves this. Thus,

the inherent familiarity with the capabilities of our visual system often belies as the understanding of its operations. Alternately, there also exist situations wherein autonomous vision systems yield results with precision far exceeding the human visual system capability. An example can be found in stereo vision, where it is possible to develop systems with three dimensional distance resolutions on the order of thousands of an inch.

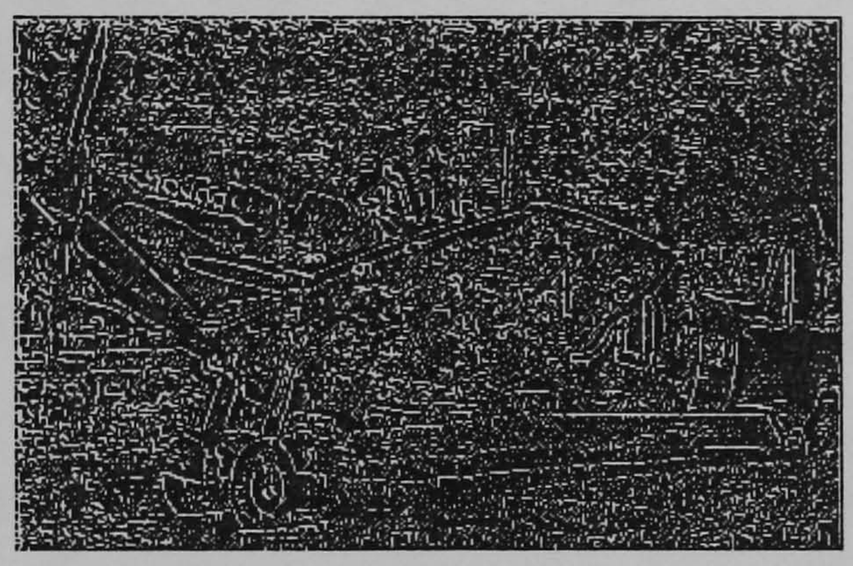


Figure 1.1c. Edge detected version of the aircraft.

The human lack of precision suggests that a certain level of ambiguity is allowed at certain levels in the human visual system. The emulation, quantifying, or coding of this ambiguity is always an exceedingly difficult problem. Typically, a few degrees rotation of an image makes little difference in the eventual interpretation by a human observer; but this rotation will probably cause major changes in the spatial distribution of sensor output data and may yield major errors in autonomous system processing. Clearly the last observation shows the scale of difficulty the current thesis has to face. Typically, an invariant three dimensional object recognition process should be effectively independent of any possible object's rotation angle and relative size change.

1.3 Image Representations

An image is defined as a two dimensional function generated by viewing (sensing) a scene; often called a picture. A sensor typically converts scene information into images, and it is the non-unique nature of this transformation that presents many of the difficulties in the processing, analysis, and interpretation of imagery. A scene is defined as a collection of three dimensional objects with some topological arrangement usually governed by the physical laws of nature. The image is represented by image functions such as, f(x,y), f(x_1, x_2), f($\vec{x}$), where the independent variables (x,y or x_1,x_2, or $\vec{x}$) are spatial coordinates related to physical locations in the sensor image plane, and f is the light's intensity at these locations. The range of values of f is known as the grayscale range. In the particular subcase where f takes on at most two values, the image is said to be binary. In binary images each of these two quantities has an associated set of units, x_1 and x_2, that are often measured with respect to or in multiples of the image plane dimensions (pixels). It is the seemingly simplistic relationship described above that yields a lot of difficulty. Since both x_1 and x_2 values in a binary image are spatial quantities, the concepts of two dimensional geometry and shape arise. Conversely, the quantity f is related to the light intensities of the scene being imaged. Together, these quantities yield a distributed array of intensities. Unfortunately though, a functional form of f($\vec{x}$) is almost never known, although for limited ranges of $\vec{x}$ a functional form may be approximated. The lack of a more mathematically tractable model capable of relating changes in $\vec{x}$ to those in f (closed-form and single-valued) presents much difficulty in algorithm development, since there is little underlying mathematical rigour.

Grayscale images allow more than two intensity values. For illustration, the human visual system has a grayscale perception range of about 64 distinct levels (6 bits). In other words, a human observer shown a 6 bit (or larger) grayscale intensity scale chart with intensity bars ranging in intensity from lowest to highest would tend to perceive smooth intensity variations (i.e., any

transition from one level to the next would not be perceived as a jump or edge). Given a 4 bit grayscale, however, the observer would perceive jumps or discontinuities due to the level changes. The previously mentioned mathematical formulation assumes that the intensity function is a scalar quantity (i.e., it spans a one dimensional range of values ranging from binary to grayscale). If other features, such as colour, were extracted, f could be modelled as a vector valued function. Practical concerns related to the f(x) representation of an image are:

(1) f has a finite range of values

(2) x has a finite resolution.

(3) both f and x are discretely quantised in practice. The discrete nature of these quantities introduces a number of concerns, such as the effect of sampling.

Image data may be time varying in nature, as a result of time-varying behaviour of scene contents or the viewing geometry. The case of time-varying images is modelled by including an additional variable, time (t), in the continuous case or, in the discrete case, by indexing the image sequence by a discrete variable (e.g., kT). This yields a representation f($l(x,y),z$,t). The resulting four dimensional spatiotemporal model presents some conceptual limitations due to human inability to readily visualise entities in more than three dimensions.

1.4 Overview of the thesis

The three dimensional space, where objects are randomly positioned, with random shapes and varying sizes, presents the main difficulty in modelling an artificial system that could effectively simulate the human visual ability. The problem of successfully recognising and effectively classifying these objects

within cluttered environments makes the problem even more complicated.

Methods traditionally used for successfully identifying three dimensional objects, include the methods of moments invariants and the normalised Fourier descriptors. Although both techniques will be extensively presented and described in a following chapter of the thesis, it is noted that both methods have achieved highly successful classification performances and their computational requirements have been significantly low (Chapter 3). Although, the classification performance of the previously mentioned methods have been highly successful, both methods have failed to present the expected global solution to the problem of invariant three dimensional object recognition. The main reasons being : their requirements for a large amount of a priori information for a successful classification, their large (normally in the order of hundreds of pictures) library and testing picture sets, and their information loss. In addition, both methods and especially moments invariants, suffer from suppression and redundancy. For all the above limitations, moments invariants and normalised Fourier descriptors, although at times used for three dimensional object identification, have failed to produce the desirable global solution to the general three dimensional object recognition problem.

The resurgence in the use of image database systems for storing a large number of images and performing a variety of image processing and database operations, has revealed another possible source for the solution to the three dimensional invariant pattern recognition problem. Image database systems possess two fundamental properties that make them attractive for our purposes. They can store a large quantity of image data using a variety of image compression techniques, and they can offer an extensive "pool of information" that can be accessed and effectively used by many end-users. In order to investigate these and many additional properties of modern image databases, an experiment was devised that aimed to achieve an exact pattern match of a two dimensional template against all possible image patterns

stored within a standard relational database environment. For reasons previously mentioned, the images stored within the relational database environment were in a compressed form, thus allowing a larger number of pictures to be stored within the system. In the work, for simplicity, a set of fifteen different compressed aircraft pictures was used. In the context of the exact pattern matching operation in the relational database system, a remarkably consistent and successful performance was demonstrated. Exact pattern matching has always been successful with the compressed database image patterns always correctly associated with a target input template. Although highly successful, the experiment with the relational database system (Empress database [1]) demonstrated a very slow classification response. In fact the matching operation took on average 10 to 15 minutes for a simple 6 by 6 tuple pattern. This overall computational requirement was considered to be prohibitively low for a real time online application, and consequently any further research in more complex fuzzy matching operations within the standard relational database environment was abandoned.

Another area of investigation has been that of adapting a modern artificial neural network architecture in implementing a position and scale invariant pattern recognition system. Such architectures can effectively classify objects requiring a minimum computational time, normally in the range of seconds. Artificial neural network systems often offer a large storage capacity that allows a large number of objects to be usefully stored within their system structure. These objects can then be used to train and test the underlying neural networks. Consequently, all information can be accessed in the same efficient and globally available way as in a relational database system. Furthermore if the input to a neural network system is appropriately coded, large storage space savings can be achieved, through effective compression techniques and hence many more additional patterns can be used for training. Artificial neural network systems also provide a number of additional properties, most notably their generalisation abilities, and these

will be presented in detail in the following chapters of this thesis.

The thesis primarily describes a new algorithmic approach for position and scale invariant three dimensional object identification. The technique is based on the implementation of a new algorithmic variation of the method by Chien and Aggarwal [2]. The new technique requires three non-coplanar views of an object to be taken and aims to create a three dimensional volume presentation of that object by merging the three respective pseudo-volumes created from the three different two dimensional object pictures. Many researchers experimenting with three dimensional object recognition, have used actual object image data in order to train and test an artificial neural network system [3][4]. This thesis is unique in that it codes such information originally within the context of the normalised quadtree data structure, and then within the data structure of a normalised volume octree. The coded information is then used for training and testing the underlying artificial neural network architecture.

The neural network is thus trained or tested on entirely synthetic coded aircraft image data. As the type of data structures involved in the pre-processing (coding) phase are efficient in terms of image compression, average storage savings of 80% are automatically inherited by the underlying neural network architecture. Relevant research work [5] in the area of quadtrees and octrees has shown that compression ratios of 6:1 or 5:1 are normally expected in quadtree/octree applications on binary segmented image data. The present experimental work verified such performance statistics. The initial normalisation of the three quadtree data structures, guaranteed that the final object volume representation, that is described by the final octree data structure, would also be invariant to any positional and size changes of the object. Experimental results presented in latter chapters of the thesis strengthened the above claims and additionally demonstrated a successful neural network classification behaviour particularly within cluttered environments.

A secondary contribution to the knowledge, can be attributed to the fact that it involves experiments using the ADAM artificial neural network [6]. This neural network architecture has only been recently developed, and limited research work has actually taken place to justify its true potential with respect to generalisation, noise sensitivity and invariant properties. As the entire project is based in a variety of research areas, ranging from relational database systems, computer vision, and image processing to computer graphics, the need for a global graphical interface has become apparent. The thesis proposed such a graphical user-interface, pointing out its particular properties with respect to an easily accessed and accumulated "pool of information", that can combine the properties of an image database, image processing algorithms, displaying routines, training and testing pattern generation procedures, training and testing graphical environments for the artificial neural network system and finally graphical performance charts. All this information will be presented and accessed in the context of a unique graphical entity thus offering considerable time savings and reductions in the overall distributed number of different entities that have to be combined for such a specific pattern recognition process.

1.5 Structure of the thesis.

Chapter 2, involves the relevant work regarding relational database systems and pattern matching. The entire relational database design concept is presented, as well as the complete pattern matching procedure. A synoptic overview of image compression techniques is presented with an additional presentation of the elementary database concepts. Finally, graphical performance results are presented regarding the experiments with compressed database images. An overall discussion of the efficiency of the entire pattern matching procedure concludes the chapter.

Chapter 3, describes the problem of three dimensional object identification in the context of classical shape descriptors. Moments

invariants and normalised Fourier Descriptors, two shape classification methods commonly applied, are presented and their performance with respect to two dimensional and three dimensional invariant object recognition is presented. An overall evaluation of the performance of these techniques is presented in the context of three dimensional aircraft recognition and a characterisation of the classification efficiency of both techniques is shown.

Chapter 4, presents an overview of artificial neural network architectures. The presentation follows a chronological order in presenting the neural network systems and their respective classification efficiency as it has been shown in invariant object recognition applications. An overall discussion at the end of chapter 4, identifies the best architectures in the context of this thesis and compares the relative advantages of all possible neural network systems.

Chapter 5, presents the problem of invariant object recognition in the context of modern neural network architecture. It shows the possible classes of the problem and argues about the efficiency of current neural network systems in integrating classical shape descriptors for invariant pattern identification. Limitations and advantages are presented, as well as a first idea of the proposed algorithmic solution to the problem in the context intended for the present thesis.

Chapter 6, presents the algorithm that was devised for implementing the volume intersection technique. The chapter offers initially general information about the quadtree, normalised quadtree and linear quadtree data structures. It explains the method for modelling the normalised linear quadtree data structure and proceeds to present the generation of the octree data structure. Chapter6, explains the theory behind the volume intersection technique and shows how the three non-coplanar aircraft two dimensional images can be combined to formulate a normalised-, position- and scale invariant-, volumetric-object description. The procedure for formulating such

a volume is shown in the simple context of a simulated 8 by 8 pixel aircraft model. For reasons related to the original necessary image pre-processing, the beginning of chapter 6 describes in detail issues of current digital image processing. The presented areas of interest, include image enhancement methods, image smoothing by gaussian filtering, image thresholding, edge detection and general image segmentation techniques.

Chapter 7, offers a detailed presentation of the experimental results using the ADAM artificial neural network. The chapter then proceeds by presenting a more realistic, though still with simulated data, experiment in which three aircraft models of 8 by 8 by 8 octants are used for training ADAM and variations of them are used for testing. Chapter 7, then uses real aircraft data from a set of six aircraft in order to train and test ADAM. Simple aircraft skeleton pictures as well as filled boundary aircraft pictures are used for training and testing. Eight different experiments were undertaken, using various levels of noise either in the training set or in the testing set or in both sets. Performance classification results are presented and a detailed discussion of ADAM's generalisation ability is presented. Four further experiments are described in which a set of entirely random object views is used to formulate the testing set. Again a well documented analysis of the classification behaviour of ADAM is presented. Finally, chapter 7 deals with the possible case when object volumes cannot be effectively formulated. An alternative approach is presented whereby the quadtrees of the intermediate processing stage are used in order to train and test ADAM. Further, classification results are once again presented. The chapter concludes with a general discussion based on all the previously presented classification data.

Chapter 8, presents a hypermedia style user-interface capable of integrating in a single operational entity, all the various different sub-parts of the pattern recognition process. At first, an overview of the advantages that lead to the selection of a specific user-interface (the BookEmulator [7]) is presented, with emphasis placed on the particular benefits derived as a result

in thesis areas such as: pattern selection, image processing, Neural Network training and testing and finally performance data presentation. A brief discussion argues how the selected user-interface can assist in the unification of all the previously mentioned distinctly different thesis areas. The discussion additionally comments on the efficiency of creating communication links between the BookEmulator and other 'outside world' picture processing facilities. Time and processing constraints are discussed and the possible effectiveness of a locally developed image processing tool, the DrawingBoard [8], is evaluated. The chapter then continues by presenting the user-implementation process in four distinct phases. The first phase, deals with the problem of inputing data and proposes various possible ways in which a meaningful input phase can be modelled. This phase pays particular attention to what is called the "book to book communication" concept. The entire input phase is described in the context of this concept and ways of allowing communication between the main interaction book, called the NeuralBook, and other arbitrary books from various possible libraries, are presented. The second step involves the possible image pre-processing that can actually take place in order to formulate the appropriate patterns for the later training and testing of the underlying neural network mechanism. This phase, shows a library of possible image processing algorithms which can be used in that context, and investigates possible ways of providing interaction between aircraft data displayed in this section of the NeuralBook and data of an "outside world" picture processing facility. In this way many more training and testing patterns can be generated without having to re-iterate the entire image processing routines right from the start. The third part, involves the implementation of the specific invariant pattern recognition algorithm within the NeuralBook. The pseudo-volume selection technique, and the final synthetic octree volume implementation are presented. The fourth and final stage in the proposed user-interface design involves the creation of the training and testing phase of the ADAM neural system as well as the design of meaningful classification charts that could graphically illustrate the classification efficiency and overall behaviour of ADAM. In the end of

chapter 8, an evaluation of the proposed system is presented together with additional time performance statistics for the specific components of the invariant pattern classification process.

Chapter 9, is the conclusion of the thesis. The key features presented in the thesis are once again emphasised and further work is identified. Further comments regarding possible improvements to parts of the thesis, so that the problem is addressed in a rather more general view, are also mentioned.

CHAPTER 2

Pattern Matching using SQL Queries on Compressed Database Images

2.1 Introduction.

The idea of using databases to deal with images and text is not a new concept. Several papers have been presented that demonstrate query processing on multimedia data. [9][10]. This chapter considers the possibility of an efficient error-free and unambiguous pattern matching operation within a standard relational database system. A typical image of a simplified database system as it was viewed by Date [11] can be seen in Figure 2.1 (Date [11]).

The implementation of such an effective multimedia query processing environment integrated within a standard relational database model will be, if successful, an important contribution towards a general query processing and optimisation mechanism for modern relational image database systems. The system in mind should use compressed images and would be able to overcome the increasing problems of excessive storage utilisation resulting from the vast number of images needed to be stored within image databases. That will require the application of pattern matching on compressed images rather than on the original pictures. In addition to the storage limitations, the pattern matching should be unambiguous and independent of the orientation and distance of the object.

The following sections offer initially an introduction to the basic

principles of relational database systems and continue by presenting a synoptic overview of present image compression techniques that is necessary in understanding the particular difficulties of applying such methods within a relational database environment. The analysis of current image compression techniques will have two main objectives. First, targeting the best possible compression method that can allow not only a fast but also an error-free image-pattern processing and retrieval and second, an easy to implement, mathematically efficient data management and manipulation mechanism that can be formulated within a relational database system. Although the presentation of the basic concepts in the areas of relational database systems and digital image compression will have an informative nature, the introduction of such concepts will assist in understanding the motives that led to the selection of the specific compression technique for the purposes of the current application and will additionally highlight the benefits and important features of relational database systems.

This chapter will also present the experimental database schema used in the present application, the data attributes and the corresponding relationships, the application of compression methods on the stored images within the database and finally, the general query processing and optimisation methods. A final conclusion will discuss whether the method used has been efficient and will additionally comment on its advantages and possible limitations.

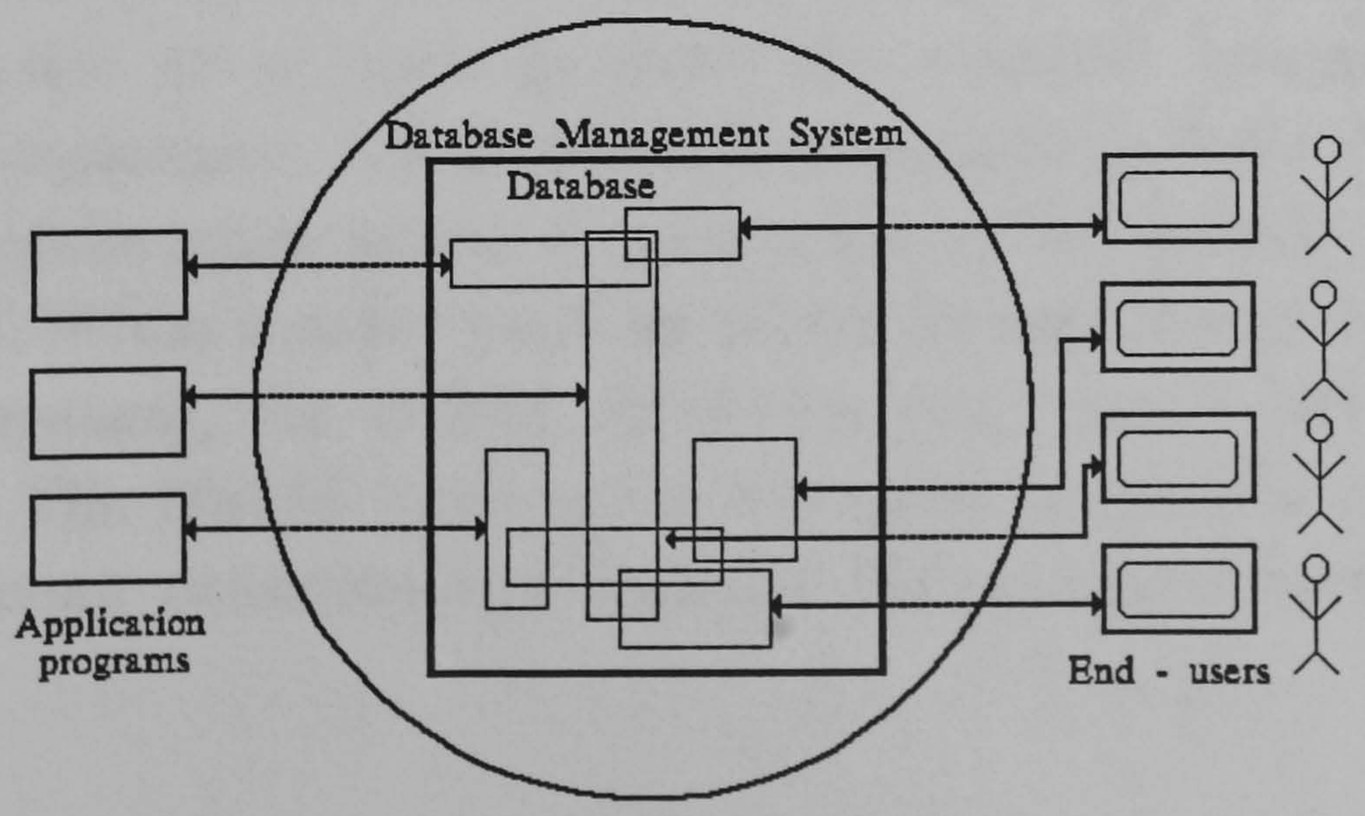


Figure 2.1 A simplified Database system.

2.2 Relational Database Systems.

Date [11], defines a database system as "a basically computerised record keeping system whose overall purpose is to maintain information and to make that information available on demand". A relational database system in particular, is an environment where data are perceived by the user as tables and where the operators at the user's disposal, (e.g., for data retrieval) are operators that generate new tables from old ones. A hierarchical database system in contrast to the relational environment is a system where data structures are used in addition to the tables required for the relational approach. A relational database system is characterised by a number of desirable properties; compactness, speed and concurrency are some of those described by Date [11]. From the notion of centralised control one can argue that with a relational database system, redundancy can be reduced, inconsistency can be avoided, the data can be shared, standards can be enforced, security restrictions can be applied and finally but most importantly integrity can be maintained and conflicting requirements can be balanced. All the above properties of relational database systems converge to show the importance of their nature, and were the main reasons for investigating their possible performance in the rather more demanding area of pattern matching on compressed binary segmented images.

The term "database system", normally refers to a software "management system" rather than to the hardware components. The previous point is important since a database alone is a collection of hardware components typically unable to perform by itself the essential operations they are designed to implement. The Database Management Systems (DBMS), is the heart of the entire environment. It is the software that handles all accesses to the database. When a user issues an access request (using some particular data sub-language), the DBMS intercepts that request and consequently analyses it. The DBMS inspects the external/conceptual mapping and the storage structure definition and executes the necessary operations on the

database. Another simplified way of viewing the function of the DBMS is that of the user-interface for the database. The user-interface is defined by Date [11] as a boundary in the system below which everything is invisible to the user and above which the user's view of the data lies. [Figure 2.2].

A relational database is based on the perception of a real world that consists of a set of basic objects called entities and relationships between these objects. Entities are normally represented by a set of attributes. An attribute is a function that maps an entity into a domain while a relationship is an association among several entities. As a vehicle for illustrating such relational concepts, a type of language is used to allow communication between users and a relational database system. SQL [12], is the standard language used for such purposes. SQL includes a number of operations that the user can apply in the form of commands to allow interaction between the database tables. In fact, three operations: selection, projection and natural join, that require no pre-definition of physical access paths to support their operation, must be supported by any relational database system. Selection, often described by the Greek symbol Σ, is an operation that applies on the lines of database table, selecting sets of rows from tables that satisfy its restrictions (i.e., people born in the United Kingdom from a table with personal data of people living in London). Projections on the other hand, operate on the columns of the database table isolating only the columns with data that users require from a set of selected database rows. Projections are symbolised by the Greek letter Π. Natural join is used to combine data from a number of different database tables and consequently create new tables. The last operation is essential in every relational database system. One can simply understand its significance by imagining a number of tables with thousands of rows. Merging information from different tables in this case will create a vast number of entries in the new resulting table. Evidently storage considerations, duplication of information and data accessing times must be considered. Consequently, joins must be efficient and a whole theory has

been introduced that deals with mechanisms for minimising row entries in database tables that result from join operations. The fact that a given system is relational according to the definition previously given, does not itself guarantee that the system in question is a 'good system'. As Date [11] noted, 'however the relational facilities do provide a good and solid basis on which to build the necessary features that go to make up such a good system'.

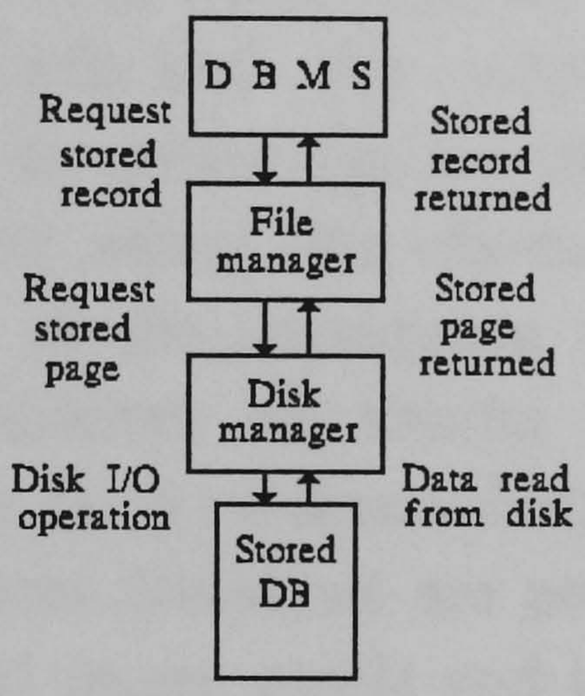


Figure 2.2 The DBMS, file and disk managers.

2.3 Image Compression Techniques.

The basic idea of image compression is to be able to represent the information in an image with fewer bits than required for storing the original and at the same time be able to reconstruct the image so that it is close or equal to the original picture. In mathematical terms, the essence in image compression techniques is the use of an invertible linear transform that can transfer the given correlated image array to an array of uncorrelated variables that can be represented by a fewer number of bits compared to the original image array. Encoding an image can be modelled as a sequence of three operations. The mapping operation that maps the input data from the pixel domain into another domain where the quantiser and the coder can be effectively used. Consequently fewer bits are required to code the mapped data to one of a smaller number of possible values so that fewer code words

with fewer bits are required. The coder in turn assigns one code word to each quantiser output.

Two classes of image compression techniques are known: spatial coding and transform coding. In the second type of coding, a linear invertible two dimensional discrete transform is applied to the given image. A subset of the transformed coefficients is retained, quantised and then stored. When it is required to generate an image (that will be close to the original), the quantised transformed coefficients are reconstructed followed by the application of the inverse transform. In the transform coding scheme two basic problems need to be solved, the choice of the discrete correlating transform and the choice of the appropriate transform coefficients to be retained. Spatial coding schemes are similar in that the basic idea is to generate an array of uncorrelated random variables from the given image by using an invertible transform. Residuals are generated so that they are less correlated than the original image pixels and hence can be represented by much fewer bits. Given the quantised residuals and parameters of the model, an image close to the original can be generated. Mean-square error between the original and the reconstructed image is usually used as a criterion for measuring the quality of the reconstructed picture. The best invertible transform is the one that produces uncorrelated transform coefficients. The Karhunen-Loeve linear orthonormal transform (KL) [13] is such a transform for finite image arrays. All other transforms from the same category like the Fourier transform[14], the Discrete Cosine transform [15] and the Hadamard transform [16], may be computationally efficient but can not achieve the optimal behaviour of the Karhunen-Loeve transform [10]. The KL transform basically minimises the ensemble mean-square error between the original and the reconstructed image by using n transform coefficients [17]. The serious computational problems involving the computation of eigenvectors of a M^2 by M^2 covariance matrix of an M by M image is considered to be the major weakness of the KL transform. The Discrete Cosine transform (DS), is an

approximate substitute for the KL transform in that the resulting coefficients are not exactly uncorrelated. Another useful technique is the Singular Values coding (SVD) [18]. This is a deterministic technique that is optimal in the least square sense, while KL transform is optimal in a statistical mean square sense. The Block Truncation coding [19], is another image data compression technique. This algorithm divides each image in 4 by 4 blocks and codes them independently. Each block is represented by its quantised representations of the sample mean, standard derivation and a 4 by 4 bit plane. This bit plane denotes whether pixels are below or above the threshold of a 1 bit moment processing non parametric quantiser [20]. Applications of data compression deal with storage and transmission of information. In transmission applications real time considerations as well as the complexity of hardware constrain the compression techniques. In storage applications the pre-processing that can be done off line results in less stringent requirements. Several additional image compression techniques have been examined, among them the Huffman coding [21], Quadtrees [22][23][24], and finally the Run Length Encoding technique [25].

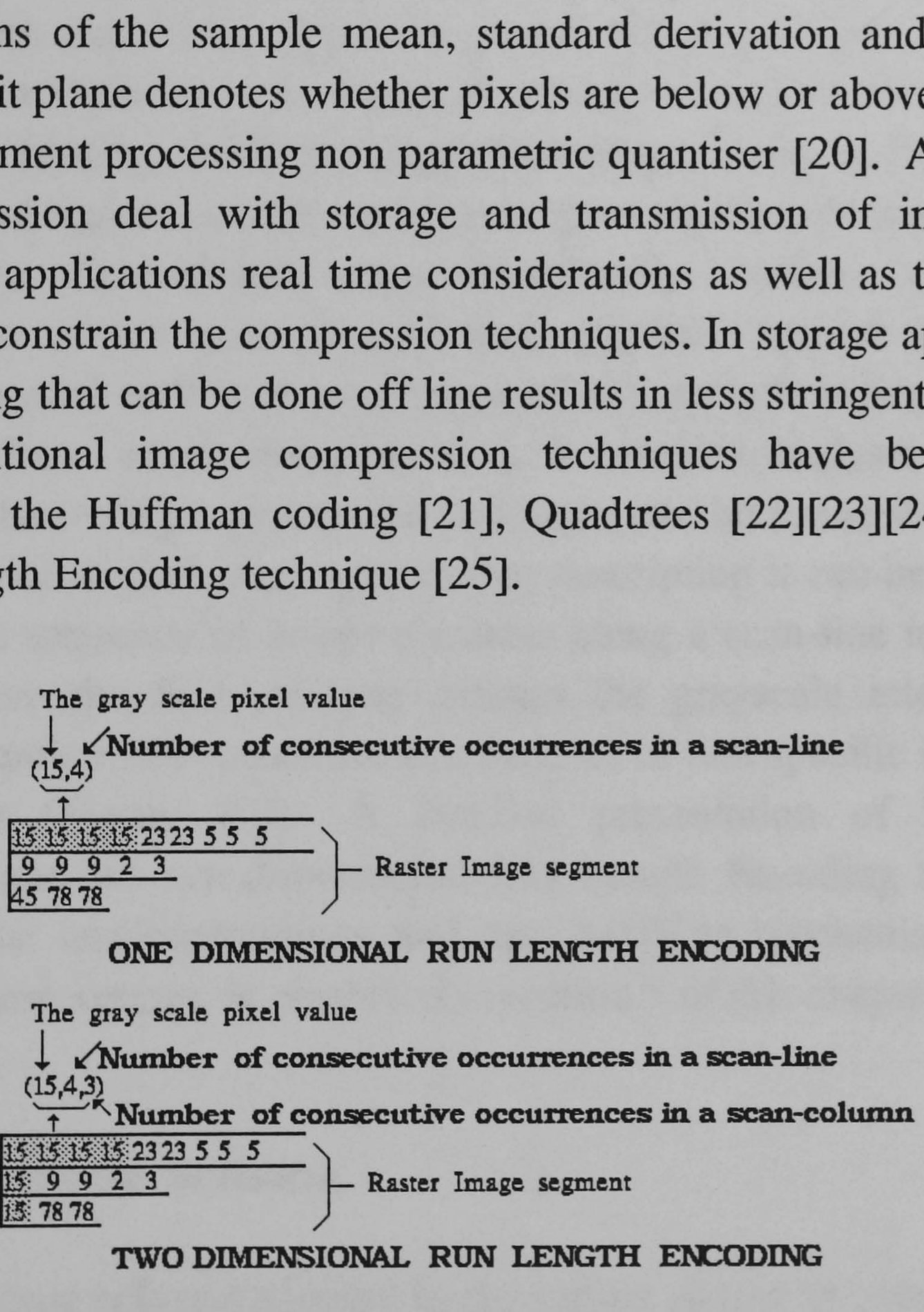


Figure 2.3 The Run Length Encoding Method.

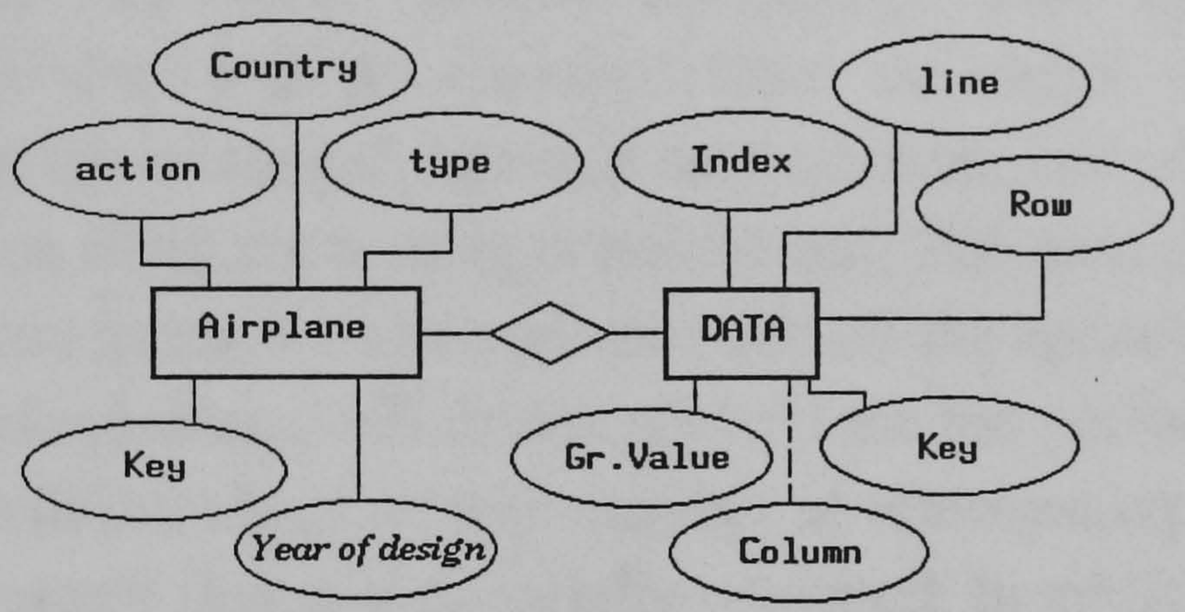
In the current application, emphasis was placed in two particular areas of

performance, the error-free reconstruction and the portability of the results within the relational database system. Huffman coding though a generally applied and well known method failed to produce compression rates of more than 70%, due mainly to the large size of the raster images used and the corresponding depth of the trees that the algorithm produced. The bit strings resulting from Huffman coding were long and an insignificant storage saving of only 40% was achieved. Quadtrees on the other hand offered storage saving of more than 70%, (typically around 80%), but their respective data structures were difficult to incorporate within a standard relational database environment. Finally, the Run Length Encoding technique, that is simple to implement and requires no tree structures, gave satisfactory results in both its one and two dimensional cases. Specifically, because of the spatial characteristics of the raster images used (aircrafts against sky), the latter method performed well with an average of 77% in both the one and the two dimensional Run Length Encoding case. A complex analysis of the general Run Length Encoding compression technique can be found in [26], but in a rather simplified digital image processing description it can be viewed as the mapping of a sequence of image elements along a scan-line into a sequence of pairs where the first attribute denotes the grayscale intensity and the second the number of consecutive occurrences of that specific intensity in the scanned line [Figure 2.3]. A detailed presentation of both the one dimensional and the two dimensional Run Length Encoding techniques and their particular implementations and data handling mechanisms within the current database system, is presented in section 5 of this chapter.

2.4 The Database Schema.

The database schema adopted in the current design [Figure 2.4], consists of two main entities. The Airplane entity that contains all attributes regarding the aircraft's technical profile, and the Data entity that contains the actual raster image data values and a number of indexes that indicate the

exact position of a specific pixel in the image plane. The general concept behind the design of the two different tables, is that information regarding actual raster image data during pattern matching will be required more frequently than will data such as an aircraft's name and the initial year of its design. Separating these, results in considerable storage savings as well as causing a significant reduction in data accessing time. In addition, information from both tables can be combined if required, by using natural join operations. Both storage minimisation and query processing time will be thoroughly examined in later parts of this chapter.



Country	☞ *The country where originally designed*
Type	☞ *Bomber, fighter, passenger carrier, etc.*
Action	☞ *Comments on successful use.*
Year of design	☞ *As stated*
Key	☞ *The indexing mechanism*

Index	☞ *Position of tuple in an image file line*
Line	☞ *Number of consecutive line pixels*
Row	☞ *Which line in an image file*
Column	☞ *Number of consecutive column pixels*
Gr. Value	☞ *The image pixel grey scale intensity*

Figure 2.4 The General Database schema.

Information from both database tables can be linked in practice using a special key attribute in both entity schemes. The primary key, represented in the current database design by an integer, is used to identify (index) each

individual aircraft. In this way it helps to distinguish aircraft data that belong to different raster pictures whose actual information reside within the same database table. The primary key attribute is the main mechanism for indexing tuple entries of the same aircraft within the database. Consequently all compressed image data entries in the relational database will contain that additional attribute, causing an undesirable information repetition along tuples of the same aircraft and resulting in increased storage requirements for the overall relational database system.

An alternative method avoids the disadvantage mentioned above by processing each individual aircraft separately. This implies that each aircraft's picture data will be retrieved from an image store, sequentially processed within the relational database environment, and will be placed back in the image store when processing is completed. The next image will then be retrieved and processing will be repeated for all the remaining aircraft data. This accessing mechanism will drastically reduce the performance of pattern matching as it will introduce a large number of unnecessary disk accesses for each different aircraft that is sequentially retrieved. In addition, users will not be able to benefit from key database facilities such as file locking and data integrity. The use of indexing methods for distinguishing data within a relational database environment, matches the general concept in database systems where primary and secondary keys are often used to allow effective query processing and computationally efficient accessing times.

To reduce the computational workload during query processing, help tables were introduced in the current relational database design. These tables provided a useful intermediate processing level for searching within the database, which in fact reduced the overall pattern matching object space. Help tables were normally small in size and accumulated data from time consuming join operations between database tables. Being small in size, help tables were easy to access and searching in them was generally faster and more efficient than in other sizeable database tables.

The aircraft data included in the relational database system resulted from compressing raster images using the Run Length Encoding compression technique. The simplicity of its output data format and the minimum computational requirements, make Run Length Encoding the best method for this thesis image compression task. Both the one dimensional and the two dimensional cases were investigated and the performance results when these methods were applied on raster images will be presented on the following pages.

2.5 The Matching Method.

2.5.1 The one dimensional Run Length Encoding case.

IMAGE PART

t g g l k
u g g j o
t l g i i
i g g j i
i l i e i

Figure 2.5

1-D Run Length Encoding

[t,?] [g,2] [l,1] [k,?]
[u,?] [g,2] [j,1] [o,1]
[t,?] [l,1] [g,1] [i,?]
[i,?] [g,2] [j,1] [i,?]
[i,?] [l,1] [i,1] [e,1] [i,?]

Figure 2.6

Consider Figure 2.5 as being the part of an image to be matched against the compressed pictures stored in the database. Figure 2.6 represents the one dimensional Run Length Encoding representation of the pattern. Tuple [t,?] in the first line is called the front tuple while tuple [k,?] is the rear tuple. All other tuples between them are called inner tuples. The symbol ? denotes that the gray scale value t does not necessarily appear once in the first line of the pattern but can be preceded by a number of other t's. Only the inner tuples can match exactly the format of any compressed image (for example, the number of gray scale values specified by the second value in each tuple reflects exactly the number of times this specific gray scale value has to appear in every image line). An important characteristic of the one dimensional Run Length Encoding case is the fact that by finding the position of the first inner tuple in each pattern line, one can automatically limit the indexes for all other tuples in each compressed pattern line, thus reducing insertions in tables and consequently the overall search time. The reason for the previous statement is the fact that in any compressed image pattern line (representing a line in a selected neighbourhood of the original picture), the same-valued pixels either in the beginning of the line or at the end, will often correspond to picture areas around the edge of the selected neighbourhood. It is therefore the lack of knowledge regarding the exact number of same-valued pixels either preceding the front pixel or following the last neighbourhood pixel that is responsible for the fact that the second index number within a compressed Run Length Encoding tuple is not accurate. For the same-valued grayscale pixels that are included between these front and rear pixels the second index number within the compressed Run Length Encoding representation is always accurate, since the number of same-valued pixels between the edges of a selected neighbourhood is known. For example, consider the fourth line of the image neighbourhood shown in Figure 2.5. Clearly, there is no knowledge of how many i's either precede the initial i or follow the last i. Hence, the second index number of the compressed Run Length Encoding tuple for the first and the last i is unknown. The opposite rule applies for the inner same-valued grayscale pixels that can simply be

seen to be two g's and one j. Accordingly, their second index number in their compressed Run Length Encoding representation is 2 and 1 respectively. Using the fact that the inner pixels are always accurately represented within the relational database it can be seen that this knowledge can be effectively used to enhance the overall query processing time and consequently optimise the entire pattern matching process.

In order to formulate the complete one dimensional Run Length Encoding pattern matching process the following steps need to be taken. First, find all lines within each database compressed image where the first inner tuple of the matching pattern occurs. Second, use the set of lines resulting from the first step and repeat the process for all inner tuples of that line. It is important that as the processing of all inner pixels progresses sequentially from the left to the right in the compressed pattern line, the lines resulting for each group of same-valued inner pixels (indicating in which image line this group of pixels was exactly found) must be included in the line set of the initial inner pixel group. Third, use the tuple position (index) attribute to eliminate the lines where the inner tuples do not appear in the sequence they should. Fourth, use the tuple index and the set of the resulting lines from the previous step to determine which of the lines include the front and the rear tuples of the current line. Fifth, save the resulting line set in a temporary table with the index of the compressed image pattern line that has just been processed and repeat the whole process for all the image pattern lines. Finally, use the indexes of the temporary table to find the sequence of lines that each have the same pattern [Figure 2.7].

2.5.2 The two dimensional Run Length Encoding case.

The fact that runs of same-valued pixels must occur in both lines and columns makes the two dimensional Run Length Encoding technique [Figure 2.8], an unattractive method to apply on raster pictures. The fact that pictures taken with a camera contain large amounts of noise, makes large runs most

unlikely. The use of graphics generated images though, will allow the effective application of two dimensional Run Length Encoding and thus offer a reason for investigating the effect of a pattern matching process within a relational database environment.

Image Pattern Lines	*Compressed Data Base Image Lines*
1st	*1,3,4,5,7*
2nd	*1,4,6,7,10*
3rd	*1,5,7,23*
4th	*1,6,8,10,11,13*

The Image Pattern was found starting at line 3 and finishing at line 6, and also starting at line 7 and finishing at line 10, as indicated by the different shading on the numbers.

Figure 2.7

The process is similar to the one dimensional Run Length Encoding case with the exception that tuples in the compressed database images as well as in the compressed target pattern segments, will include a third attribute that will describe the number of times a run appears in subsequent image columns. In order to formulate the complete two dimensional Run Length Encoding pattern matching process the following steps must be taken. First, find all lines within each database compressed image where the first inner tuple of the matching pattern occurs. Second, use the column attribute in order to create the actual line set where the first inner tuple exists. Assume for example that tuple [8,4,3] in Figure 2.8, exists in row 123 within a compressed database image. This will result in a set of rows {123, 124, 125} being formed as it is denoted by the third attribute in the [8,4,3] tuple (i.e, 3). Clearly what tuple [8,4,3] denotes is that a run of 4 identically valued pixels (with gray scale intensity 8) is repeated in 3 successive lines. Third, use the set of lines resulting from the first step and repeat the process for all inner tuples of that line using the constraint that all lines resulting from the repetition must be included in the line set of the first step. Fourth, use the

tuple index attribute to eliminate the lines where the inner tuples do not appear in the appropriate sequence. Fifth, use the tuple index and the set of the resulting lines from the previous step to determine which of the lines include the front and the rear tuples of the current line. Sixth, save the resulting line set in a temporary table with the index of the compressed image pattern line that has just been processed and repeat the whole procedure for all the image pattern lines. Finally, use the indexes of the temporary table to find out the sequence of lines where the entire pattern occurred.

The original pattern.

77788883332211
77888883322221
77888883322211

The two dimensional Run Length Encoding.

(7,2,3) (7,1,1) (8,4,3) (3,2,3) (3,1,1) (2,2,3) (1,1,1) (1,1,3)
" (8,1,2) " " " " (2,1,1) "
" " " " " " (1,1,1) "

where " denotes an empty tuple.
in every (a, b, c) [a] represents the gray scale value, [b] represents the number of pixels in the horizontal direction, and [c] represents the number of pixels in the vertical direction.

Figure 2.8 Two dimensional Run Length Encoding.

2.6 Database Performance.

2.6.1 One dimensional Run Length Encoding on edge detected aircraft images.

The images used in this experiment are edge detected pictures obtained when a number of different edge operators were applied individually on each aircraft picture (Sobel [27], Frei-Chen edge masks [28], Gradient [29]). In Figures 2.9 to 2.12, table histograms are presented displaying: the compression in bytes for the best ten images within the database [Figure 2.9], the overall time to insert each compressed image in the database according to

its size and compression rate [Figure 2.10], the compression in term of percentage for the best ten images within the database [Figure 2.11] and the compression percentage outside the database for all one dimensional Run Length Encoding images [Figure 2.12]. Looking at these results, the one dimensional Run Length Encoding gives an average of 77% compression outside the database and 55% within the database while requiring an average of 24.6 seconds to compress and insert tuples within the database. The above figures contain additional information about the compression methods that give the best results for each individual image. Figure 2.13 displays the performance during the initialisation and updating the search support tables. That particular schema is essential because it shows that processing N tuples per line is, in the worst case, a linear function of the number of tuples, for any number of tuples greater than two. Figure 2.14 verifies these observations for the worst case performance of the algorithm. It shows that searching for a number of different line size patterns is a linear function of the number of lines, with a constant coefficient A that is equal to 165. The pattern regions used for searching have a length of 51 elements and the number of tuples in each line varied from 4 to 9.

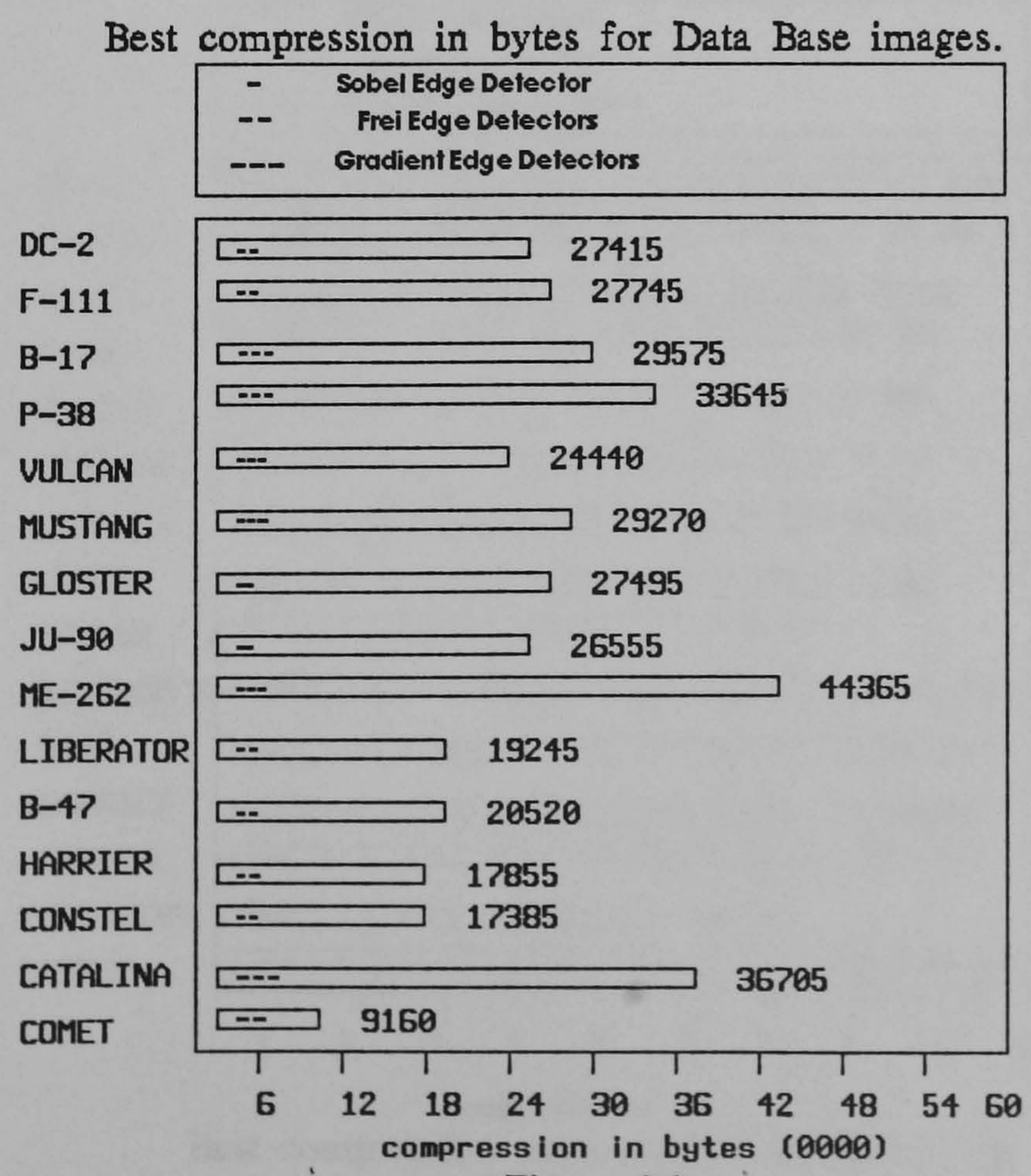


Figure 2.9

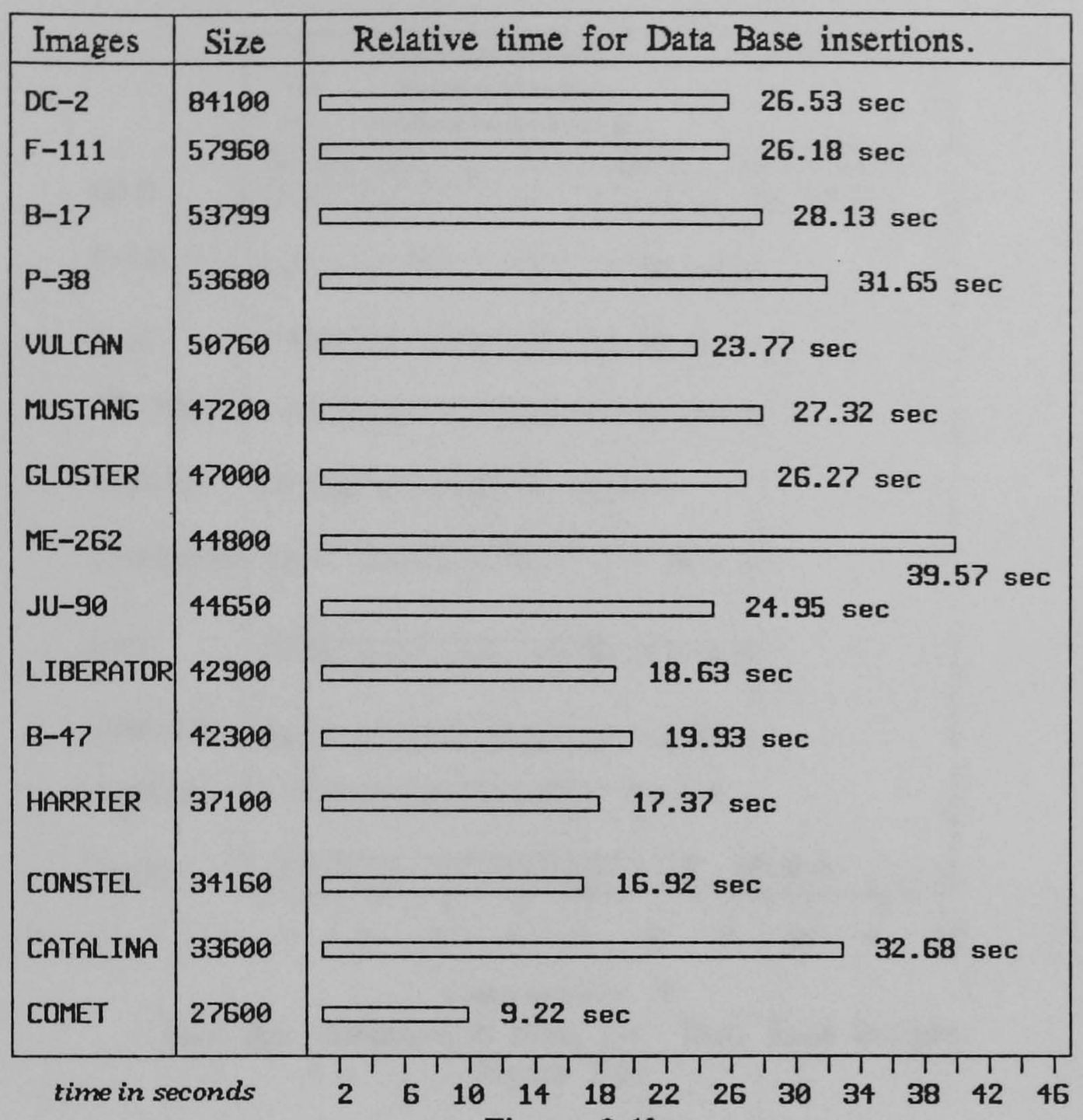
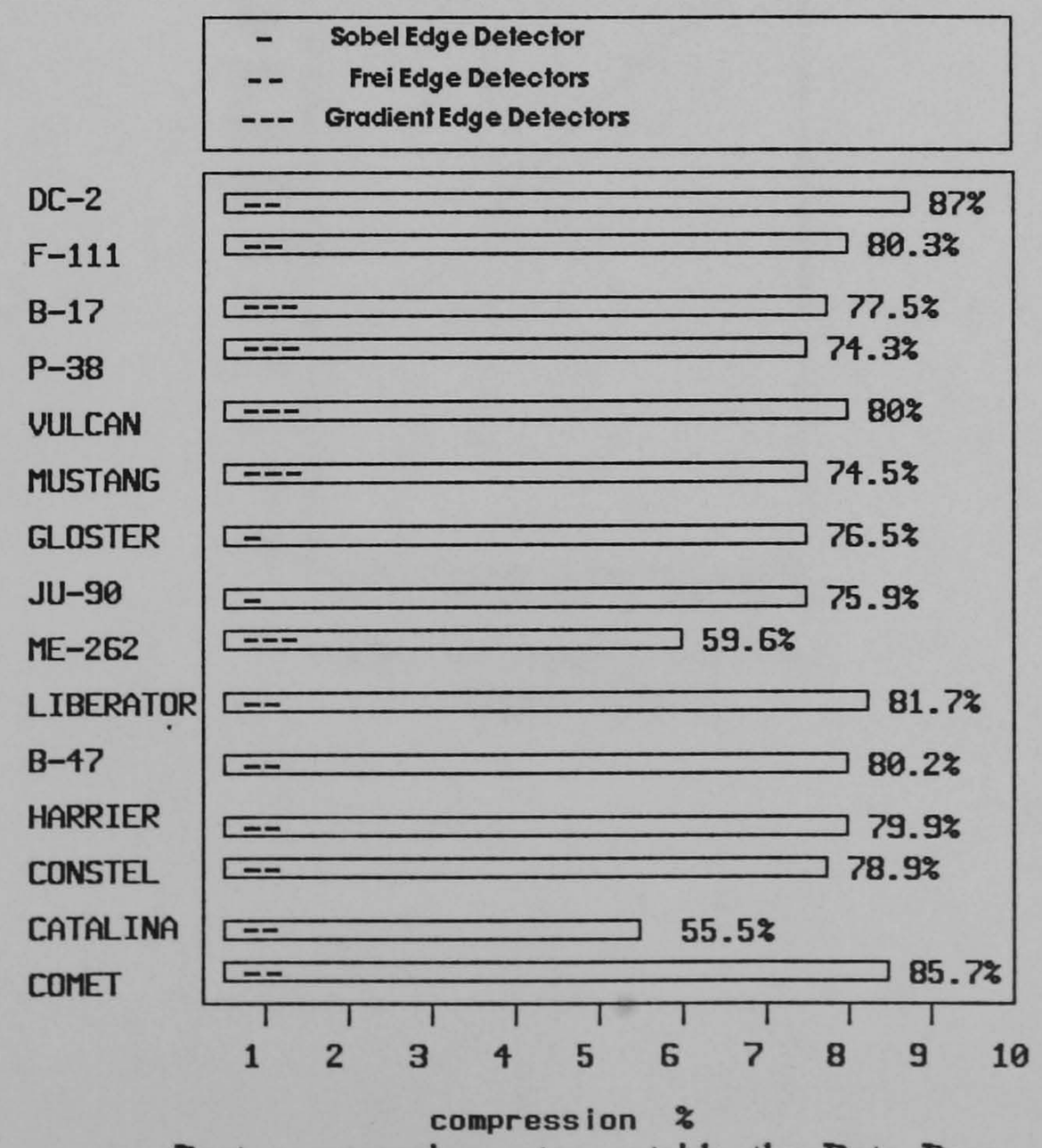
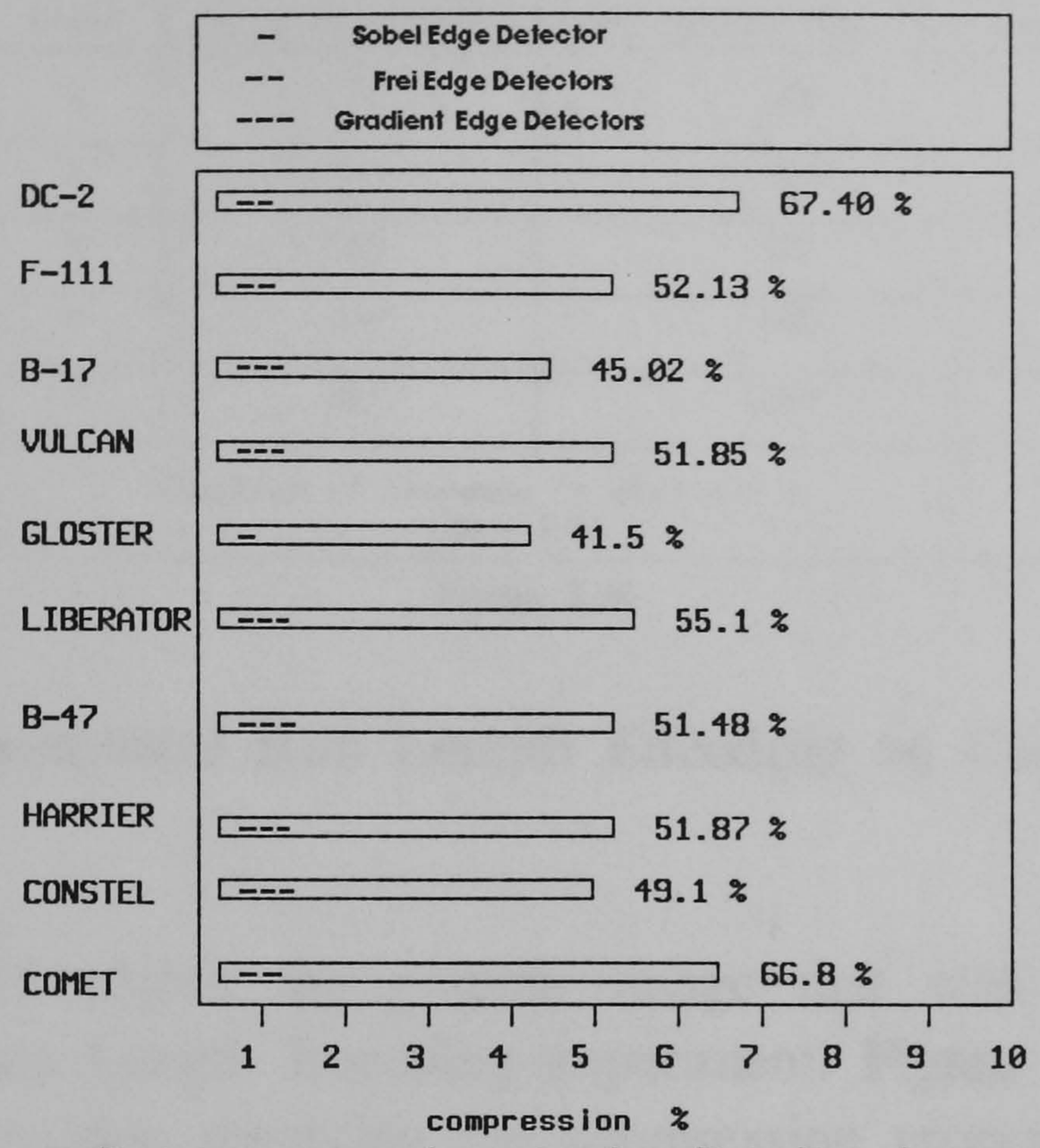


Figure 2.10



Best compression rates outside the Data Base.

Figure 2.11



Best ten compression rates for Data Base images.
Figure 2.12

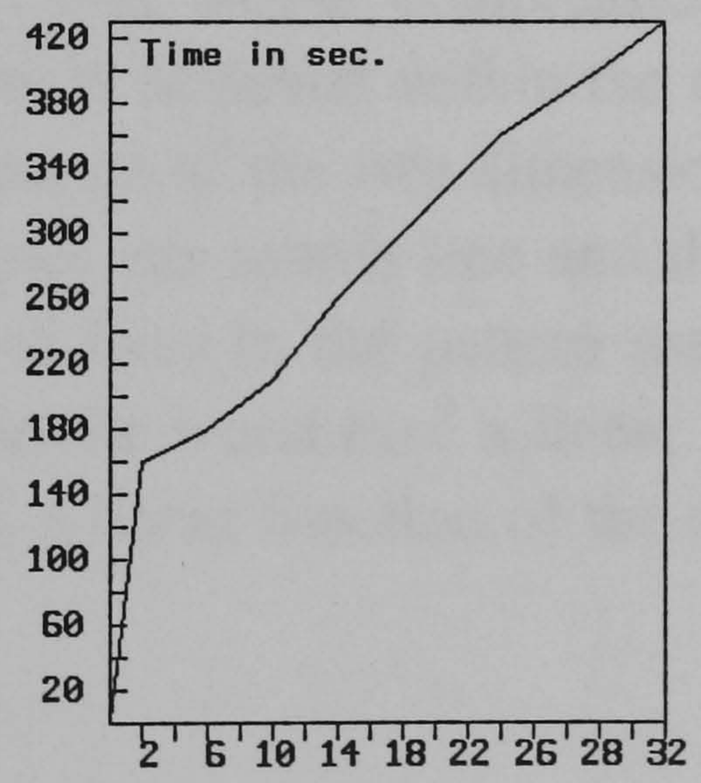


Figure 2.13

Lines	Per line elements	Total search time in seconds
1	51	165
2	102	327
3	153	495
4	204	665
7	357	1153
Function of increase ~ y(x) = A x A ~ 165		

Figure 2.14

2.6.2 Two dimensional Run Length Encoding on Computer generated images.

Figure 2.15 shows the graphic image that was used in the two dimensional Run Length Encoding experiment. Figure 2.16 gives all the statistical information regarding the compression process for images both within and outside the database, as well as the number of tuples inserted and the total time required to compress and insert these tuples within the database. It can be seen that 88.7% compression is achieved outside the database while a 70% rate is achieved within the database. Figures 2.17 and 2.18 describe the performance of the two dimensional Run Length Encoding algorithm in terms of tuples per search line and the overall search time that depends on the number of lines in the pattern used for matching. It can be seen that search time is in the worst case a linear function of the number of tuples per line as well as a linear function of the number of lines the search pattern contains.

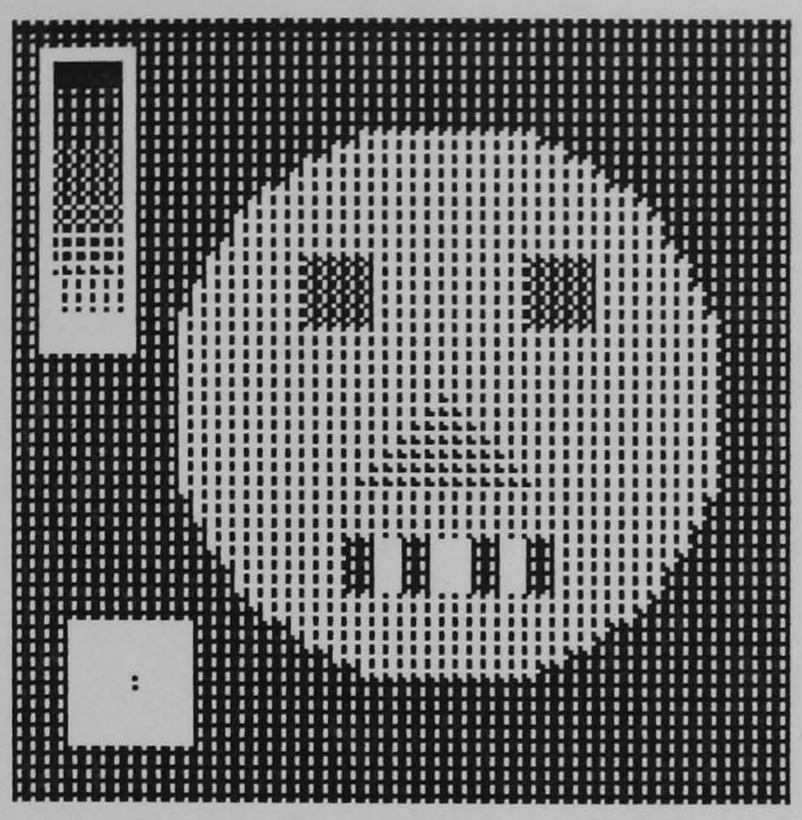


Figure 2.15

Two dimensional Run Length Encoding Statistic
Original image size in bytes > 4096
Bytes saved after One dimensional RLE > 2836
Percentage Compression after one dimensional Run Length Encoding > 69.2%
Bytes saved after Two dimensional RLE > 3634
Percentage Compression after two dimensional Run Length Encoding > 88.7%
Time to compress and insert into the DB > 2.07 sec.
Number of tuples inserted in Data Base > 154
Percentage Compression inside the Database > 77.44%

Figure 2.16

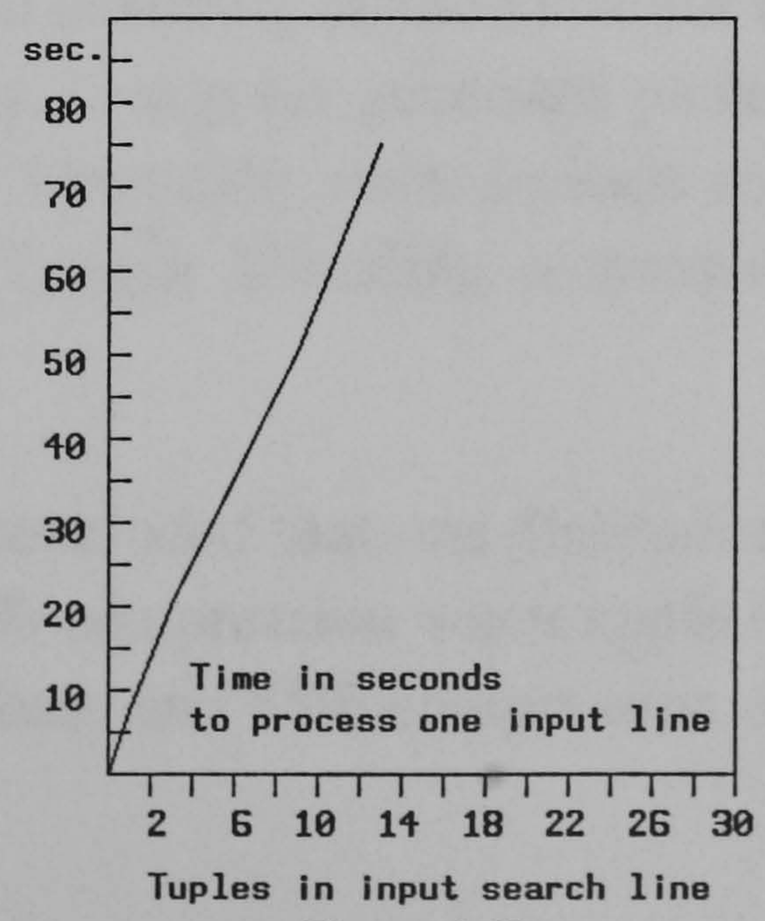


Figure 2.17

Two dimensional RLE : General searching statistics		
Number of lines	Number of tuples	Time in seconds
1	9	52.47 sec.
2	12	73.2 sec.
3	15	95.2 sec.
4	20	180.4 sec.
Function of increase Y(x) = A x $A \sim 165$		

Figure 2.18

2.7 Conclusions.

To conclude, the one dimensional Run Length Encoding technique offers two fundamental properties that simplify its application to raster images. First, all tuples in a given database stored image row should occur in a format (order) similar to that present in the lines of the image pattern used for pattern matching. Second, the index of the first inner tuple within a searching pattern can be used to minimise the overall size of the query processing tables and reduce its total computational requirements. In contrast to the above, the two dimensional Run Length Encoding method is seriously restricted by the fact that it requires a consecutive run of identically valued pixels to appear both in a line as well as in a column. Clearly, noise considerations during the object's image acquisition process, restrict large pixel runs from occurring in both lines and columns. The latter fact indicates that the two dimensional encoding method can not be 'in general' effectively applied on raster images. Computer generated pictures though, that contain a constant background of identically valued pixels are good candidates for the two dimensional Run Length Encoding technique and often compression rates.

In general, it was concluded that one dimensional Run Length Encoding gives an average of 77% compression when applied to aircraft edge detected images outside the database and 55% compression once they are inserted into

the database. It takes an average of 24.6 seconds to compress an image and insert the resulting tuples into the database. On the other hand, two dimensional Run Length Encoding resulted in 88.7% compression outside the database and 77.44% within the database when the method was applied on computer generated pictures. The total time to compress and insert tuples into the database was recorded to be 2.07 seconds. All performance times were obtained using a Sun 3/50 and pattern matching was achieved within the Empress database relational environment. The implementation has shown that searching time was, in the worst case, a linear function of the number of tuples present in each search line as well as a linear function of the number of lines the search pattern contained. It should be pointed out that all time measurements depended both on the original image size, in bytes, as well as on the rate at which the overall compression was achieved. It is evident that images with comparatively low compression rate, require a significantly longer time for searching, since their low compression ratio results in many additional tuple insertions and consequently more time to search amongst them.

The results previously presented prove, in general, the weakness of the standard relational database system in pattern matching. Although highly successful, the experiment with the relational database system demonstrated a very slow classification response. In fact the matching operation took on average 10 to 15 minutes for a simple 6 by 6 tuple pattern. This overall computational requirement was considered to be prohibitively low for a real time on-line application, and consequently any further research in more complex fuzzy matching operations within the standard relational database environment was abandoned. As mentioned, an alternative approach is presented in this thesis, where the database is only used to perform queries related to the aircraft's technical profile data (i.e., the name of an aircraft) and the actual pattern matching is achieved through an effective use of an artificial neural network system. It is proposed that the entire system will be controlled by a hypermedia style user-interface, with which users can simply

use standard relational SQL format database commands in order to interact with the relational database environment and at the same time can additionally select a variety of image patterns to teach or train the proposed underlying artificial neural network architecture. This proposed environment can guarantee the required efficient and computationally effective real time pattern matching performance required an on-line recognition system.

CHAPTER 3

Invariant Shape Analysis of 3D Objects

3.1 Introduction.

Shape analysis has long been an area of continuous research in digital image processing. Clearly much information is associated with the shapes of objects in images, and automatic image recognition systems require suitable algorithms in order to extract and analyse such information. A number of approaches to the shape analysis problem have been developed in recent years with most important these described by Pavlidis [30] in his useful overview of shape analysis methods. This chapter will describe the two most widely acknowledged two dimensional and three dimensional invariant shape analysis methods. The mathematical models of both methods (moments invariants [44] and normalised Fourier descriptors [42]) will be initially presented and emphasis will then be placed in examples of invariant aircraft classification where both techniques have been successfully applied. The chapter will be concluded with a discussion of the relative advantages and limitations of moments invariants and normalised Fourier descriptors as these are displayed in their classification performance through the presented test cases. It will also be argued whether the two methods can successfully be applied in a general invariant three dimensional object identification problem

where noise insensitivity and positional changes can significantly effect the outcome of the classification process.

The fact that human ability to interpret visual information might be effectively duplicated within a machine, has always been found intriguing by scientists. Although this type of problem have been tackled with considerable success relative to the automatic classification of two dimensional objects, there has been comparatively little research towards the automatic identification of three dimensional objects [31]. In this chapter, as well as in the entire thesis, particular emphasis is placed on the presentation of techniques that have been successfully applied to the position and scale invariant classification of aircraft. Recently, a survey of complex three dimensional object identification methods has shown that many researchers are especially interested in the particular subcase of invariant aircraft identification [32]-[37]. Research on this particular area has been significantly motivated by the need for automatic techniques in recognising different aircraft types in an air traffic control situation. Additionally, the variety of different types of aircraft represents a set of three dimensional objects with a varying degree of complexity and therefore can effectively serve as a useful and appropriate example for a general study of the position and scale invariant pattern recognition problem. A detailed presentation of the various methods that have been applied to the identification of complex three dimensional objects can be found in [38].

The main aim of shape analysis is to identify a pattern as belonging to a particular class of objects by means of matching a set of shape features. In the general shape analysis problem, a set of features that contain global shape information for the entire object represents the pattern. In many such applications the exact position or orientation of the viewed object may not be known. In such cases it is important to obtain shape features that are invariant to as many as possible object locations, orientations and sizes. Additionally, the fact that object illumination may not be uniform should also be

considered.

A set of moments is a possible valid candidate for a set of global shape descriptors. The advantage of the set of moments is that operations such as translation, rotation and changes in size are simple and closed on a complete moment set [39]. These same operations performed in the pixel or voxel domain are difficult, computationally expensive, and often lead to an approximate result since they are discrete sampled domains. Another popular shape description technique that has the same properties as moment invariants (simple and closed operations such as rotation, translation and scaling), involves the normalised Fourier descriptors [38]-[42]. These features, as well as the moments, have proven to be effective in tackling both the two dimensional and three dimensional problems [43].

3.2 The method of moment invariants.

3.2.1 Introduction.

Moments and functions of moments have been used as shape features in a number of applications with the aim of achieving invariant recognition of two dimensional and three dimensional image patterns [44]-[50][62]. Hu [44] has been the first to demonstrate the efficiency of moment invariants in invariant two dimensional pattern classification (1961). Using methods of regular (geometric) moments, he derived a set of invariant moments that has the desirable properties of being invariant under image translation, scaling, and rotation. The question though, of what was exactly achieved by including higher order moments in the context of image analysis has not been efficiently addressed, and the recovery of the image from these moments generally seems to be difficult [44]. Teague [51] has suggested the notion of orthogonal moments to recover the image from moments based on the theory of orthogonal polynomials, and has additionally introduced the Zernike moments, that allow independent moment invariants to be constructed easily

to an arbitrarily high order. Other orthogonal moments are Legendre moments, making use of Legendre polynomials. In [52], rotational moments are used to extend the definition of moment invariants to arbitrary order in a way that ensures that their magnitudes do not decrease significantly with increasing order. More recently, the notion of complex moments [53], [54], has been introduced as a simple and straightforward way of deriving moment invariants. The definitions of various types of moments and a summary of their properties can be found in [55].

3.2.2 The moments feature vectors.

The conventional definition of the two dimensional moment of order (p+q) of a function f(x,y) is

$$M_{pq} = \int_{-\infty}^{\infty} \int_{\infty}^{\infty} x^p \, y^q \, f(x, y) \, dx \, dy \qquad p, q = 0, 1, 2, \ldots \qquad (I)$$

The above equation shows an image function that can be seen as being represented by an infinite set of transform coefficients M_{pq}, that are derived from projecting the image onto a set of two dimensional polynomial basis functions. A set of moment values may be used to represent a segment of an image. Operations regarding positional and size changes can be more efficiently achieved in the moment domain than in the original pixel domain [44]. Furthermore, a truncated set of moments may offer a more convenient and economical representation of the essential shape characteristics of an image segment than a pixel based representation. Practical application of the transform though, requires that a finite number of transform coefficients be employed. In the case of a spatially discretised L by N pixel image, this is denoted by the function f(i,j). In the "real world" discrete case the equation (I) may be formulated using an approximation of double summations:

$$M_{pq} = \sum_{i=0}^{L} \sum_{j=0}^{N} i^p \, j^q \, f(i,j) \qquad p,q = 0,1,2,\ldots \qquad (II)$$

where i and j denote the dimensions of an image.

In a segmented binary image only the case where f(i,j) has the value of 1 is considered. Hence if that area in the image is labelled as R, the above equation of discrete values takes the form shown below.

$$M_{pq} = \sum_{i\,j\ \varepsilon R} \sum i^p \, j^q \, f(i,j) \qquad p,q = 0,1,2,\ldots \qquad (III)$$

again i and j denote the dimensions of an image.

Perhaps one of the most attractive features of moments is that they may easily be obtained with one pass of the entire image. Furthermore, invariance calculations involve only a few moment values and the original data are no longer required. Particular importance in position and scale invariant pattern identification is given to the concept of central moments. The equation of the central moments is given by

$$M_{pq} = \sum_{i\,j\ \varepsilon R} \sum (i-\hat{i})^p \, (j-\hat{j})^q \qquad p,q = 0,1,2,\ldots \qquad (IV)$$

$$where \qquad \hat{i} = \left(\frac{M_{10}}{M_{00}}\right)$$

$$where \qquad \hat{j} = \left(\frac{M_{01}}{M_{00}}\right)$$

Hu [44], as it has previously mentioned, has derived a set of moment functions that has the desired property of invariance under image translation and rotation. Similar invariants have been proposed for use in automatic ship identification [49]. A set of moments invariant functions based only on the second and third order moments are presented below and have been used by Dudani et al. [31] in automatic aircraft identification.

$$F_1 = (M_{20} + M_{02})$$

$$F_2 = (M_{20} - M_{02})^2 + 4M_{11}^2$$

$$F_3 = (M_{30} - 3M_{12})^2 + (3M_{21} - M_{03})^2$$

$$F_4 = (M_{30} + M_{12})^2 + (M_{21} + M_{03})^2$$

$$F_5 = (M_{30} - 3M_{12})(M_{30} + M_{12})[(M_{30} + M_{12})^2 - 3(M_{21} + M_{03})^2]$$

$$+ (3M_{21} - M_{03})(M_{21} + M_{03})[3(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2]$$

$$F_6 = (M_{20} - M_{02})[(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2] + 4M_{11}(M_{30} + M_{12})(M_{21} + M_{03})$$

$$F_7 = (3M_{21} - M_{03})(M_{30} + M_{12})[(M_{30} + M_{12})^2 - 3(M_{21} + M_{03})^2]$$

$$- (M_{30} - 3M_{12})(M_{21} + M_{03})[3(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2]$$

It is worth noticing the fact that several low order central moments have a direct analogy with the mechanics of bodies [56]. In particular M_{20} represents the horizontal centralness, M_{02} the vertical centralness, M_{11} indicates the diagonality, M_{12} represents the horizontal divergence, M_{21} indicates the vertical divergence, while M_{30} shows the horizontal imbalance and M_{03} shows the vertical imbalance.

Normalisation with respect to possible translations in the object position is achieved by computing the central moments using the equation previously shown. The central moments of any order may be additionally computed from the original moments as is clearly shown below.

$$L_{pq} = \sum_{r=0}^{p} \sum_{s=0}^{q} C_r^p \; C_s^q \; (-\hat{x})^r \; (-\hat{y})^s \; M_{p-r,\,q-s} \qquad (V)$$

$$where \qquad \hat{x} = \left(\frac{M_{10}}{M_{00}}\right)$$

$$and \qquad \hat{y} = \left(\frac{M_{01}}{M_{00}}\right)$$

$$and \qquad C_r^p = \frac{p!}{r!\,(p-r)!}$$

Rotational normalisation is achieved by taking the moments with respect to the principal axis of the object. For the object in Figure 3.1 the principal axis x' and y' are clearly defined. The major principal axis x' passes through the length of the object and the minor principal axis y' passes through the width.

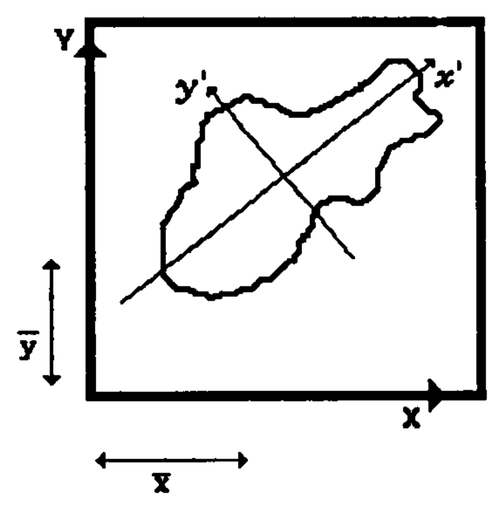


Figure 3.1. Image coordinate axis.

The principal axis moments are obtained by rotating the axis of the central moments until M_{11} equals zero. The angle, θ, from the original axis to the principal axis is defined by

$$tan\,(2\,\theta) = \frac{2\,\hat{M}_{11}}{\hat{M}_{20} - \hat{M}_{02}}$$

where $\hat{M}$ corresponds to the central moments. The angle θ obtained with the above equation may be with respect to either the major or the minor axis. One way to determine the unique principal axis is to set the additional constraints that $\phi_{20} > \phi_{02}$ and $\phi_{03} > 0$. The correct rotation angle will be $\theta + n_{\pi/2}$ where n is chosen to satisfy those constraints. Once the correct angle θ has been determined, the normalised rotated moments ϕ_{pq} can be computed from the central moments as follows

$$\phi_{pq} = \sum_{r=0}^{p} \sum_{s=0}^{q} (-1)^{q-s} C_r^p C_s^q cos\,\theta^{p-r+s} sin\,\theta^{q-s+r} \hat{M}_{p-r+q-s,r+s}$$

where $\hat{M}$ once again represents the central moments.

Regarding the problem of size normalisation, the area of a segmented binary image is given by M_{00} that may also be used to normalise other moments with respect to size. The area may be normalised to a unit length by using the following transformation

$$n_{pq} = \frac{\phi_{pq}}{\phi_{00}^{\gamma}}$$

where $\gamma = \frac{1}{2}(p+q)+1$. In summary, it has been shown that the moments can be normalised with respect to translation, rotation and size. In doing this the location of the centre of gravity $(\hat{x}, \hat{y})$, the orientation θ, and the size (area) M_{00} can be obtained. A normalised set of moments, or standard moments is characterised by the following low order moment values: $n_{00} = 1$ (normalised size), $n_{01} = n_{10} = 0$ (normalised location), and finally $n_{11} = 0$ (normalised orientation).

The higher order moments may now be effectively used directly for shape analysis and shape comparison. There are six remaining standard moments of order two and three which could be used instead of the seven moment invariants. Normalised moments with orders higher than three may be easily computed with the above procedures. For a pattern identification operation the most useful standard moments may be selected using a training data set and standard pattern recognition techniques. The selected moments can be combined to compute a discriminating function. Alternatively, since the feature measures have the physical significance of moments, it is possible to compute some standard statistical measures from them and analyse the shape of an object with respect to a set of ideal shape prototypes. This is discussed in detail in [57]. Finally, an additional advantage of the method of moments is that it is easily extended to grayscale data. The relevant equations and analysis of the application of the method of moments in grayscale images can be found in [57].

3.3 The Fourier Descriptors.

3.3.1 Introduction.

The Fourier descriptor is a method of describing the shape of a closed, planar figure. Given a figure in the complex plane, its contour can be traced, yielding a (one dimensional) complex function of time. If the contour is traced repeatedly, the periodic function that results can be expressed in a Fourier series. The Fourier descriptor of a contour is defined by this Fourier series.

To implement this method of shape description, it is necessary to sample the contour at a finite number of points. Since the discrete Fourier transform of a sequence gives the values of the Fourier series coefficients of the sequence (assuming it to be periodic), using an FFT algorithm satisfies the definition above. In addition the computational advantages of the FFT are

well known.

Once the Fourier descriptor has been computed, the operations related to positional and scale changes are simply implemented in the frequency domain by arithmetic operations on the frequency domain coefficients. While shapes may be compared in the space domain, the procedures required to adjust their size and orientation are computationally expensive. Normally an iterative type of algorithm is employed, that searches for an optimum match between the unknown shape and each reference shape.

Persoon and Fu [58] [40] presented a method of using Fourier descriptors to extract shape information. The method retained all the shape information of the original contour, and reduced the problem to one of finding an optimum size, orientation and starting point match between the sample Fourier descriptor and each reference Fourier descriptor. This optimisation process was used to check the similarity of each contour to each of the possible reference contours. While this method did retain all of the shape information, and guaranteed that an optimum size, orientation, and starting point match would be found, it was still fairly time consuming. Richard and Hemami [35] computed the Fourier descriptors using an FFT, and then normalised only their magnitudes. They were able to perform magnitude classifications efficiently using this technique, but the true distance measure including all shape information, required two FFTs for each comparison of an unknown Fourier descriptor to a reference Fourier descriptor.

3.3.2 Definition of a normalised Fourier descriptor.

Consider a closed contour K in the complex plane. If K is traced once in the counterclockwise direction with uniform velocity e, the complex function $z(t)$ with parameter t can be obtained. Choose e so that the time T required to traverse the contour is 2π. If the contour is traversed more than once a periodic function is formed, that may be expanded in a convergent Fourier

series. A Fourier descriptor of K can be defined to be the complex Fourier series expansion of $z(t)$.

$$Z(t) = \sum_{n=-\infty}^{\infty} A(n)\, e^{j n t}$$

where

$$A(n) = \int_{0}^{2\pi} z(t)\, e^{-j n t}\, dt$$

This Fourier descriptor depends on both K and the starting point of $z(t)$. In practice, K is taken from a digitised image, and thus $z(t)$ is not available as a continuous function. The frequencies of the coefficients produced by an FFT of length n range from $(-\pi/2+1)$ to $(\pi/2)$ cycles per total contour length.

3.3.3 Computation of normalised Fourier descriptors.

In order to compute the Fourier descriptors, the original sampled contour is assumed to be given in the generally known form of chain codes. This chain code representation is converted into a sequence of (x,y) coordinates. The piecewise-linear contour defined by the original chain code is appropriately filtered, and the resulting sequence of (x,y) coordinates is taken to represent accurately the original contour. The length of the contour is computed, and the contour is resampled using a spacing that is chosen to represent the total number of samples (these should be a power of 2). The Fourier descriptor is computed by simply taking the FFT of this sequence. There are obvious complications involving the definition of a sampling strategy for arbitrary contours if non-uniform sampling is employed,

therefore the common selection is that of a uniform sampling. The convergence in the case of uniform sampling is shown to be fairly rapid [59].

3.3.4 Normalisation of Fourier descriptors.

The frequency domain operations that affect the positional and scale changes of the contour follow directly from properties of the Discrete Fourier Transform (DFT) [59]. To translate a contour distance x horizontally and a distance y vertically, add $(x + j\,y)$ to $A(0)$. To scale a contour by a factor K, multiply each component $A(n)$ of the Fourier descriptor by K. Rotating the contour in the time domain requires multiplying each coordinate by $e^{j\theta}$ where θ is the angle of rotation. By linearity, multiplying each Fourier descriptor component $A(n)$ by $e^{j\theta}$ is the equivalent frequency domain operation. In order to see how the contour's starting point can be moved in the frequency domain, recall the time-shifting property of the DFT. Shifting the starting point of the contour in the time domain corresponds to multiplying the ith frequency coefficient in the frequency domain by $e^{j\,i\,T}$, where T is the fraction of a period through which the starting point is shifted. Clearly, as T goes from 0 to 2π, the starting point traverses the whole contour once.

Given the Fourier descriptor of an arbitrary contour, the normalisation procedure requires the application of a number of normalisation operations such that the contour would have a standard size, orientation, and starting point. In order to reject noise, all coefficients used in the procedure are chosen to have magnitudes as large as possible.

3.3.5 Classification of patterns using normalised Fourier descriptors.

An effective classification method is essential if we are to compare shapes with those from a library of known shapes. If two sampled contours

with parametric representations $a(i)$ and $b(i)$ are considered, the difference $c(i) = a(i) - b(i)$ can be defined. Clearly, if $a(i)$ and $b(i)$ are identical, $c(i)$ is identically zero. If $a(i)$ and $b(i)$ are not identical, the magnitudes of the $c(i)$ coefficients are an appropriate measure of the difference between $a(i)$ and $b(i)$. If $A(i)$, $B(i)$, and $C(i)$ are the Fourier descriptors corresponding to $a(i)$, $b(i)$ and $c(i)$, then due to linearity $C(i) = A(i) - B(i)$. The mean-square distance measure in the frequency domain corresponds to a time domain criterion that weights each point equally. In recognising a contour corrupted by factors such as quantisation error or poor resolution, the mean-square distance measure criterion seems appropriate [59]. This distance is only meaningful, if the positional and size changes of the inverse Fourier descriptors are similar. This is the function of the normalisation procedure.

Another distance measure used was the sum of the absolute values of the differences between the real and the imaginary parts of the two Fourier descriptors. The results were very similar to those achieved when using the mean-square measure. The latter shows that the normalised Fourier descriptor feature extraction method is of more importance than the classification procedure.

3.4 Aircraft identification using position and scale invariant features.

3.4.1 Using moment invariants.

An early paper in object identification using moments was published by Dudani et. al. [31]. The authors have presented a successful system for the automatic identification of aircraft. They have used a particular set of angles, which they named camera orientation angles, aiming to obtain moment invariance with one of the angles of the set. Consider $(x\ y\ z)$ to be a fixed coordinate system attached to a specific aircraft model with its origin at the

centre of gravity, positive x-axis towards the nose of the aircraft, positive y-axis along the right wing, and positive z-axis toward the bottom surface. Also consider ($X\ Y\ Z$) as another reference system such as $X\ Z$ is the image plane and the Y-axis is the same as the optical axis of the camera. If ($x\ y\ z$) and the ($X\ Y\ Z$) coordinate systems are initially alinged, an aircraft can be oriented by first rotating it about the camera Y-axis, then rotating it about the aircraft's z-axis, and finally rotating it about the aircraft's x-axis. Dudani et al. concluded that varying the elevation merely produces a rotation in the image plane but no change in the aircraft's size and shape.

The authors have constructed a training sample set using the perspective optical image projections of a given aircraft for various values of azimuth and roll angles. The range of values of azimuth and roll covered all *distinct views* of an aircraft. Two aircraft views were defined as distinct if and only if they produced images that did not differ from each other by a rotation, reflection, or a combination of both. For three dimensional objects like aircrafts that possess a symmetry about a plane, the range for azimuth and roll angles varied between -90 and 90 degrees for the first and between 0 and 90 degrees for the second.

In practice the authors [31] have used discretised values of azimuth and roll angles. For all the possible combinations of these discretised values of azimuth and roll angles, the images for the given object were obtained, and from each of these images a set of fourteen features was computed. The collection of these feature vectors computed from the distinct views of an aircraft, generated the training sample set for that object. Such a training sample was formed for every object in the given class of aircrafts used for the invariant pattern recognition process. For the system described in [31], an extensive experiment was conducted in order to construct the training sample set. Six different aircraft types were included in the recognition class: an F-4 Phantom, a Mirage IIC, a MIG 21, and F-105, an F-104, and a B-57. The range of azimuth and roll angles was discretised at intervals of 5^{o}. The

complete training sample set consisted of more than 3000 real images. In dealing with scaling effects, all of the training samples were subjected to a linear transformation so as to obtain zero mean and unit variance for each component when averaged over the entire set. Two distinct decision rules were applied in the classification process: a Bayes decision rule [27] and a distance weighted k-nearest neighbour rule [60]. From random viewing, a set of 132 new images consisting of 22 images for each one of the 6 different aircrafts was formed and stored for subsequent processing. In order to provide a more meaningful evaluation of the classification methods used, four human observers were asked to view the corresponding binary images and to decide upon a classification for each. Figure 3.2 presents a summary of the results. Each computer decision required 30 seconds, while human responses were made (in about 10 to 15 seconds). The authors [31], concluded that the computer outperformed all four observers to a significant degree since human performance is generally considered to provide an implicit upper bound on the classification accuracy attainable in an automatic system.

	Observer	*Number of misclassifications*						
		Phantom F-4	Mirage IIIC	MIG 21	F-105	F-104	B-57	TOTAL
Computer	*Bayes Rule*	2	0	2	2	3	1	9
	Distance - Weighted Rule	1	0	0	2	3	0	6
Human Observers	*Technical (A)*	2	1	3	2	2	0	10
	Technical (B)	1	1	5	2	2	1	12
	Technical (C)	3	2	4	0	3	2	14
	Non-Technical (D)	5	2	10	2	7	2	28

Figure 3.2. Classification results [Dudani et al. 1977]

In a paper published in 1980, Wallace and Wintz [59], experimented with three dimensional aircraft recognition using normalised Fourier descriptors. In their original two dimensional testing case, the authors have used the silhouettes of 20 different commercial aircrafts. The aircraft outlines

were traced and chain codes were created. Some of the normalised Fourier descriptors were used as a library set, and others were compared to them, with the classification being made to the closest library normalised Fourier descriptor. Using the mean-square distance measure, 95% classification accuracy was obtained, and using the absolute-value distance measure, 100% accuracy was obtained. The aircraft outlines were approximately symmetric, mainly due to quantisation errors. Given that the Fourier descriptor represents a symmetric contour, the normalisation procedure generally yielded a normalised Fourier descriptor whose inverse transform had a starting point on the real axis, and whose axis of symmetry coincided with the real axis. Since the actual experimental contours investigated by the authors, were not perfectly symmetric, the normalisation subroutine did not always result in a starting point that fell on the best estimate of the axis of symmetry but it was 'quite close' [59]. The authors [59] have concluded, that such behaviour 'indicates the effectiveness of the algorithm in rejecting noise'.

In the three dimensional case, Wallace and Wintz [59], used the same six types of aircraft as in Dudani et al [31]. An experiment was performed in which unknown aircraft figures were recognised and their orientation in space estimated. The six three dimensional aircraft images were rotated through appropriate angles to create a library of projections. The experiment consisted of identifying randomly unknown selected aircraft projections by comparing their normalised Fourier descriptors to the normalised Fourier descriptors computed from the library of projections. A number of important differences was noted between the two experiments. Firstly, the data used in Dudani et al. were constructed using physical models of aircraft and a TV camera linkage. Although this might appear to be a more realistic approach, it is definitely a more demanding experiment than one using graphically generated data. However, there are two problems with this method that are avoided by the graphical technique. Firstly, the Dudani et al. 5 degree resolution, represented an error in data generation that was avoided by the more exact graphics approach used by Wallace and Wintz [59]. Since the camera had a

finite distance from the model, parallax problems affected the images, making the camera image different from the image received from a distance. Since most practical photographs of actual aircrafts in flight are taken at a much greater distance, this error was undesirable. A limitation of the moment invariants used by Dudani et al. is that although features computed from mirror images are identical, mirror images must be distinguished in order to estimate accurately the orientation of an unknown projection. Such an operation limits significantly the estimation performance [59]. The limited range of projections considered in Dudani et al. limits effectively this problem, but an extension to the analysis of real image data presents a real problem. In the experiments by Wallace and Wintz, a hemisphere was taken to recognise aircraft outlines. 'If the angles near the front view and rear view of the aircraft are deleted, the problem becomes much easier, since shapes vary more radically with slight angle rotations when large surfaces are viewed almost edgewise' [59].

The actual recognition algorithm proceeded as follows. First, the library of projections was computed, and the normalised Fourier descriptor of each projection was computed. Next the mean-square distance from a given unknown contour to each library vector was computed. The distance to the nearest library contour was stored as the current best estimate of the minimum distance achievable. The projections to the nearest library projection were investigated by the mean-square estimation algorithm in an attempt to interpolate between the library projections. Projections resulting in negative coefficients were deleted from the estimation, and the remaining two or three were used to repeat the estimation process. When an estimation of the unknown vector in terms of a number of adjacent library projection vectors was achieved, the distance was compared to the minimum distance achieved so far. If the new distance was less, the minimum distance was updated. This process was repeated for the k nearest library projections with k being in the range of 4 to 10. If the distances to the nearest k projections were approximately equal, the full k projections were used. After a number of

possible sectors were investigated, two possible procedures were available. The estimated orientation was taken to be that of the original nearest library vector, if the estimation failed to improve on this distance. If the estimation procedure was successful, the orientation was computed by multiplying the orientations of the vectors used in the estimation by their appropriate coefficients and then summing the vectors.

The maximum classification accuracy achieved for completely general data was 88%. The authors believed that this was an excellent result for data of this type and resolution (128 by 128), but they argued that if the projection density was increased by 30 to 40% the final accuracy could well have been into the 90% to 100% range. They noted that the estimation procedure was effective in improving classification accuracy and that in general the absolute value of the distance measure was slightly superior to the mean-square distance measure. Finally, the authors [59] noted that the digital filtering used to reduce representation error and the various normalisation methods have helped classification accuracy. Given a chain code representation of the outline of an aircraft projection, the time to compute the normalised Fourier descriptor was about 0.9 seconds for a 128 by 128 pixel image. The normalised Fourier descriptor was then classified and its orientation was estimated in about 1.8 seconds on a PDP 11/45 machine. Most of this time has, understandably, been spent in computing the Fast Fourier Transform (FFT). Obviously, dedicated array processing hardware can considerable reduce these overall time requirements.

In a more recent paper by Reeves et al. [61] published in 1988, the authors investigated the problem of three dimensional shape analysis using moments and Fourier descriptors. The experiment described in [61] is based on the software used by Wallace and Wintz [59]. The same six aircraft were used, and the same method to generate the Fourier descriptor feature vectors was applied. However, most other parameters and the feature vector classification scheme were 'significantly different' [61]. The six aircraft

shown in Figure 3.3, were described by a list of parameters that were used to generate a representation of the three dimensional surface of an aircraft by a set of three and four sided polygons. Feature vector library sets of two dimensional aircraft views were formulated and library samples were created using a uniform grid of viewing angles. Each library object was normalised with respect to size and positional changes and a set of random views of the six aircraft was used to generate a set of feature vectors. The nearest Euclidean neighbour was used in order to match each test view to its closest entry in the library. The class of this entry was then used to identify the test aircraft.

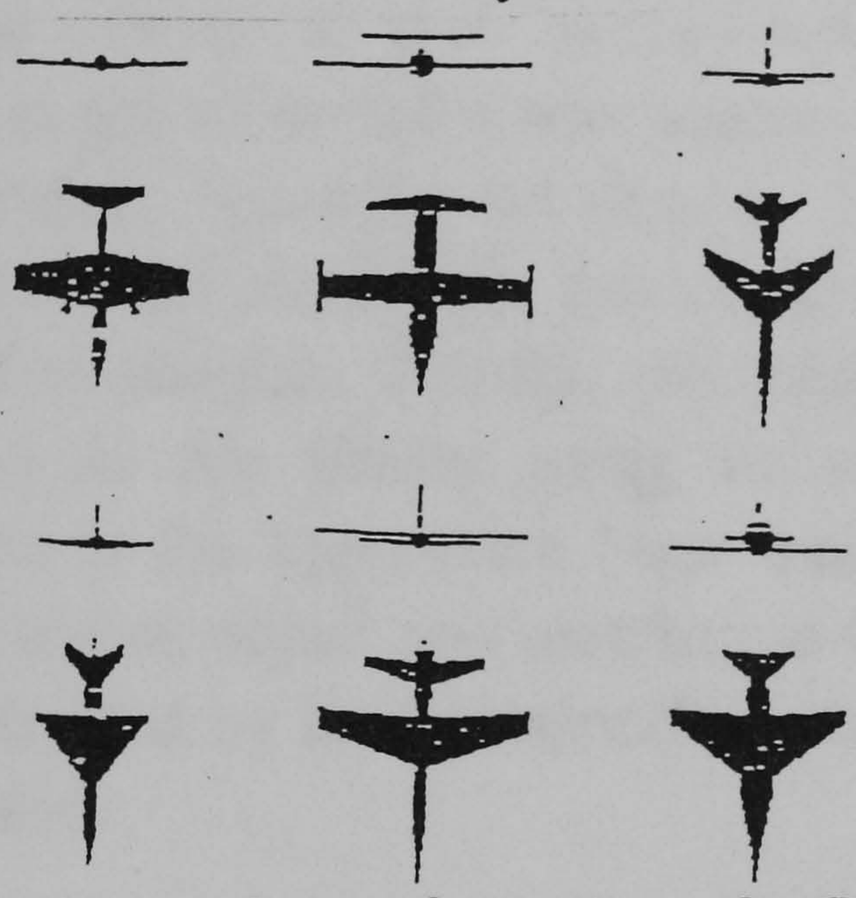


Figure 3.3 The experimental set of six aircrafts [Reeves et al 1988].

The authors have used two classification criteria. The first was the minimum Euclidean distance within the feature space. Both standard moments and normalised Fourier descriptors have feature elements that decrease in magnitude with order or frequency. Therefore, more weight was placed on the lower frequency elements, that were expected to be more robust with respect to noise.

The second criterion used by Reeves et al. [61] was that of minimum Euclidean distance after balancing the variations in feature vectors. Aiming to tackle the possibility that some feature vector elements can have a much smaller range of values than others for the set of objects of interest, the

authors dealt with objects represented by hyperspaces rather than points in hyperspace where classical feature vector 'weighting schemes could not be used' [61]. If it is assumed that all feature vector elements should have had equal importance, one balancing technique could then be applied to scale each dimension of hyperspace by the inverse of the range of the corresponding feature element [61]. The shape of the distribution of each feature vector element for all aircraft views had been unknown, therefore a simple scheme was devised that used the standard deviation of the feature element over all feature elements in the library as a weighting mechanism. The variance balancing scheme adopted in [61] was as follows. Firstly, a library of feature vectors for all aircrafts was computed following a uniform sampling of viewing angles. Secondly, the standard deviation of each one of the feature vector elements in the library was computed and the inverse of these values was used as weights. Thirdly, the authors [61] divided every feature vector element in the library using its corresponding standard deviation. That resulted in the hyperspace been scaled. Finally, before the feature vector of an unknown object was matched to the library, each one of its elements was first divided by its corresponding standard deviation and in this way it was also scaled.

Figure 3.4. Examples of clean images [Reeves et al 1988]. [Reeves et al 1988].

The authors [61] have used a basic library set of 500 views uniformly

sampled over the total sphere of viewing angles with a test set of 50 random views of each aircraft for a total size of 300 images. The images used were of 128 by 128 pixels in resolution; a set of random views for each one of the six aircrafts can be seen in Figure 3.5. A noisy test set was created using a 2 by 2 pixel local mean blurring filter, to which Gaussian noise with zero mean and standard deviation of 30 was added. This set was then applied to a 2 by 2 pixel mean filter and thresholded. Typically, 'for small numbers of feature vector elements, variance balancing was beneficial but for larger numbers of elements the results were worse' [61]. The results for clean data were slightly better than those obtained by Wallace and Wintz [59] mainly due to the more dense library that was used. The classification results using variance balancing for conventional moments on clean data, were particularly improved. Aspect ratio normalisation was also significantly improved when it was combined with variance balancing. The best results in both normalisation techniques were effectively achieved when moments of order 3 were used. Higher moment values had very little additional effect. With respect to the classification results for aspect ratio normalised moments, variance balancing was considerably effective with noisy data. There was a slight improvement recorded when more feature vector elements were added in this case.

Regarding the performance of various other types of moments, combining silhouette and boundary data did give a slight improvement over the silhouette data alone. While the authors [61] have expected that the orthogonal property can lead to better feature vector representation, the results have indicated that they performed worse than the silhouette moments from which they were derived. 'Boundary moments have in general performed much worse than silhouette moments' [61]. The moment invariants had performed in a worse way compared to any other tested technique. Moment invariants were computed in a similar fashion to that used by Dudani et al. [31]. A detailed presentation of all the various moments previously referred can be found in [61], while a table of the best obtained results for each feature vector type can be seen in Figure 3.6.

3.5 Discussion.

Normalised Fourier descriptors and the method of moment invariants are some of the oldest and most known global features for shape description. These types of shape analysis descriptors have proven effective in dealing with both two dimensional and three dimensional pattern classification problems. Although both methods have achieved significant classification successes, there remains an important class of shape description problems that global methods such as moments invariants are unable to solve. 'Whenever a major part of the object under investigation is missing, global features are all affected to such an extent that classifications performed in the usual way are impossible' [43].

Figure 3.5. Examples of noisy images [Reeves et al. 1988].

If sufficient a priori information is available, classifications can still be achieved in which comparisons are made to partial shapes created from the complete shapes, but such approaches are clearly impractical as regard to the general three dimensional pattern identification problem. Other theoretical approaches in using global features exist for partial shape recognition but offer little hope for a generally efficient procedure. A syntactic approach is used by many global features in order to solve the problem of automatic machine recognition. They typically use object grammars derived from

pattern classes and certain production rules to map the original primitive features to a final classification. 'These procedures usually progress through intermediate classifications of primitive combinations' [43]. The significant disadvantage of approaches previously mentioned is the fact that they lack an effective grammatical interference algorithm. Such type of algorithms can enable an automatic inference of pattern class grammars, based on a set of training samples. The lack of such procedures has lead to either self-made step to step grammars, or use of a man-machine interactive system capable of finding the appropriate grammars. The second method does not behave significantly different to the syntactic method when the number of classes (grammars), is 'relatively small' [43]. However, for the recognition of three dimensional objects from random viewing angles, hundreds of projections are required for the description of an object. Since these projections define many pattern classes, the labour required to derive appropriate grammars becomes 'prohibitive' [43].

Vector Length	*Percent Correctly Classified*		
	Standard Moments	Fourier Descriptors	Moment Invariants
6	93.0 (85.7)	77.7 (75.0)	68.0 (69.3)
11	93.3 (89.7)	90.7 (82.3)	
17	93.3 (89.7)	90.3 (85.0)	
24	92.7 (91.0)	91.7 (85.6)	

Figure 3.6. Classification success [Reeves et al. 1988]

In [54], Abu-Mostafa and Psaltis commented about the invariant aspects of moment invariants. The authors found that moment invariants are not in general good features. They suffer from information loss, suppression, and redundancy that often limits their classification capability. However, there are specific instances when these limitations do not necessarily decrease the overall classification performance. For instance, 'when the images do not have significant information in the higher order coefficients of their circular harmonic expansion, nothing is really lost, and when the central parts of the

images have little useful detail, nothing is really suppressed' [54]. The authors have also found that although Zernike moments invariants [55] suffer from the same information loss problems of ordinary moment invariants, they do not present the problem of information suppression or redundancy. If the expected noise is mainly centralised, the second order statistics of the noise process, can be applied to estimate a second order approximation of the probability of misclassification. In the case of white noise, the authors have concluded that higher order moment invariants were more vulnerable. 'In a problem where higher order circular harmonics are the essential features of the patterns, this vulnerability puts a rigid limitation on how well moment invariants can provide the necessary discrimination information' [39]. This factor can normally determine if moment invariants are appropriate for a problem, or else some other feature space might be sought.

3.6 Conclusions.

A synoptic description of two of the most commonly used techniques for position and size invariant object identification using shape descriptors has been presented. Normalised Fourier descriptors and the method of moments have been shown to possess a number of desirable properties that allow them to recognise two dimensional and three dimensional objects invariant of their position and relative size. The computational cost for evaluating either method is shown to be limited and normally in the range of 2 seconds at maximum for a PDP 11/70 minicomputer. The classification results, although they vary substantially according to the specific object used for recognition and the level of noise present in pictures, offer a significant successful classification ratio that in many cases was recorded to be in the order of 90% to 100%. The fact though, that both methods required an extensive training library picture set as well as a large testing set, and the additional common pre-requirement of sufficient a priori information for an effective classification procedure, pointed to the need for an improved classification mechanism that would be able to overcome the above limitations. Clearly,

both the limited computational cost and the high certainty factor recorded for both shape descriptors are highly desirable features that should be included within any improved alternative mechanism. A mechanism that is particularly considered as the most appropriate alternative to the problem of three dimensional invariant object recognition is that of an artificial neural network system. Such complex architectures will be described in detail in the following chapters of the thesis, with emphasis placed once again on the corresponding advantages and limitations of the various neural network systems.

CHAPTER 4

Artificial Neural Networks and Invariant Pattern Identification: A Review of Current Neural Network Architectures

4.1 Introduction.

The fact, that biological computation is so effective, suggests that it may be possible to create similar capabilities in artificial devices based on the design principles of biological neural systems. Artificial neural network models, appear in bibliography by many names such as connectionist models, parallel distributed processing models, layered self-adaptive systems and self organising systems. All these systems attempt to achieve good performance through dense interconnection of simple computational elements. Rather than following the von-Newmann approach in performing a sequential execution of computer instructions, artificial neural network models explore many competing hypotheses simultaneously using massively parallel networks composed of many computational elements connected by links with variable weights. Typically, a neural network consists of many processing elements, each connected to many others. An input array or sequence of numbers, is introduced into the network. Each processing element in the first neural network layer takes a component of the input array, operates on it in parallel with the other processing elements in the layer according to the transfer function - a mathematical formula that defines the element's output signal as a function of whatever input signals have just arrived and the adaptive coefficients present in the local memory - and delivers a single output to processing elements in a layer below [Figure 4.1]. Since inputs and adaptive coefficients can change over time, the network itself adapts and learns.

4.2 Problem Description.

The problem of identifying a presented pattern independent of its actual orientation and size is a major challenge for current artificial neural network architectures. Normally outside a neural network environment users apply classic shape descriptors as described in chapter 3, in order to achieve a translation, position and scale invariant pattern recognition. All attempts regarding the previously described classification techniques require a large training set of images normally in the range of thousands of patterns, and aim to yield a universally acceptable position independent result. Methods like these described in chapter 3 of this thesis, specify in general a small rotational interval, for example five degrees, and take as many as (360/5) images for each of the three main axis in order to train the underlying pattern recognition system.

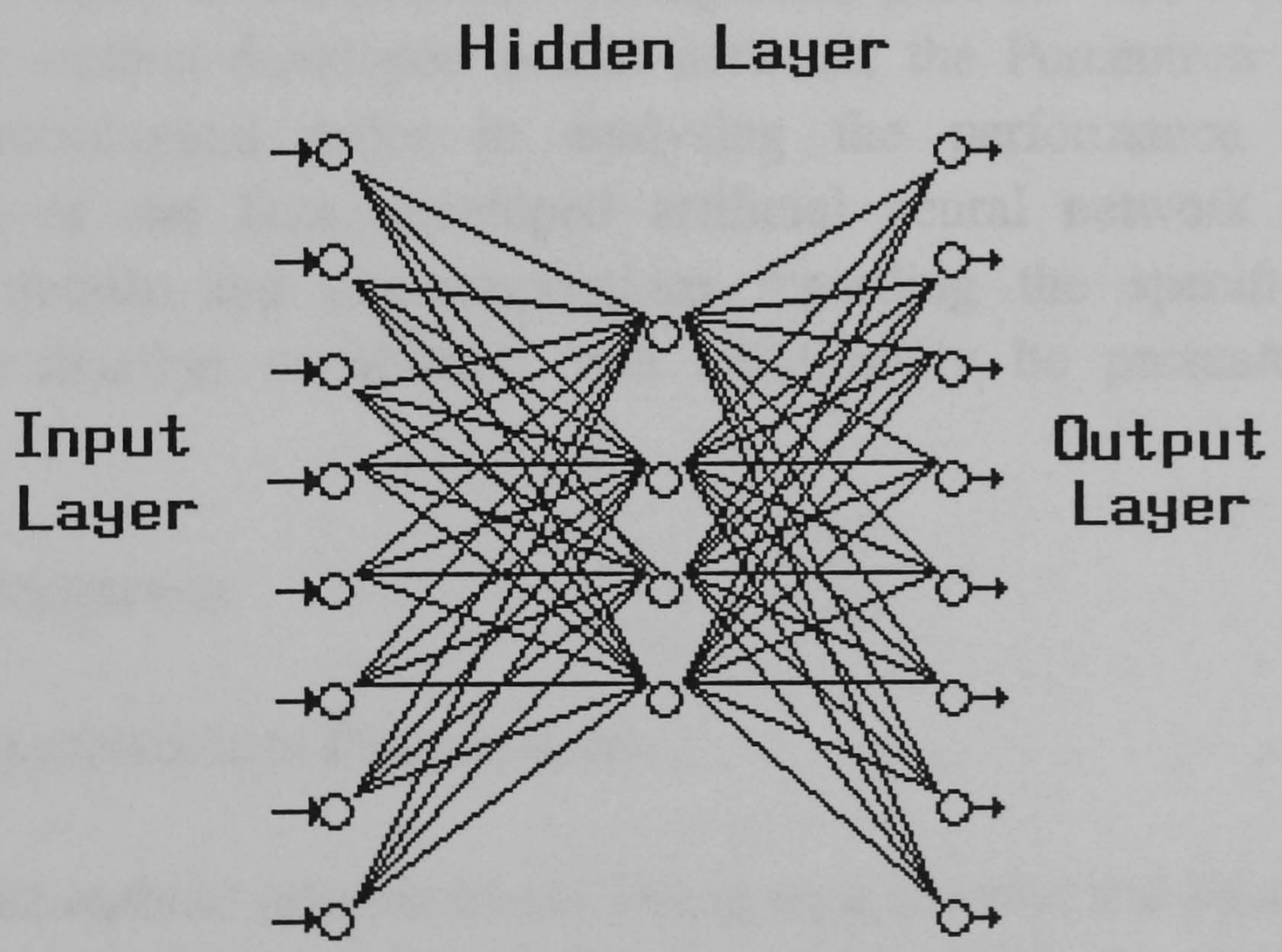


Figure 4.1: A general schema of a neural network topology.

Obviously the size of training set of patterns per object (normally in the range of thousands) is unacceptable for on-line applications and it

additionally can not take successfully into account any object rotation of less than five degrees. Modern neural network system approaches to the same problem are in several cases very similar as far as the sizes of the training sets are concerned. They are characterised though, by the generalisation property of self-organising systems that allows them to successfully classify a previously unknown pattern presented to their input layer although no prior knowledge of that image might have been previously obtained. Most of the known neural networks require extensive training with thousands of images taken from different orientations in order to identify an object independently of its rotation, scale and translation. This chapter will pay particular attention in pointing out the limitations and advantages of the known neural network systems, placing more emphasis on how these parameters can influence a general pattern identification process. An important point of discussion will be the ability of the known neural network architectures to support a rotation, translation and scale invariant pattern recognition process. The chapter will start with the earliest developed neural network, the Perceptron and will follow a chronological order in analysing the performance and the characteristics of the later developed artificial neural network systems. Performance details and characterisations regarding the specific neural network generalisation capabilities will additionally be presented when available.

4.3 The Perceptron.

4.3.1 Neural Architecture Presentation.

In 1958, Rosenblatt introduced the Perceptron, a biological neural model that took into account much of the foregoing work in the area of neurobiology, and which represented a 'major attempt to place the complete learning sequence of an artificial neural network on a rigorous mathematical basis' [63]. Rosenblatt [64] [65] succeeded in proving that learning of an input-output relationship will indeed occur in a linear summation network

under very general conditions of repeated presentation of input and desired output patterns. It is necessary that weighting elements follow certain rules of growth and that a solution always exists for the set of values of the weighting elements. Although improved implementations of the original Perceptron were developed by Rosenblatt, the original Perceptron was a single layer artificial neural network that could be used with both continuous and binary values and decide whether an input belongs to one of the two classes of patterns. The elementary Perceptron is a supervised learning, feed-forward recall artificial neural network that uses a weighted matrix to store the connection strengths from each input layer processing element to every output layer processing element. It is a discrete time artificial neural system that stores mappings between inputs and binary outputs. In order to identify a general two dimensional pattern, a more advanced implementation of the original single layer Perceptron, the cross coupled Perceptron [Figure 4.2], was introduced [68]. The Cross Coupled Perceptron is a time dependent feedback system, since both the sequence of input patterns and the internal activity determine the network's response. The purpose of the cross coupled system is to learn not only recognition of isolated patterns, but also any linear transformation (translation, rotation, etc.) of patterns that occur in a close time sequence.

As Rosenblatt [63] stated, the problem may be posed as follows: If the machine is shown and forced to give the correct response to a sequence of patterns P_i, each followed by some transformation of the same pattern $T(P_i)$, will it tend to give the correct response to the presentation of $T(P_j)$ following learning of P_j alone, where P_j is not one of the patterns in the sequence P_i ? It is apparent that the nature of the network transformations between row input signals and association or learning areas is crucial.

4.3.2 Research Reports.

It has been difficult to obtain any performance related data regarding the

recognition ability of the cross coupled Perceptron. Consequently attention was focused on the classification success of the standard Perceptron. In this context, experimental results presented by Block [66] and Marill et al. [67] are presented showing the performance of a typical Perceptron in identifying letters of the english alphabet. Block used the 26 letters of the alphabet in a standard position to train the Perceptron [Figure 4.3]. It can be clearly seen that as the number of training samples increased so did the Perceptron's classification success. As the number of letters used for training reached 15, the classification success was recorded to be almost closed to perfection.

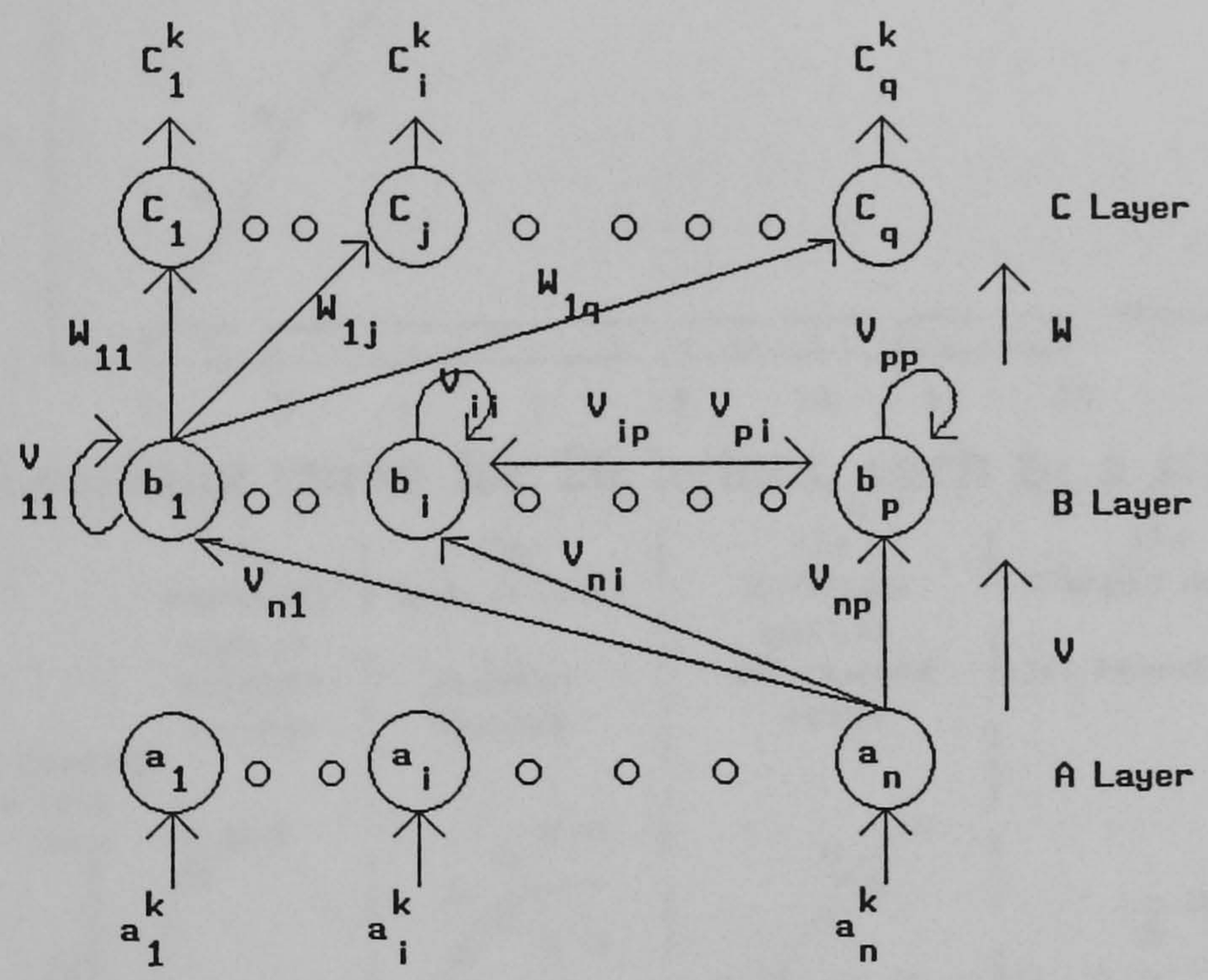


Figure 4.2 : The Cross Coupled Perceptron.

Block [66] has also individually experimented with the letters E and X, under normal and noisy conditions. He used both letters in order to train the Perceptron [Figure 4.4] and has additionally introduce the concept of a "trainer's error". Block has introduced two types of training for the Perceptron: error correction and forced learning. Any wrong decision in selecting either of the two training methods (mis-identification of the stimuli)

was regarded as a "trainer's error". From his results in Figure 4.4, it can clearly be concluded that the Perceptron has been significantly resilient to noise and "trainer's error" and has achieved an average classification success of 80%.

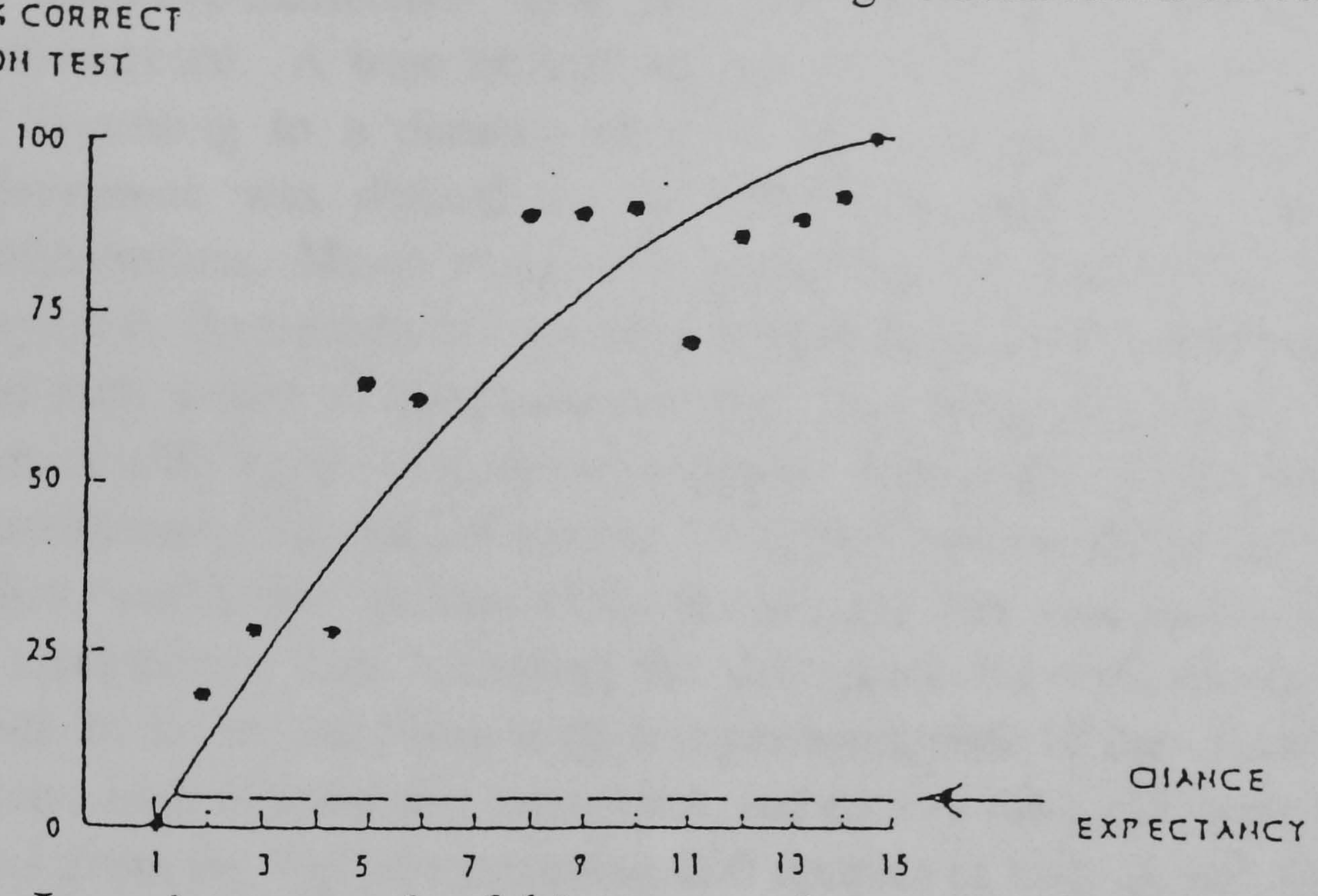


Figure 4.3 : Learning curve for 26 letters, each in a standard position [Block 1962].

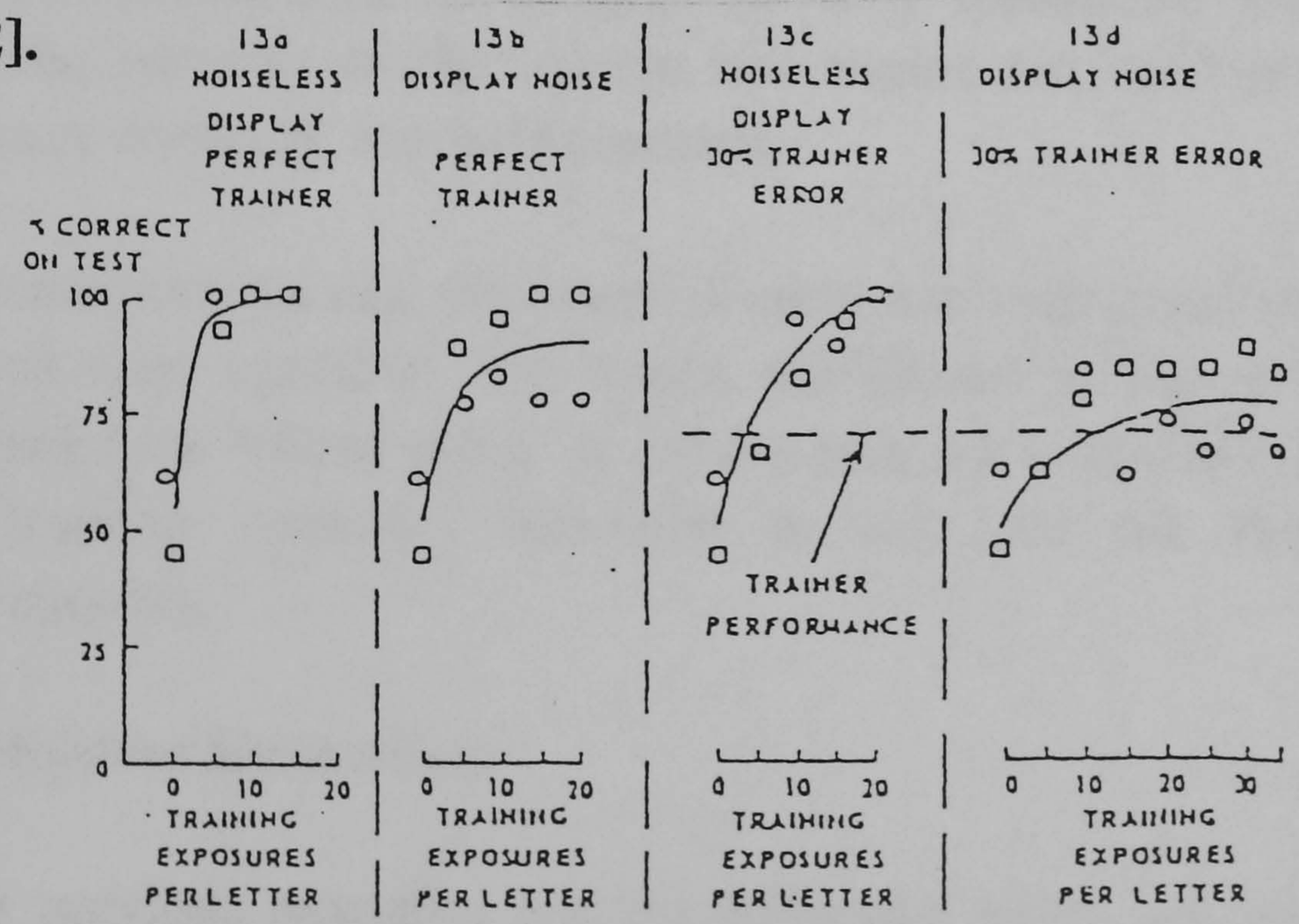


Figure 4.4 : Effects of noisy display and imperfect training on learning of E or X letter discrimination [Block 1962].

Marill et al. [67], used a similar test set to that used by Block [66], with the difference that they used the letters A and B. They also introduced the concept of "effective-ineffective" sets and the classification difficulty measure of divergence. A train or test set was defined as "effective" or "ineffective" according to a distance measure of its probability density function. Divergence was defined as the difference between arbitrary probability distributions. Marill et al. [67] noted that 'in the context of recognition systems, divergence is a quantity defined for pairs of probability densities, and such a pair of distributions comes into being as a result of applying a set of tests to two categories of inputs'. Specifically, in the first experiment the authors [67] formed various "effective" subsets of the set of eight tests, then "ineffective" subsets of the set of eight tests, and finally for purposes of comparison, they calculated the divergence for each of eight individual tests. In the second phase of their experiment, eight of the selected sets of tests were each coupled to a categoriser, and each of the eight systems so formed was given the task of recognising 200 samples of letter A and 200 of letter B. The percentages of samples correctly recognised are given in Figure 4.5. The receptor of the system was varied but the nature of the discrimination A versus B, was held constant.

In the second experiment, the discrimination was varied and six possible discriminations were consider. The results are plotted in Figure 4.6. Each point being based on 400 samples. It can be once more noticed that as the number of training examples increased so did also the Perceptron's classification success.

4.3.3 Neural System Evaluation.

From the previous examples and the additional tables presented in the corresponding references, it can be seen that Perceptron, operates effectively if a linearly separable pattern is presented, with an estimated range of successful classifications from as low as 54% to a maximum of 98%.

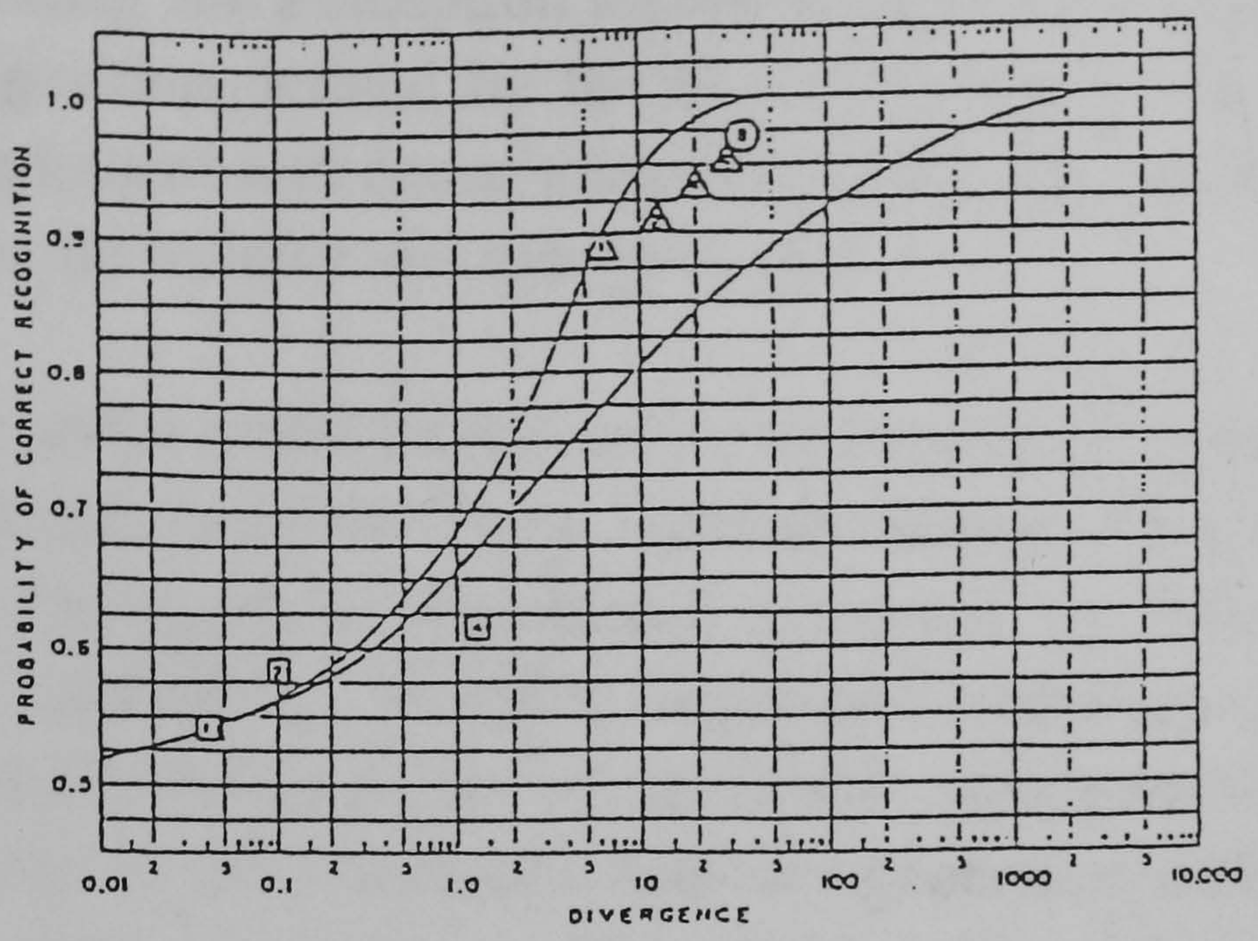


Figure 4.5 : Discrimination of A letter versus B letter. Observed probability of correct recognition as a function of divergence, for eight inputs. The triangular and square points represent, respectively, so-called 'effective' and 'ineffective' subsets. The number within each point indicates the number of tests used in the discrimination [Marill et al. 1963].

As a foundational building block toward successively more powerful models, the Perceptron has been a significant success and has been applied in areas like speech processing, image processing and control. Rosenblatt [68] proved that if the inputs presented to the neural network, from two classes are separable, then the Perceptron convergence procedure converges and positions the decision hyperplane between those classes. The only difficulty in that case emerges when the decision boundaries oscillate continuously with inputs being not separable and their distributions are overlapping.

Rosenblatt concluded that the Perceptron training algorithm makes no assumptions concerning the shape of the underlying distributions but rather focuses on errors that occur 'where distributions overlap' [68]. Among other

remarks regarding the Perceptron's capabilities it is noticed that Perceptron adaptation algorithm defined by the Perceptron's convergence procedure is simple to implement and doesn't require storing any more information than this present in the weights and the threshold.

The Perceptron might be limited by its linear separability conditions, but it has a well understood behaviour, adequate storage capacity, and immediate recall. The Perceptron's limitations, presented in Rosenblatt's original contribution, included a lengthly supervised learning, inability to code nonlinearly separable classifications and finally poor generalisation for which Rosenblatt notes : 'the Perceptron does not generalise well to similar forms occurring in new positions in the retinal field, and its performance in detection experiments, where a familiar figure appears against an unfamiliar background, is apt to be weak' [68].

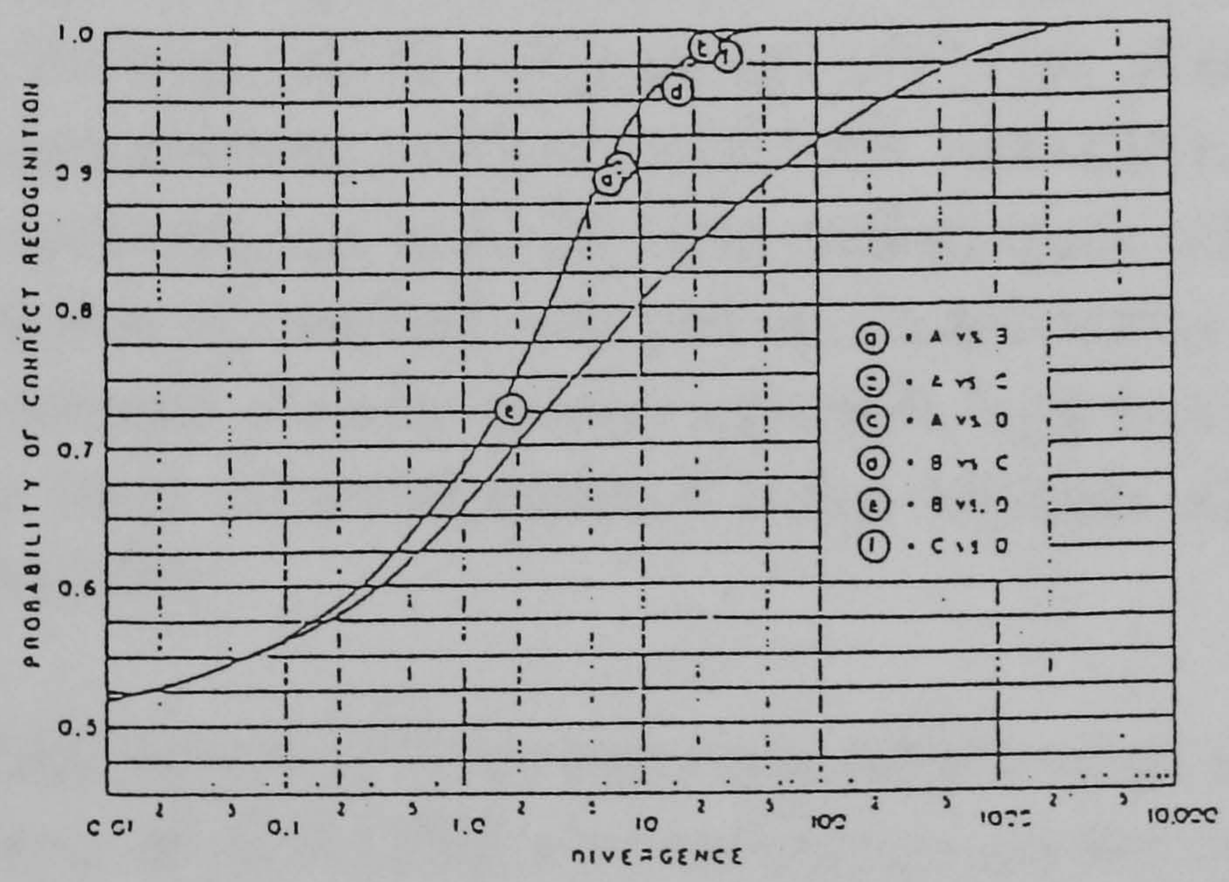


Figure 4.6 : Discrimination using a receptor consisting of tests 4 and 6. Observed probability of correct recognition as a function of divergence [Marill et al. 1963].

The fact though that Rosenblatt introduced a very primitive idea of the

later widely used Back-propagation theory in his later improved Perceptron neural systems, points out the Perceptron's significance as one of the pioneering neural network systems.

4.4 The ADALINE and MADALINE Artificial Neural Systems.

4.4.1 Neural Architecture Presentation.

In the early 1960's methods for adapting nonlinearly separable logic functions were developed and lead Widrow [69] to create a new type of neural network called ADALINE and later its improved version the MADALINE. In ADALINE, all input and output values are binary, while weights are essentially continuously variable, and are automatically adjusted by the adaptive algorithm, thereby controlling the decision making function of the neuron. Generalisation is consider to be successful if the neuron responds correctly, with high probability, to input patterns that were not included in the training set. In early papers [69] [70] Widrow deals with ADALINE implementations and concludes that ADALINE is a two layer feed-forward Perceptron, that splits its input pattern space with a hyperplane, thereby being capable of realising only pattern classifications that are linearly separable. The research into non-linearly separable logic functions lead to the design of a new form of neural network called Multiple ADAptive LInear NEuron (MADALINE).

MADALINES [Figure 4.7] are two layer feed-forward neural networks, composed of a layer of ADALINE's whose outputs are fed into a single fixed logic element, typically an 'OR' logic device or a majority vote taker element. The MADALINE is a hetero-associative, nearest-neighbour pattern matcher that stores pattern pairs using the least mean-square error correction encoding procedure [72]. In order to be able to identify taught patterns, Widrow and Winter [71], noted that MADALINE incorporates a minimal disturbance principal. When the network is presented with an input and it

responds correctly, no adaptation is done. When a correction to the neural network is needed, the network is disturbed as little as possible. Widrow and Winter [71] stated that, 'when the response is wrong, it is obvious that one or more of the first layer ADALINES that is giving a wrong vote, needs to be changed'. The minimal disturbance principal points to change the ADALINES which confidence level is close to zero. The magnitude of the weight change vector needed to cause this ADALINE to reverse its output is smallest. Overall then, the weights in the network are minimally disturbed. Thus, new patterns are accommodated with least likelihood of disturbing the solution for the patterns the network has already been trained on. Given that the patterns are presented acyclically, Widrow and Winder showed that these rules would lead to a network solution if one exists.

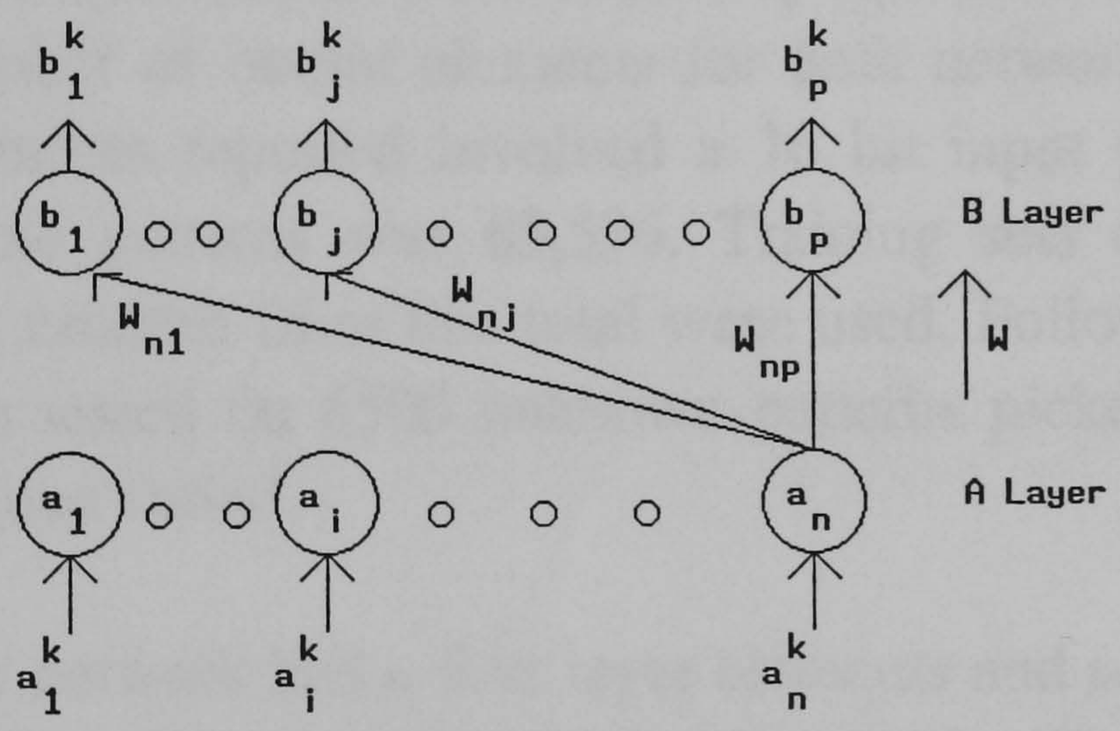


Figure 4.7: Topology of MADALINE.

4.4.2 Research Reports.

In [71], Widrow and Winter provided a rigorous presentation of experimental results of training a MADALINE. The algorithm used is known as the MRII algorithm. The general idea behind their training is described in Figure 4.8.

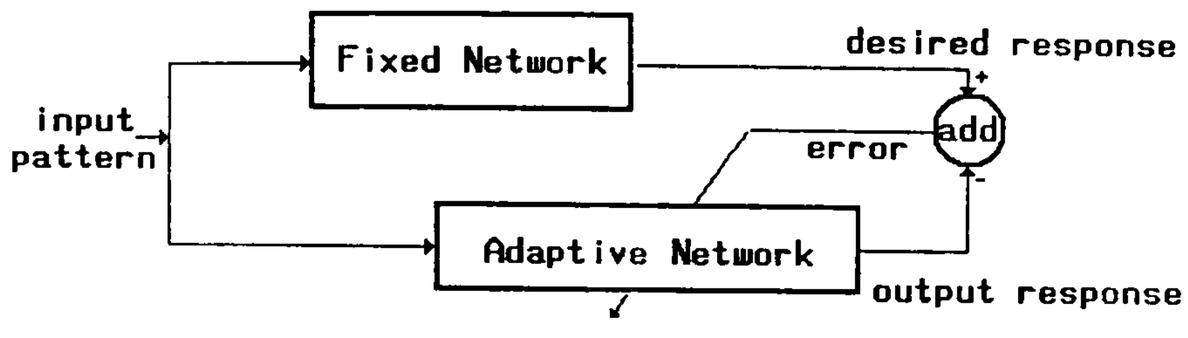


Figure 4.8: Training an adaptive network to emulate a fixed net.

As Widrow noticed that, 'a neural net is not expected to do well on a problem that it can not solve due to limitations of its architecture'. For this reason, they used a neural network as a source of training data for networks trained by MRII. Input patterns were fed to a fixed network with known architecture and simultaneously to the network being trained. The fixed network's response was used as the desired response for training the adaptive network. The number of output elements for both networks was the same. The simulation studies reported involved a 16 bit input pattern. The total number of possible patterns was 65,536. Training sets of 650 and 1500 patterns picked at random from this total were used. Following training, the network was then tested on 6500 unknown patterns picked randomly. The network architectures varied.

If a particular network had n first layer elements and m output elements, it was referred to as an n-feed-m network. The weights in the fixed network were chosen at random. This provided an arbitrary mapping between inputs and outputs. The performance of the adaptive network as training progressed was nonmonotonic. The input pattern to hidden layer output pattern map was changing frequently as well as were the separations made by the output units. Figure 4.9, is a typical plot of the network performance on the training set and on the generalisation test set as a function of the number of patterns presented.

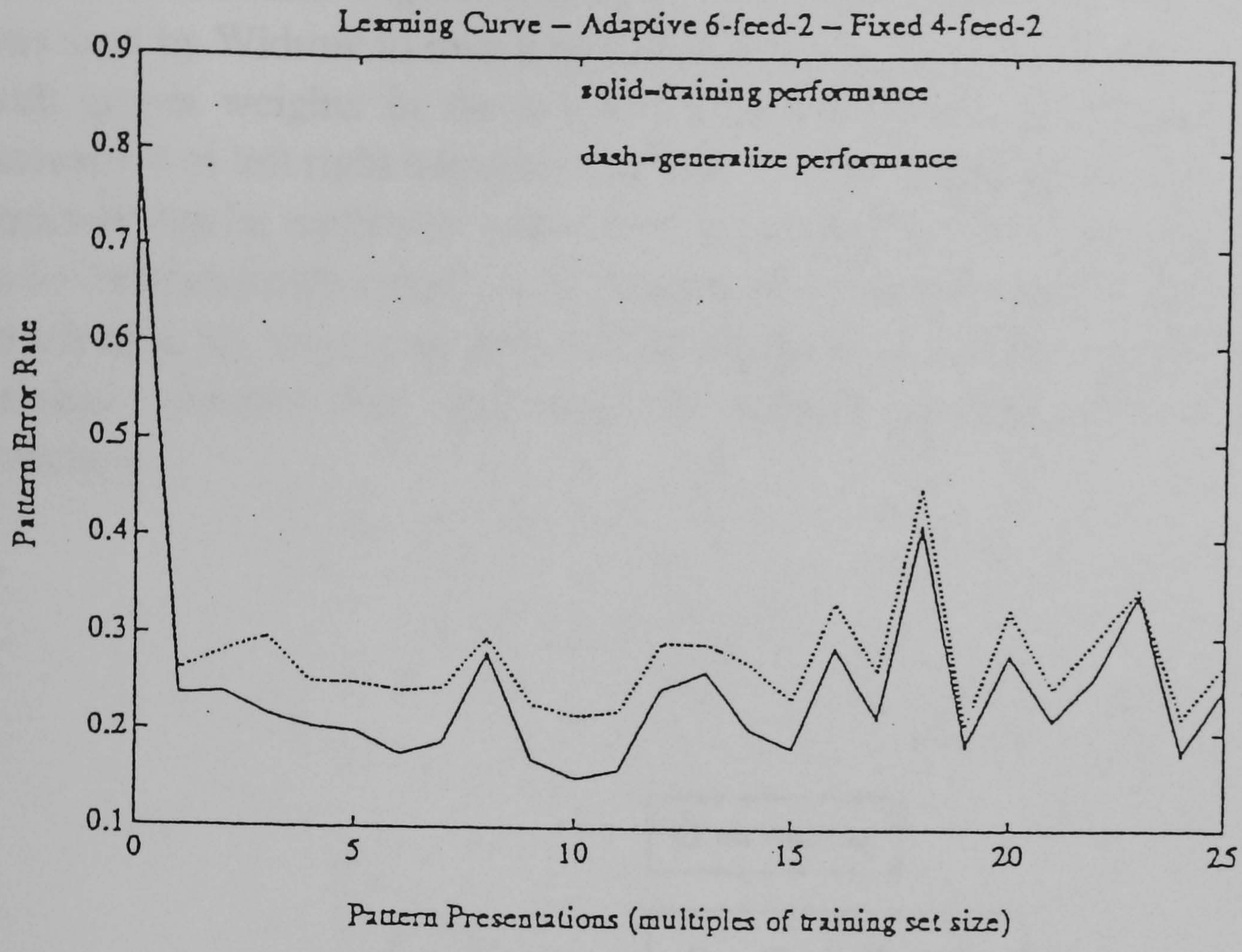


Figure 4.9: Typical learning curve for the MRII algorithm [Widrow and Winter 1988].

The algorithm rarely achieved perfect performance on the training set, 'but occasionally came quite close' [71]. The experiment was conducted 25 times, with each run starting from a different set of initial conditions in the adaptive network. The run was stopped when a set training error rate goal was achieved. For those runs never achieving the training goal, training was terminated after a maximum number of presentations, were made.

In [73] Widrow et al. described an approach to MADALINE that resulted in an invariant of scaling, translation and rotation, recognition of patterns. That particular version of MADALINE encapsulates the main

objective of the present chapter (that of invariance), and will consequently be described in detail. Figure 4.10 shows the neural network configuration that was used by Widrow to map a retinal image into a single bit output such that, with proper weights in the neurons of the network, the response will be insensitive to left right translation and/or up-down translation. 'The same slab structure can be replicated with different weights, to allow the retinal pattern to be independently mapped into additional single bit outputs, all insensitive to left right, up down translation' [73]. Figure 4.11, shows a single line of the original structure that was used to achieve up and down translation invariance.

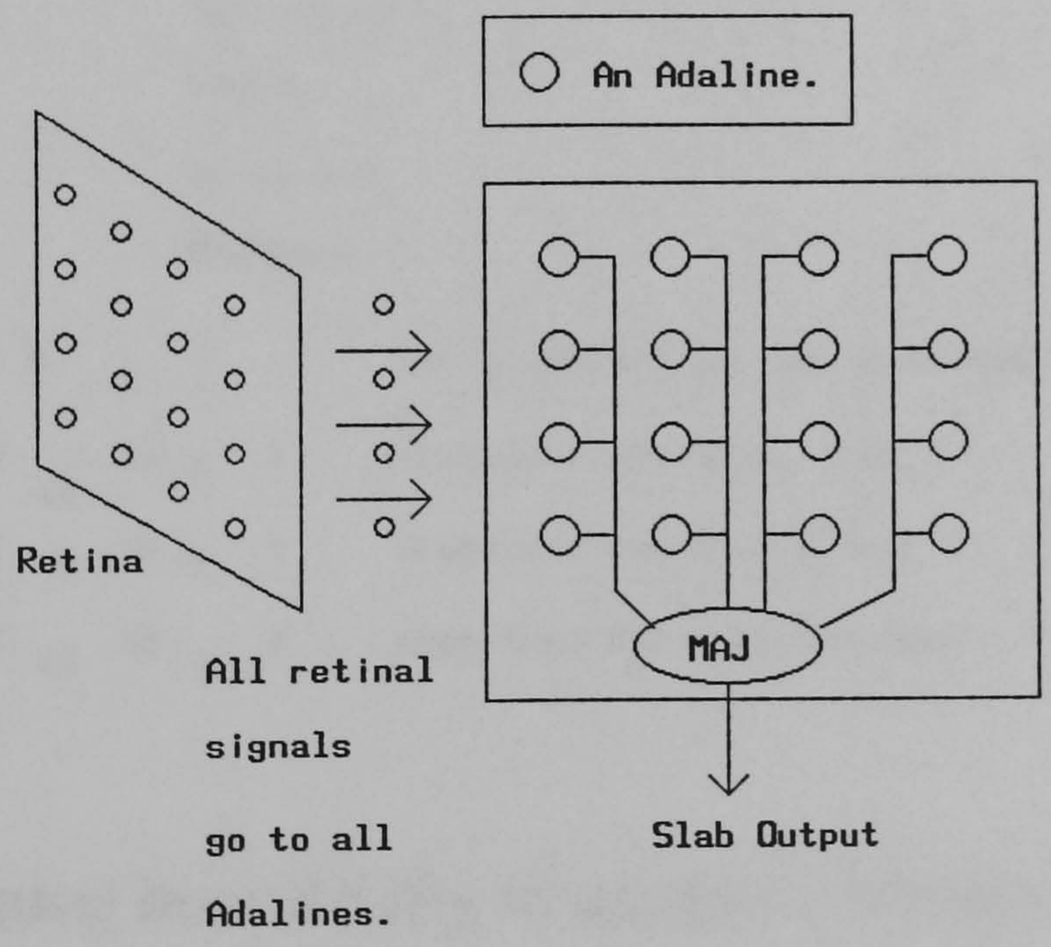


Figure 4.10: One slab of a left right, up and down translation invariant network [Widrow et al. 1988].

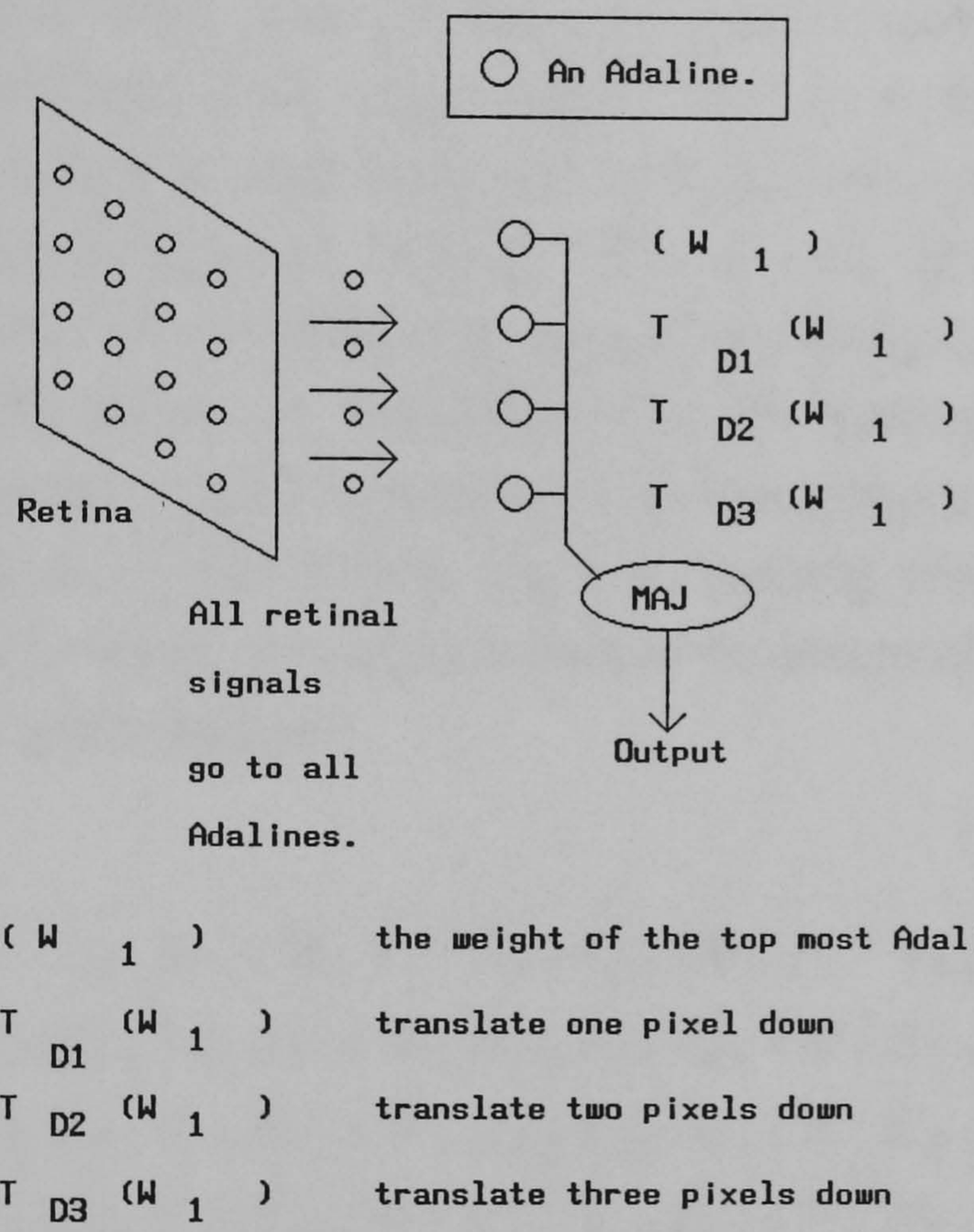


Figure 4.11: Training insensitivity to up down translation [Widrow et al. 1988].

Figure 4.12, presents the final structure that was independent during 90 degree rotations and any left, right, up or down translation. The final weight matrix proposed by Widrow that resulted in a neural network that was independent from translations and integer multiples of 90 degrees pattern rotations is given in Table A. In order to achieve invariance to scaling, the same principles that were used to translation and rotation invariance, were again applied. Widrow [73] commented that, 'establishing a point of expansion on the retina so that input patterns can be expanded or contracted with respect to this point, two ADALINE's can be trained to give similar responses to patterns of two different sizes if the weight matrix of one were expanded about the point of expansion like the patterns themselves. The amplitude of the weights must be scaled in inverse proportion to the square of the linear dimension of the retinal pattern. Adding many more slabs, the invariance net can be built around this idea to be insensitive to pattern size as well as translation and rotation'.

$$\left\{\begin{array}{llll} R_{C1}(W_1) & T_{R1}R_{C1}(W_1) & T_{R2}R_{C1}(W_1) & T_{R3}R_{C1}(W_1) \\ T_{D1}R_{C1}(W_1) & T_{R1}T_{D1}R_{C1}(W_1) & T_{R2}T_{D1}R_{C1}(W_1) & T_{R3}T_{D1}R_{C1}(W_1) \\ T_{D2}R_{C1}(W_1) & T_{R1}T_{D2}R_{C1}(W_1) & T_{R2}T_{D2}R_{C1}(W_1) & T_{R3}T_{D2}R_{C1}(W_1) \\ T_{D3}R_{C1}(W_1) & T_{R1}T_{D3}R_{C1}(W_1) & T_{R2}T_{D3}R_{C1}(W_1) & T_{R3}T_{D3}R_{C1}(W_1) \end{array}\right\}$$

Table A : The final weight matrix that allows the neural system to be independent of translations and rotations of integer multiples of 90 degrees.

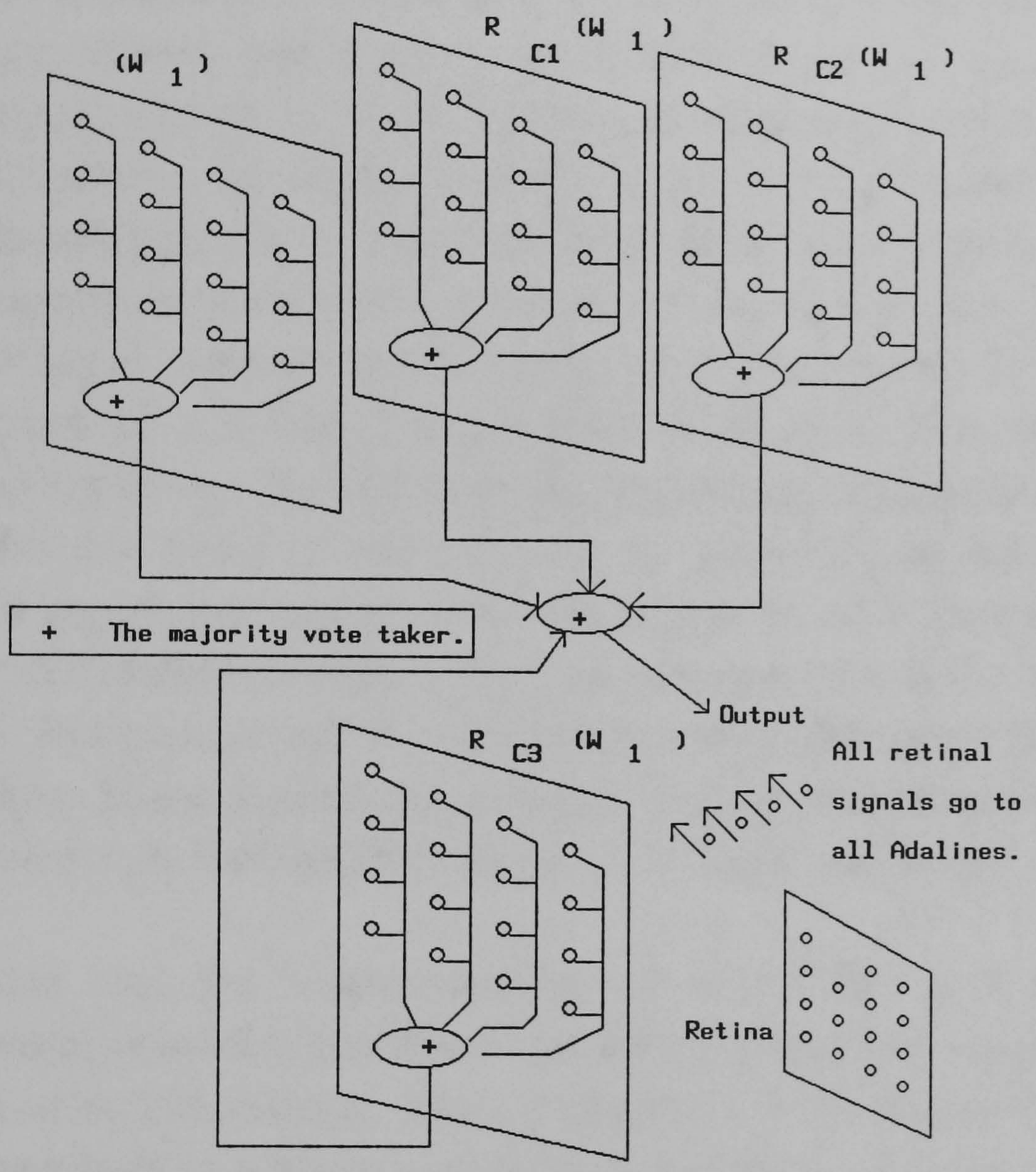


Figure 4.12: A neural network for translational and rotational invariance [Widrow et al. 1988].

4.4.3. Neural System Evaluation.

The results obtained in [71] led to the following observations. The larger training set provided a narrower gap between training and generalisation performances. The effect of increasing the number of first layer elements in the adaptive network over the number in the fixed network was expected to improved the training performance dramatically but would negatively effect the generalisation process. 'It was felt that an increased number of first layer units would increase the degrees of freedom available to the adaptive network to solve the training set. At the same time, these extra degrees of freedom should hamper generalisation by allowing extraneous solutions that solve the training set but do not represent the underlying relationship' [71]. Following the framework of a neuron analogy [74], ADALINES and MADALINES became quickly known for their learning capabilities, leading to a wide range of applications in fields of research such as control, noise cancellation, and most of all signal processing. In recent papers Widrow commented about MADALINE's storage strength to hold as many patterns as the input vector's dimensions and pointed out the well understood mathematical limitations of his paradigm. These limitations included, the lengthly encoding time, the inability to learn on line, and the restriction to linear mappings.

The fact that the implementation of ADALINE and MADALINE artificial neural networks, introduced the theory of least mean-square error, in the design of neural systems, has a particularly wide scientific importance and has given birth to a whole new field of specialised research under the the title of 'Adaptive signal processing'.

4.5 The Back-propagation Artificial Neural Network.

4.5.1 Neural Architecture Presentation.

Since single-layer networks proved severely limited in what they could

represent, and consequently in what they could learn, the entire field of neural networks went into virtual eclipse. The invention of the back-propagation algorithm has played a key role in the resurgence of interest in artificial neural networks.

The elementary back-propagation is a three layer (Perceptron) artificial neural system with feed-forward connections from the input layer processing elements to the middle (hidden) layer processing units and feed-forward connections from the middle (hidden) layer processing elements to the output layer processing elements. In general, it is possible to have several hidden layers, connections that skip over layers, recurrent connections and lateral connections. The elementary back-propagation is a hetero-associative function-estimating artificial neural network that stores arbitrary pattern pairs, using a multilayer gradient descent error-correction encoding algorithm [75].

In order to identify patterns presented to its input, the back-propagation [75], involves two phases. The forward pass, when an external input pattern is passed through the network from the input units towards the output units, leading to an external output pattern. The external output pattern is compared with the desired external output pattern and the error signal for each output unit is produced. The backward pass, where the error signals of the output units are passed backward from the output units towards the input units. The error signals for input and hidden units are evaluated recursively for each next lower layer. To evaluate an error signal for an input or a hidden unit, the error signals of the units to which this unit is connected have to be taken into consideration.

The forward pass requires: first a selection of the next training pair from the training set and application of the input vector to the neural network's input and second a calculation of the network's output. Calculation in multilayer neural networks is done layer by layer, starting at the layer nearest to the

inputs. The weighted sum of each neuron's inputs is calculated, and a "squash" function (Appendix A) takes it to produce the output value for each neuron in that layer. Once the set of outputs for a layer is found, it serves as input to the next layer. The process is repeated, layer by layer, until the final set of network outputs is produced.

The backward pass consists of two phases. First a calculation of the error between the network output and the desired output and second an adjustment of the weights of the neural network in a way that minimises the error. The back-propagation encoding algorithm performs the input to output mapping by minimising a cost function. The cost function is minimised by making weight connection adjustments according to the error between the computed and the desired output layer processing element values. The cost function that is typically minimised is the squared error (the squared difference between the computed output value and the desired one), for each processing element across all patterns in the data set. The weight adjustment procedure is derived by computing the change in the cost function with respect to the change in each weight. The element that makes the back-propagation algorithm so powerful is that this derivation is extended to find the equation for adapting the connections between the input and the hidden layers of a multilayer artificial neural network, as well as the penultimate layer to output layer adjustments. The key element of the extension to the hidden layer adjustments is the realisation that each middle layer processing element's error is a proportionally weighted sum of the errors produced at the output layer.

4.5.2 Research Reports.

Troxel et al. [76], presented a method for classifying objects independent of their position, rotation, or scale, while Yang and Guest [77] presented a similarly invariant to rotation, scale and translation technique, for character identification. In [76], the objects to be classified were multifunction laser

radar data of tanks and trucks at various aspect angles. A segmented doppler image was used to mask the range image into candidate targets. 'Each target was then compared to stored templates representing the different classes. The template and image were transformed into the magnitude of the Fourier transform with log radial and angle axis, feature space. A multilayer Perceptron neural network using the back-propagation algorithm performed the classification with an accuracy near 100%' [76].

The position invariance was accomplished by taking the magnitude of the Fourier transform of the input scene. Shifts in the input scene affected the phase portion of the Fourier transform and had no affect on the magnitude portion. Rotational invariance was accomplished as rotations within the input scene resulted in exact equal rotations in the magnitude of the Fourier transform plane. When the magnitude of the Fourier transform plane was mapped onto polar coordinates, rotations in the input plane resulted in linear shifts along the angle axis. Scale invariance was achieved by using the Fourier transform scaling property. 'If the radial axis is logarithmically scaled, uniform scaling of the input would result in a linear shift along the $\ln f_r$ axis' [76]. Smaller targets were shifted in the positive direction with larger targets shifted in the negative direction. In the same time, rotational changes were translated into linear shifts along a periodic f_θ axis. Changes in scale were translated into linear shifts along a non-periodic $\ln f_r$ axis, that could be thought of as extending to infinity above and below the region chosen to be analysed. A linear shift reflected a shift of the chosen region. A correlation between the target and the template invariant spaces was used to indicate how much scale and rotation difference there was. When the scale and rotational changes were known, a properly scaled and rotated template was chosen from a template bank. The classification experiment was contacted on three sets of input data created from laser range templates. Set one, was set up to demonstrate the classification between true autocorrelations and out-of-class correlations. Set two, was set up to demonstrate the classification between rotated in-class correlations and out-

of-class correlations. Set three, was set up to test the complete robustness of the classification process. It was thought that regarding to the position, scale, rotation invariant correlation peak, all trucks and tanks would be similar. 'Therefore same type objects that were similar, but not necessarily equal, aspect angles, rotations and scales should be able to be separated from one another' [76].

The results for set 1 data, were obtained with a total of 1000 training file iterations. The network was trained with 50 first layer nodes and 2 second hidden layer nodes. The results have showed a near perfect classification and the properly trained neural system made strong decisions one way or the other. The results for set 2 data, were obtained with a total of 2000 training file iterations. The network was trained with 150 first hidden layer nodes and 4 second hidden layer nodes. The classification accuracy was approximately 95%. The results for set 3 data, were obtained with a total of 45000 training file iterations. The results were little changed after about 25000 training file iterations but 'the network was allowed to run in the attempt of completely classifying the training data. This goal was not reached' [76]. The network was trained with 200 first hidden layer nodes and 5 second hidden layer nodes. The overall classification accuracy rate was at 85%. In the case when only the test data were considered, this rate decreased to just below 80%. Troxel et al. [76] commented that, 'these results are still very promising in that they classify training data very well and with a large figure of merit. However, with the very good classification obtained in the training data, it seems reasonable to assume that better results would be obtained with a larger data base to train with'.

In [77], Yang and Guest presented a rotation invariant pattern recognition technique using the back-propagation neural network in order to identify specific letter patterns. They formed a set of training patterns by incrementally rotating the letters A, T, H and R. Some examples are shown in Figure 4.13.

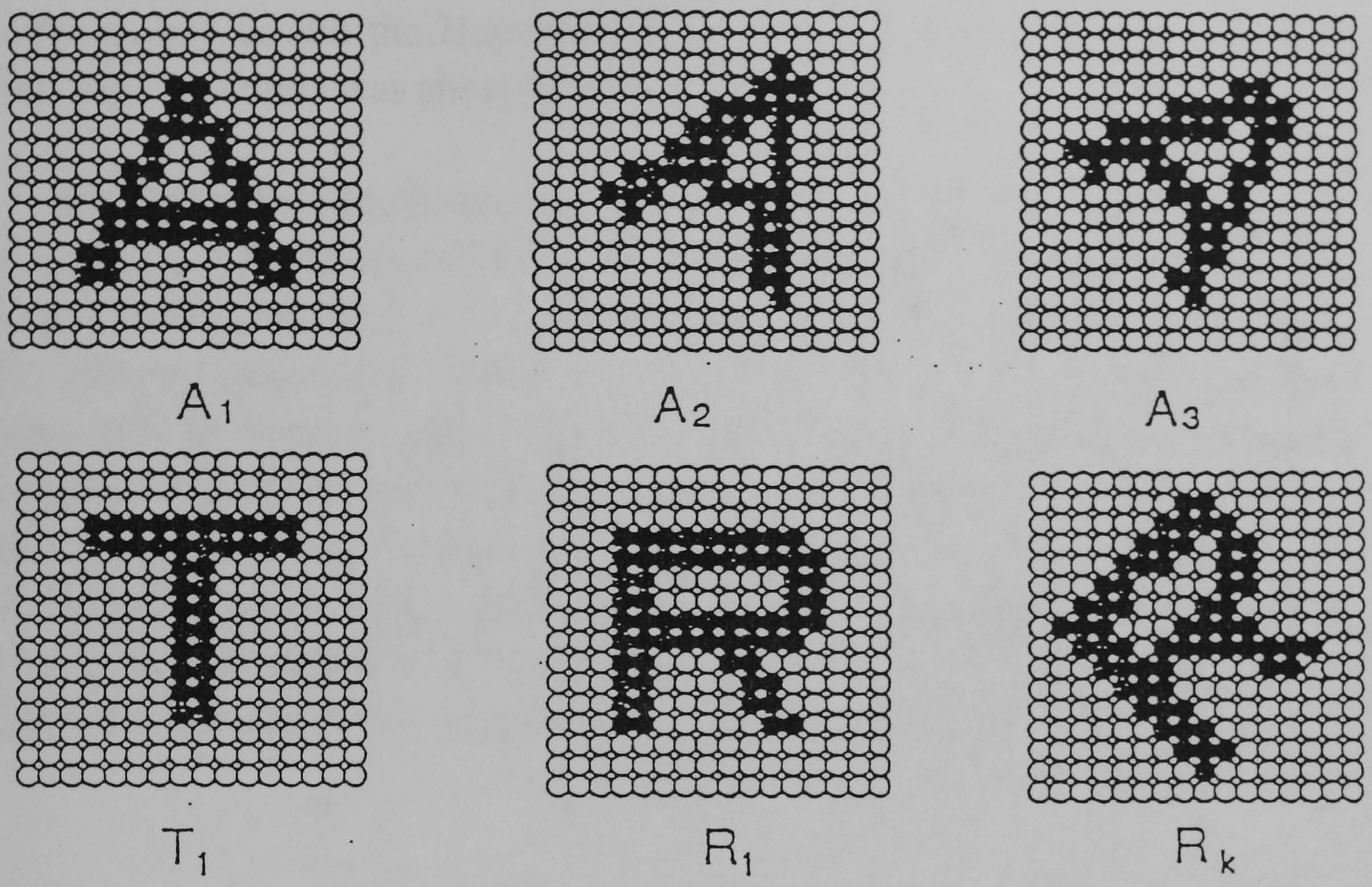


Figure 4.13: A training set of patterns [Yang and Guest 1987].

An actual output recorded to be above 0.9 from any neuron of which the desired value is 1 was considered as correct. Similarly, an actual output below 0.1 was considered as 0. The learning rate α was controlled by the following equation

$$\alpha_i = \alpha_o \; X \; (0.9)^i + 0.1$$

where $\alpha_o = 0.3$ and i was the number of the training cycle. To decide the angle increase, the network was trained with unrotated letters, A_1, T_1, H_1 and R_1. Then rotated versions were sent into the network. Two of the outputs are plotted in Figure 4.14. 'It is clear from the diagram that the deviation of the actual outputs of neurons from the desired values increased with the rotation

angle Θ, and so did the Hamming distances' [77]. The Hamming distance between two letters was about 60.

From Figure 4.14, the increase in the angle was 15 degrees. The number of angles was therefore 24 (360/15). The training cycle contained, a total of 24 by 4 = 96 members. It took about 30 cycles for the network to converge. The average output after learning is shown in Figure 4.15. The average was taken over 48 training angles. Tests of the trained neural network proved that it could correctly recognise all four letters in almost any orientation. 'Only a few exceptions were observed, these were probably due to the lack of nonlinearity, since after increasing the constant α in the sigmoid function $f(u)$, they disappeared' [77]. Figure 4.16 shows the results of the test of four different angles that were not included in the training set.

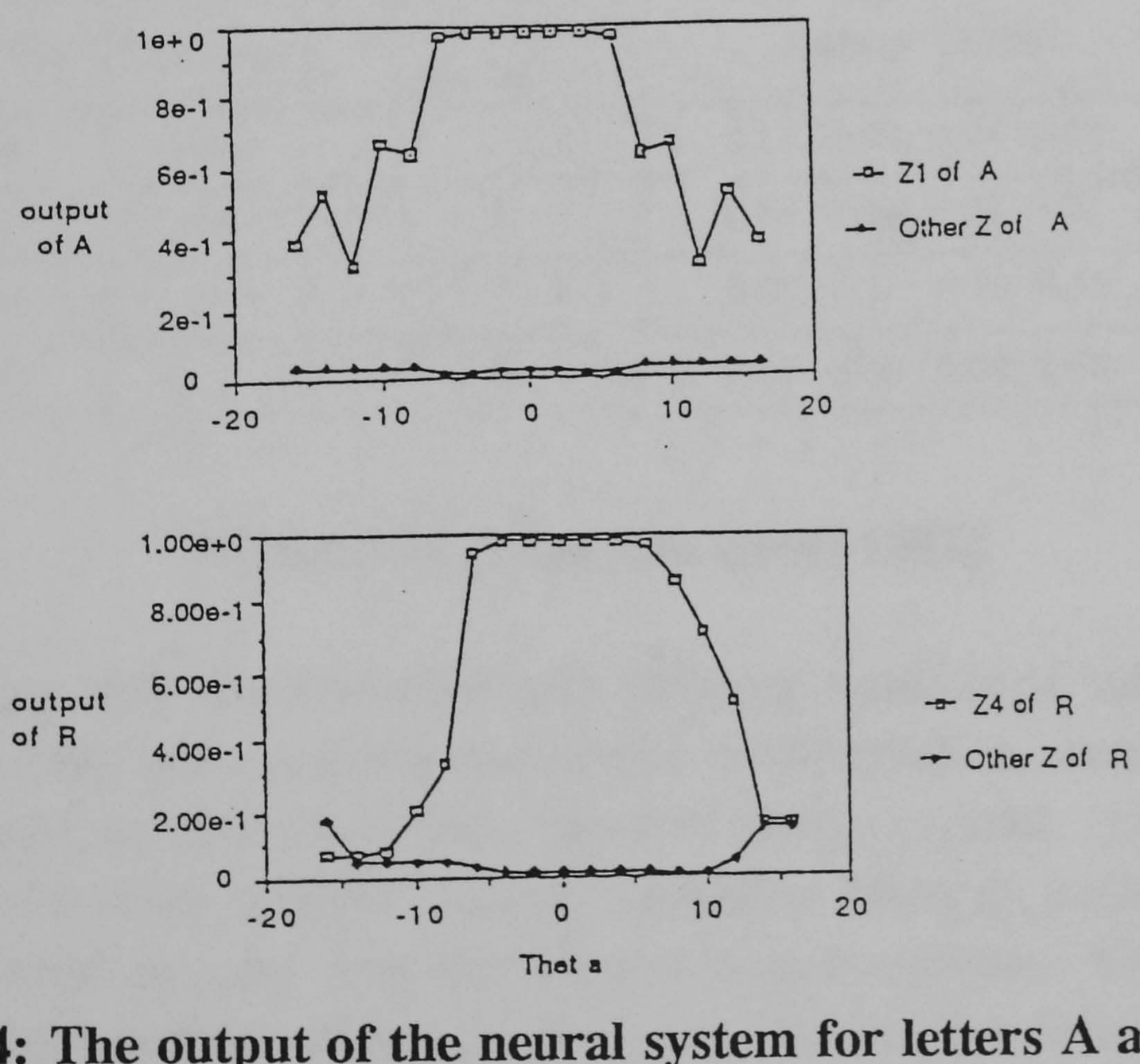


Figure 4.14: The output of the neural system for letters A and R, while the rotational angle Θ changes. Z is the output vector of the network, and Z_1 denotes the output of the first output layer processing element [Yang and Guest 1987].

letter	Target Output (z_1, z_2, z_3, z_4)	Actual Output (averaged)
A	(1 0 0 0)	(0.96 0.03 0.02 0.04)
T	(0 1 0 0)	(0.03 0.93 0.05 0.05)
H	(0 0 1 0)	(0.02 0.02 0.94 0.07)
R	(0 0 0 1)	(0.01 0.03 0.05 0.95)

Figure 4.15 [Yang and Guest 1987].

letter	θ(degree)	Target Output (z_1, z_2, z_3, z_4)	Actual Output
A	290	(1 0 0 0)	(0.95 0.08 0.01 0.08)
T	98	(0 1 0 0)	(0.09 0.92 0.02 0.01)
H	35	(0 0 1 0)	(0.02 0.01 0.90 0.10)
R	310	(0 0 0 1)	(0.01 0.09 0.03 0.92)

Figure 4.16 [Yang and Guest 1987].

The same task was also tried with different numbers of hidden neurons. It was noted that more neurons would give better result in terms of accuracy and the neural network could learn faster in terms of cycles. But less hidden neurons could allow a wider angular separation between training numbers. 'This happened because that less hidden neurons reduced the freedom of internal representations of input patterns that were mapped from a higher to a lower dimensional space' [77]. The authors also noticed that a higher constant value for α, about 3 or 5, in the sigmoid function would improve the convergence speed of the network. A larger α should mean a sharper curve

for $f(x)$. When the authors though, presented an untaught pattern to the neural network it considered that as a learned pattern and consequently it send out a wrong signal.

4.5.3 Neural System Evaluation.

Back-propagation is not guaranteed to find the global error minimum during training, but only the local error minimum. This situation often leads to severe oscillations in the changes of weights during training and can lead to the abandonment of the training at that point. Several questions must be answered in order to solve this problem. They would concern the proper selection of a number of hidden layer processing elements, the size of the learning rate parameters, and the amount of data necessary for creating the proper mapping. The generally good performance found for the back-propagation algorithm is somewhat surprising considering that it is a gradient search technique that may find a local minimum in the least mean-square cost function instead of the desired global minimum. Suggestions have been made in order to improve the performance and reduce the occurrence of local minima. They included the introduction of extra hidden units, lowering the gain term used to adapt weights, and making training start with different sets of random weights. The problem of local minima normally corresponds to clustering two or more disjoint class regions into one. This can be minimised by using multiple starts with different random weights and a low gain to adapt weights. Although various methods have been used to improve the learning time for the back-propagation neural network, it is generally accepted that a major limitation for this type of neural system is the need for presenting the training data set numerous times, with a number been in many cases in the order of thousands of times.

The number of studies and applications of back-propagation artificial neural network have grown rapidly. Extensive analysis has centered in areas like image processing, speech processing, temporal processing, prediction

and optimisation, diagnostics, control, character recognition, knowledge processing, text and sentence processing, and signal processing. In general, back-propagation's strengths include its ability to store many more patterns than the number of the input layer dimensions, and its ability to acquire arbitrarily complex nonlinear mappings. Back-propagation's limitations are its extremely long training time, its encoding requirements and the inability to know how to precisely generate any arbitrary mapping procedure. Finally the back-propagation's learning laws suffer from the same basic problem: they move downhill in short jumps. Thus, even if the network's initial weight vector lies within the attractive basin of a global minimum of the error surface, getting to the bottom can take many jumps. Furthermore, the magnitude of the gradient vector gets smaller when a shallow area is reached, thus shortening the jumps that are taken, which is exactly the reverse of what is needed. Hecht-Nielsen [78] commented that the back-propagation error surfaces, 'because of combinatoric permutations of the weights that leave the network input-output function unchanged, these functions typically have large numbers of global minima. This causes the error surfaces to be highly degenerate and to have numerous 'troughs'. Secondly, error surfaces have a multitude of areas with shallow slopes in multiple dimensions simultaneously. These typically occur because particular combinations of weights cause the weighted sums of one or more hidden layer processing elements to be larger in magnitude. When this occurs the output of that processing element is insensitive to small weight changes, since these simply move the weighted sum value back and forth along one of the shallow tails of the sigmoid function. Thirdly, it is now established that local minima do actually exist. Beyond these three facts, little is known. As the answers to our questions become available, our ability to use the back-propagation neural network will be increased even further'.

4.6 The Adaptive Resonance Theory Neural Architectures.

4.6.1. Neural Architectures Presentation.

The understanding of human memorisation arises a number of serious problems, as new memories are stored in such a way that existing ones are not forgotten or modified. This creates the dilemma how can the brain remain plastic, able to record new memories as they are created, and yet retain the stability needed to ensure that existing memories are not erased or corrupted in the process? The stability-plasticity dilemma asks : how can a learning system be designed to remain plastic, or adaptive, in response to significant events and yet remain stable in response to irrelevant events? How does the system know how to switch between its stable and its plastic modes to achieve stability without rigidity and plasticity without chaos? In particular, how can it preserve its previously learned knowledge while continuing to learn new things? And what prevents the new learning from washing away the memories of prior learning?

Conventional artificial neural networks have failed to solve the stability-plasticity dilemma. Learning a new pattern often erases or modifies previous training. If there is only a fixed set of training vectors, the network can be cycled through these repeatedly and may eventually learn them all. If however, a fully trained network must learn a new training vector, it may disrupt the weights so much that complete retraining might be required. In a case of real world data, the neural network will be exposed to a constantly changing environment and may never see the same training vector twice. Under these circumstances, a back-propagation neural network will often learn nothing; it will continuously modify its weights, never arriving at satisfactory settings. A developing theory called Adaptive Resonance Theory, or ART, suggests a solution to the above mentioned problems. ART neural networks are algorithms maintaining the plasticity required to learn new patterns, while preventing the modification of patterns that have been learned previously.

The binary adaptive resonance theory (ART1) neural system [81], was the first prototype to be designed. This is a two layer, nearest neighbour

classifier that stores an arbitrary number of binary spatial patterns using competitive learning. ART1 has the topology shown in Figure 4.17. It is an unsupervised learning, feedback recall artificial neural network. It has inter layer feedback connections between the input and the output layer processing elements that facilitate a resonation upon proper match between encoded and input patterns. The input layer processing elements accept inputs from the environment while each one of the output layer processing elements represents a pattern class. During operation the output layer processing elements employ an invisible competition that is used to choose the proper class for the presented input.

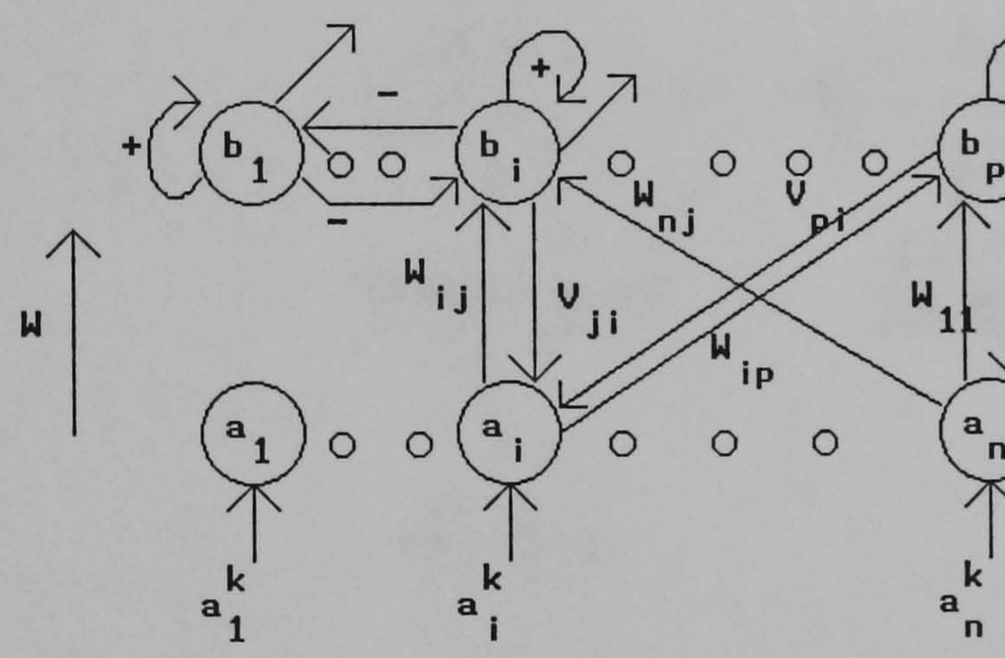


Figure 4.17: Topology of the ART1 artificial neural system.

The adaptive resonance theory (ART2) neural system [82], was introduced later and it is an extension of the ART1 neural network. ART2 is a two layer, nearest neighbour classifier that stores an arbitrary number of analogue spatial patterns using competitive learning.

4.6.2 Research Reports.

Among the many research papers that are investigating the properties of either ART1 or ART2 artificial neural networks particularly interesting is a

paper by Carpenter and Grossberg [83]. The importance of this specific paper is the fact that it introduces the concept of invariance during rotation, translation and scaling in pattern recognition. In [83], Carpenter and Grossberg exploit the properties of the ART1 and ART2 neural architectures using the system of Figure 4.18.

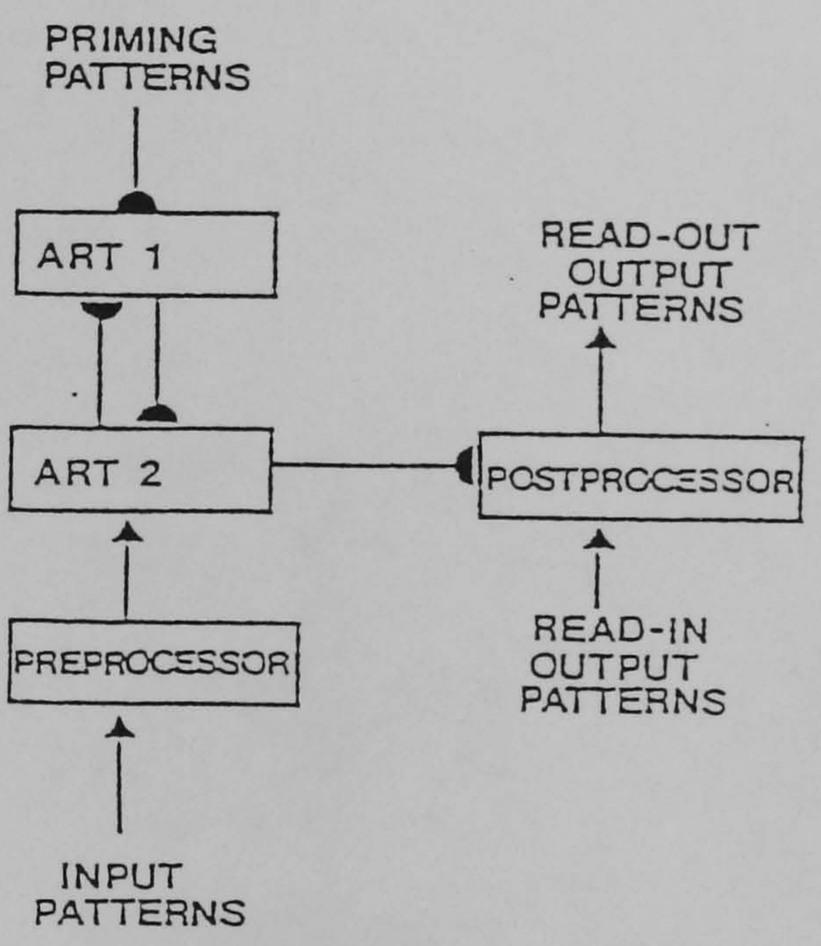


Figure 4.18: Outline of a self-organising invariant pattern recognition and recall architecture [Carpenter & Grossberg 1987].

As the authors [83] commented that, 'the overall circuit design embodies, in a primitive way, an intentional learning machine in which distinct cognitive, homeostatic, and motor representations are self-organised in a coordinated fashion'. Before the input pattern was processed by the invariant filter, the pattern to be recognised was detached from the image background. This was accomplished, by using a range detector focussed at the distance of the figure, and detaching the figure from contiguous ground by spatially intersecting the range pattern with a pattern from another detector that was capable of differentiating figure from ground. 'If the figure derived

in this way is noisy and irregular, the vigilance parameter may be set at a value that adjusts for the expected level of noise fluctuations generated by the imaging devices' [83]. In addition, as in Figure 4.19, the figure boundary may be extracted, edge linked and normalised using a boundary segmentation process. The completed boundary can then serve as the input pattern to ART2 [Figure 4.20].

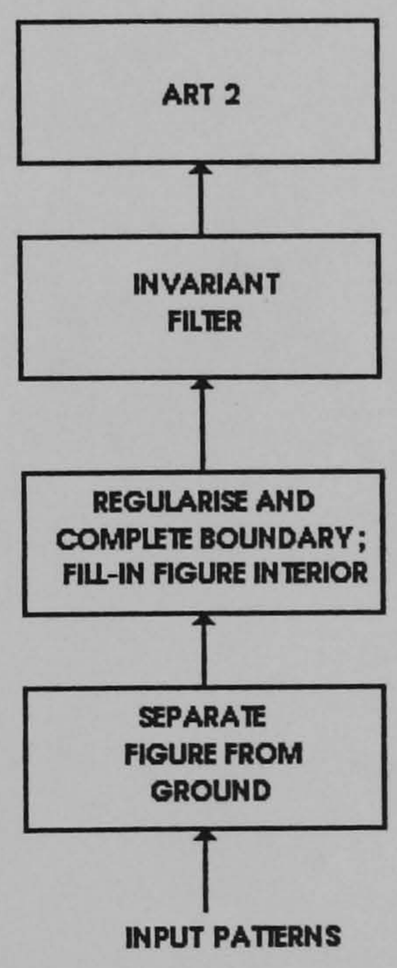


Figure 4.19: A possible pre-processor for the architecture [Carpenter & Grossberg 1987].

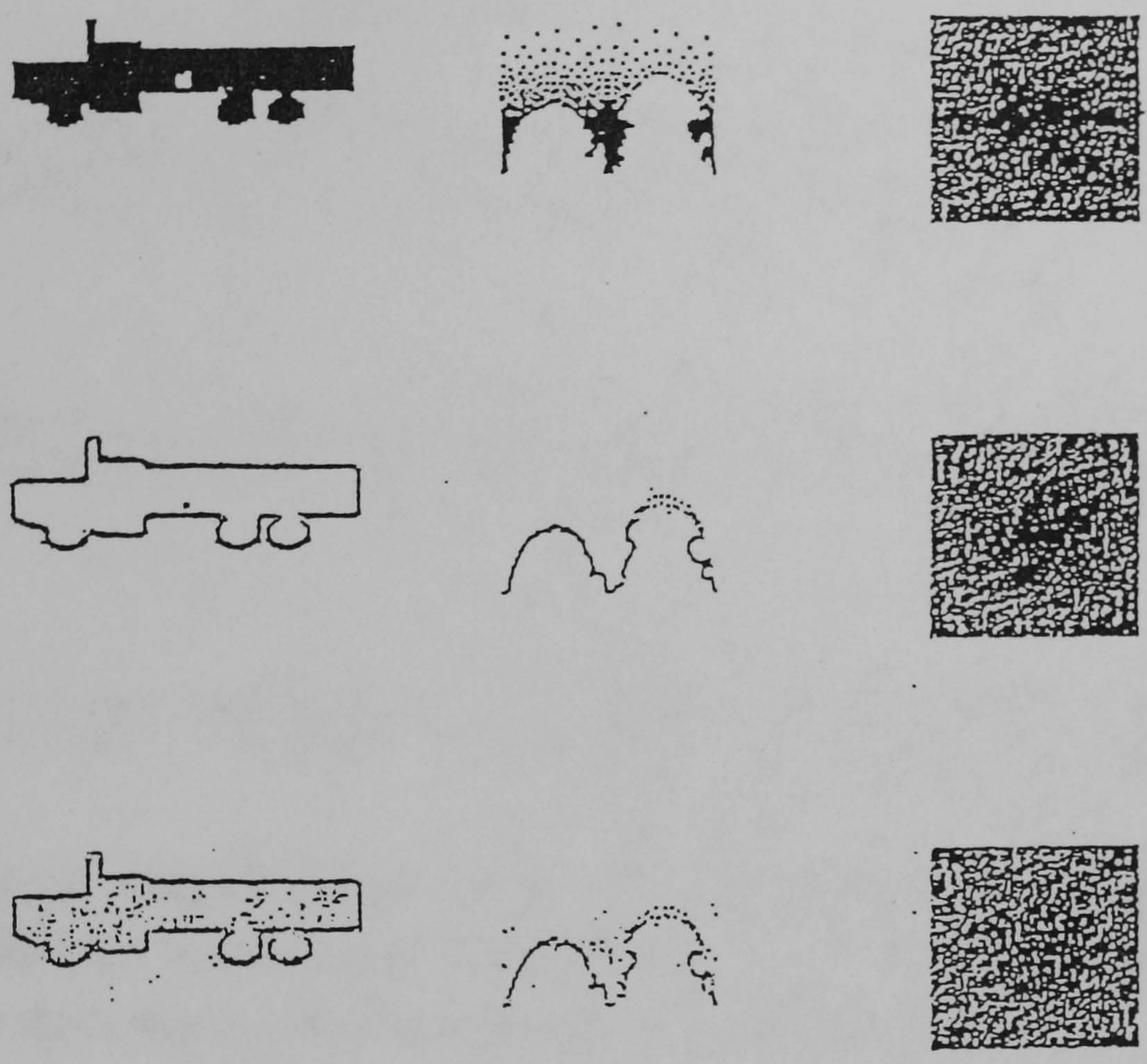


Figure 4.20: The left column depicts input images, the middle column log-polar maps of the image, and the third column Fourier transforms amplitudes of the maps, which input to ART2. The first image in row 1 is transformed into that in row 2 by choosing the maximally activated oriented mask at each position. The first image in row 3 is derived in the same way from a noisy version of the original figure [Carpenter & Grossberg 1987].

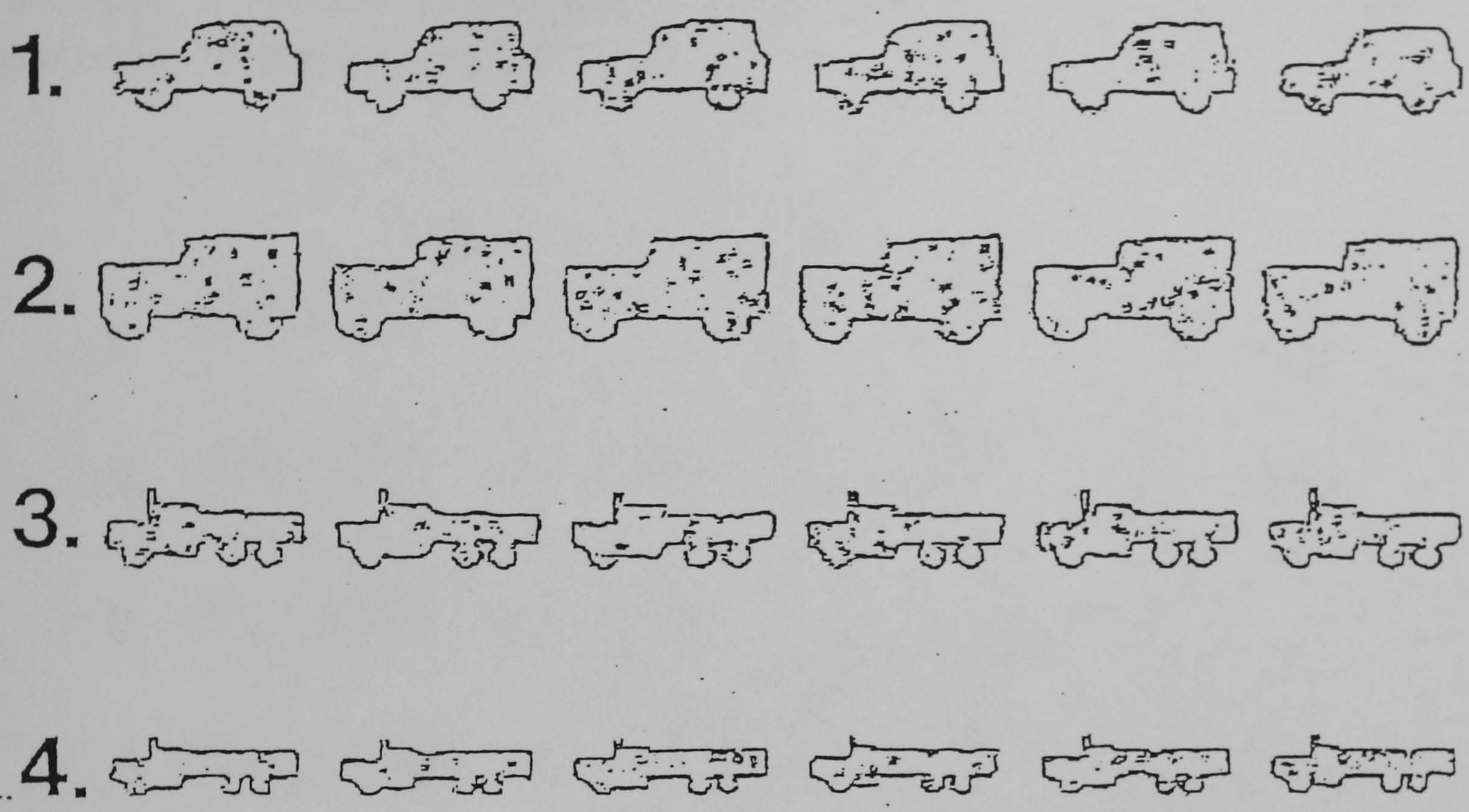


Figure 4.21: Classification of 40 noisy exemplars of 4 trucks by ART2 into 4 correct categories. Noise level was 10%. Only six trucks per category are displayed [Carpenter & Grossberg 1987].

Figure 4.21, describes the correct classification of 40 noisy exemplars of 4 trucks into 4 categories. Pairs of trucks were chosen to be very similar to one another. Figures 4.22, and 4.23 describe simulations of invariant pattern recognition. Trucks were translated, rotated, and shrunk with respect to each of 4 prototypes, and then subjected to noise. Figure 4.22 describes a consistent classification in 5% noise of 32 exemplars (eight from each prototype) into 5 consistent categories. Exemplars of one truck type were split into two categories. Figure 4.23 describes a consistent classification in 10% noise of 32 exemplars into 7 consistent categories. Exemplars of one truck type were split into four categories. These results were derived in the initial runs using very coarse boundary filtering techniques and preliminary choice of parameters. 'Inspection of the last three rows of Figure 4.22 and the last five rows of Figure 4.23 illustrates how well the architecture has

managed to separate subtle differences, obscured by image transformation and noise, of this pair of trucks' [83].

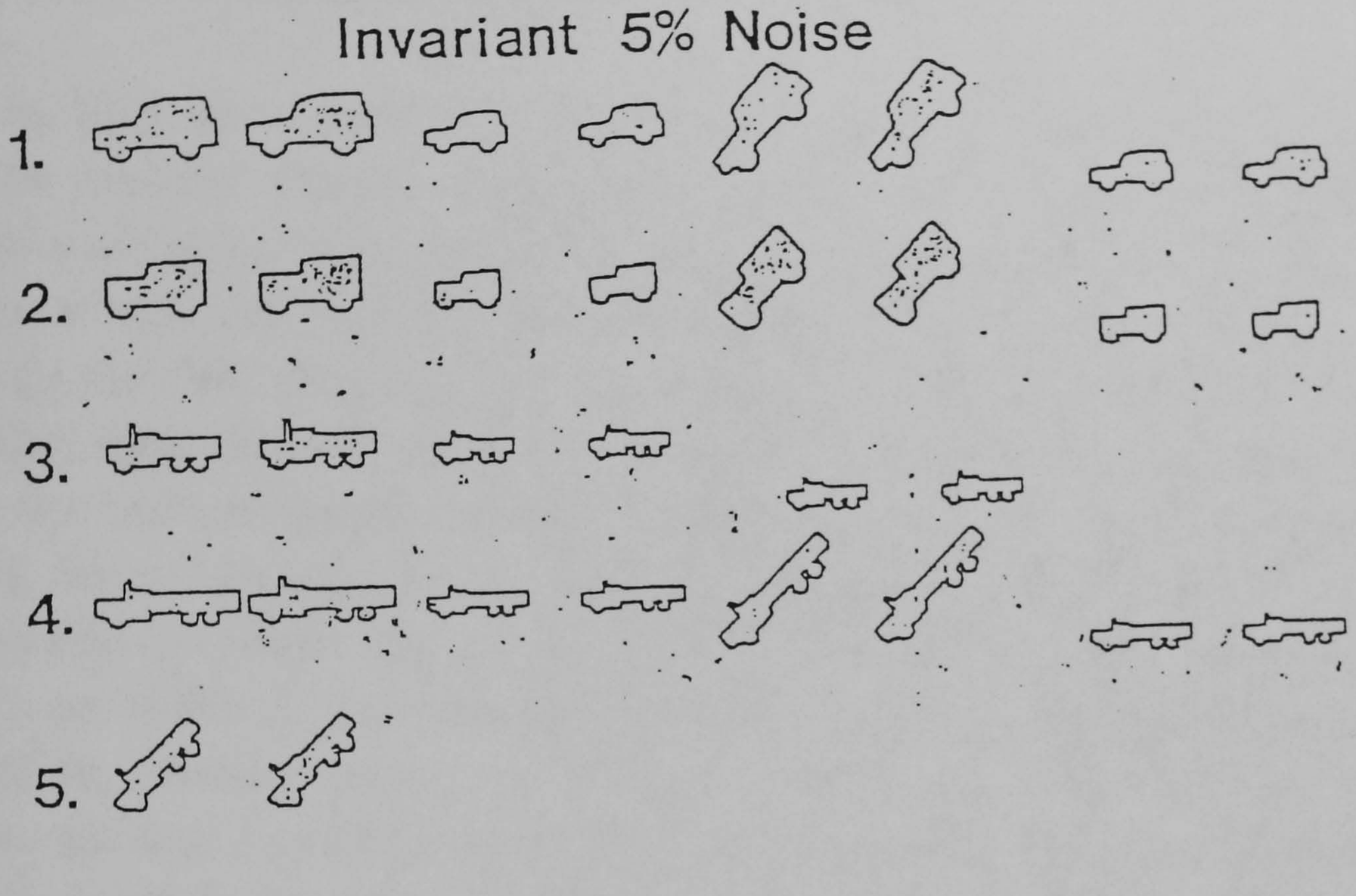


Figure 4.22: Invariant classification of 32 noisy exemplars of 4 rotated, translated and contracted trucks by ART2 into 5 consistent categories. Rows (1), (2) (3,5) (4). Noise level 5% [Carpenter & Grossberg 1987].

Among the various published papers that followed the above paper is an abstract by Rak and Kolozy [84]. In that short abstract the authors commented that they, 'have assembled a computer simulation of classical preprocessing and neural associative memory to produce rotation, scale and translation invariant object recognition. This work demonstrates robust classification among similar object silhouettes, with varying amounts of noise and object obscurity. ART was trained with four pairs of similar 0% noise objects to correctly form eight recognition categories. Subsequent testing on 10 to 60% noise objects demonstrated correct ART classification of all objects up to 50% noise. Further testing with the objects partially obscured

demonstrated correct recognition up to 20% obscurity and indicated that object edge obscurities represented the most difficult recognition problem'.

4.6.3 Neural Systems Evaluation.

In ART neural systems, the order of search and learning itself, depends on the learning history. Unfamiliar external input patterns automatically create a self adjusting search for the best matching class; if no matching class exists, a new one will be created. Familiar external input patterns access directly the class they belong to. In order to guarantee the direct access of a familiar input pattern, under any circumstances, bottom up learning is used such that both the Weber law rule - larger input patterns imply smaller LTM (Long Term Memory) traces- and the associative decay rule, -weights of connections between inactive input units and active output units are very small- are fulfilled. Learning performance by an ART network is stable with regard to arbitrary sequences of external input patterns, thereby "stable" means that a new external input pattern does not erase patterns that have been already learned. An essential reason for this stability is that the adaptation of the LTM traces, both in top down and bottom up direction, only occurs at the end of an unsuccessful search. There are two important aspects in view of this learning stability. First, if the network cannot generate an appropriate match since the external inputs come too quickly, then no learning occurs. Second, learning automatically stops if the storage capacity of the network is fully utilised. Carpenter and Grossberg, proved that this stability of learning is guaranteed if the ART neural network does not violate the so-called "2/3 rule". This rule regulates whether output units are super activated or not; more exactly; due to this rule output units may fire only if they receive inputs from two of their three input sources, namely, attentional gain control, bottom up input, and top down input.

Hoping that further detailed experimental results will be presented in the future, the presentation of the Adaptive Resonance Theory artificial neural

networks is concluded by summarising the main strengths and limitations of the presented architectures. ART neural architectures strengths include their ability to place an arbitrary number of patterns into categories of varying degrees of complexity and their stable encoding. An ART1 limitation is its restriction to binary patterns that was later corrected in the ART2 neural network system.

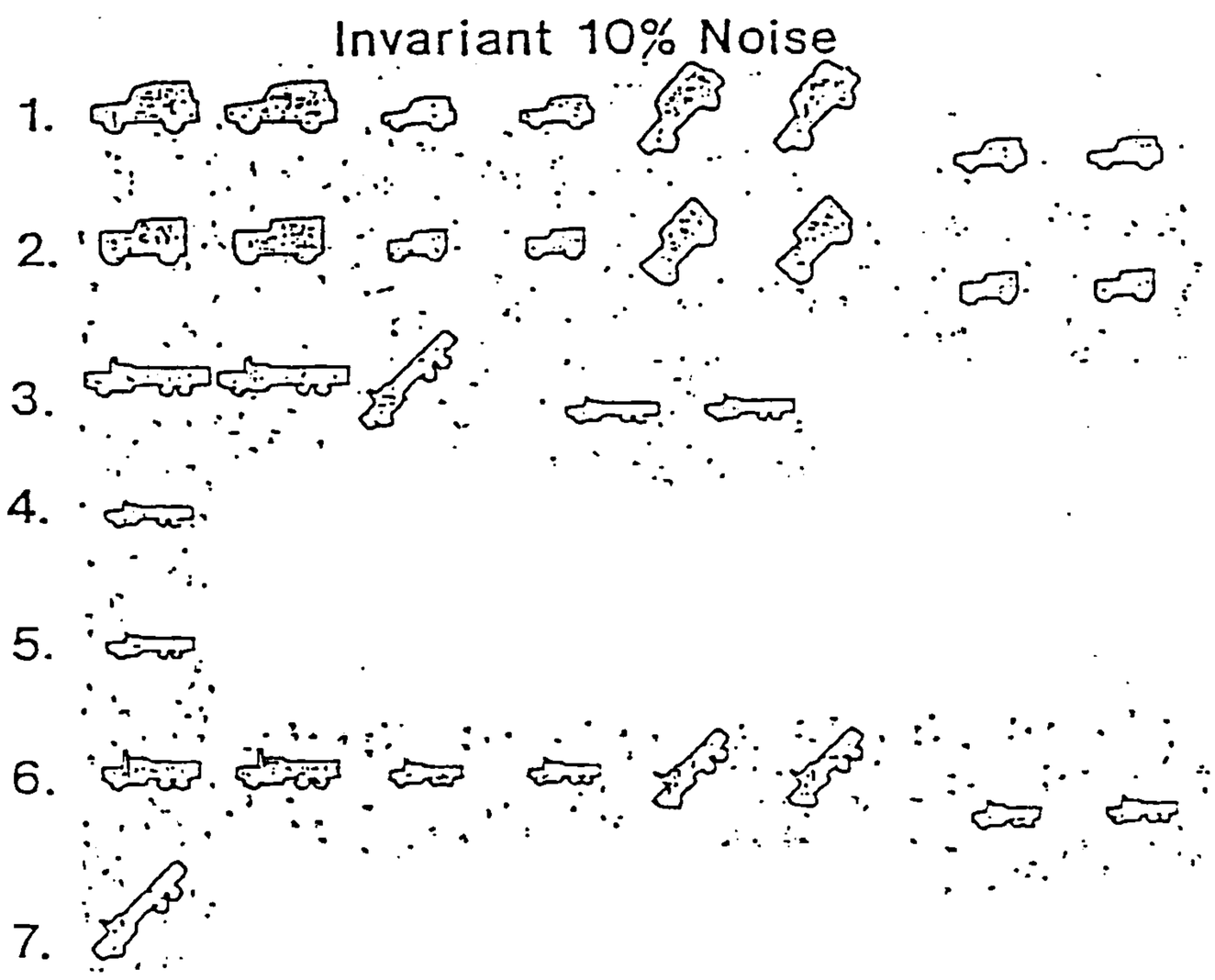


Figure 4.23: Invariant classification of 32 noisy exemplars of 4 trucks by ART 2 into 7 consistent categories. Rows (1) (2) (3,4,5,7) (6). Noise level 10% [Carpenter & Grossberg 1987].

4.7 The Hopfield Artificial Neural Network.

4.7.1 Neural Architecture Presentation.

Artificial neural networks like the Perceptron, are non recurrent neural systems, that is, there is no feedback from the output of a network to its

inputs. The lack of feedback ensures that the networks are unconditionally stable. They can not enter a state that the output wanders for ever from state to state, never producing a usable output. The topology of Hopfield's neural system [94] is shown in Figure 4.24. This recurrent associative network has n processing elements, each of which receives inputs from all the others. The input that a processing element receives from itself is ignored. All of the processing element output signals are bipolar. The network has an energy function associated with it; whenever a processing element changes state, this energy function always decreases. Starting at some initial position, the system's state vector simply moves downhill on the network's energy surface until it reaches a local minimum of the energy function. This convergence process is guaranteed to be completed in a fixed number of steps.

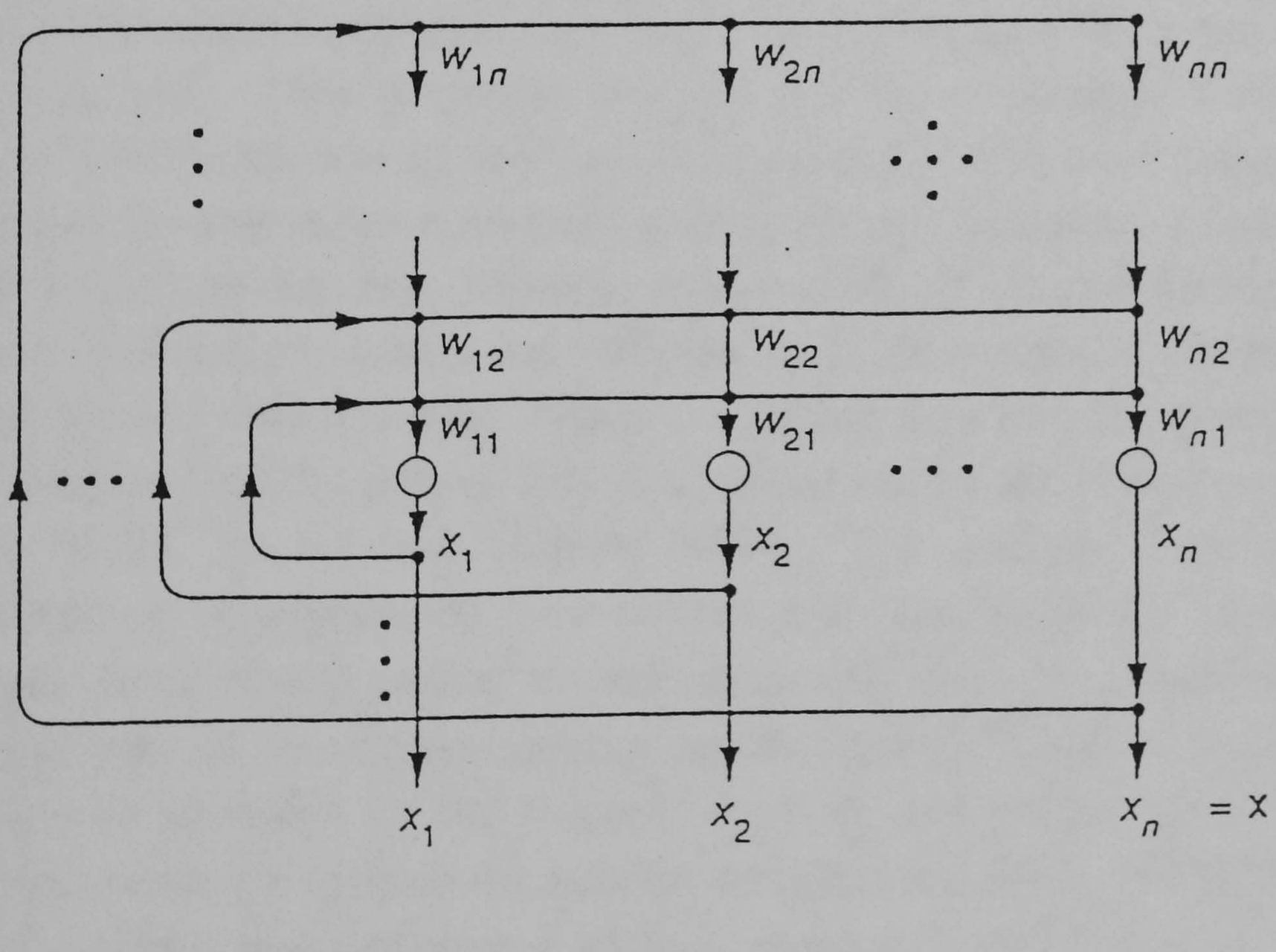


Figure 4.24: The Hopfield neural network [Hopfield 1982].

4.7.2 Research Reports.

Although Hopfield neural network has been used in a variety of research areas such as image processing, control, signal processing and pattern recognition, there has not been possible to locate and specific papers where the positional and scale invariant properties of Hopfield neural network system were clearly described. Therefore once again, attention was focused in two specific papers: one by Fuchs and Haken [95] and another by Troudet and Tatabai [96]. In [95], the authors presented an interesting formulation that can allow a scale, rotation and translation invariant pattern recognition process to be implemented with the Hopfield neural network architecture. Although they didn't present any figures regarding the time involved to accomplish pattern identification, nor did they present any data for the translation and scale invariance case, the authors though presented several images of their experimental results in relation to rotation invariance together with their mathematical theory. The authors introduced the idea of an adjacent vector and a potential function V in order to describe a test vector q moving in time. They portrayed this idea as the overdamped motion of a particle in the landscape of the potential function V and have consequently formulated the test vector q as being a function of a parameter λ (position in space). If λ is less than zero, the only stable stationary solution for the motion equation of q is zero. In this case [Figure 4.25] the system is subcritical, all patterns offered vanish in time. When λ is greater than zero the system is in a stable solution and the pattern that is a part of one of the prototype patterns [Figure 4.26], is restored [Figure 4.27]. The authors correlated the appearance or disappearance of a pattern with a variable d_k. Figure 4.28 shows the cases when λ varies. It can be seen that when λ is greater than zero the input part of an original pattern results with the pass of time, in the pattern been identified by the Hopfield network and displayed. In order to allow invariance for rotation the authors introduce a function $(v^{(k)}|T|q)$, where T is an operator that performs a scaling, rotation or translation on q. They presented a number of images showing that they algorithm performed well in a certain range of a rotation angle α [Figure 4.29].

Figure 4.25: Disappearance of a pattern [Fuchs & Haken 1988].

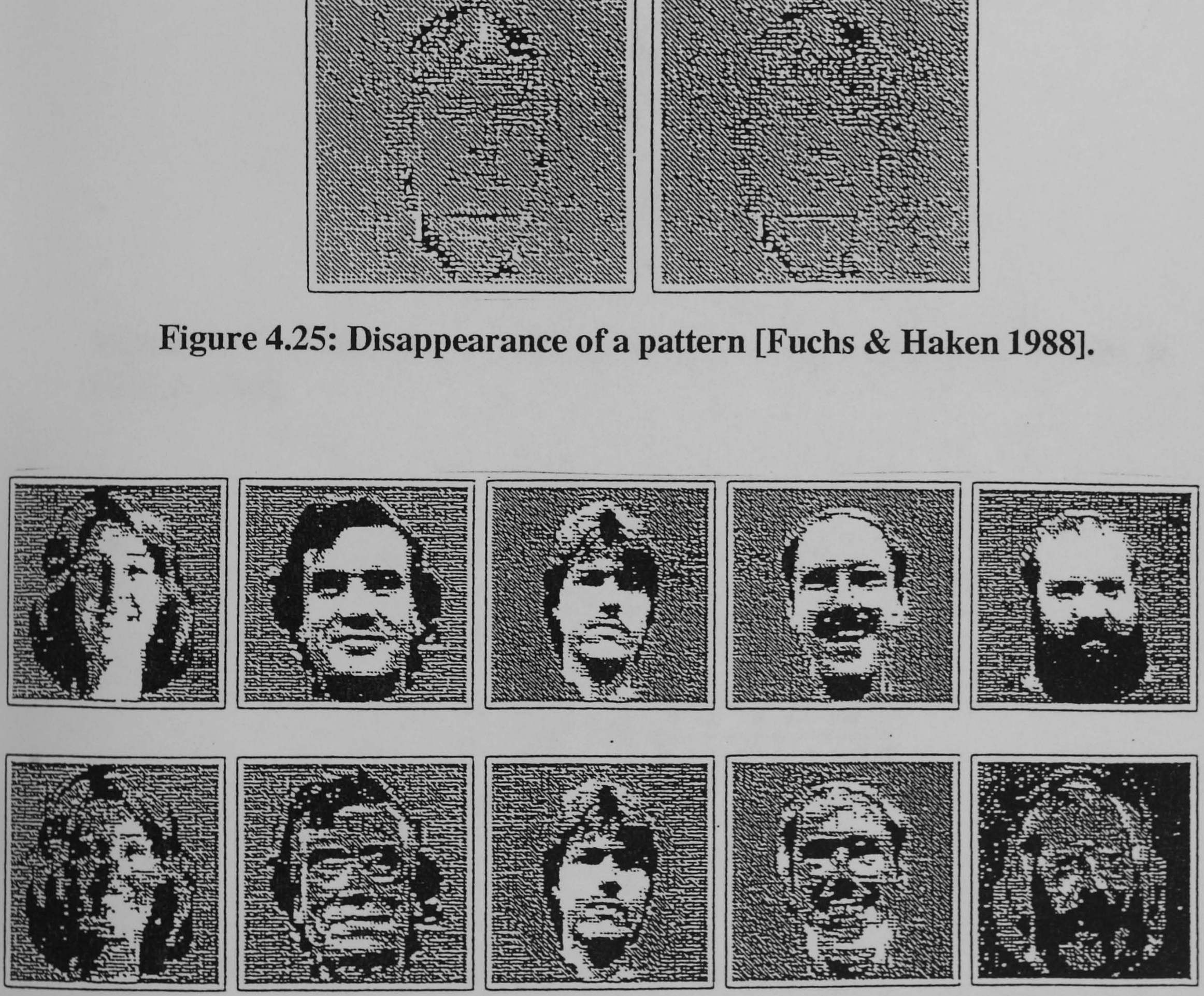


Figure 4.26: Set of basic patterns and their adjoint vectors [Fuchs & Haken 1988].

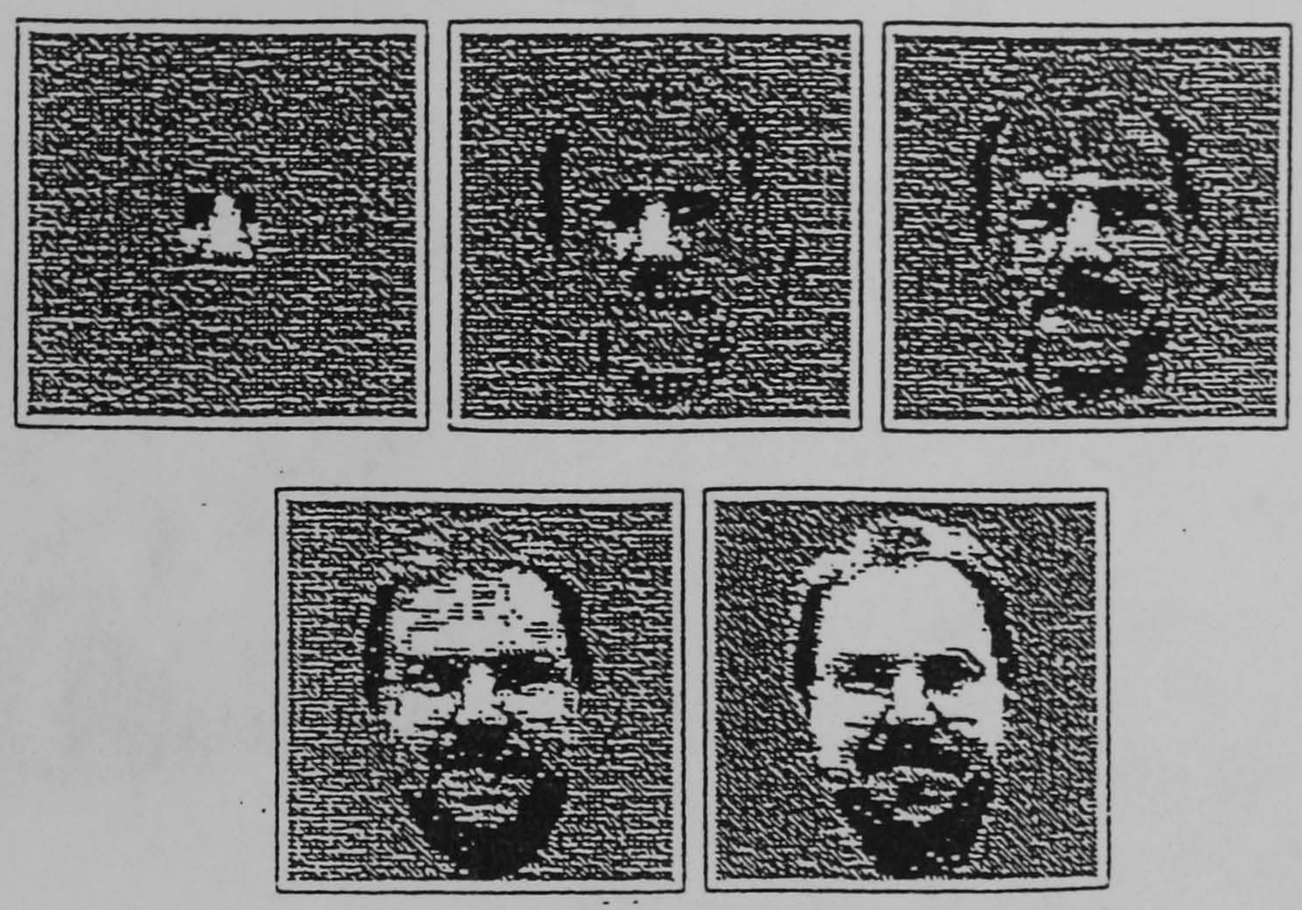


Figure 4.27: Restoring a pattern by the dynamical process [Fuchs & Haken 1988].

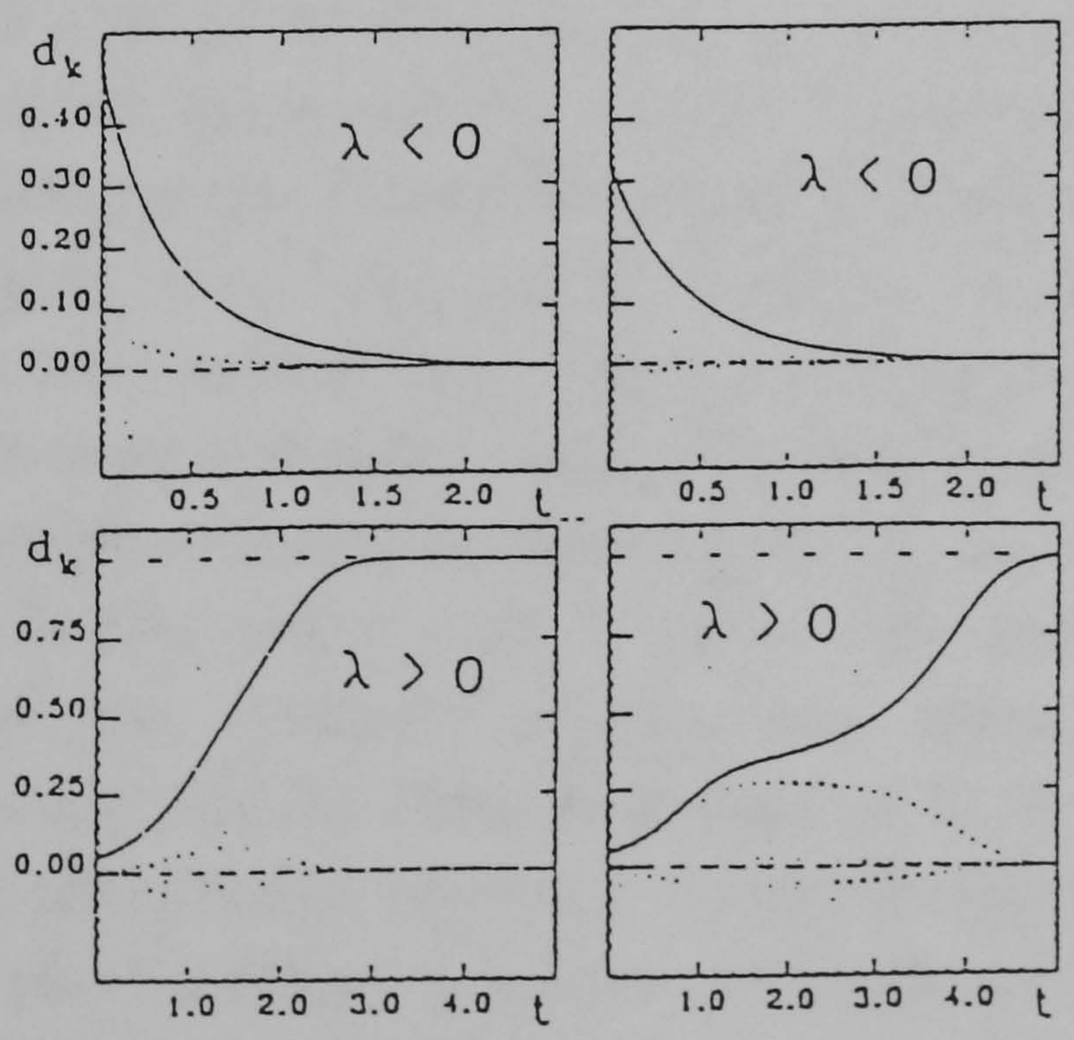


Figure 4.28: Time dependence of the expansion coefficients d_k [Fuchs & Haken 1988].

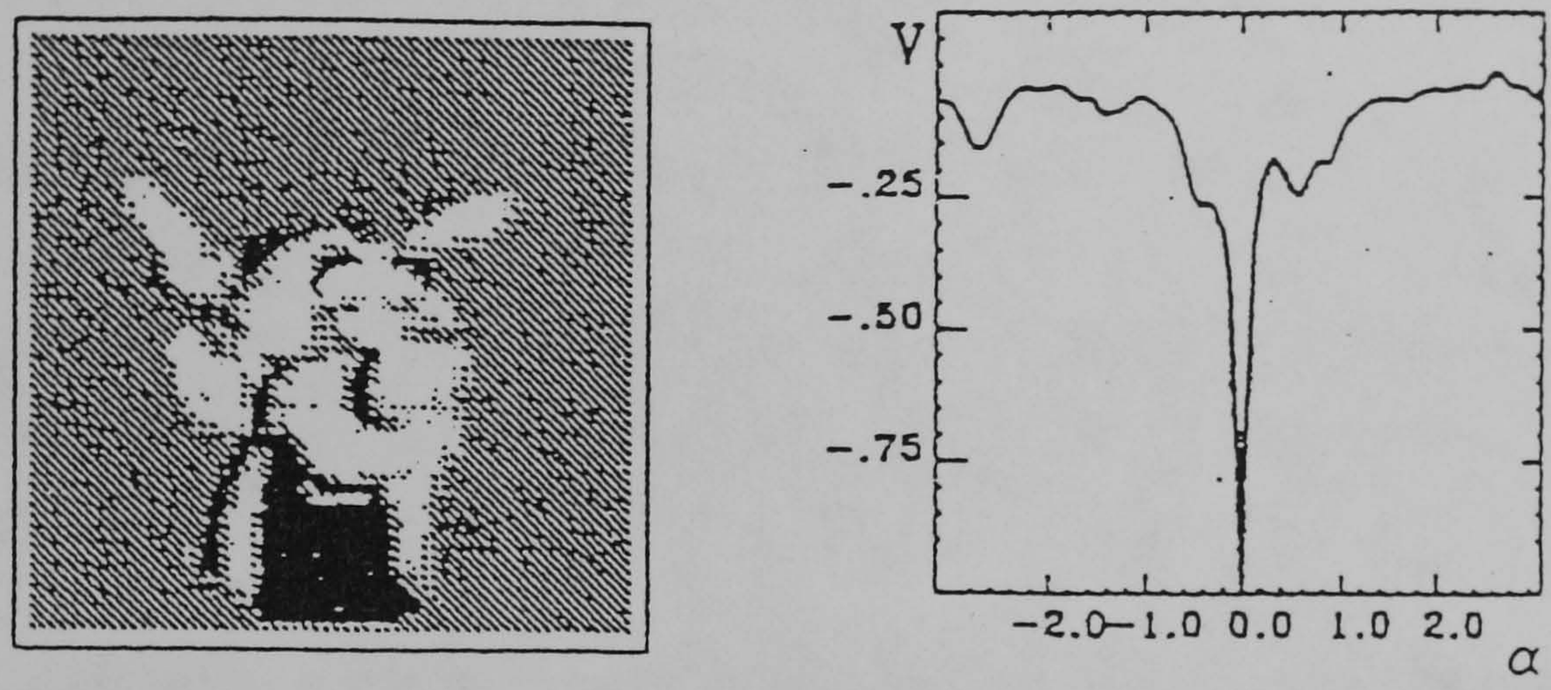


Figure 4.29: Potential under rotation [Fuchs & Haken 1988].

They concluded that their formalism 'treated images as multicomponent vectors, as well as continuous functions in space and time' [95]. In [96], Troudet and Tabatabai, presented an adaptive neural network architecture based on the Hopfield neural network that could perform segmentation in real time for mixed gray level and binary pictures. In their approach the composite picture was divided into binary and image blocks based on a dichotomy measure were computed using a Hopfield type of neural network architecture. The original picture in Figure 4.30(2A) was subdivided into 32 by 32 blocks of n=256 pixels each (16 by 16) and whose intensities adaptively defined the synaptic weights and input currents of 256 interconnected neurons. For the class separation of 16 by 16 pixel character blocks and 4 by 4 pixel image blocks, a composite picture was created by allocating to each pixel a binary value so as to indicate that its intensity is identified with the average intensity of the corresponding group of pixels. The second and third images of Figure 4.30 (2B & 2C), show the image reconstructed using the mean values at t=0 and after a computation of 200 ns

per block. The fourth image (2D) indicates the blocks that have been classified as character-blocks (dark) and those which have been classified as image blocks after a computation of 200 ns per block. 'Out of the 1024 blocks of the composite picture generated in the third image, 739 blocks were found to be of a character type and 285 of an image type. As a result, while the intensities of the pixels of the original image were given by 8 bits, only 1.32 bit per pixel was needed to generate the third image after the segmentation by the neural network. In terms of data compression it meant a 95% efficiency of the neural network "segmentor" with respect to the exact resolution' [96].

The authors concluded that 'this fairly low loss of algorithmic efficiency is by large compensated by the enhanced computational power of the neural network which is of the order 10^5 faster than the VAX 8650 where the class separation of a 256 pixel block takes approximately 120 ms'.

4.7.3 Neural System Evaluation.

Independent to its initial state is, the Hopfield neural system always reaches a stable state in a finite number of processing element update steps. It turns out, however that the Hopfield network does not always go from an initial state to the nearest stable state. Sometimes it goes to a stable state that is farther away than the nearest stable state and there is no way known to overcome this problem.

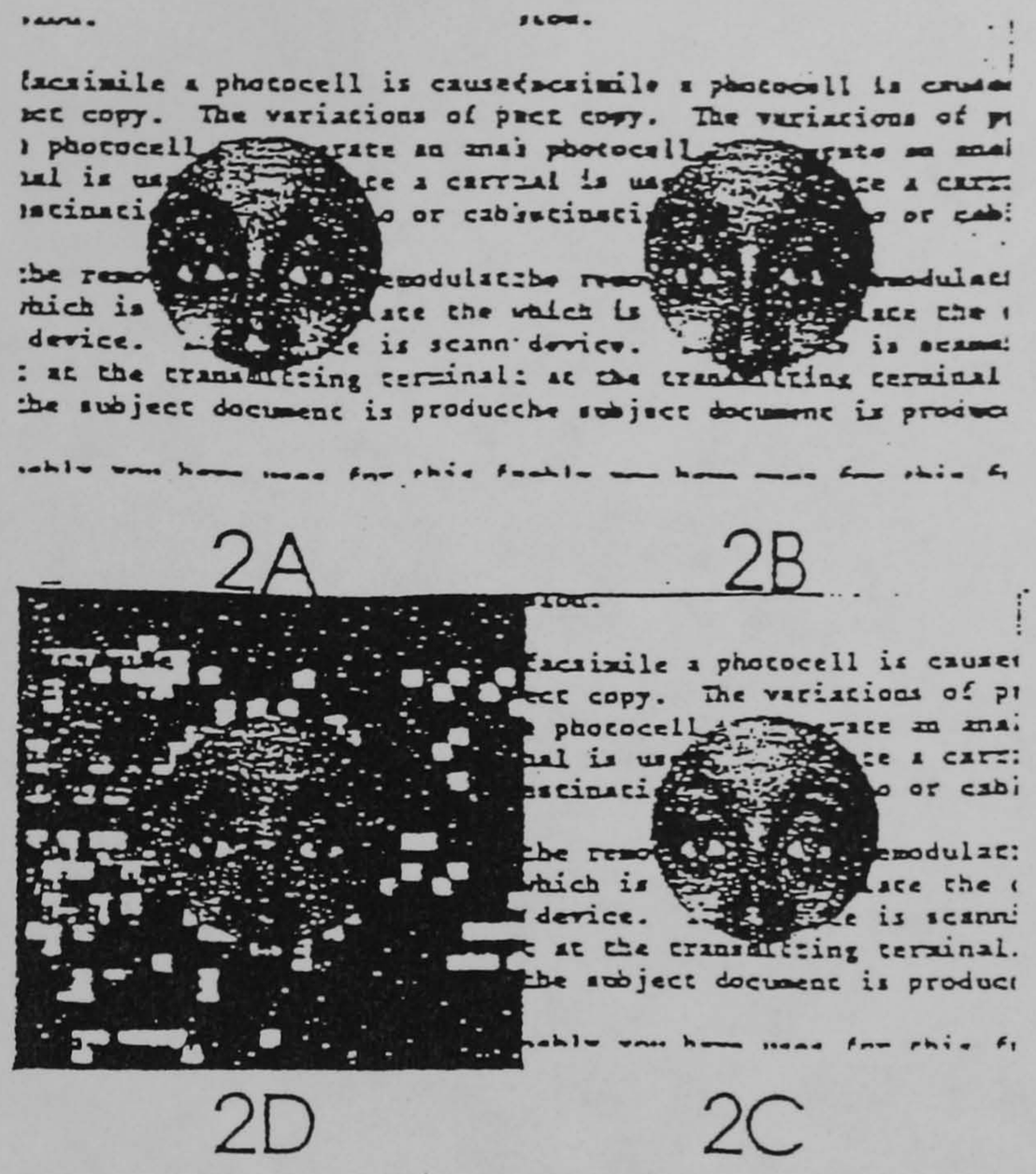


Figure 4.30 [Troudet & Tabatabai 1988].

4.8 The Counterpropagation Artificial Neural Network.

4.8.1 Neural Architecture Presentation.

Counterpropagation neural network [103], is a mapping type neural system that combines two well known algorithms; the self organisation feature map of Kohonen and the Grossberg 'Avalanche' outstar [104]. Together they possess properties not available in either one alone. A mapping type neural network is a network that self-organises to implement an approximation to a function or mapping. The Counterpropagation artificial neural network is a three layer hetero-associative, nearest neighbour pattern matcher that stores arbitrary spatial patterns using a combination of Hebbian encoding and Kohonen's learning vector quantiser. It is an unsupervised learning, feed-forward neural system, that is represented by the topology of Figure 4.31.

It is essential to notice that since Counterpropagation uses the Euclidean metric that implies that the neural network treats local distance differences on an isotropic basis. Hecht-Nielsen [103] commented that, 'while the approach can clearly compensate for positional differences in probability density, it ultimately will do a poor job of correctly representing the local probability density function values, unless changes in coordinate value in one dimension are roughly as important as changes in all other dimensions'. This implies that it is essential for the feature coordinates to be are appropriately scaled using both linear and nonlinear scaling. Another neural network based approach to this scaling problem is by mapping the feature space of a particular pattern recognition problem onto a rectangular grid using Kohonen's self organising map neural system. This approach can ensure that feature vectors images will lie on a euclidean feature and with a uniform probability density. There can be though cases where the previous approach can radically distort the geometry of the class sets.

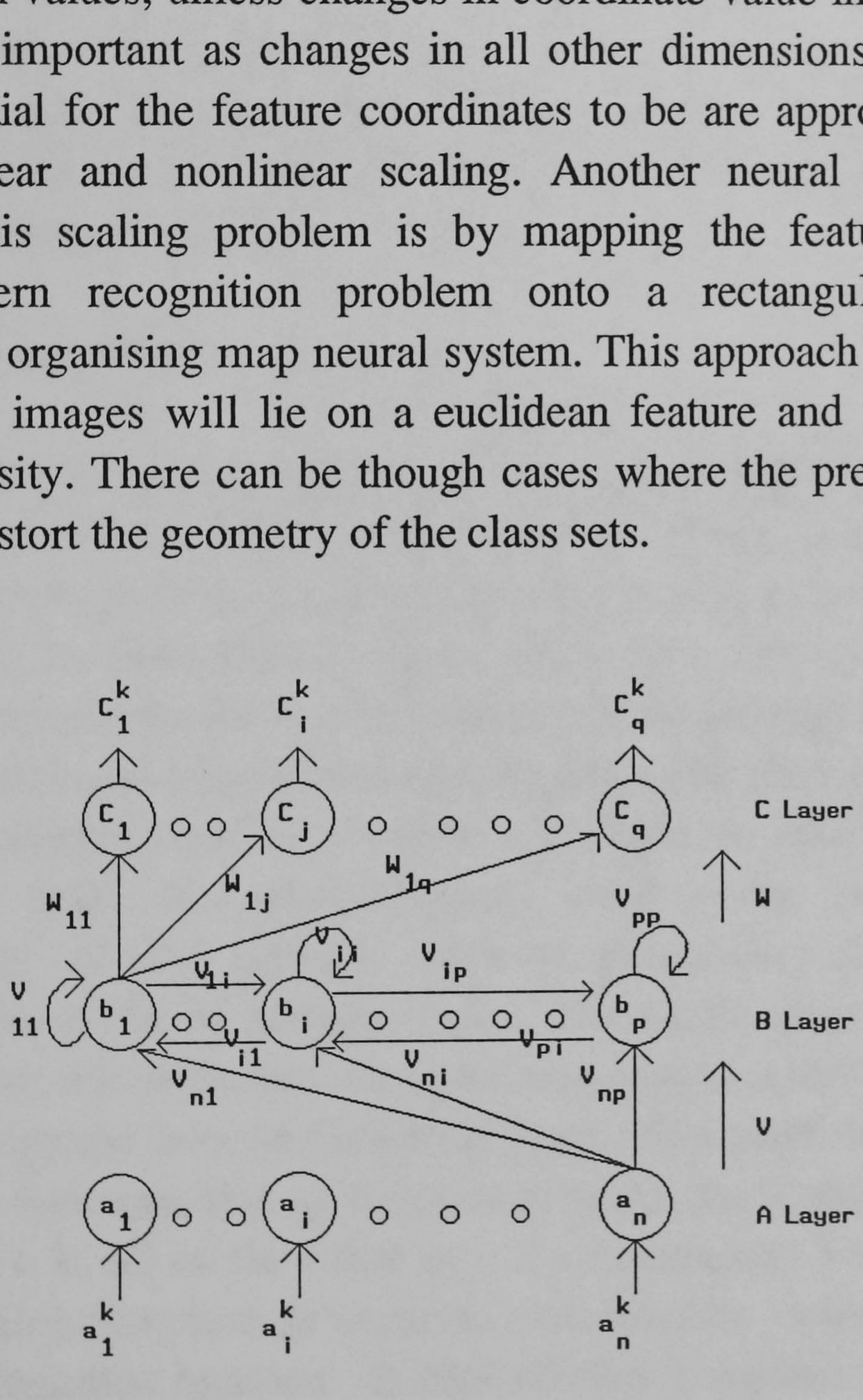


Figure 4.31: Topology of Counterpropagation.

The data vectors in the Counterpropagation neural network, self organise according to the probability density of the presented data. Counterpropagation strengths include its ability to perform nearest neighbour mappings for an arbitrary set of pattern pairs in a self-programming look up table fashion. Another strength of the Counterpropagation artificial neural system is its ability to learn nonlinear mappings. A limitation of Counterpropagation is its need for a hidden layer processing element for each output layer pattern. This drawback can be improved by allowing multiple winners and adjusting in a fashion similar to that done by slow learning version of ART neural systems.

4.8.2 Research Reports.

In [105] Hecht-Nielsen commented about the possibility of using the Counterpropagation artificial neural network for pattern recognition. As it is already known most pattern recognition problems involve the classification of an unknown multi-dimensional feature vector into one of many possible classes. In such problems the sets of feature vectors belonging to that class is reasonably smooth and simple, although the class sets may overlap . In these cases the Counterpropagation network can function much like a Bayes classifier. The basic idea Hecht-Nielsen [105] notes, 'is to equip the Grossberg layer of the network with M processing elements - each representing one of the M pattern classes. During training, as each feature vector X is entered into the fan out units of the input layer of the network a class vector is entered into the Grossberg layer. This class vector has an 1 in the component corresponding to the class to which the feature vector pattern belongs and 0's in all of the other m - 1 components. Thus, the network learns the mapping from feature vector to class number - which is the essence of a pattern recognition function'. If interpolation is applied the classifier can be considered as a multi-class Bayes classifier. Figure 4.32, shows an example of how Counterpropagation neural network can be used for pattern recognition. The example involves guiding a robot arm to a gripper fixture on

a satellite. The robot arm is assumed to be attached on a upcoming space station. The arm has a camera mounted that has been approximately attached on the satellite's gripper fixture. The gripper fixture is shaped like the letter "I". The robot camera sees a scene similar to that shown in the upper left corner of Figure 4.32. The problem is to determine the orientation angle θ and the apparent length X of the gripper fixture. This will allow the robot arm to extend and rotate to the proper distance and angle in order to grasp the gripper fixture. The result was that the angle measurement portion of the problem was solved to within three degrees accuracy, using the Counterpropagation neural system in sixty hours of effort.

The image of the gripper fixture was 32 by 32 pixels in size. The first step was to threshold the image at a grayscale level of 127 and assign a value of 0 to those pixels with grayscale values at or above this level and 1 to those with lower grayscale values. This formed a binary image vector having 1024 components. This vector was entered directly into a Counterpropagation network with 12 Kohonen units and 2 Grossberg units. For the purposes of the experiment all images were taken from the same range. 'As each image was entered, the correct θ value was supplied to the $\dot{\theta}$ output unit of the Grossberg layer; the other Grossberg layer output unit was used for $\dot{X}$ and was not used in these experiments' [105]. The result was that, after a period of training, an arbitrary gripper image could be entered and the network would identify the gripper fixture's orientation θ to within plus or minus three degrees of accuracy. The network has used interpolation with two winners to achieve this result. 'Counterpropagation networks can be used for pattern recognition situations in which a quasi-Bayesian classifier is desired or where template matching and template interpolation is desired' [105].

4.8.3 Neural System Evaluation.

Hecht-Nielsen commented that, 'methods such as Counterpropagation

that combine paradigms in building block fashion may produce networks closer to the brain's architecture than any homogeneous structure' [105]. It does indeed seem that the brain cascades various specialised modules to produce the desired computation. The Counterpropagation network functions as a look up table capable of generalisation. The training process associates input vectors with corresponding output vectors. These vectors may be binary or continuous. Once the network is trained, application of an input vector produces the desired output vector. The generalisation capability of the network allows it to produce a correct output even when it is given an input vector that is partially incomplete or partially incorrect. This makes the network useful for pattern recognition, pattern completion and signal enhancement applications.

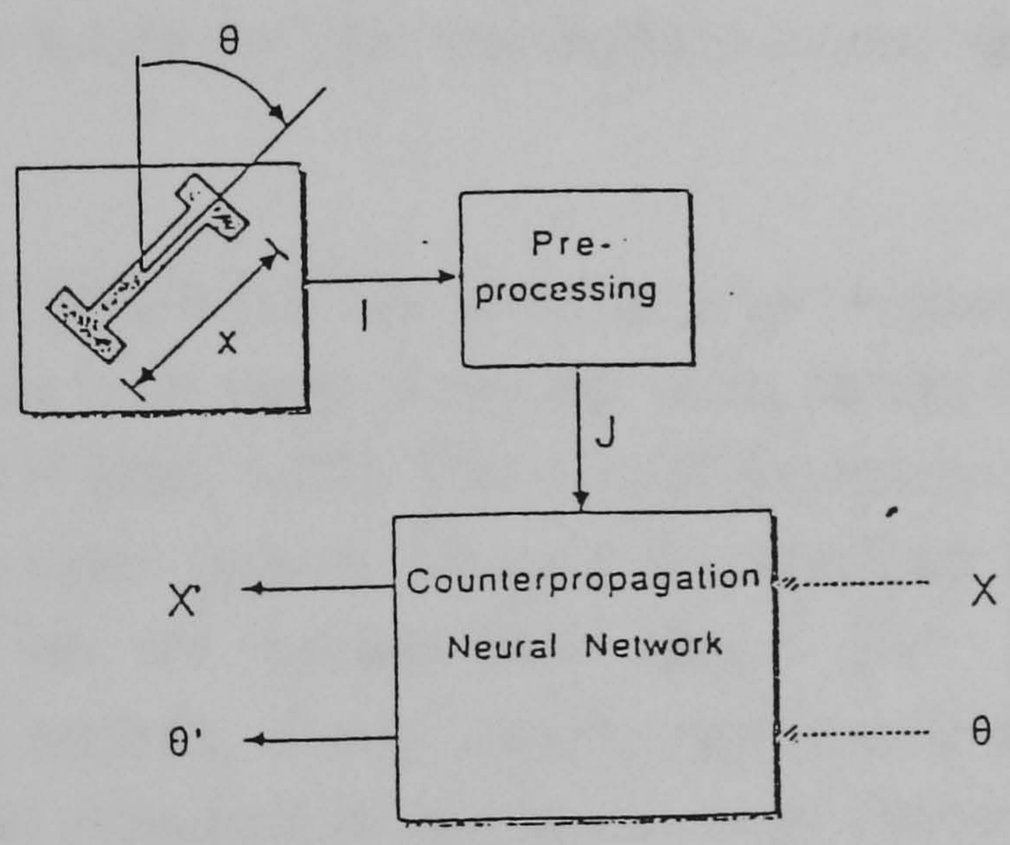


Figure 4.32: A Counterpropagation network used to translate an image directly into pattern measurements X and θ [Hecht-Nielsen 1988].

4.9 The ADAM Artificial Neural System.

4.9.1 Neural Architecture Presentation.

Willshaw's associative net is a distributed mapping memory with binary

inputs that evokes an association between an input pattern and the required output pattern. It can be visualised as a matrix of initially unlinked wires, one horizontal wire for each of the input bits and a number of vertical wires one for each output bit. During teaching each input is presented along the bit pattern with which it is to be associated. This pattern appears on the vertical wires, whilst the input appears on the horizontal wires. A weight of 1 (link) is set in the memory matrix wherever an active vertical wire crosses an active horizontal wire. This simple learning rule uses only binary links, so that once formed a link remains in place. If a new pattern requires a link in an empty position, one is formed, but if the position already has a link, then nothing is altered. Learning takes place in one pass without the need of an iterative process. In recall the input pattern is presented and the output pattern is calculated by summing the number of links in each column that are activated by the input. These totals are then thresholded to recover the original binary pattern.

ADAM neural system [6] has been an improvement on the Willshaw's net. ADAM's matrix is in many ways the same as the Willshaw's net but is split into two parts [Figure 4.33]. The reason for splitting is the introduction of an intermediate class pattern. Overall the memory still associates input with output but via an intermediate stage. The introduction of the intermediate class pattern allows much more accurate recall, since the characteristics of the class pattern can be precisely determined.

The operation of ADAM memory is as follows. The input data (binary or non-binary) are fed to a set of logical decoders. These apply a simple binary decoding to the tuples of size N fed from the input array. The form of mapping from the input array to the decoders effects the operation of the system. 'The careful mapping of the input lines to the decoders provides varying degrees of coupling between the knowledge fields. Furthermore, the effect has important consequences for the way weights are paged in to the memory in implementations of the system. The decoders partially

orthogonalise the input pattern to enable the use of a linear classifier to be successful on non-linearly separable problems' [6]. The classification is achieved using simple correlation matrix memories with binary weights and functions in exactly the same way as the Willshaw's neural system, except in how the stored patterns are thresholded during recall. The memory acts as a two layer network with the pattern to be recalled being placed on the output of the second memory. The first layer of units are forced to take on a pattern made up of K points set to one for every new association presented. This allows successful recall by thresholding this layer of outputs to recover exactly K elements set to one. This is then passed to the second memory that uses a threshold set to K to recover the stored associated pattern. Because both training and recall require only one presentation of the input pattern, the process is very rapid.

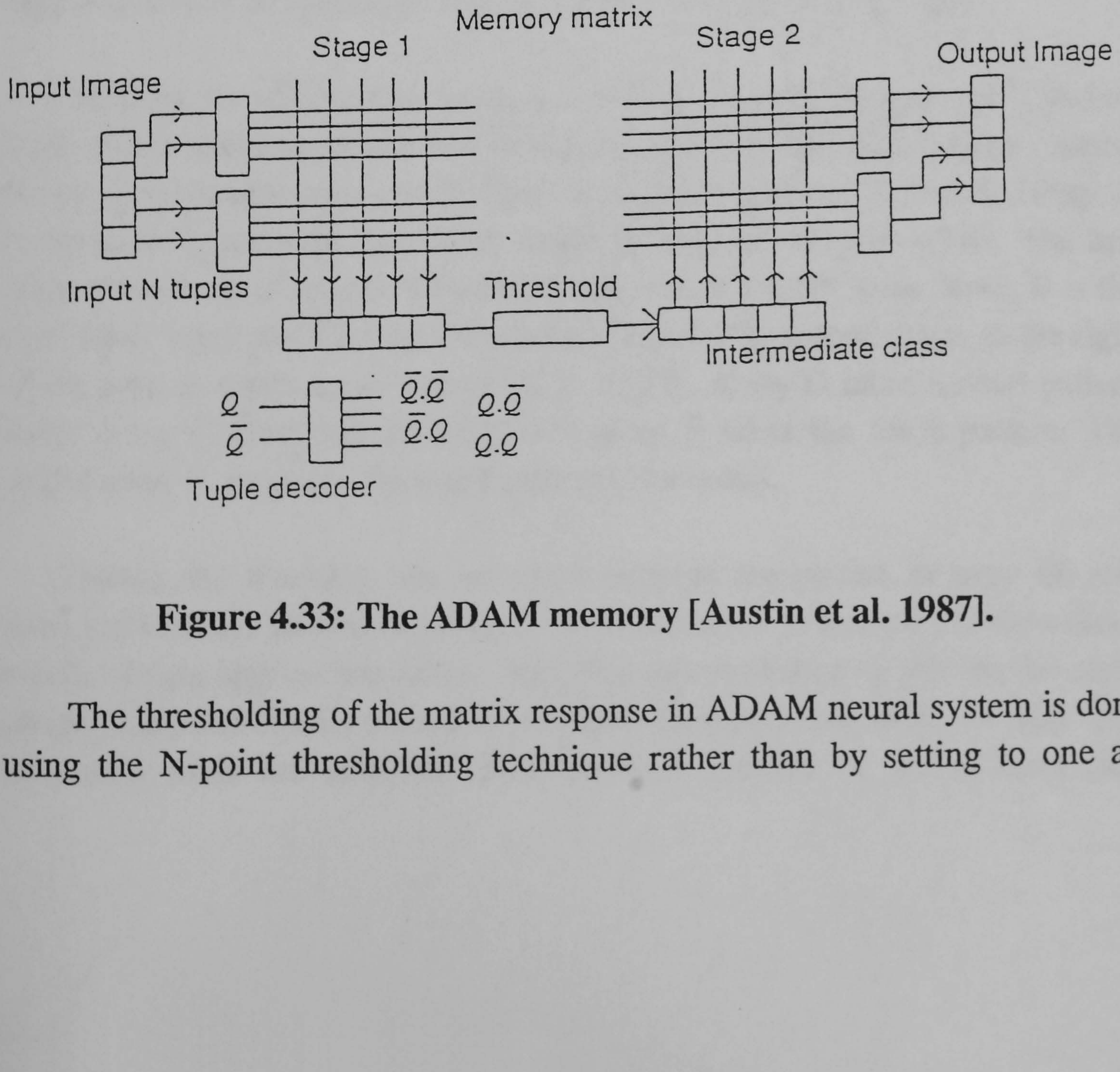


Figure 4.33: The ADAM memory [Austin et al. 1987].

The thresholding of the matrix response in ADAM neural system is done using the N-point thresholding technique rather than by setting to one all

values above or equal to a certain level, and setting the rest to zero. N-point thresholding selects the n highest values and sets those to one, setting all the remaining values to zero. This effectively provides a dynamic threshold level that is adjusted until a fixed number n of bits is recovered. This technique is more effective in recalling the associated pattern than the standard static threshold method. Furthermore, as the class pattern that is being recalled has a known number of bits set to one, the value of n can be determined. Consider the example shown in Figures 4.34a and 4.34b.

The neural system was taught using a pattern with two elements set at logical 1 (N = 2). Array E in Figure 4.34a, demonstrates the recall process using a corrupted key pattern. If a threshold is used that selects the two highest elements and sets only these to 1 then a perfect recall is obtained. The output from the memory can be considered as an identifier for the input patterns [Figure 4.34a]. In normal experiments though, two particular patterns need to be associate: the key pattern and the teach pattern.

This can be achieved by using a two stage associative memory. The first stage of the memory associates the key pattern and the class pattern, where the key pattern can cause recollection of the class pattern. The second stage is in the same form as the first. Both stages are shown in Figure 4.34b. The first stage is made up of arrays A,B, and C where A is the key array input, B is the class array input and C is the class array output. The second stage, to the right of the first, is made up of arrays D, E, and F. Array D takes a class pattern (from array C after thresholding) and array E takes the teach pattern. The output array F, contains the teach pattern after recall.

During the teaching process class patterns are placed in array D, and teach patterns are placed in array E. An association is formed between these two by setting appropriate links. 'Recall is accomplished by placing the class pattern (derived from a recall in the first memory) into array D' [6]. The horizontal rows are activated and the total number of active lines that

intersect with a link is summed for each column. The recall process forms the values shown in array F. These values are then again thresholded in order to produce the original pattern. An application of the threshold method results in the pattern in array G.

No new threshold procedure is used in this case, because the class pattern that is placed in array D is summed so as to be free from errors. To conclude, the key pattern and the teach pattern need to be associated. The key pattern is first associated with a class pattern, containing N elements set at 1. The class pattern is then further associated with the original input teach pattern, resulting in an association to be formed. The recalled class pattern is thresholded by selecting the N highest responding elements, while the teach pattern is thresholded at exactly N elements.

4.9.2 Research Reports.

In an interesting paper [106], Austin has presented a system based on ADAM neural network that can result in an invariant pattern recognition. Consider Figure 4.35 that illustrates graphically how the method operates.

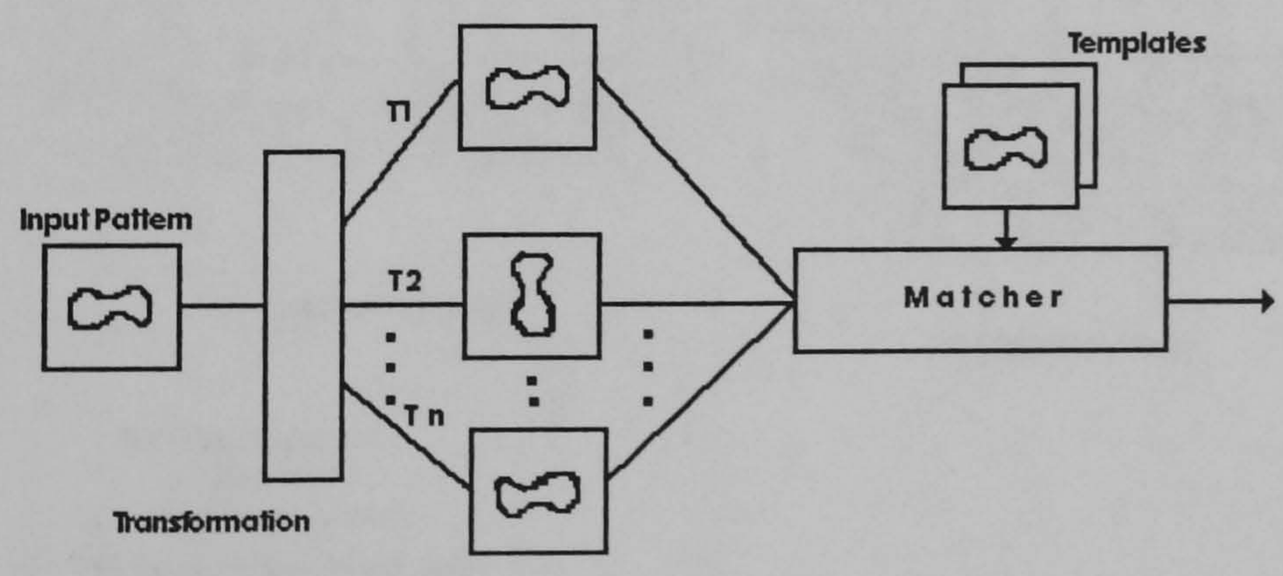
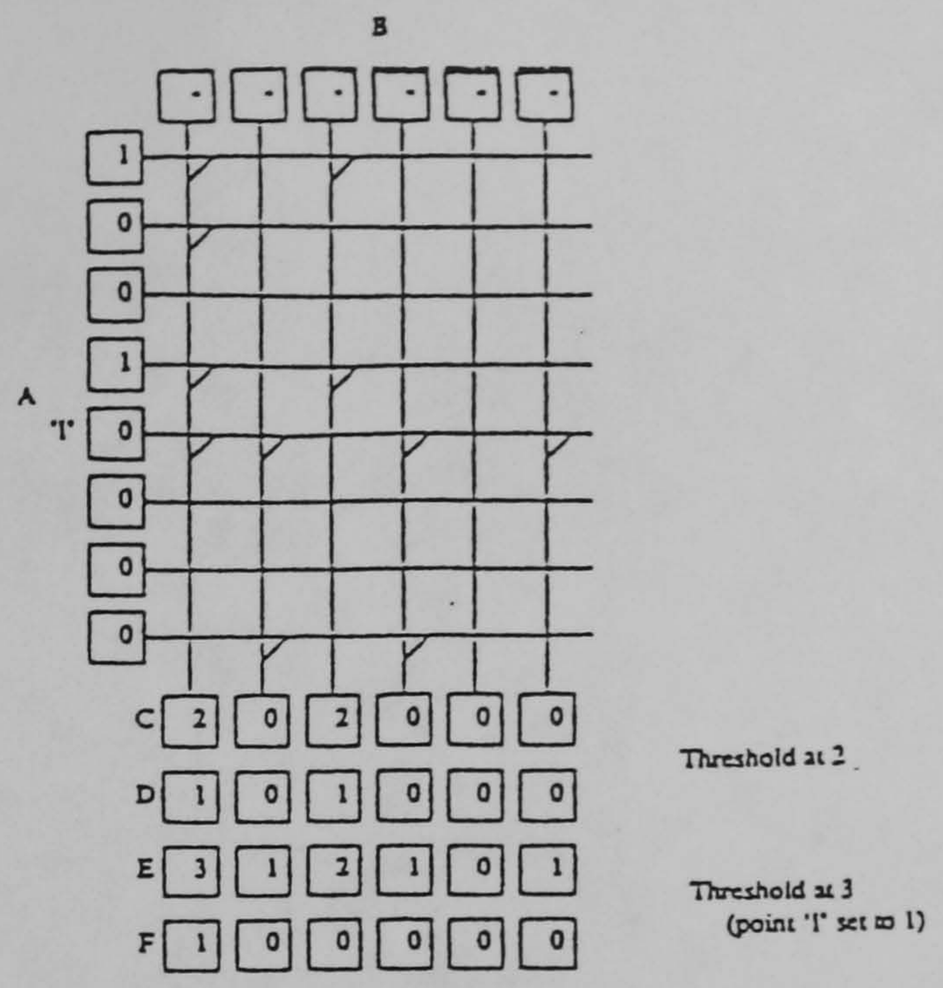
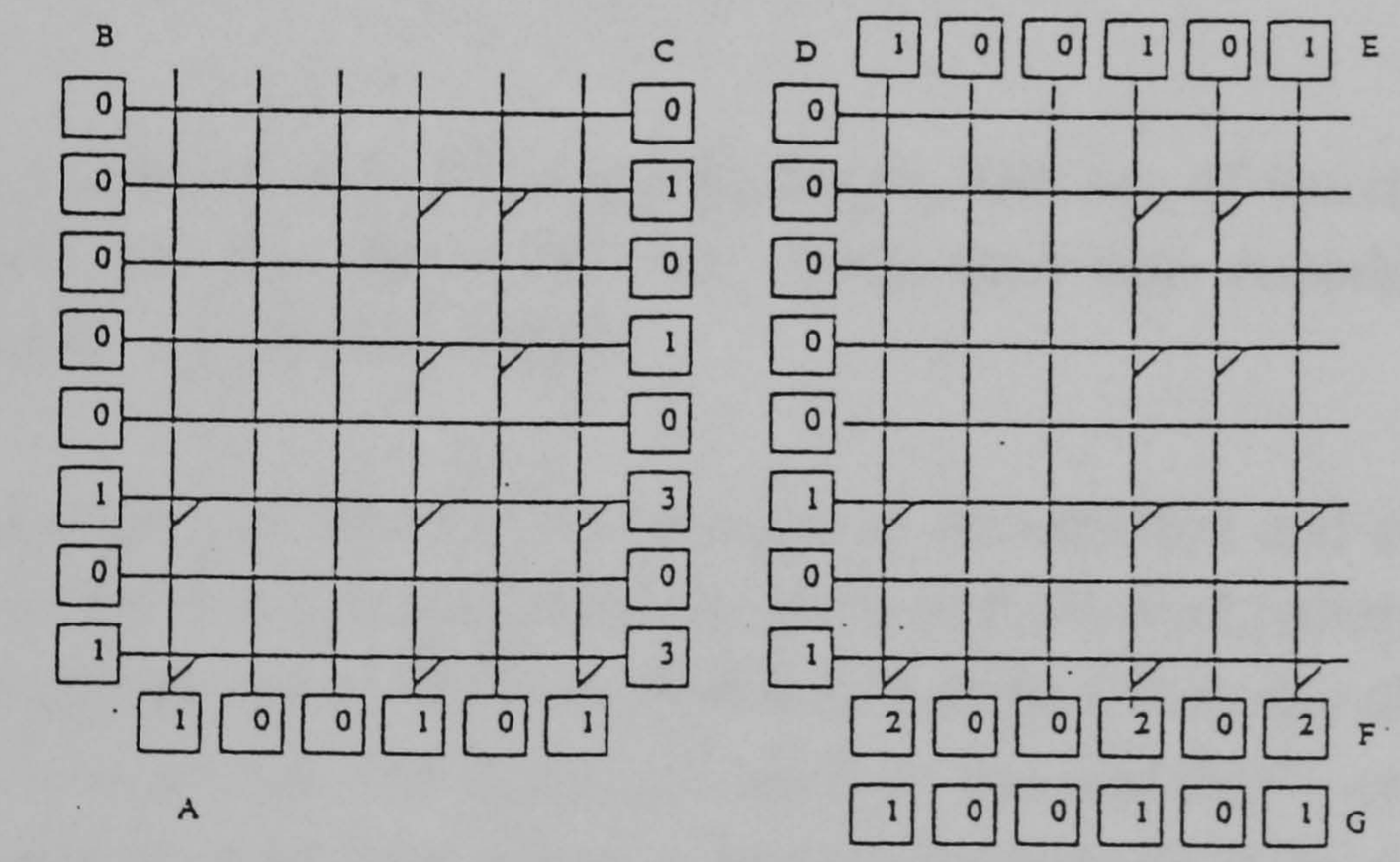


Figure 4.35: The basic approach in Austin's invariant pattern recognition system [Austin 1989].



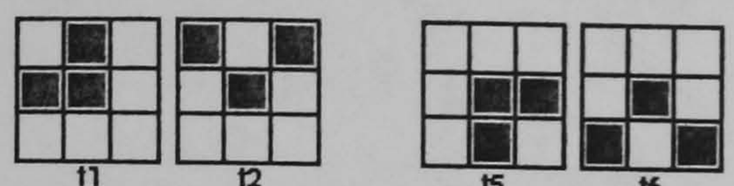
A = Key array
B = Teach array
C = Recalled pattern when testing with the pattern in A
D = Pattern in C after thresholding at 2
E = Recalled pattern when testing on the pattern in A including point 'i' at 1.
F = Pattern in E after thresholding at 3

Figure 4.34a [Austin et al. 1987].



B : Teach Class array
A : Key array
C : Class Response array
D : Thresholded Class (N = 2) array
E : Teach array
F : Response Class array
G : Threshold recall array

Figure 4.34b [Austin et al. 1987].



Pattern number	a	b	c	d	e	f
t1	1	0	0	0	0	1
t2	1	0	0	0	1	0
t3	0	1	0	0	1	0
t4	0	0	1	1	0	0
t5	1	0	0	1	0	0
t6	0	1	0	0	0	1
t7	1	0	1	0	0	0
t8	0	0	0	1	0	1

Transformation codes

000000 100001 000000
100001 100001 000000
000000 000000 000000

Pattern T1 when coded

Figure 4.36: A set of eight rotated patterns, the set of transformation codes assigned for the patterns (r=6, N=2) and one encoded pattern corresponding to T1 [Austin 1989].

Any input pattern needed to be recognised was rotated and scaled so as to produce a set of transformed patterns. Each transformed pattern was then checked by a matcher, to verify how closely it matched with any of the stored patterns. The best match had been indicated at the matcher's output. Each transformed pattern was first given a transformation class in order to be distinguished. This label indicated the scaling and rotation applied to the input pattern in order to achieve the particular transformed pattern. The transformation class is an N point pattern, made up of a set number of bits set to 1 within a field of r bits. Each transformation class was different, with N bits randomly distributed over its r elements. Austin [106] commented that, 'once the assignment of transformation classes is computed each of the transformed patterns are selected in turn and the elements at one in each

transformed pattern are replaced by the transformed class pattern, the elements at zero in each transformed pattern are replaced by a field of r zeros'. Once all the patterns are coded, they are all combined into one matrix by logically ORing them together. Figure 4.36 shows the entire process while additional performance related data are presented in Figure 4.37.

Figure 13. From [Austin 1989]

Simple example shape to be recognised after rotation

Rotation Degrees	Average Class Confidence	Recovered Rotation	
		θ	ρ
0	98.0	0	0
22.5	59.0	-14	0
45.0	65.0	12	0
67.5	59.0	-10	0
90.0	46.0	-8	0

Figure 4.37: Results of recognising an object at various rotations [Austin 1989].

4.9.3 Neural System Evaluation.

The two stage association in ADAM has a number of advantages. The class pattern acts as an intermediate stage with a known number of set bits, allowing the N-point thresholding technique to be used on noisy, incomplete or otherwise corrupted input. This would be impossible to accomplish if the input were associated directly with the output since there would not then be a known number of bits set to one, and so the N-point technique is in-operable. The class pattern entering the second memory is a hopeful noise-free vector that allows accurate recall in the second matrix of the final output pattern. The use of the class pattern is also storage efficient, saving on the size of memory required. If an a by b pixel image is to be associated with an x by y output image, then (a b x y) bits of storage are required to make the matrix.

If an intermediate class pattern of k bits is used, then the storage requirements become ($k\ a\ b + x\ y$).

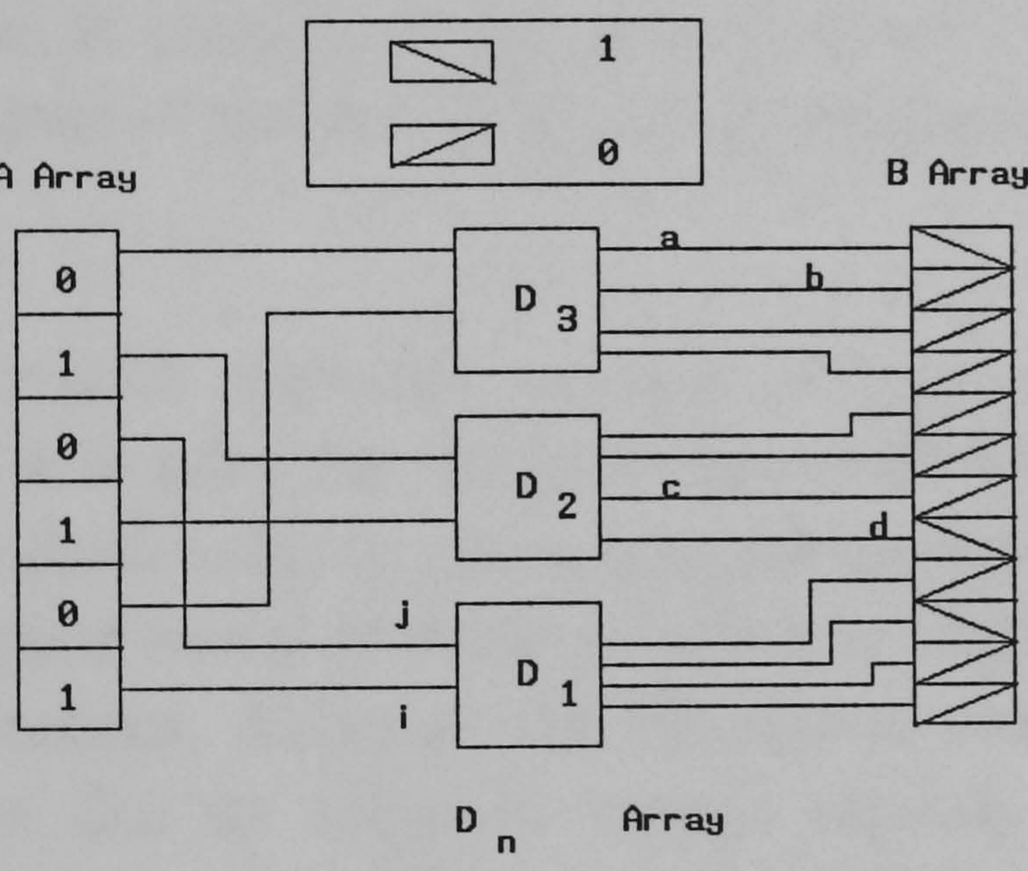
The use of the N-tuple pre-processing [Figure 4.38] has two major advantages; it copes with non-linearly separable patterns, and so allows ADAM neural system to resolve such problems as the XOR one. The tupling function provides a mapping that transforms any input into one that is linearly separable given the right sampling by the tuples. It also ensures that the inputs to the memory matrix are sparsely coded, so that there are not many active lines in any input, and this consequently helps prevent the memory matrix from becoming saturated.

ADAM learns new examples with one pass through the matrix and so does not require the back-propagation of errors, or repeated iterations. However ADAM has no adaptive internal representation and so can not code higher order features of the input, unlike the multilayer perceptron. This certainly limits ADAM's generalisation abilities, that mainly come from its tupling function. Sampling the input, the features that allow classification are not discovered explicitly but left to probability that some of them fall in regions sampled by different tuples. Since the tuple sampling is random, it is likely that important features are detected by at least some of the tuples, and generalisation occurs from these.

4.10 General Discussion.

It is generally accepted that authors should define a set of specific criteria, on which the comparison of a number of different techniques that all applying to the same field of research can be based. In the present application, four different criteria were decided to be used. Firstly, the criterion of good generalisation, i.e., how well a neural network responds to patterns with which it has or has not been trained on. Secondly, the criterion of efficient storage capacity, that is the ability to store a large number of

patterns and consequently be able to recognise a large range of different objects. Thirdly, the criterion of a small in size training pattern set, which reflects the ability of recognising an object without having to be trained on it for a long time, and finally the criterion of invariance to translation, rotation and scaling during pattern recognition. The last simply reflects the neural network system's ability to recognise an object with which it was trained, no matter if the testing picture does not reflect the same viewing object angle but an alternative view from a different position to the one originally used for teaching the neural system. A comprehensive classification is displayed in Figure 4.39.



State Table						
Sts	i	j	a	b	c	d
S1	0	0	1	0	0	0
S2	0	1	0	1	0	0
S3	1	0	0	0	1	0
S4	1	1	0	0	0	1

Figure 4.38: The N-tuple method.

Parallel to the artificial neural architectures previously presented, a number of additional artificial neural systems were investigated, always within the frame of position and scale invariant pattern identification. Among them, the "Brain State in a Box" neural network [79][80], the Cognitron and Neocognitron [85-90], the "Self-Organising Feature Maps" [91][92][93], the Bidirectional Associative Memories (BAM's) [97][98], and finally the Boltzmann machine artificial neural network [99-102]. Although all reviewed examples dealt with aspects of the general pattern recognition problem, few cases were actually found where the wider three dimensional position and scale invariant object recognition problem was investigated. Consequently, a detailed presentation of all the above neural systems was avoided. Nevertheless, the results regarding these neural systems as far as their pattern recognition potential is concerned, are to be presented below together with the general conclusions of the use of the previously presented artificial neural networks.

The first neural network reviewed was the Perceptron. The idea behind Perceptron is that it is pilot for the latter presented neural networks. The Perceptron model allows users to understand the theory and potential of the early 1960's developed neural network architectures that led to current more efficient implementations. Although the Perceptron possesses a number of desirable properties like its adequate storage capacity and its immediate recall, it is still considered to be a prototype that currently helps as a reference but is also widely substituted in all application by the more recent neural network models. Its improved versions like the cross coupled Perceptron allow an invariant to rotation, scaling and translation pattern recognition but are severely limited, as all Perceptron models, by its linear separability condition. For its latest property as well as its mainly use as a reference, Perceptron has been excluded from the set of possible neural network implementations for the proposed application.

ADALINEs and MADALINEs are generally considered to be an

improvement to the Perceptron. They have been developed in early 1960's, but although their capacity is capable of holding twice as many patterns as the number of units in their input layers, and their mathematics are well understood , they were all unable to support a position invariant pattern recognition process until the mid 1980's. It was in 1987 when the MADALINE Rule II was introduced when this problem was partially solved with the use of weight matrices that could vary according to the rotation, translation and scaling factor that was required. That simply means that in order to specify an overall independent system one should use a large number of complex weight matrices, that will definitely result in a 'poor to handle' system still partially capable though of supporting an invariant to translation, rotation and size pattern recognition operation, being limited mainly by the plethora of different variations of small rotational angles. In addition, the MADALINE's lengthly encoding time and its restriction to linear input output mappings, resulted in excluding MADALINEs and ADALINEs from the final set of possible neural network architectures for the intented pattern recognition application.

A neural network model like the Boltzmann machine, although it has been extensively used in pattern recognition and image processing applications and is able to perform nonlinear mappings between arbitrary sets of patterns, it is not suitable for the intented application, in the same way that the "Brain State in a Box" model, and the Back-propagation artificial neural network fail to be accepted. The reason is simply the fact that they all require a lengthly training time, which is the case of Boltzmann machine is prohibitively long. In addition, the Boltzmann machine has the ability to stuck in local minima and requires an extensive encoding time, while the Back-propagation is limited by its inability to know how to precisely generate any arbitrary mapping procedure, and the Brain State in a Box neural network model requires an encoding supervision. For all the previously stated limitations but primarily because of the long training time, all the previously four neural network models were excluded from the set of

possible neural network architectures for the project.

The problem of sufficient storage capacity is the main limitation for rejecting neural networks like the Hopfield, and the Bidirectional Associative Memories. Both neural network architectures have a low storage capacity which although it might coincide with an extreme memory plasticity in the case of the Bidirectional Associative Memories (which allows emergent statistical similarities to appear from excessively noisy data sets) and in the case of Hopfield neural networks with the ability to handle analogue data in continuous time, it still represents a major limitation for the proposed approach. In addition Hopfield's network limitation with respect to its restricted learning and its lack of a straight-forward encoding scheme, as well as the Bidirectional Associative Memory's restriction to binary bipolar valued pattern pairs, justify the final decision to exclude these particular neural network paradigms from the remaining applicable neural network architectures.

Kohonen's Self Organising Feature maps artificial neural network was also excluded mainly because it is shown to require an extensive encoding time and it is unable to add new classes of patterns without complete retraining. It must although be noticed, that Kohonen's neural network has a number of desirable abilities, such as the performance of non-parametric pattern classification and the provision of real time nearest neighbour response.

From the previous discussion it is evident that the Adaptive Resonance Theory 2 neural network, Hecht-Nielsen's Counterpropagation neural network, and Austin's Advanced Distributed Associative Memory, are the remaining possible neural network architectures for the needs of the intented application. Understandably all three can perform efficiently on the above problem, but ADAM's previously presented abilities and the fact that is a relatively 'unexplored' neural paradigm made it the most likely neural

network architecture to be used. Furthermore, the Advanced Distributed Associative Memory is a rather recent neural network paradigm, with capabilities for pattern recognition of three dimensional objects that have yet to be thoroughly investigated.

Not extensive training [3]

Position Independent [4]

	1	2	3	4
Adaline & Madaline		✔		✔
Back-propagation	✔	✔		✔
Brain State in a Box	✔	✔		
Cognitron & Neocognitron	✔	✔		✔
ART 1 & ART 2 ☞	✔	✔	✔	✔
Counterpropagation ☞	✔	✔	✔	✔
Hopfield	✔		✔	✔
Kohonen	✔			✔
BAM's	✔	✔	✔	✔
Boltzmann Machine	✔	✔		✔
ADAM ☞	✔	✔	✔	✔

Good Generalisation [1]

Large Pattern Capacity [2]

Figure 4.39: The Performance Evaluation table.

CHAPTER 5

Invariant Pattern Recognition and Artificial Neural Systems: Categories and Criteria.

5.1 Introduction.

The goal of pattern recognition is to effectively classify objects into one of a number of possible categories (classes). The objects of interest are called patterns and can loosely define a natural scene. By carefully noticing patterns, proposing and formulating rules, an understanding of the underlying classification processes can be achieved. A learning system [Figure 5.1], a general form of a neural network, can be viewed as a higher level environment that itself can assist in the design of a decision making system (often called a classifier). Learning in such an environment consists of choosing or adapting parameters within the model structure [Figure 5.2]. The learning system has available a finite set of samples of solved classes. The data for each case consist of a pattern of observations and the corresponding successful classifications. The objective of the learning system is to customise the classifier's structure to the specific problem by finding a general way in which to relate any particular pattern of observations to one of the specified classes. In a basic description of a classification problem [Figure 5.3], each sample of a solved problem consists of observations and the corresponding correct class memberships. Usually, a single conclusion or class will result for a given pattern of observations. While there may be problems in cases where multiple class membership is important, these can often be decomposed into several subproblems involving individual class membership.

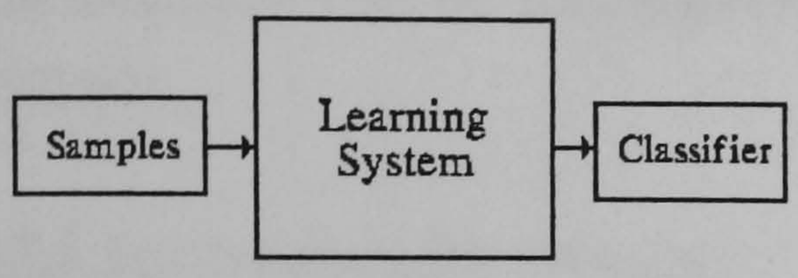


Figure 5.1. A Learning System.

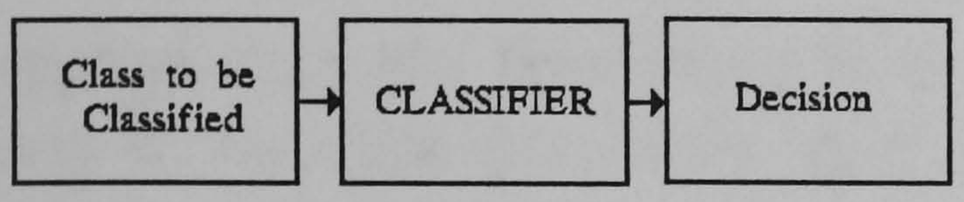


Figure 5.2. A Classification System.

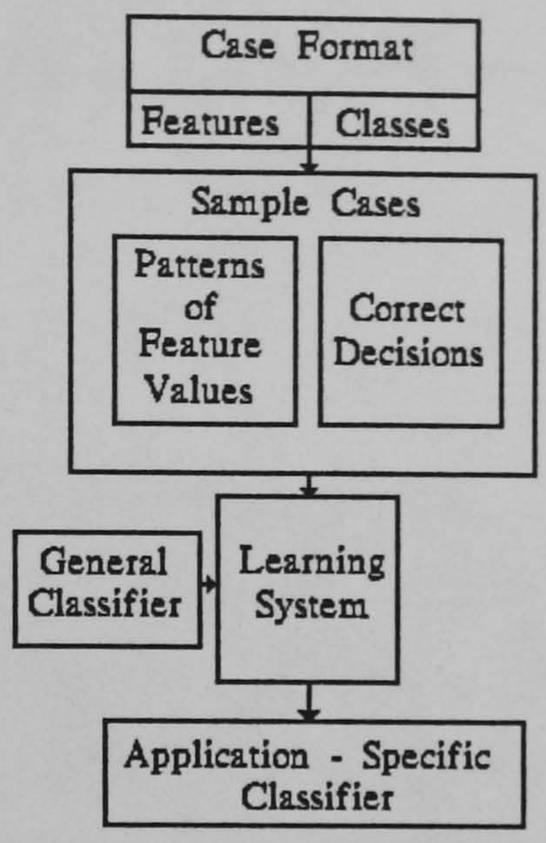


Figure 5.3. A basic Classification Representation.

5.2 Invariance Techniques.

The argument in favour of learning systems is that they have the potential to exceed the performance of experts and the potential to discover new relationships among concepts and hypotheses by examining the record of successfully solved cases. Thus, the process of learning, automatically holds out the promise of incorporating knowledge into the system without the need of a knowledge engineer. Yet, there are strong practical reasons to expect that what can be learned directly from sample experience alone is limited, if it ignores the context within which problem solving is carried out. Thus, there is a need to combine domain-specific expert knowledge with learning approaches. One of the primary tasks of pattern recognition is to design classifiers that are invariant to position transformations. Neural Networks have proven to be extremely powerful classifiers [107] [108]. Whereas the boundaries of decision regions created by conventional classifiers [27] usually belong to relatively restricted sets of hyperplanes, neural networks are able to employ much more general decision boundaries. They are therefore often able to achieve levels of performance beyond the reach of conventional classifiers [109]-[111]. Having in mind the effectiveness of biological

systems, the hope is that the characteristics essential to invariant classification can be formulated and utilised in the simplified artificial neural systems.

5.2.1 Invariance by structure.

The idea of imposing invariance on a neural network by structuring it appropriately has been proposed by various authors [112]-[115]. A simple way to visualise this, is by creating connections between the neurons that force transformed versions of the same input to have the same output. Consider, for example, an input image that is to be classified independently of positional changes and a neural network that takes this image as input. Rotational invariance can be enforced structurally as follows: let a_j be any neuron that receives inputs from the image, and let w_{ji} be the connection weight leading to that neuron from pixel i in the image. If $w_{ji} = w_{jk}$ for all i and k that lie at equal distances from the centre of the image, the neural network is invariant to positional rotations since such a rotation will not change the total input to any neuron. The same weight has to be duplicated for every same radial distance pixel in order to maintain rotational invariance, and avoid the loss of angular information. The number of connections required is therefore growing prohibitively fast for images of realistic size. For a more realistic classifier [114] the number of connections necessary for translational invariance alone can be shown to be N^4 for an input image consisting of N pixels with only one object present. Clearly, this is not a realistic approach when images usually contain at least 10^4 pixels for satisfactory resolution.

5.2.2. Invariance by training.

Since neural networks are such powerful classifiers, it is natural to ask if their classification ability can be used to obtain directly transformation

invariance. To achieve this goal, a number of different aspect object views should be used as input to the neural system. If the neural architecture is able to learn to discriminate these objects properly, and if the number of examples shown is large enough, the net will hopefully also generalise correctly to unknown transformations. This approach was used by Rumelhart et all. [116], to obtain rotational invariance for the 3 X 3 objects that they investigated, and has also been suggested by Lang and Hinton [117] for speech recognition.

Invariance by training has two disadvantages. It is not clear how a neural system, trained to recognise new objects invariantly, can use this training to be able to additionally recognise new objects invariantly. Thus such an architecture must be trained on many aspects of each object. 'Although procedures based on symbolic decomposition of the objects might eliminate this problem, such methods are currently poorly understood' [117]. With current neural network techniques it would be necessary to retain within the neural system all transformations for every new object to be identified. The other problem with the approach is that the demands placed on the classification system might be very severe - more so than even a neural network classifier can cope with. This is especially true if the dimensionality of the feature space is high. 'In a high dimensional feature space, it is generally significantly difficult to recognise transformed versions of an object if the feature space is not carefully chosen to be suitable for such transforms' [116].

5.2.3 Invariant Feature Spaces.

It has long been recognised that it is possible to extract features that are invariant to transformations of an input [118]-[120]. If such features are used, the classifier does not need to delineate the range of transformations of an object with complicated decision boundaries. The only differences between different instances of the same object are then due to such factors as noise

and occlusion. A related technique, that uses invariant feature spaces has been applied in optical classifiers [121]. The advantages of using an invariant feature space are the fact that the number of features can be reduced to realistic levels, the requirements on the classifier are relaxed and the invariance for all input objects is ensured. The main disadvantage of using invariant feature spaces is the need to first calculate the features before the classifier can be employed. Clearly then for an image recognition application the obtained image can not be as input to the neural network without some pre-processing. To minimise the effect of this problem, invariant feature spaces are required that are computationally efficient. The fact that not all feature spaces are equally suitable for a given problem, dictates that the selected method for an invariant feature space should be flexible enough to allow the choice of a space suitable for whatever problem is under investigation. Invariant feature spaces that have been used in conjunction with neural networks include wedge-ring samples of the magnitude of the Fourier transform [122], the magnitude of the Fourier transform in log-polar coordinates [123] and moments [76]. These feature spaces are well known to have difficulties when noise is present, and the remaining two feature spaces are not invariant to all possible transformations.

5.3 Criteria of a position invariant pattern recognition.

Having in mind the disadvantages of the previously described categories of invariance in neural systems and the fact that neural architectures are excellent classifiers but themselves alone can not possibly resolve the position invariance constraints, it has become apparent that special attention should be paid in the preprocessing step of the proposed position invariant pattern recognition process so as to be effective and general. Emphasis was placed in a new form of invariant feature space represented by the normalised quadtree/octree data structure. It was extensively investigated whether such structures can be effectively used for a position invariant, noise independent pattern recognition. Particularly emphasis was given in devising a technique

that could effectively minimise the large training pattern set as well as preserving its general position invariant profile. The algorithm should also be easy to implement and have a clear mathematical and theoretical structure. Finally, the computational requirements particularly important in neural network applications were considered, pointing to an algorithm that will require a few teaching patterns as input and a new neural network architecture that could be trained in one pass thus significantly reducing a generally computationally expensive teaching procedure.

CHAPTER 6

Normalised Hierarchical Volume/Surface Data Structures for Position Invariant Pattern Recognition.

6.1 Introduction.

In today's world of expanding automation, autonomous analysis of complex image data is a critical technology. Problems slowing the growth of this field are traditional image analysis methods that lack the speed, generality, and robustness that many modern image analysis problems require. While artificial neural networks promise to improve on traditional techniques of image analysis, systems composed purely of neural networks have difficulty performing all the diverse analysis required of an autonomous system. Artificial neural networks are being considered to augment and if necessary, replace digital computers in time critical applications. Such systems offer a computational platform that is highly parallel, adaptive, and potentially fault tolerant.

In previous chapters, a number of well known position and scale invariant techniques (moments invariants [44], normalised Fourier

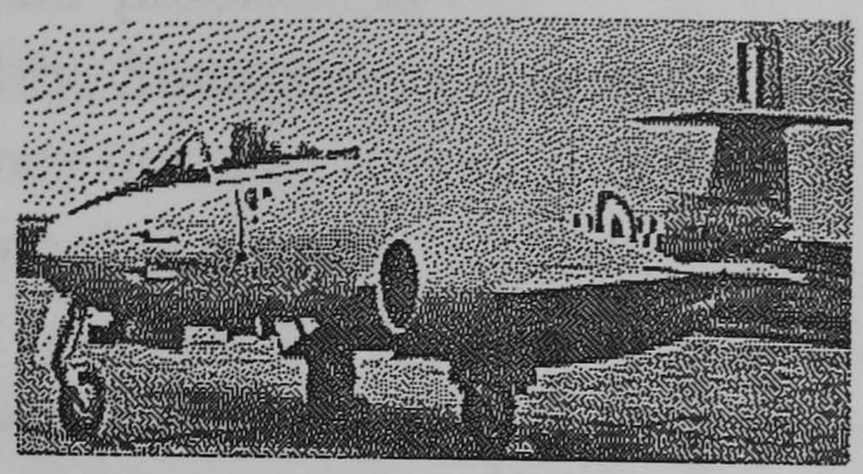
descriptors [42]) have been described, in the framework of an invariant three dimensional object identification process. It was concluded that, although these techniques often show a highly efficient classification performance, they generally fail to produce a consistent, noise resilient and successful position and scale invariant pattern recognition behaviour. Therefore, emphasis is placed in this chapter in the presentation of an invariant three dimensional classification system that is based on the one hand in a volumetric representation of an object and on the other hand, in a powerful neural network classifier. This combination is expected to produce an enhanced invariant three dimensional object classification behaviour, which will demonstrate the highly desirable classification successes of moments invariants and normalised Fourier descriptors, but will additionally be consistent, noise resilient and computationally efficient.

The present chapter aims to introduce a method capable of recognising an aircraft from any viewpoint in uncluttered noisy scenes. Current neural network based methods are limited by the time required to learn to recognise an object, as well as by the number of pictures required to be processed in order to achieve successful classification results. Typically, a number in the order of thousands of pictures per object can often be used in training multilayer neural networks. In the present chapter a new method that uses a single presentation of an image during training will be presented. The aim is to achieve the same successful position invariant classification results for three dimensional objects. The proposed method uses for this purpose a novel pre-processor, that is based on the properties of the quadtree and octree data structures combined with a neural network classifier. Both data structures have been extensively studied during the last decade [124-128], and have been shown to posses two significant properties. First, they can be made independent of the orientation, translation and scaling of the actual object they represent. Second, they offer high compression ratios, normally in the range of 5:1 or 6:1; a considerable advantage when large numbers of digital pictures are to be stored within a storage medium.

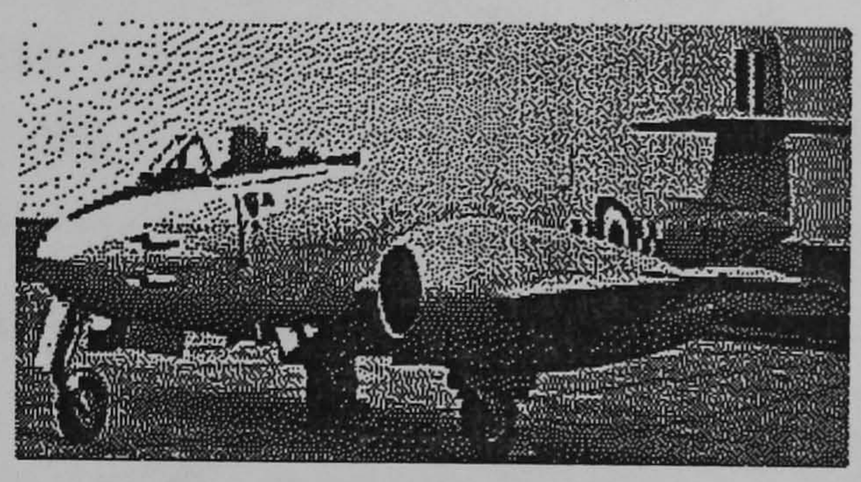
This chapter will be structured into four parts. A general description of the image processing procedures that were used during the input pre-processing steps, a detailed description of the quadtree and normalised quadtree data structures, the theory behind the octree generation and the properties of that data structure, and finally the actual method used in order to train and test the ADAM artificial neural network [106].

6.2 The Image Processing Procedures.

In the current work the various image processing algorithms used, can be classified into two main categories. First, image enhancement algorithms that aimed to improve the visual presentation of the original raster picture and second, image segmentation procedures that successfully clustered the enhanced image into meaningful regions for the intended pattern classification stage. In image segmentation emphasis was placed primarily in edge detection operators and secondly in histogram thresholding techniques. A synoptic view of the overall image pre-processing procedures used can be viewed as a sequence of the following four steps. First, a 3 by 3 pixel Gaussian kernel was used to smooth the original raster image, and reduce any present noise. Second, an edge detection segmentation procedure, based on the Maxgrad edge operator [129], was applied on the smoothed image. Third, the resulting raster data were histogram normalised over the full 512 gray scale intensities range. Finally, the enhanced image from the previous step was thresholded, thus eliminating the lowest 5% of edges (that normally correspond to noise). All remaining high grey scale intensities greater than zero were reduced to 1, resulting in the final binary edge map picture. It is clear from the above algorithmic presentation, that Gaussian smoothing masks and image segmentation using edge detection operators are the most interesting areas of image enhancement in the present application and therefore they would be thoroughly reviewed in the following sections of this chapter.



The original raster image.



The enhanced picture.

Figure 6.1. A typical image enhancement example.

6.2.1 The Image Enhancement Process.

Image enhancement processes consist of a collection of techniques that aim to convert a picture or improve the visual appearance of an image, to a form better suited to human or machine analysis [Figure 6.1]. Image quality is generally studied in a variety of methods, all depending on the current image type. The various methods used in image enhancement are rather loosely structured, because in many cases a universally acceptable optimisation criterion is difficult to be defined. Enhancement usually concerns contrast stretching, smoothing, and pseudo-colouring. The algorithms are usually based either on frequency or spatial domain techniques. Processing techniques in the first category are based on modifying the Fourier Transform of an image. The spatial domain methods, refer to the image plane itself and approaches in this category are based on the direct manipulation of pixels in an image. The choice of techniques

depends mainly on the problem and its overall processing requirements, convenience and fidelity. Smoothing is performed by low pass and high pass filtering in the frequency domain, while integration and differentiation are the respective standard techniques in the spatial domain.

One of the difficulties in image enhancement is that different algorithms or methods are needed for different types of noise. A box type smoothing or nearest neighbourhood type smoothing, removes Gaussian noise [130], however it sometimes diffuses noise around the neighbourhood if it is applied to peak noise or scan line noise without care. Median filtering [26], can remove this kind of noise considerably better but it also tends to smooth out or blur fine structures in an image and may cause image degradation [Figure 6.2]. Sometimes image enhancement techniques known as spatial filters, have different variations. Among them the numerous mean filters [13], the Nagao and Matsuyama technique [131] that uses convolving operations with pentagonal and hexagonal masks for random noise removal without blurring sharp edges, and the recently presented Kundu method [132] that applies a two dimensional generalised mean filter for impulse noise removal. It has been recently shown that if simple mean and variance of gray tones are used as local grey tone statistics information of the higher order spatial characteristics of the gray tone distribution are lost. In order to obtain the spatially dependent grey tone statistics a facet model [133] has be used that fitted a simple polynomial surface to the grey tone intensities of a region.

The area of image enhancement, a long time favourite area of research in digital image processing, contains a large number of techniques. In the current chapter an additional number of enhancement methods to the ones already mentioned were examined. Among them, the histogram equalisation technique [134], the direct histogram specification [135], the histogram hyperbolisation [136], the nearest neighbour smoothing [137], the selective averaging smoothing [138], the maximum likehood smoothing [139], unsharp masking [140] and the cubic convolution [141]. Although all the above

methods have desirable properties and can perform effectively on raster images, it was concluded that histogram equalisation is the most suitable technique in the current implementation as it is both effective, computationally efficient and has a well understood mathematical background [Appendix B] [Figure 6.3]. In the current application the raster images used, were images of six different aircraft types, created from precise detailed models and their corresponding pictures when the models were presented in front of photographic camera with a 512 by 512 pixel resolution [Figure 6.3].

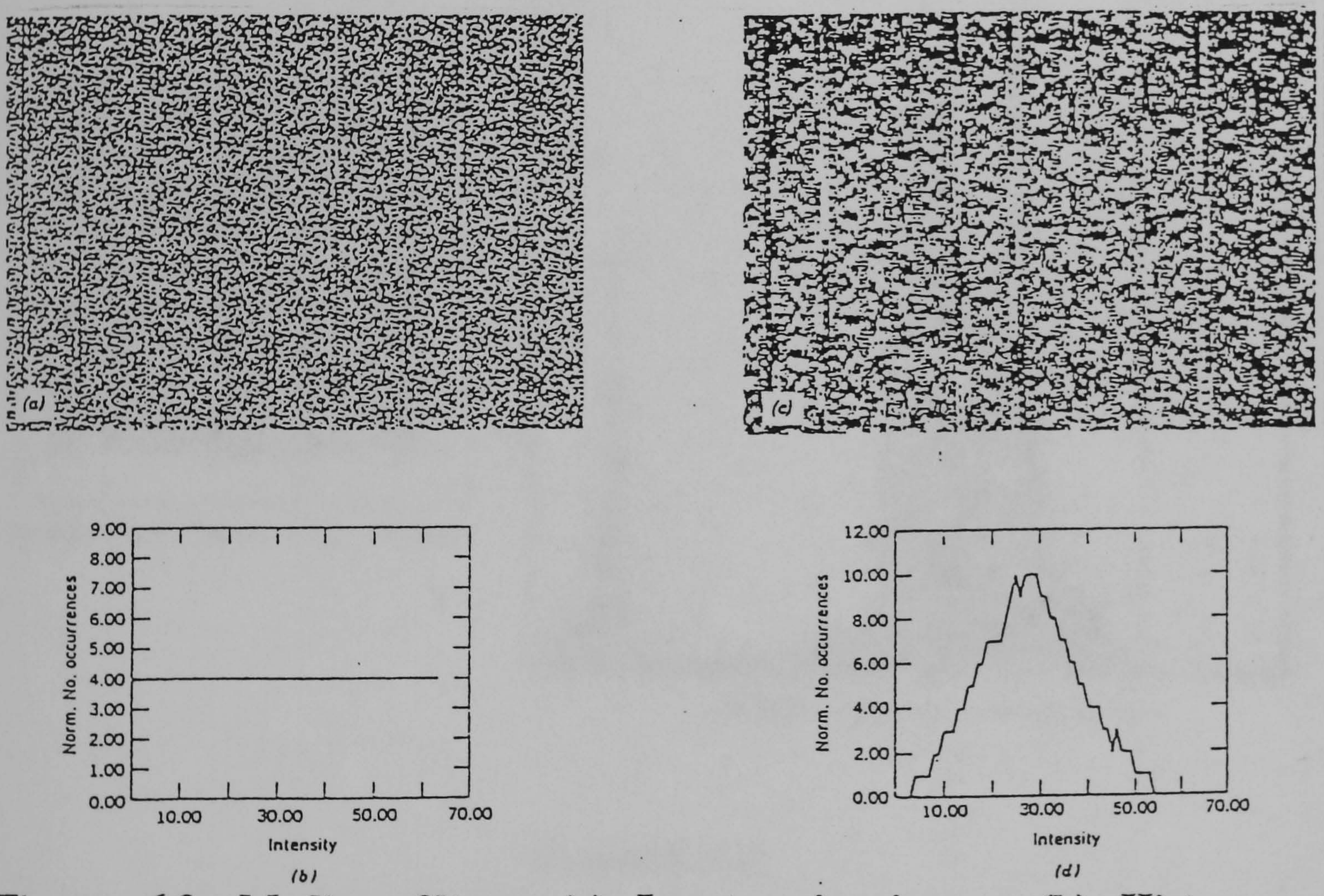
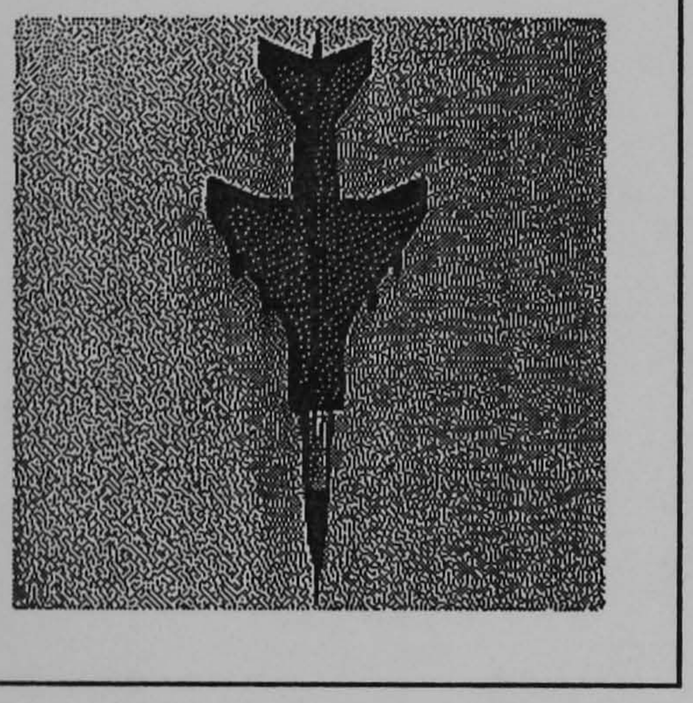
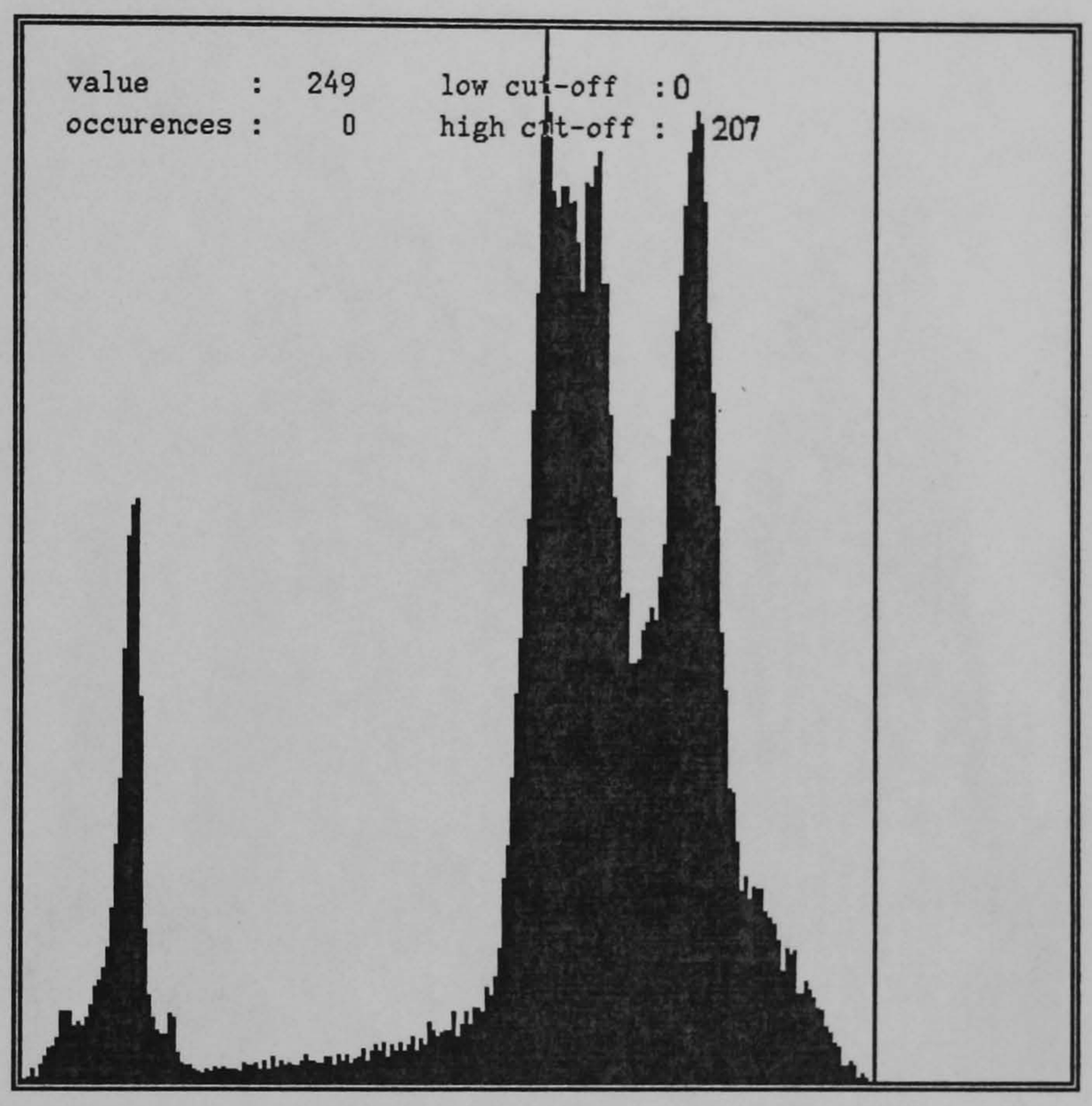


Figure 6.2. Median filters. (a) Input noise image. (b) Histogram (normalised). (c) Median filtered noise image. (d) Median filtered noise histogram

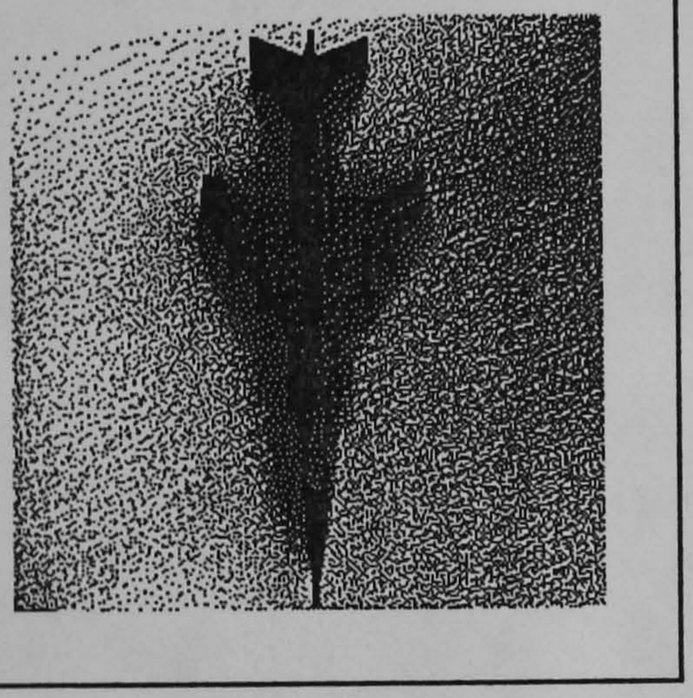


Original aircraft picture. [512 by 512 pixels]

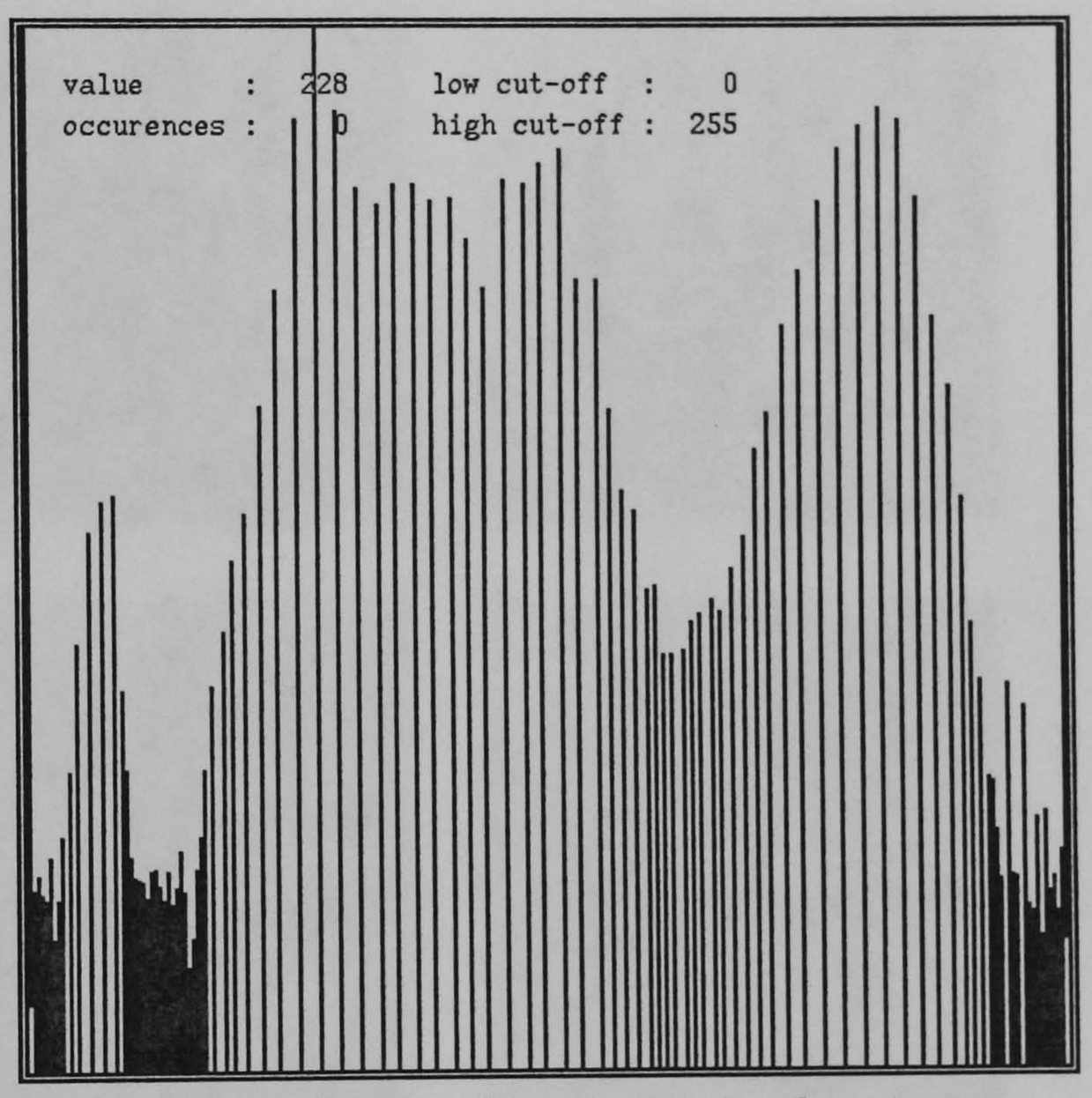


Histogram of the original aircraft picture.

Figure 6.3(a)



The enhanced picture.



Histogram of the enhanced aircraft image.

Figure 6.3(b)

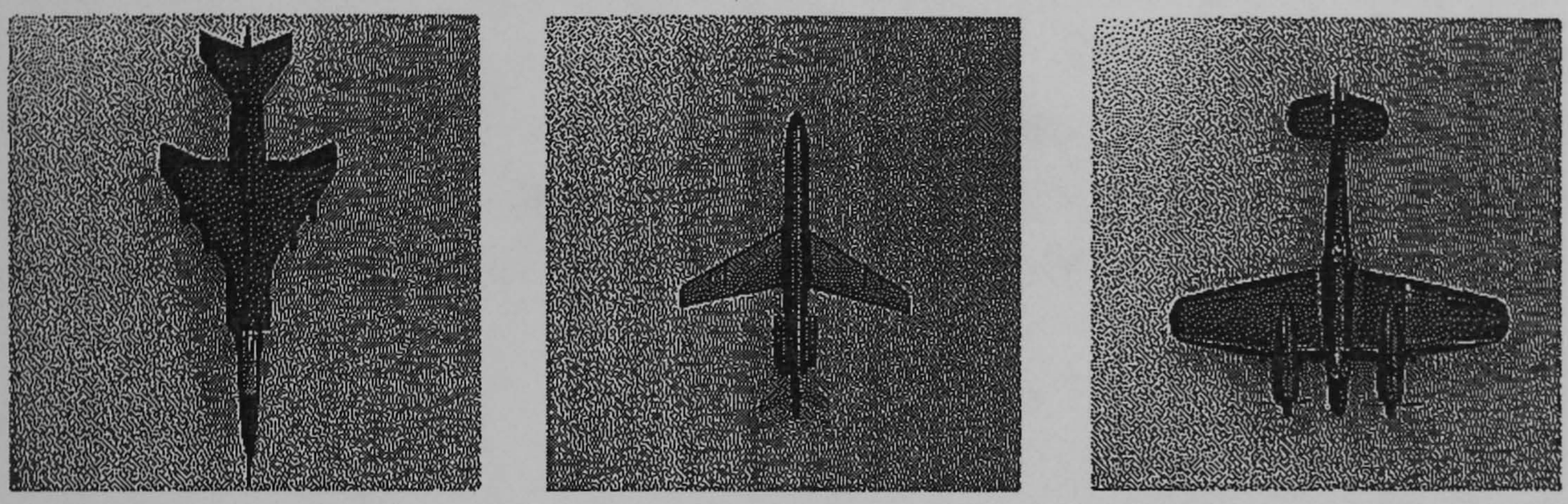


Figure 6.4. The six aircraft models used in the experiments.

6.2.1.1 Gaussian smoothing.

When applying a smoothing operator, there are two main facts to be born in mind. The first is the suppression of noise, and the second is the removal of signals that give much detail for the specific resolution chosen. For the first purpose, median filtering and related procedures generally present a significantly better performance when applied to images corrupted with impulse noise but do not correspond favourably to a linear model; neither do they correspond to low pass filtering, this being perhaps the most obvious technique for noise suppression. For the second purpose, a soundly based means of removing unnecessary detail is provided by the low pass filter.

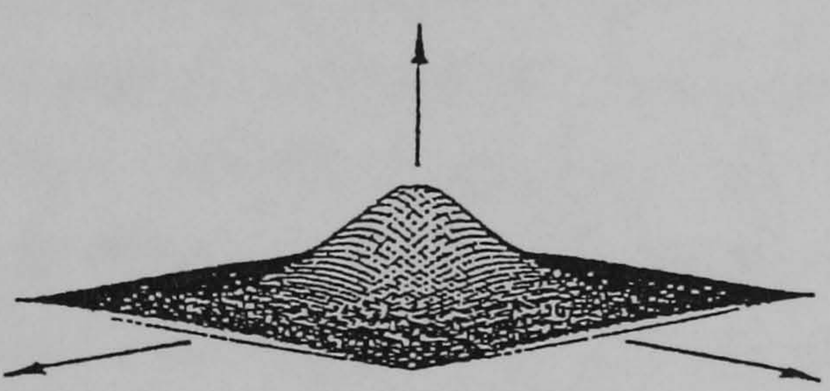


Figure 6.5(a)

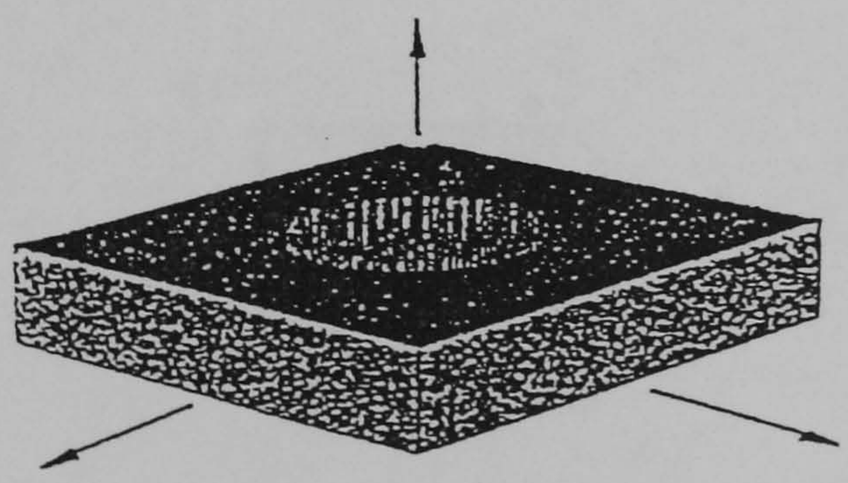


Figure 6.5(b)

Gaussian smoothing operators fall into the above category of crucial operators of particular importance in image processing. Gaussian smoothing

is used both to suppress image noise and to approximate the action of low pass filters by producing images of different resolutions for further detailed analysis. Thus it is vital that these operations should be optimised; efforts to obtain even a small improvement would be justified since this type of operation is used so widely. A filter that has a Gaussian like impulse response function is essentially a low pass filter [Figure 6.5(a)]. Gaussian filters are also useful in the synthesis of high pass [Figure 6.5(b)]. The Gaussian filter function is interesting as it is the function with the minimum product of second moments in space (time) and frequency. The Gaussian is also separable and this separability is important in the realisation of multidimensional filters. The Gaussian function is also called the normal density function and is subject to the central limit theorem in probabilities [142]. The central limit theorem illustrates an interesting property of the repeated convolution of simple nonnegative functions - the results tend to a Gaussian function. This property provides an economical means of computing Gaussian convolutions. In digital image processing 3 by 3 and 5 by 5 pixel neighbourhoods are often used during convolutions of Gaussian masks with raster pictures. Figure 6.6(a) shows a simple 3 by 3 pixel neighbourhood with the specification numbers for each one of the pixels and in Figure 6.6(b) a view of a random two dimensional spatial image plane is presented with the simple 3 by 3 pixel mask in it.

A_0	A_1	A_2
A_5	A_4	A_3
A_6	A_7	A_8

Figure 6.6(a). A typical 3 by 3 pixel neighbourhood.

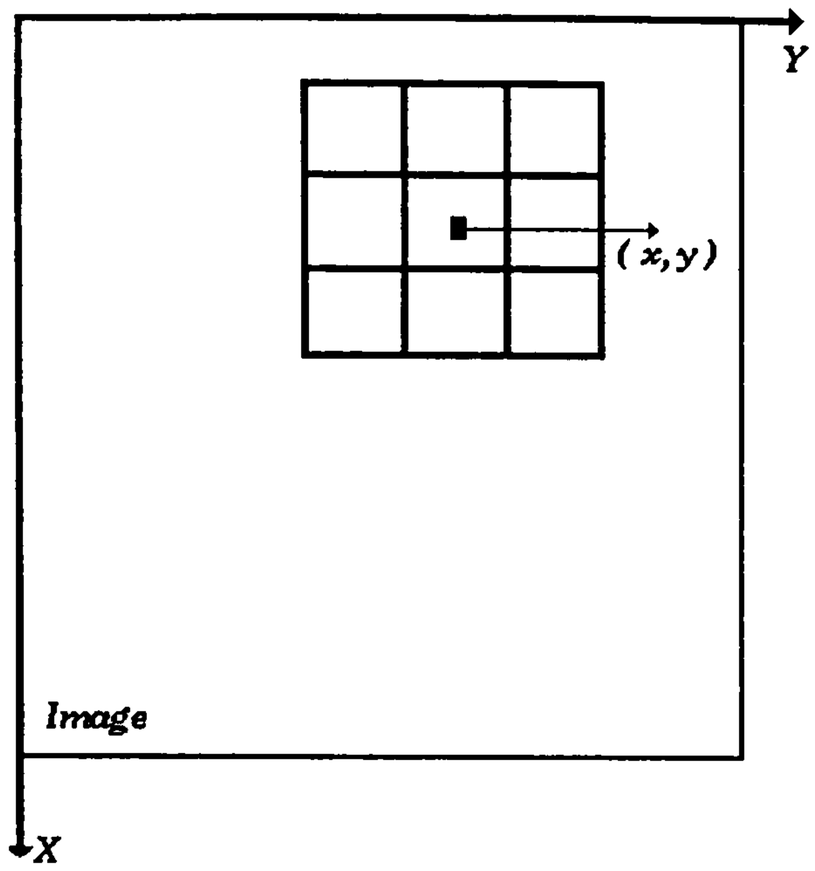


Figure 6.6(b). A random 2D spatial image plane.

In such a neighbourhood, the pixel in the centre (A_4) is considered to be the centre of a two dimensional axis system. Respectively, A_3 pixel is situated at (1,0) and A_2 is situated in position (1,-1). Studding the equation of a Gaussian function it can be simply derived that both (-1,1) and (1,-1) pixel pairs will produce the same result (as both x-component and y-component will be squared). Therefore, evaluating a Gaussian mask of a 3 by 3 pixel neighbourhood will only require evaluating the function for three pixel positions, A_4, A_3, and A_2. The general convolution process, regarding a Gaussian mask and a raster image can be described as follows:

$$G(f(x, y)) = f(x\text{-}1, y\text{-}1) * G(A_2) + f(x\text{-}1, y) * G(A_3) +$$

$$f(x\text{-}1, y\text{+}1) * G(A_3) + f(x, y) * G(A_4) +$$

$$f(x, y\text{+}1) * G(A_3) + f(x\text{+}1, y\text{-}1) * G(A_2) +$$

$$f(x\text{+}1, y) * G(A_3) + f(x\text{+}1, y\text{+}1) * G(A_2) =$$

$$(f(x-1, y-1) + f(x-1, y+1) + f(x+1, y-1) +$$

$$f(x+1, y+1)) * G(A_2) +$$

$$(f(x-1, y) + f(x, y-1) + f(x, y+1) +$$

$$f(x+1, y)) * G(A_3) + f(x, y) * G(A_4)$$

G(i) denotes the result of the Gaussian function when applied at the coordinates defined by the i pixel in the specified 3 by 3 neighbourhood. Many scientists have used Gaussian masks in their work. Marr and Hildreth [143] argued that meaningful intensity changes in a two dimensional image can occur at a variety of spatial scales. 'These changes', they said, 'are best located by maxima in the first spatial derivative, or by zeros in the second spatial derivative of the low pass filtered image' [143]. For these reasons a radially symmetric second derivative operator represented by a Laplacian operator applied to a two dimensional Gaussian function was used [Figure 6.7(a-b)]. They have shown that both the size and the standard deviation of the Gaussian can be adjusted according to the desired spatial scale. Gaussian filtering has also been applied by Canny [144] in his work on edge detection. Canny computed optimal edge operators that can be approximated by derivatives of Gaussians. A typical example of applying Gaussian smoothing to a raster image in the context always of the current application on aircraft pictures can be seen in Figure 6.8.

6.2.2 The Image segmentation process.

Image segmentation requires the processing of large quantities of visual information, with noise from the sensing mechanisms and additional data obscuring the semantically significant entities that are to be perceived. The complexities in the design and implementation of such image understanding systems typically have led to a decomposition of the problem into distinct

subsystems for segmentation and interpolation, often referred to as 'low-level' and 'high-level' processing, respectively.

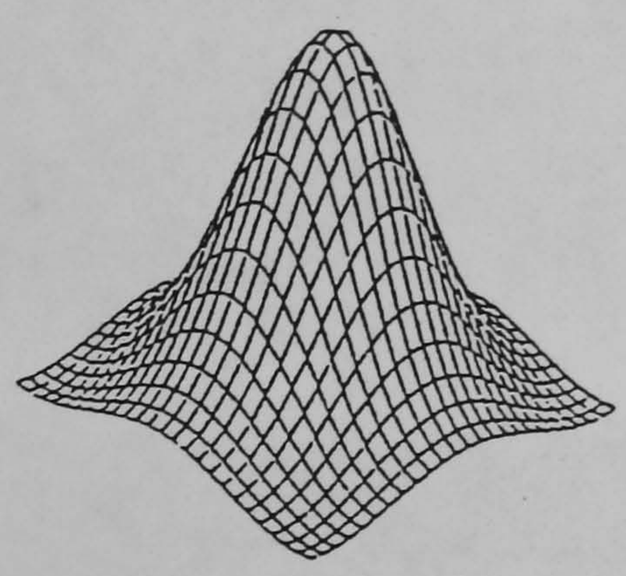


Figure 6.7(a). A typical 3D Gaussian surface

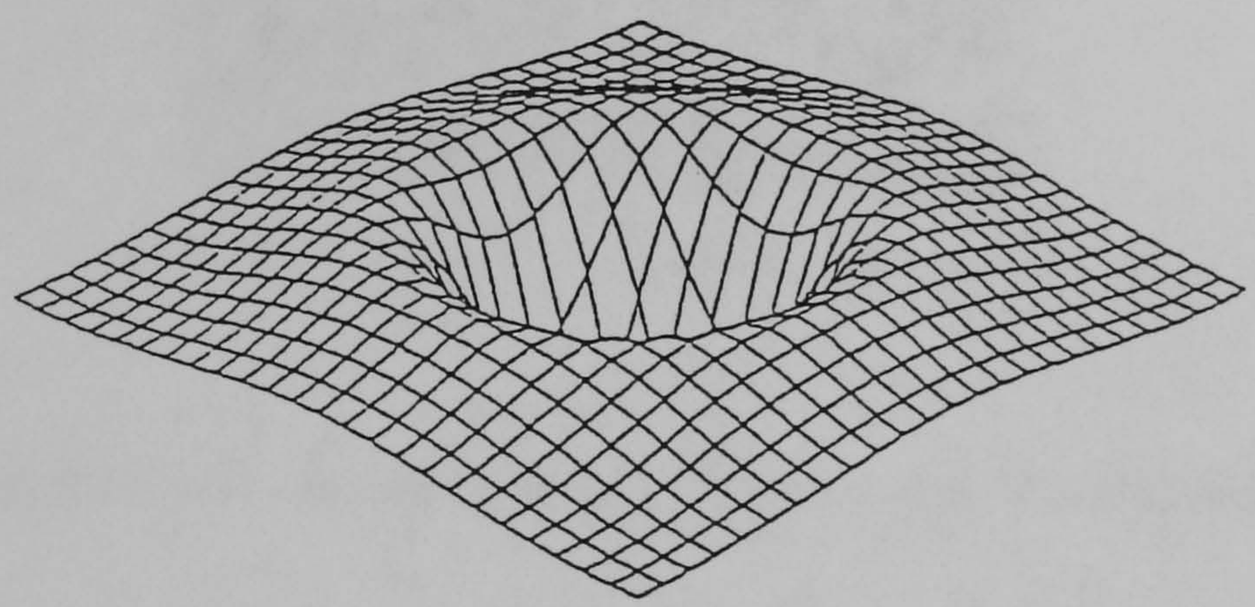
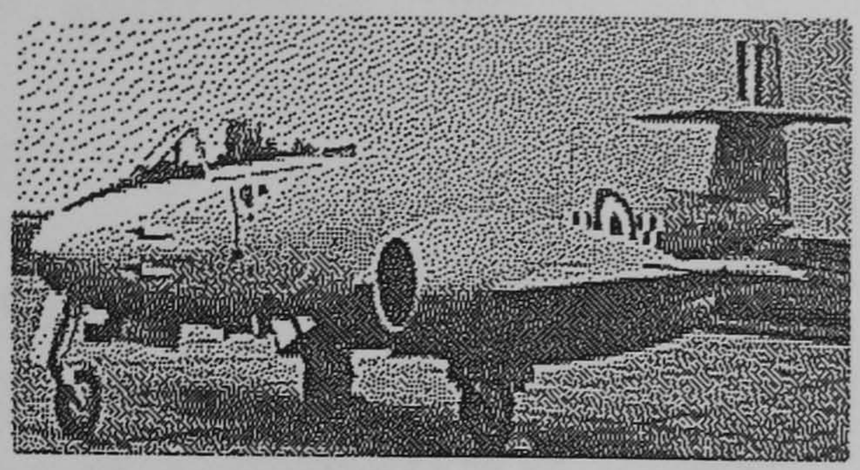
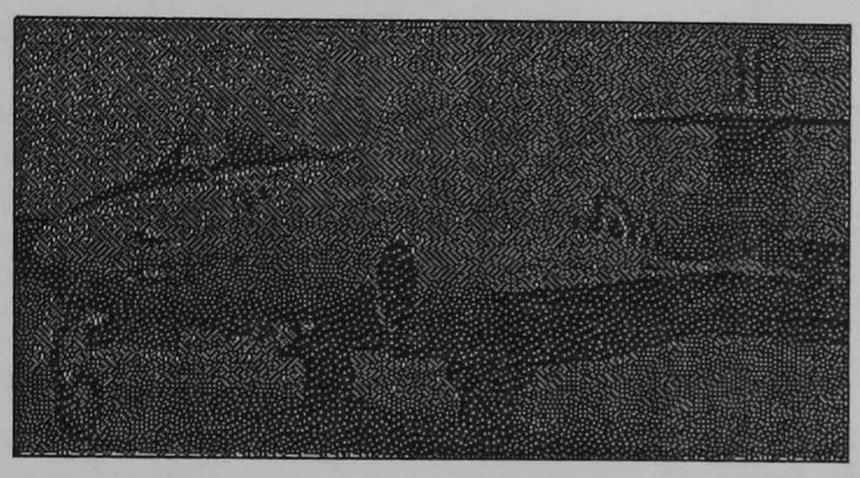


Figure 6.7(b). A Laplacian Gaussian function

The approaches to segmentation are divided into two broad categories, boundary formation and region formation. The tools for extraction of boundaries involve spatial differentiation, non-maxima suppression, relaxation processes, and grouping of local edges into segments. Approaches to region formation include region growing under local spatial guidance, histograms for analysis of global feature activity, and finally an integration of the strengths of each of the above by a spatial analysis of feature activity. Image segmentation is the division of an image into different regions, each having certain properties. It is the first step of image analysis that aims at either a description of an image or a classification of the image if a class label is meaningful. Image segmentation is a critical component of an image recognition system because errors in segmentation might propagate to feature extraction and classification.



The original raster image.



Smoothed image with a Gaussian Kernel of size 3 by 3 pixels.

Figure 6.8. An example of Gaussian smoothing.

The problem of image understanding can be viewed as one of performing initial segmentation using a number of general procedures, feeding this low-level output to a high level system, and then allowing feedback loops so that the interpretation processes can influence a refined segmentation. From this point of view, the segmentation processes provide a compact description of the location and characteristics of visually distinct areas of the image. However, the local analyses may generate a great deal of spurious activity because objects in images do not appear as uniformly coloured areas (like in graphics areas) but rather have natural textural variations and shadows. Thus, the integration of local processing into globally consistent boundaries and regions is not at all straightforward. The segmentation techniques can be categorised into three classes. First, characteristic feature thresholding or clustering. Second, edge detection, and third, region extraction. In the current stage of the present chapter emphasis

will be placed on edge detection methods.

6.2.2.1 Edge detection.

Boundaries of objects tend to show up as intensity discontinuities in an image. Boundaries in images are extremely important, and often an object can be recognised from only a crude outline. This fact provides the principal motivation for representing objects by their boundaries. One might expect that algorithms could be designed that find the boundaries of objects directly from the gray level values in the image. When these boundaries have complicated shapes, the task is difficult. Much greater success has been obtained by first transforming the image into an intermediate image of local gray level discontinuities, or edges and then composing these into a more elaborate boundary. In this case, boundaries that are highly model-dependent may be decomposed into a series of local edges that are highly model-independent.

A digital edge occurs on the boundary between two pixels when the respective brightness of the two pixels is significantly different. The difference may depend upon the distribution of intensity values around each of the pixels. There are two main types of edges, the step and the roof edges. The first occurs between each pair of neighbouring pixels where one pixel is inside the brighter region of an image and the other is outside the region. If an image region is scanned, observing the intensity values steadily increasing and then after a certain point observe that the intensity values are steadily decreasing, it is likely to be said that there is an edge at that point of change from increasing to decreasing brightness values. Such edges are called roof edges. Therefore, it is clear from the use of the word ‘edge’ that edge refers to places in the image where there appears to be a jump in brightness value or a local extremum in brightness value derivative.

It is through the use of edge detection techniques that local contrast

differences can be determined. An edge is an area between two regions that differ according to some measure of homogeneity. An edge operator is a mathematical operator with a small spatial extent, designed to detect the presence of a local edge in the image function. It is difficult to define a priori which local edges correspond to relevant boundaries in the images. The fact that different kinds of edge operators perform best in different task domains has prompted the development of a variety of operators. However, the unifying feature of most useful edge operators is that, they compute a direction that is aligned with the direction of maximal grey level change, and a magnitude describing the strength of this change.

Since edges are a high spatial frequency phenomena, edge finders are also usually sensitive to noise. Whereas smoothing filters are low pass filters, edge enhancement filters are high pass filters that aim to enhance edges. One simple way to interpret jumps in value or local extrema of derivatives when referring to a discrete array of image values, is to assume that the discrete array of values comes about by sampling a real valued function F defined on the domain of the image which is bounded and connected subset of the real xy-plane.

The finite difference typically used in the numerical approximation of the first order derivatives is usually based on assuming the function F to be linear. From this point of view, the jumps in value or extrema in derivative, must refer to points of high first derivative of F or to points of relative extrema in the second derivatives of F. Edge detection must then involve fitting a function to the sample values. Edge finders should then regard the digital picture function as a sampling of the underlying function F, where some kind of random noise has been added to the function values. Sharp discontinuities can reveal themselves in high values for estimates of first partial derivatives. Relative extrema in first directional derivative can reveal themselves as zero crossings of the second directional derivative. Thus, if it were known that the first and second partial derivatives of any possible

underlying image function had known bounds, then any estimated first or second order partials that exceed these known bounds must be due to discontinuities in value or in the derivative of the underlying function. The last is the basis for the gradient magnitude and the Laplacian magnitude edge detectors, that often appear in literature.

Edge detection operators can be said to fall into three main classes. First, operators that approximate the mathematical gradient operator. Second, template matching operators that use multiple templates at different orientations and third, operators that fit local intensities with parametric edge models. In order to establish the most appropriate edge detection technique for the thesis, a number of available methods were reviewed including: the sharpening method using gradient operators [26], the Roberts operator [145], the Prewitt masks [29], the Sobel edge operator [146], the Kirsch edge finders [147], the Frei Chen edge masks [28], the Laplacian operator [26], the Davies operator [148], the Rosenfeld edge detector [149], the Hueckel operator [150], the Shammugan-Dickey Green edge operator [151], the well known Marr Hildreth edge operator [152], the Haralick edge detector [153], the Canny edge detector [144], the Difference of Boxes edge finders [154], the Nalwa interpolation scheme [155], the edge detection using Walsh functions [156],and finally the Maxgrad operator [129].

From all the previously mentioned edge operators Canny's edge finder is considered to be the one most widely applied, efficient and successful edge operators. The fact that Canny's operator has been so widely used and the appearance of a number of several new powerful edge operators, had led in the selection of a new edge operator for the current application. The new operator, the Maxgrad edge detector, was originally introduced in a recent paper by Imme [129]. To justify the selection of this particular edge operator, experimental results that compared the performance of Canny's edge finder with that of the Maxgrad edge operator will be presented. The performance criteria which a good edge operator must meet, are these specified in Canny's

well known thesis. Namely, good detection of edges, good localisation and a single response to an edge. Figure 6.11, demonstrates the performance of both operators with respect to the good localisation and good detection criteria while Figure 6.9 shows the performance of both operators regarding the single response criterion. Imme's [129] comments regarding both tables were that, 'the Maxgrad method never displaces edges and therefore fulfills Canny's criterion of good localisation of the edge. The Canny method displaces edges frequently for small structures and less frequently for the larger structures. For small structures with width one or two, both methods depend on σ with respect to the good detection criterion. The smaller the σ the better is the detection rate'.

ABLE 1 - Number of errors for the Canny and Maxgrad operator for noise free structures with rying σ and structure width.

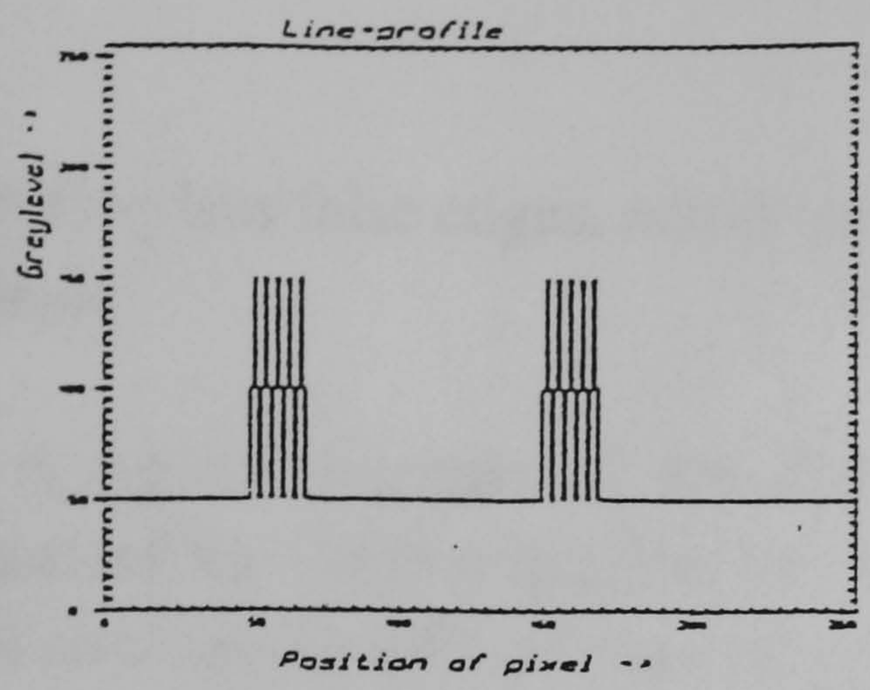
method	σ	w	Structure width 1 dis-placed edges	1 unde-tected edges	2 dis-placed edges	2 unde-tected edges	5 dis-placed edges	5 unde-tected edges
Canny	0.7	5	0	0	0	0	0	0
Maxgrad	0.7	5	0	0	0	0	0	0
Canny	1.4	11	4	4	0	0	0	0
Maxgrad	1.4	11	0	0	0	0	0	0
Canny	2.0	15	4	20	4	4	0	0
Maxgrad	2.0	15	0	18	0	0	0	0
Canny	3.0	21	4	20	4	20	4	4
Maxgard	3.0	21	0	16	0	18	0	0

BLE 2 - Number of errors for the Canny operator and Maxgrad for noisy structures with varying and structure width.

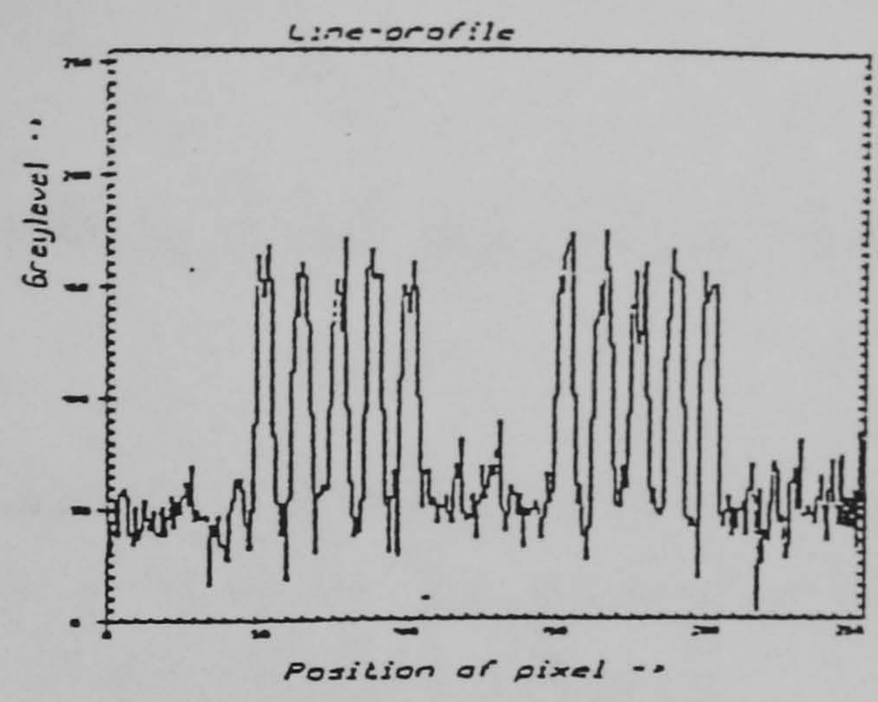
method	σ	w	Structure width 1 falsy detected	1 unde-tected	2 falsy detected	2 unde-tected	5 falsy detected	5 unde-tected
Canny	0.7	5	64	0	60	0	54	0
Maxgrad	0.7	5	76	0	69	0	57	0
Canny	1.4	11	45	8	39	0	29	0
Maxgrad	1.4	11	34	6	30	0	22	0
Canny	2.0	15	36	18	29	3	19	0
Maxgrad	2.0	15	24	13	23	1	17	0
Canny	3.0	21	15	19	18	18	10	3
Maxgard	3.0	21	13	15	9	13	7	0

Figure 6.9 [Imme 1989].

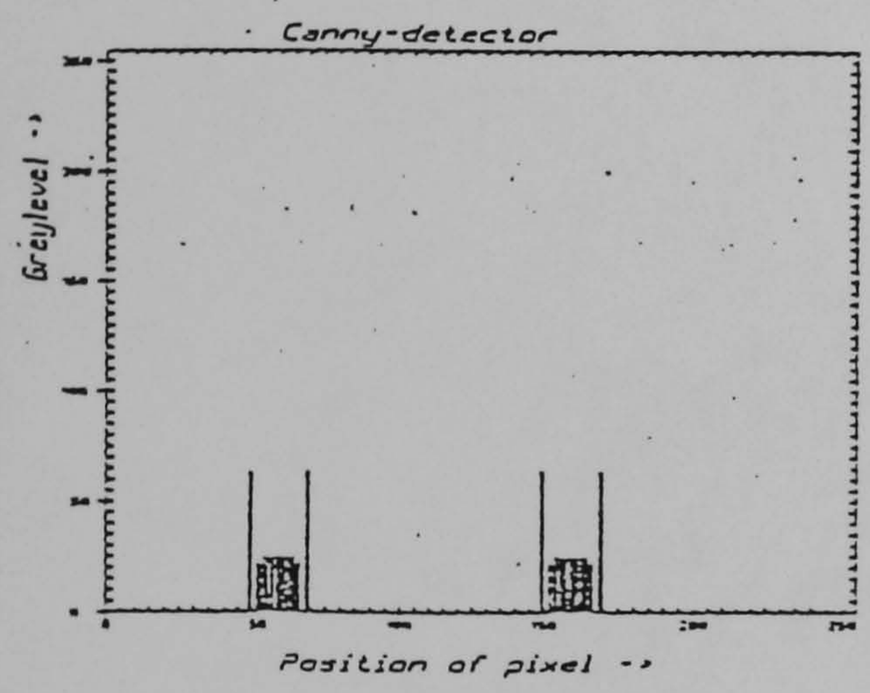
From Figure 6.9 it can be concluded that for a low σ the detection rate of false edges for the Canny method is less than for Maxgrad method and for both methods all edges are detected. 'For larger σ the Maxgrad always produces less false edges and omits less edges than the Canny method' [129].



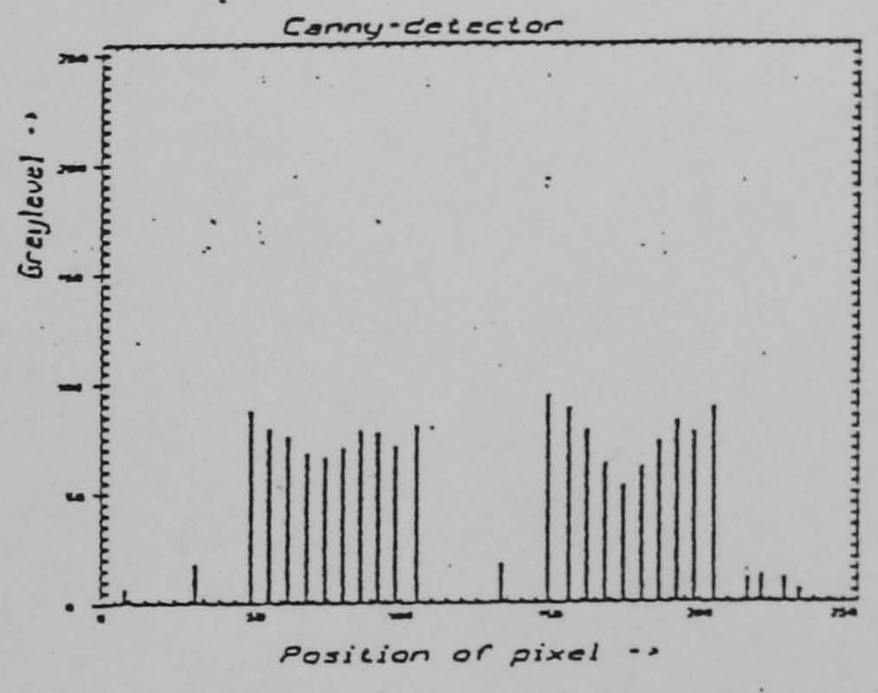
1a: original input signal with structure width = 1



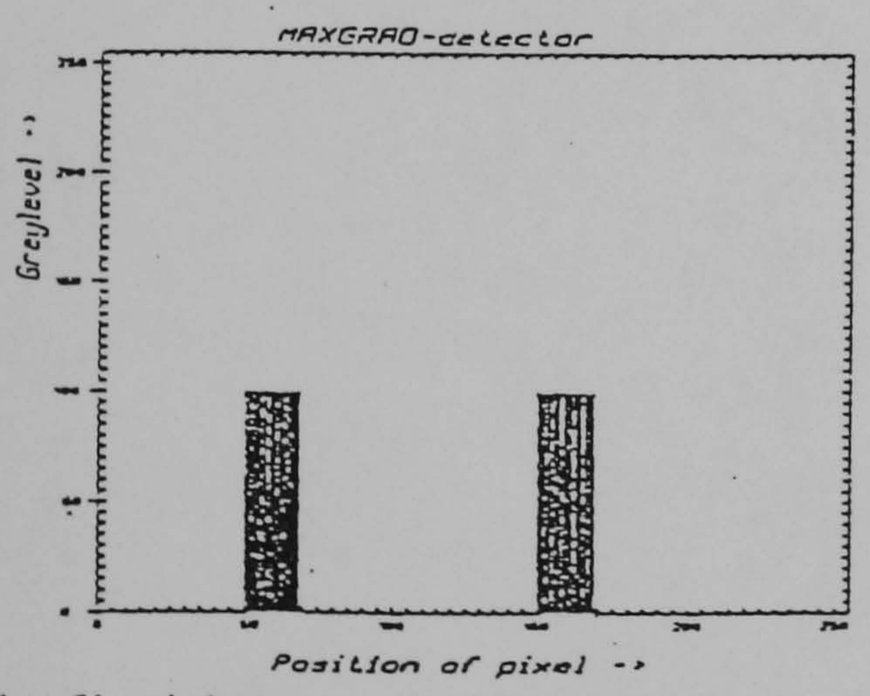
2a: noisy input signal with structure width = 5



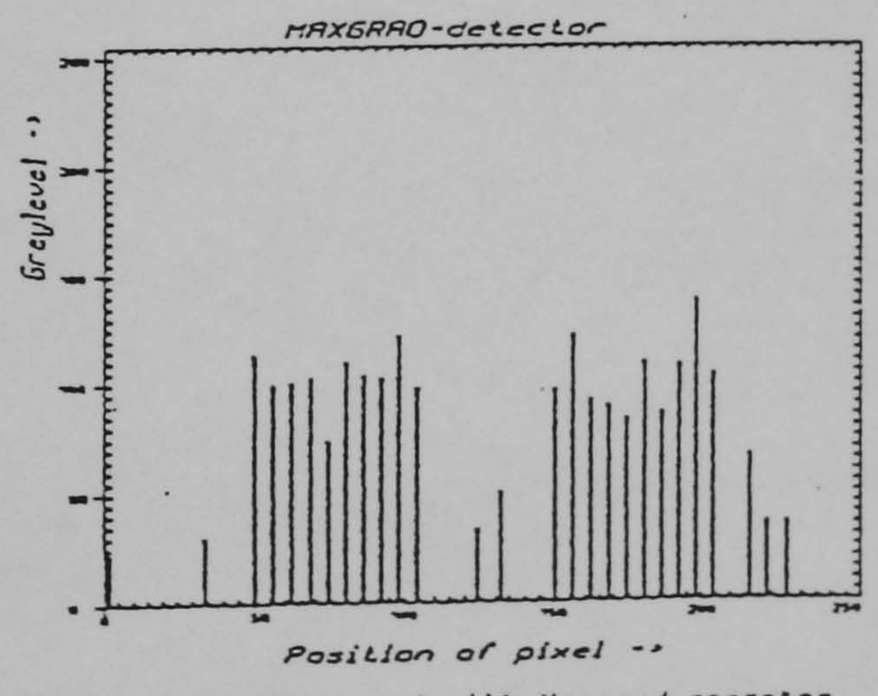
1b: Signal 1a processed with Canny operator and σ=1.4



2b: Signal 2a processed with Canny operator and σ = 3.0



1c: Signal 1a processed with Maxgrad operator and σ=1.4



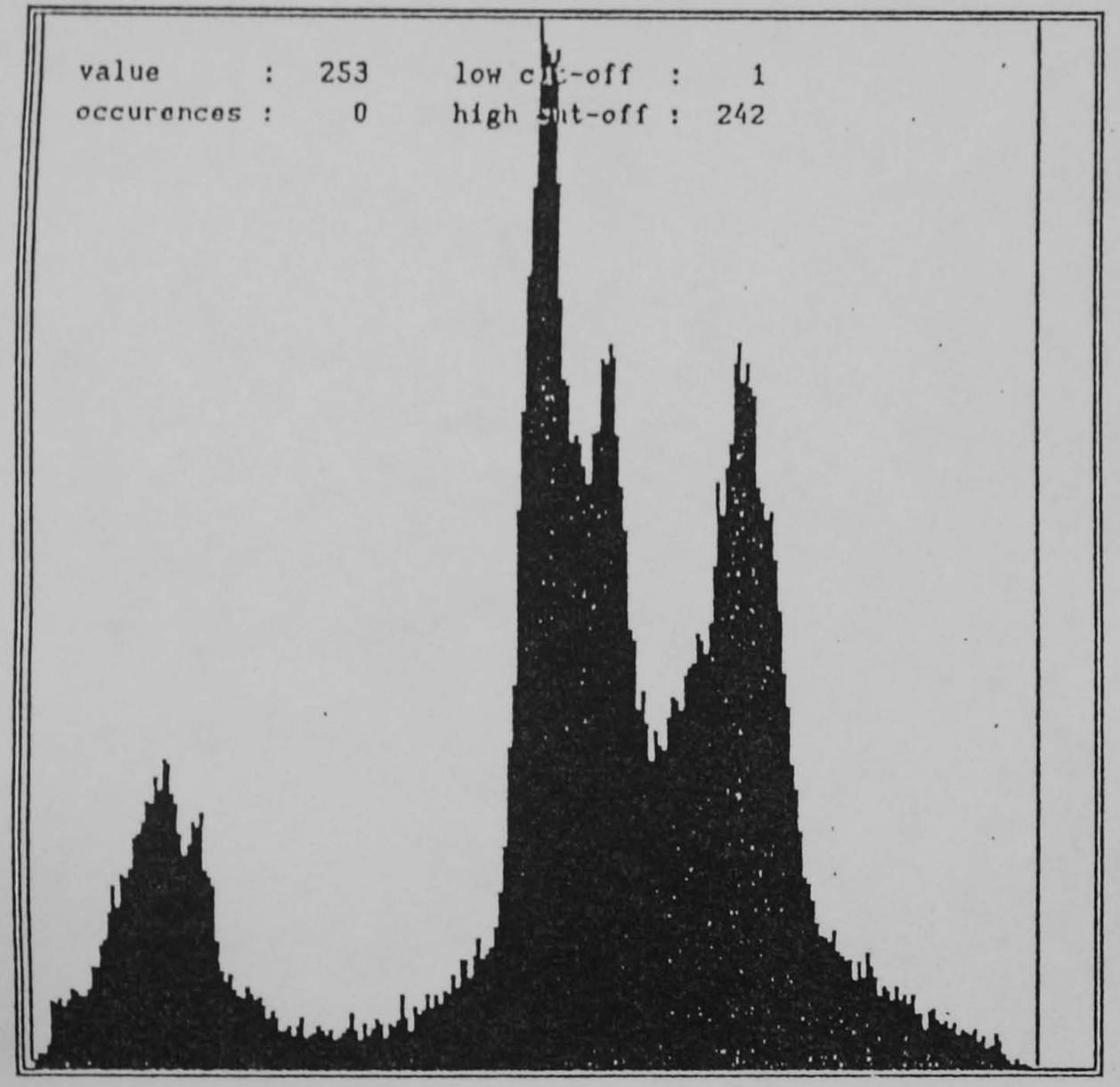
2c: Signal 2a processed with Maxgrad operator and σ = 3.0

Figure 6.10 [Imme 1989].

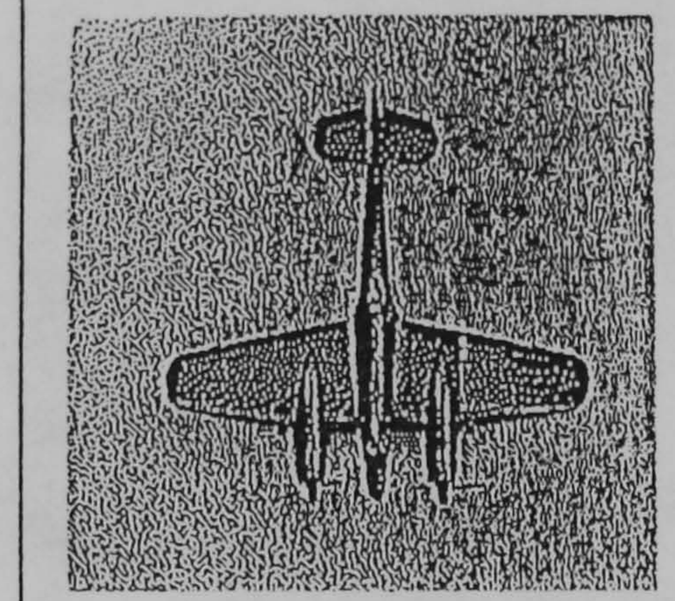
From Figure 6.10 it can be seen that both methods fulfill the criterion of Canny's postulate, namely the single response of an edge detector. Imme [129] concluded that, 'all by all, the Maxgrad operator supersedes the Canny operator in the precision of detecting the right localisation of the edge and in

producing less false edges, admitting that these signals are a hard test for both methods'.

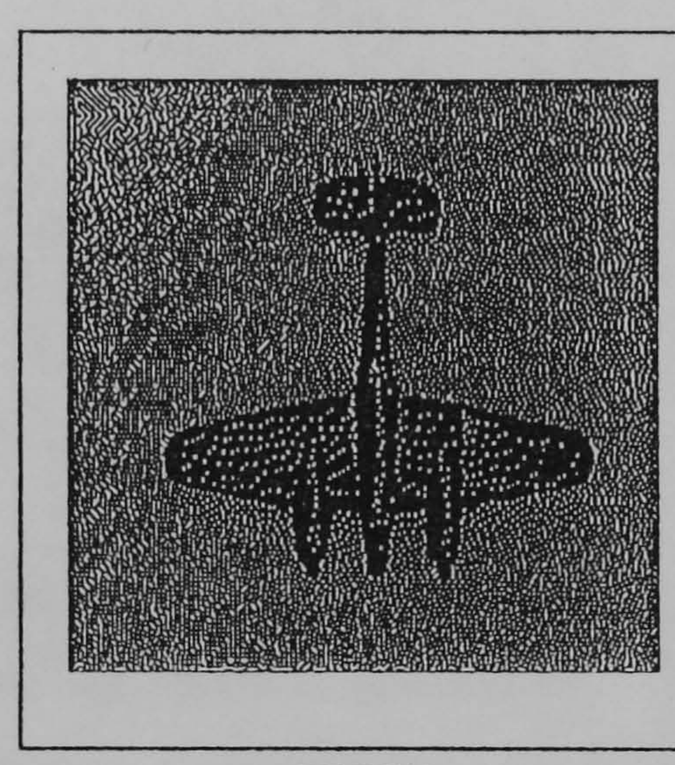
A typical example of the above image processing techniques in the context of the current application can be seen in the following illustrations from an actual aircraft pre-processing sequence [Figure 6.11 (a-c)].



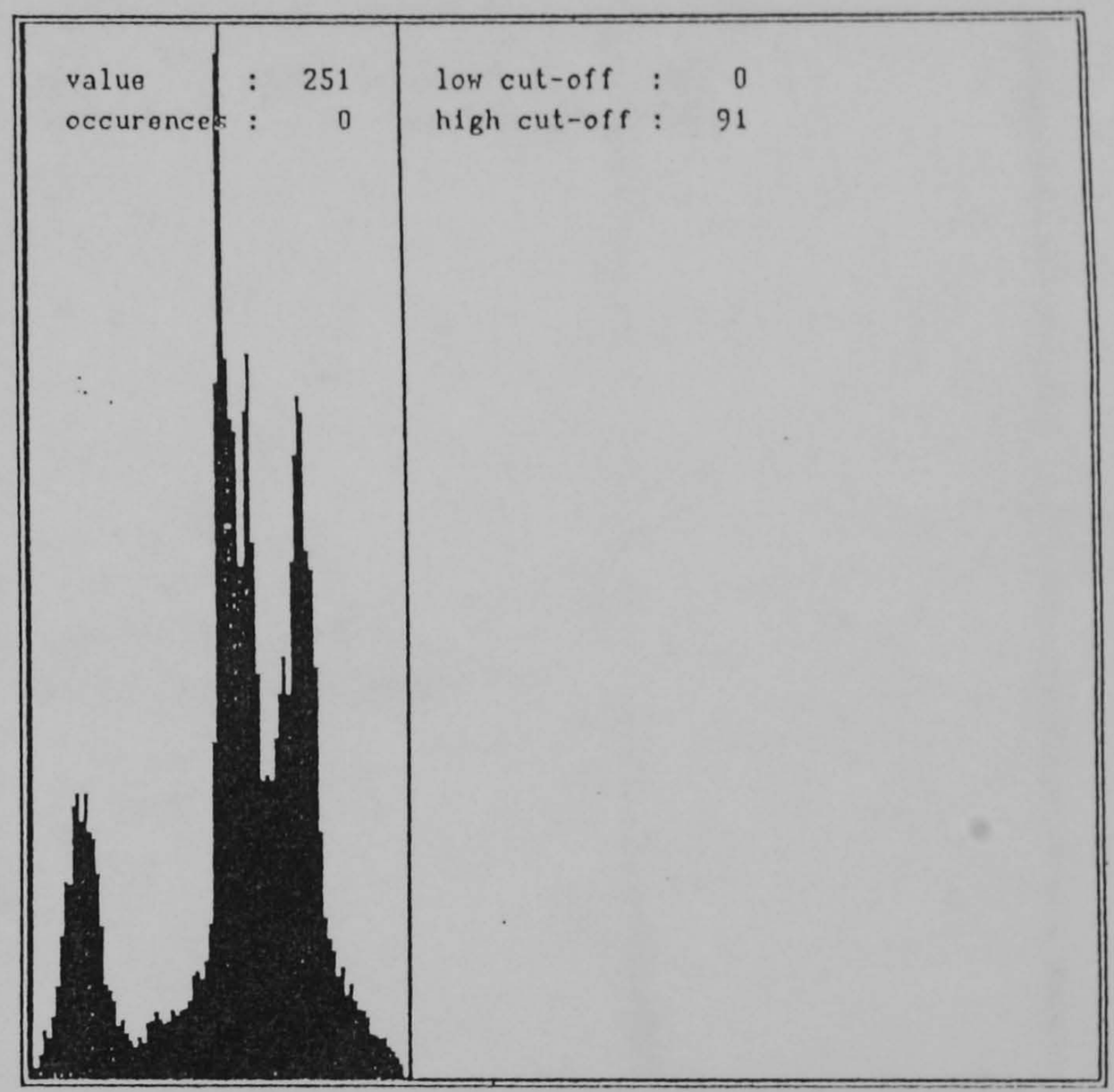
The histogram of the original top view aircraft image.



[LEFT]
The original aircraft top view image.
[512 by 512 pixel picture]

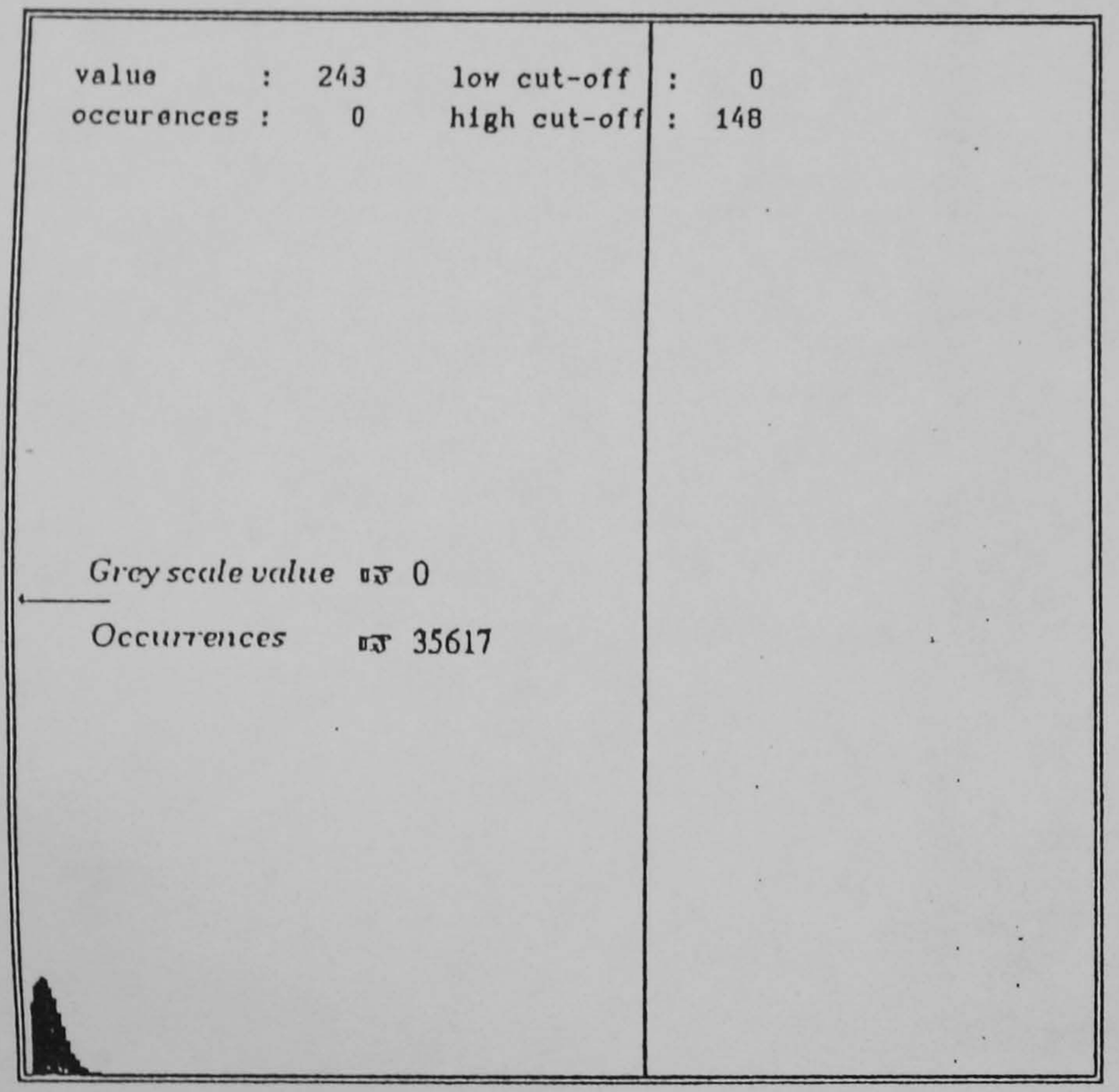


[RIGHT]
The resulting raster image after smoothing using a Gaussian mask of size 3X3 pixels

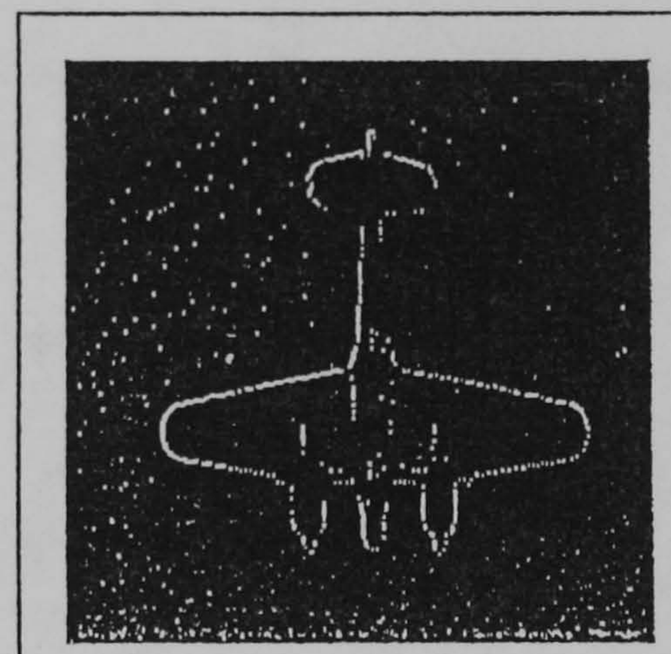


The histogram of the resulted Gaussian image.

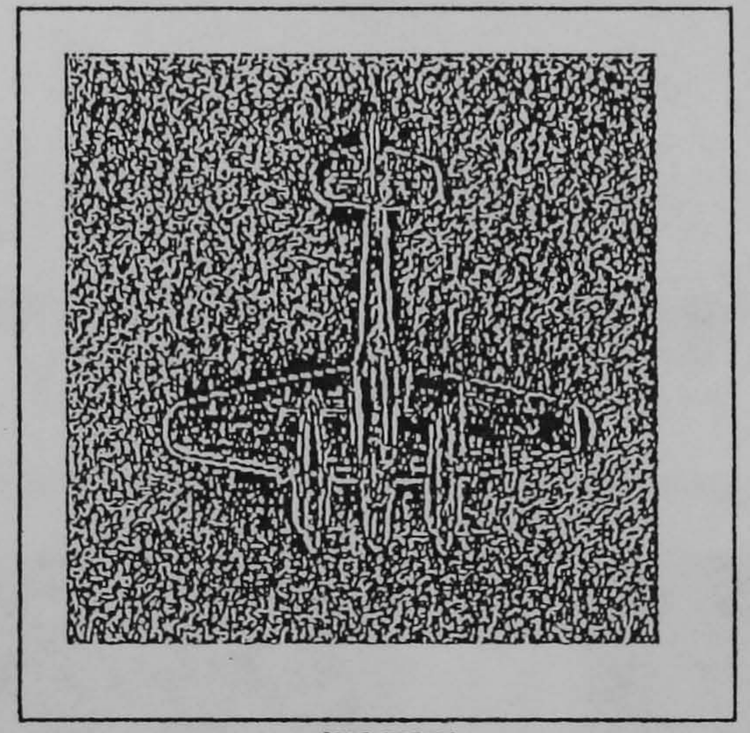
Figure 6.11(a)



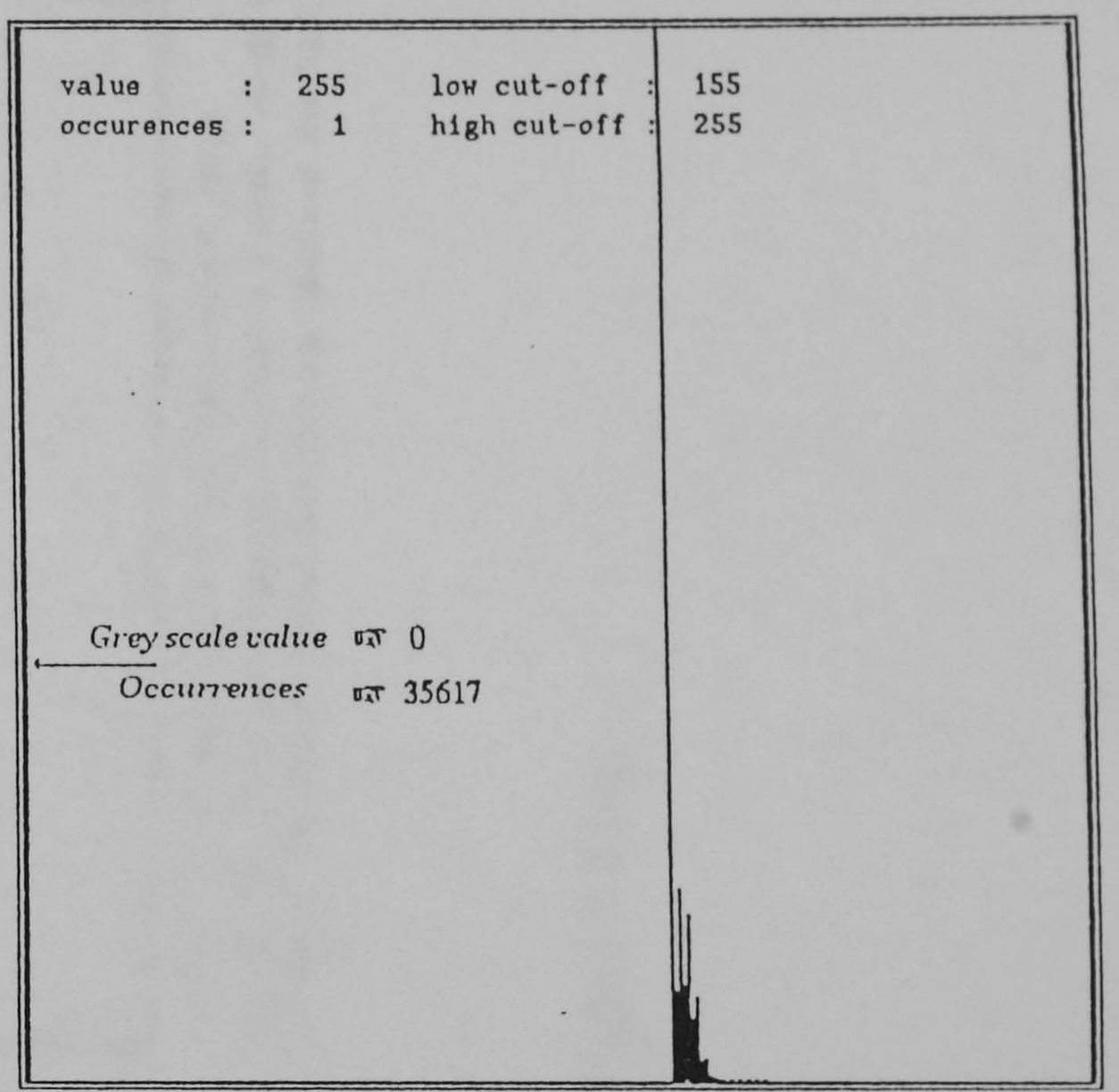
The histogram of the edge detected aircraft image.



[LEFT]
The raster image resulted after edge detection with the Maxgrad edge operator.

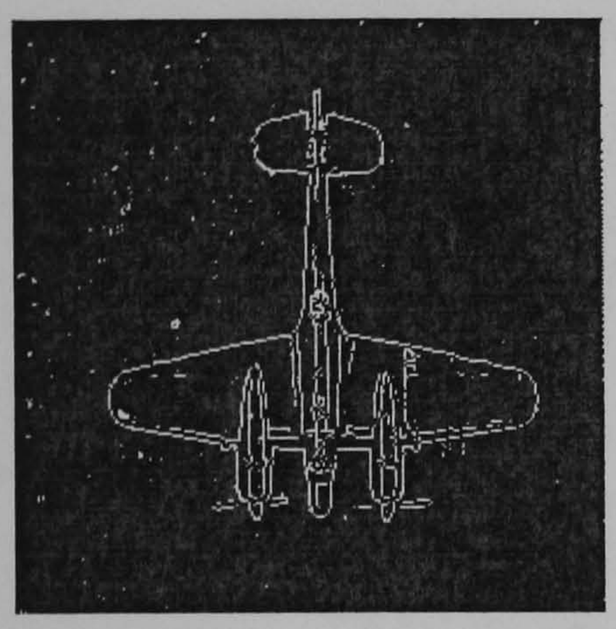


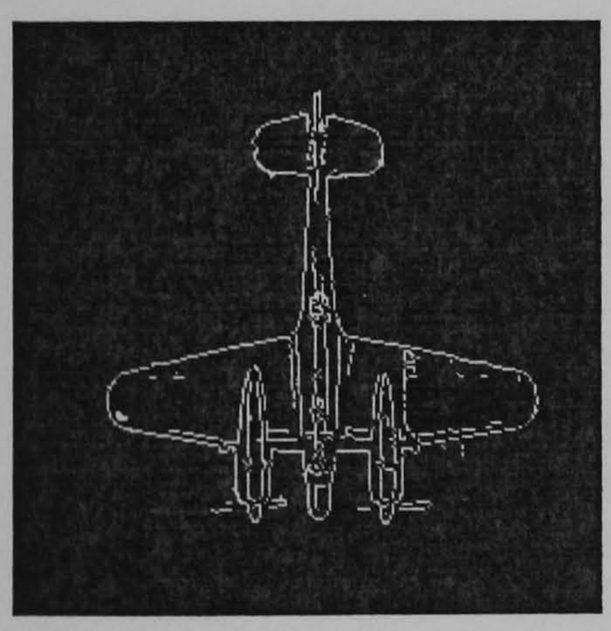
[RIGHT]
Raster image of the shifted edge map created by shifting the histogram towards the high intensities.
(threshold >= 155)



The histogram of the new edge map.

Figure 6.11(b)





value : 93 low cut-off : 0
occurences : 0 high cut-off : 255

Grey scale value ☞ 0 255 ☜ *Grey scale values*
Occurrences ☞ 63136 2400 ☜ *Occurrences*

Above : The histogram of the final binary aircraft image.
Left Up: The picture after histogram thresholding (thresh. <=165)
Left Down: The picture after median filtering. (masks of size 2&4)

Figure 6.11(c)

6.3 Quadtrees.

Region representation is an important area of research in digital image processing. The various representations currently in use include the run length technique, the chain codes and the binary trees. A recently proposed idea by Klinger [157], the quadtree data structure, has received considerable attention. Recently, the renewed interest in quadtree data structure is mainly due to the significant work of Hunter and Steiglitz [158][159][161] as well as in ongoing research work showing the interchangeability of the quadtree representation with other representations like the boundary codes [162] and rasters [163] [Figure 6.12].

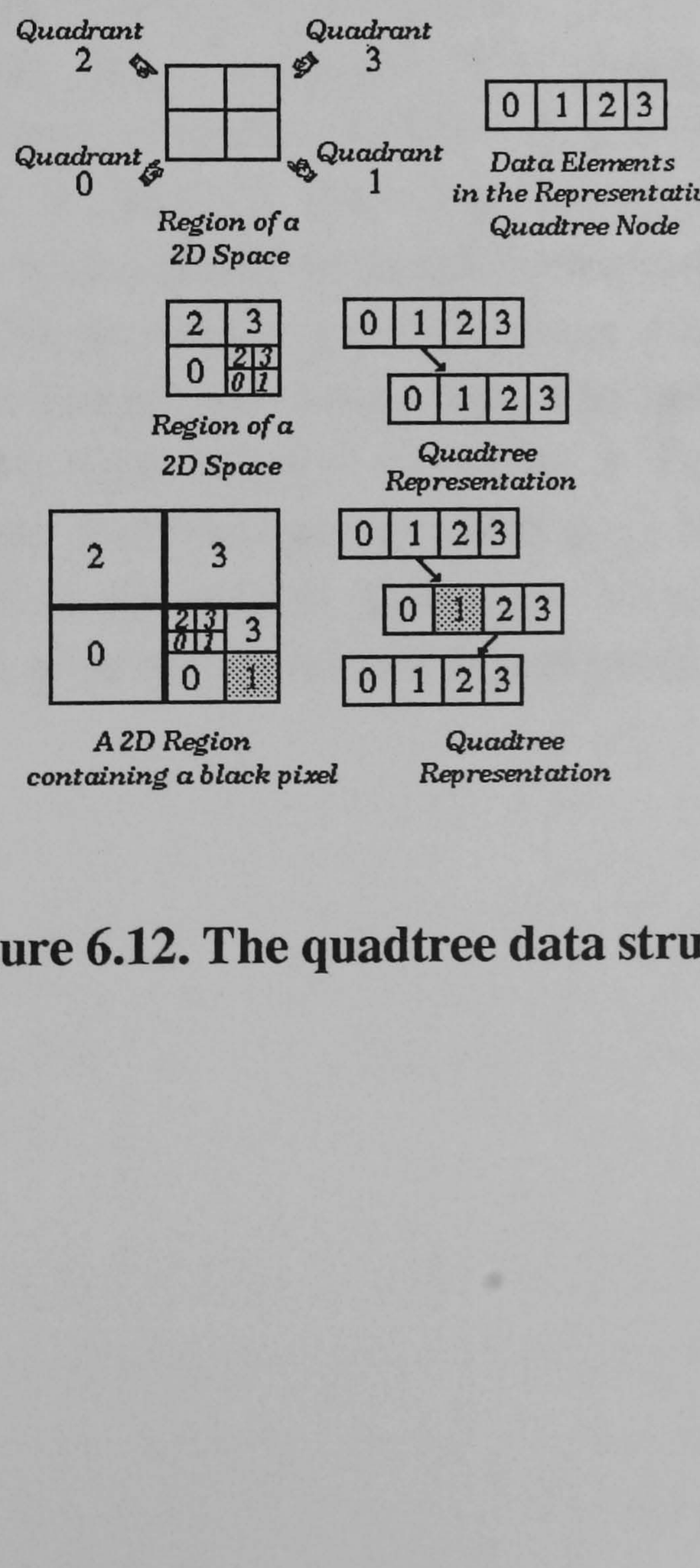


Figure 6.12. The quadtree data structure.

6.3.1 Definition of the Quadtree Data Structure.

The quadtree was originally created as an alternative method to the binary array image representation aiming to save space by grouping together similar valued image regions. More importantly though, the quadtree's hierarchical nature offers additional significant savings in execution time. In particular, computational algorithms based on quadtree's basic image processing operations display execution times that are only dependent on the number of blocks in the images and not their size (e.g., for connected component labelling [160]). Given an image of size 2^N by 2^N pixels, a quadtree representation can be formulated as follows: For every same valued pixel in the image, the tree consists of a single node labelled with that value. Otherwise, the image is divided into quadrants, and the root is given four children, representing these quadrants. The process is then repeated as follows: If a quadrant consists entirely of a single valued pixels, its corresponding node is labelled with that value; if not, it is split into subquadrants and its node obtains four children representing them, and so on. The final result of this process is a tree of degree four (a 'quadtree') whose leaf nodes represent image blocks that have constant values [Figure 6.13]. Clearly, the quadtree representation described is likely to be significantly uneconomical for grey scale images, in which large blocks of constant value will be rare. Therefore the current application is mainly focused on binary images, as they arise after edge detection is performed on a grey scale image.

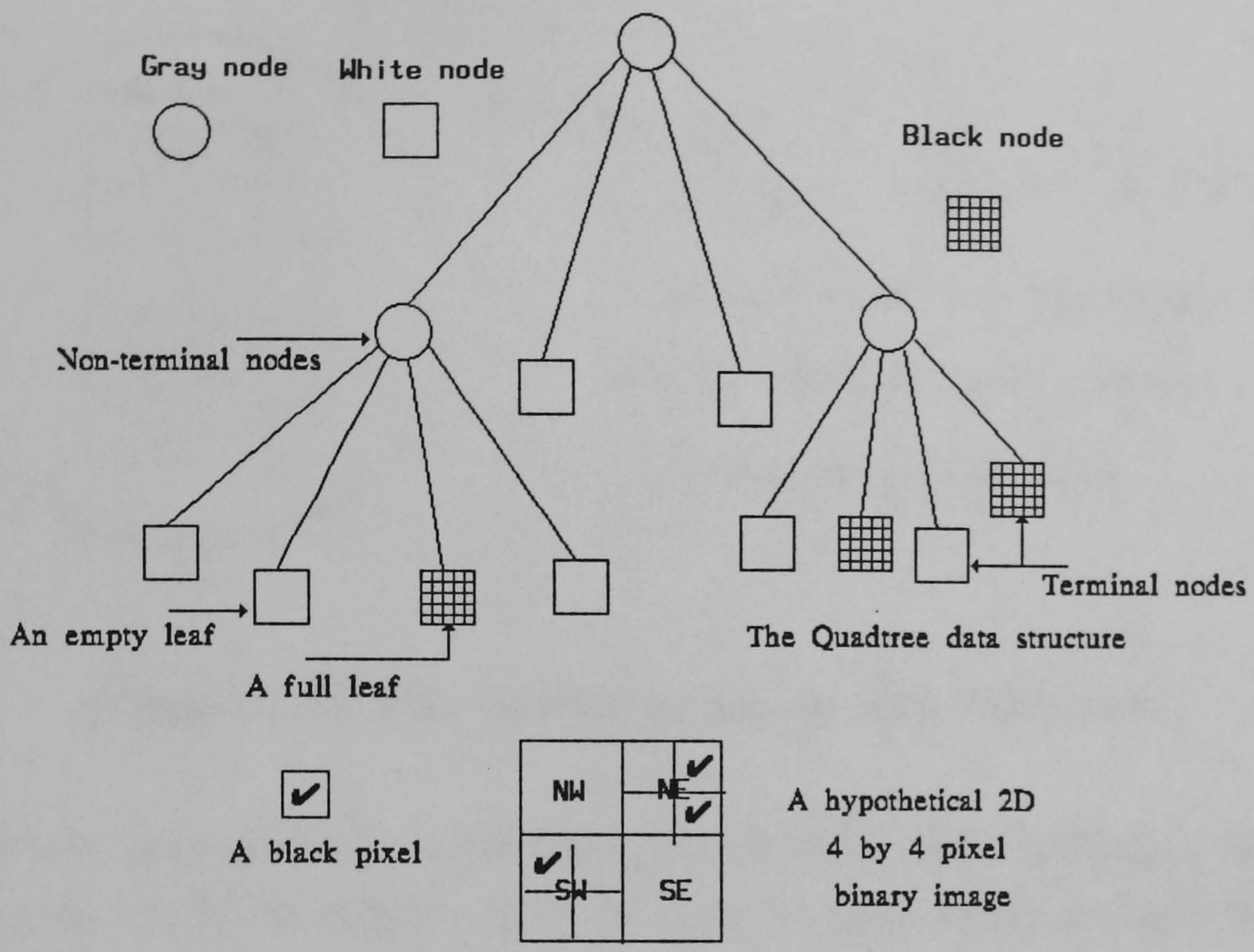


Figure 6.13. Terminal and non-terminal nodes in a quadtree.

There are two main reasons for applying quadtrees. First, they exploit the coherence of many pictures and are considered to be efficient methods for image compression. Second, they describe in their unique format the underlying picture structure. In picture processing, the last property results in algorithms capable of obtaining image 'impressions' at varying resolutions, and focusing attention quickly on areas of interest. As in both types of application, investigating the coherence of pictures in order to reduce storage requirements is required, the quadtrees are usually stored in the form of a linked tree structure [Figure 6.14]. This has links from each father quad to its four sons, and often additional links from son to father when backtracking is required, although the latter requirements can often be efficiently implemented using stacks.

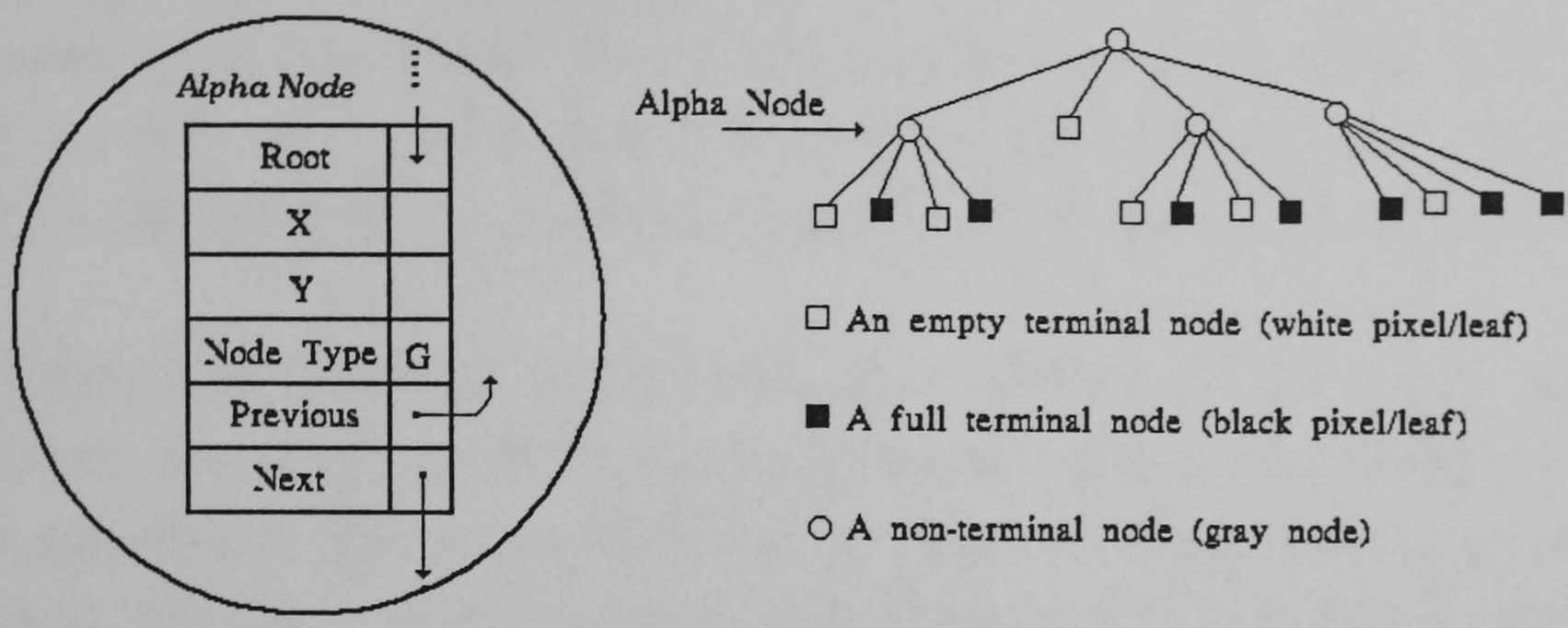


Figure 6.14. The linked quadtree data structure.

As it has been already mentioned, the regular decomposition of a region by quadrants has been largely used to resolve grey scale images into binary representations, and the dynamic data structure used to support and describe such a process is generally referred to as the quadtree of an image. Originally introduced by Klinger [157], the quadtree has been undergoing a continuous improvement. Klinger and Rhodes [161] described quadtrees by means of a systematic indexing of their elements with keys based on the node ancestors. Hunter and Steiglitz [159] applied a "rope net" consisting of pointers suitably linking a node to it neighbours. Samet [162][163], Dyer et al. [5] and Shneier [164] introduced a pointer based structure with a node linked only to its father and sons. Jones and Iyengar [165] presented a forest-like structure with root and each tree addressed by a numeric key. In the general quadtree implementation, the storage requirements for pointers from a node to its sons is not trivial and processing pointer chains in external storage is generally time consuming due to the required large number of page faults. Consequently, there has been a considerable interest in pointerless quadtree representations which can be grouped into two categories. The first treats the image as an ordered collection of leaf nodes. Each leaf is represented by a locational code corresponding to a sequence of directional codes to locate the

leaf along a path from the root of the tree. Locational codes have been used by a number of researchers to represent quadtree data structures [161] [166] [167], as well as in other related contexts [168]. The second represents the image in the form of a preorder traversal of the nodes of its quadtree [169].

Pointerless quadtree representations similar to the ones previously described, are used in the current application and are generally known as linear quadtrees. They are attractive because of their relative compactness and their efficiency in maintaining data [Figure 6.15]. A linear quadtree can be viewed as a sequential file that only supports the operation NEXT() that yields, in sequence, the leaf nodes of the tree. Many image operations can be performed by scanning the file, in its natural order, while keeping only one or two nodes in working storage. The computation of moments, and set of operations like union and intersection are of the possible operations [170] [166] [167]. The first "linearisation" of quadtrees can be found in Knuth [171], who has stored t-trees (t>=2) sequentially "with the structure implicit in the locations of nodes". Gargantini [169] [172] [173] [174] stored only the black nodes of quadtrees as sorted arrays while Abel [175] and Abel and Smith [124] adopted a B^+-tree structure. One has to compare not only memory requirements but also the ease with which elementary operations can be carried out, in order to decide which representation is best. At the moment, linear quadtrees have been successfully applied in the design of algorithms for a variety of operations [169]. In linear quadtree representation each pixel is encoded using a quaternary integer, the digits of which encode, from left to right, the successive quadrant subdivisions according to the following convention: 0 for the northwest; 1 for the northeast; 2 for the southwest and 3 for the southeast quadrant. Thus for a pixel P=($p_{n-1}\ p_{n-2} \cdots p_0$), p_{n-1} represents the first subdivision of the original picture into four quadrants, and p_0 the last partition. For example consider the picture of Figure 6.15 and the corresponding linear quadtree. Pixel P=(0123) identifies a pixel which belongs to the northwest quadrant in the first subdivision, to the northeast in

the second, to the southwest in the third and finally to the southeast in the last and fourth partition. Clearly, if four pixels such as 1000, 1001, 1002, and 1003 belong to the same quadrant relative to the n-partition, all are collapsed into one block, represented by a unique code, the last digit of which is considered as a "wild card". This process is often called condensation [169]. Clearly in the example of the previous figure, condensation would be applied a maximum of three times until 1XXX is obtained.

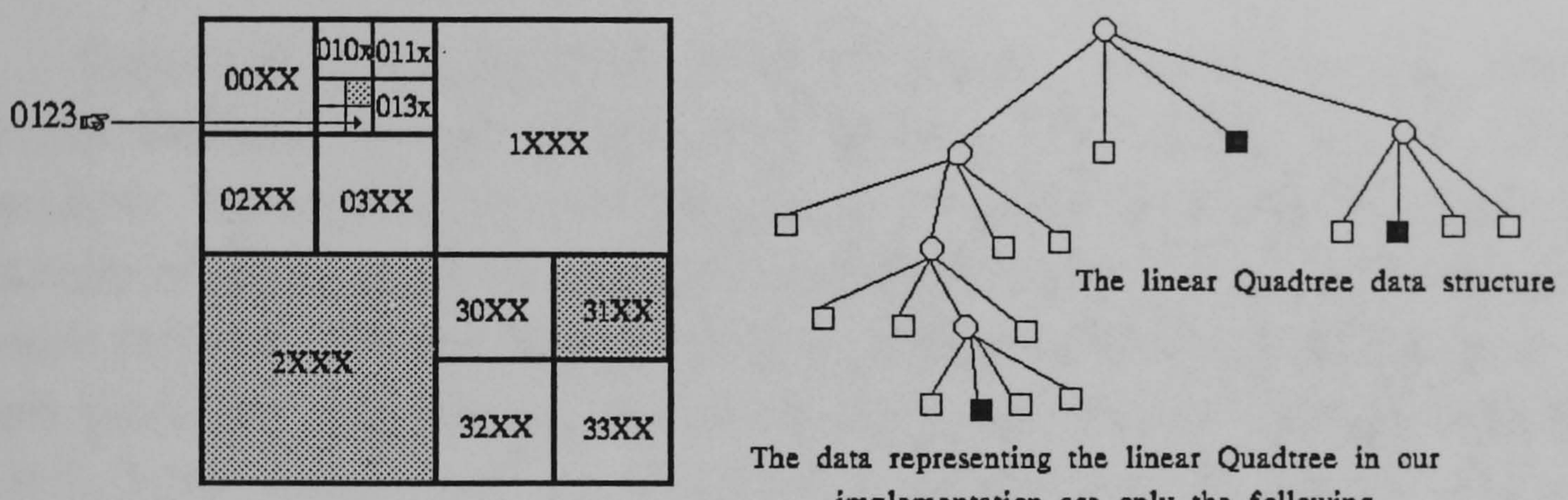


Figure 6.15. The linear quadtree representation.

6.3.2 Advantages and Limitations

As has been previously mentioned, most quadtree applications have focused on the obtainable data compression, and this has implied a linked tree structure. A linear structure, in which a storage location is assigned to every location in a quadtree down to a defined pixel level of leaf quads, does not at first seem to be useful. 'The number of storage locations required is approximately 1.3 times those needed for a pixel plane representation of the picture, which in any case is exactly what the bottom layer of the tree comprises. However because quadtrees exploit area coherence, it is possible to write to and to read from quadtrees much faster than to the corresponding

pixel plane' [163]. Comparing the linear quadtree with the linked quadtree data structure, the absence of links in the former makes it more storage efficient than might be expected. As the number of bits assigned to the links depends on the maximum number of quads allowed in the tree, linked quadtrees for data with small values consist mainly of links, not data. A linear quadtree allows an increase in the speed of data input and retrieval from the quadtree. Links do not have to be processed, and it is often possible to traverse the tree with equal facility.

Quadtrees are hierarchical data structures used mainly for compact representations of two dimensional images. The advantage of using a quadtree representation for images is that mathematically simple and computationally efficient tree traversal algorithms can be formulated that allow fast execution of operations such as superposition of two images, area and perimeter calculation, and moments computation. Klinger and Dyer [176] have shown that the quadtree representation of images can yield substantial data compression ratios. In their experiments, image compression ratios ranging between 3:1 and 33:1 were recorded, with 5:1 and 6:1 being the general compression factor. Tanimoto and Pavlidis [177] have additionally developed a recursive refining algorithm for edge detection using quadtrees and have demonstrated its computational savings.

However, the quadtree representation has certain disadvantages. The quadtree representation of an object is directly affected by the object's location, its orientation and relative size. A small change in any of these parameters could certainly create a different quadtree data structure. To tackle the effects caused by an object's translation within an image, Li et al. [178] defined a normal form of quadtrees. They proved that this quadtree representation is unique for any image under a class of translation. However, the problems arising from rotations and size changes were not investigated. Chien and Aggarwal [2] have recently proposed a representation scheme, the normalised quadtree structure, that is invariant to object translation, rotation

and size changes.

6.3.3 The Normalised quadtree Data Structure.

Instead of generating a quadtree for the entire image, Chien and Aggarwal [179] have created a normalised hierarchical data structure for each object in the image, the normalised quadtree [Figure 6.16]. The particular implementation of the normalised quadtree data structure occurs as follows. First, 'the object is normalised to an object-centered coordinate system, with the centroid as the origin and its principal axes as coordinate axes, the object is then scaled to a standard size. In this way, the normalised quadtree of an object is dependent only on the shape of the object, but not affected by its location, orientation, or relative size. With those invariant properties, the quadtree of an object can be used as a shape descriptor to aid in the identification of objects in images if appropriate views of the objects are present and is an information preserving shape descriptor' [179] [Figure 6.17]. The geometric properties of the centroid and principal axes of an object (invariant to positional and size changes) are important in the implementation of the normalised quadtree data structure. The definitions of the principal axes and principal moments are briefly discussed in the following lines, while a clear mathematical proof of the entire procedure can be found in [179]. The notions of moments, moments of inertia, and principal axes originate from mechanics. Let J be a two dimensional object, with a density function f(x,y).

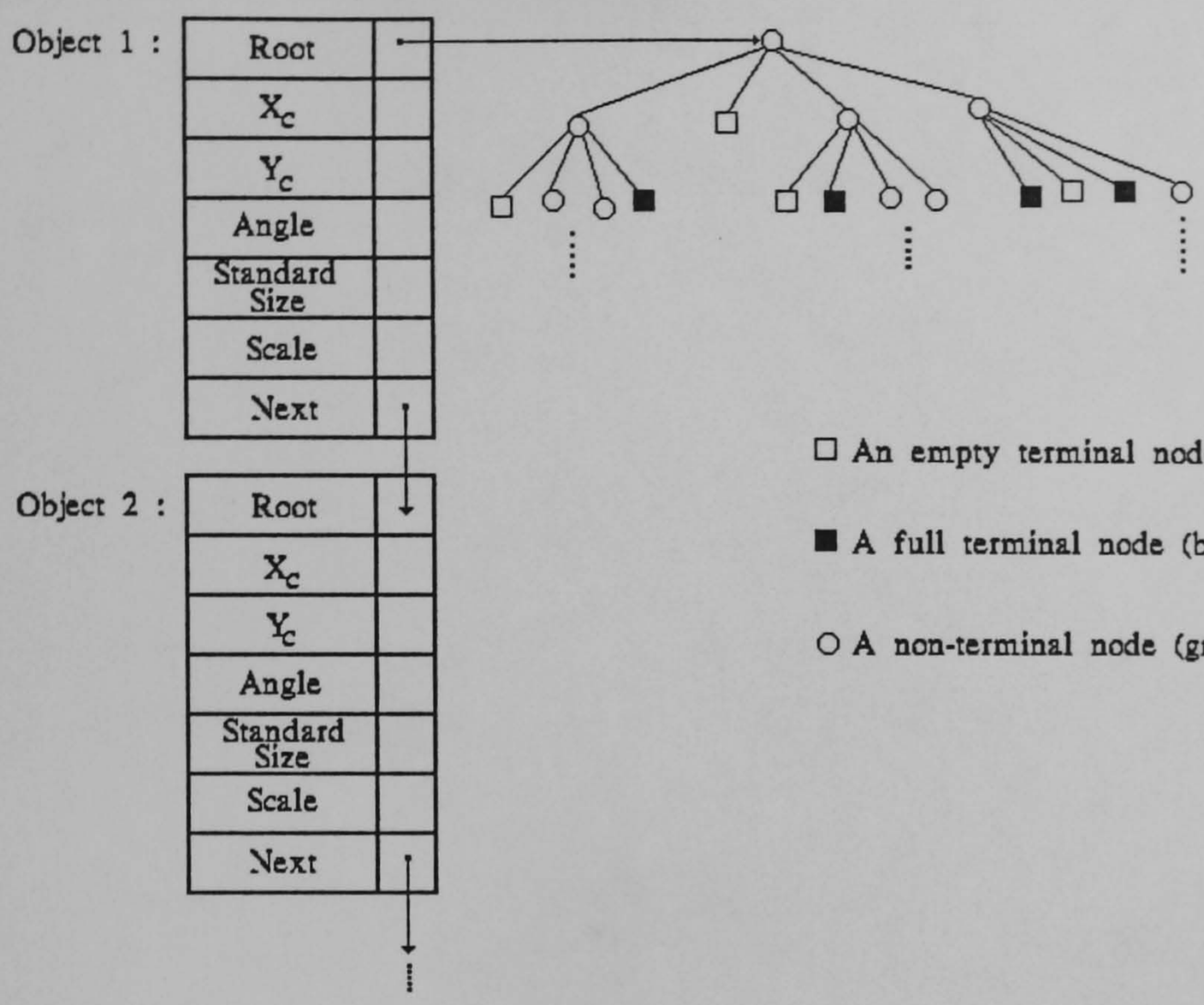


Figure 6.16. The linked normalised quadtree data structure [Chien & Aggarwal 1983].

The moment $M_{p\ q}$ of order (p+q) of J is defined as

$$M_{pq} = \sum_{J} x^p\ y^q\ F\ (x\ ,y\ )$$

where f(x,y) is the mass of the point at (x,y) and p and q are non-negative integers. Let $(x_c\ ,y_c\ )$ be the centroid and M_J the mass of J. Then

$$M_J = M_{00} \quad a\ n\ d \quad (x_c\ ,y_c\ ) = (\frac{M_{10}}{M_{00}}\ ,\frac{M_{01}}{M_{00}}\ )$$

The central moment I_{pq}, with respect to the centroid (x_c, y_c), is defined as

$$I_{pq} = \sum_{J} (x - x_c)^p (y - y_c)^q f(x, y)$$

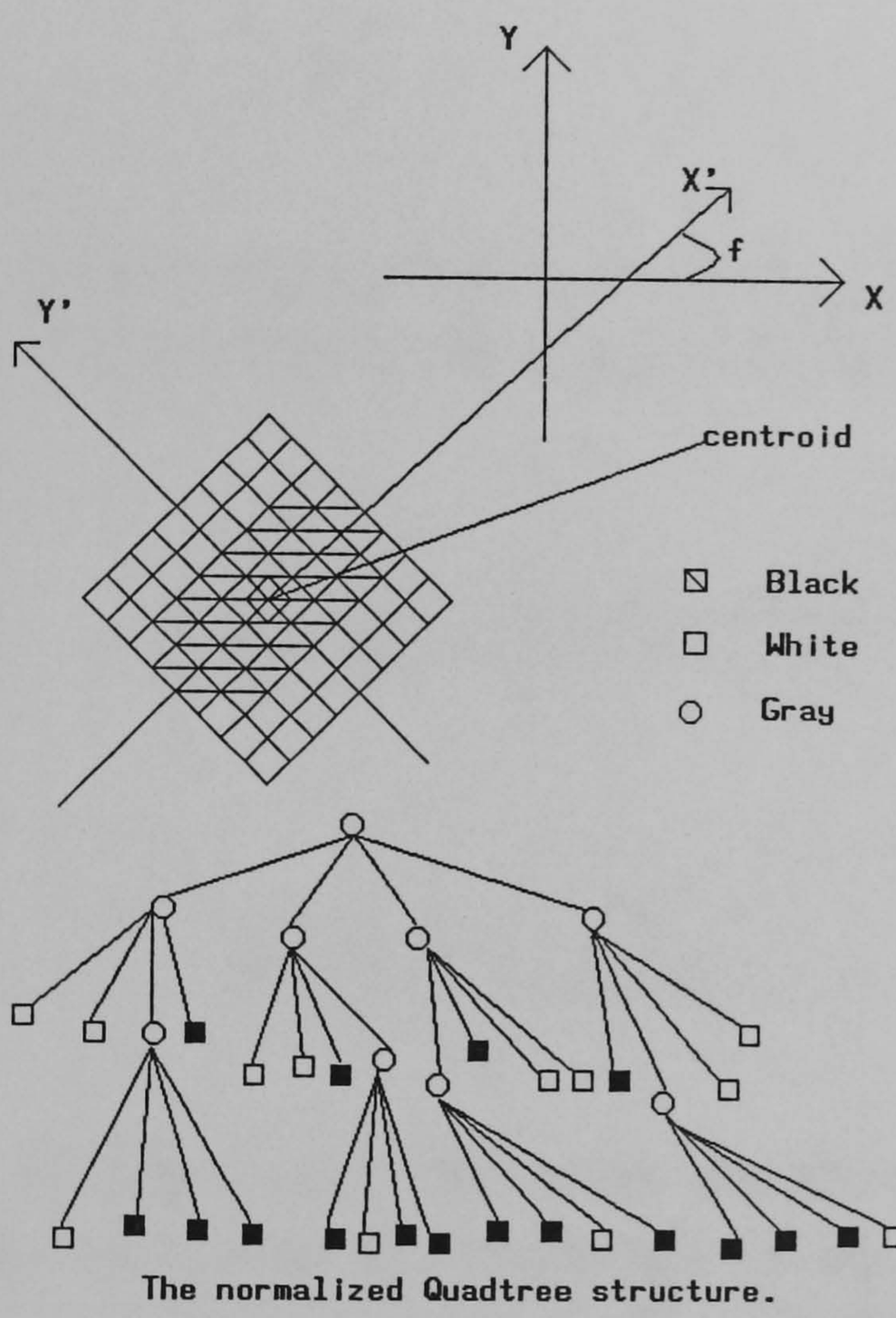


Figure 6.17. The normalised quadtree procedure and representation [Chien & Aggarwal 1983].

Of particular interest for the normalised quadtree data structure design are the second order moments: the moments of inertia I_{20} and I_{02}, and the product of inertia I_{11}. If the coordinate axes are rotated about the centroid, the central moments will change accordingly and the product of inertia will be

zero at some rotational angle Θ_p. The new coordinate axes were named principal axes, and the moments of inertia were called principal moments of order 2. The principal axes were obtained by computing the eigenvectors of the moment of inertia matrix from the following equation [45]:

$$\begin{bmatrix} I_{20} & -I_{11} \\ -I_{11} & I_{02} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = K \begin{bmatrix} x \\ y \end{bmatrix}$$

'The eigenvectors ([x,y]) correspond to the two principal axes, and the eigenvalues (K) are associated with the principal moments. It has been shown [45][180] that

$$K = \frac{(I_{20}+I_{02}) \pm \sqrt{(I_{20}-I_{02})^2+4I_{11}^2}}{2}$$

$$\Theta_p = \frac{1}{2} tan^{-1} \frac{2I_{11}}{I_{20}-I_{02}}$$

There are two principal axes: the major and the minor principal axes (associated with the maximum and minimum of the moments of inertia, respectively). There are also two choices for the direction of the major principal axis. To specify a unique orientation of the major principal axis, two additional constraints should be tested: (1) $I_{20} \geq I_{02}$ and (2) $I_{03} < 0$. If the first condition is violated, Θ_p is increased by 90^o. If the second condition is not satisfied, Θ_p is increased by 180^o. The final value of Θ_p is then defined as the principal angle' [179].

The application of the invariant properties of the centroid and principal axes in pattern recognition is not a new concept and have been used by many

researchers. A complete system of two dimensional moment invariants under positional and size changes have been derived by Hu [45]. Rutovitz [181] used the centroid and major principal axis in order to locate a chromosome, while Chow and Aggarwal [182] defined the modified principal axes, and used them as shape descriptors aiming to analyse planar curvilinear moving objects.

Studying all the above information supports the opinion that some form of pre-processing of the original image data is necessary in order to normalise each object within the picture with respect to its centroid and principal axes. The pre-processing should additionally take place prior to the generation of the object's quadtree data structure. A minimal square enclosing the object, whose sides are parallel to the principal axes and with the centre at the origin, should also be found. The picture area defined by this square is then scaled to a standard size square. The square's standard size is a power of two and 'the nearest to the size of the minimal enclosing square since the quadtree representation of an object will remain unchanged only when the size change is a power of two' [179]. The quadtree representation of the object can then be generated based on the scaled area, and should consequently be invariant to positional and size changes.

6.4 The Octree Structure

The aim of this chapter is to describe a position and size invariant three dimensional object identification algorithm and demonstrate its effective implementation within the previously presented ADAM artificial neural architecture. The normalised linear quadtree data structure is a typical two dimensional area representation scheme which although it possesses the desirable positional and size invariance properties required by the thesis, can't satisfy the added constraint of the third dimension. An upgraded version of the normalised two dimensional linear quadtree data structure (the octree) must clearly be introduced that could satisfy the additional requirements of

the current application.

6.4.1 Introduction.

The availability of an efficient and complete representation of three dimensional objects is crucial to many computer vision. Among the representation methods that have been used in the past, are the polyhedra described by polygons whose vertices are given as coordinate triples and space arrays whose elements are binary numbers as they correspond to empty space. There has been an extensive research into polyhedra applications because these methods usually offer greater accuracy in description for fewer bits of memory. In general though, the lack of effective and practical three dimensional object representation schemes and associated algorithms for manipulation, analysis and display has caused significant delays in many areas of technology [184]. Used techniques have a limited range of applicability as a result of shortcomings in two major areas. First, 'representation capabilities are not sufficiently robust to simply handle the object complexities required in a realistic environment. Second, manipulation and display algorithms performing functions such as interference detection and hidden surface removal require extremely large numbers of calculations in practical situations. They usually exhibit exponential (often quadratic) growth in the number and complexity of the objects. 'It is not believed that near-term progress in hardware technology will render such schemes practical for the vast majority of potential applications. Much more efficient data structures and algorithms that take advantage of current hardware trends are needed' [184].

Voelcker and Requicha [185] have classified three dimensional representation schemes in the following six categories:

(1) Primitive instancing. Families of objects are defined parametrically using a shape type and a limited set of parameter values per object.

(2) Spatial enumeration. An object is represented by a list of the cubical spatial cells.

(3) Cell decomposition. A generalised form of spatial enumeration in which the disjoint cells are not necessarily cubical or even identical.

(4) Constructive solid geometry. Objects are represented as collections of primitive solids that are connected using Boolean operations.

(5) Sweep representation. A solid is defined as the volume swept by a two dimensional or three dimensional shape as it is translated along a curve.

(6) Boundary representation. Objects are represented by their enclosing surfaces.

All the various previously mentioned methods for three dimensional object representation may loosely be categorised into three main classes, volumetric descriptions, wire-frame descriptions, and surface descriptions. An efficient three dimensional indexing scheme that greatly reduces the magnitude of these problems is the octree representation scheme [186]. Octrees are developed as efficient methods for space array operations and can be considered as three dimensional extensions of quadtree methods. Using octrees, objects can be represented compactly [187] and statistical image processing operations can be efficiently formulated. Furthermore this new three dimensional modelling scheme and its associated linear growth algorithms allow objects of arbitrary complexity to be encoded, manipulated, analysed and displayed interactively in real time or 'close to real time in parallel, low cost hardware' [189]. The goals of octree encoding scheme have been as follows:

(1) To develop a capability for representing any N dimensional object to any specified resolution in a common encoding format.

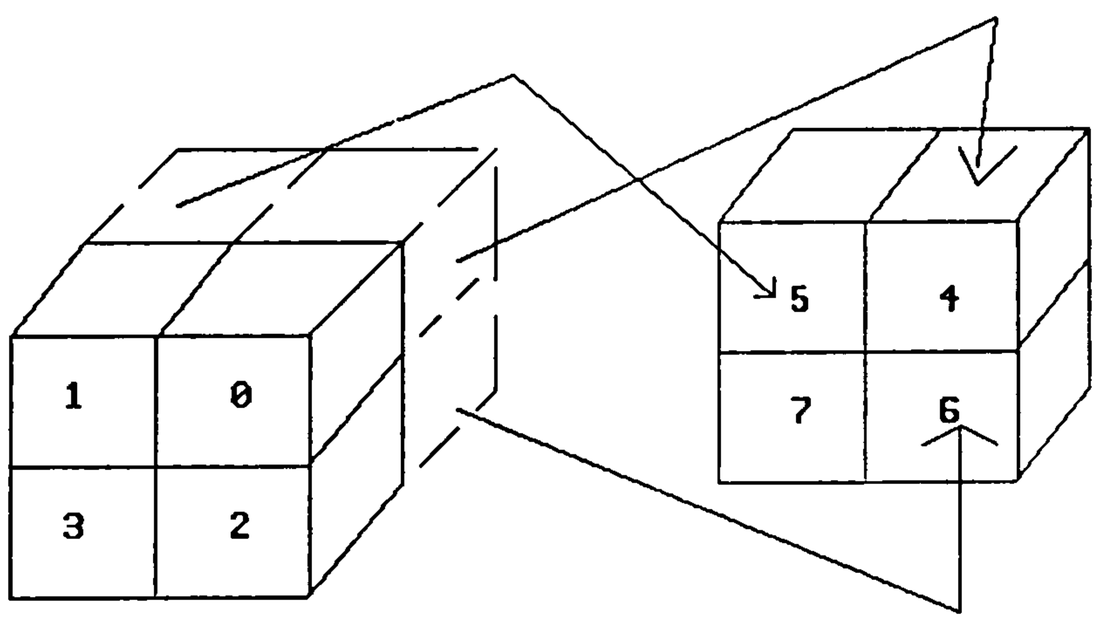
(2) To operate on any set of objects using Boolean operations and geometric operations.

(3) To implement a computationally efficient (linear) solution to the N dimensional interference problem.

(4) To develop the capability to display in linear time any number of objects from any viewpoint with shading, shadowing, orthographic or perspective view and smooth edges.

(5) To develop a scheme that do not require floating point operations, integer multiplication or integer division.

It can be concluded that, the performance of octree encoding scheme degrades gracefully as the complexity of the situation increases, and users have the ability to trade off computation against processing precision. This means that a coarse image can be generated very quickly with the higher fidelity details emerging later as more processing is carried out [187] [188] [189].



The labelling sequence in an Octree.

Figure 6.18. Numbering the octree's octans.

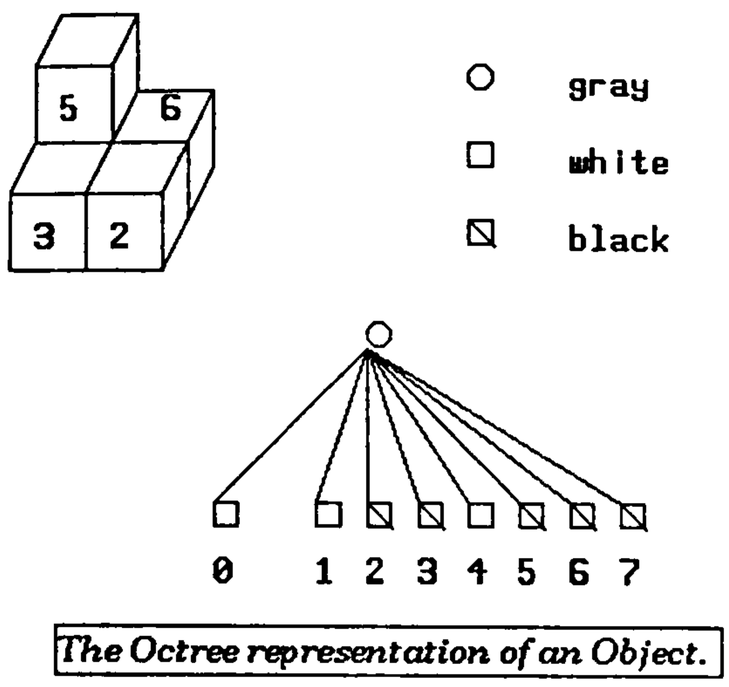


Figure 6.19. An example of an object and its corresponding octree representation.

6.4.2 The Octree Data Structure.

To create hierarchical indices for octree representation, a cubic 'object space' large enough to enclose the object is considered. The latter is frequently referred to as the 'universal cube'. The octree is a regular cellular decomposition of the object space. The object space is subdivided into eight cells of equal size. If any one of the resulting cells is homogeneous, implying that it lies entirely inside or outside the object, the subdivision stops. If however, the cell is heterogeneous, that is, intersected by one or more of the object's boundary surfaces, the cell is sub-divided further into eight subcells. The sub-division process stops when all the leaf cells are homogeneous to some degree of precision. An octree can be considered as a 2^n x 2^n x 2^n array of unit cubes. Each unit cube (octant) has associated with it a label, either full or empty. For the object of Figure 6.19 the corresponding simple octree structure is shown. An octree representation usually reduces memory requirements over an object array. At the same time, it is more complex to access the data in octree form. The geometric information contained in an octree is implicit, and can be retrieved by use of procedures. Roughly, the

location of an octant is derived by traversing the tree, and the size of the octant is determined by the level of the tree at which its corresponding node resides.

Nodes in the octree data structure are of two basic types : terminal and partial. Terminal nodes are leaves, and are further classified as either full or empty. A full leaf represents an octant that is completely filled by the modeled object. Such leaves are represented by 1 in the formal description of an octree data structure. An empty leaf represents an octant that lies outside the modeled object. Such leaves are formally represented by 0. Partial nodes represent octants that lie on the boundary of the modeled object. They are therefore neither full or empty. Partial nodes are similarly classified into two sub-groups: dividing and non-dividing. Normally, any octant represented by a partial node is further subdivided, and the resulting oct-tuple is formally represented by T. This process can obviously go on interminably without ever reaching any definitive object boundary. At some point truncating the octant subdivision is necessary. By replacing a P-node with a special nondividing node, formally represented by S, this is achieved and S-nodes now specify the shape of the object boundary. Using set theory, an octree o_i can be defined as follows:

$$\text{For } i \ \varepsilon \ \{0, 1, 2, 3, 4, 5, 6, 7\},$$

$$o_i ::= 0 \mid 1 \mid \mathrm{P}$$

$$\mathrm{P} ::= \mathrm{T} \mid \mathrm{S},$$

where the oct-tuple T=(o_1 , o_2 , o_3 , o_4 , o_5 , o_6 , o_7) is an ordered set of octrees. Each element of the oct-tuple T is called an octant. Suppose that a child node of a given octree o, is represented by o[x,y,z], where x,y,z, is the position of the octant when o is placed in a Cartesian coordinate system. Clearly, then T=(o[0,0,0], o[0,0,1], o[0,1,0], o[0,1,1], o[1,0,0], o[1,0,1],

o[1,1,0], o[1,1,1]).

The octree data structure for the representation of three dimensional objects has been investigated for several years [188] [189]. An octree can be generated from different forms of sensor data, for instance, intensity images from multiple views [179] or a series of slice images acquired in computerised tomography [190]. Initially, the octree representation was used to describe the volume occupied by a three dimensional object. Calborn et al. [197], proposed the polytree structure in which a leaf cell can be one of five types: full, empty, vertex, edge or surface. More recently, Chien and Aggarwal [179] proposed and implemented an algorithm to generate a Volume/Surface octree from multiple views (silhouettes) of an object.

The octree's requirement of eight pointers from each internal node to its children makes its space requirements quite substantial. To overcome this disadvantage the concept of pointerless representation for octrees, has been developed [166] [124] [191] where each leaf is represented by a locational key. A locational key is a numeric index obtained from the coordinates of an octant. Linear octrees are attractive because of their compactness and appropriateness in maintenance of data. In certain applications pointerless structures lead to a better space-efficient algorithm than their pointer-based counterpart. The present chapter aims at generating linear octrees for three dimensional objects obtained by the volume intersection method. In similar cases where algorithms have been developed to implement the previously mentioned technique, it has been observed that pointerless representations were more efficient in space and time than an algorithm corresponding to pointer-based conventional octrees [Figure 6.20].

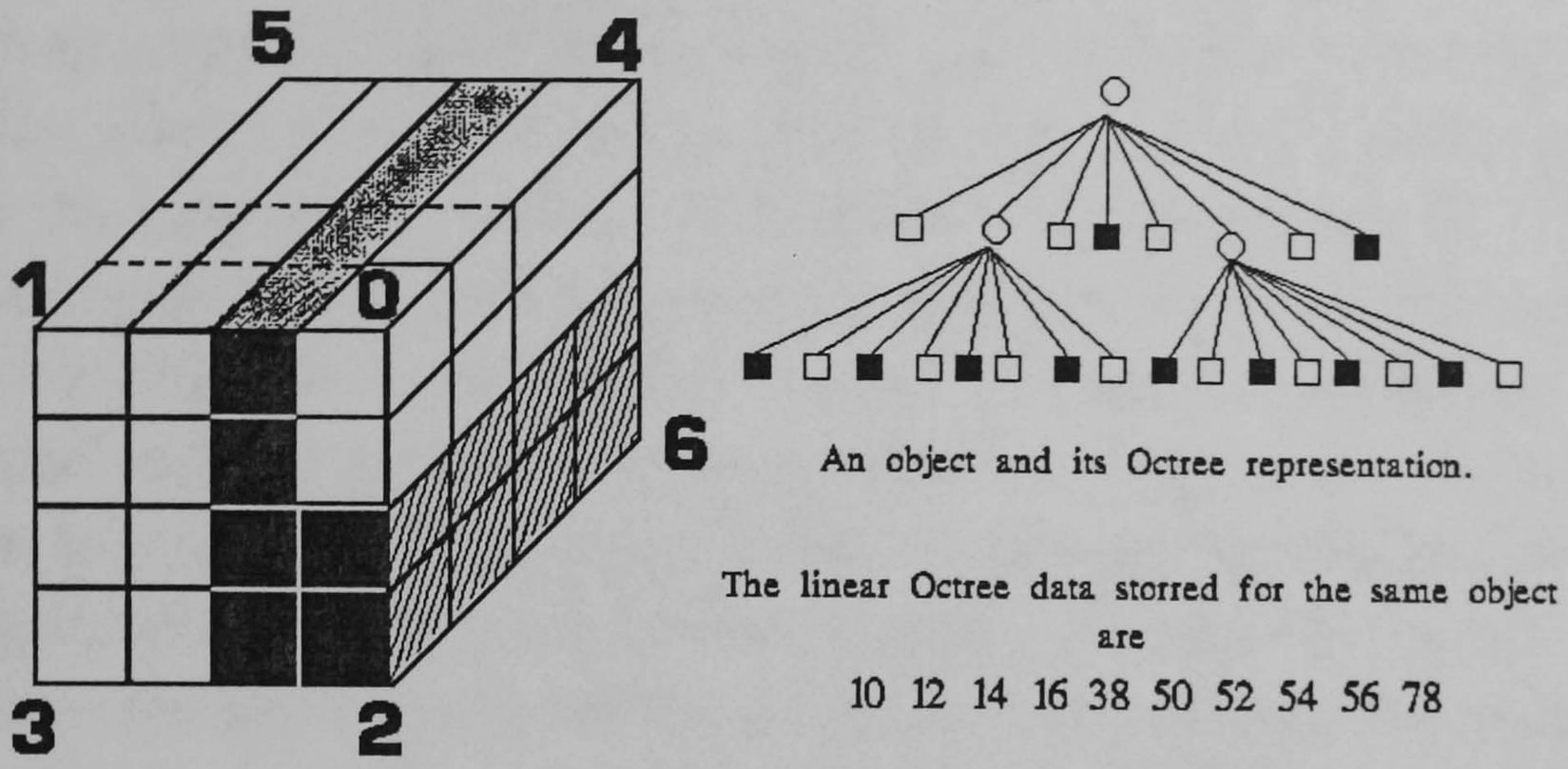


Figure 6.20. The linear octree representation.

6.4.3 Advantages and Limitations.

The information contained in the encoded octree representation of an object is identical to that available in the previously mentioned spatial enumeration representation. From a storage point of view, the data are stored in a hierarchical tree structure in which the nodes represent disjoint cubes of exponentially decreasing size. There are several additional advantages in the octree data structure. First, it uses a single primitive shape, the cube where an arbitrary object can be represented to the precision of the smallest cube. A single set of manipulation and analysis algorithms is required for all objects. No additional new techniques are required in order to handle more complex shapes. Operations such as hidden surface display show only linear growth because all objects are kept spatially pre-sorted at all times. When the octree is properly traversed, space regions are visited in a uniform direction in space. The trees representing the objects to be displayed are simply traversed in a specific order, depending on the view direction. Converting an object from an alternate representation format to the octree encoding format requires a spatial sorting. After this initial sorting, however, new objects generated

from them by boolean and geometric operations never need to be started again, even when the objects are moved or the viewpoint is changed. Boolean operations benefit from the octree structure in that the algorithms simply traverse the input trees in order while generating the output tree. The nodes at a level together with the higher nodes completely describe the entire object to the resolution of the specific level. Therefore, algorithms can be formulated that can operate at an appropriate level and avoid the bulk of the data that is contained at the lower levels. In such an application, objects can be represented at a high level (coarse resolution) in the main memory while the higher resolution detail resides on secondary storage until needed. This arrangement can be very efficient in interference detection algorithms. The calculation of object properties such as mass and centre of gravity is very efficient in the octree representations. Starting from the octree's root, a maximum and minimum mass can simply be calculated at each tree level. When the precision of the average of the two values is within the tolerance required by the particular application, the operation is terminated with often only a small fraction of the node values having been accessed. An additional advantage lies in the ability to perform partial calculations that are then passed to the next lower level. In this respect the calculations are similar to those required for a Fast Fourier Transform. Substantial reductions in computation can result. None of the union, translation, scaling, rotation and numerous display algorithms developed to date, require floating point operations, integer multiplications or divisions. Thus the algorithms can be implemented in relatively inexpensive hardware. The processing of each node generates 0 to 8 independent sub-calculations. Therefore, a system can be designed in that operations are performed in parallel.

Nevertheless, the octree data structure suffers from several limitations. The bounding surface of an object is represented by the set of square facets between the empty and full cells and is therefore an approximation of the original surface by square polygons. For objects with complex detail, the octree requires a large number of cells to represent the object accurately.

Precise detail, such as surface curvature can often be lost in an octree representation. The main disadvantage of the octree encoding technique is its large memory requirement. Several million bytes of node storage may be necessary to represent real objects. In the current application the combined use of pointerless octree data structures where only full nodes are encoded, and a substantial reduction from the original 512 by 512 pixel resolution to a more moderate 64 by 64 pixels, has resulted in general significantly improved storage requirements (reductions on the order of several tens of thousands of pixels per image). In addition, the reduction in the original pixel resolution to 64 by 64 pixels has created a sharply improved object boundary representation that has greatly contributed to the overall successful pattern recognition procedure. A detailed analysis of the use of the pointerless octree data structure can be found in sub-section 6 of this chapter.

6.5 The Volume Intersection Technique.

6.5.1 Introduction.

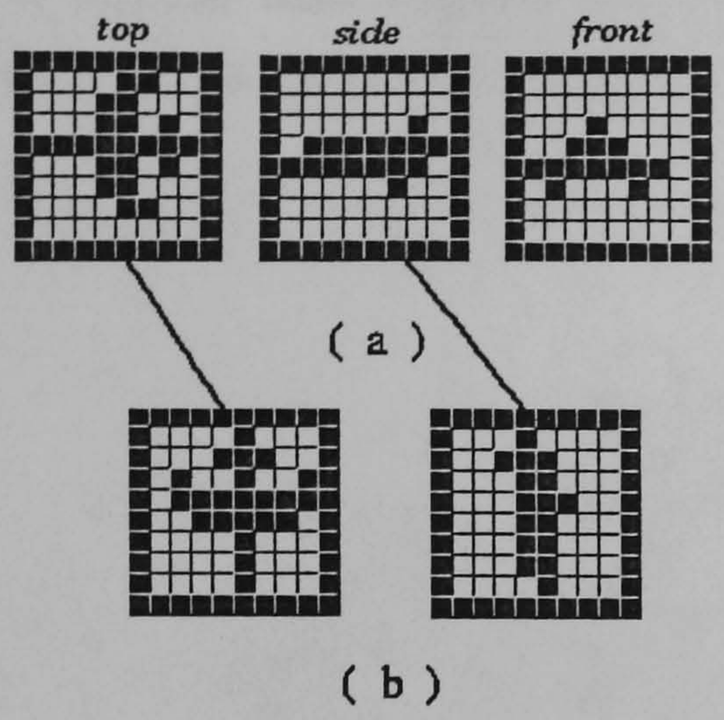
The problem of generating octrees from a number of images was first addressed by Conolly [192]. He proposed a technique to generate the octree of an object from an arbitrary view. This octree structure could be cumulatively updated by subsequent object views. In his approach the data for each view were initially converted into a quadtree. Since the coordinate axes associated with each quadtree were not necessarily aligned with those of the octree, the quadtree needed to be transformed into the octree coordinate system. However, the author failed to point out the complexity involved in the transformation and did not provide enough justification for the conversion of input data to a quadtree.

Generating octrees from quadtrees has been a very popular approach. It was first proposed by Yau and Shihari [190] who generated the octree of an object from the quadtrees of its cross sections. Later, Chien and Aggarwal

[193] developed a technique to generate the octree of an object from the quadtrees of its three non-coplanar views, in which the "generalised" coordinate system of the octree is specified by the three views. More recently, Veestra and Ahuja [194] have developed a technique to construct the octree of an object from its 'face', 'edge', and 'corner' views. In all the previously mentioned approaches, the coordinate axes of the quadtrees are aligned with those of the octree. This criterion is not satisfied in Conolly's work, where the coordinate axes of the octree are fixed, and available views may be arbitrary. While the approach reported by Conolly is generally applicable it does not take the advantage of the controllability in model construction in order to simplify the octree generation process. In model construction, it is often accepted that a certain amount of control over the sensing configuration exists.

6.5.2 The general octree method description.

In a recent interesting paper by Chien and Aggarwal the idea of volume and surface octrees from silhouettes of multiple views is introduced [195]. Their method can be classified into three main stages. First, obtain three images of the object using no coplanar views. Normalise each image using the method described in previous parts of this chapter, and create the respective normalised quadtrees. Second, use the normalised quadtrees to generate three pseudo-normalised octrees. Finally, use the Volume Intersection technique [196] (i.e., merge the three pseudo-volumes) and create the final normalised octree structure. Based on the above general algorithmic presentation, a more specific methodology has been formulated for the precise implementation of the above procedure regarding each individual view. In this chapter the top, side and front views of the object were only used for convenience. In order to create the normalised octree structure, the processing method followed for each particular view is described below in details. Figure 6.21(a) shows the original three views of a simulated 8 by 8 pixel aircraft model.



a. The original three 2D views of the first aircraft.

b. The transformed 2D images of the top and side pictures.

Figure 6.21. The set of three original aircraft views and their preliminary pre-processing.

6.5.3 The front view image.

The processing starts with a 2-D image that represents the front view of the object. No preliminary processing is required prior to the application of the quadtree encoding algorithm [Figure 6.21(a)]. Assuming that the image is normalised, the method proceeds to create the normalised quadtree representation of the image. The quadtree structure is created by starting at the lower left quadrant and continues at the lower right, upper left and upper right, respectively. In order to create the octree structure that corresponds to the front view volume of the object, the next two rules were followed. First, start from the root of the quadtree and reverse all subsequent quadtree nodes and leaves (e.g., consecutive node leaves a,b,c,d of the quadtree will become d,c,b,a on the octree). Second, begin from the left most sub-quadtree and moving to the right, for every quadtree node copy all the four node's children so as to create the eight necessary octree branches (e.g., d,c,b,a leaves of a quadtree node will then becomes d,c,b,a,d,c,b,a in the final octree node). Figure 6.22 shows the three dimensional volume created by the front view image, while Figure 6.23 shows the respective quadtree data structure of the

front view image and Figure 6.24 displays the algorithmic conversion from the quadtree data structure to the respective three dimensional octree data structure.

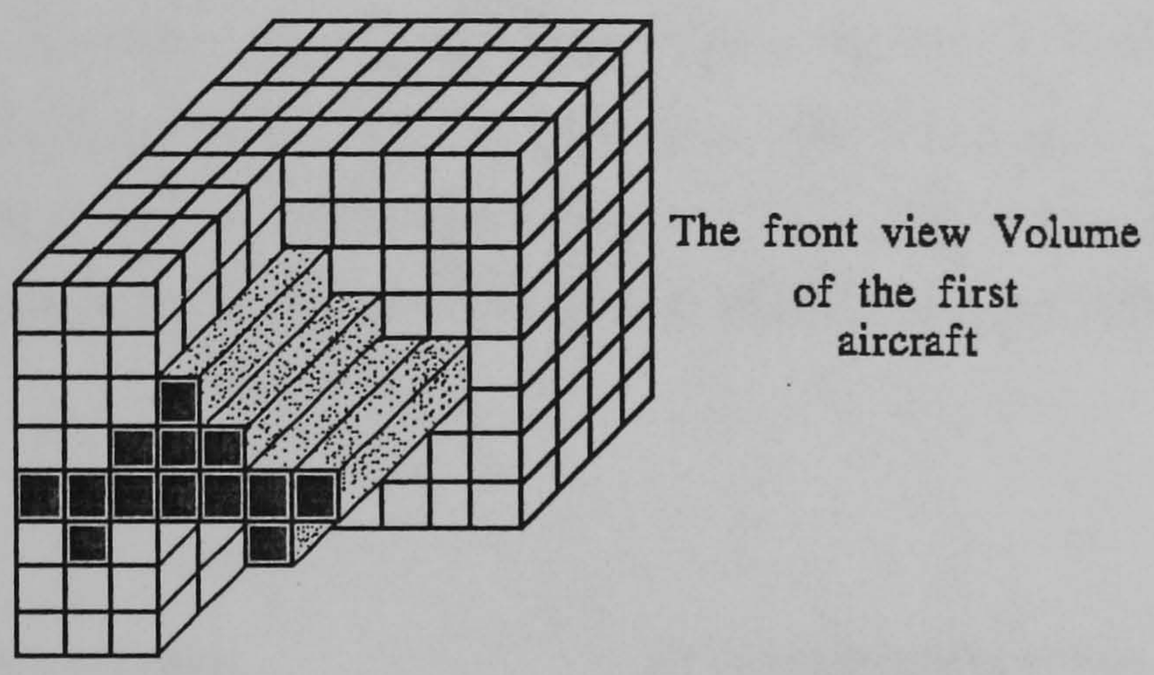


Figure 6.22. The front view volume of the first aircraft.

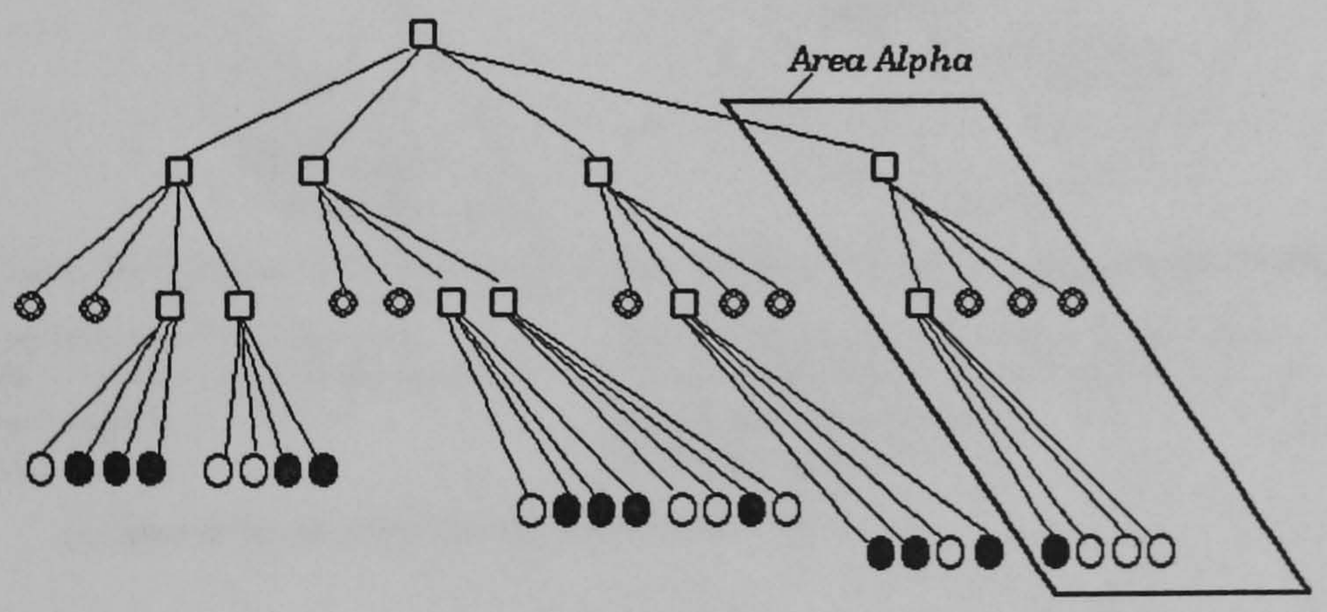


Figure 6.23. The quadtree of the front view image.

6.5.4 The top view image.

A preliminary processing step is required in order to create a new 2-D image by rotating the original 2-D top view image as described in Figure 6.21(b). The next step involves generating the quadtree structure of the new top view image. In order to create the octree structure each two consecutive nodes on the quadtree are taken, and are then copied into the octree as described in Figure 6.26. Figure 6.25(a) shows the volume of the top view image, while Figure 6.27 shows the corresponding quadtree of the top view picture.

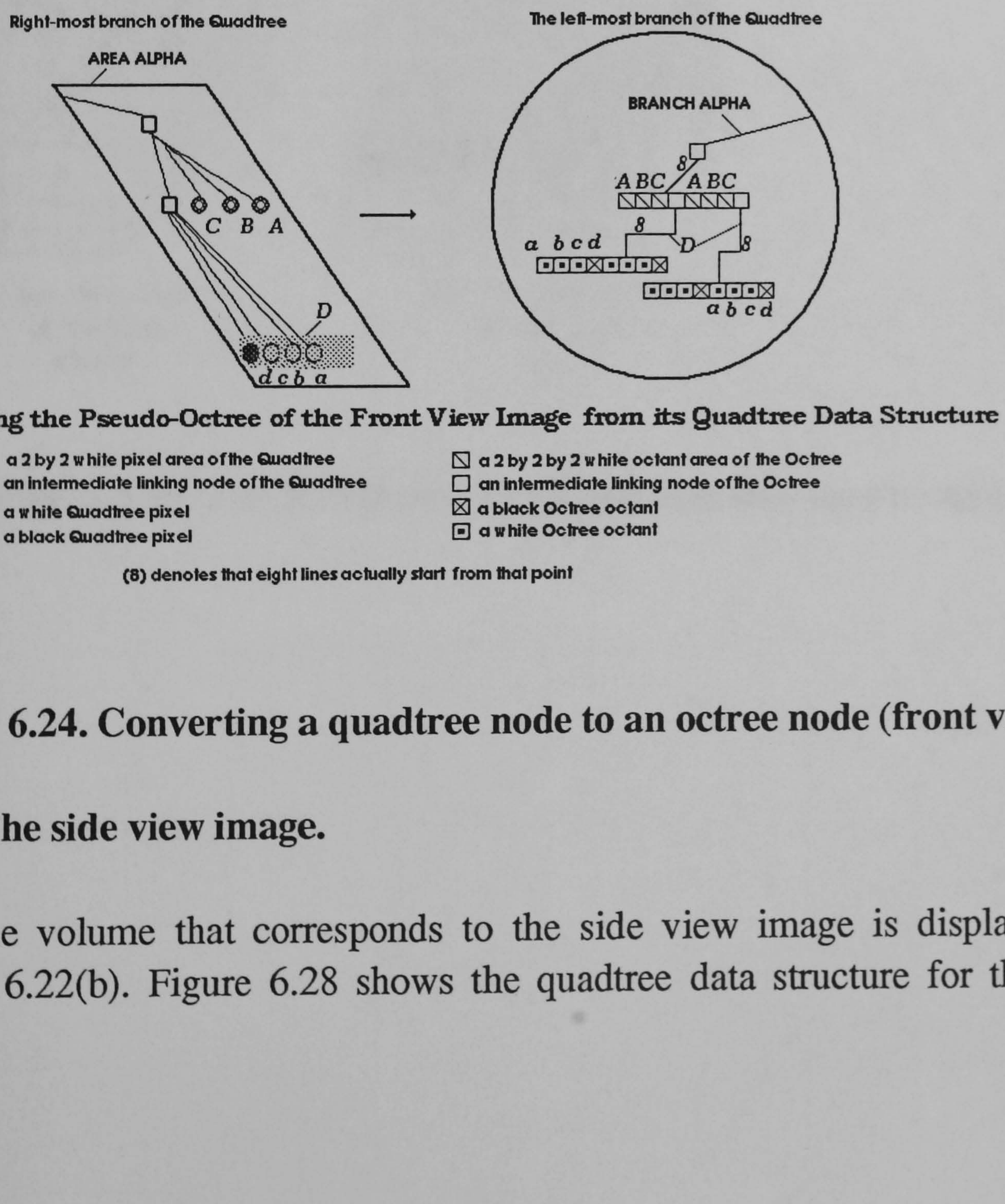
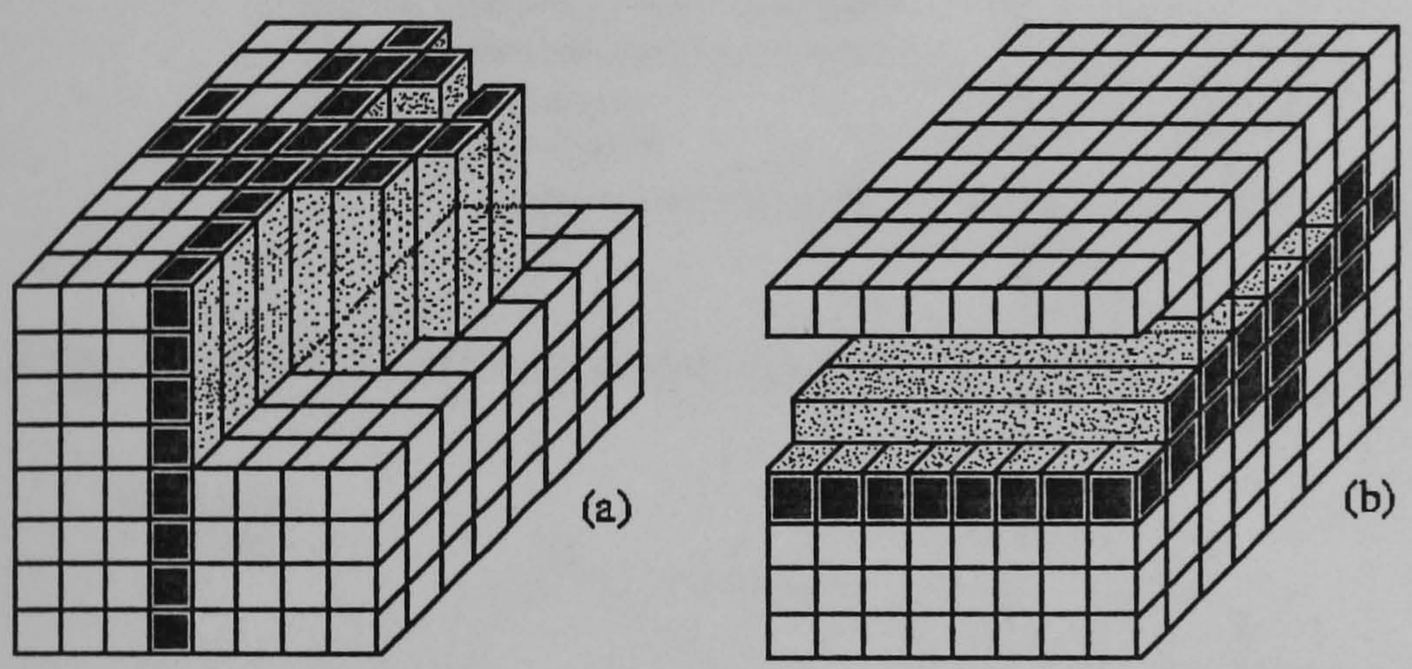


Figure 6.24. Converting a quadtree node to an octree node (front view).

6.5.5 The side view image.

The volume that corresponds to the side view image is displayed in Figure 6.22(b). Figure 6.28 shows the quadtree data structure for the side

view image while Figure 6.29 demonstrates the algorithm for converting a quadtree node into an octree node regarding the side view picture case. It is important to notice that each node in the octree is produced when each of the main four quadrants of the corresponding quadtree is copied on the double number octant and also on the immediate successive octant (e.g., quadrant 2 is copied on octant four (4=2x2) and also on octant five [5=(2x2)+1]).



The top view Volume of the first aircraft.

The side view Volume of the first aircraft.

Figure 6.25. The 3D volumes of the top and side view images.

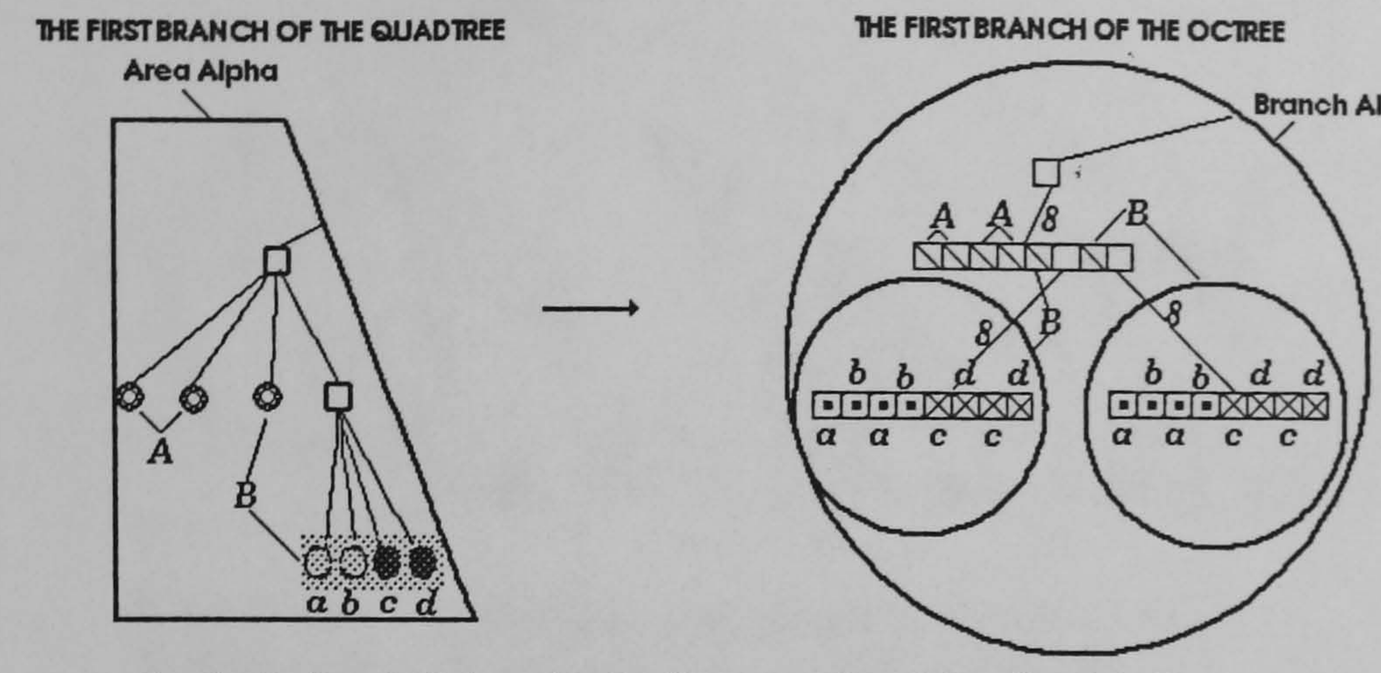


Figure 6.26. Converting a quadtree node to an octree node (top view).

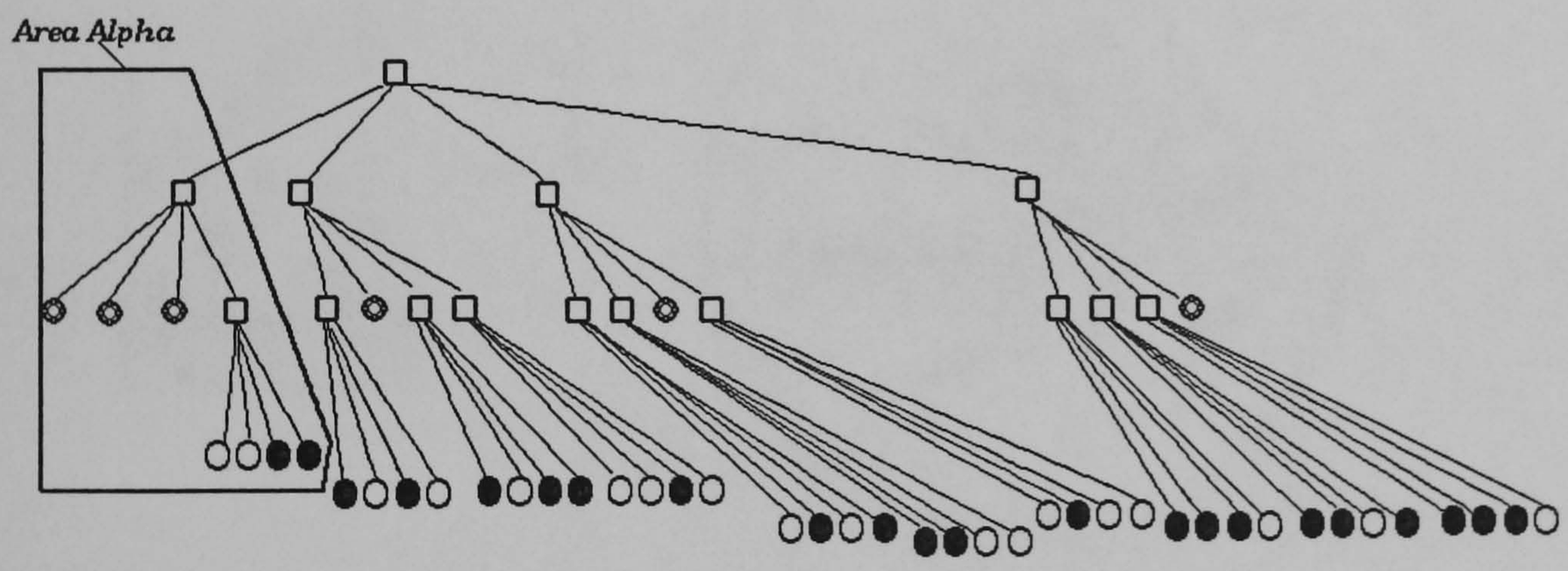


Figure 6.27. The quadtree of the top view picture.

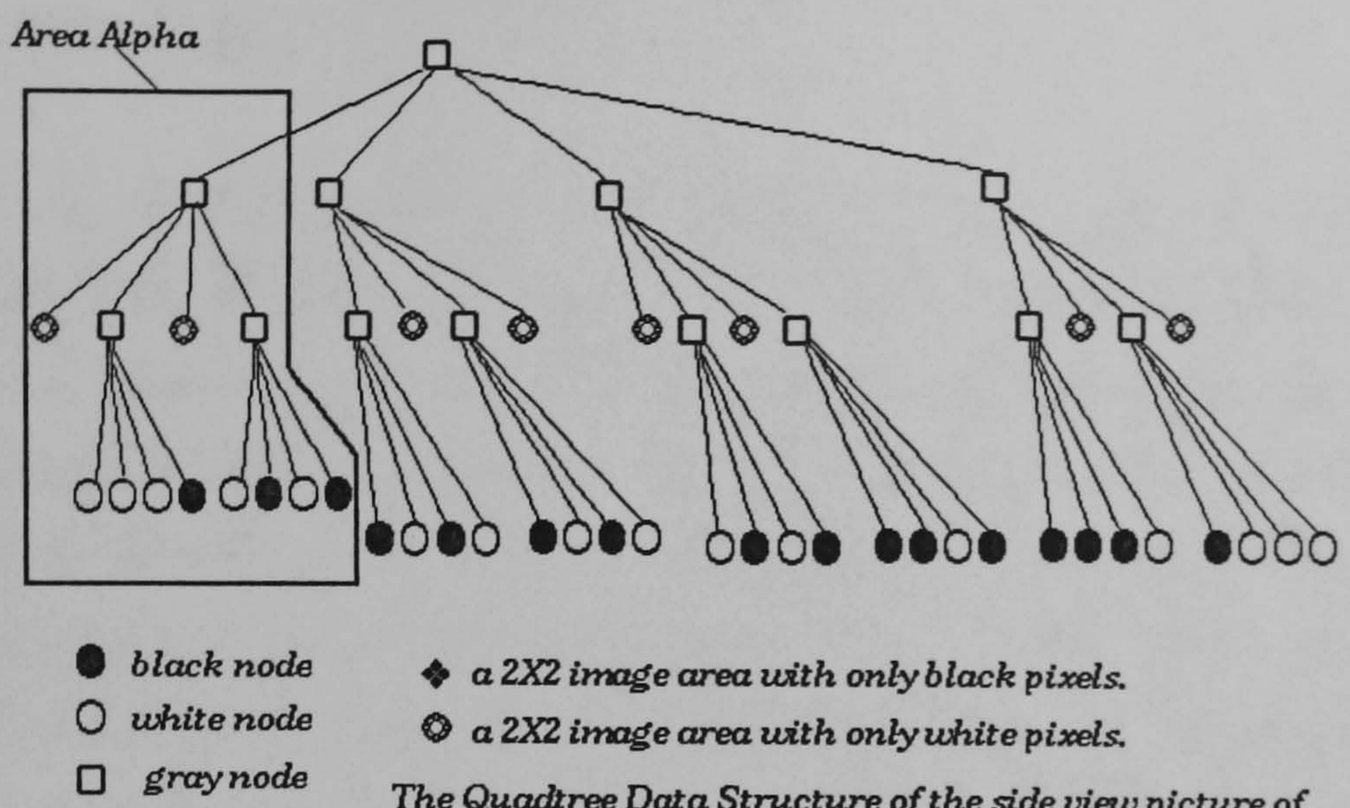


Figure 6.28. The quadtree of the side view image.

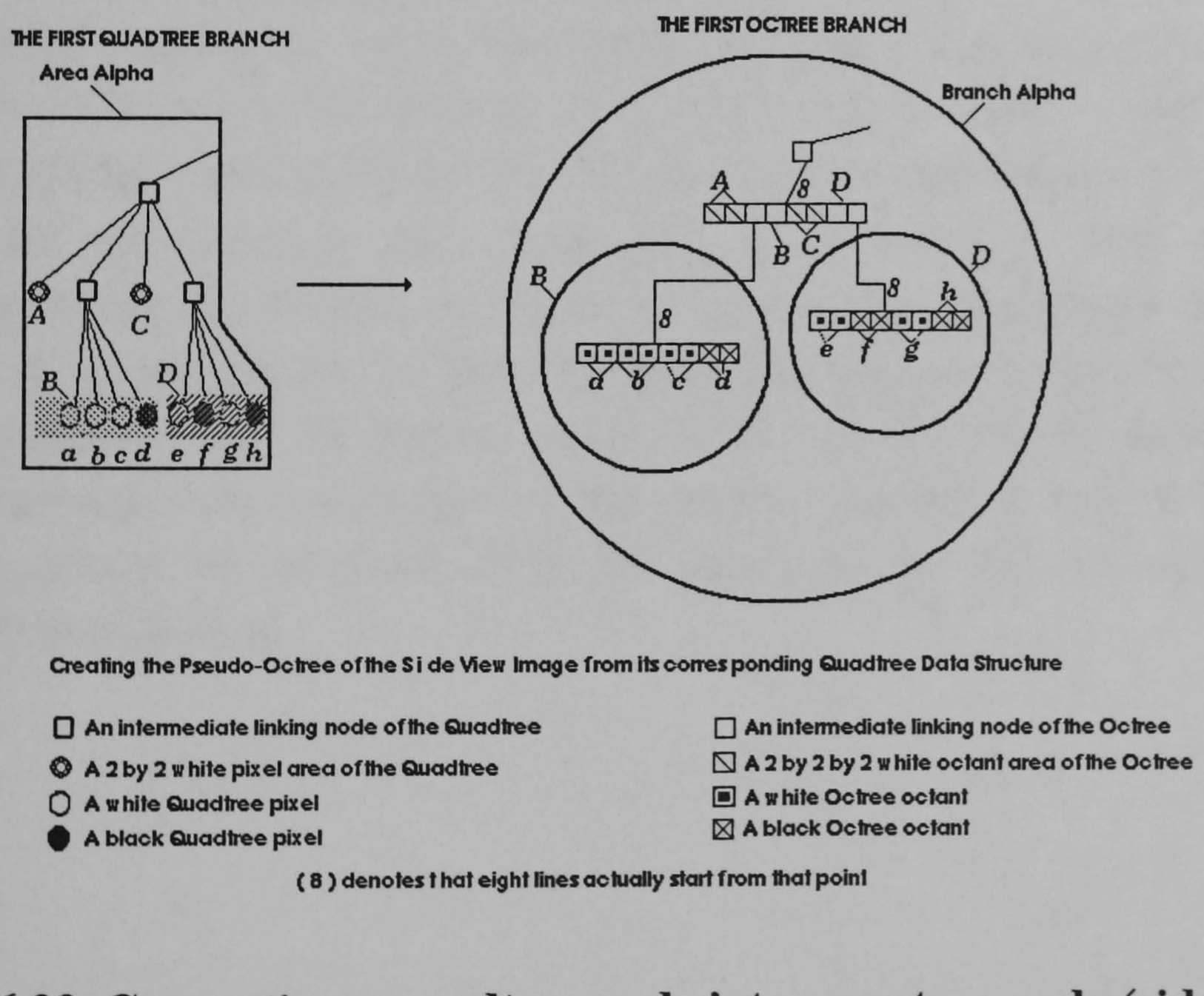


Figure 6.29. Converting a quadtree node into an octree node (side view).

6.5.6 The final octree.

In order to generate the final octree structure that will describe the volume of the 3-D object of Figure 6.30, the technique known as Volume Intersection is applied [Figure 6.31]. Each one of the three normalised pseudo-octrees is merged with the others and the final node is full if all the nodes are full otherwise it is empty. Merging also involves the GRAY nodes. In the end, all nodes to be merged have to be in the same tree level and be either of full or empty type, thus nodes may have to be expanded with the addition of more branches and further nodes in the octree in order to obtain the same tree level for merging. In the example, the resulting normalised octree is shown in Figure 6.32 while the raster image used for training the ADAM neural network [106] is shown in Figure 6.33. This synthetic image results from a simple in - order traversal of the final octree data structure. The octree is traversed at its highest level (leaves only) using a left to right direction. If the octree's highest level is less than its maximum size (i.e., for a 64 by 64 pixel image, the octree size is 6 [2^6=64]), then necessary adjustments are performed in order to create the appropriate additional number of octree levels for the traversing mechanism. These involve the simple calculation of the overall object areas with identically valued pixels and consequently the estimation of the precise number of octants within it. Clearly, octrees of the same level are necessary for the intented volume intersection technique.

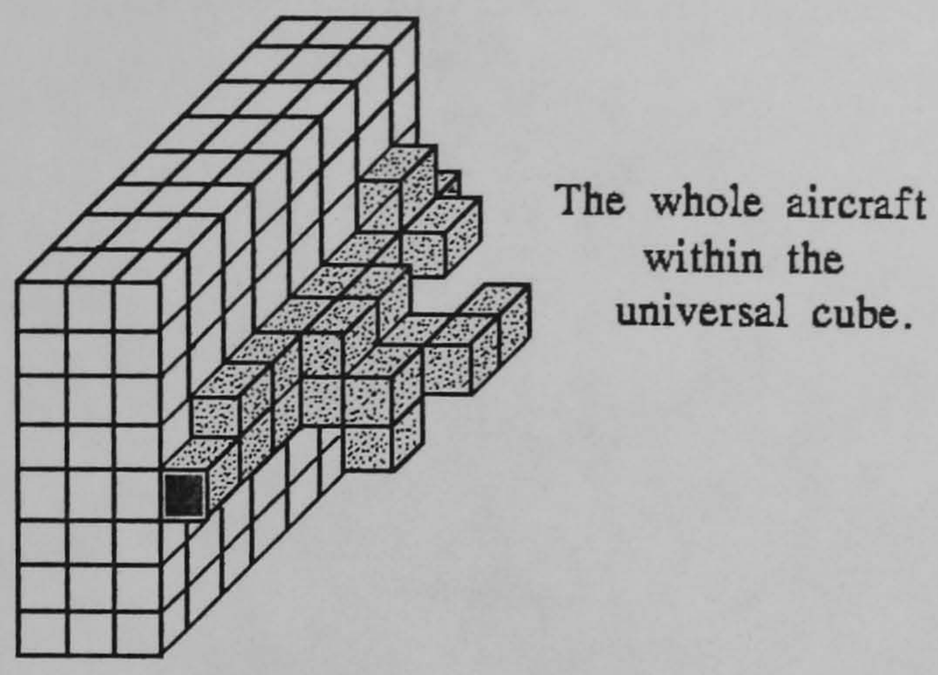


Figure 6.30. The final volume of a 3-D aircraft.

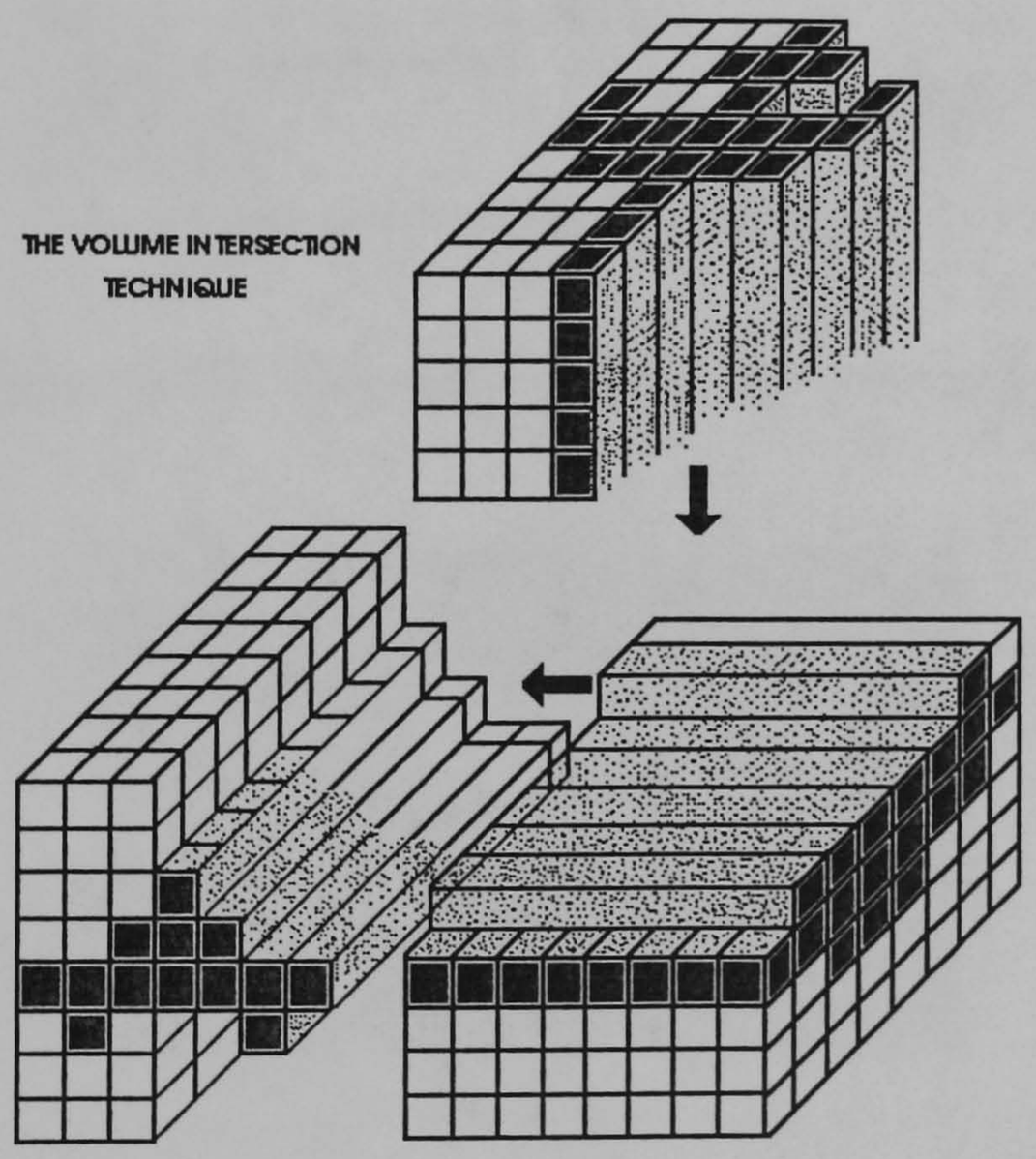


Figure 6.31. The Volume Intersection Technique.

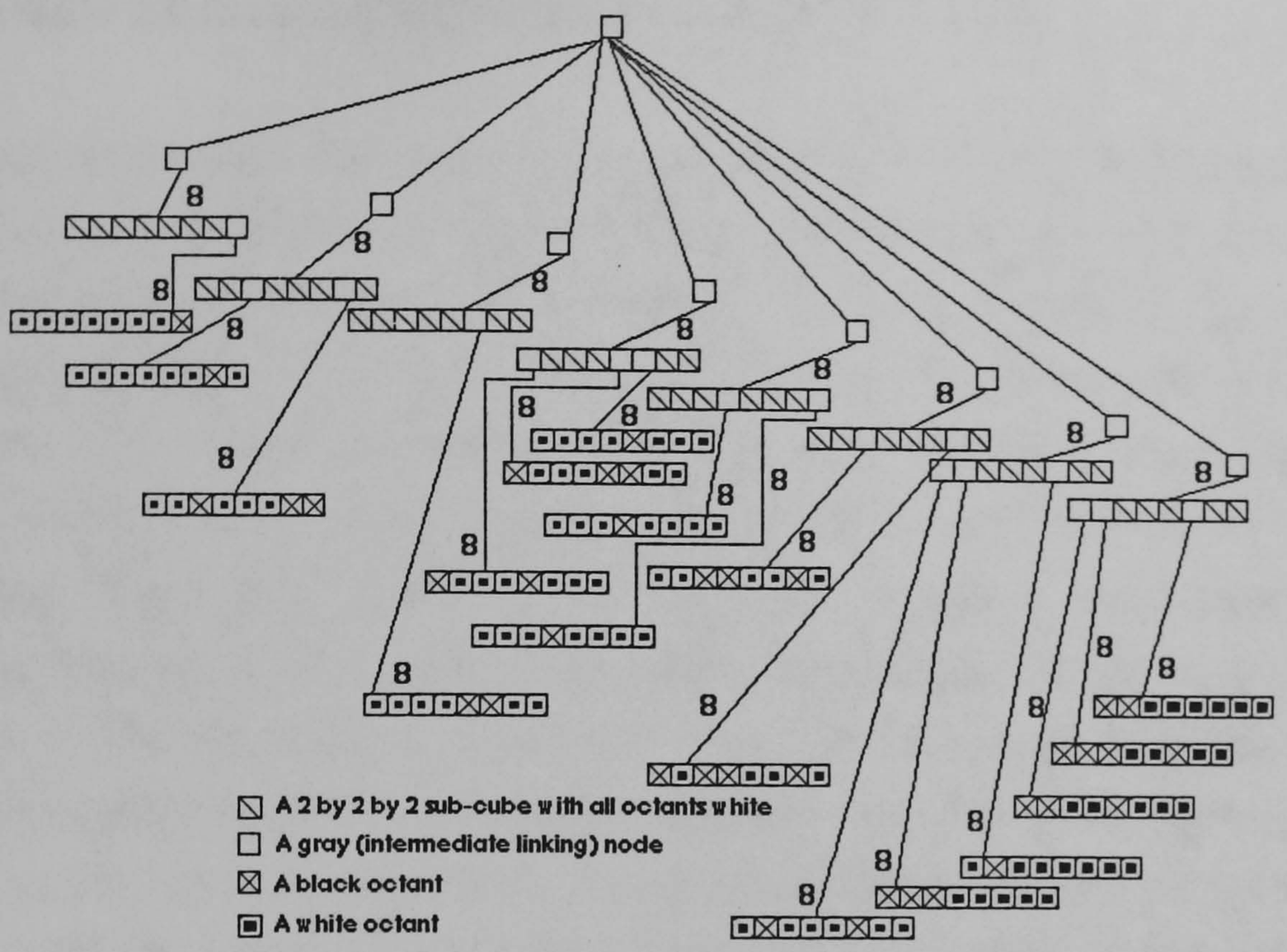


Figure 6.32. The final octree data structure.

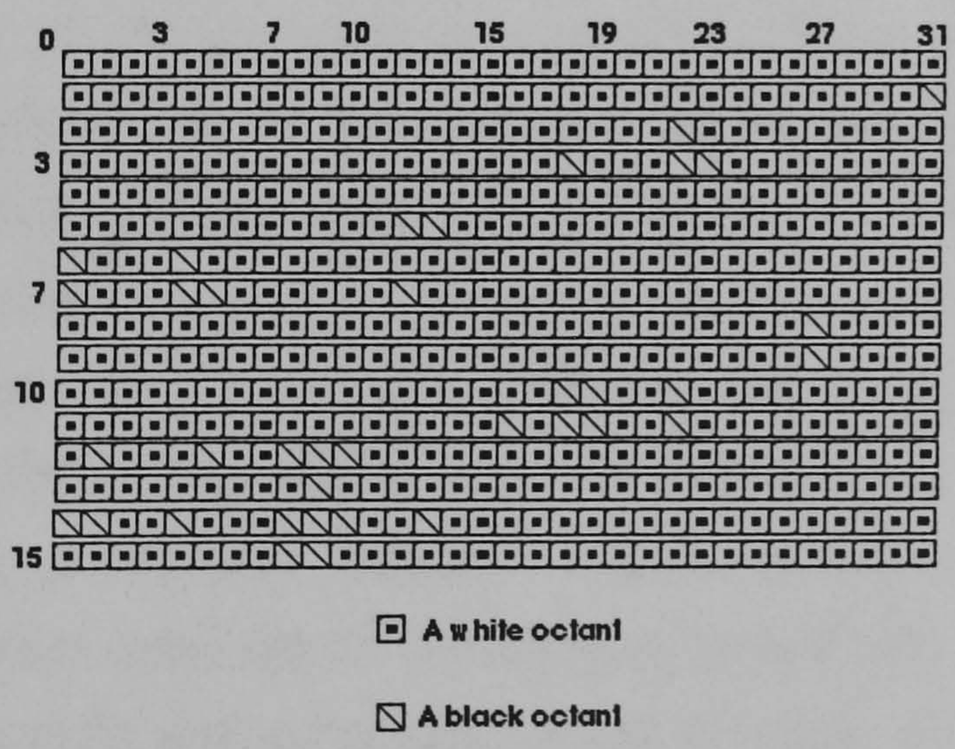


Figure 6.33. The synthetic image used for training ADAM.

6.6 Storage and computational considerations.

As has been already mentioned, the most significant advantage obtained from the use of quadtrees is their generally high image compression rate. In the present application, six types of aircraft edge detected images have been used [Figure 6.34]. The original images obtained by the camera system were of 512 by 512 pixel resolution. Clearly, the quadtree encoding of such pictures would create rather large data structures that would be difficult to manipulate. Typically, for a 64 by 64 pixel image a total number of 64^3 octants is obtained. The latter fact, and the obvious disadvantages of high resolution in binary images, where for example breaks in an object edge map might drastically modify a generally sensitive pattern recognition procedure, have led to the decision to reduce the original resolution of the pattern set. In practise, a 64 by 64 pixel resolution has been used, that allowed a maximum depth of order 6 [2^6=64] in the quadtrees and produced better 'closed' contours of objects. Having in mind that the method used linear quadtrees/octrees data structures, where only full nodes need to be coded, the average percentage of image area covered by black pixels had been in the order of 25% [Figure 6.35]. This resulted in an immediate average storage saving of 75%, which finally using the properties of homogeneous areas in quadtrees was increased to 78%. Needing to store only 22% of the original information in order to efficiently encode an object might not seem a significant compression ratio in the case of two dimensional image data. The fact though that the current application deals with three dimensional objects implies that the current amount of data to be stored are in fact 64 by 64 by 64 pixels and clearly storing information for an average of only 1000 pixels (i.e., 25% of 64 by 64) per object represents a major storage saving on the order of 99.99% (1000 pixels out of 64^3). A number of factors have led to this high compression ratio. First, edge detected aircraft pictures contain a large number of black areas that did not need to be stored. In a typical thesis aircraft picture the ratio of black to white pixels is normally 3 to 1. That

indicates a minimum compression rate of 75% as only 25% of the entire picture needs to be stored for an error-free reconstruction of the original image. Second, the use of pointerless region descriptors such as the quadtree data structure, did not provide any additional storage requirements for pointers within the quadtree, that could substantially reduce the overall achieved compression rate. Finally, the volume intersection method requirement for merging three different object views in order to create the object's volume representation, increases the final compression rate as all three cells at a specific space position need to be white for the final object cell to be white. Anything less than three white cells would create a final black cell and consequently would increase the compression rate because the new cell resulting from merging would not need to be stored.

The computational considerations of the previously described technique can be classified into two main groups: the overall computational time for the image pre-processing techniques and the computational requirements of volume intersection. The first group involves the time required for reducing the original image resolution to 64 by 64 pixels, the application of the Maxgrad edge detector, the enhancement of the edge map and the final thresholding operation. Figure 6.36(a) shows the average times required by the first group, while Figure 6.36(b) displays the time required by the Volume Intersection technique. In the second group, the times obtained in the experiments matched the times presented in [189] [174], and were on the order of 1.8 seconds for the quadtrees and 2.1 seconds for the octrees. As the Volume Intersection technique does not require all quadtrees from the beginning, the process can be implemented in parallel, as one after the other all three required images are obtained. Therefore, the maximum time recorded in our experiments has been 3.9 seconds for the entire Volume Intersection method. All the experiments were timed on a Sun 3/50 workstation, and the total time for the whole pre-processing operations totaled 25.8 seconds. Relative time duration for the important computationally expensive edge detection operation on a Sun 3/60 showed

overall edge detection times down by 49 to 57% at around 9.8 seconds and consecutively total pre-processing time requirements of 12.7 seconds.

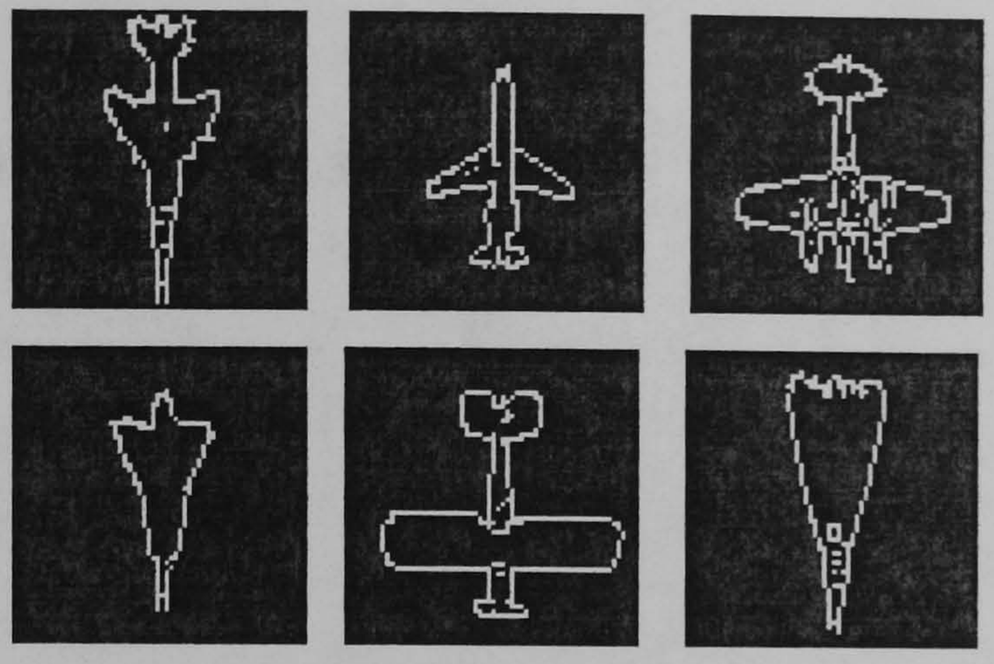


Figure 6.34. The set of six 64 by 64 pixel edge detected images.

6.7 Conclusions.

A new algorithm has been presented for implementing the Volume Intersection technique. The algorithm aimed to create a training platform for teaching artificial neural networks whereby any object could be taught in a single cycle and with the least number of different picture views possible. The new algorithm was applied on binary edge detected images that were created after pre-processing the original raster picture data using several image processing methods. It then continued by creating the normalised quadtree data structures of three non-coplanar object pictures. Finally, the three normalised quadtrees obtained in the previous phase were merged to create a normalised octree that represented a view of the object that is independent of position, rotation and relative scale changes. In the end, by a simple in order traversal of the octree a synthetic binary image can be created that can by itself be used to train artificial neural networks. The method had offered substantial storage savings of up to 99.9% and required on average a moderate 12.7 seconds for coding each one of six different types of aircrafts.

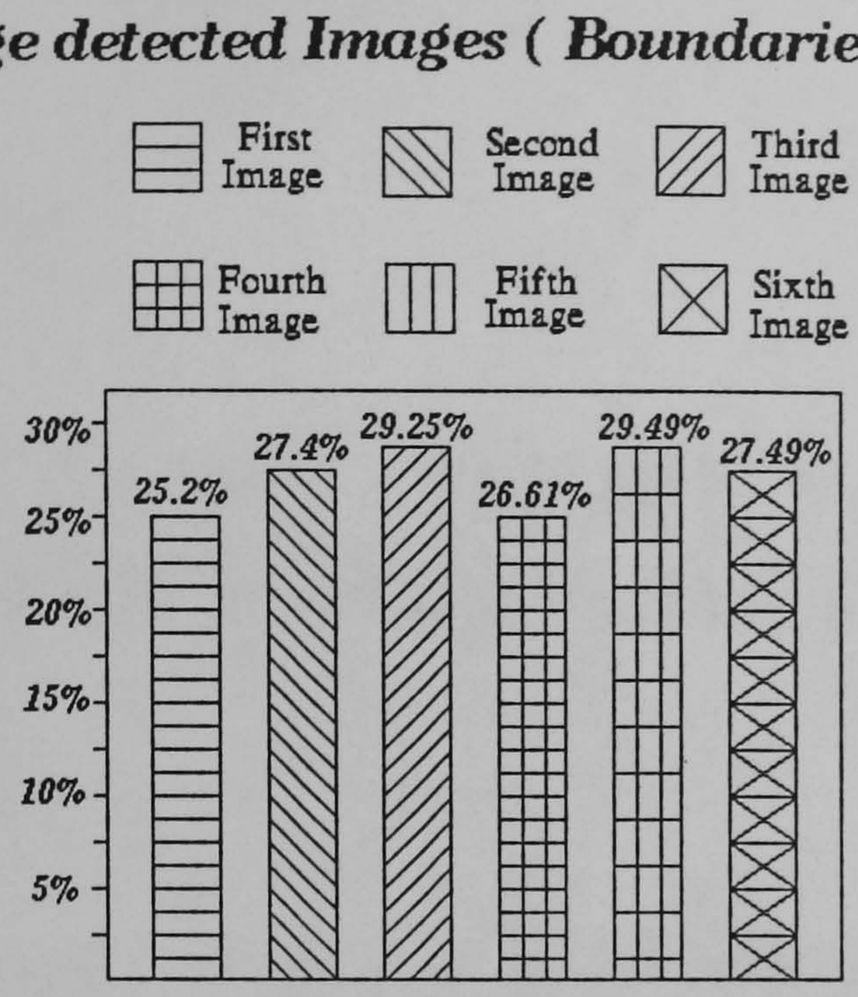


Figure 6.35. The average picture cover by black pixels.

Image Pre-processing Algorithms	Sun 3/50	Sun 3/60	
Reduction of the Original Resolution	1.2 s	0.6 s	(a)
Edge Detection	20 s	9.8 s	
Edge Map Enhancement	0.4 s	0.3 s	
Histogram Thresholding	0.3 s	0.25s	
Quadtree Encoding	1.8 s	0.7 s	(b)
Octree Encoding	2.1 s	1.05s	

Figure 6.36. The overall pre-processing time requirements.

CHAPTER 7

Experimental Results Obtained from Training and Testing the ADAM Artificial Neural System.

7.1 Introduction.

In the previous chapter the method used for training and testing the ADAM artificial neural system has been introduced. A presented experiment using an 8 by 8 pixel image has been shown and the entire process from its initial image pre-processing stages to the final synthetic object volume picture creation has been described. In this chapter, a number of experimental performance data will be presented, aiming to prove that on the one hand ADAM can successfully generalise and on the other hand to show, using the resulting classification success rates, that the proposed technique is effective in its implementation and its general behaviour is consistent and highly accurate.

The presentation of results is structured in four main categories. In the first, an example involving random objects is presented. Three 4 by 4 by 4 octant objects are used and the precise generation of all pseudo-quadtrees is presented for a single chosen candidate. The merging volume creation is once again displayed and the final pattern used to train the ADAM neural system is shown. The aim is to provide a simple and easy to follow example, that can be studied in detail in all its aspects. Finally, performance data are presented reflecting ADAM's response when only three original images have been used to train the neural system and a set of thirty six patterns have been applied for testing its classification abilities. ADAM is shown to have no problem in successfully classifying any of the presented patterns, in either noise-free or noise-corrupted testing cases.

The second category presents an experiment involving actual aircraft data but in a rather small resolution to allow once again an easy follow through of the entire procedure. Three 8 by 8 by 8 octant simulated aircraft pictures are presented for training ADAM and the neural system is again tested successfully on a thirty six pattern set. The third category presents the actual experiments with real aircraft model data. The experiments involve two types of aircraft patterns: simple aircraft skeletons, and aircraft filled boundaries. Six different aircraft models have been used and ten different set of results from equivalent number of experiments are reported. In particular, two set of experimental results are shown that involve testing ADAM with entirely unknown object volumes obtained form completely random object views. The results in that last type of experiments verify the successful behaviour of ADAM when it was trained with various data from different object views. Finally, the last category shows a possible training scheme for ADAM in the case where object volumes cannot be obtained. In this case, ADAM is trained with the synthetic images of the intermediate level normalised quadtree data structures. The method is shown to have a potential successful application in a variety of orientations but it is shown that it can not be applied in the general case.

7.2 An example of random 3D objects.

7.2.1 Introduction.

The verification of the entire procedure presented in the previous chapter involves the individual checking of the quadtree and pseudo-octree data structures in each one of the three required non-coplanar object views. Here, a general example involving small size 4 by 4 by 4 octant objects is presented. The aim is to show the performance of ADAM neural network in classifying the three objects. For testing, an array of six pictures is used for each individual object. The testing array is structured as follows. The first picture represents an actual object volume image with 5% localised noise

(i.e., only object structural octants are changed from white to black). In the same way the second picture has 10% localised noise. The remaining four pictures in the testing array for the first object contain actual object volume data corrupted with various levels of random added noise. Noise levels of 5%, 10%, 15% and 20% are used. Figure 7.1 shows the three different objects to be used for training the artificial neural system.

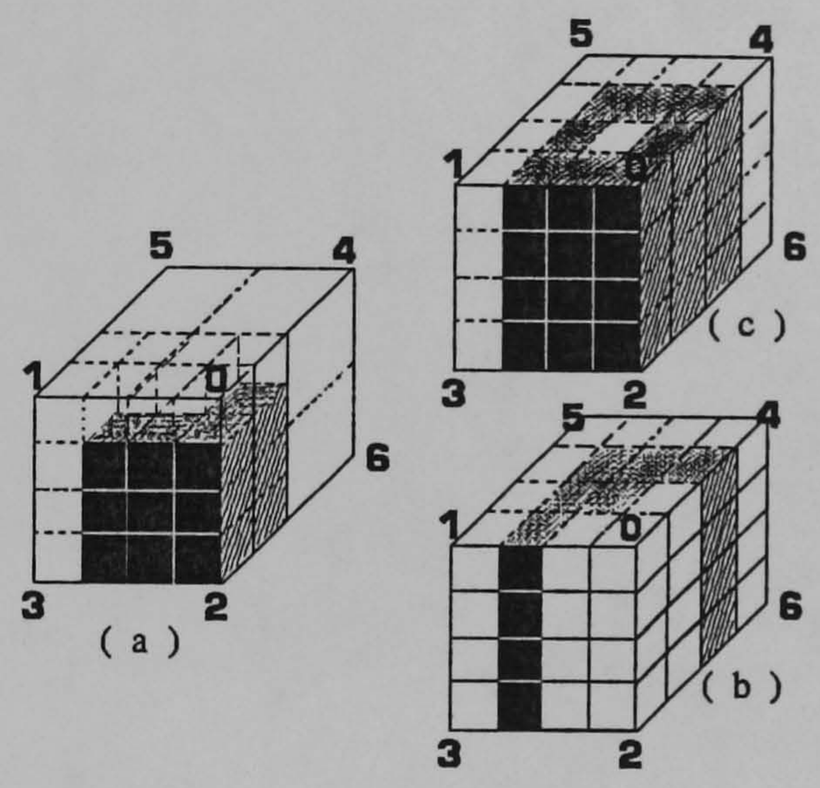
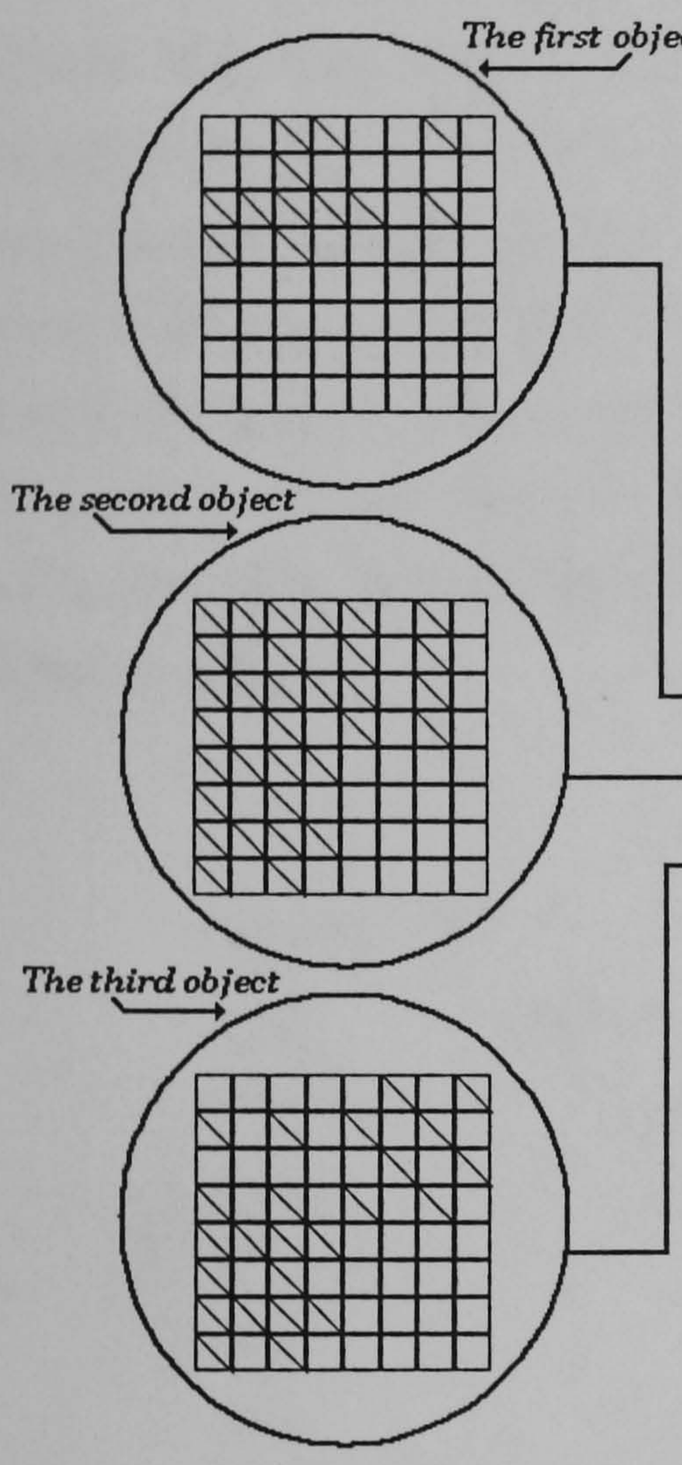


Figure 7.1. The three objects used in the first experiment.

7.2.2 Training ADAM: A performance evaluation.

Figure 7.2 below shows the three resulted synthetic object volume images that were used to train and test the artificial neural system. The individual performance for each one of the images used for training ADAM is clearly displayed in the performance graphs below. Each testing set of patterns for every image contained six pictures. The first two pictures used, had no random added noise but were rather corrupted with localised noise of 5% and 10%. The image degradation in the previous cases had the form of aircraft skeleton bits being removed (i.e., aircraft's wing been cut). The remaining four pictures in the six images testing set were aircraft pictures with 5%, 10%, 15% and 20% random added noise. The images used in this

first test case in order to train ADAM neural network were 8 by 8 pixel in size and were created after traversing the final pseudo - octrees in the way previously described in chapter 6.



Neural Network trained with just three object pictures and tested with a set of 36 images. Each horizontal pattern test array involves two images with 5% and 10% localised added noise and four other pictures with 5%, 10%, 15% and 20% random added noise.

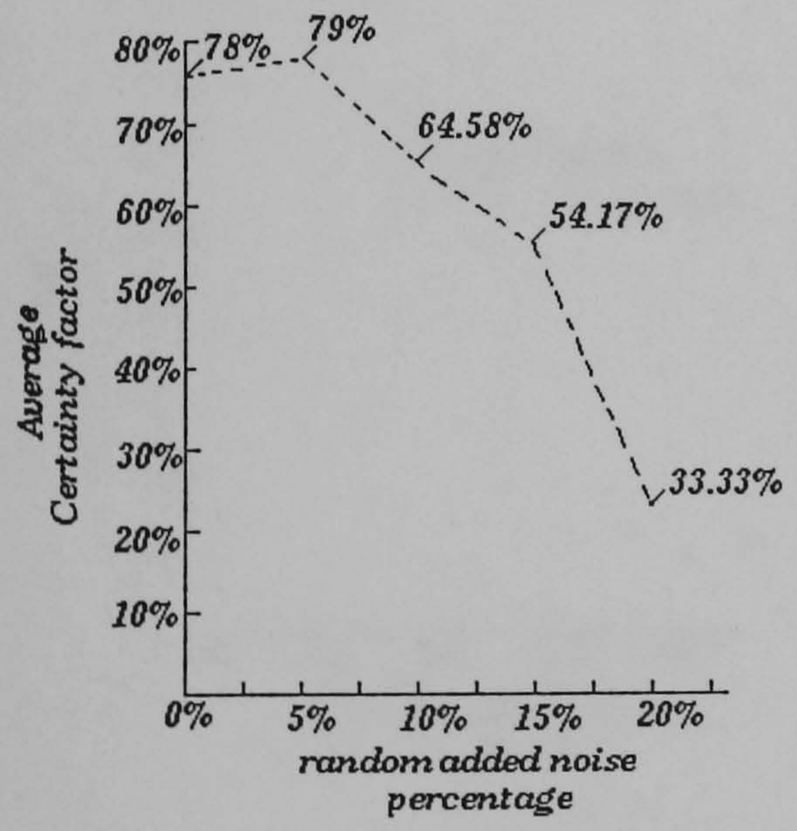
No misclassification has been occurred

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
First object	81.25	75.00	79.17	64.58	54.17	33.33
Second object	83.33	77.08	78.00	56.25	56.25	52.00
Third object	89.58	70.83	75.00	81.25	62.50	37.50
	Random Noise Levels		*5%*	*10%*	*15%*	*20%*

Figure 7.2. The 4 by 4 by 4 octant training picture set and performance data.

Figure 7.3 presents the classification results when ADAM was trained only with the first six picture test set. There has been no misclassifications and the classification success had steadily decreased as the neural system was tested with images that contained higher level of random added noise. The small upturn movement observed when ADAM was trained with 5% random

added noise can be justified when the parameters that define the first performance classification success are more closely described. As it has already been mentioned, the first performance data corresponded to the average of two testing patterns that although they contained no random added noise they did contain a form of localised noise of 5% and 10% respectively. From the first row in Figure 7.2 it can be seen that the neural network responded better in the 5% localised noise case than in the 10% localised noise case. Furthermore, the performance in the 10% localised noise case was worse than in the 5% random added noise case. Clearly, as the rate of noise increases the neural network's performance worsens. Since an average is used to determine the final value of the first index in the graph, the lower second value that reflects the 10% localised noise case, directly influences the final outcome.



Neural Network trained with three object images and tested with an array of six pictures.

The first index at 0% random added noise represents the average of two pictures that were corrupted with 5% and 10% localised noise, while the four remaining pictures in the test array were images with 5%, 10% 15% and 20% random added noise.

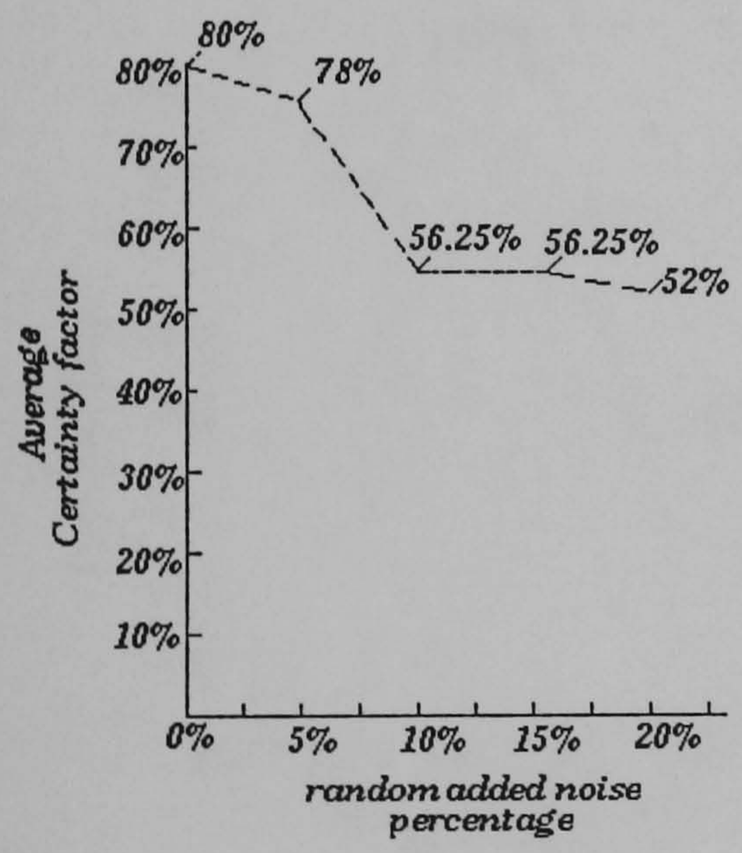
No misclassifications have occurred

Figure 7.3. Performance of the first testing set.

Being twice as noisy as the next immediate noise index level in the graph, the 10% localised noise index value understandably represents a less successful performance than the one represented by the 5% random added noise index

and it is the main reason for the apparent discontinuity of the first graph. Similarly the result of the 5% localised noise case being not significantly higher than the 80% mark does not help to increase the final averaged result.

The second testing set of another six images shows a rather unambiguous behaviour in comparison to the performance displayed by Figure 7.3. Figure 7.4 presents the performance graph of the second test array of six new pictures. Clearly, the classification success decreases as the level of random added noise increases. An interesting observation in that second graph is the considerably higher confidence in the 20% random added noise case and the flat performance of the neural network classifier in both the 10% and the 15% random added noise phases of the experiment.



Neural Network trained with three object images and tested with an array of six pictures.

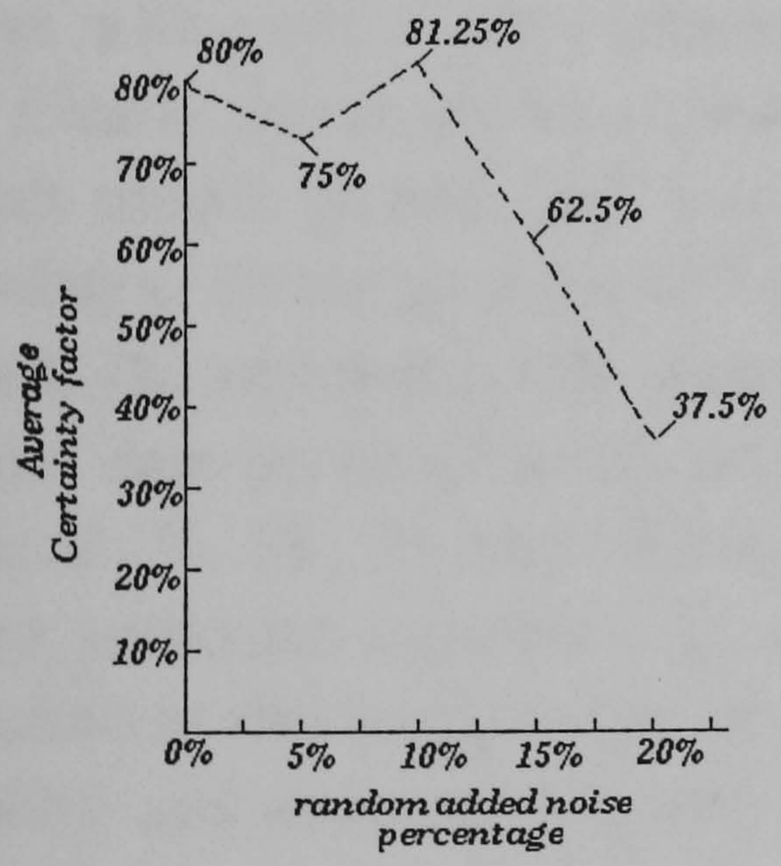
The first index at 0% random added noise represents the average of two pictures that were corrupted with 5% and 10% localised noise, while the four remaining pictures in the test array were images with 5%, 10% 15% and 20% random added noise.

No misclassifications have occurred

Figure 7.4. Performance of the second testing set.

Once again, the lower classification success for the 10% localised noise case can be noticed in Figure 7.2 but the fact that the index itself is relatively the same as the result in the 5% random added noise case, does not contribute in reducing the average for the 0% random added noise case substantially.

Figure 7.5 displays the performance of the last testing set of six pictures. Once again a discontinuity is present between the 5% random added noise pattern and the 10% random added noise pattern. After close observation of the actual testing patterns the reason for this anomaly became apparent. Adding random noise to a picture except from the main image corruption effect that it causes, aims additionally to alter the actual object skeleton contour present in the underlying picture. The more noise added in a picture the more distorted the object skeleton will be and the more difficult the recognition of the underlying object will become using the artificial neural system. In the pattern used for the 10% random added noise testing, the actual bits randomly altered within the test image were entirely bits that did not belong to the underlying object skeleton.



Neural Network trained with three object images and tested with an array of six pictures.

The first index at 0% random added noise represents the average of two pictures that were corrupted with 5% and 10% localised noise, while the four remaining pictures in the test array were images with 5%, 10% 15% and 20% random added noise.

No misclassifications have occurred

Figure 7.5. Performance of the third testing set.

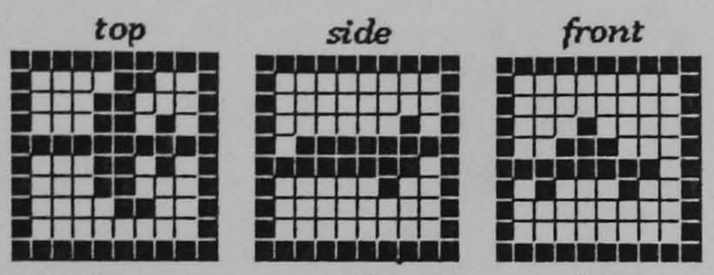
Obviously, leaving the underlying object skeleton intact during the testing phase will significantly enhance the classification success of the underlying neural network classifier. Once again as in the previous two cases there has been no misclassifications and the confidence had in general decreased as the

level of random added noise increased.

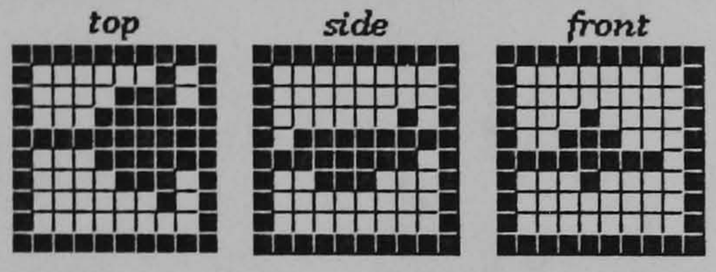
7.3 An example of simulated aircraft models.

7.3.1 Introduction.

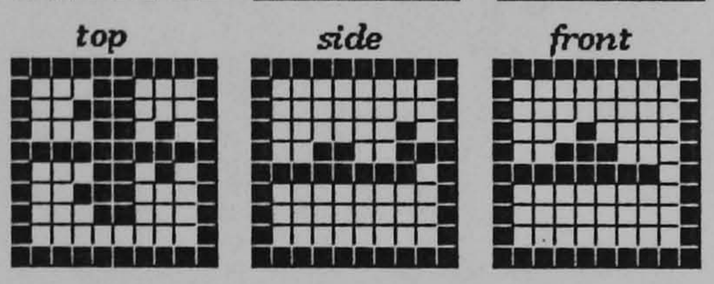
Here, a general example involving simulated aircraft data of size 8 by 8 by 8 octants is presented. The example will present the individual creation of quadtree and pseudo-octree data structures of the top, the side and the front view of one of three different objects selected in this experiment for training ADAM. The aim is to show the performance of ADAM neural network in successfully classifying the three aircraft models. For testing an array of six pictures is used again for each individual object. The testing array is structured as follows. The first two pictures represent actual aircraft volume images with parts of the corresponding aircraft model missing (i.e., a wing cut). Clearly this is not the case of randomly adding noise in the underlying aircraft model picture, but a rather mild form of highly localised noise degradation. In the previous two testing picture only aircraft skeleton bits are altered. The remaining four pictures in the testing set contain actual aircraft volume data corrupted with various levels of random added noise. Noise levels of 2%, 5%, 7% and 10% are used. It must be born in mind that the aim of this particular experiment is to be used as a representative of the entire procedure as this is applied in the context of aircraft data but in a rather easy to follow and understand format. In that context, Figure 7.6 shows the three different sets of picture views for each aircraft model to be used for training the artificial neural system.



The three non-coplanar object views of the first simulated aircraft model.



The three non-coplanar object views of the second simulated aircraft model.



The three non-coplanar object views of the third simulated aircraft model.

Figure 7.6. The sets of views of three simulated aircraft models.

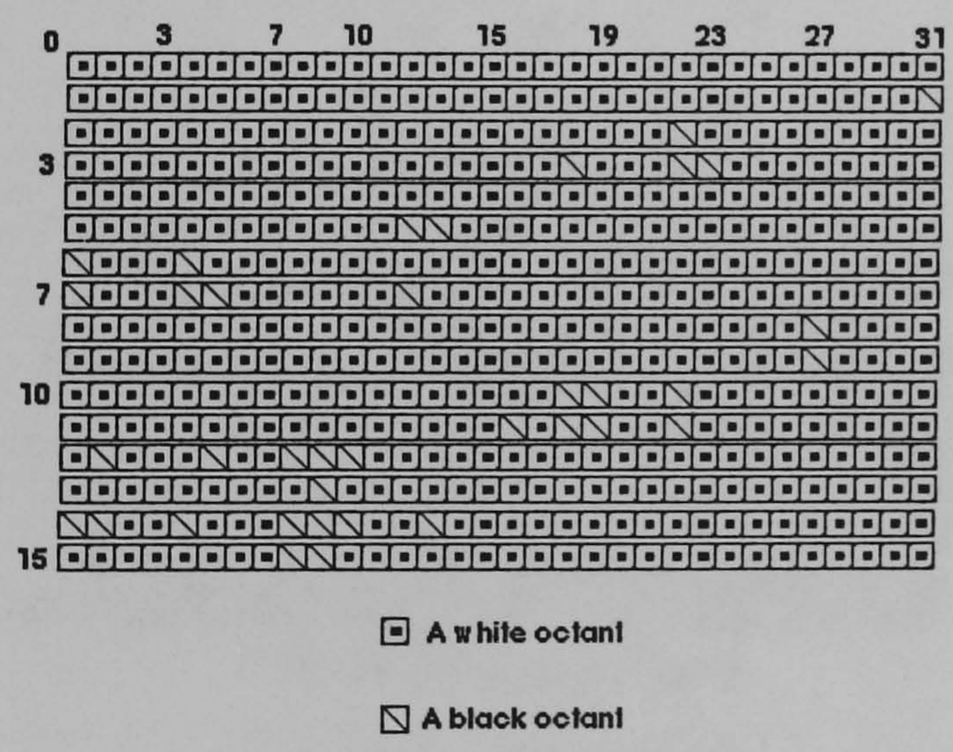
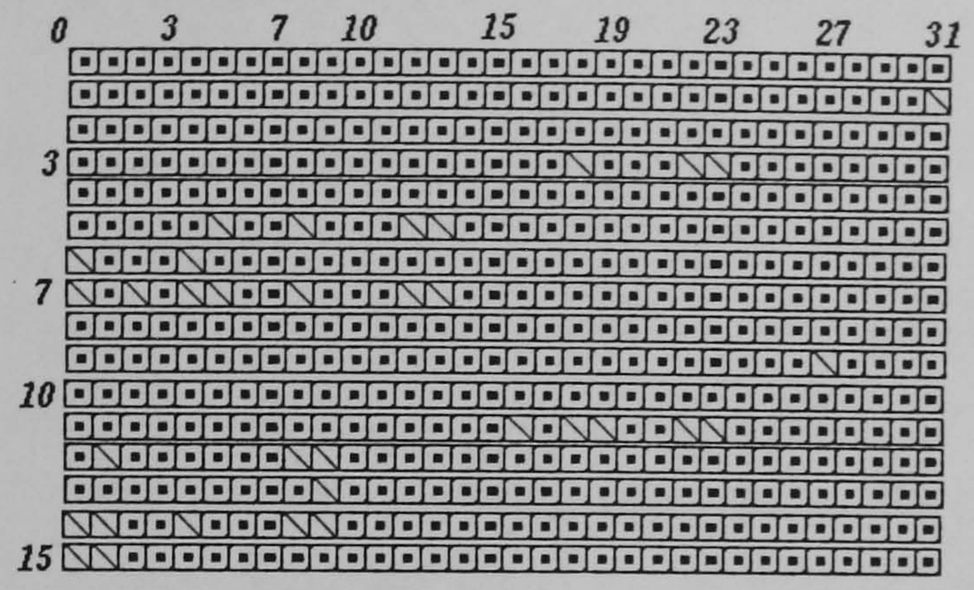
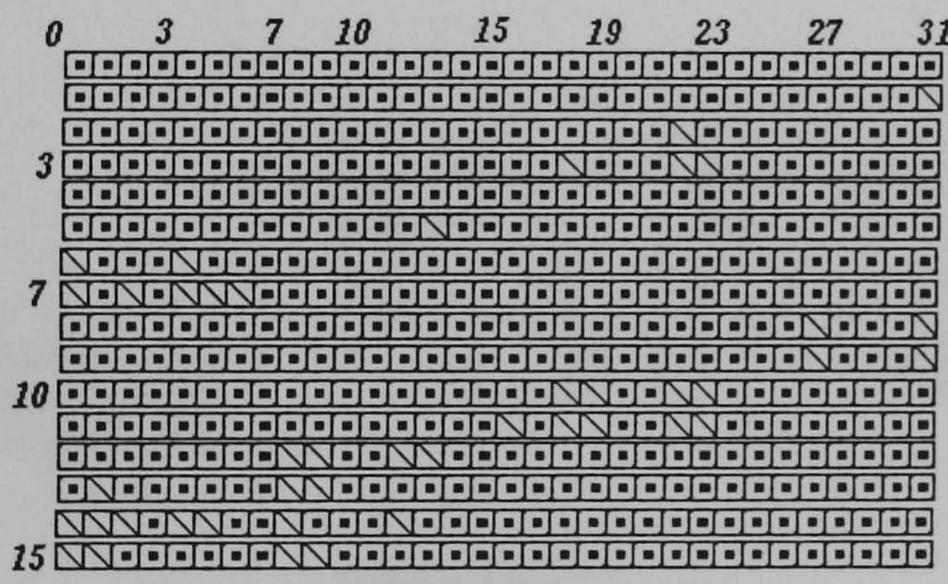


Figure 7.7. The synthetic image of the first aircraft model.



The synthetic image created from the octree structure of the second aircraft.

Figure 7.8. The synthetic image of the second aircraft model.

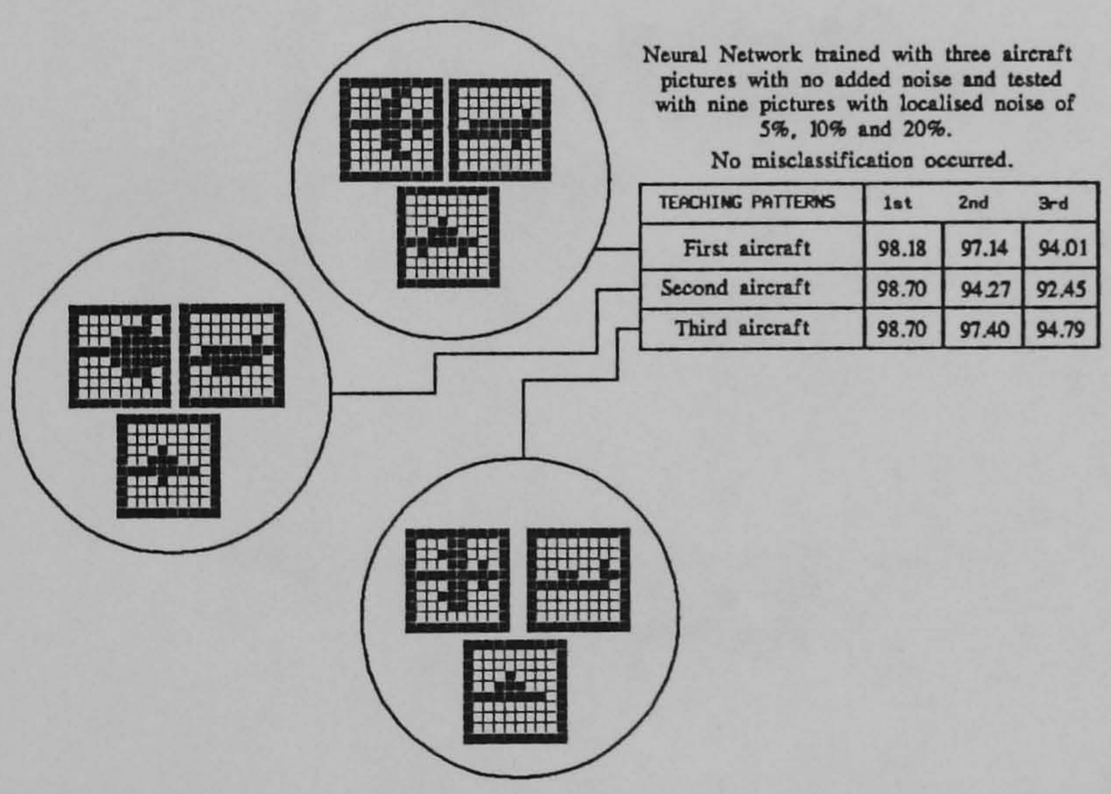


The synthetic image created from the octree structure of the third aircraft.

Figure 7.9. The synthetic image of the third aircraft model.

7.3.2 Training ADAM: A Performance Evaluation.

In order to test the classification performance of the artificial neural system, ADAM was trained using the three original simulated aircraft pictures, shown in Figures 7.7 to 7.9. The neural network was then tested using a simple set of 9 images with various levels of random added noise. The levels of noise used were 5%, 10% and 20%. The numerical results of this experiment are shown in Figure 7.10 while a graph representing the average classification success for the experiment according to the level of random added noise is shown in Figure 7.11. From both figures, it is clear that ADAM performed successfully with no misclassification recorded and with an average confidence of 95%. The classification success was steadily decreasing throughout the experiment, as the level of random added noise in the simulated aircraft volume pictures increased.



TEACHING PATTERNS	1st	2nd	3rd
First aircraft	98.18	97.14	94.01
Second aircraft	98.70	94.27	92.45
Third aircraft	98.70	97.40	94.79

Figure 7.10. ADAM's performance results when trained with simulated aircraft data.

If the results of this second test case are compared against the ADAM's performance when trained and tested with the random 4 by 4 by 4 octant objects, an important set of similarities can be seen. First, in both test cases

the neural network successfully classified objects without any misclassifications. Second, the neural system's behaviour towards random noise was always resilient although in both test cases the confidence ratios were decreasing as the amount of random added noise in the objects was increased. Finally, in both cases the neural network's classification success rate has been recorded to be significantly high; normally in the range between 85% to 90%. The only recorded difference in ADAM's pattern identification behaviour, has been the significantly higher confidence rate in the second case when 8 by 8 by 8 octant simulated aircraft patterns were used. In this test case classification success rates have increased by an average 7% in comparison to the first test case and were between 92% and 98%.

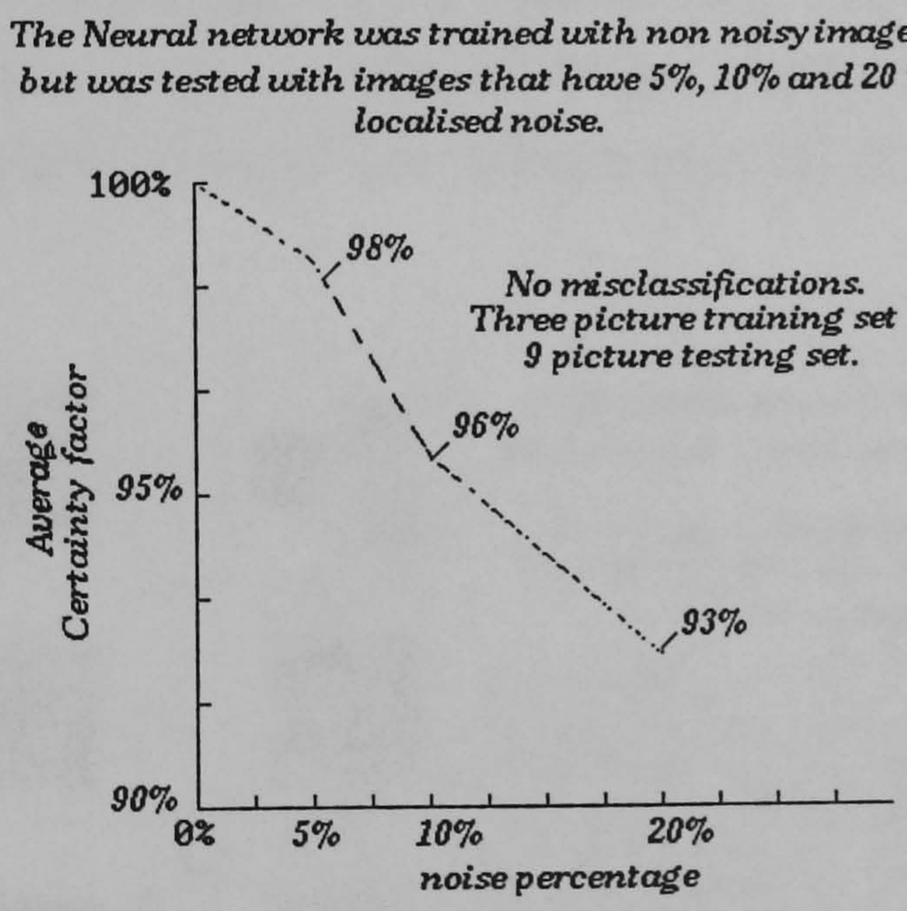
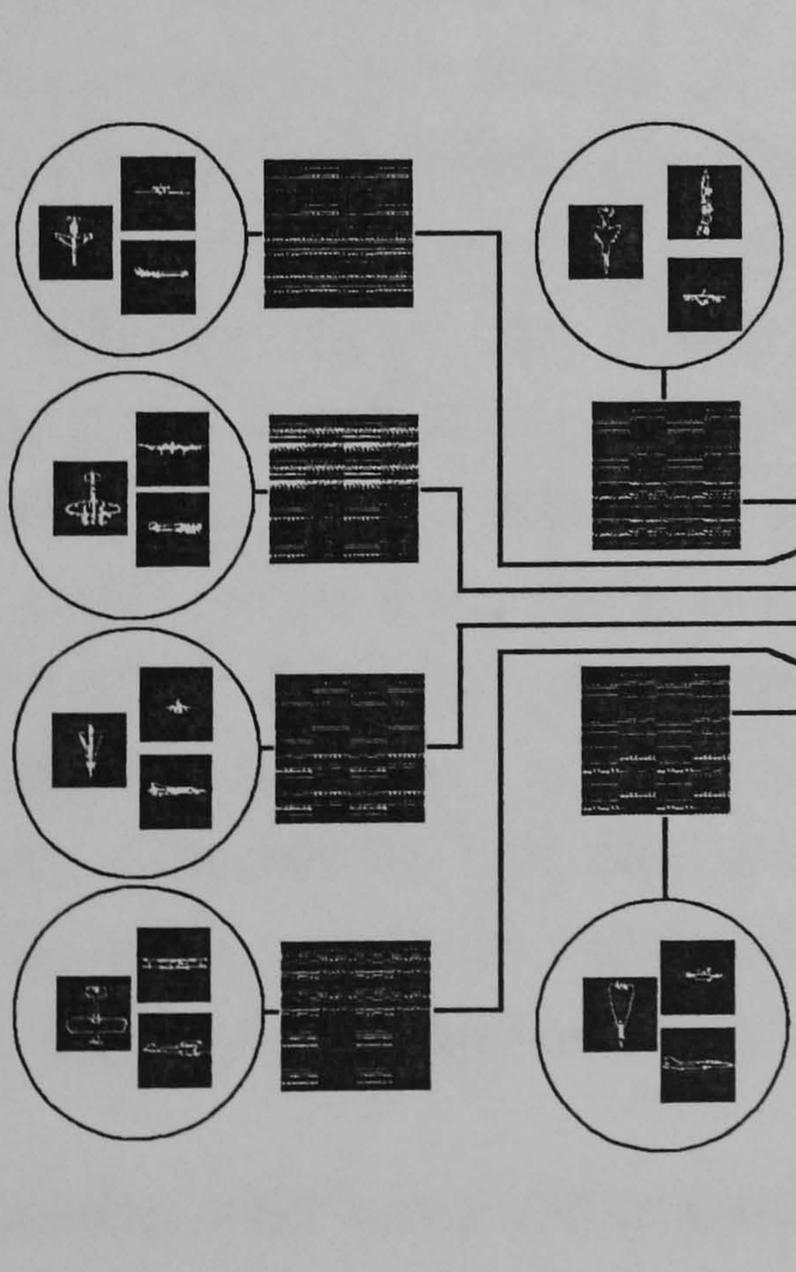


Figure 7.11. ADAM's average classification success when tested with 9 simulated aircraft pictures.

7.4 Examples of training and testing ADAM with real aircraft data.

7.4.1 Introduction.

The aim of these examples is to prove that ADAM can successfully classify real aircraft data and furthermore to show that the more patterns one uses to train ADAM the better the classification factor will become. The latter implying that the underlying neural network architecture can effectively generalise. Another aim is to prove that recognition of three dimensional objects can in fact be successful independent of the actual object's position or relative sizes. The invariant pattern classification success with the limited training pattern set requirement, will make the application of ADAM an important and significant contribution in the current research for efficient position invariant pattern recognition using artificial neural network systems.



Edge Detected Images (Boundaries Only)
Neural Network trained with non noisy images.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	98.45	97.28	93.40	84.08	79.03	72.19
CARAVELLE	91.14	90.87	93.43	84.08	79.03	72.19
BEAUFIGTER	97.60	96.14	93.34	83.91	78.92	71.98
SAAB DRAGEL	97.89	97.27	93.47	84.11	79.12	72.28
BRISTOL F2B	97.82	96.38	93.35	84.03	79.06	72.23
F14 TOMCAT	98.84	97.91	93.45	84.10	79.08	72.24

No misclassification occurred

Figure 7.12. Classification data of the first experiment.

A performance presentation discussion in the end of these set of experiments will attempt to summarise all the experimental data and comment on the recognition process relative advantages and limitations. Two types of aircraft models are to be used. The simple skeleton boundaries and the filled skeleton boundaries. It is argued, that the filled skeleton boundaries will perform better under the Volume Intersection method, since more actual object surface will be available for the final pseudo-volume merging operation. The experiments that will follow will provide useful performance related data in order to justify whether such claims are valid.

7.4.2 Training with a non-noisy pattern set (Simple Skeletons Only).

In this experiment the original set of six real aircraft volume pictures was used to train ADAM. For testing, a set of thirty six pictures had been formed in groups of six images for each one of the six different aircrafts. The classification performance data are displayed in Figure 7.12, while a performance graph of the average classification success can be seen in Figure 7.13. ADAM had successfully classified all thirty six patterns used for testing and its confidence decreased as the level of random noise added to the test patterns increased. This fact clearly demonstrates that one picture and a single cycle for training are sufficient to provide a noise-free successful classification for the six aircrafts used in the experiment. The only less efficient result in the first experiment is the actual value of the confidence which although was high enough (around 93%) for low noise corruption levels, became a rather undesirable 72% for higher levels of noise. Subsequent experiments will aim to increase this low classification success rate for the higher levels of randomly added noise in the expense though of generally larger training patterns sets and longer training times.

7.4.3 Training with a non-noisy pattern set (Filled Boundaries).

Once again the original six aircraft images were used with the only

addition that this time the set with filled boundaries was selected. For testing, again a set of thirty six filled boundaries images was used. The classification results are presented in Figure 7.14. The average confidence graph can be seen in Figure 7.15 and shows the expected decline in classification success rate as the level of randomly added noise increased. In that particular case though, the discontinuity recorded in the confidence around the 2% random added noise level needs to be justified. Figures 7.16 and 7.17 describe the percentage of localised noise used for each one of the two pictures that form the result for the 0% random added noise level. As it is already described in a similar situation during the classification phase of the three simulated aircraft models previously presented, the obtained average of both pictures is severely influenced by the percentage of localised noise applied to them. Figure 7.16 clearly shows that although the average noise level in the simple skeleton pictures has been much less than the immediate random added noise level in the graphs (i.e., 2%), in the case of filled boundaries the average percentage of localised noise in pictures has been recorded to be 3.5%, almost twice as the immediate random added noise level of 2%. Clearly, the more noise added in a test pattern the less the confidence of the classification will become. And this is the case in that particular experiment. Apart from this remark, the average classification success in the filled boundaries experiment has not presented any significant differences to the case of simple skeleton boundaries and once again a low confidence was noticed for the 10% random added noise level.

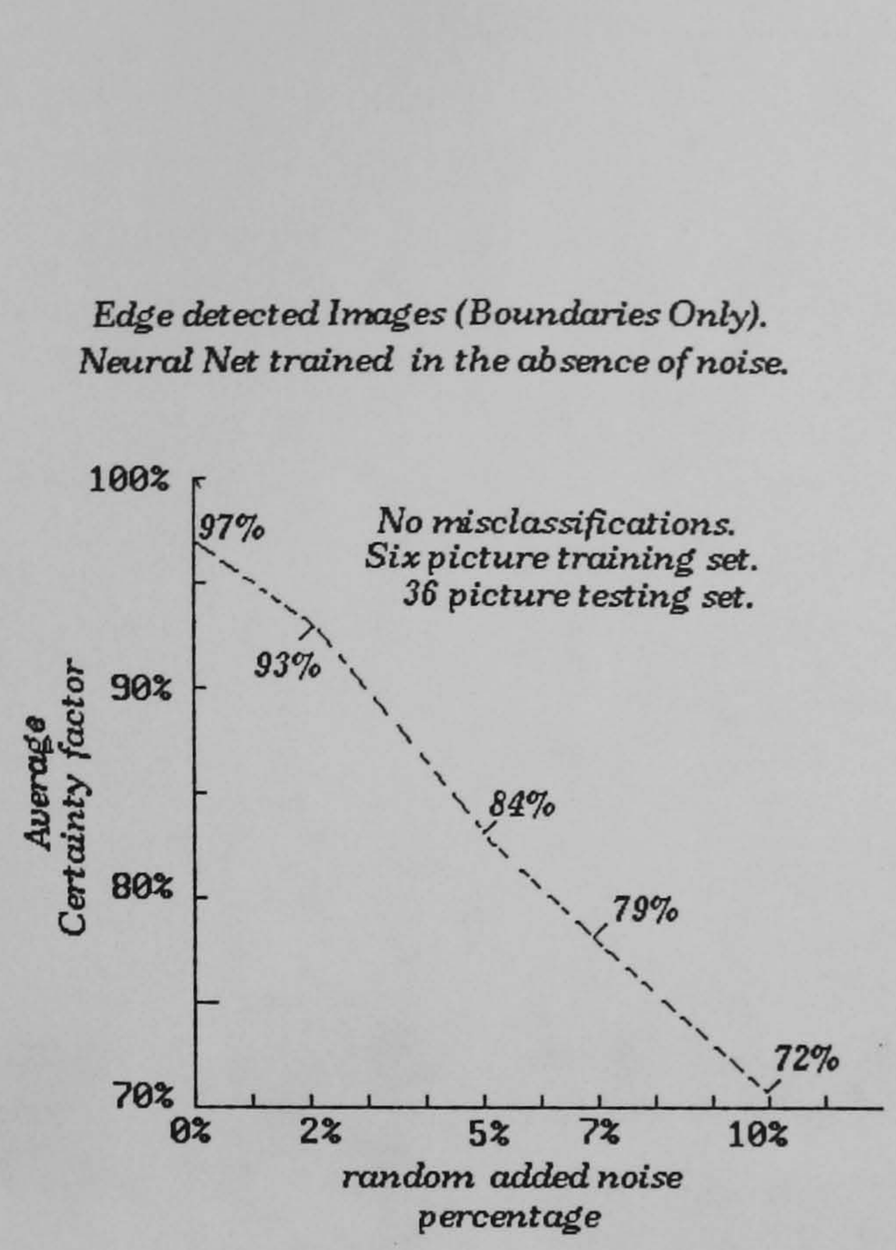


Figure 7.13. Average classification success graph for the first experiment.

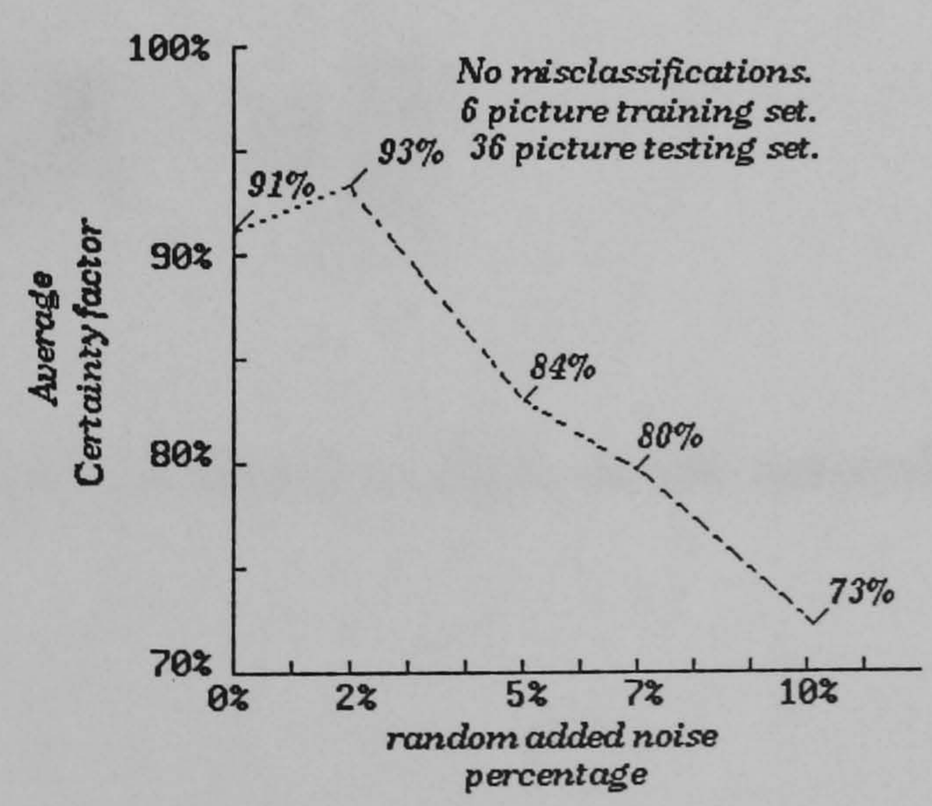
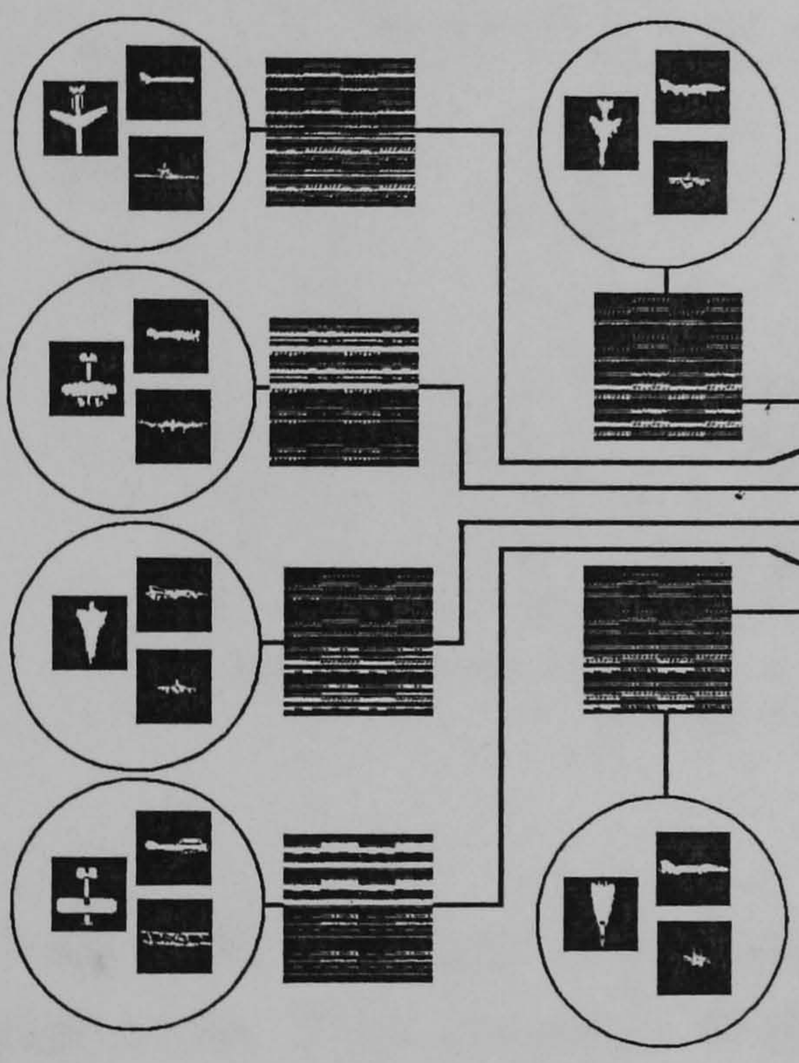


Figure 7.15. Average classification success graph for the second experiment.



Edge Detected Images (Filled Boundaries)
Neural Network trained in the absence of noise.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	94.96	94.77	93.74	84.87	80.23	73.76
CARAVELLE	91.78	93.52	93.65	84.84	80.11	73.54
BEAUFIGTER	93.39	91.81	93.73	85.02	80.34	73.63
SAAB DRAGEL	92.32	93.87	93.85	85.13	80.54	74.17
BRISTOL F2B	91.69	87.10	93.65	84.93	80.25	73.46
F14 TOMCAT	91.91	95.33	93.83	85.05	80.44	74.04

No misclassification occurred.

Figure 7.14. Classification data of the second experiment.

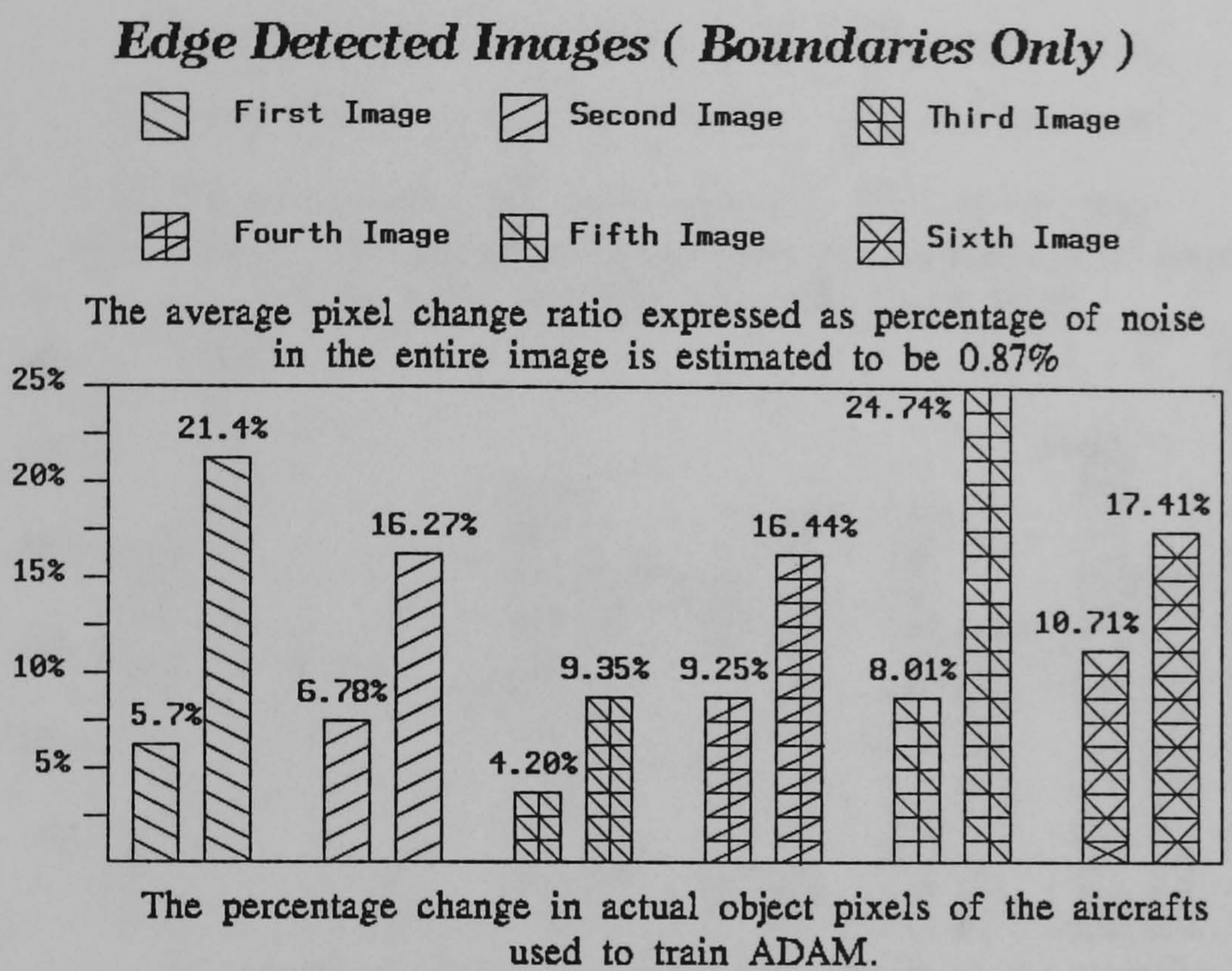


Figure 7.16. The average percentage of localised noise in the simple boundaries case. Two pictures with localised noise were used for each one of the six aircraft test patterns.

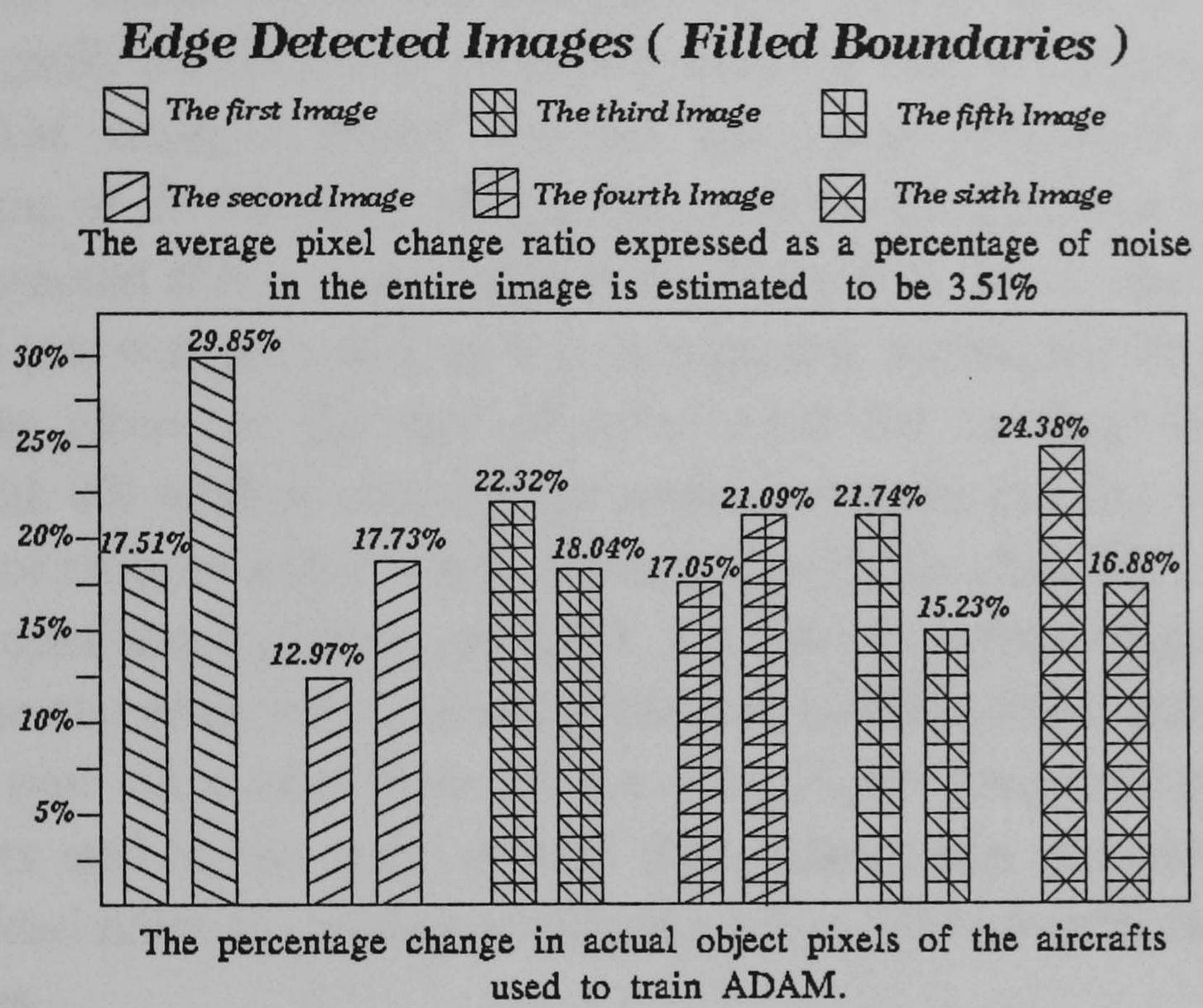
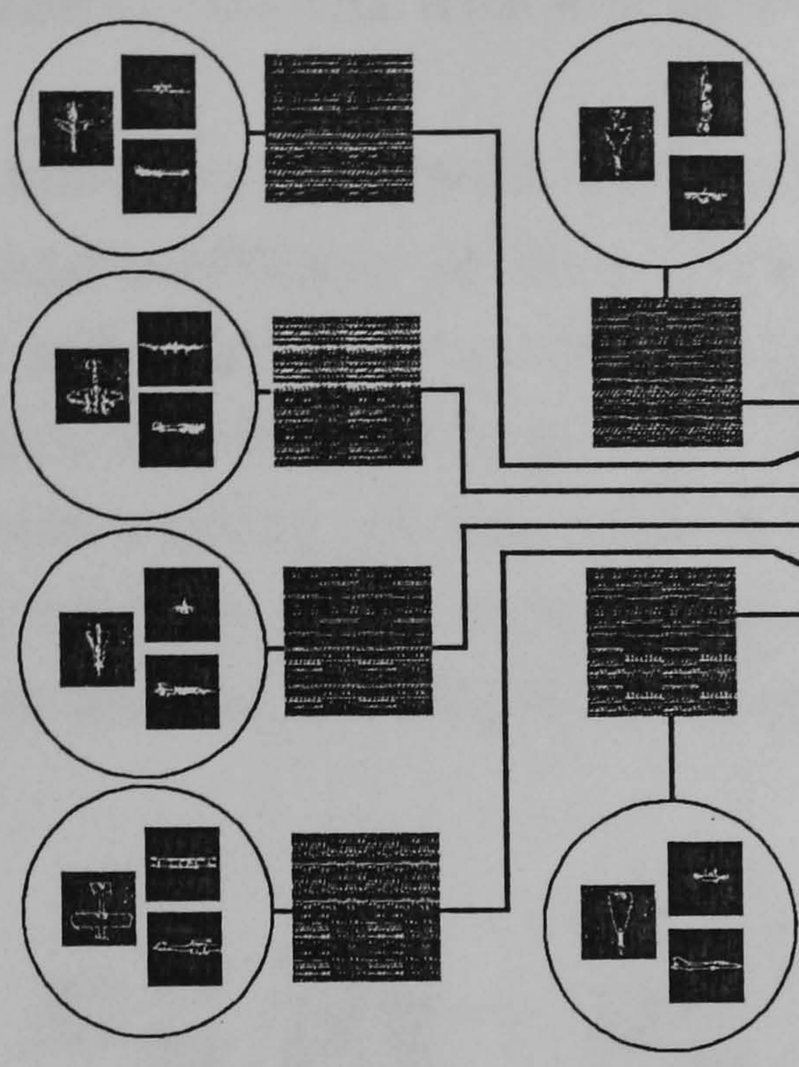


Figure 7.17. The average percentage of localised noise in the filled boundaries case. Two pictures with localised noise were used for each one of the six aircraft test patterns.

7.4.4 Training ADAM with 4% noise (Simple Skeletons).

In this experiment in addition to the original six aircraft training set, a set of six aircraft volume images with 4% random added noise is also used for training. The classification performance data can be seen in Figure 7.18, while the graph of the average classification success is displayed in Figure 7.19. ADAM artificial neural network has again performed a successful classification of all 36 used test patterns and the average confidence has in general increased and was recorded to be above 80% in all cases. The higher confidence rate was recorded, as it was expected, around the level of random added noise closer to the rate of noise used for training ADAM. Since patterns with 4% random added noise were used in the training phase, and the only level of random added noise closer to the 4% level is 5%, it is there that the higher confidence of 96% appeared. An other important observation is the fact that the classification success for the non-random added noise cases (0% level) has now decreased from the previous higher levels recorded in other experiments and it has been around 85%. Obviously, the introduction of random added noise in training averaged against the non-random added noise test patterns.



Edge Detected Images (Boundaries Only)
Neural Network trained with 4% noisy pictures.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	85.42	84.48	92.79	96.61	90.14	81.43
CARAVELLE	79.33	79.18	92.77	96.59	90.11	81.38
BEAUFIGTER	84.73	83.83	92.71	96.55	90.14	81.28
SAAB DRAGEL	85.20	84.68	92.79	96.60	90.20	81.50
BRISTOL F2B	84.94	83.84	92.79	96.58	90.18	81.48
F14 TOMCAT	85.88	85.14	92.80	96.60	90.14	81.43

No misclassification occurred.

Figure 7.18. Classification data of the third experiment.

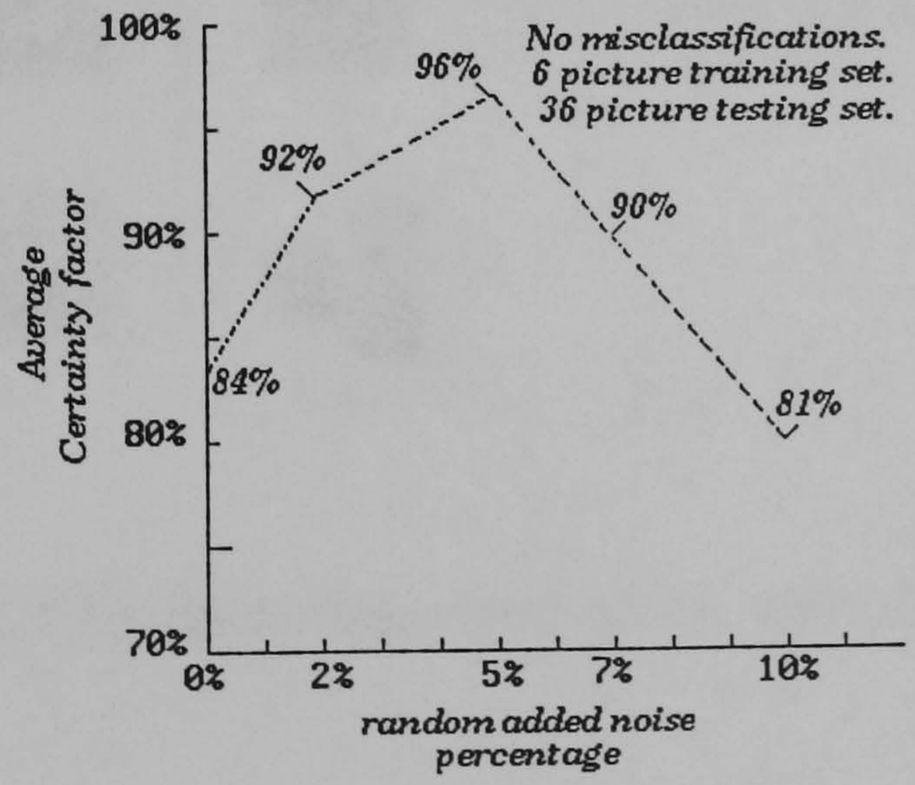
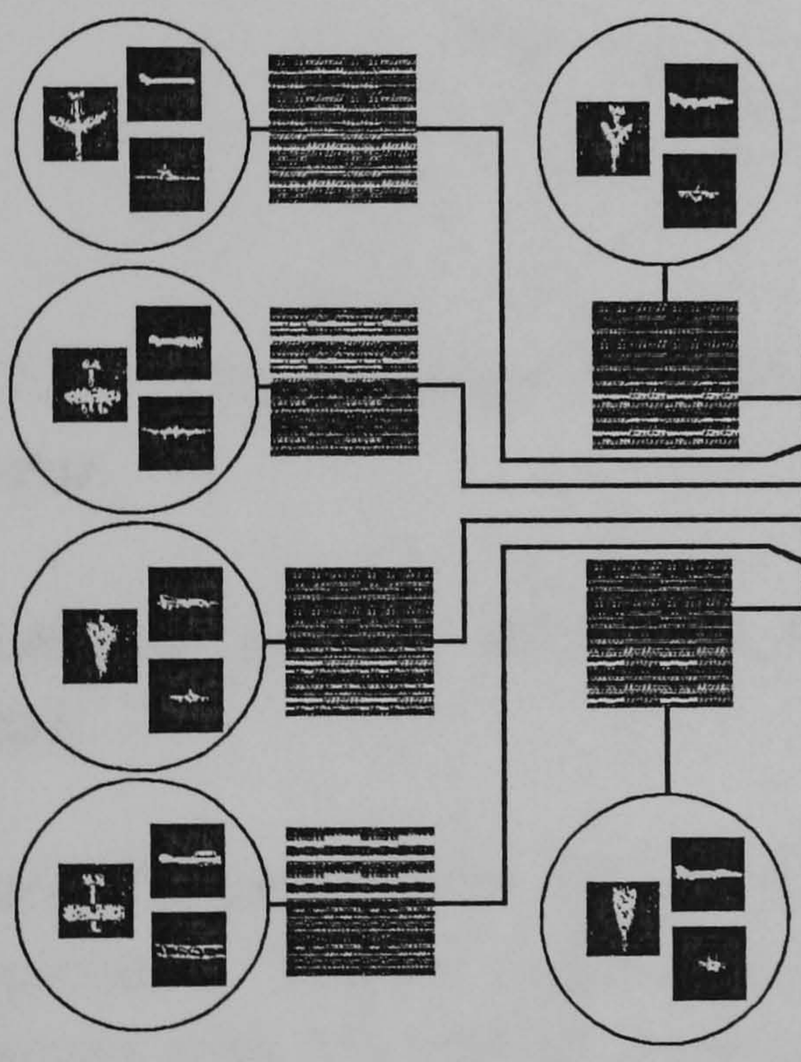


Figure 7.19. The average classification success graph for the third experiment.

7.4.5 Training ADAM with 4% noise (Filled Boundaries).

The classification results of this experiment are displayed in Figure 7.20. The average confidence graph is shown in Figure 7.21. No anomaly has been recorded and once again the higher classification success appears around the 5% random added noise level in accordance with the fact that a 4% random added noise training set was used. A slightly better classification success for the 10% random added noise case is also recorded in comparison to the equivalent level in the simple skeletons graph.



Edge Detected Images (Filled Boundaries)
Neural Network trained with 4% noise.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	83.78	83.40	93.24	96.64	90.65	82.42
CARAVELLE	80.85	82.00	93.25	96.65	90.52	82.91
BEAUFIGTER	82.24	81.07	93.30	96.71	90.67	82.21
SAAB DRAGEL	81.40	82.65	93.34	96.69	90.78	82.67
BRISTOL F2B	80.91	77.18	93.32	96.68	90.61	82.05
F14 TOMCAT	81.49	83.82	93.30	96.67	90.72	82.58

No misclassification occured.

Figure 7.20. Classification data of the fourth experiment.

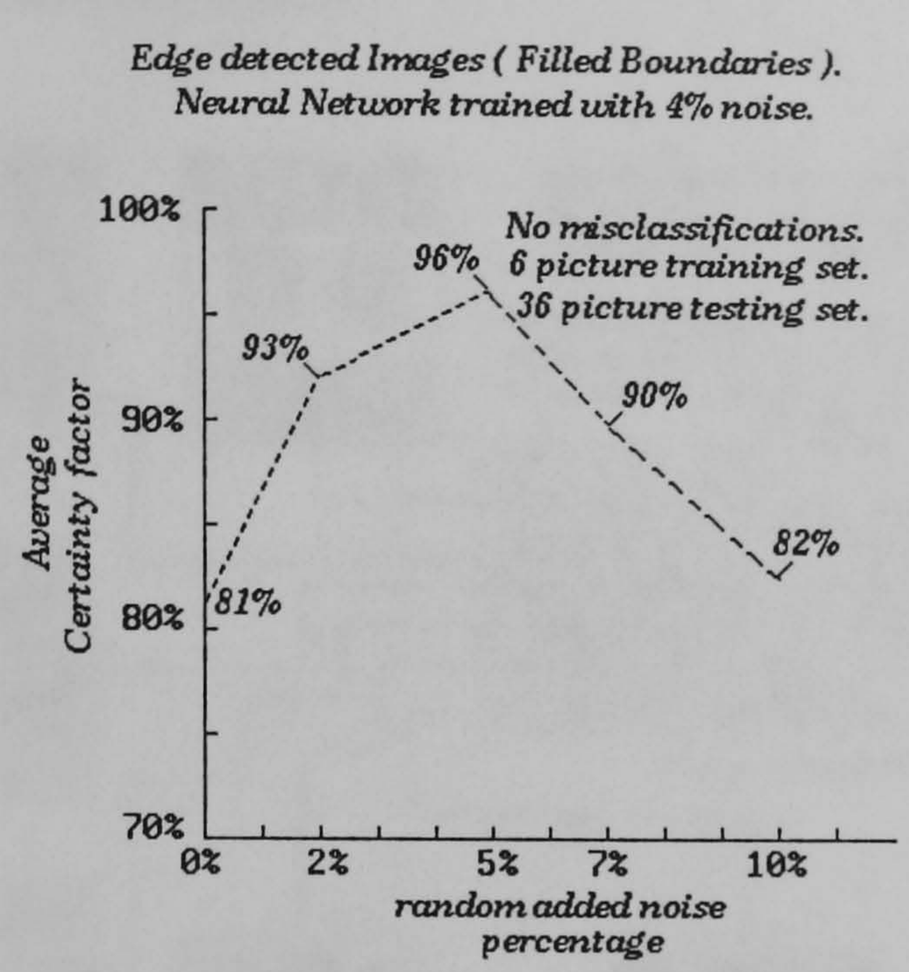
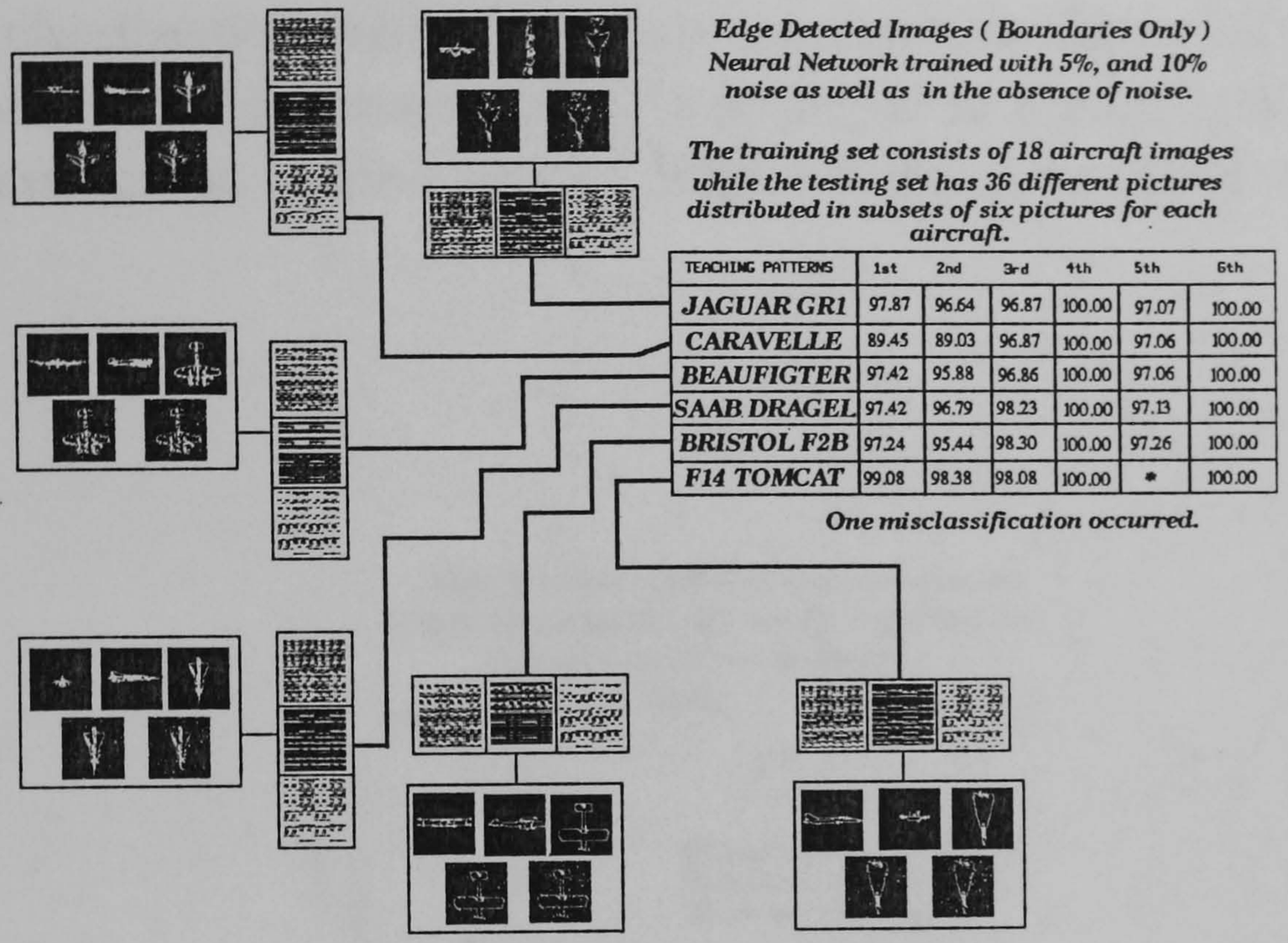


Figure 7.21. The average classification success graph for the fourth experiment.

7.4.6 Training ADAM with two random added noise levels (Simple Skeletons).

In this fifth experiment, the ADAM artificial neural network was trained with 18 pictures. The six original aircraft volume images, and 12 more split in two groups with 5% and 10% random added noise. A new set of thirty six images was used for testing. The classification performance of ADAM was almost perfect with only one misclassification recorded in 36 test patterns. The actual classification results are displayed in Figure 7.22, while the average confidence performance is shown in the graph of Figure 7.23. The success in classification has soared and it was recorded in all cases to be above 90%. Particularly, for the two testing levels of 5% and 10% that were exactly the same noise levels with the noise present in the noisy patterns used to train ADAM, the classification success reached the absolute perfect level (100%). It is also important to be noticed that the confidence rate for the

non-random added noise test patterns had again reached the levels originally seen in the first experiment (93%).



TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	97.87	96.64	96.87	100.00	97.07	100.00
CARAVELLE	89.45	89.03	96.87	100.00	97.06	100.00
BEAUFIGTER	97.42	95.88	96.86	100.00	97.06	100.00
SAAB DRAGEL	97.42	96.79	98.23	100.00	97.13	100.00
BRISTOL F2B	97.24	95.44	98.30	100.00	97.26	100.00
F14 TOMCAT	99.08	98.38	98.08	100.00	*	100.00

Figure 7.22. Classification data of the fifth experiment.

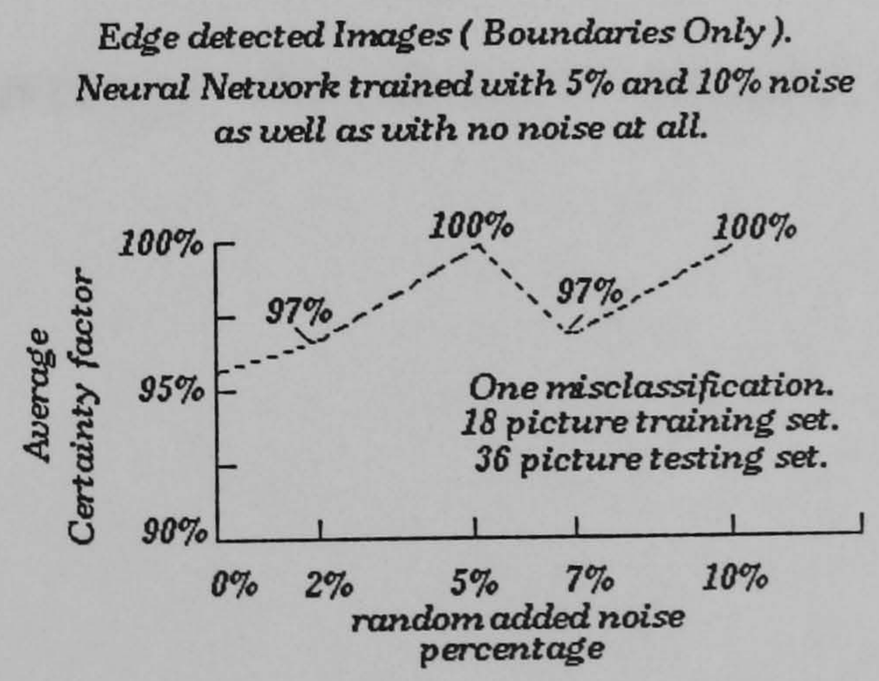


Figure 7.23. The average classification success graph for the fifth experiment.

7.4.7 Training ADAM with two random added noise levels (Filled Boundaries).

The classification performance data are shown in Figure 7.24, while the average classification success graph is displayed in Figure 7.25. A slightly lower classification success was recorded for the non-random added noise patterns.

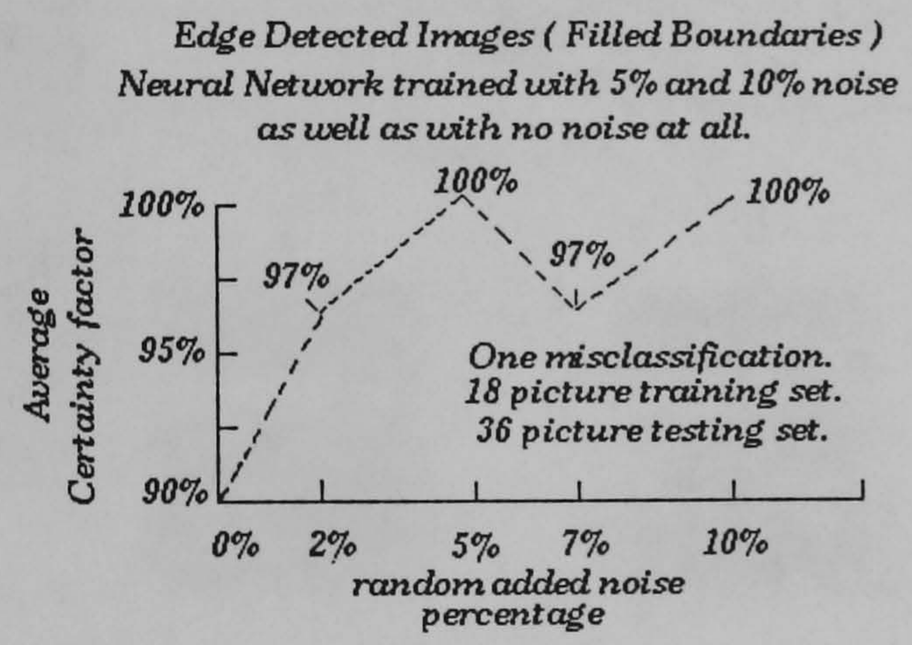
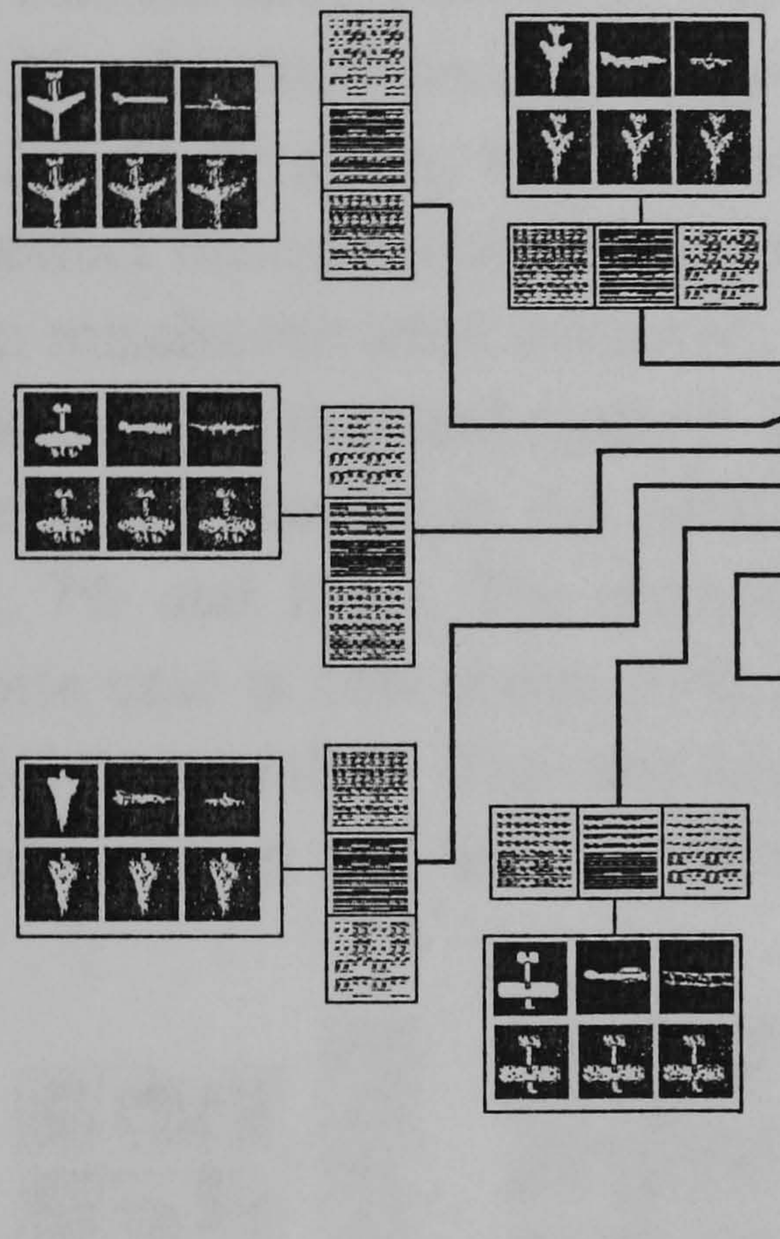


Figure 7.25. The average classification success graph for the sixth experiment.



Edge Detected Images (Filled Boundaries)
Neural Network trained with 5% and 10% noise as well in the absence of noise.

The training set consists of 18 aircraft images while the testing set has 36 different pictures distributed in subsets of six pictures for each aircraft.

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	94.22	93.68	96.92	100.00	97.09	100.00
CARAVELLE	92.95	93.45	97.70	100.00	96.95	100.00
BEAUFIGTER	92.54	91.06	97.84	100.00	97.31	100.00
SAAB DRAGEL	90.18	92.15	96.93	100.00	97.09	100.00
BRISTOL F2B	90.21	85.61	96.93	100.00	97.07	100.00
F14 TOMCAT	89.90	93.24	97.84	100.00	*	100.00

One misclassification occurred.

Figure 7.24. Classification data of the sixth experiment.

7.4.8 Training ADAM with three random added noise levels (Simple Skeletons).

The classification results of this particular experiment are displayed in Figure 7.26, while the average classification success performance is shown in Figure 7.27. Once again, the ADAM artificial neural system performed an almost perfect classification on all the thirty six different test patterns used. Only one misclassification occurred and the average confidence was 100% for all the pictures that had random added noise testing levels the same with the noise levels present in the patterns that were used for training ADAM (i.e., 5%, 7% and 10%). The classification success rate for the non-random added noise case is now above 95% and the system clearly shows that it can successfully generalise. The training set consisted of 24 different images while the testing set was fixed to thirty six pictures.

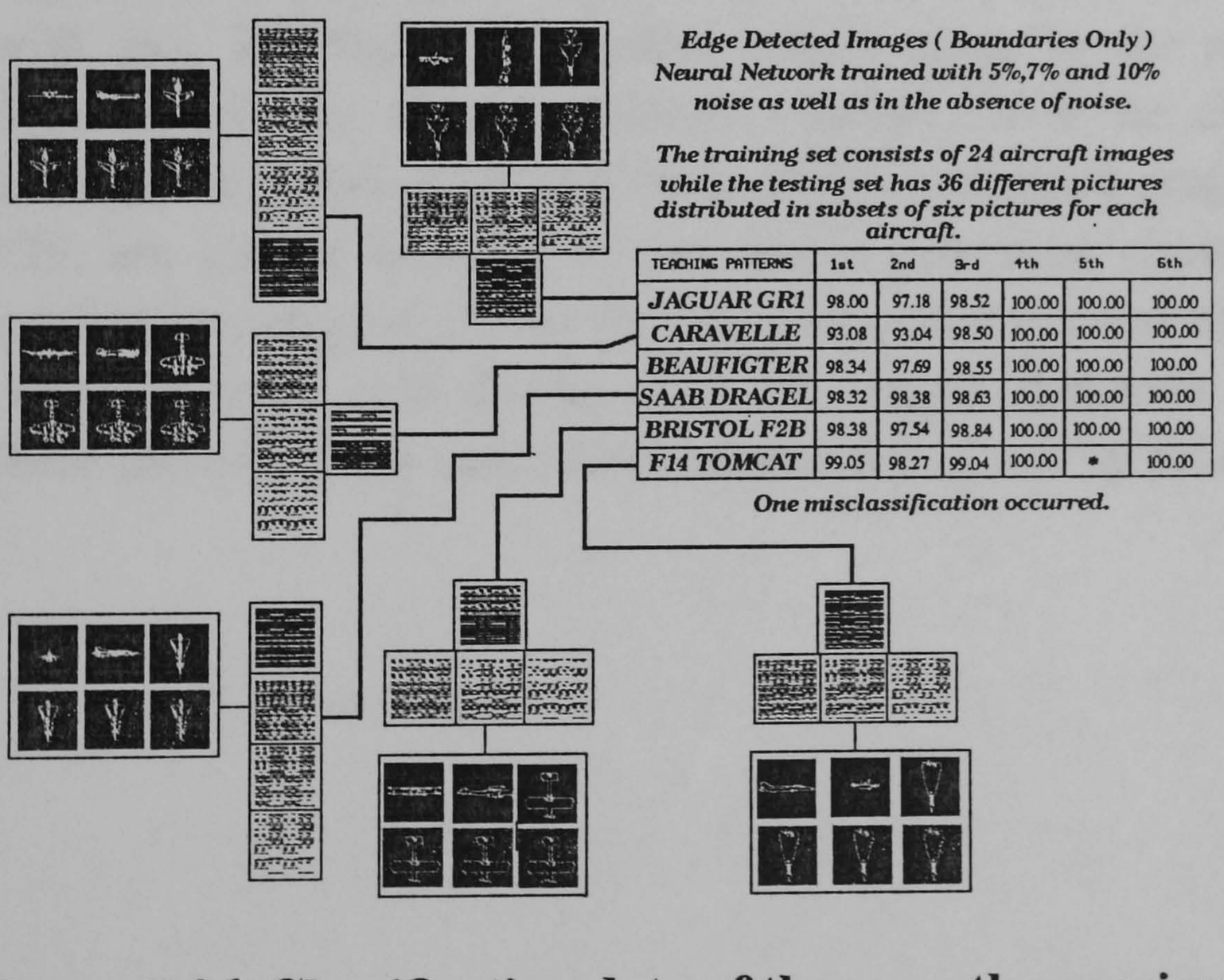


Figure 7.26. Classification data of the seventh experiment.

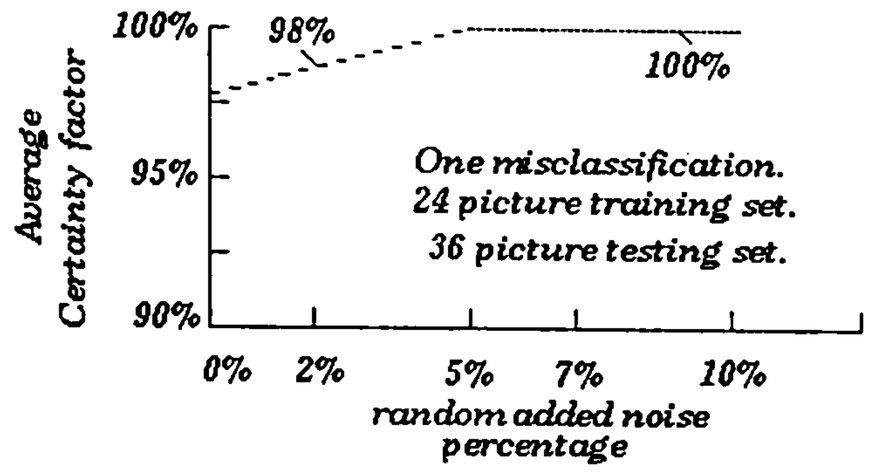
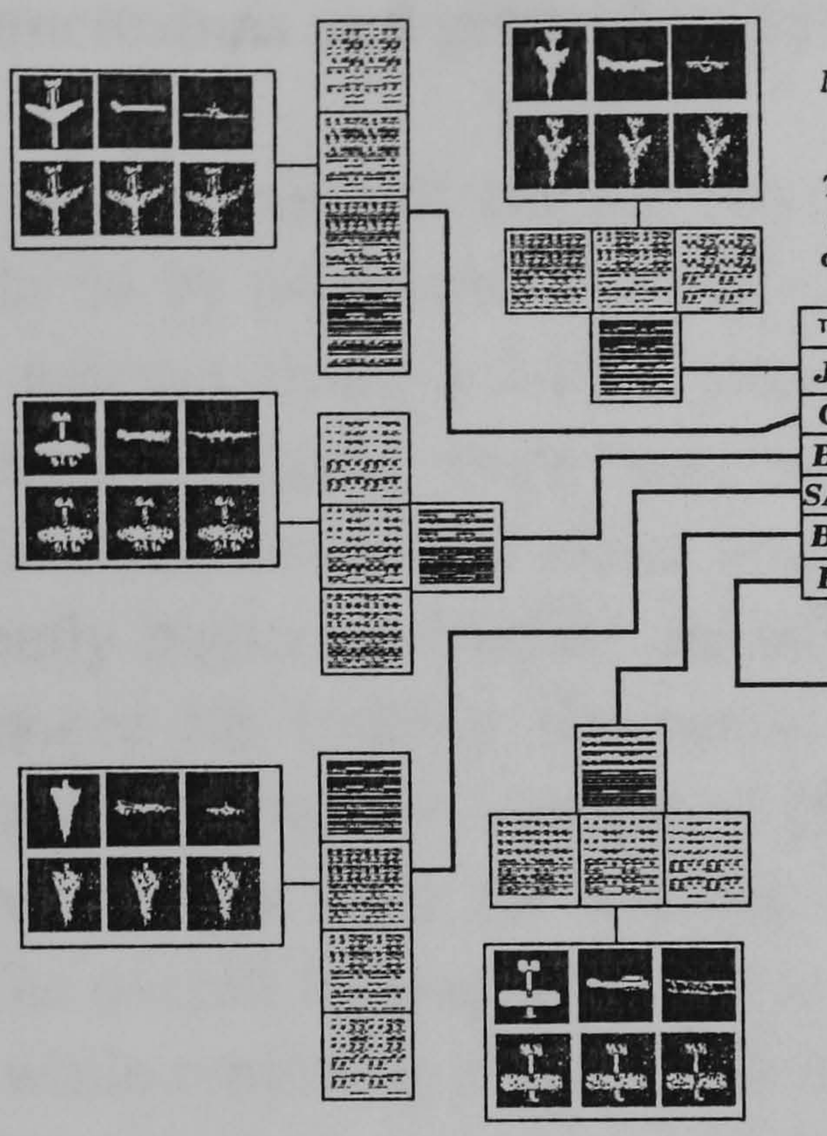


Figure 7.27. The average classification success graph for the seventh experiment.

7.4.9 Training ADAM with three random added noise levels (Filled Boundaries).

No significant difference was recorded when the neural system was trained with 5%, 7% and 10% random added noise to the performance presented above for the simple skeletons. Indeed, both the classification results presented in Figure 7.28, and the average confidence graph shown in Figure 7.29, are almost identical to the results presented above. A rather lower classification success (95%) for the non-random added noise case is noticed, than the same case displayed in the simple skeletons experiment (98%). There has only been one misclassification in a testing set of thirty six patterns.



Edge Detected Images (Filled Boundaries)
Neural Network trained with 5%,7% and 10% noise as well as in the absence of noise.

The training set consists of 24 aircraft images while the testing set has 36 different pictures distributed in subsets of six pictures for each aircraft.

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	95.08	95.31	98.66	100.00	100.00	100.00
CARAVELLE	93.56	94.55	98.83	100.00	100.00	100.00
BEAUFIGTER	95.30	94.34	97.97	100.00	100.00	100.00
SAAB DRAGEL	93.42	94.54	98.69	100.00	100.00	100.00
BRISTOL F2B	91.37	89.47	97.39	100.00	100.00	100.00
F14 TOMCAT	*	94.62	98.87	100.00	100.00	100.00

One misclassification occurred.

Figure 7.28. Classification data of the eightieth experiment.

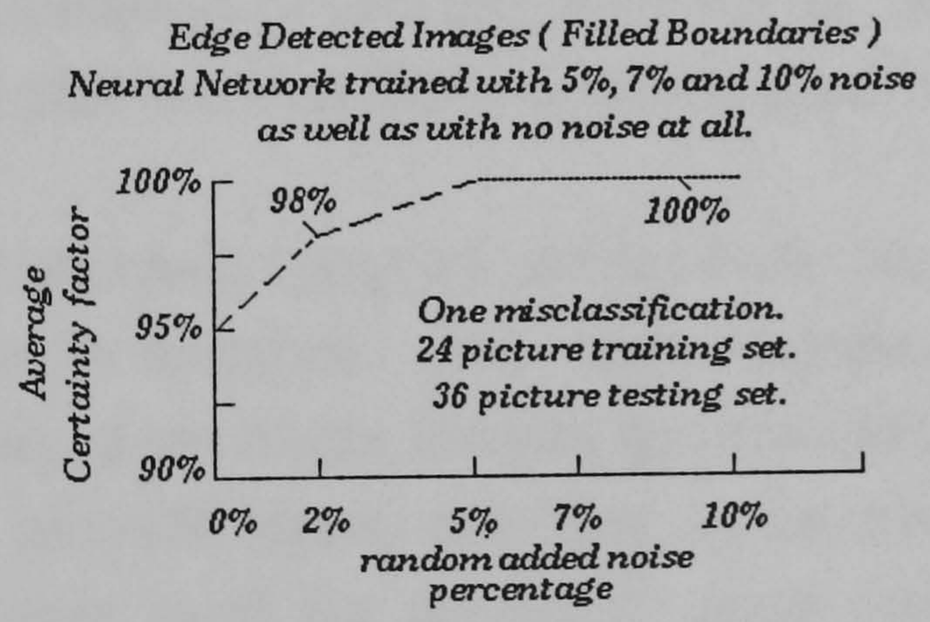


Figure 7.29. The average classification success graph for the eightieth experiment.

7.4.10 Conclusions and general performance assessment.

The original aircraft images used were 512 by 512 pixels, and were reduced to 64 by 64 pixels. This reduction in image size, was achieved in a two step process, using a 2 by 2 pixel neighbourhood averaging. Although clear structural details were lost, the reduction in size has resulted in substantial reduction in the noise levels present in the original picture and consequently higher probability for pattern matching, as well as structurally better images for training the neural network. The N-tuple size used in ADAM was four and the number of bits set in each class pattern was three. The octree images used for training ADAM were all of size 512 by 512 pixels. The overall training time per image on a Sun 3/50 averaged about 50 seconds while retrieving associations took on average 45 seconds per image. The time required on average to read an octree image was 5 seconds. Only one synthetic octree image is required for each one of the six different aircraft types used in the experiments. The neural network has been trained originally with or without noise although in later stages and in order to investigate its generalisation properties ADAM has been trained both with noise as well as with pictures that have no added random noise.

To summarise, two basic types of images have been used. Edge detected images with simple boundaries, and edge detected images with filled boundaries. For testing, a set of six images has always been used for each one of the six different aircraft types, resulting in an overall testing set of 36 images. The six images used for testing in each one of the six prototype aircraft included: two images (a,b) with no added random noise (having a significant number of deleted structural object pixels [15% on average]), and four images (c,d,e,f) of the original object volume with added random noise of 2%, 5%, 7% and 10%. In the first set of experiments a set of only six octree images with no added random noise was used. When simple boundary pictures were used, ADAM showed a higher confidence towards the two pictures with no added random noise, while the classification success

decreased as the level of added noise in pictures increased. The neural network showed a confidence ranging from 97% for the structurally distorted images (a,b), to 72% for the octree images with 10% added random noise (f). Aircraft classification has always been correct. When pictures with filled boundaries were used, ADAM showed a similar behaviour ranging from 92% for pictures (a,b), to 73% for the images with 10% added random noise (f). There was a small difference in that specific graph between the pictures (a,b) and the images with 2% added random noise (c). The latter are shown to have a higher confidence ratio than the former. The reason behind that anomaly is simply the fact that the percentage of object pixels distorted in pictures (a,b) was as much as 3.5%. Altering a large number of pixels on the filled boundaries of an object results in a form of localised noise addition, quite similar to a general random noise addition in an object's octree volume.

In general, for the first set of experiments in the absence of any added random noise in the training set of six pictures, ADAM always showed greater preference towards the test pictures with the lower quantity of added random noise (localised noise). For the second set of experiments six octree pictures with 4% added random noise were used for training. When edge detected images with simple boundaries (skeletons) were used, ADAM showed higher confidence around the level of noise that was closer to the level of noise with which the neural network was trained. As a 4% added random noise was used for training, ADAM responded favourably at 5% added random noise in testing. The confidence curve shown for the second set of experiments had a peak value at 5% added random noise and lower confidence ratio at (+/-) 5% from both sides of the peak value. ADAM was recorded to have a maximum confidence of 96% and an overall minimum value of 81% for the pictures with 10% added random noise (f). It is important to realise that there were no misclassifications. In the case of filled boundaries edge detected pictures, the neural network performed almost identically with its peak again found near the (b) images and the overall minimum value at 81% on the side with the (a,b) images. The reason for the

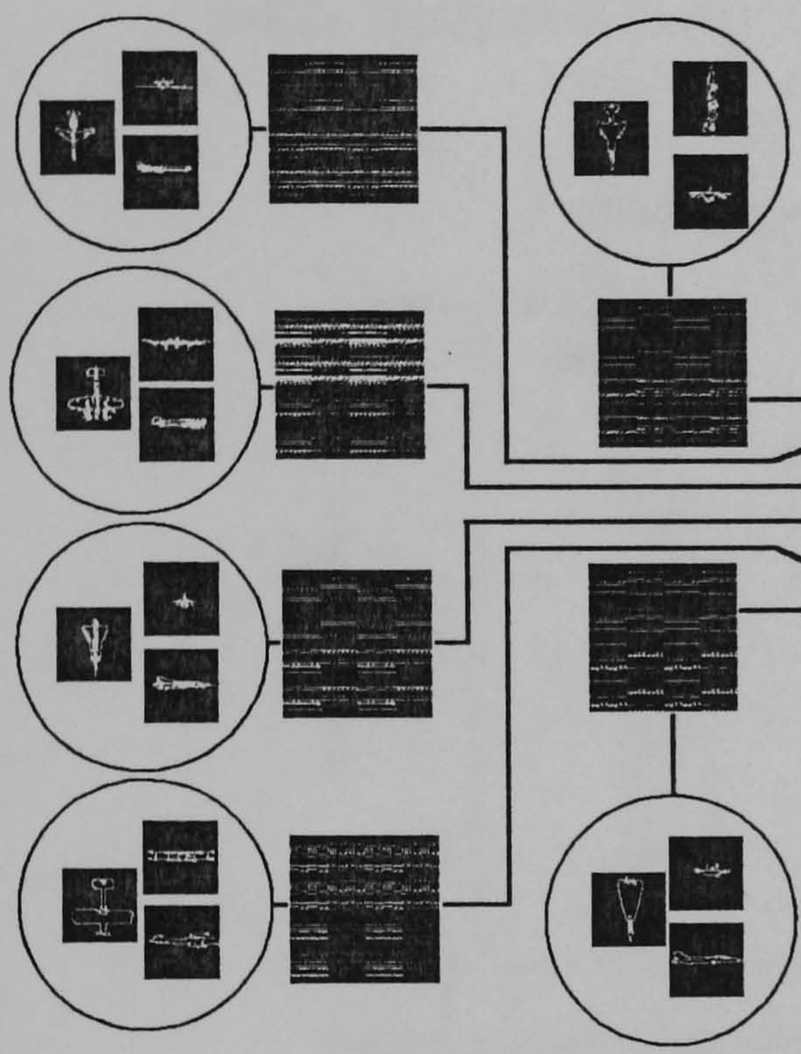
later differentiation is the same as that described previously in the first set of experiments.

To conclude, for the second set of experiments when the ADAM was trained with 4% added random noise it performed most favourably with the images where the noise level was the same as the noise level it was trained with. In addition the overall confidence had improved significantly from the first set of experiments with the lowest confidence rate being above 80%. There were again no misclassifications. In order to test the generalisation properties of the current neural network a new set of experiments was designed for patterns with added random noise and patterns with no added random noise. In the first of two different sets of experiments pictures with 5% and 10% added random noise were used as well as the original octree image with no added random noise. When edge detected images with simple boundaries were used, the network responded most favourably with those images that had the same noise levels as these used in the training image set (that is, 5% and 10%). The confidence was enhanced dramatically and in all cases was recorded to be above 95%. One misclassification occurred resulting in an overall average of 97% successful classifications. There were 18 images in the training set and 36 images in the testing set. Similar results were obtained when the neural network was trained with edge detected images with filled boundaries, except that a moderately lower confidence for the pictures with no added random noise was found that can be associated once again to their high ratio of localised noise. In the last set of experiments ADAM was trained with four pictures for each aircraft (a training set of 24 images). Three images were with added random noise (5%, 7% and 10%) and the last one was the original octree image with no added random noise. In this last experiment the neural network showed an almost perfect confidence (100%) for the majority of images while for the pictures (a,b) the percentage in both edge detected simple boundaries pictures and filled boundaries pictures has been clearly above 95%. The only misclassification that has occurred was rather the neural network's inability to select the

closest class pattern from a set of two space vectors that were very close to each other within the 3D vector space.

7.5 Experiments with positional variations.

In this phase of the experimental work, the position invariant classification capabilities of the proposed algorithm with its neural network classifier were tested. The experiments involved both simple skeleton pictures and filled boundaries images. A set of 54 different pictures of aircrafts from different views was taken for each one of two proposed sets of experiments. The 54 different aircraft views were obtained using a camera and rotating manually each one of the six different aircraft models in nine random orientations. In the simple skeleton pictures test, the 54 images from different views were split in groups of three, and then merged to provide ADAM with a set of 18 testing images. Then the resulted 18 images were all corrupted with 5% random added noise. The final set of 36 images was used to train ADAM artificial neural network. In both cases the neural system was trained only with the six original noise-free aircraft volume pictures. Figure 7.30 shows the actual classification results for the simple skeletons test set, while Figure 7.31 shows the corresponding average classification success graph for the same case. Figures 7.32 and 7.33 display the same performance results for the filled boundaries case.



Edge Detected Images (Boundaries Only)
Neural Network trained with non noisy images.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	97.96	96.78	97.90	*	82.45	83.43
CARAVELLE	93.80	97.81	95.15	78.89	83.46	80.86
BEAUFIGTER	99.13	99.82	96.62	84.41	85.34	82.02
SAAB DRAGEL	98.60	99.45	98.83	83.98	84.02	84.50
BRISTOL F2B	94.27	94.14	99.20	80.70	80.55	84.54
F14 TOMCAT	99.52	99.69	98.25	*	83.98	83.89

Non noisy Image Set.

Picture Set with 5% Randomly added noise.

Testing using Images of Random Views.
Two misclassifications occurred.

Figure 7.30. The classification results for the position invariant case with simple skeleton pictures.

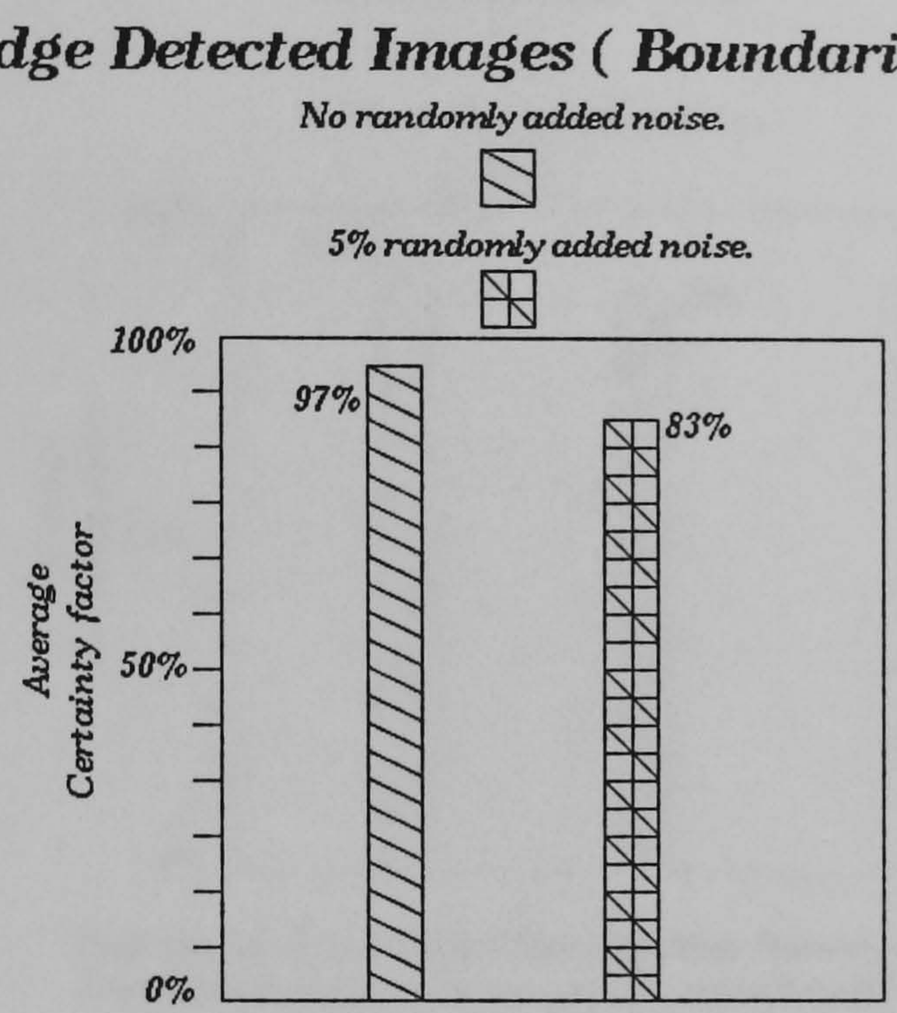


Figure 7.31. The average classification success graph for the position invariant case with simple skeleton pictures.

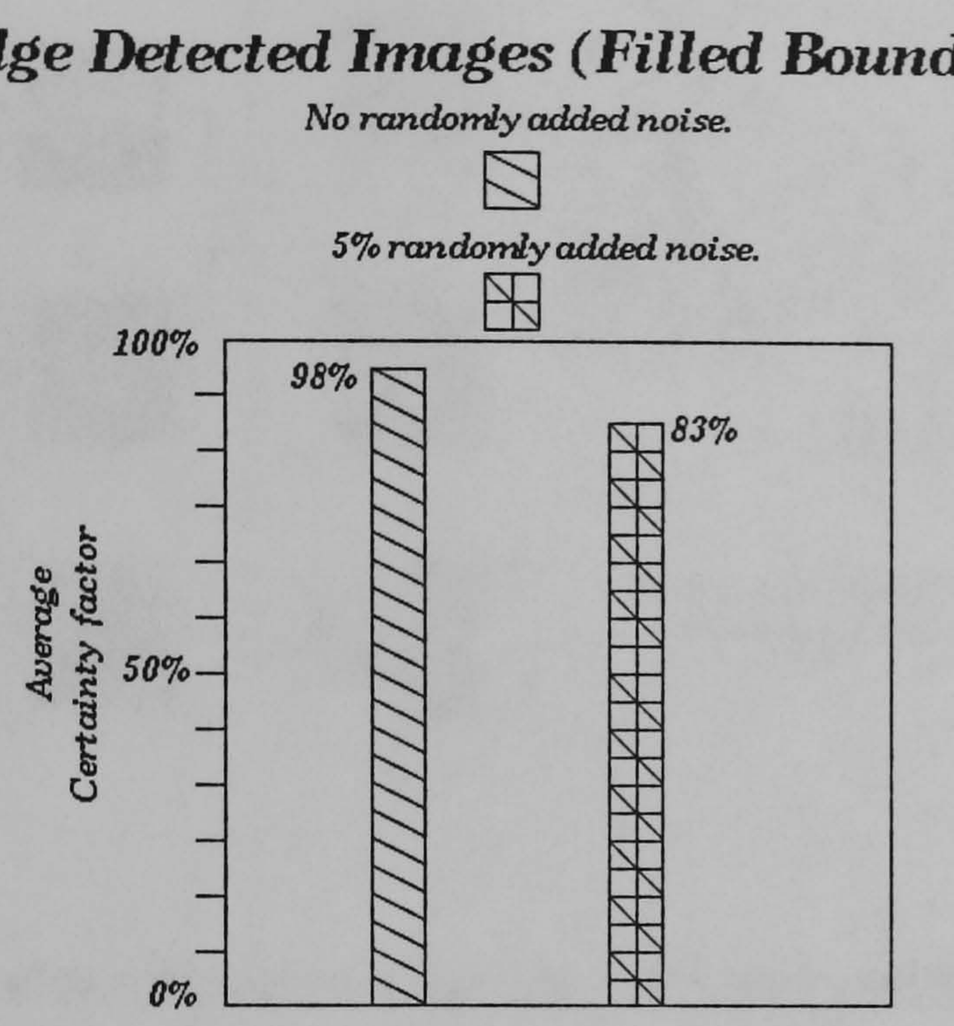
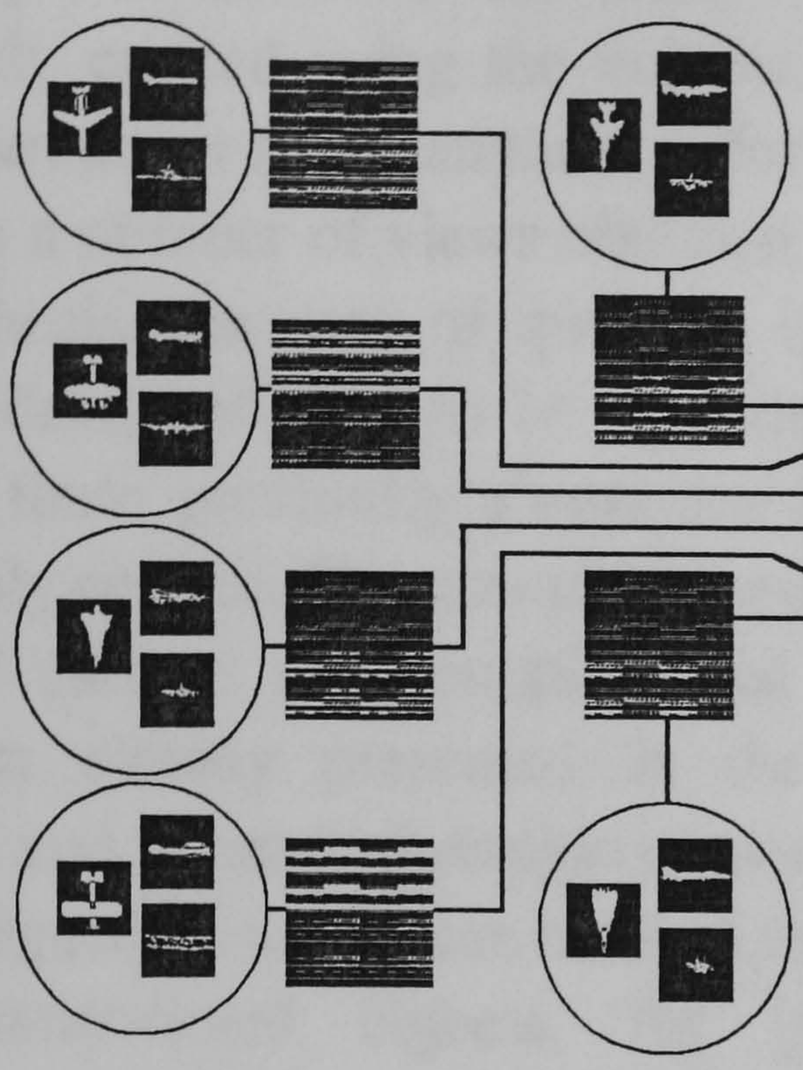


Figure 7.33. The average classification success graph for the position invariant case with filled boundaries pictures.



Edge Detected Images (Filled Boundaries)
Neural Network trained in the absence of noise.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	98.83	97.24	99.09	83.90	83.19	84.66
CARAVELLE	93.73	*	97.28	80.04	*	83.77
BEAUFIGTER	99.11	99.42	98.19	84.78	85.56	84.66
SAAB DRAGEL	99.04	99.44	99.00	85.05	85.82	85.09
BRISTOL F2B	92.79	93.13	99.08	80.02	80.10	84.86
F14 TOMCAT	*	99.64	96.81	*	84.90	83.41

Images with no randomly added noise.

Image Set with 5% randomly added noise.

Testing using Images of Random Views.
Four misclassifications occurred.

Figure 7.32. The classification results for the position invariant case with filled boundaries pictures.

The overall classification of ADAM artificial neural network was satisfactory with only two misclassifications from a set of 36 test patterns in the simple skeletons case and a slightly higher number of four misclassifications in the filled boundaries case. On average the classification success for the non-random added noise half of each experiment has been 98% while for the random added noise part has been 83% in both the simple skeletons and the full boundaries experiments. Having in mind the original test set size of random orientations and sizes pictures (i.e., 54 test views), the results can be easily regarded as highly successful.

7.6 Training ADAM with Quadtree synthetic data.

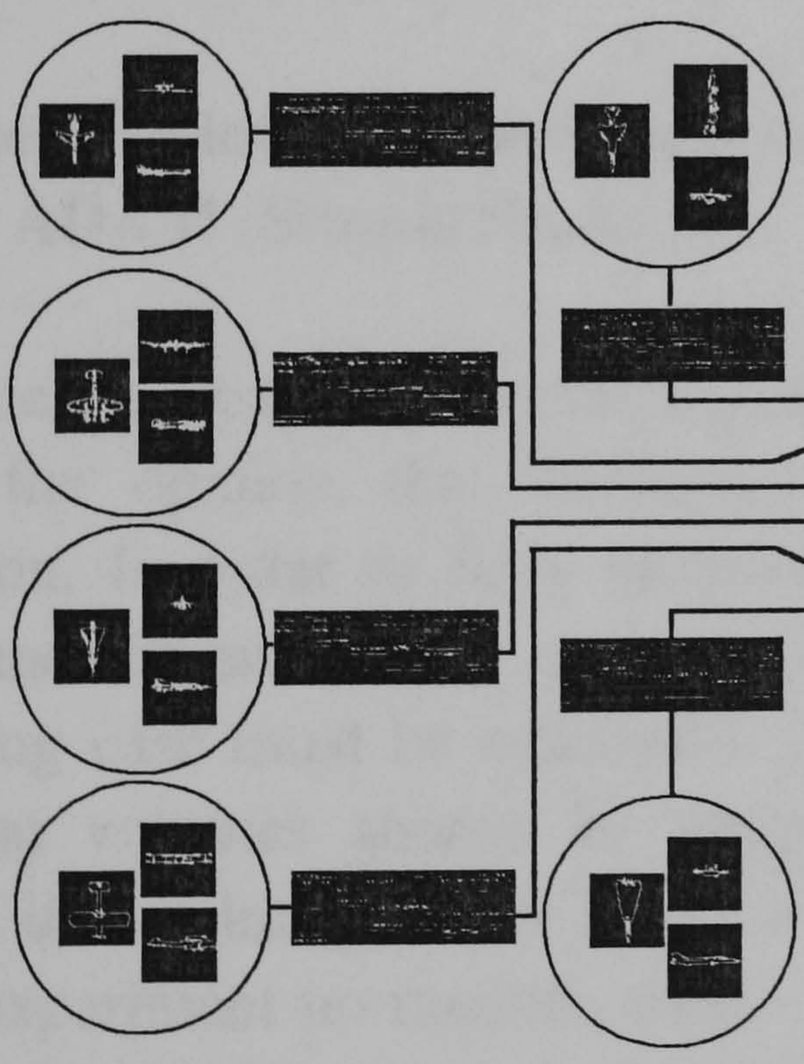
7.6.1 Introduction.

A special situation can arise when the aircraft's volume cannot be effectively created using the volume intersection technique. This case can arise when either less than three different aircraft views are available or there has been a number of views obtained from the same view plane. In this part of the thesis, the case of quadtree data structure application for invariant pattern identification is to be exploited, as an alternative approach to cases such as these previously mentioned where an aircraft's volume cannot be effectively created. The normalised quadtree data structure is the intermediate form of surface representation that is obtained through the use of the algorithm already presented in the previous chapters. The normalised quadtree and its many desirable properties formulate a surface descriptor that, as it is argued by many, can be used instead of octrees in the representation of three dimensional objects, for invariant pattern recognition. Three experiments will be presented that will help in deciding whether any such claims have any validity.

7.6.2 Training ADAM using Quadtrees with various levels of random added noise (Simple Skeletons).

In this experiment ADAM was trained with the combined synthetic pattern that results when the three normalised quadtree images of the three different aircraft views, that will later be used for merging and creating the volume of the aircraft, are combined in one pattern. In this way, six synthetic patterns representing the three different aircrafts were created and used for training the ADAM artificial neural system. Using the same concept, thirty six additional patterns were created for testing. These were split into four categories. The first two patterns in each one aircraft test array, were patterns with no random added noise but with a rather low number of their skeleton bits removed. The later, implies a form of localised noise application which has been either in the order of 5% or 10% of the actual skeleton bits being changed. The remaining four test patterns for each one of the six following aircraft test arrays were consisted by one image with 2% random added

noise, one with 5%, one with 7% and a last image with 10% random added noise. Classification has surprisingly being successful with no misclassification recorded for any of the thirty six different patterns used in the experiment. Figure 7.34 shows a graphical representation of the actual classification results while Figure 7.35 displays a graph of the average classification success for the above experiment. The results so far are impressive and seem to justify the claims made that there is no actual need for performing any volume merging and using octrees. The average confidence shows an impressive 90% rate in its lowest level and once again there has been no misclassifications recorded. Similar results were exactly obtained when filled boundaries were used for training.



Edge Detected Images (Boundaries Only)
NeuralNetwork trained with and without noise.

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	99.15	99.48	97.80	94.55	92.76	90.15
CARAVELLE	96.89	96.97	97.78	94.49	92.72	90.08
BEAUFIGTER	98.85	99.28	97.86	94.60	92.81	90.17
SAAB DRAGEL	99.14	99.33	97.86	94.60	92.83	90.15
BRISTOL F2B	98.78	99.32	97.75	94.47	92.72	90.06
F14 TOMCAT	99.31	99.63	97.84	94.57	92.76	90.10

Non noisy Image Set.

Picture Set with 2%, 5%. 7% and 10% Randomly added noise.

No misclassification occurred.

Figure 7.34. The classification results in the Quadtree training experiment.

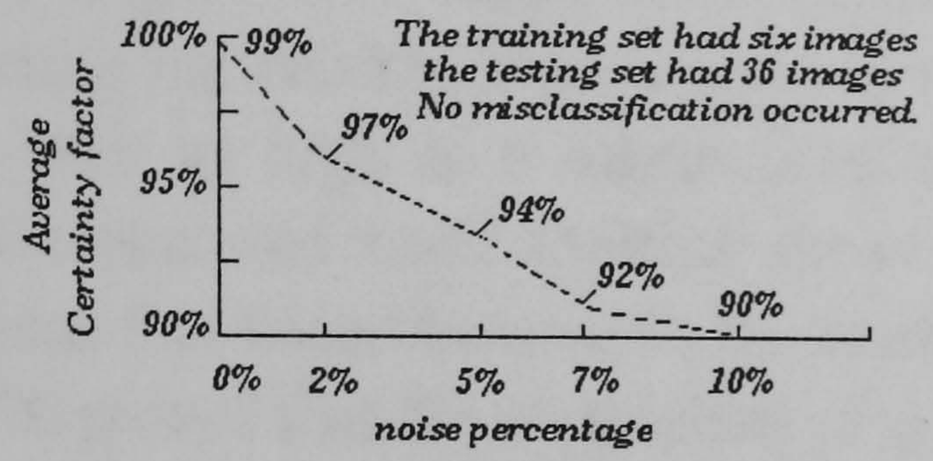


Figure 7.35. The average classification success graph for the Quadtree training experiment.

7.6.3 The position invariant case and the application of Quadtrees for training ADAM (Simple Skeletons).

In the previous experiment almost perfect classification results tend to impose the opinion that quadtrees are more efficient for the current application. In order to fully establish the applicability of quadtrees, their performance in an entirely position invariant and size independent training and testing case must be examined. In the octree case no constrains in the order that volumes should be merged were imposed neither were they imposed in the intermediate normalised quadtree image formation. In the current experiment no random added noise is to be used. The training set of six synthetic quadtree patterns in the immediately above experiment is once again used. For testing, the order of images used to create the training patterns is changed, so as to formulate a test array of six images for each one of the six different aircrafts. Two quadtree synthetic aircraft views out of the three that formulate each training pattern are always left intact but their position is changed within the new testing pattern. One new object view is introduced for each one of the six aircrafts. If the 1,2,3 pattern symbolises

the original training pattern sequence of views for each one of the six different aircrafts, and 4 represents the newly introduced aircraft view to be used for testing, then the testing array for each aircraft is formulated as shown in Figure 7.36. Figure 7.37 shows the classification results obtained after training and testing the ADAM artificial neural network. Although the failure rate has not been as high as it might been expected and only six misclassifications have occurred from a testing set of thirty six images, the actual misclassifications that have occurred in the most general case denoted as 4,3,1 in Figure 7.36 proves that the application of quadtrees is not suitable for a general position invariant case. Clearly, a priori knowledge of the order in which the object views are used to create each testing pattern must be known in order for ADAM to perform successfully. Furthermore in complicated cases involving various levels of noise the quadtree will fail with a rather steeper rate than that witnessed in the noise-free position invariant experiment. To conclude, there are cases, for example when strictly orthogonal object views are used, where the quadtree application will have similar if not better results than the octree case, but in the general position invariant, noise corrupted case the quadtree will fail.

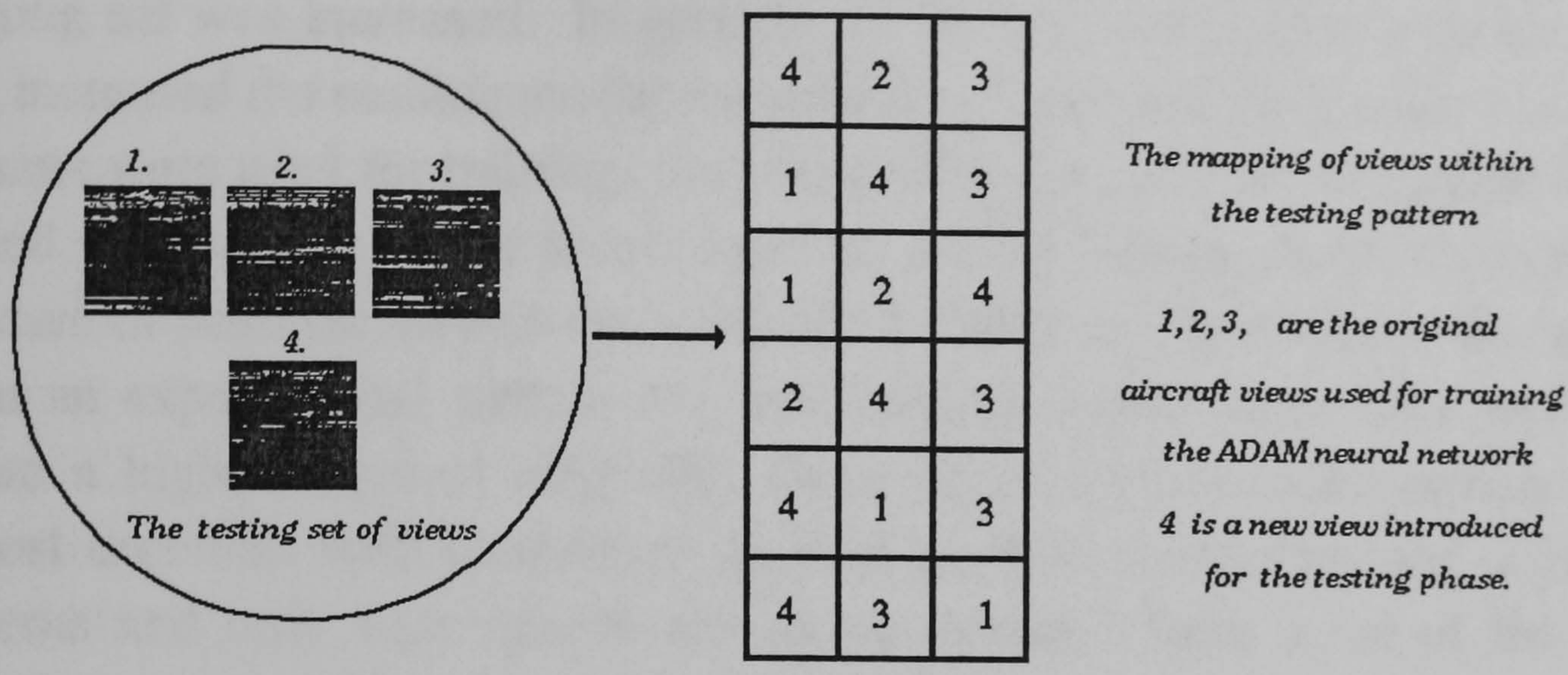


Figure 7.36. The formula used to create each test array for the Quadtree experiment.

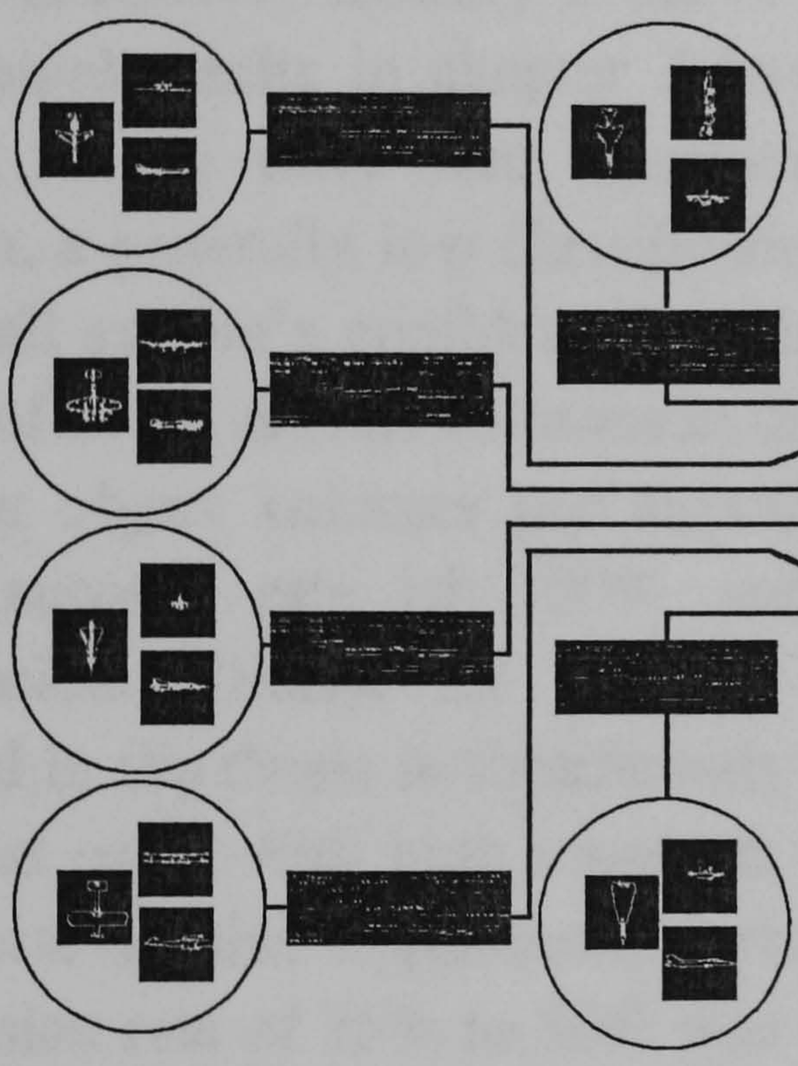
7.7 Conclusions.

Several experiments either with general objects or with simulated aircraft modes were originally presented. Through them, the actual algorithmic implementation could be seen and the exact volume intersection method was able to be verified. A number of experiments using simplified structures were presented and the effectiveness of ADAM classification was witnessed in a variety of cases and under various levels of noisy environments. A further series of experiments were also presented regarding actual aircraft data. In the last case, ADAM neural system was able to successfully recognise and classify aircraft models in a variety of views and with various levels of random additional noise ranging from 2% to 10%. Images were also used for testing that have a considerable number of skeleton bits changed, simulating a form of localised noise corruption. In almost every single case ADAM could successfully identify the tested pattern. The generalisation abilities of the neural system were verified and its

overall classification success soared to 100% in many cases as the size of its training set was increased. In general, as the level of randomly added noise was increased the confidence decreased but as more and more noise corrupted patterns were used for training, the classification success of the neural system soared when similar noise levels were present in patterns used for testing. In the case of position invariance, a set of 54 different object views was used to form an experimental pattern set, and random added noise was applied to create a higher level of difficulty. Once again ADAM's performance was almost excellent with confidence as high as 98% on average for noise-free patterns and only four misclassifications recorded from a set of thirty six tested patterns. When quadtrees were used to train the neural network, ADAM showed a remarkably successful classification performance in a variety of cases but failed to perform successfully in the general position invariant experiment. Obviously, the application of quadtrees can be successful in many cases and especially when orthogonal object views are used to create the final three dimensional volumes, but the required a priori knowledge of the exact order of the views that are to be used for merging, forbids quadtrees to be used in the general position and scale invariant pattern recognition case.

Finally, it is important to notice the similarities and differences in the pattern classification behaviour described in this thesis and compare the overall pattern recognition performance of the proposed invariant three dimensional object identification algorithm to the behaviour of the methods described in chapters 2 and 3. The method proposed in this thesis, based on the properties of the normalised octree data structure together with the ADAM neural classifier, has been shown to possess a set of highly desirable characteristics. It has a simple to understand and implement mathematical form, similar to the mathematical models of both moments invariants and normalised Fourier descriptors. It offers the same high classification success rates as these produced in similar test cases by moments invariants and the normalised Fourier descriptors and has the additional ability to store through

the ADAM's storage property, as many different patterns as they can be stored within a relational database environment. In addition to all the above highly desirable properties, the object volumetric representation introduced in this thesis is proved to be significantly more noise resilient in comparison to moments invariants and the normalised Fourier descriptors. The results in chapter 7 prove that, as the level of random added noise in aircraft test volumes increases so decreases the overall system's classification success rate.



Edge Detected Images (Boundaries Only)
Neural Network trained without noisy .

The Teaching Set of Six Synthetic Images together with the results obtained from the group of 36 Testing Synthetic Images

TEACHING PATTERNS	1st	2nd	3rd	4th	5th	6th
JAGUAR GR1	91.34	92.15	93.48	83.32	82.96	*
CARAVELLE	94.53	91.99	94.19	86.43	86.51	83.12
BEAUFIGTER	89.69	89.20	89.87	*	73.58	*
SAAB DRAGEL	93.38	94.46	95.63	86.66	84.36	*
BRISTOL F2B	89.74	91.51	90.20	83.59	79.90	*
F14 TOMCAT	95.08	91.23	92.55	84.43	86.33	*

Change one out of three quadtree pictures — Change two out of three quadtree pictures

* Change all three quadtree images.

Six misclassifications occurred.

Figure 7.37. Classification results in the position invariant case for the Quadtree experiment.

More importantly though, the total number of misclassifications which is closely related to the overall system's confidence, does not increase dramatically and in the worst case was recorded to be 11% of the overall test set size. This supports the generally expected strong classification performance of ADAM and shows that the method of volumetric object

description together with an efficient neural network classifier can considerably outperform the generally noise dependent behaviour of both moments invariants and the normalised Fourier descriptors. An additional significant advantage of the octree based method is that it generally requires a much smaller training set as this commonly expected in moment invariants and normalised Fourier descriptors cases. In chapter 3, it was shown by Dudani et al. [31] and Persoon et al. [61] that a typical training library set in the order of five hundred to six hundred different object views is required by a system based on moments invariants or normalised Fourier descriptors in order to effectively identify a set of six different aircraft models. Clearly, experimental results in chapter 7 have proved that a single image is only required for an error free classification of six different aircraft types. Although, a generally low classification success rate is obtained in this way, the overall system's confidence can be further enhanced by adding a small number of noisy aircraft volumes in the training set. In this way, a total set of only four object volumes per aircraft is sufficient in order to achieve an overall success rate of 100% and an almost noise resilient pattern classification. Clearly, the size of training object set in the approach described in the thesis is significantly less than the library set size described in the test cases with both moments invariants and the normalised Fourier descriptors. Another important observation is the fact that although an overall compression rate of 75% to 80% was achieved within the relational database system of chapter 2, the method proposed in the thesis that is based on volumetric object descriptions can yield in practise a high rate of almost 99.99% utilising its efficient three dimensional representation of objects. Although high compression rates have not been the main intention of the presented invariant object classification algorithm, they considerably enhance the overall method's image as a large number of two dimensional aircraft views is normally required in aircraft identification experiments and reducing storage requirements offers an added advantage within research environments.

CHAPTER 8

The NeuralBook: A Proposed Graphical Environment for Modelling 3D Pattern Recognition.

8.1 Introduction.

As it has been mentioned in previous chapters, the main contribution of the present thesis is towards the design of a clearly structured and computationally efficient recognition process for the identification of three dimensional objects independent of their space positions and relative sizes. The above statement indicates an application that is closely related to computer vision and digital image processing research interests rather than to research areas such as graphical user-interface design and hypermedia application environments.

The issue of a graphical user-interface came into consideration at a later stage in the thesis as the need for a complete and general view of an general object manipulation, processing and displaying system became apparent. Within such a graphical user interaction environment the entire object identification procedure could be initiated and closely monitored in each one of its different phases. Furthermore, through the current user-interface, classification results could be effectively displayed and efficiently stored for future analysis and modelling purposes. The recent appearance of new human computer interaction metaphores [198], that avoid the use of widely applied pull down selection menus, together with the parallel significant growth in the use of hypermedia type graphical user interaction systems [199], have decisively influenced the final decision to implement a hypermedia style user-interface responsible for the modelling of the underlying pattern classification procedure.

This chapter aims to present the proposed user-interface design process, placing particular emphasis in the categorisation of the general user-interaction procedure in four distinctly separate but bilaterally interacting phases. Each one of these phases will be individually presented and the underlying discussion will focus on their relative design advantages and possible limitations.

8.2 The NeuralBook User-Interface

8.2.1 Introduction

Based on the locally implemented user-interface model, the Book Emulator [7], a proposed user-interface to neural network applications for pattern recognition, the NeuralBook, was designed. The NeuralBook's implementation process can be classified into four main parts. The input phase, the image pre-processing phase, the pattern generation phase, and finally the neural network training and testing phase. Each one of the above categories is itself sub-divided in a number of sub-phases. The input phase can be further divided in two parts, according to whether the input aircraft's picture name is typed in or selected from an appropriate name list of already stored aircraft models. An alternative approach in selecting a new aircraft pattern for the input phase is by transferring the actual aircraft picture from an already existing book of aircraft pictures to the NeuralBook. In the latter case, the actual aircraft's image denotes the image pattern to be used for further processing within the NeuralBook. This method is clearly closer to the natural approach of selecting patterns and requires no memorisation of individual aircraft names. The image pre-processing phase can be structured in a variety of ways depending on the specific sequence of digital image processing operations the user wishes to perform on the selected image that result from the previous input selection phase. Typically, a sequence of image enhancement operations followed by gaussian smoothing, edge

detection, edge map enhancement, histogram thresholding and spurious noise peaks removal can be encountered. Each one of the previously mentioned image processing operations can be separately viewed in subsequent NeuralBook pages, where the actual image processing results on the selected aircraft image can be displayed together with various time performance and image related statistics. A histogram presentation can also be additionally displayed, whenever such presentation is meaningful, to allow a complete view of gray scale intensities distribution within a processed image and provide a clear picture of the performance efficiency of digital image processing procedures that were applied on the selected aircraft picture. The pattern generation phase depends on the selected neural network model as well as on the specific pre-processing of neural network patterns intented to be used. In the present application, the pattern generation phase is further divided into three sub-phases. The non-coplanar aircraft views selection process with its corresponding aircraft pseudo volume image generations, the selection of an octree/quadtree algorithm for training/testing the underlying neural network architecture, and finally the actual volume merging phase with the generation of the training/testing synthetic image pattern and the numerous time performance and picture related statistical information. The last of the main phases, regarding the training and testing of neural networks, is somehow straightforward in its content and presentation. It is split into three parts: the training phase, the testing phase and finally the performance graph phase. The training phase involves typically the introduction of the training patterns with the association of a specific class pattern number for each one of the patterns that are to be used for training. The testing phase, simply displays the pattern that was used for testing, its classification success rate and the class pattern vector that was retrieved. The performance graph sub-phase is optional and it is in this phase where appropriate performance tables can be drawn and stored within the NeuralBook, providing a useful future reference of the performance of specific experiments.

8.2.2 The input phase.

Figure 8.1 presents the part of NeuralBook responsible for the input phase. New objects may be entered into the system simply by typing the name of the binary file containing the object's raster data. If the object is already in the underlying file store, a simple click with the mouse within the box area provided in the far right part of the right page is required. This concept in clearly indicated by the pointing finger images, pointing at the available boxes. In the present design, provisions are taken so that the system can facilitate a maximum of 150 pictures representing aircrafts of various types with a standard resolution of 64 by 64 pixels. This resolution does not restrict the actual resolution of the input pictures, it rather provides a standard fixed resolution for all images displayed within the NeuralBook pattern selection phases. The above input procedure implies that users should bear in mind the exact file name of each individual new aircraft pattern, that might be introduced for training and testing within the NeuralBook graphical environment. Clearly, although such an ability can be expected by users, it can often lead to numerous wrong entries during the input phase as users tend to forget the precise file names among the plethora of data files in various directories. A suitable input procedure that can overcome this limitation can be achieved with the introduction of the "book to book" communication process. It is often preferred that images and information related to them, are kept all together in the form of an IndexBook. Figure 8.2, displays a view of such a possible IndexBook. A procedure whereby users can move an aircraft's picture from an IndexBook into the NeuralBook so as specifying the aircraft pattern to be used for further processing, can be used to implement the "book to book" communication concept. Such mechanism although not described here in great detail, can after all be effectively implemented using the current state of the Book Emulator user-interface with the introduction of book shelves. When an aircraft's image is selected from the IndexBook, the corresponding aircraft's data, gray scale intensity histogram, and binary raster bitmap are automatically made available for transfer into the NeuralBook. Bringing NeuralBook into the foreground, moving into the first empty page within the image processing section and

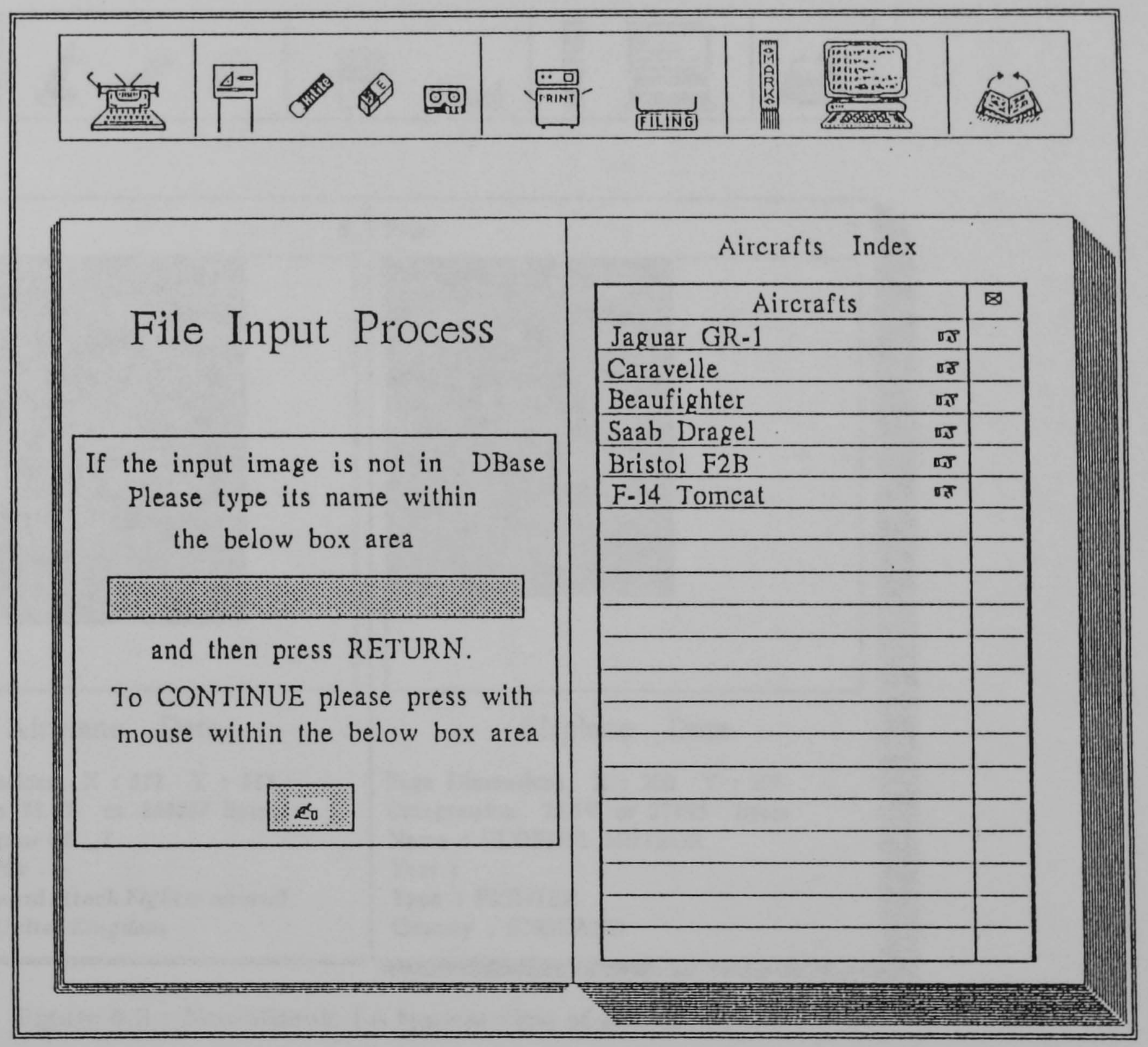
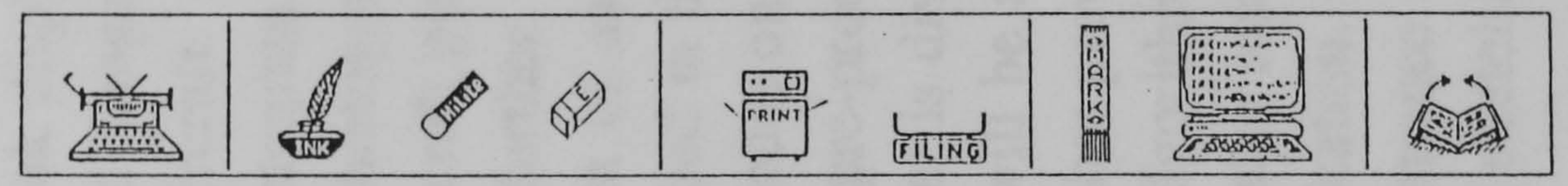
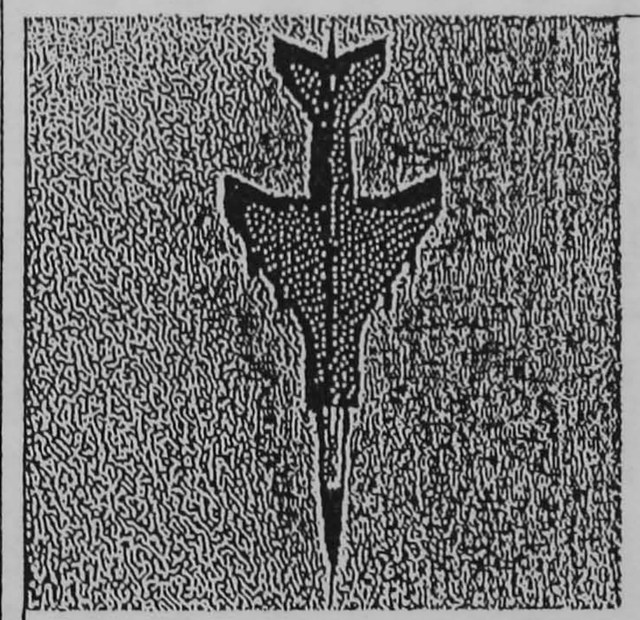


Figure 8.1 : NeuralBook [The standard input phase]



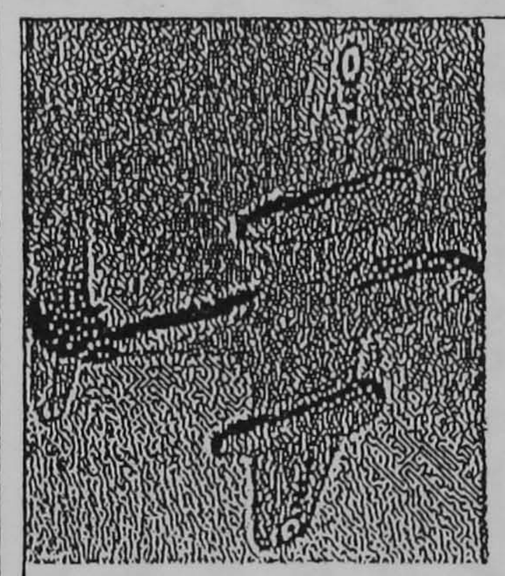
Page 6



Airplane Data

Page Dimensions X : *512* Y : *512*
Compression *78.7%* or *206307* Bytes
Name : *Jaguar GR_1*
Year : *1970s*
Type : *Ground Attack Fighter aircraft*
Country : *United Kingdom*

Page 7



Airplane Data

Page Dimensions X : 200 Y : 235
Compression 76.5% or 27495 Bytes
Name : GLOSTER METEOR
Year :
Type : FIGHTER
Country : ENGLAND

Figure 8.2 : NeuralBook [A typical view of an index book]

then using the mouse, would in fact restore all the IndexBook's selected aircraft related information into that page [Figure 8.4].

8.2.3 The image processing phase.

After an aircraft pattern has been selected, the NeuralBook displays its contents as shown in Figure 8.5. On the left page an enlarged image of the selected aircraft is displayed together with various aircraft related information. At the lower left corner in the same page a histogram of the aircraft's grey scale intensities is additionally shown. The histogram can provide meaningful information as to whether the original aircraft picture is noisy and therefore might require the application of further image enhancement techniques. The right hand page includes a list of available image pre-processing algorithms that can be selected and applied in the pre-processing of the original aircraft pictures. Clicking with the mouse on any of the far right boxes on the right page, defines the image pre-processing algorithm to be applied. When clicking within a box, a number is displayed indicating the order of pre-processing (i.e., which algorithm will be applied first). Clicking within an already selected box causes the number previously displayed within the box to disappear thus implying that the algorithm is no longer desired. Figures 8.6 and 8.7 show the image pre-processing results, as they are displayed within the in the present NeuralBook application. On the left page of Figure 8.6, the edge detection results and performance related statistics are shown, while on the right page the result's after stretching the edge map histogram can also be seen.

The use of histograms in both pages and the various numerical and image related data, can clearly help to create a complete picture of each specific step in the pre-processing phase of the selected aircraft. Finally, on the left page in Figure 8.7, the image obtained after histogram thresholding is displayed, while on the right page the final aircraft binary image is presented after "salt-and-paper" noise removal. Performance statistics and image

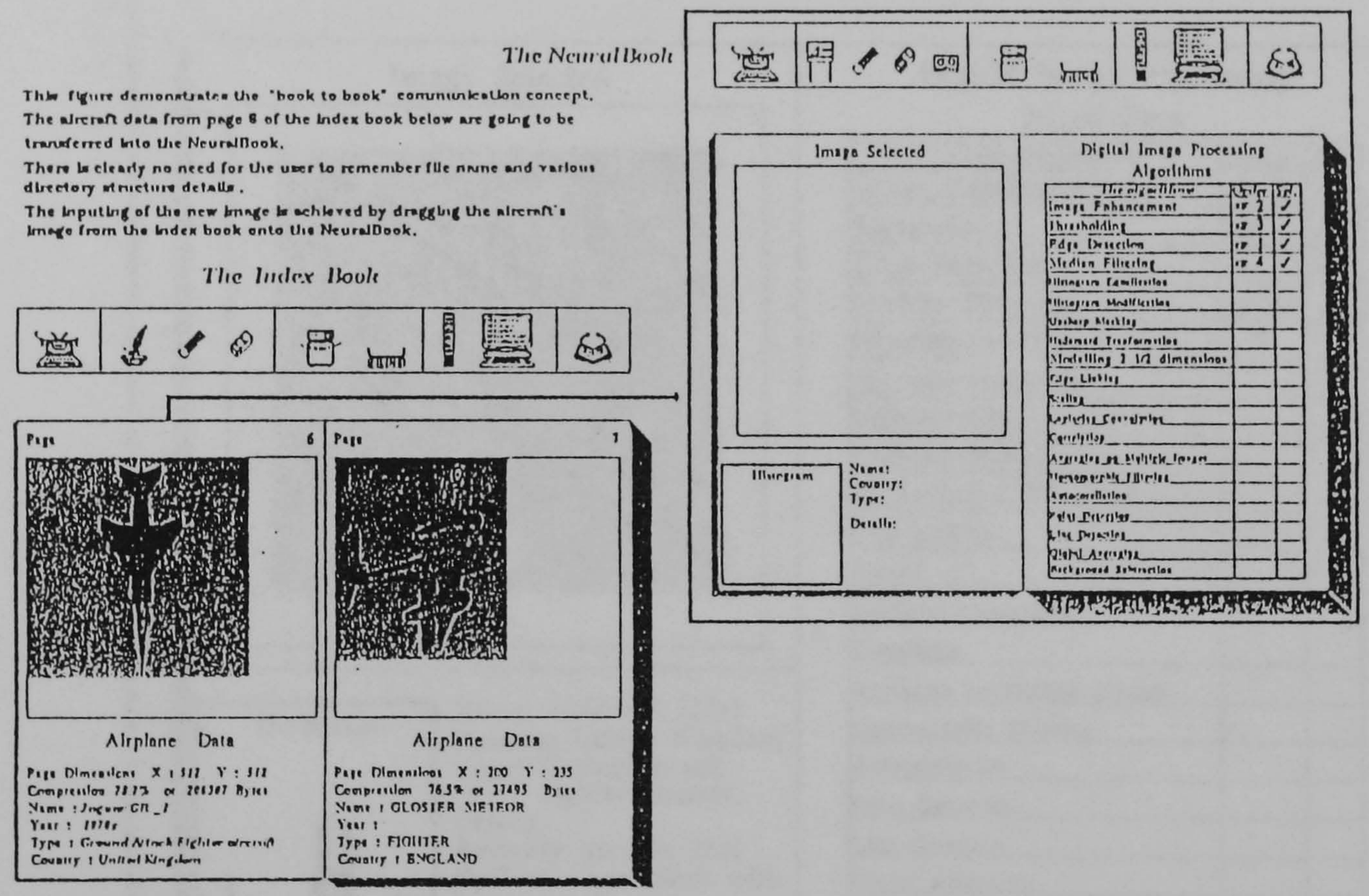


Figure 8.4. The NeuralBook [The concept of 'Book to Book' communication for input].

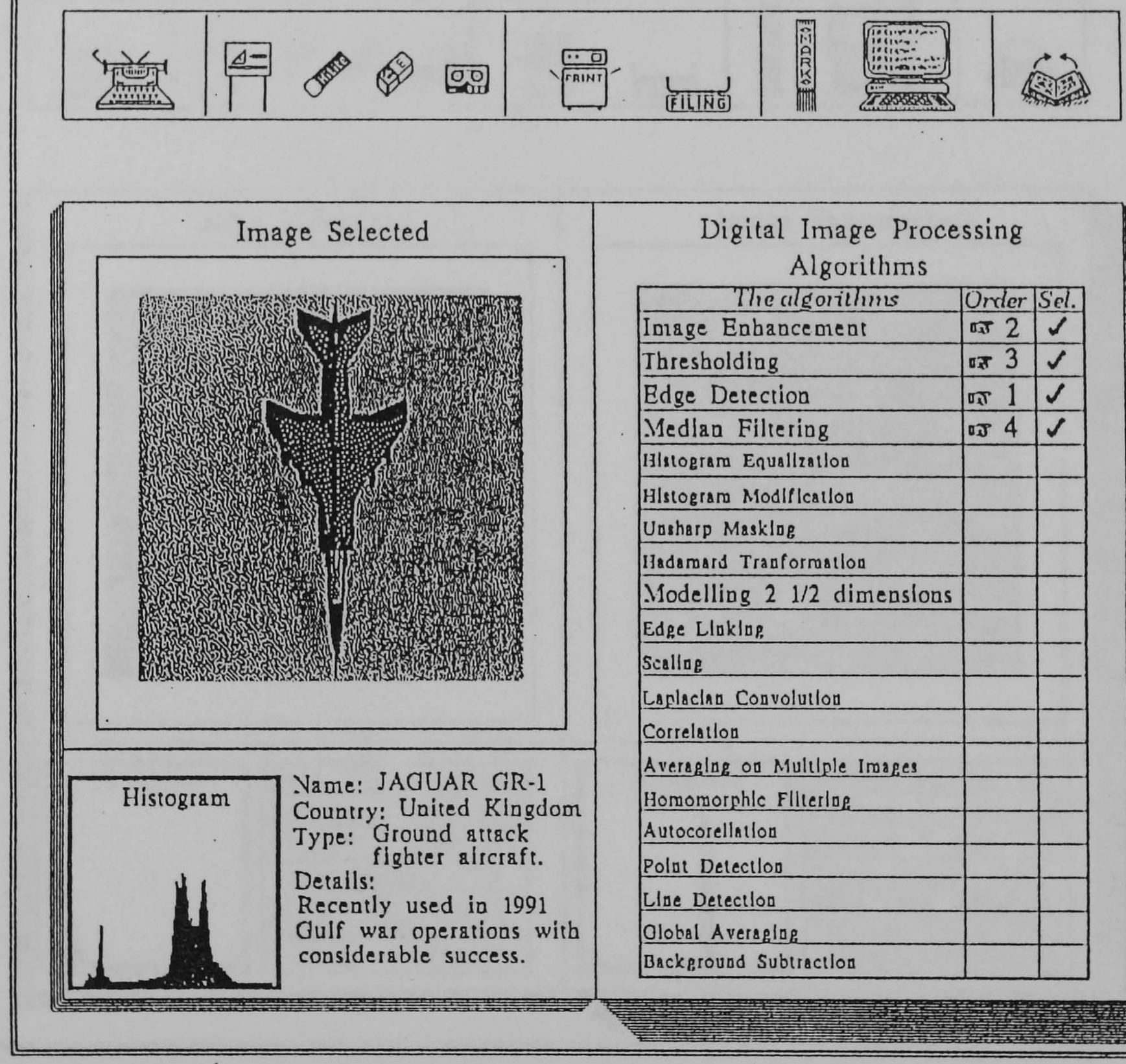


Figure 8.5 : NeuralBook [The initial view within the Image Pre-processing Phase]

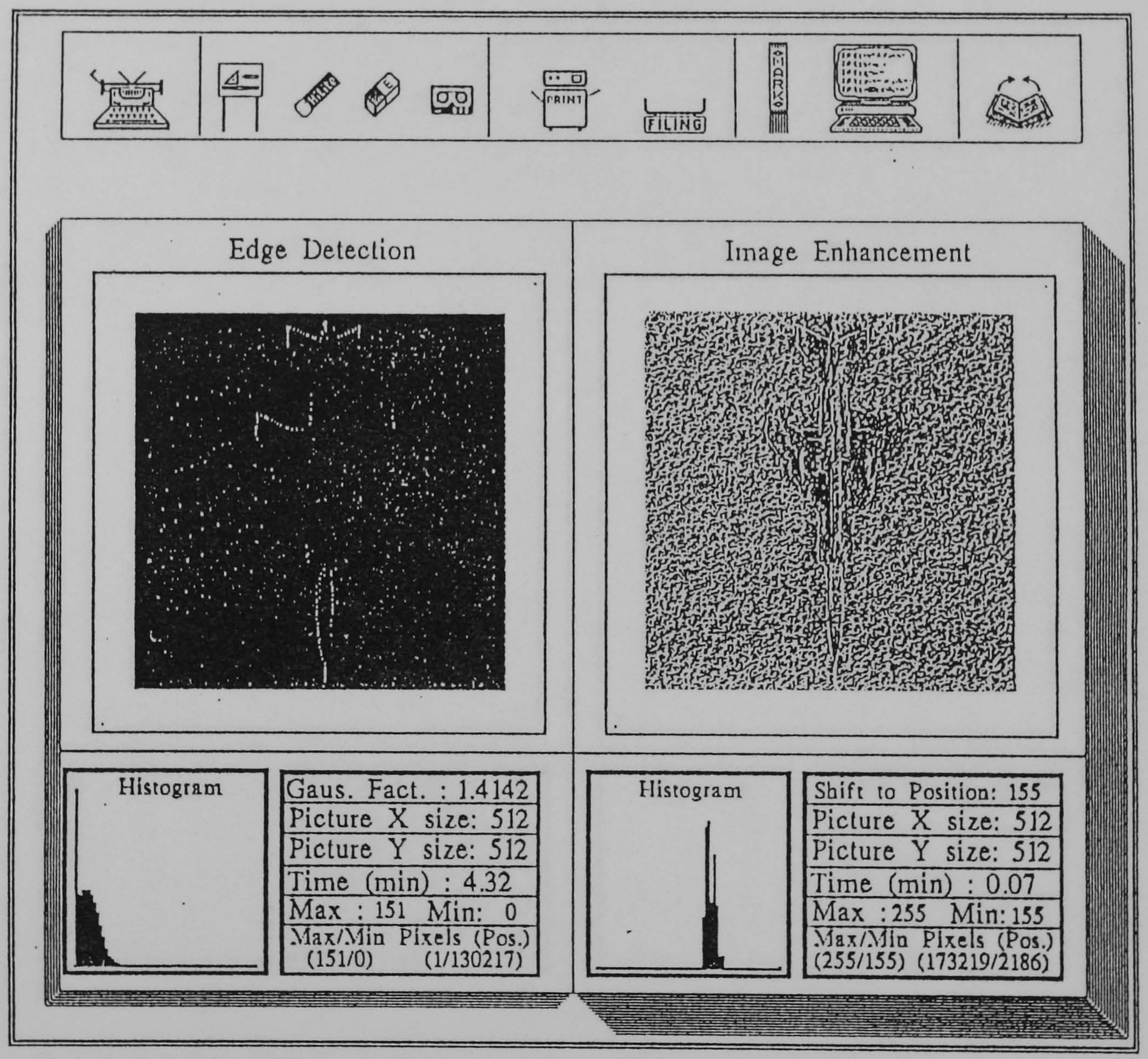


Figure 8.6 : NeuralBook [Additional views within the Image Pre-processing Phase]

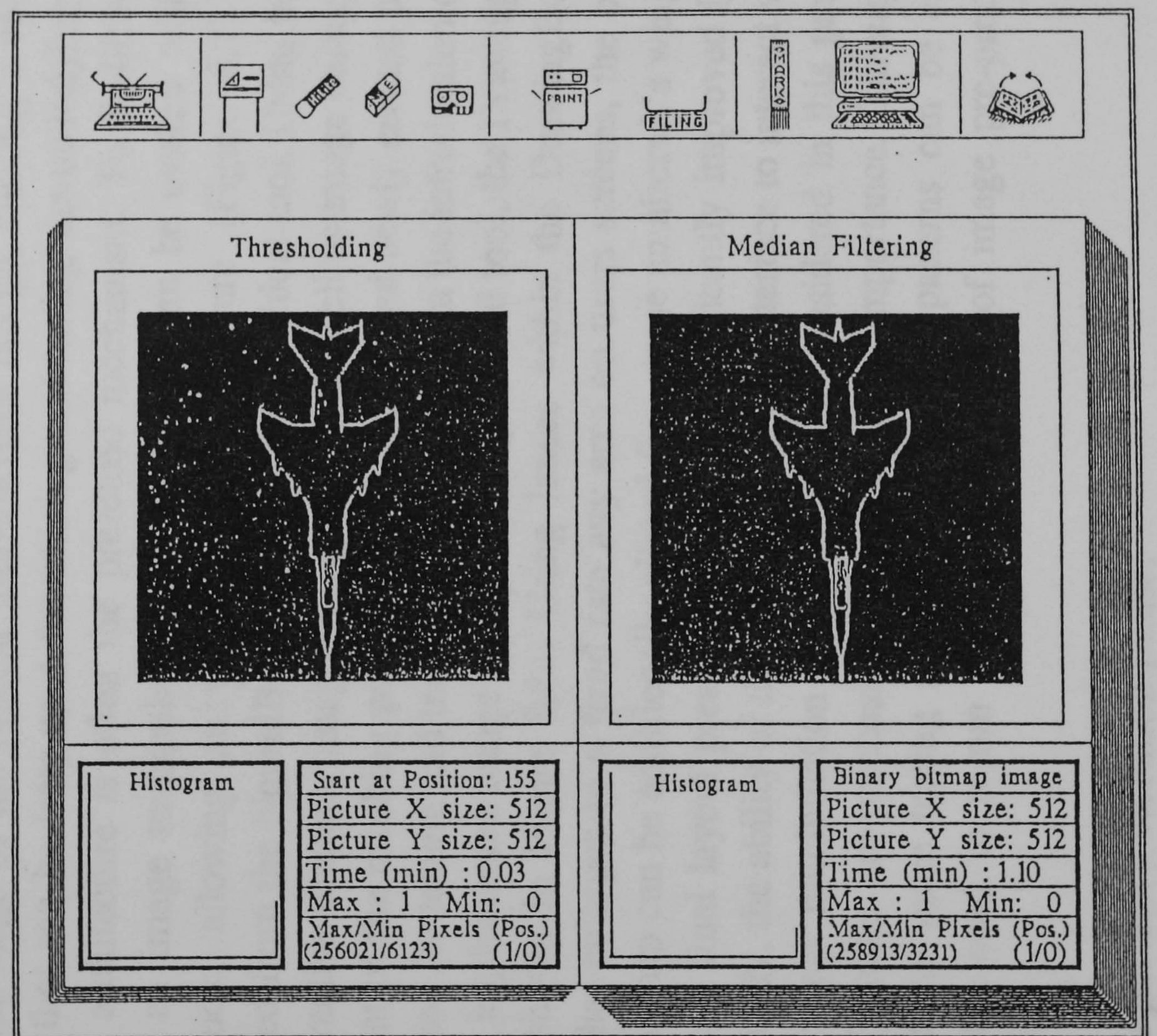


Figure 8.7 : NeuralBook [Additional views within the Image Pre-processing Phase]

related data are once again provided, although clearly the use of histograms has now been limited, as the process is applied on binary data rather than raster images.

In this stage of the present user-interface design, creating a variety of patterns that can be later used for training and testing the underlying neural network architecture is often the preferred mechanism. To facilitate such requests an image manipulation graphical tool can be evoked within the NeuralBook, allowing an aircraft's binary picture (Figure 8.7) to be transferred from the NeuralBook and onto the graphical tool. Using facilities like interactive rotation, image size reduction as well as image enlargement, the format of the resulting pattern can be altered and newly created patterns can be used for testing and training. The results from the introduction of such a locally implemented image manipulation graphical tool, the DrawingBoard [8] are shown in Figure 8.8a. Using lenses within the DrawingBoard, a facility that enlarges to a fixed rate any area on users screens, the object's final edge map can be additionally altered (i.e., erase an aircraft's wing), and the object's final layout presentation can be significantly improved [Figure 8.8b]. Clearly, the ability of the NeuralBook user-interface to interact with an external image manipulation graphical tool, is considered in this particular phase of image pre-processing of considerable importance. Through this facility many more testing and training aircraft patterns can be created, without the need to re-run the entire sequence of image pre-processing algorithms.

8.2.4 The pattern generation phase.

The pattern generation phase in NeuralBook varies according to the intented application of the underlying artificial neural network system. In this thesis, the method that has been applied in order to create the patterns required for training and testing the ADAM artificial neural system was presented to a great extent in the previous chapters. When the image pre-

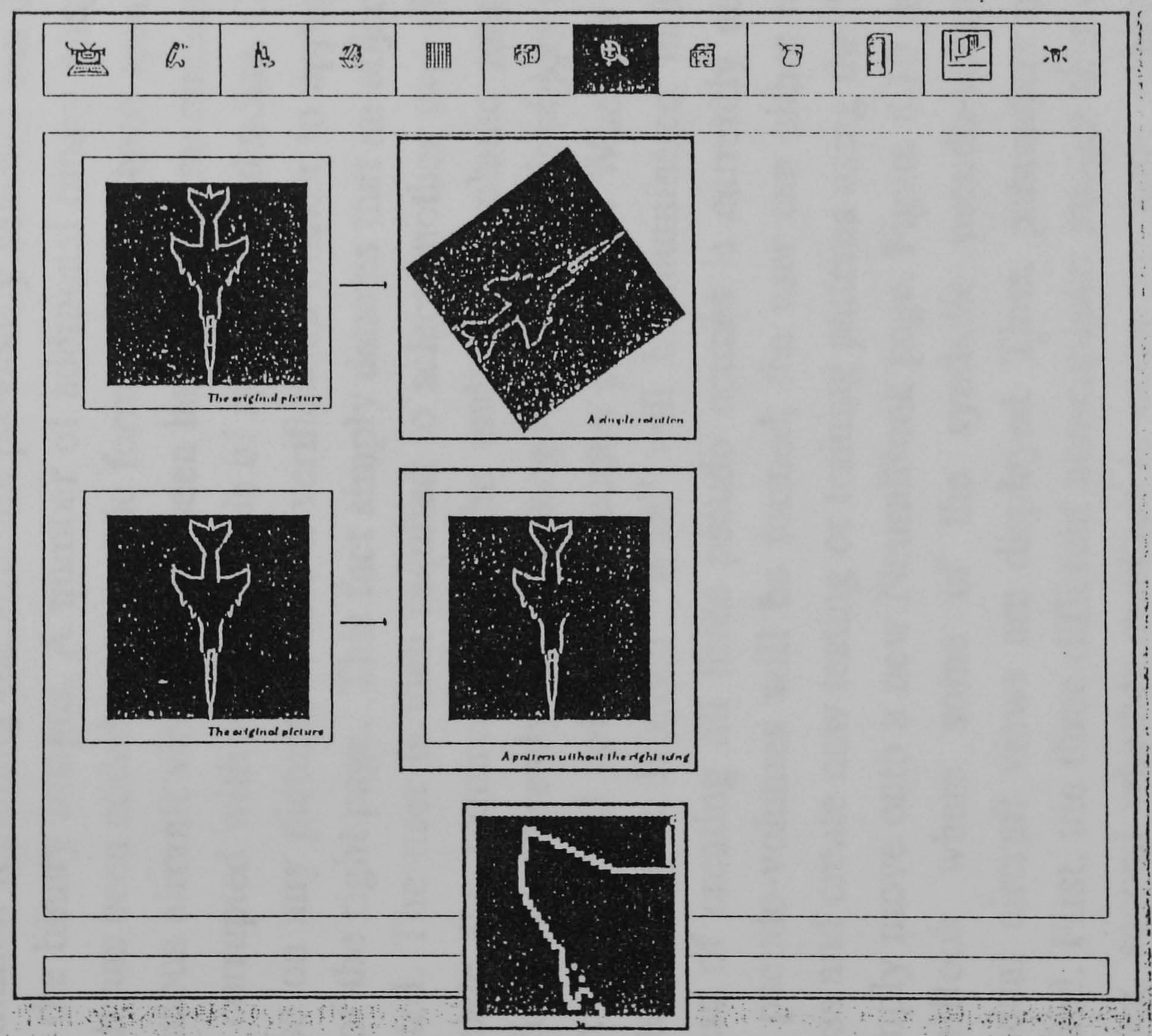


Figure 8.8. The NeuralBook [Picture manipulation using an external graphical tool (DrawingBoard)].

processing phase is completed, NeuralBook displays the picture of Figure 8.9. The graphical environment from which three non-coplanar images of an object are required to be selected is shown. The aim once again is to create the pseudo-octrees specified by the volume intersection algorithm and use the resulting volume image to train and test ADAM. On the left page, information regarding the aircrafts previously selected can be seen, together with the binary images. A number of additional binary images, each one of which has been coded in a specific format can also been seen. Actual pictures of various aircraft views can be seen having either no random added noise or been corrupted with various levels of random added noise. Clicking with the mouse on any picture, causes a verification marker to appear at the bottom part of the right page. This fact simply denotes that an object view has been selected. The user is then required to select another two views in order to complete the requirements of the underlying volume merging algorithm. Clicking on an already selected image will delete this object view from being a part of the final volume intersection procedure. When all three views are selected the box labelled "Wait" will be illuminated indicating that the process of creating all three pseudo octrees is currently active. When the three pseudo-volumes will be formed, the user can either repeat the entire process and create new testing or training patterns using the "Repeat" button, or simply move onto a new NeuralBook page. Figure 8.10 shows the part of NeuralBook where some of the synthetic pseudo-volume pictures of individual aircraft views are displayed. Three parameters need now to be specified. First, the three different non-coplanar images that will be used for merging, second whether quadtree or octree synthetic image data are to be applied and third, whether the neural network system will be trained or tested with the resulted 3D object volume. The method of selection is once again by clicking with the mouse on the labelled areas provided on the bottom part of the right page. Double clicking will cancel any previous selections. Attention is now focused on the graphical environment displayed in Figure 8.11. The previously selected three pseudo-volume images can be seen. This simple illustration indicates that the process of merging the three pseudo-volumes is

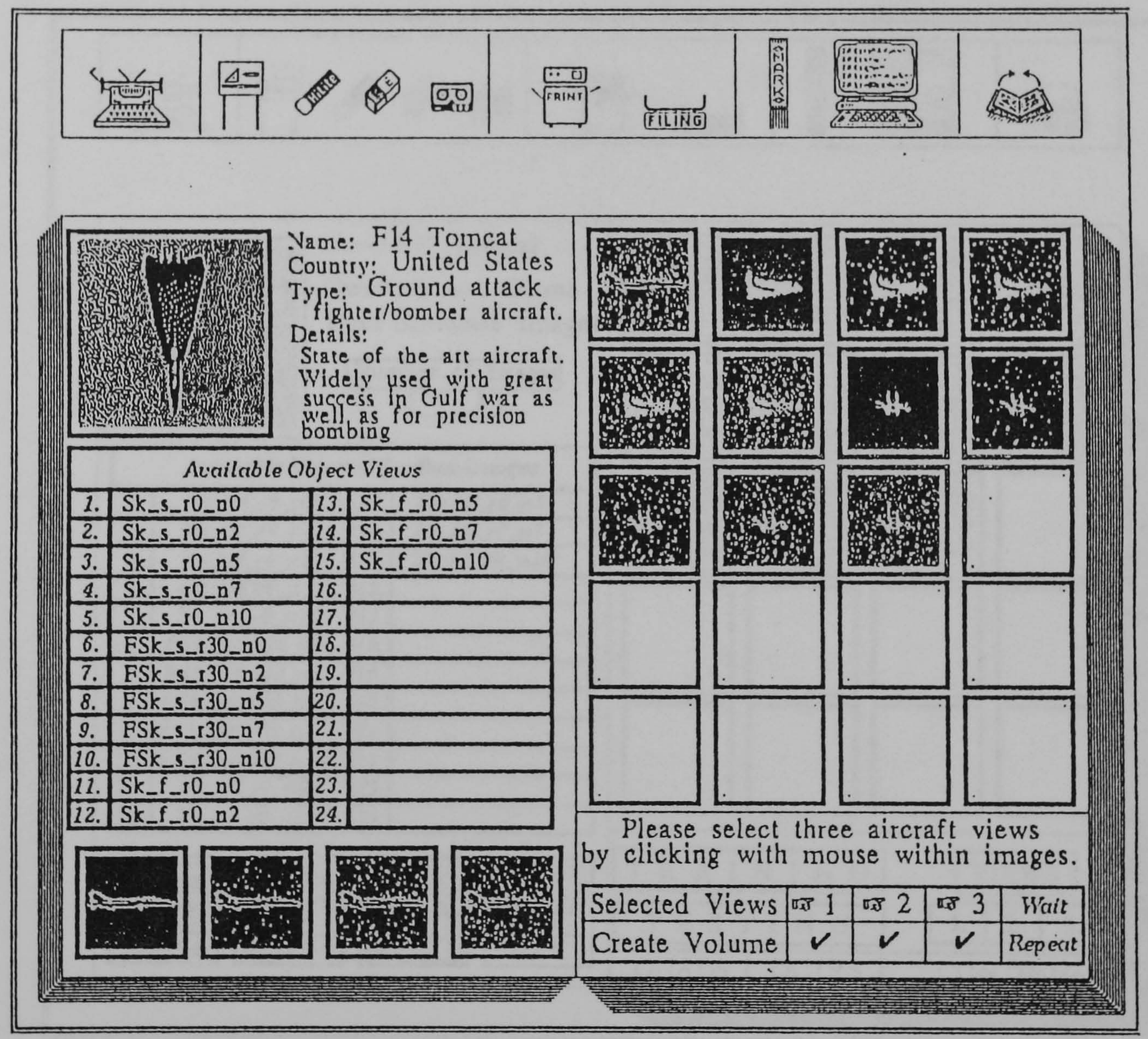


Figure 8.9.
NeuralBook
[Selecting the three non - coplanar views within the pattern generation phase]

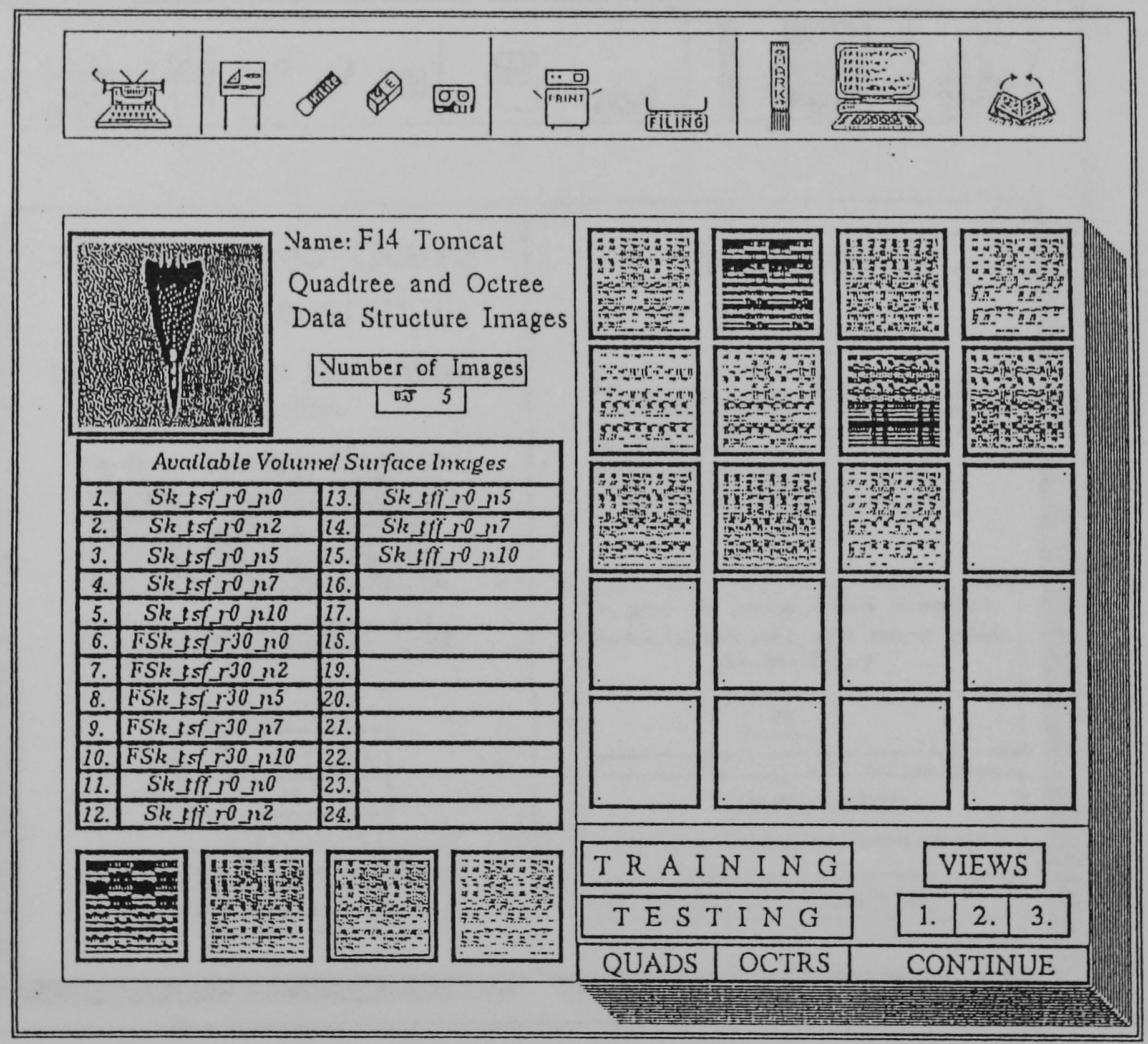


Figure 8.10.

NeuralBook

[Selecting the synthetic non - coplanar views within the pattern generation phase]

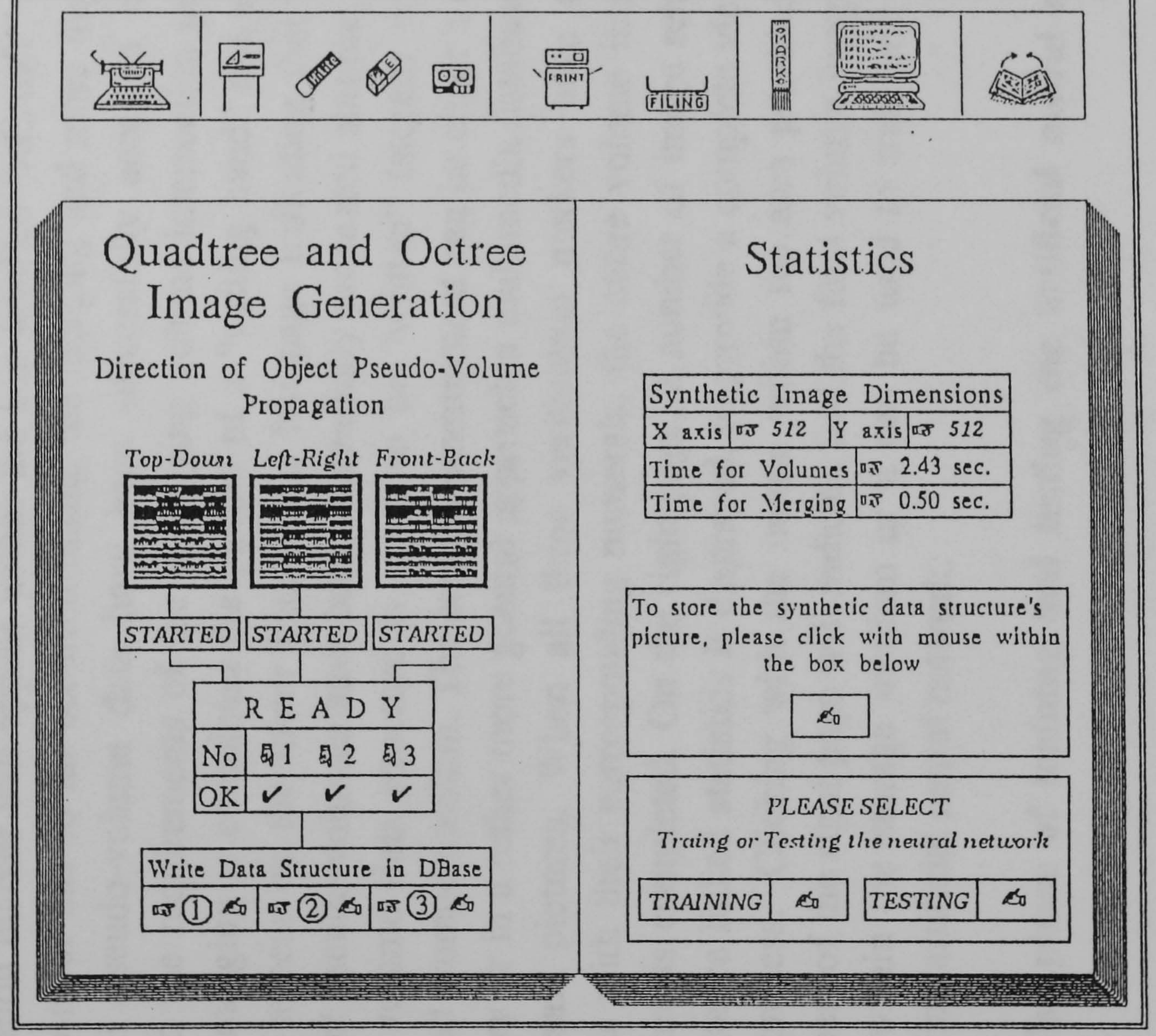


Figure 8.11.

NeuralBook

[The Volume Intersection technique within the pattern generation phase]

about to be initiated. In fact, the labelled boxes below each one of the three pseudo-volumes indicate whether the processing of each individual synthetic image has started. When all three synthetic images are being processed a verification marker will appear within the box labelled "READY" denoting that no error has so far occurred during the merging and more importantly that all pseudo-volume data have been successfully accessed and have fulfilled the requirements of the underlying volume intersection procedure. When merging is completed the picture of a "writing hand" and a number should appear on the lower part of the left page indicating that all three different pseudo-volumes have been successfully processed and that the final object volume data structure is ready to be "written" (stored) within the provided storage medium. The storage environment can be either a system's directory or in a rather more general approach a multimedia database system of aircraft pictures. When all three verification markers have appeared together with their corresponding numerals, the entire volume intersection procedure is completed. On the right page, a number of image related and performance linked statistics is displayed, to provide a complete view of the entire process. Clicking with the mouse within the area provided on the lower part of the right page will indicate that the final volume image can be stored within the storage medium or it can be used to train and test the underlying artificial neural network.

8.2.5 The phase of training and testing the artificial neural network system.

If training the neural system with the resulting image of the previous step is required, an additional number for each different class pattern must be provided. In the case of ADAM, a class vector with a size of 16 bits has been selected together with an N tuple size of 4 and a number of bits set in each class vector of 4. Briefly, this fact implies that the numbering of each different object used for recognition should be obtained by setting three bits to 1 from a numerical string of 16 bits. Figure 8.12 demonstrates the

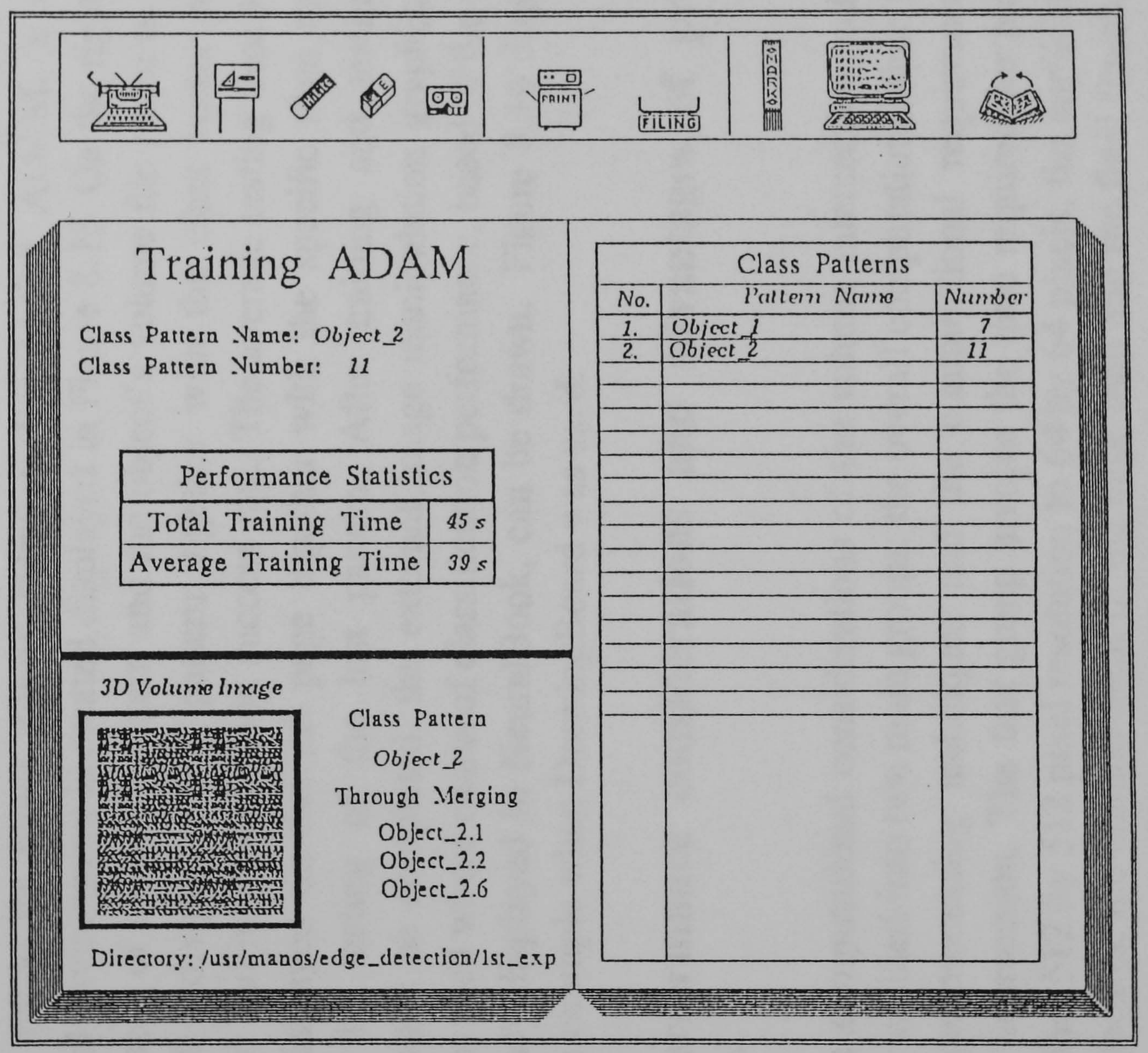


Figure 8.12.
NeuralBook
[Training the underlying ADAM neural network system]

graphical environment for the above operation and displays the additional information regarding the class vectors and their corresponding file names. When teaching is completed, the required computational time is displayed on the left page together with the average teaching time for all taught patterns up to that particular phase during training. For testing ADAM, a similar graphical environment is created displayed in Figure 8.13. On the right page, the number of the class pattern and the object's volume file name is shown for each individual tested pattern together with the class vector number retrieved and its classification success rate. The average testing time is once again displayed on the left page together with the specific time required during the testing of the last pattern. After training and testing has successfully be completed, the external image manipulation graphical tool can be evoked where charts of classification performance, based of the results previously displayed in NeuralBook, can be drawn. Figure 8.14 displays a possible example when DrawingBoard was used.

8.3 Performance considerations and advantages of present design.

The computational considerations of the volume intersection technique can be classified into two main groups: the overall computational time for the image pre-processing techniques and the computational requirements of volume intersection. The first group involve the time required for reducing the original 512 by 512 pixel resolution to 64 by 64 pixels, the application of edge detection, the enhancement of the edge map and the final thresholding operation. Figure 8.15(a) shows the average times required for the first group, while Figure 8.15(b) displays the time required for the volume intersection technique. In the latter, time was on the order of 1.8 seconds for the quadtree generation and 2.1 seconds for the octrees generation. As the volume intersection technique does not require that all quadtree structures should be available from the beginning, the process can be executed in parallel with each subsequent image being processed as it becomes available. The

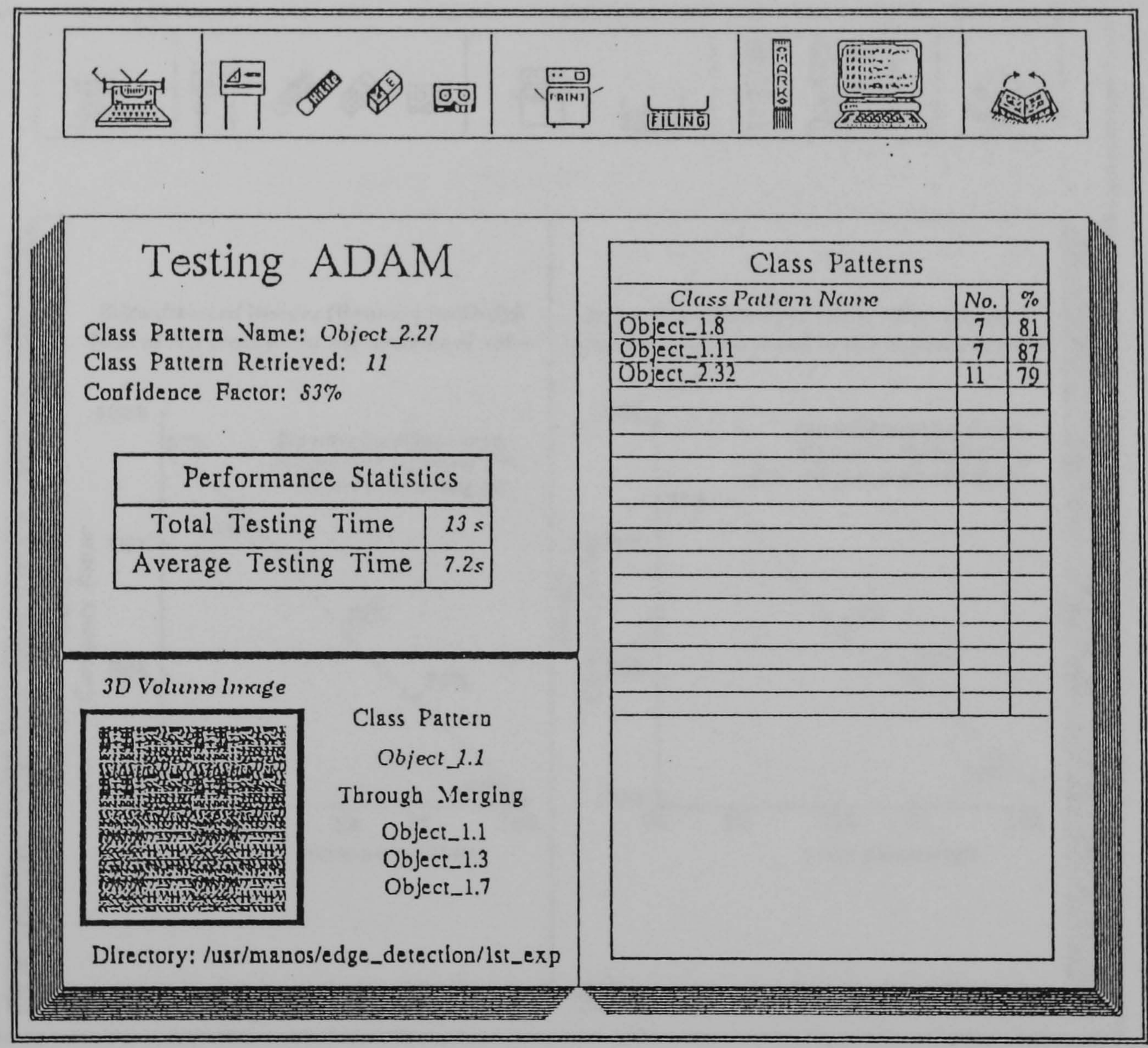
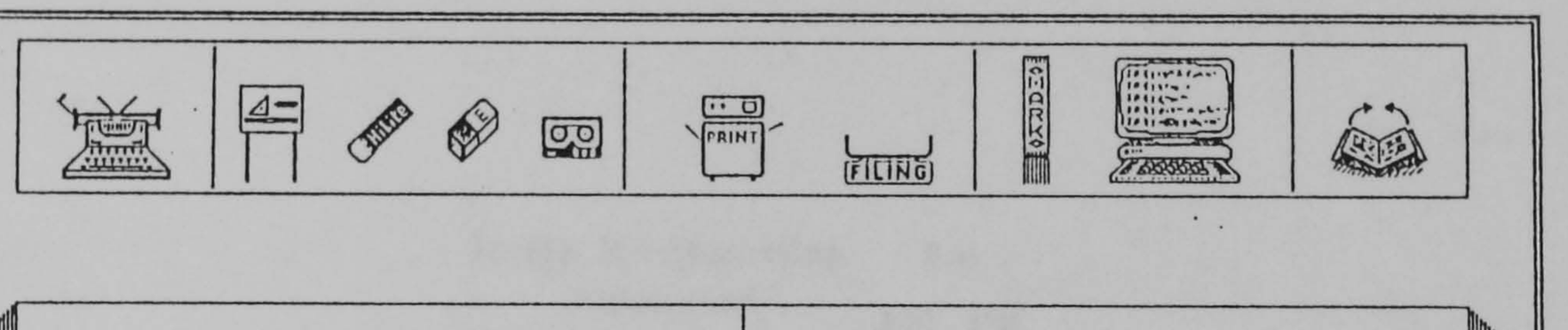


Figure 8.13.

NeuralBook

[Testing the underlying ADAM artificial neural network system]



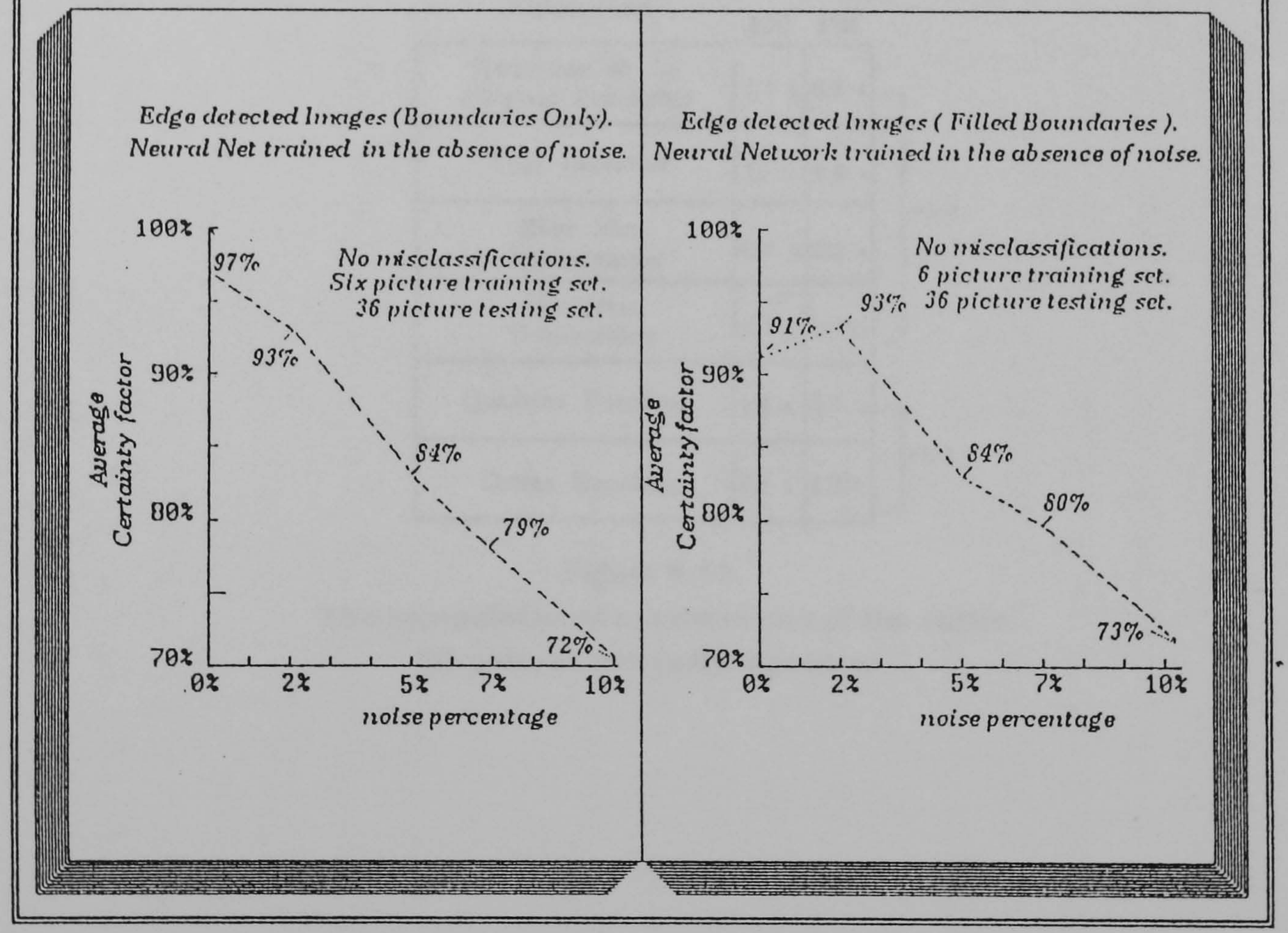


Figure 8.14.

NeuralBook

[Creating performance charts using the DrawingBoard facility]

Image Pre-processing Algorithms	Sun 3/50	Sun 3/60	
Reduction of the Original Resolution	1.2 s	0.6 s	(a)
Edge Detection	20 s	9.8 s	
Edge Map Enhancement	0.4 s	0.3 s	
Histogram Thresholding	0.3 s	0.25s	
Quadtree Encoding	1.8 s	0.7 s	(b)
Octree Encoding	2.1 s	1.05s	

Figure 8.15.
The computational requirements of the entire 3D pattern recognition process.

maximum time obtained in the experiments has been 3.9 seconds for the volume intersection method. All time measurements regarding the experiments were obtained using a Sun 3/50 workstation, and the total time for the pre-processing operations totaled 25.8 seconds. Relative time duration for the important computationally expensive edge detection operation on a Sun 3/60 showed an overall edge detection time down by 49 to 57% at around 9.8 seconds and consecutively a total pre-processing time requirement of 12.7 seconds.

The execution of an entire cycle of training or testing ADAM within the NeuralBook should take an average time of 15 seconds, showing an added computational cost of almost 2 seconds as compared to the case where the entire process is executed outside the proposed user-interface. The extra time cost can be totally justified if a number of additional advantages provided by the NeuralBook design is taken into consideration. The NeuralBook user-interface provides a graphical environment where results of each individual experiment can be stored in a unique way and filed under a single entity that itself can be retrieved at any time without the need of memorising file names, performance chart data files, directory names or time statistics that might be often separately stored. User need not re-run image pre-processing algorithms to visualise the resulting pictures but rather select the part of the book that refers to that specific experiment and simply see the raster bitmaps and any time statistics exactly as they were presented during the actual original experiment. Using several books from various experiments through the NeuralBook's external image manipulation graphical tool facility, experimental data can be exchanged and the underlying experiments can be performed with different training and testing pattern sets. The NeuralBook itself can be used as a unique environment whereby actual aircraft images can be displayed, image pre-processing information can be presented, and neural network training and testing tables can be shown together with informative performance graphs. The external image manipulation graphical tool itself can contain, additional properties that can be effectively incorporated within

the NeuralBook during the image pre-processing phase. Within an external graphical image manipulation tool, such as the DrawingBoard [8], binary and raster bitmaps can be inputed, rotated, enlarged and reduced in size, resulting in a number of desirable graphical algorithms been already implemented within the current graphical tool and therefore, a reduced need for additional external graphical utilities for any of the previously stated elementary image processing operations.

8.4 Conclusions.

A new user-interface has been presented for training and testing artificial neural networks. A specific application of the NeuralBook has be shown in the context of a position invariant pattern recognition procedure, the structure and objectives of which had been presented to a sufficient extent in previous chapters. The NeuralBook user-interface is shown to require no significant additional computational cost in training neural systems. It provides users with a unique graphical environment in which actual aircraft data can be displayed, image pre-processing information can be presented, and neural network training and testing tables can be shown together with any performance related graphs.

CHAPTER 9

Conclusions

9.1 The Thesis of Invariant Pattern Recognition.

The thesis has described a new method, through which the invariant three dimensional object classification problem can be effectively tackled in the majority of its cases. The technique is based on a new algorithmic approach to the known Chien and Aggarwal idea of intersecting three dimensional object pseudo-volumes in order to create an invariant three dimensional unique object volume description. The introduction of the normalised quadtree data structures, ensures that all intermediate generated pseudo-volumes correspond to normalised two dimensional data structures and hence represent invariant position and size pseudo-volume descriptions. The intermediate pseudo-octree data structures, named after the fact that the three dimensional volumes they represent do not correspond to real object volumes, are generated from previously normalised quadtrees and thus are themselves normalised representations of the three dimensional object views. The final actual octree data structure, resulting from merging the three required pseudo-octrees of the different non-coplanar object views, is traversed in a pre-order fashion in order to create a synthetic binary image. Such patterns are finally used to train and test the underlying artificial neural network architecture.

The above summarised version of events hides a significant number of desirable properties all of which ensure an efficient and impressive overall performance. The use of quadtrees and especially linear pointerless

quadtrees, offers an average compression rate of about 87%, and consequently an additional increase in the number of patterns to be used for testing and training. The overall computational requirements for creating the previously mentioned data structures are minimum and can be significantly improved if faster and more sophisticated computer systems are used. The use of normalisation techniques in the pre-processing stages eliminates the problem of invariance in the actual training set. Therefore, not only does the system maintain its overall high classification confidence but in the same time the required training set is reduced to a a small set of pictures and the generally time-consuming training process (as it is normally the case in backpropagation systems) is additionally drastically reduced. The idea of a "pool of information", that anyone can simply and efficiently access, normally associated with database systems, is also achieved in the current system design. By training the underlying artificial neural network architecture with a large number of different objects, normally in a compressed form so as to allow a better utilisation of the available computer memory, a vast number of stored information is offered in a simply accessible and storage efficient way. Finally the selection of ADAM, enhances further both the available system's storage capacity as well as the duration of the overall training phase. It also offers a better insight in ADAM's generalisation capabilities as well as its three dimensional object classification potential. In the same time it investigates the general behaviour of a relatively recent and certainly not yet widely researched artificial neural network architecture.

The thesis has shown that the new algorithm for volume intersection together with a neural network classifier, can effectively identify three dimensional objects in both uncluttered and noisy environments. In particular, a single picture of each aircraft is enough, in order to achieve an error-free classification result in a variety of noisy environments. Although, no misclassifications were recorded in the experiments, it has been obvious that if noise-free pictures are used for training ADAM then the system's

confidence during testing with cluttered aircraft patterns falls almost linearly as the level of random added noise in test patterns increases. If the set of pictures used for training ADAM, is increased by adding noise corrupted aircraft pictures, then a substantial, if not dramatic, change in the overall average neural network confidence occurs. As more cluttered aircraft patterns of various noise levels are added into the training set, the average classification success rate rises to reach eventually 100%. This impressive behaviour is also recorded in two entirely different experiments involving tests with position and scale invariant data patterns. In the latter case, a slight increase in the number of misclassifications has also been occurred, but in no case did it exceeded 17% of the overall applied testing set.

9.2 The Contributions of the Thesis.

The thesis has introduced a new algorithmic approach to the known method of volume intersection introduced by Chien and Aggarwal. Through this new method a normalised, position and scale independent, volumetric description of a three dimensional object can be formulated. Using this synthetic pattern approach in training ADAM, a number of highly desirable advantages are offered with most important of all the final classification success rate that was achieved. The originality of introducing a neural network classifier with an effective data pre-processing phase has been rewarded with a number of classification rates as high as 100% and with a steady and in most cases error-free classification performance. When compared to other classification techniques that were previously presented, the new method produced an average classification success of 90% which although lower than some, was nevertheless achieved with minimum time requirements and a significantly small training set.

9.3 Future Work.

An important, aspect of the current thesis has been its minimum

computational requirements. These have been consistently shown to be on the order of 15 seconds. The majority of the time is lost during the edge detection operation. Although, the main aim of the thesis has not been the design of a fast edge operator but rather a powerful and precise one, the introduction of sophisticated parallel hardware and the possibility of a new powerful as well as computationally inexpensive edge operator could almost half the entire pattern recognition time. Such improvement will be certainly most welcomed in the future.

The fact that the proposed interface is constrained to the present pattern recognition process as this is described in the current thesis, results in a very application specific implementation. A future improvement should definitely involve the expansion of the user-interface design, so that it includes as many different types of neural network systems as possible, and as many types of data that can be used as input to the underlying neural systems. The implementation of a more general user-interface should take appropriate provisions for a better quality performance graphs sketching through its possible linkage with powerful graphics packages. Finally, the ability to introduce a number of different types of input patterns should certainly involve communication links with a variety of available data libraries (i.e., libraries of pictures, libraries of recorded signals, libraries of statistical and economic data).

An additional future task that should greatly contribute in the overall pattern recognition process evaluation, should be the research into the abilities of the presented method in tracking a majority of different objects other than aircrafts and its possible extensions and implementations in other forms of current artificial neural network architectures.

9.4 Epilogue.

The current explosion of interest in neural network systems is based on a

number of scientific and economic expectations. We can be sure that neural networks will not replace conventional computers, eliminate programming or unravel the mysteries of mind. We can expect better understanding of massively parallel computation to have an important role in practical tasks and in the behavioural sciences, but only through interaction with other scientific approaches to these problems. As always, specific structures of problems, disciplines, and computational systems are the cornerstone of success. The main hope of massively parallel research is that it will provide a better basis for such efforts. Clearly, this newly reincarnated field has some interesting and promising results to share, but it is not known how these results will scale up to move real world related tasks. Numerous important research questions are emerging from this relative new area of research. Are specific models more appropriate for given classes of computations than other models? How does the sample set of learning situations affect the resulting characteristics of the neural system both during training and during testing? How can supervised and unsupervised learning be combined in such systems? How do the various interconnection structures affect the computational and operational characteristics of these systems? What algorithms can be formulated using the massively parallel paradigm of neural networks? Some of these questions were answered in this thesis in the context of a specific position and scale invariant three dimensional aircraft identification process. Further additional future work is certainly required if it is to fully understand the potential offered by artificial neural network systems.

Appendix A

Activation functions.

Let ϕ denote the typical output of an elementary artificial neuron. Typically this can be represented by the following equation

$$\phi = \sum_{i=1}^{n} x_i w_i$$

where x represents the input to the elementary neuron and w the corresponding weights for each individual input line. The final summation result, ϕ, is typically further processed by an activation function F in order to produce the neuron's final output signal ϕ'. This has normally the form of a simple linear function, i.e.,

$$\phi' = K \phi$$

where k is a threshold function, and

$$\phi' = 1 \quad \text{if } \phi > T$$

$$\phi' = 0 \quad \text{if } \phi \leq T$$

where T is a constant threshold value, or a function that more accurately simulates the nonlinear transfer characteristic of the biological neuron and permits more general network functions.

If F compresses the range of ϕ, so that ϕ' never exceeds some low limits regardless of the value of ϕ, F is called a *squashing* function. The squashing function is often chosen to be the logistic function or "sigmoid" (meaning S-shaped). This function is expressed mathematically as

$$F(x) = \frac{1}{1+e^{-x}}$$

Thus

$$\phi' = \frac{1}{1+e^{-\phi}}$$

By analogy to analogue electronic systems, we may think of the activation function as defining a nonlinear gain for the artificial neuron. This gain is calculated by finding the ratio of the change in ϕ' to a small change in ϕ. Thus, gain is the slope of the curve at a specific excitation level. It varies from a low value at large negative excitations (the curve is nearly horizontal), to a high value at zero excitation, and it drops back as excitation becomes very large and positive. Grossberg [200] found that this nonlinear gain characteristic solves the noise-saturation dilemma that he posed; that is, how can the same network handle both small and large signals? Small input signals require high gain through the network if they are to produce usable output; however, a large number of cascaded high-gain stages can saturate the output with the amplified noise (random variations) that is present in any realisable network. Also, large input signals will saturate high-gain stages, again eliminating any usable output. The certain high-gain region of the logistic function solves the problem of processing small signals, while its regions of decreasing gain at positive and negative extremes are appropriate for large excitations. In this way, a neuron performs with appropriate gain

over
a wide range of input levels.

Another commonly used activation function is the hyperbolic tangent. It is similar to the logistic function and is often used by biologists as a mathematical model of nerve-cell activation. Used as an artificial neural network activation function it is expressed as follows

$$\phi' = \tanh(x)$$

Like the logistic function, the hyperbolic tangent function is S-shaped, but is symmetrical about the origin, resulting in ϕ' having the value 0 when ϕ is 0. Unlike the logistic function, the hyperbolic tangent function has a bipolar value for ϕ', a characteristic that has been shown to be beneficial in certain networks (i.e., backpropagation).

This simple model of the artificial neuron ignores many of the characteristics of its biological counterpart. For example, it does not take into account time delays that affect the dynamics of the system; inputs produce an immediate output. More important, it does not include the effects of synchronism or the frequency-modulation function of the biological neuron, characteristics that some researchers feel to be crucial.

Despite these limitations, networks formed of these neurons exhibit attributes that are strongly reminiscent of the biological system. Perhaps enough of the essential nature of the biological neuron has been capture to produce responses like the biological system, or perhaps the similarity is coincidental; only time and research will tell.

Appendix B

Histogram equalisation.

Consider the transformation function

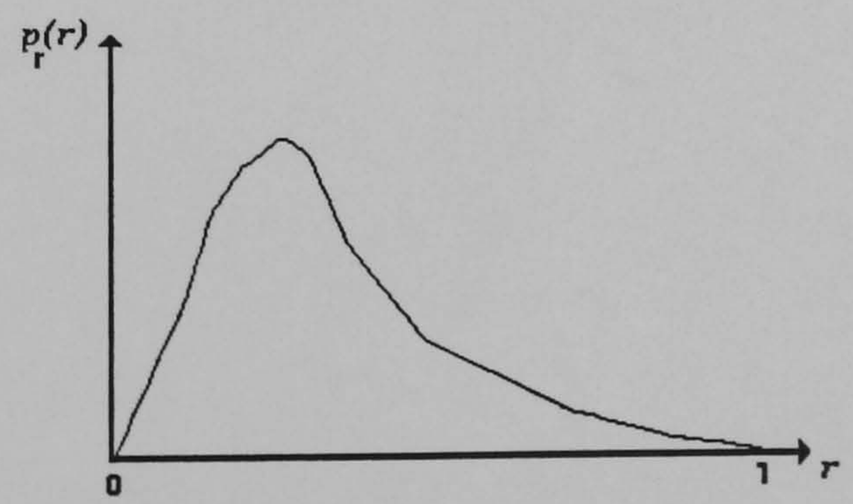
$$s = T(r) = \int_0^r p_r(w)\, dw$$

and $0 \leq r \leq 1$, where w is a dummy variable of integration. The rightmost side of the above equation is recognised as the cumulative distribution function or r. The two conditions that must be satisfied by the transformation function are

a. T(r) must be single-valued and monotonically increasing in the interval 0<=r<=1.

b. 0<=T(r)<=1 for 0<=r<=1.

The above conditions are satisfied by the transformed function since the cumulative distribution function increases monotonically from 0 to 1 as a function of r (Figure A.C.1).



Gray scale probability density function of a dark image.

Figure A.C.1

From the above equation the derivative of s with respect to r is given by

$$\frac{d\,s}{d\,r} = p_r\,(r\,)$$

Substituting $\frac{d\,s}{d\,r}$ into the general equation of the probability density function of the transformed gray levels yields

$$p_s\,(s\,) = \left[p_r\,(r\,)\frac{1}{p_r\,(r\,)}\right]_{r\,=\,T^{-1}(s\,)} = [\,1\,]_{r\,=\,T^{-1}(s\,)} \equiv 1 \qquad 0 \le s \le 1.$$

which is a uniform density in the interval of definition of the transformed variable s. It is noted that this result is independent of the inverse transformation function. This is important because it is not always easy to obtain $T^{-1}\,(s\,)$ analytically.

The foregoing development indicates that using a transformation function equal to the cumulative distribution of r produces an image whose gray scale levels have a uniform density. In terms of enhancement, this implies an increase in the dynamic range of the pixels, which can have a considerable effect in the appearance of an image.

References

[1] **Rhodnius Incorporated,** "Empress : Application Design.", Manual 2, Version 2.4, 1987.

[2] **C.H. Chien, J.K. Aggarwal,** "Volume/Surface Octrees for the representation of three dimensional Objects", Comp. Graphics Image Proc., Vol. 36, 1986, 100-113.

[3] **E. Harrigan, J. R. Kroh, W. A. Sandham, T. S. Durrani,** "Seismic Wavelet Extraction Using Artificial Neural Networks", IEE Second Inter. Conf. on Artificial Neural Networks, 18-20 November 1991, Bournemouth U.K., IEE Publication Vol. 349, 1991, 95-99.

[4] **V. Z. Kepuska, S. Mason,** "Automatic signalised point recognition with feed-forward neural network", IEE Second Inter. Conf. on Artificial Neural Networks, 18-20 November 1991, Bournemouth U.K., IEE Publication Vol. 349, 1991, 359-363.

[5] **C. R. Dyer, A. Rosenfeld, H. Samet,** "Region representation: boundary codes from quadtrees", Commun. Ass. comput. Mach., Vol. 23, 1980, 171-179.

[6] **J. Austin, T.J. Stonham,** "An associative memory for use in image recognition and occlusion analysis", Vision and Image Computing, Vol. 5, No 4, Nov. 1987, 251-261.

[7] **I.D. Benest,** "A Hypertext System with Controlled Hype", In: McAleese, R & Green, C (Eds), "Hypertext State of the Art", Albex, 1990, 52-63.

[8] D. Dukic, Ian Benest, "DrawingBoard: Adopting the Drawing Office Metaphor for Page Composition", Sun User'91 Conf., Birmingham, UK, 10-12 September 1991, 227-239.

[9] S. Christodoulakis, "Multimedia data base management : Application and problems", Position paper In Proceedings of ACM-SIGMOD Conference, Austin Texas May 28 - 31, ACM, New York 1985, 304 - 305.

[10] COMPUTER, Special issue on multimedia communications, 18, 10, 1985.

[11] C. J. Date, "An introduction to Database Systems", Volume 1, Addison-Wesley, 1986.

[12] C. J. Date, "A Critique of the SQL Database Language", ACM SIGMOD Record 14, No 3, Nov. 1984.

[13] A. Rosenfeld, A. C. Kak, "Digital Picture Processing", Chapter 5, Vol. 1, Academic Press, New York, 1982.

[14] J. W. Cooley, J. W. Tukey, "An algorithm for the Machine Calculation of Complex Fourier Series", Math. of Comput., Vol. 19, 1965, 297-301.

[15] N. Ahmed, T. Natarajan, K. R. Rao, "Discrete Cosine Transform", IEEE Trans. Comput., Vol. C-23, 1974, 90-93.

[16] Pratt W. K., Andrews H. C., Kane J., "Hadamard transform image coding", IEEE Proc. 1969, Vol. 57, No 1, 58-68.

[17] P. A. Wintz, "Transform Picture Coding", Proc. IEEE, Vol. 60, July 1972, 809-820.

[18] A. K. Jain, "Image Data Compression: A Review", Proc. IEEE, Vol. 69, March 1981, 366-406.

[19] E. J. Delp, O. R. Mitchell, "Image Compression using Block Truncation Coding", IEEE Trans. Comm., Vol. COM-27, No 9, September 1979, 1335-1342.

[20] A. N. Netravali, J. O. Limb, "Picture Coding: A Review", Proc. IEEE, Vol. 68, March 1980, 349-389.

[21] J. Amsterdam, "Data compression with Huffman coding", May 1986, BYTE, 99-108.

[22] A. Rosenfeld, "Quadtrees and Pyramids for pattern recognition and image processing", Proc. 5th Int. Conf. on Pattern Recognition, Miami Beach, 1-4 Dec 1980, 802-811.

[23] G. M. Hunter, "Linear transformation of pictures represented by Quadtree", Comput. Graphics Image Processing, Vol. 10, No 3, July 1979, 289-296.

[24] H. Samet, "Region representation : Quadtrees from binary arrays", Comp. graph. Image Proc., Vol. 13, 1980, 88-93.

[25] H. Meyr, H. G. Rosdolsky, T. S. Huang, "Some Results in Run-Length Coding", Proc. 1973 IEEE Inter. Conf. on Comm., Paper 48 D.

[26] W. K. Pratt, "Digital Image Processing", Wiley J. & Sons, 1978, 630-633.

[27] Duda R. O., Hart P. E., 'Pattern Classification and Scene Analysis", Willey, NY 1973.

[28] W. Frei, C. C. Chen, "Fast Boundary Detection : A Generalization and a new Algorithm ", IEEE Trans. Computers, Vol. C-26, No 10, 1977, 988 - 998.

[29] J. M. S. Prewitt, "Object enchancement and extraction", Picture Processing and Psychopictorics, B. S. Lipkin and A. Rosenfeld (Eds.), New York, Academic Press, 1970.

[30] T. Pavlidis, "A review of algorithms for shape analysis", Comput. Graphics Image Proc., Vol 7, 1978, 243-258.

[31] S. A. Dudani, K. J. Breeding, R. B. McGhee, "Aircraft Identification by Moments Invariants", IEEE Trans. Comput., Vol C-26, No 1, Jan. 1977, 39-45.

[32] R. L. Cosgriff, "Identification of shape", Ohio State Univ. Res. Found., Columbus, OH, Rep. 820-11, ASTIA AD-254 792, Dec. 1960.

[33] J. Sklansky, G. A. Davidson, "Recognising three dimensional objects by their silhouettes", J. Soc. Photo-Opt. Instrum. Eng., Vol 10, Oct. 1971, 10-17.

[34] S. A. Dudani, "An experimental study of moments methods for automatic identification of three dimensional objects from television images", Ph.D. dissertation, Ohio State Univ., Columbus, OH, Aug. 1973.

[35] C. W. Richard, H. Hemami, "Identification of three dimensional objects using Fourier descriptors of the boundary curve", IEEE Trans. Syst. Man Cybern., Vol. SMC-4, July 1974, 371-378

[36] J. G. Advani, "Computer recognition of three dimensional objects from optical images", Ph.D. dissertation, Ohio State Univ., Columbus, OH, Aug.

1971.

[37] S. A. Dudani, "Moment methods for the identification of three dimensional objects from optical images", M.Sc. thesis, Ohio State Univ., Columbus, OH, Aug. 1971.

[38] R. B. McGhee, "Automatic recognition of complex three dimensional objects from optical images", Proc. U.S. - Japan Seminar on Learning Control and Intelligent Control, Oct. 1973.

[39] G. H. Granlund, "Fourier preprocessing for hand print character recognition", IEEE Trans. Comput., Vol. C-21, Feb. 1972, 195-201.

[40] E. Persoon, K. S. Fu, "Shape description using Fourier descriptors", IEEE Trans. Syst. Man Cybern., Vol. SMC-7, Mar. 1977, 170-179.

[41] T. Pavlidis, F. Ali, "A hierarchical syntactic shape analyser", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-1, Jan. 1979, 2-9.

[42] C. T. Zahn, R. Z. Roskies, "Fourier descriptors for plane closed curves", IEEE Trans. Comput., Vol. C-21, Mar. 1972, 269-281.

[43] T. P. Wallace, O. R. Mitchell, K. Fukunaga, "Three dimensional Shape Analysis using local shape descriptors", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-3, No 3, May 1981, 310-323.

[44] M.-K., Hu, "Pattern recognition by moment invariants", Proc. IRE, Vol. 49, Sept. 1961, 1428.

[45] M.-K., Hu, "Visual pattern recognition by moment invariants", IRE Trans. Inf. Theory, Vol. IT-8, Feb. 1962, 179-187.

[46] F. L. Alt, "Digital pattern recognition by moments", in Optics Character Recognition, G. L. Fischer et al. Eds, Washington, DC, Spartan, 1962, 153-179.

[47] K. Udagawa, J. Toriwaki, K. Sugino, "Normalisation and recognition of two dimensional patterns with linear distortion by moments", Electron. Comm. Japan, Vol. 47, No 6, 1964, 34-46.

[48] R. G. Casey, "Moment normalisation of handprinted characters", IBM J. Res. Develop., Vol. 14, Sept. 1970, 548-557.

[49] F. W. Smith, M. H. Wright, "Automatic ship photo interpretation by the method of moments", IEEE Trans. Comput., Vol. C-20, Sept. 1971, 1089-1094.

[50] S. S. Reddi, "Radial and angular moment invariants for image identification", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-23, Mar. 1981, 240-242.

[51] M. R. Teague, "Image analysis via the general theory of moments", J. Opt. Soc. Amer., Vol. 70, Aug. 1980, 920-930.

[52] J. F. Boyce, W. J. Hossack, "Moment invariants for pattern recognition", Pattern Recognition Lett., Vol. 1, No 5-6, July 1983, 451-456.

[53] Y. S. Abu-Mostafa, D. Psaltis, "Recognitive aspects of moments invariants", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-6, Nov. 1984, 698-706.

[54] Y. S. Abu-Mostafa, D. Psaltis, "Image normalisation by complex moments", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-7, Jan. 1985, 46-55.

[55] C.-H. Teh, R. T. Chin, "On Image Analysis by the Methods of Moments", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-10, No 4, July 1988, 496-512.

[56] V. E. Giuliano et al., "Automatic pattern recognition by a gestalt method", Information and Control, 4, 1961, 332-345.

[57] E. Persoon, K. S. Fu, "Sequential Decision Procedures with Prespecified Error Probabilities and Their Applications", Tech. Rep. TR-EE 74-30, School of Electrical Engineering, Purdue University, West Lafayette, IN, 1974.

[58] A. P. Reeves, A. Rostampour, "Shape analysis of segmented objects using moments", Proc. 1981 Pattern Recognition Image Processing Conf., Dallas, TX, 1981, 171-174.

[59] T. Wallace, P. A. Wintz, "Fourier descriptors for extraction of shape information", final report of research for the period Nov. 1, 1975-Oct. 31, 1976, Contract No. F 30602-75-C-0150.

[60] S. A. Dudani, "The distance-weighted k-nearest neighbour rule", IEEE Trans. Sysy. Man. Cybern., Vol. SMC-6, Apr. 1976, 325-327.

[61] A. P. Reeves, R. J. Prokop, S. E. Andrews, F. P. Kuhl, "Three dimensional shape analysis using moments and Fourier Descriptors", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-10, No 6, Nov. 1988, 937-943.

[62] K. S. Fu, "Syntactic Methods in Pattern Recognition", New York:Academic, 1974.

[63] F. Rosenblatt, "The Perceptron : A perceiving and recognising

automation", (project PARA), Cornell Aeronautical Laboratory Report, 85-460-1, 1957.

[64] F. Rosenblatt, "The Perceptron : A theory of statistical separability in cognitive systems", Cornell Aeronautical Laboratory report, VG-1196-G-1, 1958.

[65] F. Rosenblatt, "Perceptron simulation experiments", Proc. IRE, Vol. 48, 1960, 301-309.

[66] H. D. Block, "The Perceptron : A model for Brain Functioning. I", Reviews of Modern Physics, Vol. 34, No. 1., January 1962, 123-135.

[67] T. Marill, D. M. Green, "On the Effectiveness of Receptors in Recognition Systems", IEEE Trans. Infor. Theory, January 1963, 11-17.

[68] F. Rosenblatt, "Principles of Neurodynamics", Washington: Spartan Books, 1962.

[69] B. Widrow, "Adaptive sampled-data systems - A statistical theory of adaptation", 1959 WESCON Convention Record: Part 4, 74-85.

[70] B. Widrow, "Generalisation and information storage in networks of adaline 'neurons'", In M. Yovits, G. Jacobi and G. Goldstein (Eds), Self-Organising Systems, 1962, 435-461, Washington: Spartan books.

[71] B. Widrow, R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition", IEEE Computer, March 1988, 25-39.

[72] B. Widrow, "Learning phenomena in layered neural networks", Proc. IEEE First Inter. Conf. on Neural Networks, Vol. II, San Diego: IEEE, 1987, 411-430.

[73] B. Widrow, R. Winter, R. Baxter, "Layered neural networks for pattern recognition", IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-36, 1988, 1109-1118.

[74] B. Widrow, M. Hoff, "Adaptive switching circuits", 1960 WESCON Convention Record: Part 4, 1960, 96-104.

[75] P. Werbos, "Beyond regression : New tools for prediction and analysis in the behavioural sciences", Ph.D. Dissertation, Harvard University, 1974.

[76] S. Troxel, S. Rogers, M. Kabrisky, "The use of neural networks in PSRI target recognition", Proc. IEEE Inter. Conf. on Neural Networks, Vol. I, San Diego: IEEE 1988, 593-600.

[77] H. Yang, C. Guest, "Performance of backpropagation for rotation invariant pattern recognition", Proc. First IEEE Inter. Conf. on Neural Networks, Vol. IV, San Diego: IEEE 1987, 365-370.

[78] R. Hecht-Nielsen, "Neurocomputing", Addison - Wesley, 1990, 128-131.

[79] J. Anderson, J. Silverstein, S. Ritz, R. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model", Psychological Review, 84, 1977, 413-451.

[80] J. Anderson, P. A. Penz, M. T. Gately, D.Collins, "Radar signal categorisation using a neural network", Abstracts First Annual INNS Meeting, Boston MA., 1988, 422.

[81] G. Carpenter, S. Grossberg, "Adaptive Resonance Theory: Stable self-organisation of neural recognition codes in response to arbitrary lists of input patterns", Eighth Annual Conference of the Cognitive Science Society,

Hilsdale, NJ.: Lawrence Erlbaum Associates, 1986, 45-62.

[82] G. Carpenter, S. Grossberg, "ART 2: Self organisation of stable category recognition codes for analog input patterns", Proc. First IEEE Inter. Conf. on Neural Networks, Vol. II, San Diego: IEEE 1987, 727-736.

[83] G. Carpenter, S. Grossberg, "Invariant pattern recognition and recall by an attentive self organising ART architecture in a nonstationary world", Proc. First IEEE Inter. Conf. on Neural Networks, Vol. II, San Diego: IEEE, 1987, 737-746.

[84] S. Rak, P. Kolodzy, "Invariant object recognition with the adaptive resonance (ART) network", Neural Networks Supplement: INNS Abstracts, 1, 1988, 461.

[85] K. Fukushima, "Cognitron: A self organising multilayered neural network", Biological Cybernetics, 20, 1975, 121-136.

[86] K. Fukushima, "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements", IEEE Trans. Systems, Science and Cybernetics, Vol. SSC-5, No. 4, October 1969, 322-333.

[87] K. Fukushima, "A hierarchical neural network model for associative memory", Biological Cybernetics, 50, 1984, 105-113.

[88] D. H. Hubel, T. N. Wiesel, "Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of a cat",Journal of Neurophysiology, 28, 1965, 229-289.

[89] K. Fukushima, "A Neural Network for Visual Pattern Recognition", IEEE Computer, March 1988, 65-74.

[90] K. Fukushima, "A Neural network Model for Selective Attention", Proc. First IEEE Inter. Conf. on Neural Networks, Vol. II, San Diego: IEEE 1987, 11-18.

[91] T. Kohonen, "Automatic formation of topological maps in a self organising system", In E. Oja and O. Simula (Eds), Proc. 2nd Scandinavian Conference on Image Analysis, Espoo: Suomen Hahmontunnistustutkimuksen Seuro, 1981, 214-220.

[92] N. Nasrabadi, Y. Feng, "Vector quantisation on images based upon the Kohonen self organising feature map", Proc. IEEE Inter. Conf. on Neural Networks, Vol. I, San Diego: IEEE 1988, 101-105.

[93] T. Kohonen, K. Makisara, "Representation of sensory information in self organising feature maps", In J. Denker (Ed.), AIP Conference Proceedings 151 : Neural Networks for computing, New York: American Institute of Physics, 1986, 271-276.

[94] J. J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities", Proc. National Academy of Science, 79, 1982, 2554-2558.

[95] A. Fuchs, H. Haken, "Pattern Recognition and Associative Memory as Dynamical Processes in Nonlinear Systems", Proc. IEEE Inter. Conf. on Neural Networks, Vol. I, San Diego: IEEE 1988, 217-224.

[96] T. Troudet, A. Tabatabai, "An Adaptive Neural net approach to the segmentation of mixed gray-level and binary pictures", Proc. IEEE Inter. Conf. on Neural Networks, Vol. I, San Diego: IEEE, 1988, 585-592.

[97] B. Kosko, "Bidirectional Associative Memories", IEEE Trans. Systems, Man and Cybernetics, Vol. SMC-18, 1988, 42-60.

[98] B. Kosko, "Adaptive Bidirectional Associative memories", Applied Optics, Vol. 26, 1987, 4947-4960.

[99] G. Hinton, D. Ackley, T. Sejnowski, "Boltzmann machines: Constraint satisfaction networks that learn", Carnegie - Mellon University, Department of Computer Science Technical Report, CMU-CS-84-119, 1984.

[100] J. Mousouris, "Gibbs and Markov random systems with constraints", Journal Statistical Physics, 10, 1974, 11-33.

[101] S. Geman, D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-6, 1984, 721-741.

[102] S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimisation by simulated annealing", Science, vol. 220, 1983, 671-680.

[103] R. Hecht-Nielsen, "Counterpropagation networks", Proc. First IEEE Inter. Conf. on Neural Networks, Vol. II, San Diego: IEEE 1987, 19-32.

[104] S. Grossberg, "A prediction theory for some nonlinear functional differential equations: I. Learning of lists", Journal of Mathematical Analysis and Applications, Vol. 21, 1968, 643-694.

[105] R. Hecht-Nielsen, "Applications of counterpropagation networks", Neural Networks, Vol. 1, 1988, 131-140.

[106] J. Austin, "ADAM: An Associative Neural Network Architecture for invariant pattern classification", Proc. First IEE Conf. on Artificial Neural Networks, Vol. 313, 1989, 196-200.

[107] R. P. Lippmann, "An introduction to computing with neural nets",

IEEE ASSP Magazine, Vol. 2, Apr. 1987, 4-22.

[108] D. E. Rumelhart, J. L. McLelland, "Parallel Distributed Processing", Cambridge, MA: MIT Press, 1986.

[109] R. P. Gorman, T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar signals", Neural Networks, Vol. 1, 1988, 75-90.

[110] E. Barnard, D. Casasent, " A comparison between criterion functions for linear classifiers, with an application to neural nets", IEEE Trans. Syst. Man Cybern., Vol. 19, Sept. / Oct. 1989.

[111] D. J. Burr, "Experiments on neural net recognition of spoken and written text", IEEE Trans. Acoust. Speech Signal Process., Vol. 36, July 1988, 1162-1168.

[112] K. Fukushima, "Neocognitron: A self-organising neural network model for a mechanism of pattern recognition unaffected by a shift in position", Biol. Cybern., Vol. 36, 1980, 193-202.

[113] C. L. Giles, R. D. Griffin, T. Maxwell, "Encoding geometric invariances in higher-order neural networks", Neural Information Processing systems, D. Z. Anderson, Ed., Denver, CO: AIP, 1988, 301-309.

[114] R. Winter, B. Widrow, "Madaline rule II: a training algorithm for neural networks", Proc. IEEE Int. Conf. Neural Networks (San Diego), July 1988, Vol. I, 401-408.

[115] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang, "Phoneme recognition using time-delay neural networks", IEEE Trans. Acoust. Speech Signal proc., Vol. 37, Mar. 1989, 328-339.

[116] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning internal representations by error propagation", Parallel Distributed Processing, Cambridge, MA: MIT Press, 1986, Ch. 8, 318-362.

[117] K. J. Lang, G. E. Hinton, "The development of the time-delay neural network architecture for speech recognition", Tech. Report CMU-CS-88-152, Department of Computer Science, Carnegie Mellon University, 1988.

[118] S.-I Amari, "Invariant structures of signal and feature spaces in pattern recognition problems", RAAG Mem., Vol. 4, 1968, 553-566.

[119] F. Sadjadi, E. L. Hall, "Three dimensional moment invariants", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-2, Mar. 1980, 34-45.

[120] K.-I. Kanatani, "Transformation of optical flow by camera rotation", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-10, Mar. 1988, 131-143.

[121] D. Casasent, "Computer generated holograms in pattern recognition: A review", Opt. Eng., Vol. 24, Sept./Oct. 1985, 724-730.

[122] D. E. Glover, "An optical Fourier/electronic neurocomputer automated inspection system", Proc. IEEE Int. Conf. Neural Networks (San Diego), July 1988, Vol. I, 569-576.

[123] A. Khotanzad, J. H. Lu, "Distortion invariant character recognition by a multilayer perceptron and backpropagation learning", Proc. IEEE Int. Conf. Neural networks (San Diego), July 1988, Vol. I, 625-632.

[124] D. J. Abel, J. L. Smith, "A data structure and algorithm based on a linear key for a rectangle retrieval problem", Comp. Vision. Graphics Image Proc., Vol. 24, No. 1, Oct. 1983, 1-13.

[125] F. W. Burton, J. G. Kollias, "Comment on the explicit quadtree as a structure for computer graphics", Comput. J., Vol. 26, No. 2, May 1983, 188.

[126] B. G. Cook, "The structural and algorithmic basis of a geographic data base", Proc. First Intr. Advanced Study Symposium Topological Data Structures for Geographic Information Systems, G. Dutton Ed., Harvard Papers on Geographic Information Systems, 1978.

[127] G. M. Hunter, "Efficient computation and data structures for graphics", Ph.D. dissertation, Dept. Electrical Engin. and Comp. Science, Princeton Univ., N. J., 1978.

[128] S. Ranade, A. Rosenfeld, H. Samet, "Shape approximation using quadtrees", Pattern Recognition, Vol. 15, No. 1, 1982, 31-40.

[129] M. Imme, "A new method for edge detection", Proc. 3rd IEE Inter. Conf. Image Proc. and its applications", No. 307, 18-20 July 1989, Warwick Univ., U.K., 656-659.

[130] P. J. Burt, "Fast, Hierarchical Correlations with Gaussian-like kernels", Comput. Vision Lab., Univ. Maryland, Techn. Rep. TR-860, Jan 1980.

[131] M. Nagao, T. Matsuyama, "Edge preserving smoothing", Comput. Graphics Image Proc., Vol. 9, 1979, 394-407.

[132] A. Kundu, S. K. Mitta, P. P. Vaidyanatha, "Application of two dimensional generalised mean filtering for removal of impulse noises from images", IEEE Trans. Acoustics, Speech, Signal Proc., ASSP-32, 1984, 600-609.

[133] J.-S. Lee, "Digital Image Enhancement and Noise Filtering by use of

local statistics", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-2, No. 2, March 1980.

[134] E. L. Hall, "Almost Uniform Distribution for Computer Image Enhancement", IEEE Trans. Computers, Vol. C-23, No. 2, February 1974, 207-208.

[135] R. Hummel, "Histogram modification techniques", Comput. Graphics Image Proc., Vol. 4, 1975, 209-224.

[136] W. Frei, "Image enhancement by histogram hyperbolisation", Comput. Graphics Image Proc., Vol. 6, 1977, 286-294.

[137] L. S. Davis, A. Rosenfeld, "Noise cleaning by iterated local averaging", IEEE Trans. Syst. Man Cybern., Vol. SMC-8, 1978, 705-710.

[138] A. Rosenfeld, "Snow removal - a noise stripping process for picture signals", IEEE Trans. Inf. Theory, Vol. IT-8, 1962, 129-144.

[139] D. Harwood, H. Hakalathi, " A symmetric nearest-neighbour smoothing algorithm", Comput. Graphics Image Proc., in preparation.

[140] B. E. Bayer, P. G. Powell, "A method for digital enhancement of unsharp grainy photographic images", Advances in Computer Vision and Image Processing, Vol. 2, 1986, JAI Press Inc., 31-88.

[141] V. Di Pietro, G. Molenaar, "Unusual Martian surface features", Third Edition, 1982, Mars Research, Clenn Dale, MP 20769.

[142] A. V. Oppenheim, R. W. Schafer, "Digital Signal Processing", Englewood Cliffs, NJ: Prentice-Hall, 1975.

[143] D. Marr, E. Hildreth, "Theory of edge detection", Proc. Royal Soc. London, Vol. B207, 1980, 187-217.

[144] J. F. Canny, "Finding edges and lines in images", Master's thesis, Mass. Inst. Technol., A. I. Labs. Techn. Report 720, June 1983.

[145] L. G. Roberts, "Machine Perception of three dimensional Solids", Optical and Electro-Optical Inform. Proc., J. T. Tippett et al. Eds., Mass. Inst. Technol. Press, Cambridge, Mass., 1965, 159-197.

[146] D. H. Ballard, C. M. Brown, "Computer Vision", Prentice-Hall, 1982, 75-87.

[147] R. A. Kirsch, "Computer determination of the constituent structure of biological images", Computers and Biomedical Research, 4, 3, June 1971, 315-328.

[148] D. L. Davis, "Edge detection in digital images using small mask and vector operators and the method of polar histograms", M.Sc. Thesis, Univ. Tennessee, Knoxville, 1978.

[149] A. Rosenfeld, "A nonlinear edge detection technique", Proc. IEEE Letters, 58, 5, May 1970, 814-816.

[150] M. Hueckel, Journal ACM, Vol. 18, 1971, 113-125.

[151] K. F. Shanmugan, F. M. Dikey, J. A. Green, "An optimal frequency domain filter for edge detection in digital pictures", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-1, 1979, 37-49.

[152] E. C. Hildreth, "Implementation of a theory of edge detection", MIT, Cambridge, A. I. Memo 579, 1980.

[153] R. M. Haralick, "Edge and region analysis for digital image data", Comput. Graphics Image Proc., Vol. 12, 1980, 60-73.

[154] T. O. Binford, "Inferring surfaces from images", Artif. Intell., Vol. 17, 1981, 205-244.

[155] V. S. Nalwa, "Edge detector resolution improvement by image interpolation", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-9, No. 3, 1987.

[156] F. O'Gorman, "Edge detection using Walsh functions", Artif. Intell., Vol. 10, 1978, 215-223.

[157] A. Klinger, "Patterns and research statistics", Optimising Methods in Statistics (J. S. Rustagi, Ed.), NY: Academic, 1971.

[158] G. M. Hunter, K. Steiglitz, "Linear transformation of pictures represented by quad trees", Comput. Graph. Image Proc., Vol. 10, No 3, July 1979.

[159] G. M. Hunter, K. Steiglitz, "Operations on images using quadtrees", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-1, 1979, 145-153.

[160] H. Samet, "Connected component labeling using quadtrees", Journal ACM, Vol. 28, No. 3, July 1981, 487-501.

[161] A. Klinger, M. L. Rhodes, "Organisation and access of image data by areas", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-1, 1979, 50-60.

[162] H. Samet, "An algorithm to convert rasters to quadtrees", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-3, 1981, 93-95.

[163] H. Samet, "Region representation: quadtrees from boundary codes", Commun. Ass. comput. Mach., Vol. 23, 1980, 163-170.

[164] M. Shneier, "Path-length distances for quadtrees", Inf. Sci., Vol. 23, 1981, 45-67.

[165] L. Jones, S. S. Iyengar, "Representation of a region by a forest of quadtrees", IEEE PRIP 81 Conf., Dallas, 1981, 57-59.

[166] I. Gargantini, "An effective way to represent quadtrees", Comm. ACM, Vol. 25, No. 12, 1983, 905-910.

[167] M. A. Oliver, N. E. Wiseman, "Operations on quadtree encoded images", Comput. Journal, Vol. 26, No. 1, 1983, 83-91.

[168] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing", IBM Canada, 1966.

[169] I. Gargantini, "An effective way to store quadtrees", Comm. Ass. Comput. Mach., Vol. 25, 1982, 905-910.

[170] E. Kawaguchi, T. Endo, "On a method of binary picture representation and its application to data compression", IEEE Trans. Pattern Analysis Machine Intell., Vol. PAMI-2, Jan. 1980, 27-35.

[171] D. E. Knuth, "The Art of Computer Programming, Fundamental Algorithms", Vol. 1, Addison-Wesley, Reading, 1968, 401.

[172] I. Gargantini, "Linear oct-trees for fast processing of three dimensional objects", Comput. Graphics Image Proc., Vol. 20, 1982, 365-374.

[173] I. Gargantini, "Translation, rotation and superposition of linear quadtrees", Int. J. Man Mach. Stud., Vol. 18, 1983, 253-263.

[174] I. Gargantini, H. H. Atkinson, "Linear quadtrees: a blocking technique for contour filling", Pattern Recognition, Vol. 17, 1984, 285-293.

[175] D. J. Abel, "A B^+-tree structure for large quadtrees", Comput. Vision Graph. Image Proc., Vol. 27, 1984, 19-31.

[176] A. Klinger, C. R. Dyer, "Experiments on picture representation using regular decomposition", Comp. Graph. Image Proc., Vol. 5, 1976, 68-105.

[177] S. Tanimoto, T. Pavlidis, "A hierarchical data structure for picture processing", Comp. Graph. Image Proc., Vol. 4, 1975, 104-119.

[178] M. Li, W. I. Grosky, R. Jain, "Normalised Quadtrees with respect to translations", Proc. PRIP_81, Dallas, TX, Aug. 3-5 1981, 60-62.

[179] C. H. Chien, J. K. Aggarwal, "A Normalised Quadtree Representation", Proc. IEEE 1983, CH1891-1/83, 121-126.

[180] P. W. Likins, "Elements of engineering mechanics", NY: McGraw-Hill, 1973.

[181] D. Rutovitz, "Machines to classify chromosomes?", Human Radiation Cytogenesis", H. J. Evans et al. Eds, Amsterdam: North-Holland, 1967.

[182] W. K. Chow, J. K. Aggarwal, "Computer analysis of planar curvilinear moving images", IEEE Trans. Comput., Vol. C-26, 1977, 179-185.

[183] S. L. Tanimoto, "A pyramid model for binary picture complexity",

Proc. PRIP-77, Rensselaer Poly Inst., New York, June 6-8 1977, 25-28.

[184] J. K. Aggarwal, L. S. Davis, W. N. Martin, J. W. Roach, "Survey: Representation methods for three dimensional objects", Progress in Pattern Recognition, Vol. 1, L. N. Kanal and A. Rosenfeld Eds., NY: North-Holland, 1981, 377-391.

[185] A. Requicha, H. Voelcker, "Geometric Modeling of mechanical parts and machining processes", COMPCONTROL 1979, Sopron, Hungary, Nov. 1979.

[186] B. G. Baumgart, "Geometric Modeling for computer vision", STAN-CS-74-463, Computer Sc., Stanford Univ, Oct. 1974.

[187] A. Baer, C. Eastman, M. Henrion, "Geometric Modeling: A survey", Comput. Aided Design, Vol. 11, No. 5, Sept. 1979.

[188] S. N. Srihari, "Representations of 3D digital images", Comput. Surveys, Vol. 13, 1981, 394-424.

[189] C. L. Jackins, S. L. Tanimoto, "Quadtrees, Octrees, and K-trees: A generalised approach to recursive decomposition of euclidean space", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-5, No. 5, 1983, 533-539.

[190] M. M. Yau, S. N. Srihari, "A hierarchical data structure for multidimensional digital images", Comm. ACM, Vol. 26, No. 7, 1983, 504-515.

[191] H. Samet, "Data structures for quadtree approximation and compression", Comm. ACM, Vol. 28, 1985, 973-993.

[192] C. I. Conolly, "Cumulative generation of Octree models from range

data", Proc. Int. Conf. Robotics, Atlanta, March 13-15 1984, 25-32.

[193] C. H. Chien, J. K. Aggarwal, "Identification of 3D objects from multiple silhouettes using Quadtrees/Octrees", Comp. Vision Graphics Image proc., Vol. 36, 1986, 256-273.

[194] J. Veestra, N. Ahuja, "Efficient Octree generation from silhouettes", Proc. IEEE Comp. Soc. Conf. on Comp. Vision and pattern recognition, Miami Beach, June 22-26, 1986, 537-542.

[195] C. H. Chien, J. K. Aggarwal, "Computation of Volume/Surface Octrees from Contours and silhouettes of multiple views", Proc. CVPR-86, IEEE comp. Soc., June 1986, 250-255.

[196] A. G. Requicha, "Representations for rigid solids: Theory, methods and systems", Comp. Surveys, Vol. 12, No. 4, 1980, 437-464.

[197] I. Calborn, I. Charkrarartn, V. Vanderschel, "A hierarchical data structure for representing the spatial decomposition of euclidean space", IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAMI-5, No. 4, 1985, 24-31.

[198] R. M. Baecker, W. A. S. Buxton, "Readings in Human-Computer Interaction", Morgan Kaufmann, Palo Alto. Calif., 1987.

[199] M. Frisse, "From Text to Hypertext", BYTE, Oct. 1988, 247-253.

[200] S. Grossberg, "Contour enhancement, short-term memory, and consistencies in reverberating neural networks", Studies in Applied Mathematics, Vol. 52, 217-257.

[201] D. O. Hebb, "Organisation of behaviour", NY:Science Editions, 1949.