A Hybrid Neural Network Architecture for Texture Analysis in Digital Image Processing Applications

By

Michael John Arrowsmith

Thesis submitted to the University of Central Lancashire

In partial fulfilment of the requirements for the degree of

Doctor of Philosophy

June 2002

The work presented in this thesis was carried out in the Department of Engineering and Product Design, University of Central Lancashire, Preston, England.

Declaration

I declare that while registered with the University of Central Lancashire for the degree of Doctor of Philosophy I have not been a registered candidate or enrolled student for another award of the University of Central Lancashire, or any other academic or professional institution during the research programme. No portion of the work referred to in this thesis has been submitted in support of any application for another degree or qualification of any other university or institution of learning.

Michael John Arrowsmith

Abstract

A Hybrid Neural Network

Architecture for Texture Analysis in

Digital Image Processing Applications

By

Michael John Arrowsmith

A new hybrid neural network model capable of texture analysis in a digital image processing environment is presented in this thesis. This model is constructed from two different types of neural network, self-organisation and back-propagation. Along with a brief resume of digital image processing concepts, an introduction to neural networks is provided. This contains appropriate documentation of the neural networks and test evidence is also presented to highlight the relative strengths and weaknesses of both neural networks. The hybrid neural network is proposed from this evidence along with methods of training and operation. This is supported by practical examples of the system's operation with digital images. Through this process two modes of operation are explored, classification and segmentation of texture content within images.

Some common methods of texture analysis are also documented, with spatial grey level dependence matrices being chosen to act as a feature generator for classification by a back-propagation neural network, this provides a benchmark to assess the performance of the hybrid neural network. This takes the form of descriptive comparison, pictorial results, and mathematical analysis when using aerial survey images.

Other novel approaches using the hybrid neural network are presented with concluding comments outlining the findings presented within this thesis.

Acknowledgements

I wish to thank firstly my Director of Studies Dr Martin Varley, his constant support and guidance throughout all aspects of this research project has been invaluable. I would also like to thank my Second Supervisor Dr David Heys and my External Supervisor Prof Phil Picton for their suggestions on the construction of the thesis.

I must also thank my wife Sue and my daughter Eleanor for their patience, love and understanding. Their support has been greatly appreciated.

Finally I would like to thank my parents to whom I dedicate this thesis.

Table of Contents

List of Figures

List of Tables

List of Appendices

Abbreviations / Acronyms

2D	Two Dimensional
A/D	Analogue to Digital Converter
ANN	Artificial Neural Network
BPNN	Back-Propagation Neural Network
Bps	Bits Per Second
Byte	8 Bits
CCD	Charge Coupled Device
DCT	Discrete Cosine Transform
FFT	Fast Fourier Transform
GHz	10^9 Hertz
Kbytes	1024 bytes
Landsat	Land Satellite Mapping Programme managed by National Aeronautics and Space Administration (NASA) and the US Geological Survey .
Mbytes	1048576 bytes
MHz	10^6 Hertz
MR	Magnetic Resonance
MRF	Markov Random Field
Pixel	Picture Element
RAM	Random Access Memory
SAR	Synthetic Aperture Radar
SOM	Self-Organising Map
SPOT	French and European Space Agency's optical series satellite. (Système probatoire d'observation terrestre)
SGLDM	Spatial Grey Level Dependence Matrix

Chapter 1 Introduction

The term digital image processing is distilled by Jahne [Jahne 95] "Human beings perceive most of the information about their environment through their visual sense. While for a long time images could only be captured by photography, we are now at the edge of another technological revolution which allows image data to be captured, manipulated, and evaluated electronically by computers."

This project concentrates its focus upon the area of identifying textures within images and the ability to segment textures out of images via the implementation of artificial neural networks, often referred to as neural networks.

Kulkarni [Kulkarni 1994] identifies the capabilities of neural networks when acting in the digital image processing realm:-

- Remote Sensing, segmentation of Landsat images for various purposes such as agriculture, forestry, mineral resources, meteorology and military.
- Medical Image Processing, classification and segmentation of X-ray and magnetic resonance (MR) images.
- Fingerprint Processing, feature extraction and pattern recognition.
- Character Recognition, automatic data entry from print and script.
- Characterisation of Faces, optical recognition for security applications.
- Data Compression, acting as encoders and decoders for compressing images or data streams.

Section 1.1 provides a brief overview of the image processing terminology and image formats. It addresses how images are stored in computer memory and the type of image storage format commonly used. The common process of histogram equalisation used within this thesis to ensure all images have a commonality across them is also documented.

The aims of the research required to produce this thesis are identified in Section 1.2. The primary goals and activities of this research are also documented.

This chapter concludes with Section 1.3, which contains an overview of the thesis. A brief resume of each chapter is documented to give a concise description of their contents.

1.1 Introduction To Digital Image Processing

Images are made up from pixels (picture elements), each pixel donates a single point of light intensity in the image. The resolution defines the number of pixels in the image and the range of light intensities that each pixel can represent. Typically in most image processing a pixel is represented by a single byte, therefore the number of light intensities available is 2^8, which is 256, giving a range of 0 to 255 grey levels. Although not used in this research, larger formats using a word or long word representing the value of a pixel can give grey scale range of 0 to 65,535 and 0 to 4,294,967,295 respectively. A majority of images available in the research domain are standardised on two image sizes: 256 by 256 pixels and 512 by 512 pixels, with both of them using a single byte to store pixel information. This file format is often referred to as a raw image. There is no information stored in the file about the dimensions or resolution of the image, so it is only possible to store monochrome images because of the lack of descriptive information within the file format that would be needed to hold a colour palette. The images chosen for presentation in this thesis use the characteristics from the above to allow readers familiar with this image format to make meaningful comparisons of the hybrid neural network to previous work carried out by other methodologies.

All processing and manipulation of images carried out in the project was done via sampling the image from the video memory of the personal computer displaying them through custom software that has been developed. Appendix A explores some of the issues discovered during the development process.

Images are constructed from a matrix of co-ordinates made up from rows and columns. these are referred to as the x dimension for rows and y dimension for columns.

Equation (1.1.1) shows the construction of an image. Pixel intensity P is represented in the spatial domain by $p(y,x)$

$$P(y,x)=\begin{bmatrix} p(1,1) & p(1,2) & . & p(1,x_{max}) \\ p(2,1) & p(2,2) & . & p(2,x_{max}) \\ . & . & . & . \\ p(y_{max},1) & p(y_{max},2) & . & p(y_{max},x_{max}) \end{bmatrix} \tag{1.1.1}$$

This method of representing an image as a matrix of pixels is common throughout all realms of digital image display.

A large amount of imagery is presented in this thesis and to ensure that all the images have the same qualities before they are processed, a pre-process operation of grey scale equalisation is applied to them. This ensures that all the images contain the same grey scale range.

This operation relies upon manipulating the order of pixel values against the theoretical maximum value they can possess. Take for example an image of 256 by 256 pixels with grey scales being represented from 0 to 255, if the majority of pixels had a grey scale value that was below 100 then the image would look dark. A statistical operation that could be carried out to improve the brightness of the image would be to artificially alter the histogram of grey scale values [James 1987].

The process would be to determine the ideal average (z) for the number of pixels assigned to each grey level, with x_{max} and y_{max} being the maximum number of rows and columns in the image, and g_levels being the number of grey scales used in the image.

$$z = \frac{x_{max} y_{max}}{g\_levels} \tag{1.1.2}$$

Using this average, the number of pixels assigned to each grey level could be equalised. If the number of grey levels is less than the average some are added to this level from the next highest grey level until the average is reached. If however the number is greater than the average then the next grey level is skipped and those pixels assigned to a higher grey scale.

Developing this process further, Low [Low 1991] depicts the use of the cumulative frequency in adjusting the histogram of grey scale values. The cumulative frequency $t(g)$ is used in equation (1.1.3) to determine which grey levels are to be used in the corrected image.

$$F(g) = \max\,(0, \text{round}\left(\frac{g\_levels \cdot t(g)}{x_{max} y_{max}}\right) - 1) \tag{1.1.3}$$

Where $F(g)$ is the grey level in the corrected image.

g is the current grey scale.

Round, is the result rounded to the nearest valid grey scale.

For example, the following histogram (Figure 1.1) is constructed from a image of dimensions of 5 by 6 pixels. It is equalised using equation (1.1.3) to produce the results shown below in Figure 1.2 and in Table 1.1.

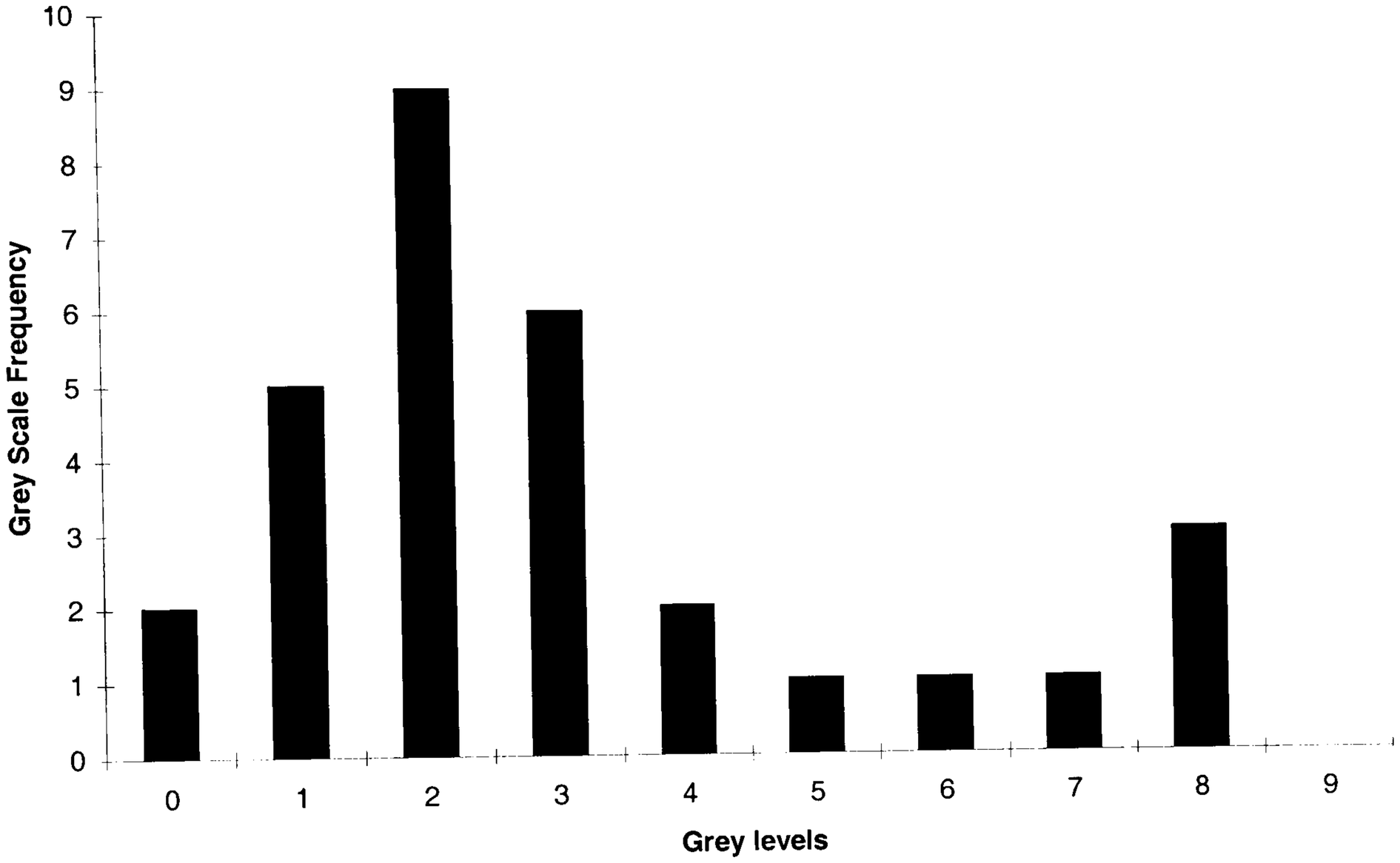


Figure 1.1 Histogram of 5 by 6 Image before Equalisation

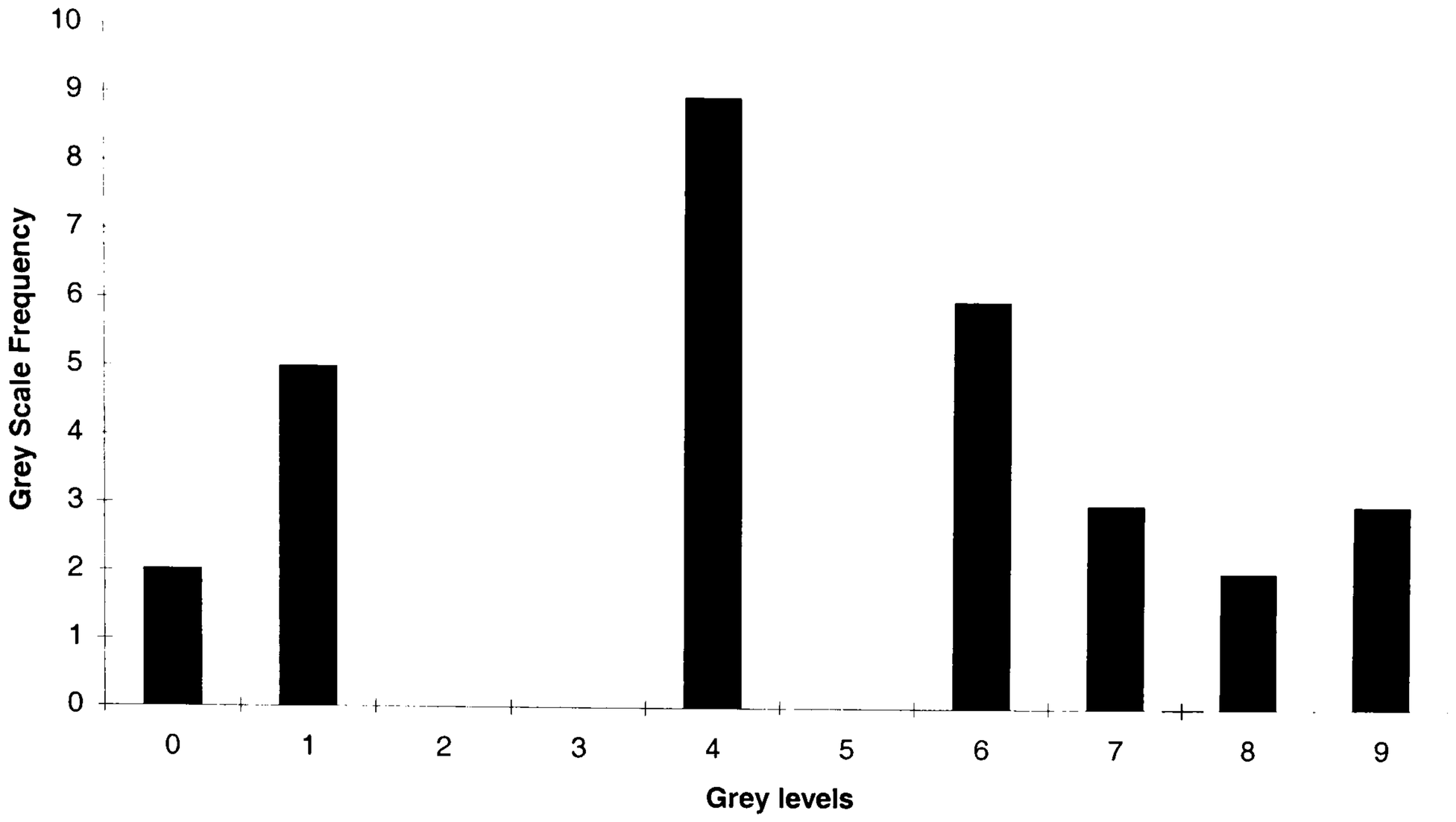


Figure 1.2 Histogram of 5 by 6 Image after Equalisation

g	$t(g)$	$F(g)$
0	2	0
1	7	1
2	16	4
3	22	6
4	24	7
5	25	7
6	26	8
7	27	8
8	30	9
9	30	9

Table 1.1 New Grey Scale Values of 5 by 6 image after Equalisation

A practical example of this can be seen in the following image Figure 1.3. The image could be considered to be very bright, with its grey scale histogram skewed to towards the higher values. Performing histogram equalisation upon it results in Figure 1.4.

Figure 1.3 Airplane Image before Equalisation

Figure 1.4 Airplane Image after Equalisation

The change can be seen from the histograms shown in Figure 1.5. The majority of grey scales were originally clustered around the grey scale level of 200. However after equalisation this cluster has decreased in brightness to around 150. Note that the shape of the distribution of grey scales is still the same as before but is elongated by the equalisation process.

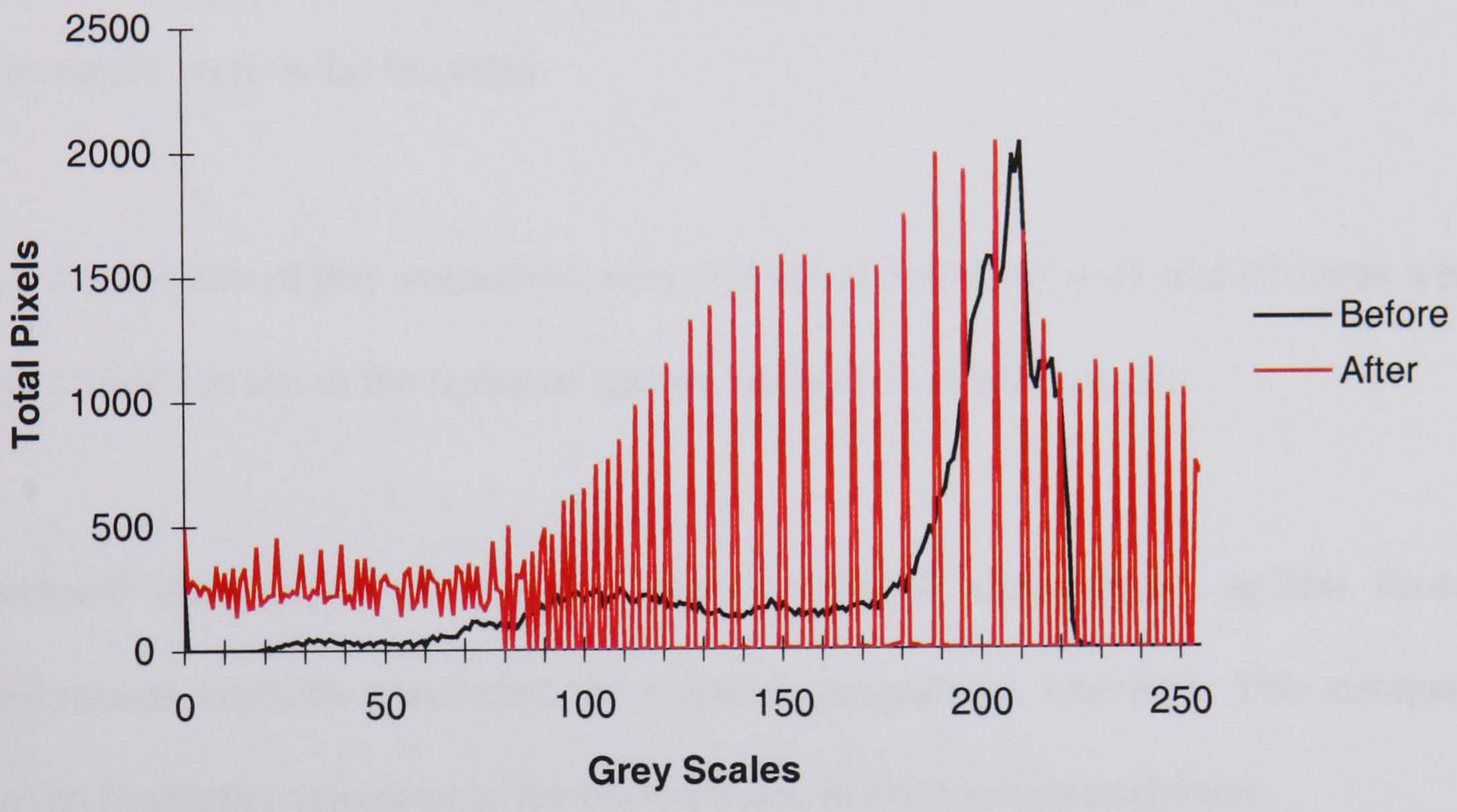


Figure 1.5 Histogram of grey scale intensities for Airplane Image

1.2 Aims of the Research

The primary goal of the research presented in this thesis, is to provide evidence of capabilities of neural networks in the realm of texture analysis in digital image processing and to assess their performance against existing documented conventional methodologies.

Neural networks were to be evaluated, with their relative strengths and weakness investigated. Practical examples of the operation in a digital image processing environment were to be recorded.

Upon completion of this evaluation, new and novel neural network architectures were to be identified for use in the realm of texture analysis in digital images.

A critical comparison of the new neural network architectures against those of conventional methods concluded the research programme activities. This comparison gives an in-depth evaluation of the performance of the systems under test.

1.3 Overview of the Thesis

- **Chapter Two** provides a literature review of the work carried out in the field of texture analysis and the methodologies applied.
- **Chapter Three** introduces the concept of texture within a digital image processing environment. Some common methodologies that are able to perform texture analysis are examined.
- **Chapter Four** examines neural networks, with both natural and artificial networks being discussed. Two neural networks are investigated in detail with practical demonstrations of them working upon digital images included.
- **Chapter Five** brings together the two neural network models discussed in Chapter Four and proposes a new hybrid neural network system. A description of the architecture is given along side the training processes required to allow the operation within a digital image processing environment. Again a practical demonstration is given of the hybrid neural network architecture in operation.
- **Chapter Six** demonstrates the capabilities of the hybrid neural network with two modes of operation. The first mode of operation is classification of benchmark textures. The second mode shows the hybrid neural network to be capable of segmenting real world images. The hybrid neural network is compared with another method of texture analysis, a spatial grey level dependence matrix feature generator acting in conjunction with a back-propagation neural network classifier. The results are documented and supported by analysis.

- **Chapter Seven** highlights the novel aspects contained within this thesis and the original contributions to knowledge. Some conclusions are drawn with regards to the work carried out over this research project. This chapter concludes by proposing some new concepts based upon the hybrid neural network that could lead to further work and research.
- **Appendix A** reviews the processes that were used to generate the neural network models used in this thesis. A brief discussion of the software development life cycle is given with a description of some of the data structures developed to implement a neural network in software.
- **Appendix B** gives some insight into the use of cross-correlation coefficients from chapter three when implementing them in software. This is accompanied by an example.
- **Appendix C** provides details on the benchmarking process used of classifying spatial grey level dependence matrices with a back-propagation neural network, this is used to asses the performance of the hybrid neural network
- **Appendix D** includes an accompanying paper to the thesis published during this research project.
- **Appendix E** shows the real world imagery and results generated by testing the hybrid neural network and spatial grey level dependence matrices in chapter six.

Chapter 2 Literature Review

This chapter offers a literature review of the work carried out in the field of texture analysis. It examines applications and methodologies applied to texture analysis using digital image processing. It is presented in four sections, each section based upon a particular grouping of methodologies.

Section 2.1 presents statistical methods such as first and second order statistics.

Section 2.2 highlights model based approaches such as Markov Random Fields and Fractals.

Section 2.3 covers signal processing approaches to texture analysis, methods such as spatial domain, Fourier, Gabor and Wavelets are presented.

Section 2.4 explores all aspects of the application of neural networks.

2.1 Statistical Methods

A popular statistical method of texture analysis is the use of Spatial Grey Level Dependence Matrices (SGLDMs) or co-occurrence matrices. Functions generated from these matrices have been used for various applications, Parkkinen et al, Starovoitov et al, and Oh et al [Parkkinen et al 90, Starovoitov et al 1998, Oh et al 1999] have used them to generate measures of periodicity held with textures. Jan and Hsueh [Jan and Hsueh 1998] provide a method of deriving the optimum window size for using matrices to generate a measure of periodicity. Various optimisations to the original co-occurrence functions initially proposed by Haralick et al [Haralick et al 1973] have been explored. Walker et al [Walker et al 1995] cross validates the functions to increase their performance. Laws [Laws 1980] uses them to validate a new suite of functions that can apply labels to texture, these being uniformity, density, coarseness, roughness, regularity, linearity, directionality, frequency and phase. Gotlieb and Kreyszig [Gotlieb and Kreyszig 1990] also generate a suite of texture descriptors using Haralick's original functions when considering the Brodatz [Brodatz 1966] series of textures. He and Wang [He and Wang 1991] derive texture units produced from Haralick's functions to classify Synthetic Aperture Radar (SAR) images. Practical aspects of the matrices are addressed by Clausi and Jernigan [Clausi and Jernigan 1998] by reducing the computational overhead of large co-occurrence matrices by using linked lists to extract data from the matrices. Jawahar and Ray [Jawahar and Ray 1996] propose using the co-occurrence functions in conjunction with first order statistics as a texture classifier. A similar approach is used by Al-Janobi [Al-Janobi 2001] to use matrix functions with texture descriptors to classify Brodatz images. The data held within the matrices can be used to find structure within textures as shown by Zucker and Terzopoulos [Zucker and Terzopoulos 1980] using a chi square test upon the matrices. Second order statistics are

not just limited to functions applied to spatial grey level dependence matrices, to detect orientated texture Chaudhuri et al [Chaudhuri et al 1993] use Hough transforms.

The use of first order statistics are also applied in texture analysis as depicted by De Natale [De Natale 1996] who examines the use of histograms.

Another method of statistical analysis is the use of cross-correlation. Lin et al [Lin et al 1997] use cross-correlation in conjunction with a Hough transform to detect periodicity within textures. Colour textures are considered by Paschos [Paschos 1998] who uses cross-correlation to produce chromatic correlations with luminance within a texture.

2.2 Model Based Methods

Model based methods are created from the ability to describe a texture or a method of synthesising it.

Markov Random Fields (MRFs) demonstrated by Cross and Jain [Cross and Jain 1983] provide a method of synthesising textures artificially. Suen and Healey [Suen and Healey 1999] apply this methodology to model and classify colour textures. Bhatt and Desai [Bhatt and Desai 1994] apply them in a different manor in the identification of textures when performing an image restoration function. Markov Random Fields are also used as a post processor validating the output of other classifiers as shown by Lorette et al [Lorette et al 2000].

Binary patterns are used as texture descriptors by Ojala and Pietikainen [Ojala and Pietikainen 1999] who use them to classify both Brodatz and real word imagery.

Fractals also provide an artificial method of producing textures and can be used in image analysis as shown by Pentland [Pentland 1984]. Kasparis et al [Kasparis et al 2001] develops the use of fractals further by operating them in conjunction with Gabor filters to segment Brodatz images.

Liu and Picard [Liu and Picard 1996] create a model of texture based upon periodicity, directionality and randomness. Lu [Lu 1998] improves upon this method by applying the three parameters to wavelet transforms. Another model is proposed by Pala and Santini [Pala and Santini 1999] where the shape of an object is used in conjunction with a sampled texture from a database to determine the objects identity.

As well as mathematical models portraying texture, descriptions of the make up of a texture are also available. Julesz [Julesz 1996] presents a theory with a Texton representing textural elements. The relationship between the Textons determines the appearance of the texture. Another model is suggested by Healey and Enns [Healey and Enns 1998] who offer Perceptual Texture Elements (Pexel), each pexel attribute being generated from height, density and regularity.

2.3 Signal Processing Methods

Spatial domain filters demonstrated by Reed and Wechsler [Reed and Wechsler 1990] have been proven to be capable of segmenting both synthesised and Brodatz textures. Unser [Unser 1986] also shows that their performance is comparable with spatial grey level dependence matrices and highlights that they give a computational saving.

Fourier domain filtering has been applied to numerous image processing problems including texture analysis as shown by Coggins and Jain [Coggins and Jain 1985]. Shen and Bie [Shen and Bie 1992] use frequency matrices to label twenty Brodatz textures. Hsu et al [Hsu et al 2000] applies multiple Fourier transforms at different resolutions to segment textures. Directional textures are identified by an approach presented by Tsai and Hsieh [Tsai and Hsieh 1999] where frequency components in the Fourier domain are isolated by a Hough transform.

Gabor filters also possess the ability to identify frequency components as shown by Clark and Bovik [Clark and Bovik 1987]. Strand and Taxt [Strand and Taxt 1994] evaluate Gabor filters with co-occurrence matrices and local frequency operators with favourable results. Ma and Manjunath [Ma and Manjunath 1996] model Gabor filters with wavelets to classify textures. To ensure optimum performance from Gabor filters Clausi and Jernigan [Clausi and Jernigan 2000] provide a review in their operation. Frye and Ledley [Frye and Ledley 2000] take a different approach by using a discrete cosine transform (DCT) sliding window across regions of interest to generate spatial frequency values.

Wavelets offer an alternative to Gabor filters as demonstrated by Chang and Kuo [Chang and Kuo 1993]. Wang et al [Wang et al 1998] proposes a new feature of texture coarseness using two dimensional wavelet transforms. Van de Wouwer et al [Van de Wouwer et al 1999] extend the use of wavelets into the colour spectrum by analysing each colour channel in turn. Portilla and Simoncelli [Portilla and Simoncelli 2000] use wavelet coefficients generated from synthesised textures to offer a new method of modelling textures.

2.4 Neural Network Methods

Applying neural networks to image processing tasks has been undertaken in a number of different modes, such as acting as pre and post processors for conventional methodologies. De Ridder et al [De Ridder et al 1998] shows neural networks to be capable of filtering images. Neural networks have also been used to classify the output data from spatial grey level dependence matrices [Fukue et al 1998, Muhamad and Deravi 1993, Oja and Valkealahti 1996, Visa 1990, Visa 1992, Valkealahti and Oja 1998]. The concept of modular neural networks introduced by Van Hulle and Tollenaere [Van Hulle and Tollenaere 1993] has given rise to their use as post-processors. Daugman and Ruiz del Solar [Daugman 1989, Ruiz del Solar 1998] use a neural network to classify the output of a bank of Gabor filters. Gabor filters were also used by Lampinen and Smolander, Ma and Manjunath, Mather et al, Raghu and Yegnanarayana [Lampinen and Smolander 1996, Mather et al 1998, Ma and Manjunath 1996, Raghu and Yegnanarayana 1997] to produce a system capable of indexing aerial photographs. Drimbarean and Whelan [Drimbarean and Whelan 2001] also use neural networks driven by Gabor filters to classify texture in colour images.

Bahlmann et al [Bahlmann et al 1999] uses self-organising maps to produce an automated quality control system to inspect textile seams. Karras et al [Karras et al 1998] also produced an automated image recognition system but this time the self-organising map classifies wavelet coefficients. Daniell et al along with Greenberg and Guterman [Daniell et al 1992, Greenberg and Guterman 1996] have both used multi-layer neural networks capable of target recognition.

Self-organising neural networks have been demonstrated to perform texture segmentation by Wang et al and Yoshimura and Oe [Wang et al 1996, Yoshimura and Oe 1999], with Murtagh [Murtagh 1995] using them to classify large astronomical star catalogues. Raghu et al, Biebelmann et al and Bhattacharya et al [Raghu et al 1995, Biebelmann et al 1996, Bhattacharya et al 1997] have also demonstrated the ability of a modular neural network to classify texture. Goltsev and Wunsch [Goltsev and Wunsch 1998] produced another modular neural network based upon texture class to classify textures within an image.

The ability of neural networks in the classification of ground cover from satellite images has been demonstrated by Augusteijn et al [Augusteijn et al 1995] via the use of feed-forward cascade correlation neural network architecture. Bischof et al, Bruzzone et al, Serpico et al [Bischof et al 1992, Bruzzone et al 1997, Serpico et al 1996] apply error back-propagation in multi-layer networks to a similar task when classifying regions of Landsat images.

Neural methodologies are not just limited to the visible spectrum of data acquisition. Stewart et al [Stewart et al 1994] shows self-organising maps that can classify

millimetre-wave radar returns. Inggs and Pasquariello et al [Inggs and Robinson 1999, Pasquariello et al 1998] also use self-organising maps to classify radar returns of ships. Ghinelli and Bennett [Ghinelli and Bennett 1998] use synthetic aperture radar (SAR) to produce a crop monitoring system. Marana et al [Marana et al 1997] makes use of Hough space as an input to a Kohonen [Kohonen 1998] network to classify textures.

Chapter 3 Texture Analysis

A brief overview of texture, texture is presented in this chapter.

Section 3.1 provides a breakdown on the definition of texture, how texture may be represented within images.

Sections 3.2 through 3.6 identify some of the common methodologies that currently exist for texture analysis in the image processing realm.

Section 3.7 offers a comparison of the different methodologies and identifies a benchmark for use in Chapter 6 when evaluating the performance of the hybrid neural network architecture.

3.1 Texture

Haralick [Haralick 1979] describes texture held within an image by the number and types of its primitives and the spatial organisation or layout of its primitives. Sklansky [Sklansky 1978] states that a region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying or approximately periodic.

Rao [Rao 1990] expands upon this by offering a taxonomy for texture description that can be illustrated through the Brodatz [Brodatz 1966] collection:-

- **Strongly ordered (Deterministic) textures** such as that shown in Figure 3.1 have primitive elements in their composition such as the bricks in the brick wall.
- **Weakly ordered (Structured) textures** can be seen to possesses some repetitive structure, however this can be intermittent as shown in Figure 3.3 where the knot breaks up the grain. The composition of textures may also be made up from elements such as the pebbles shown in Figure 3.2.
- **Disordered (Stochastic) textures** possess neither repetition nor orientation in their structure. Figure 3.4 shows the random mottled effect of the pressed cork.

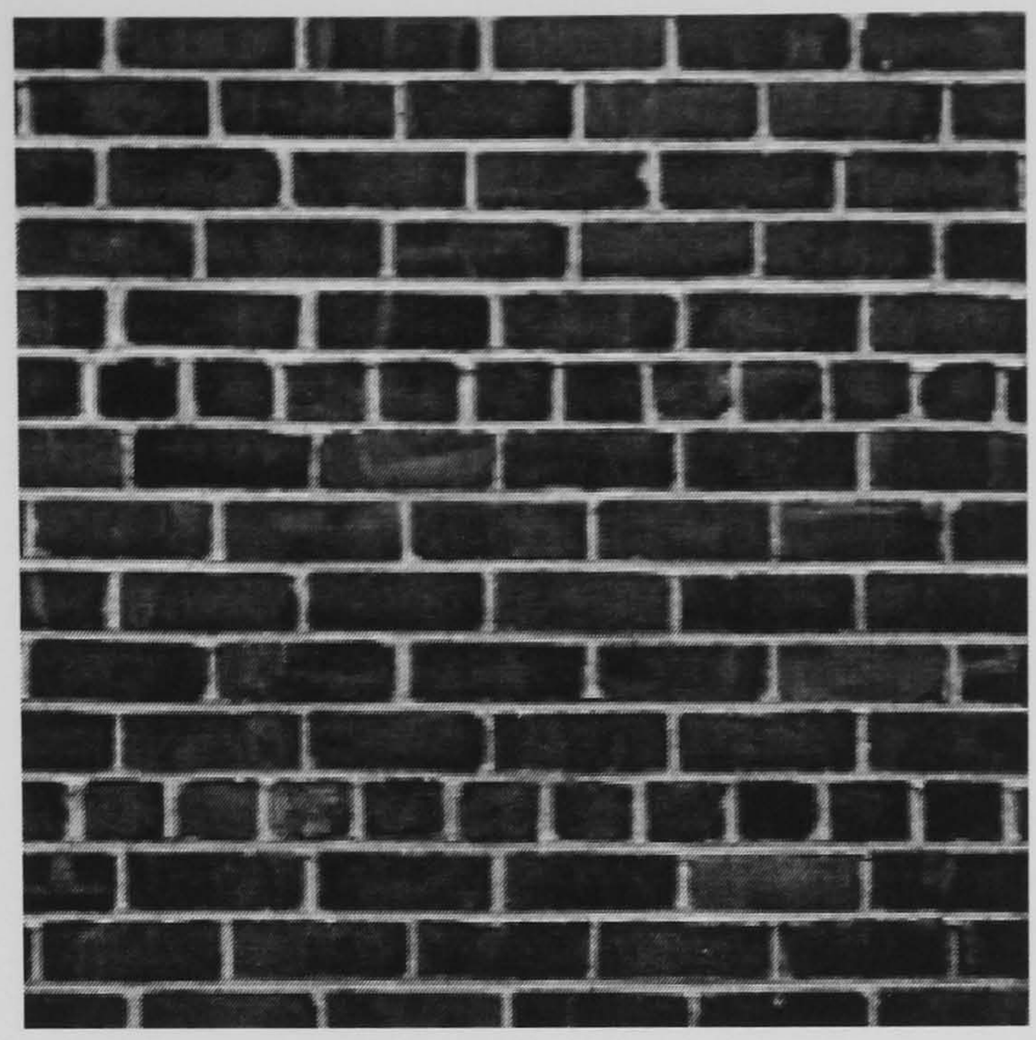


Figure 3.1 Brick Wall

Figure 3.2 Pebbles

Figure 3.3 Tree Stump

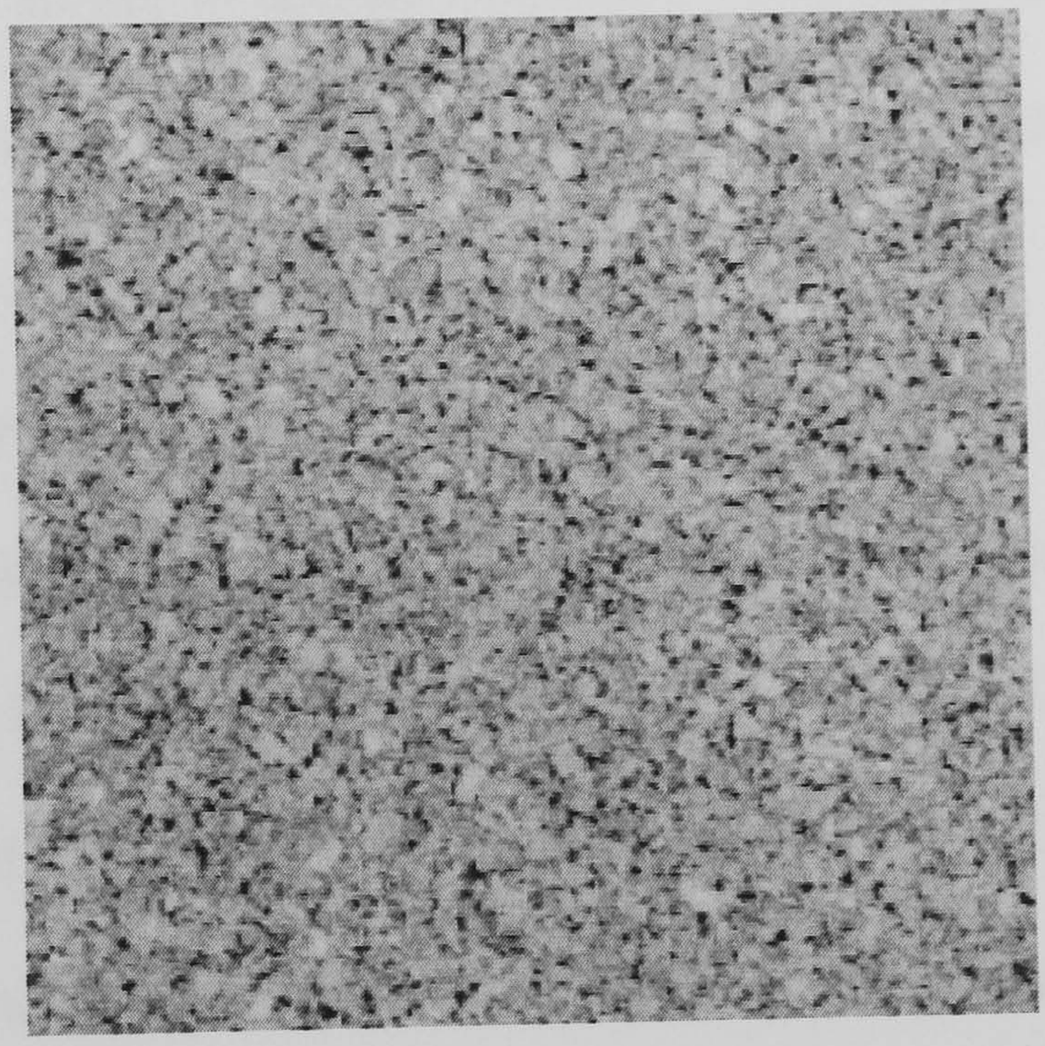


Figure 3.4 Pressed Cork

3.2 Common methods of texture analysis

Various surveys of different texture analysis methods have been performed [Conners and Harlow 1980, Haralick 1979, Pal and Pal 1993, Van Gool et al 1983, Weszka et al 1976] identifying numerous routes that can be used. Some of the popular methods that are often used as benchmarks for image processing research are: Fourier Spectrum, Wavelets, Cross-correlation and Second Order Statistics.

- Fourier Spectrum; by converting an image into the Fourier spectrum (Amplitude vs. Frequency) Weszka et al [Weszka et al 1976] showed ring filters applied in this domain before conversion back to the visible spectrum can classify features from Landsat images.
- Wavelets offer the ability to record frequency verses time at multiple resolutions. Laine and Fan [Laine and Fan 1993] along with Mallat [Mallat 1989] have applied wavelets in texture classification applications.
- Cross-correlation involves running a sample texture across an image, and using a correlation coefficient to indicate matches between samples, as documented by Haralick [Haralick 1979] and reviewed again by Sonka et al [Sonka et al 1993].
- Spatial Grey Level Dependence Matrices (SGLDMs) or Co-occurrence matrices, [Haralick 1973], a method of texture analysis where matrices are created to represent distribution of pixels throughout an image.

3.3 Fourier Spectrum

Most texts on image processing such as James [James 1987], give the algorithms for Fast Fourier Transforms (FFTs). A benchmark image processing application Matlab [Matlab 2000] for IBM compatible personal computers provides the capability of both forward and inverse FFTs on bitmap images. Using this software package, Fast Fourier Transform operations upon images can be visualised. Take the Brodatz texture D20 (French Canvas) Figure 3.5. A Fast Fourier Transform performed upon this image results in Figure 3.6.

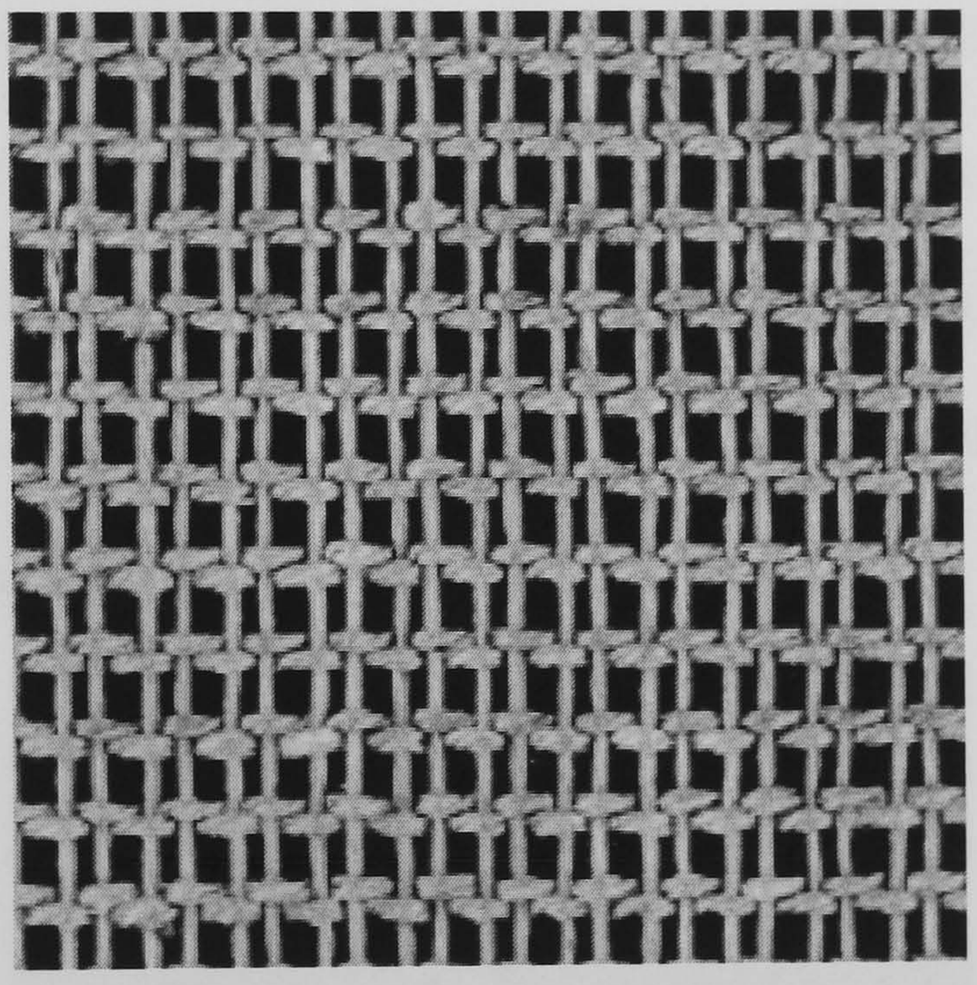


Figure 3.5 D20 French Canvas

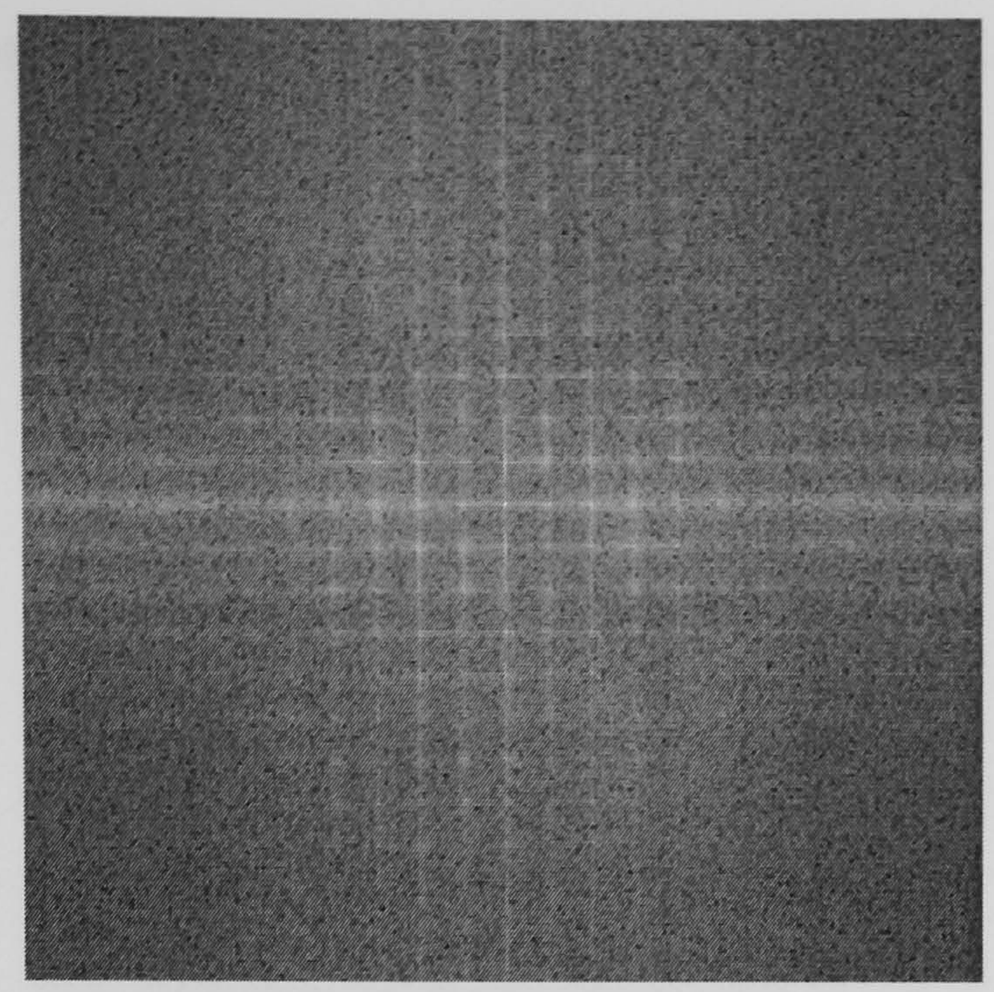


Figure 3.6 Fast Fourier Transform on D20

This representation of texture in the Fourier domain can be used to segment images, as different textures will have different representations in the Fourier domain. An example of this process can be seen by manipulating the image in Figure 3.7. This image is constructed from four Brodatz textures. Top left is D20 (French Canvas x 4), top right is D55 (Straw Matting), bottom left is D105 (Cheesecloth) and bottom right is D21 (French Canvas reduced by ½).

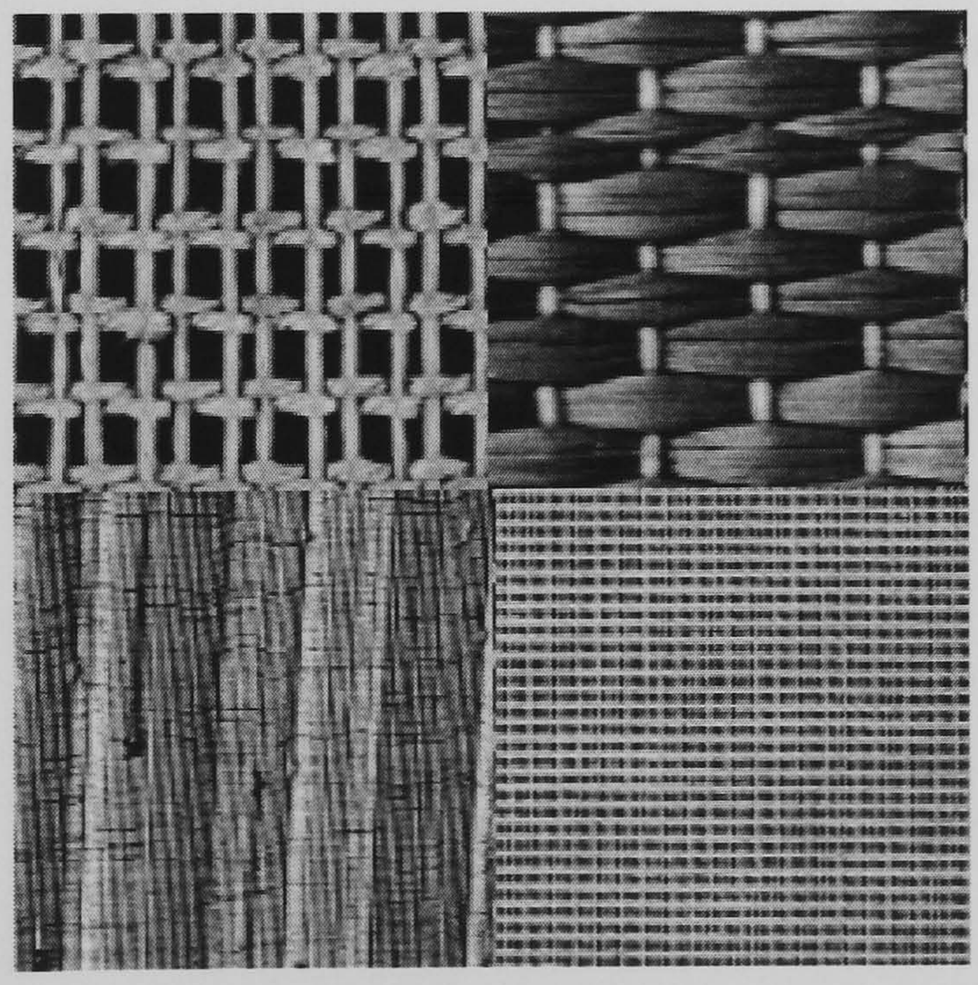


Figure 3.7 Collage of four Brodatz Textures

A Fast Fourier Transform is performed on Figure 3.7 giving the image Figure 3.8.

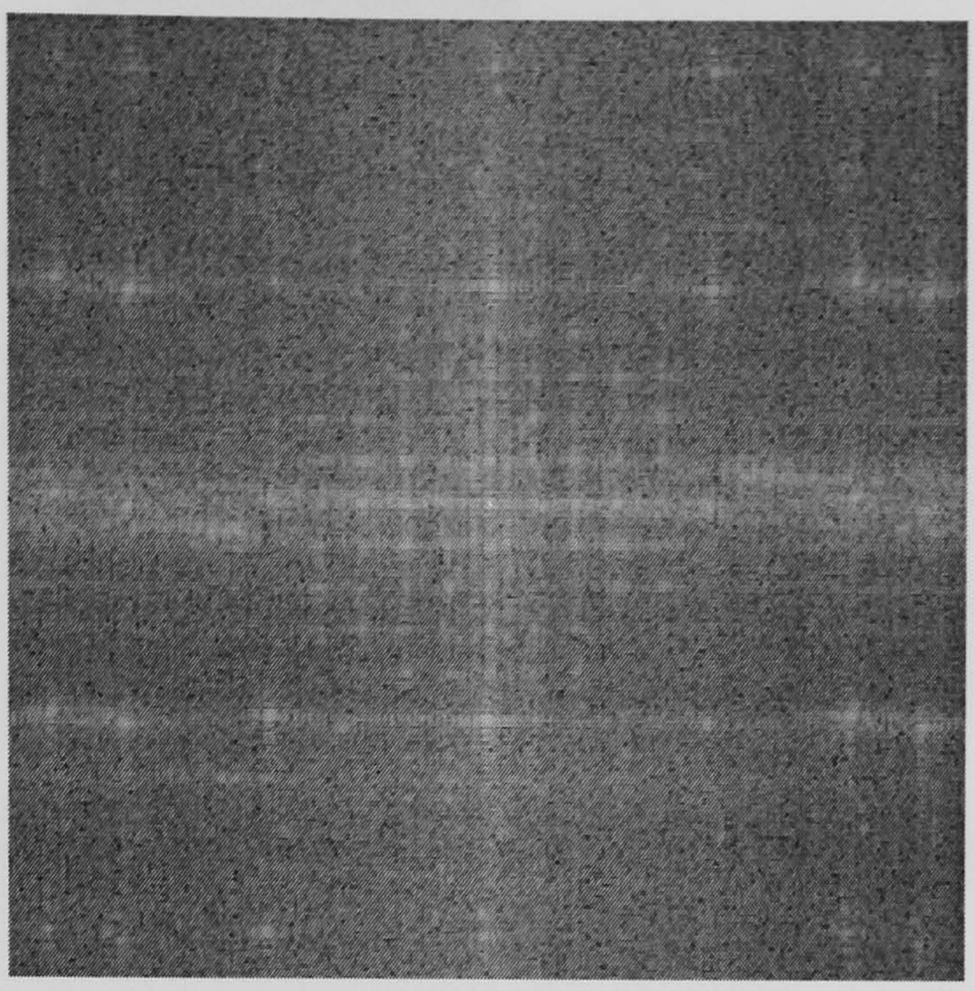


Figure 3.8 FFT of Brodatz collage

This time the Fourier domain contains elements of all four textures from the Brodatz collage image Figure 3.7. To segment a single texture out of the image using its Fourier domain representation is possible by multiplying Figure 3.8 with a Fourier transform of the wanted texture to be segmented, in this case D56. Then an inverse Fast Fourier Transform is applied to it to re-create an image, this is thresholded to create a binary mask Figure 3.9.

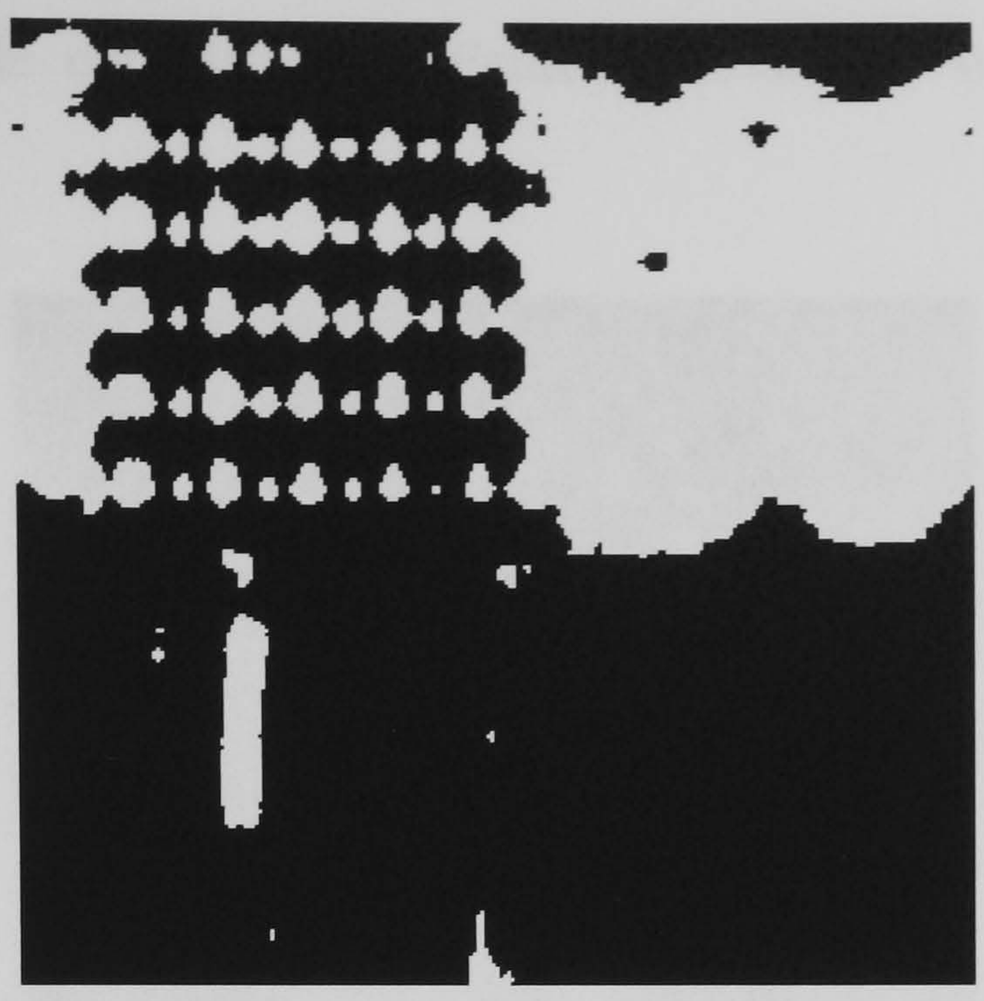


Figure 3.9 Inverse Fast Fourier Transform

Applying an AND function to Figure 3.7 and Figure 3.9 gives a segmented image with the texture D56 remaining.

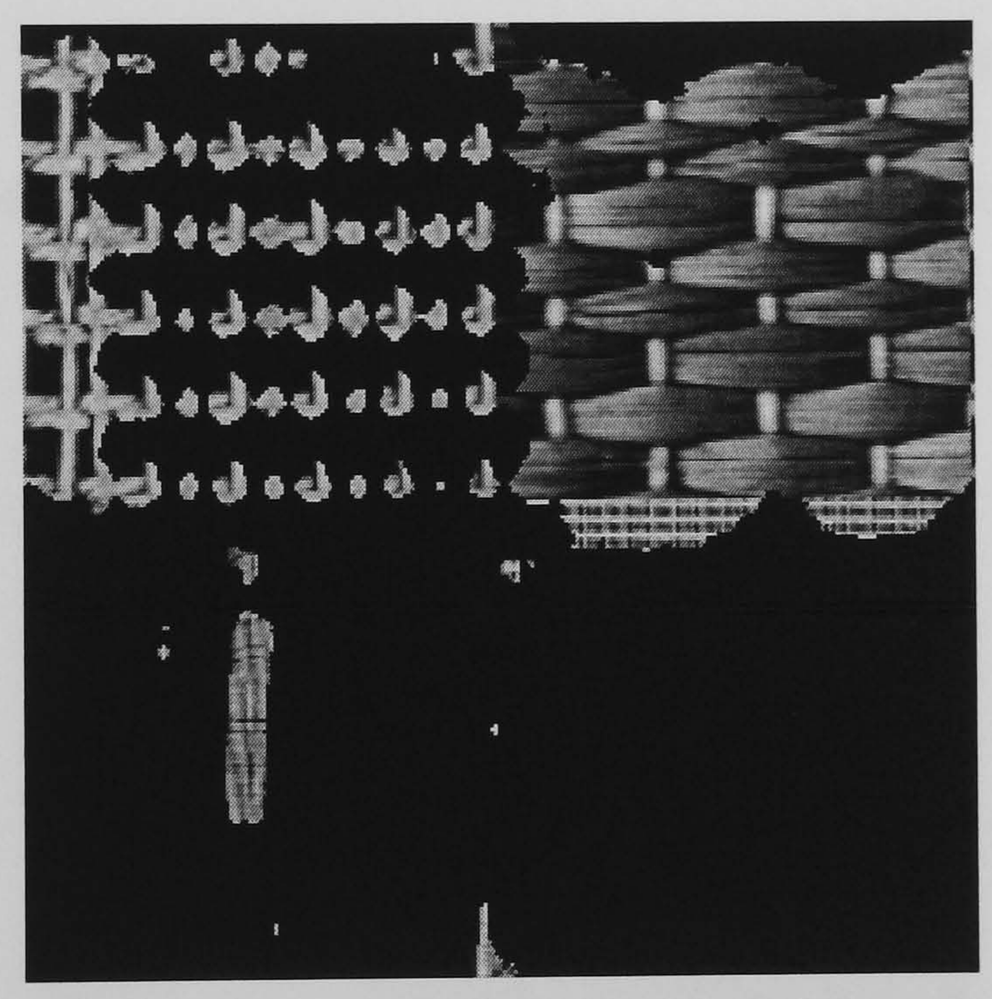


Figure 3.10 Fourier Segmented Image of Brodatz Collage

The following example considers Fast Fourier Transforms when using real world images Figure 3.11.

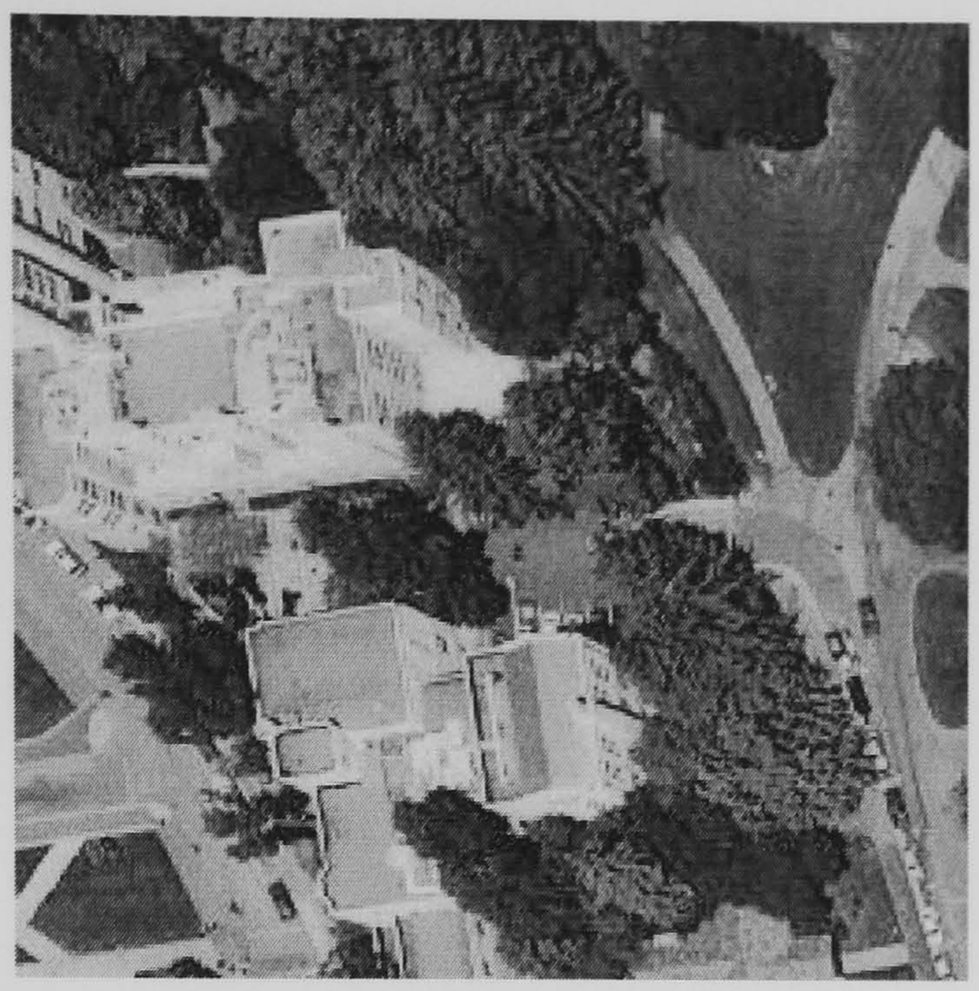


Figure 3.11 Aerial Image 1

Multiplying an FFT of Figure 3.11 by an FFT of a small sample image of trees and performing an inverse FFT gives the image in Figure 3.12. Matlab has the ability of zero padding the smaller image to the same size as Figure 3.11 to allow this multiplication.

Figure 3.12 FFT of Aerial Image 1

Thresholding this image generates Figure 3.13

Figure 3.13 Threshold FFT of Aerial Image 1

Performing an AND function on Figure 3.11 and Figure 3.13 gives the segmented image below Figure 3.14

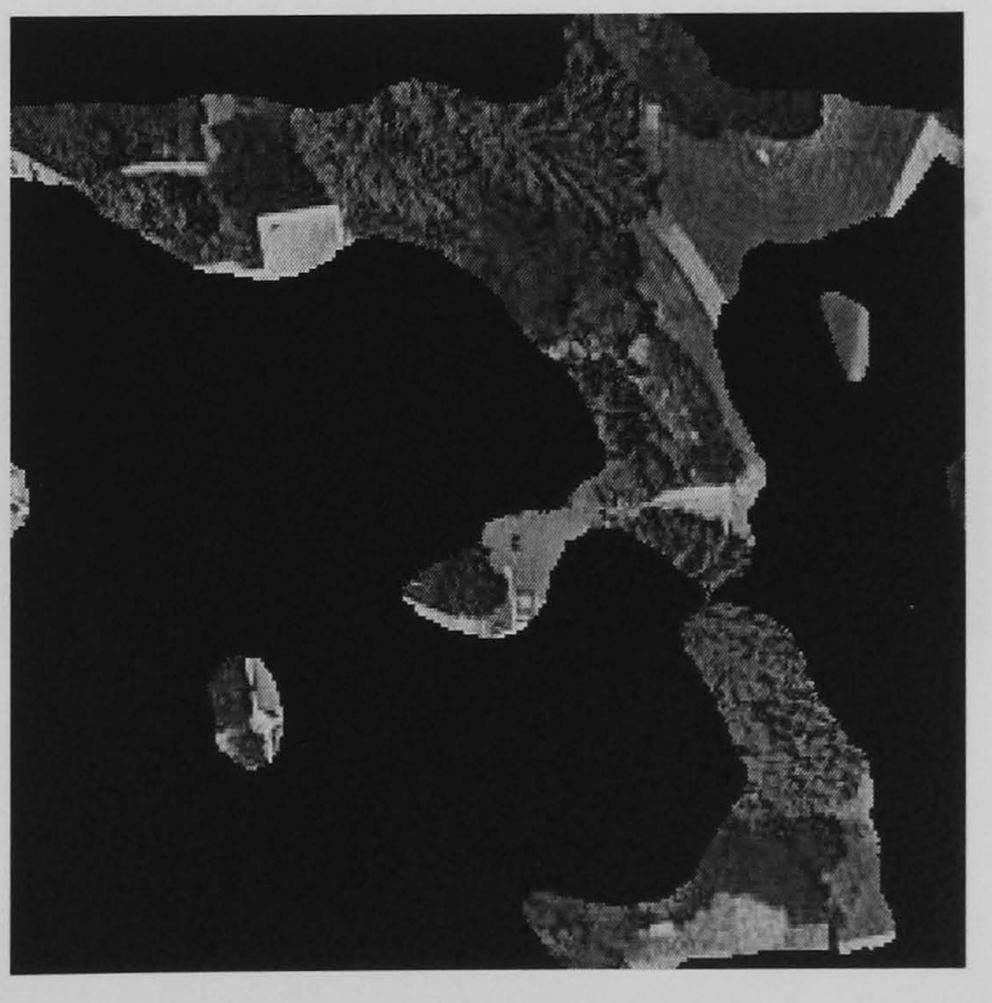


Figure 3.14 Segmented Aerial Image 1 by FFT

3.4 Wavelets

The application of wavelets is based upon scale rather than frequency as with Fourier analysis. Whereas Fourier analysis relies upon breaking a signal up into a series of sine waves, wavelet analysis breaks up the signal into a series of shifted and scaled waveforms. This analysis can take the form of pyramid / tree structured transforms. The pyramid wavelet structure breaks decomposes the image into a set of frequency channels.

If the Brodatz texture D20 (Figure 3.5) is considered again, then the following wavelet analysis gives Figure 3.15.

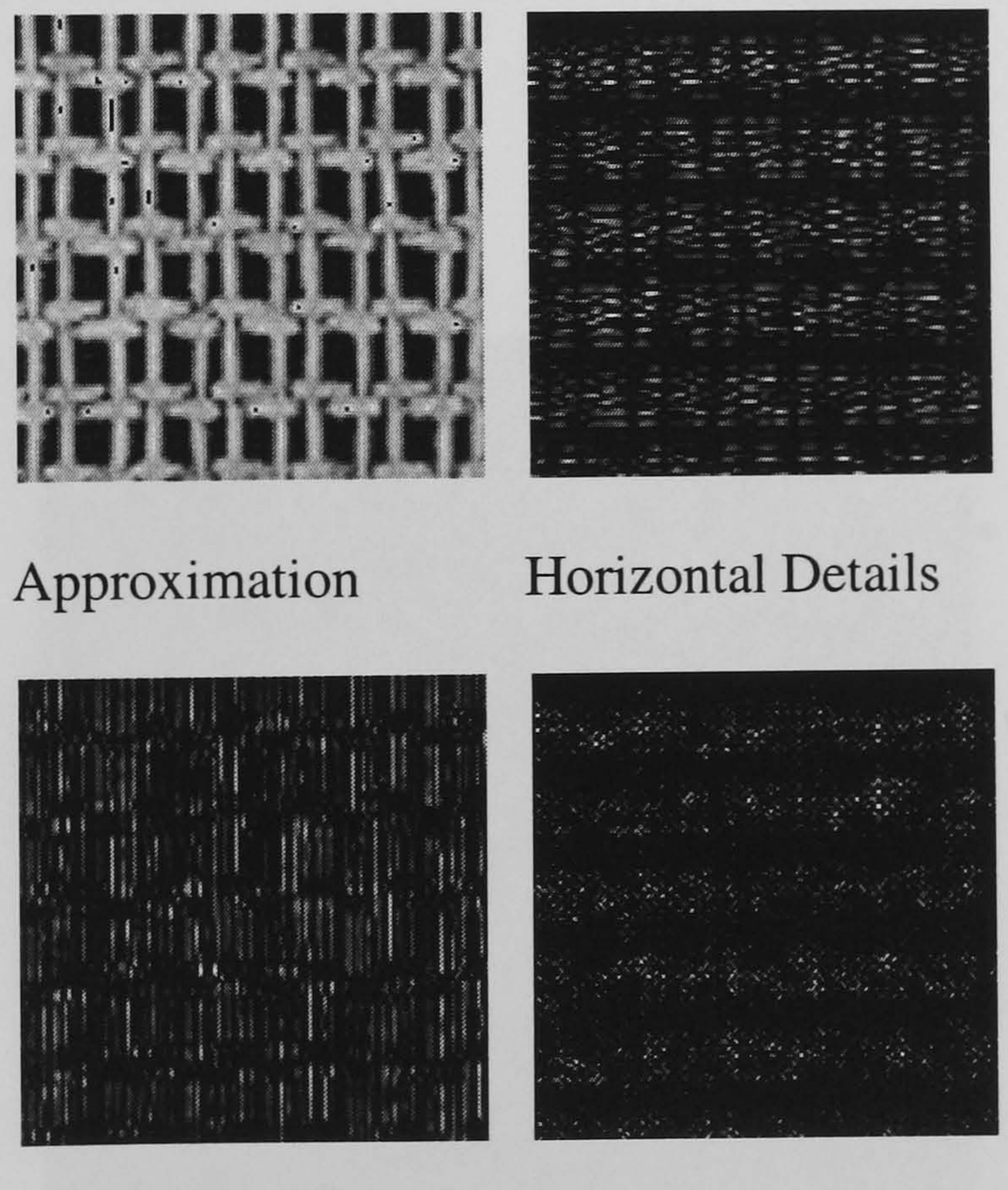


Figure 3.15 Discrete Wavelet Analysis on D20

The reconstructed images in Figure 3.15 were generated from the wavelet toolbox in Matlab [Matlab 2000] and show that the decompositions differ greatly for horizontal, vertical or diagonal properties. A Daubechies [Daubechies 1992] 8 tap wavelet was used in the analysis as this was demonstrated by Chang and Kuo [Chang and Kuo 1993] to give the optimum results when considering texture classification. These wavelet reconstructions can be combined to make masks capable of segmenting textures held within images.

Considering the Brodatz collage image (Figure 3.7) again, a wavelet reconstruction is made from vertical and diagonal elements Figure 3.16. The choice of elements is made by manual inspection of all the elements as which isolates the wanted texture. This time with a threshold function applied to the level one wavelet coefficients before reconstruction, Mallat [Mallat 1989]. In this case all negative wavelet coefficients are zeroed.

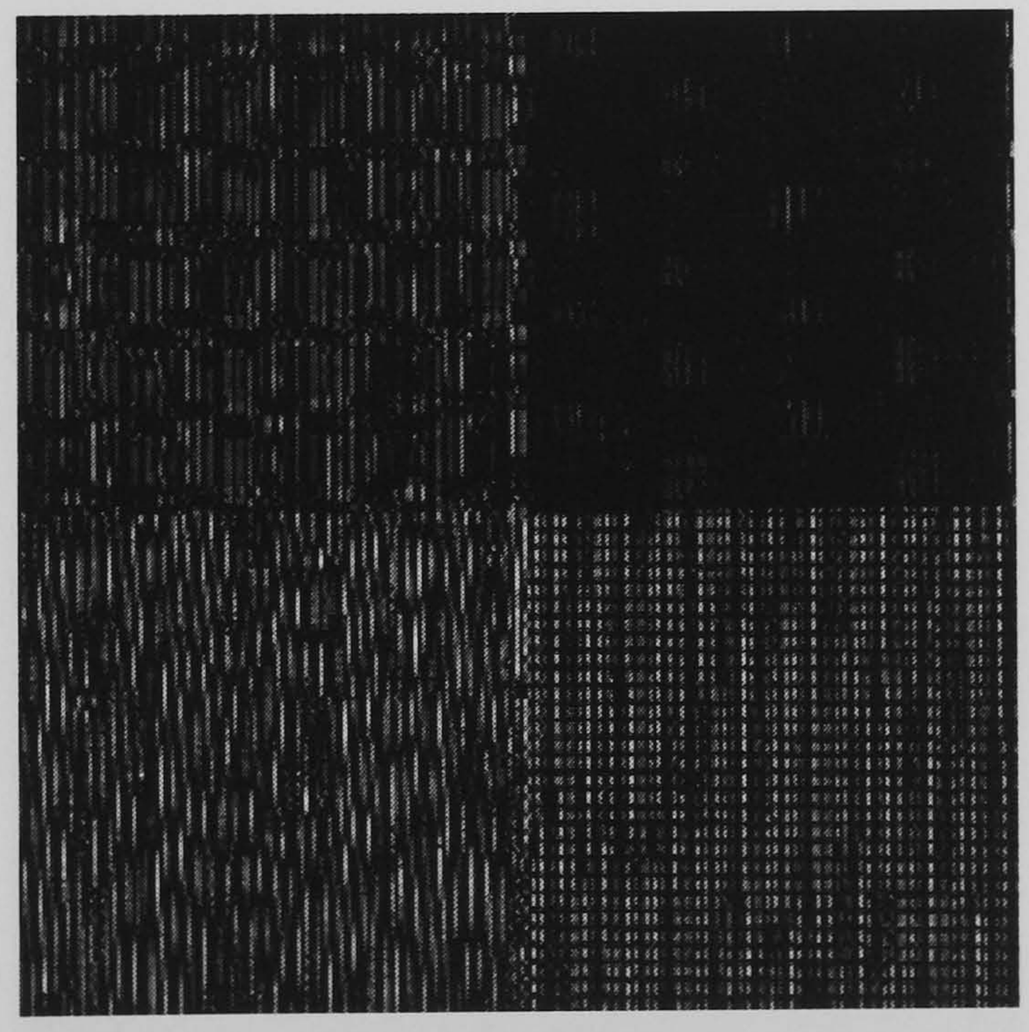


Figure 3.16 Wavelet Reconstruction upon Brodatz Collage using Vertical and Diagonal Detail Coefficients

This image is converted to a binary image Cziria et al [Cziria et al 2001], with a dilate and median cut operation applied to generate the mask in Figure 3.17.

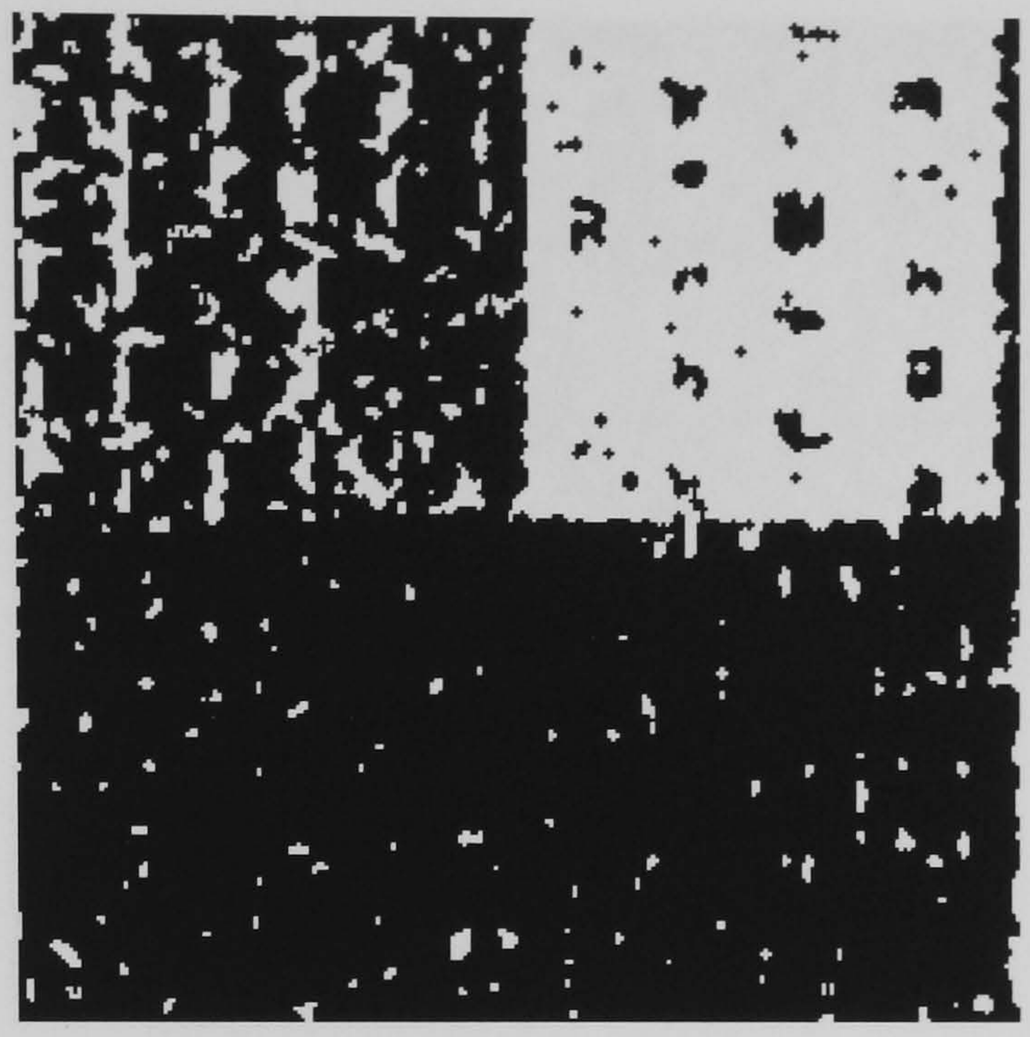


Figure 3.17 Wavelet Segmenting Mask for Brodatz Collage

Applying this mask to the Brodatz collage image gives Figure 3.18.

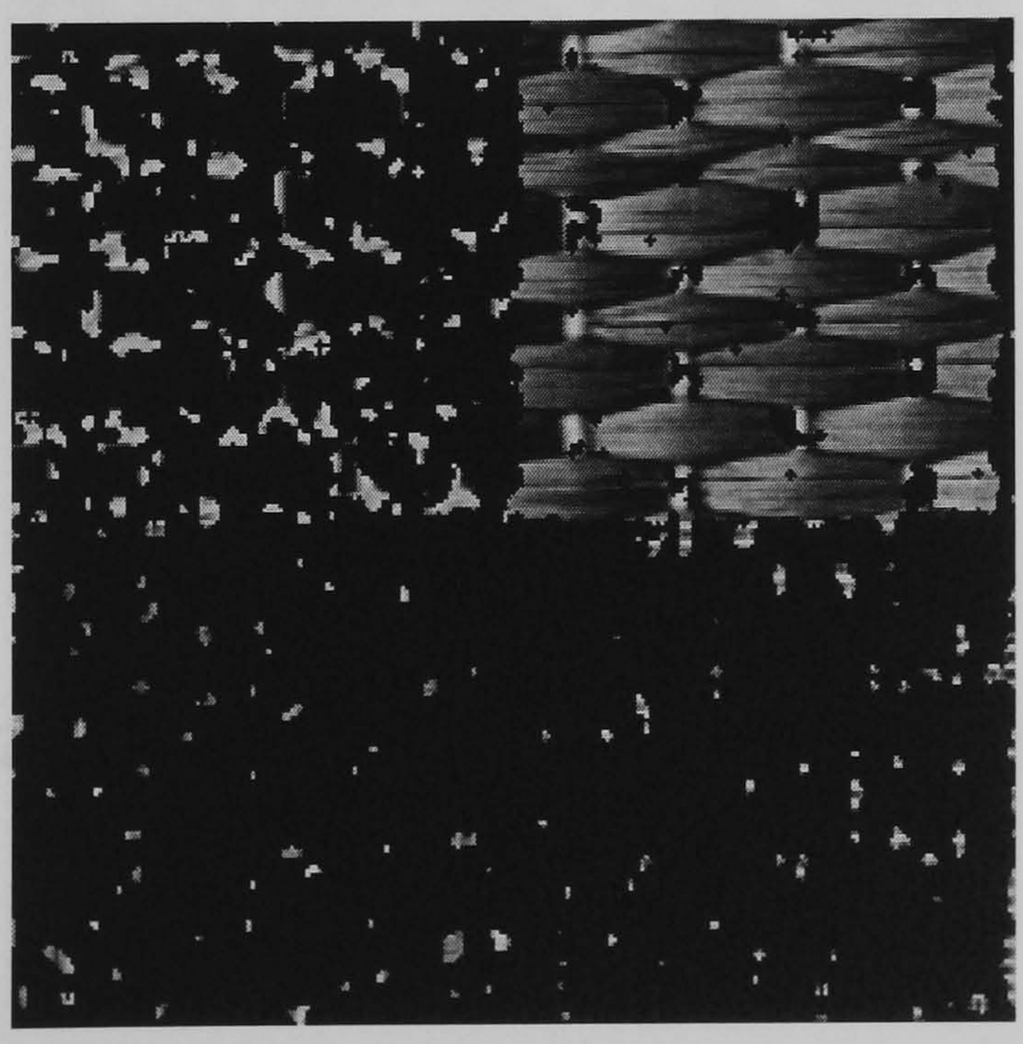


Figure 3.18 Wavelet Segmented Brodatz Collage

Applying a similar process to Aerial Image 1 (Figure 3.11) but this time reconstructing using horizontal and vertical components results in Figure 3.19 and Figure 3.20.

Figure 3.19 Wavelet Reconstruction of Aerial Image 1

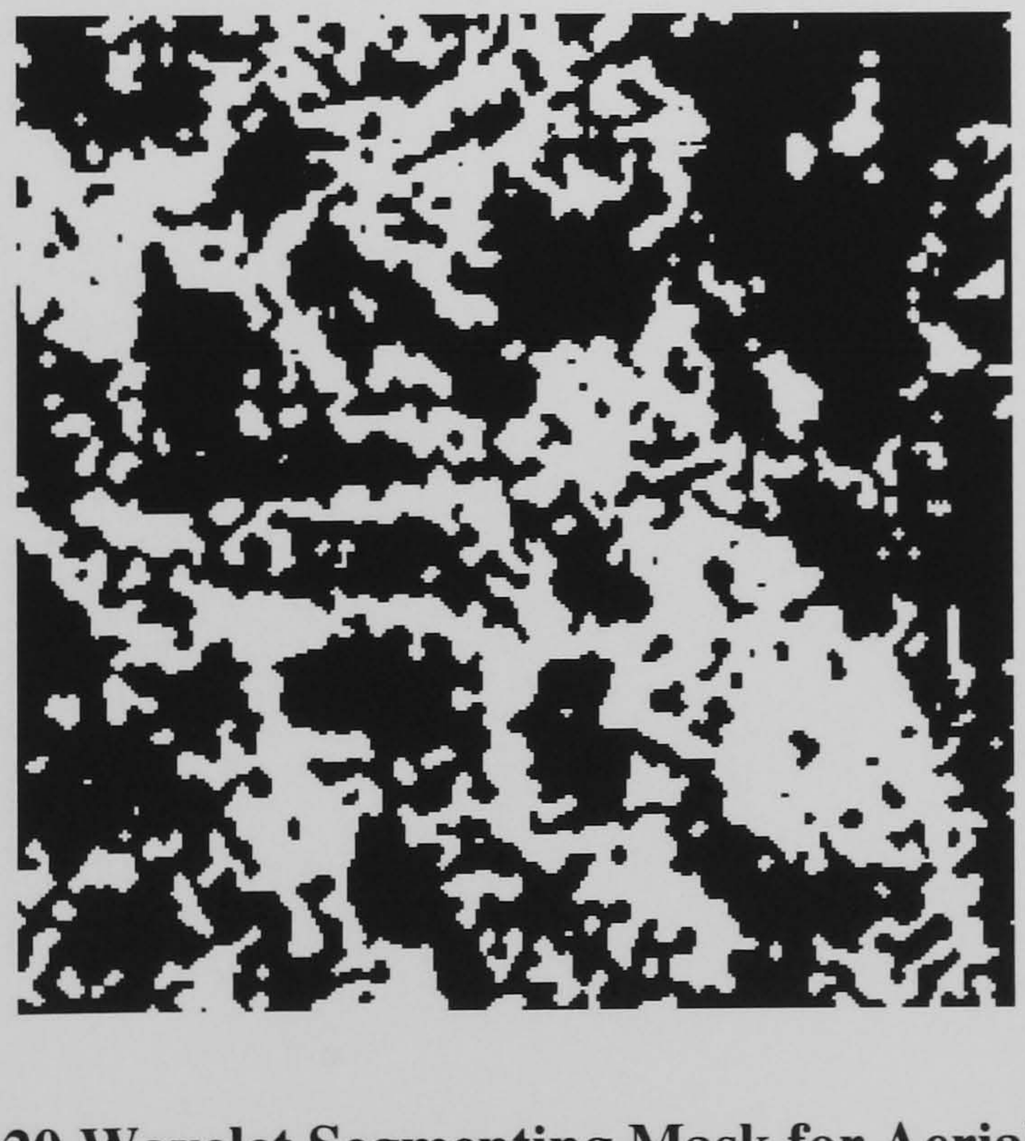


Figure 3.20 Wavelet Segmenting Mask for Aerial Image 1

Using the mask from Figure 3.20 upon Aerial image 1 results in Figure 3.21.

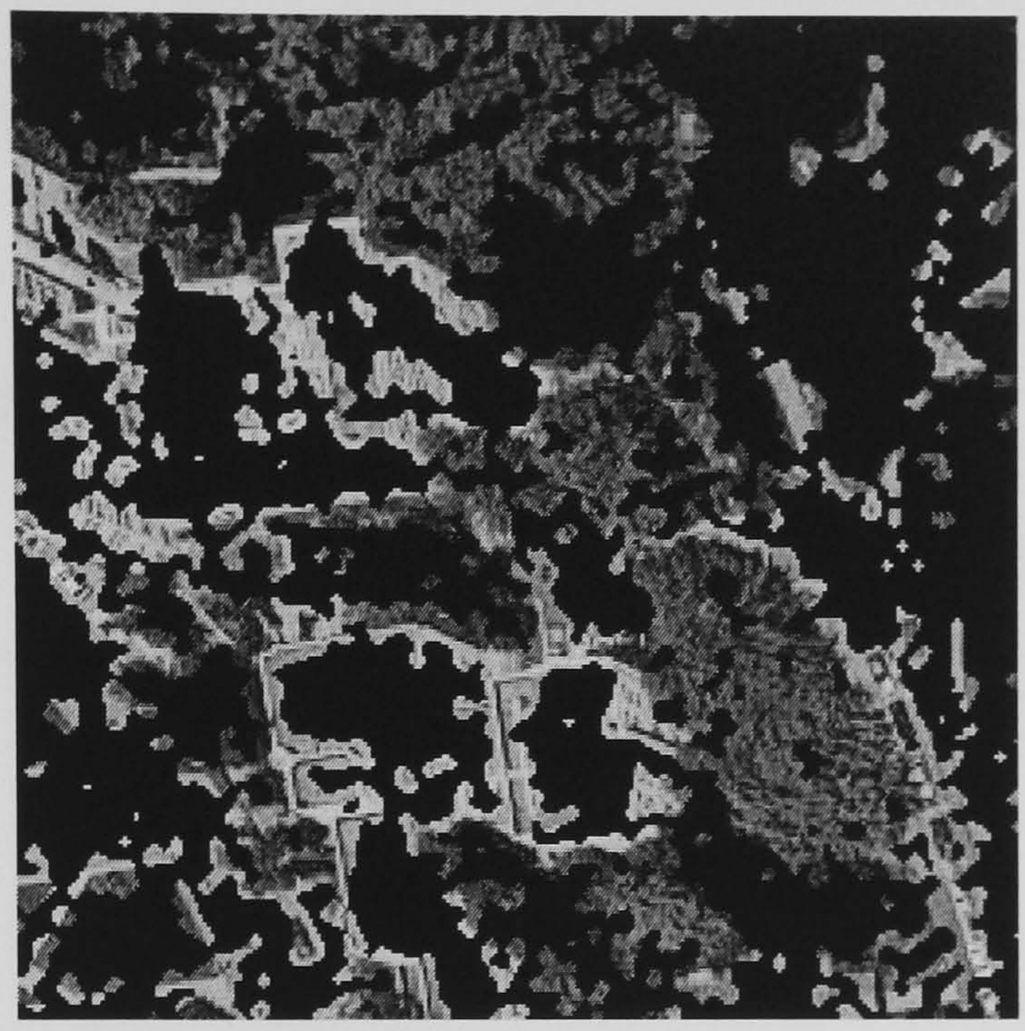


Figure 3.21 Wavelet Segmented Aerial Image 1

3.5 Cross-correlation

Cross-correlation provides a quick and simple method of detecting texture within an image. A section of the image that contains a texture that needs to be identified or isolated is selected and placed into a mask. This mask is then passed across the image and the correlation coefficient (r) is calculated from (2.4).

$$r = \frac{N\sum_{i=1}^{N}(a_i b_i) - G_a G_b}{\sqrt{\left(\left(N\sum_{i=1}^{N}(a_i^{\,2}) - G_a^2\right)\left(N\sum_{i=1}^{N}(b_i^{\,2}) - G_b^2\right)\right)}} \tag{2.4}$$

Note that a elements come from the image mask and the b elements from the comparison mask. N is the number of elements in the mask.

$$G_a = \sum_{i=1}^{N} a_i \qquad \text{(Sum of } a \text{ elements)}$$

$$G_b = \sum_{i=1}^{N} b_i \qquad \text{(Sum of } b \text{ elements)}$$

Using this equation it is possible to find matches between textures within an image. As the mask is run across the image, the correlation coefficient is calculated. If it approaches the value of one, then a near match is found (an exact match produces one). After the entire image has been processed, this coefficient is examined. The distance between the peaks gives an indication of any periodicity that may be present with the image. Appendix B expands on the implementation of this process in software and some of the limitations encountered, as well the methods applied to produce the following images.

This method of texture analysis is ideal for applications where regular periodicity within a texture is to be considered. This is demonstrated by using the Brodatz collage image Figure 3.7. Applying the cross-correlation algorithm to this image whilst using the D56 image to populate the comparison sample mask results in Figure 3.22.

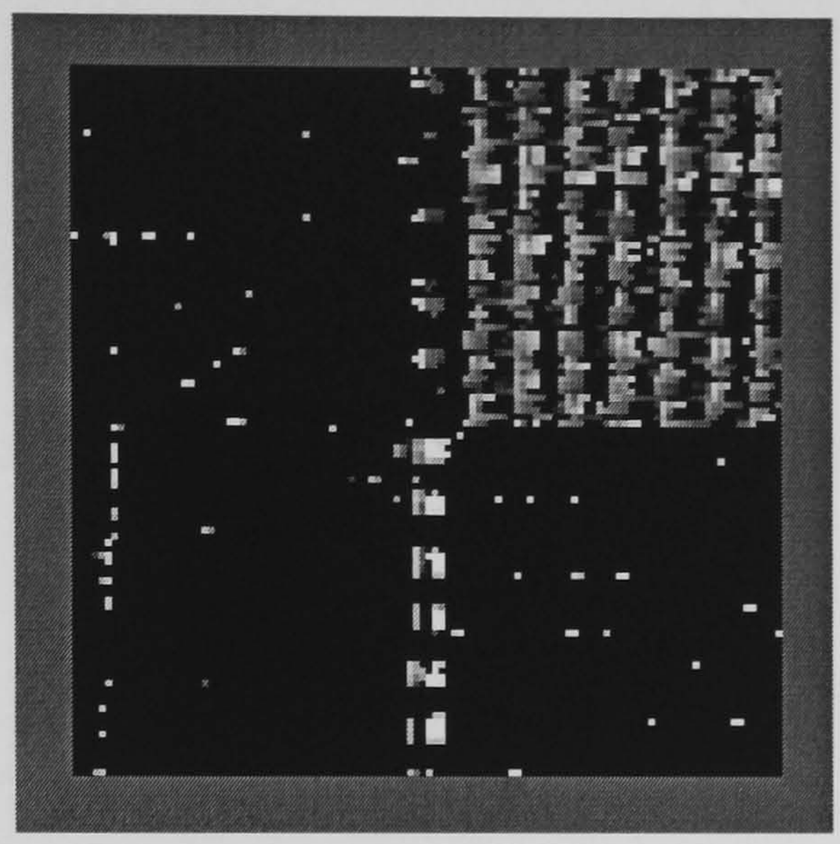


Figure 3.22 Cross-correlated collage image

The algorithm correctly indicates the presence of the D55 texture in the upper right hand portion of the image.

Figure 3.23 shows cross-correlation with a sample of trees in the comparison mask when applied to Aerial Image 1 Figure 3.11. Any peaks from the cross-correlation function indicated a match with the trees. The new image was created as the mask was passed across the sample image. Any matches above a threshold value (0.6) were copied to the new image, non-matches were replaced with a black pixel.

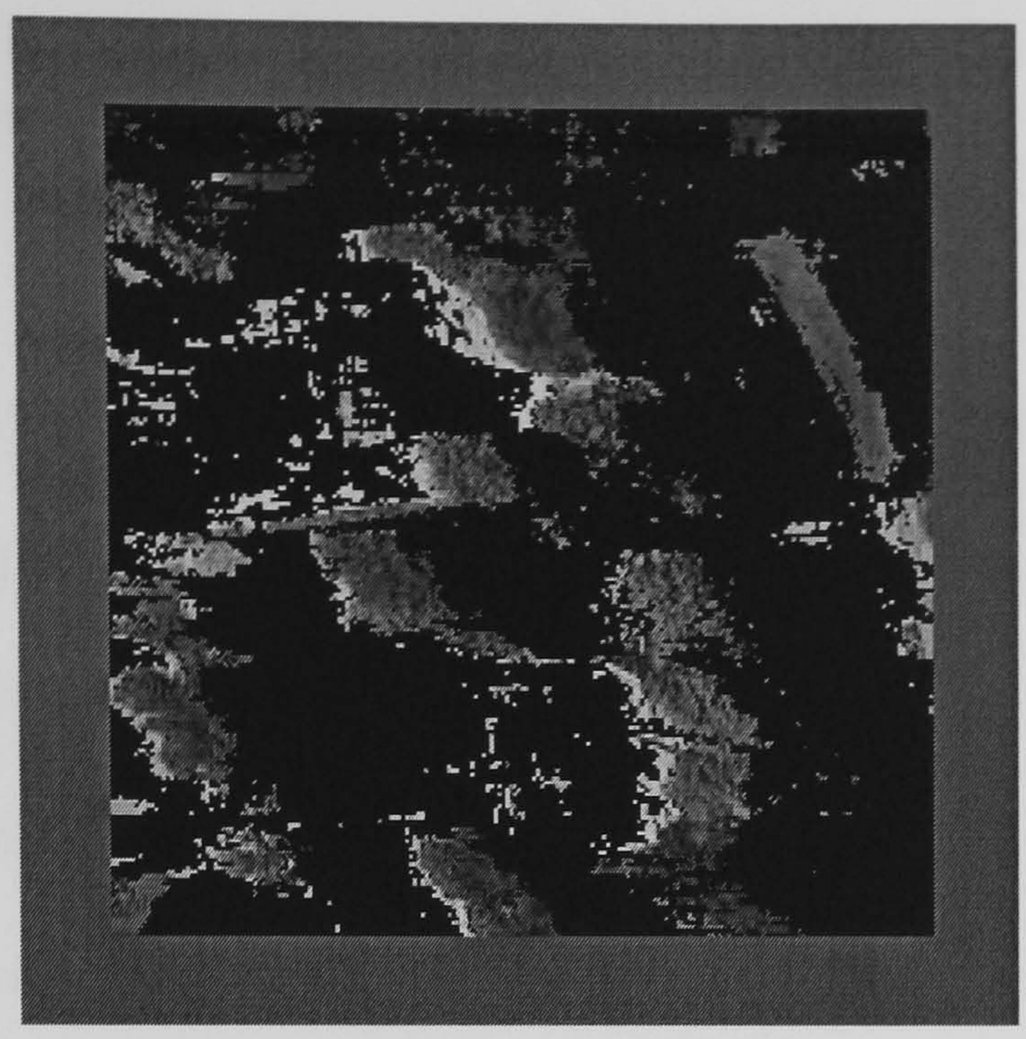


Figure 3.23 Cross-correlation of Aerial Image 1

3.6 Spatial Grey Level Dependence Matrices

Baraldi and Parmiggiani, Weszka et al [Baraldi and Parmiggiani 1995, Weszka et al 1976] have proved spatial grey level dependence matrices capable of classifying Landsat images, with Marceau et al [Marceau et al 1990] performing similar work upon SPOT images. Haddon and Boyce [Haddon and Boyce 1993] have also proven them capable of operating on aerial images.

3.6.1 A Review of Spatial Grey Level Dependence Matrices

Spatial grey level dependence matrices can be specified in a matrix of relative frequencies. The number of neighbouring pairs of pixels $I(x_1, y_1)$ and $I(x_2, y_2)$, one with a grey level of i, and the other with a grey level of j separated by a distance d and angle θ can be recorded as $S_{d,\theta}(i, j)$.

An entryin a matrix I at position $I(i, j)$ gives the number of times within an image that grey scale i is orientated with j as shown :-

$$I(x_1, y_1) = i \text{ and } I(x_2, y_2) = j$$

Of course the number of angles and sampling distances, if not limited, can cause a huge computational overhead. Conners et al [Conners et al 1984] used four angles of 0, 45, 90 and 135 degrees and eight sampling distances of 1, 2, 4, 6, 8, 12, 16 and 20 pixels.

Angles between 180 and 360 degrees are not considered, as pixel pair orientations are calculated in a forward and reverse motion.

E.g. If pixel $I(x_1, y_1)$ is orientated with another pixel $I(x_2, y_2)$ at 90 degrees this is equivalent to $I(x_2, y_2)$ being orientated to $I(x_1, y_1)$ by 270 degrees.

Functions can be defined to extract information from the matrices, Haralick [Haralick 1979] identifies several functions :-

- Homogeneity (Energy) equation (2.5.1) is the squares of all the elements in the matrix summed. It measures textural uniformity such as pixel pair repetitions. This enhances any large values from the matrix and highlight periodicity and direction of a texture.

$$Energy_{d_1\theta} = \sum_i \sum_j S^2{}_{d,\theta}(i, j) \qquad (2.5.1)$$

- Contrast (Inertia) gives the grey level range in the region that is being sampled. It provides an indication of any spatial frequencies. So using equation (2.5.2), the contrast would have a small value if there are a small number of grey levels are used.

$$Contrast_{d_1\theta} = \sum_i \sum_j (i - j)^2 S_{d,\theta}(i, j) \qquad (2.5.2)$$

- Entropy (2.5.3), this parameter measures disorder within the texture. If there are a large number of small values in the matrix then the entropy becomes large. It tends to be inversely proportional to the energy function.

$$Entropy_{d,\theta} = \sum_i \sum_j S_{d,\theta}(i,j)\log(S_{d,\theta}(i,j)) \qquad (2.5.3)$$

- Variance (2.5.4) increases when the grey levels differ from their mean value (*u*). It provides a similar measure to that of contrast, with both functions possessing similar behaviour.

$$Variance_{d,\theta} = \sum_i \sum_j (i-u)^2 S_{d,\theta}(i,j) \qquad (2.5.4)$$

The following examples show how the energy and contrast functions can give information on the composition of the texture being sampled.

Take an image sample with a grey level range of 0 to 9 pixels with dimensions of 10 by 8 pixels, overleaf:-

2	2	2	8	8	2	2	2	7	7
2	2	8	8	2	2	2	2	7	7
2	8	8	8	2	2	2	7	7	7
8	8	8	2	2	2	7	7	7	2
8	8	8	8	2	7	7	4	4	4
8	8	2	2	2	7	7	7	2	2
8	2	2	2	7	7	7	2	2	8
2	2	2	7	7	7	2	2	8	8

Table 3.1 Image Sample A

If θ is 0° and d =1 when sampling, the following co-occurrence matrix is created:-

i \ j	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	42	0	0	0	0	12	12	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	4	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	12	0	1	0	0	26	0	0
8	0	0	12	0	0	0	0	0	22	0
9	0	0	0	0	0	0	0	0	0	0

Table 3.2 Co-occurrence Matrix Sample A

If another sample image is considered, Table 3.3. This time it has no texture.

5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5

Table 3.3 Image Sample B

If θ is 0° and d =1 when sampling, the following co-occurrence matrix is created: -

i \ j	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	144	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Table 3.4 Co-occurrence Matrix Sample B

	Energy	Contrast	Entropy	Variance
Sample A	3518	738	3751	1074
Sample B	20736	0	310	36

The contrast value highlights the fact there is no texture present within the sample.

Some practical examples of spatial grey level dependence matrices are demonstrated by applying the energy and contrast functions on the aerial image shown in Figure 3.11. An operation was performed to segment out all the trees of the image. Higher values of energy and contrast were used to segment out the trees / woodland, whereas the buildings would have lower values of contrast and would be left remaining in the image. Normalising the spatial grey level dependence matrices to contain values that lie between 0 and 1 the parameters for segmentation took the form of:-

Energy Function: Band Pass 0.1 to 1.00 AND Contrast Function: Band Pass 0.0 to 0.1

The image was sampled in blocks at multiple resolutions to produce different sized SGLDMs for a performance comparison.

Figure 3.24 shows the results of being sampled at 4 by 4 pixel blocks with θ at 0°, 45°, 90°, 135° and d at 1 and 2 pixels.

Figure 3.25 shows the results of being sampled at 8 by 8 pixel blocks with θ at 0°, 45°, 90°, 135° and d at 1, 2, 4 and 6 pixels.

Figure 3.26 shows the results of being sampled at 16 by 16 pixel blocks with θ at 0°, 45°, 90°, 135° and d at 1, 2, 4, 6, 8 and 12 pixels.

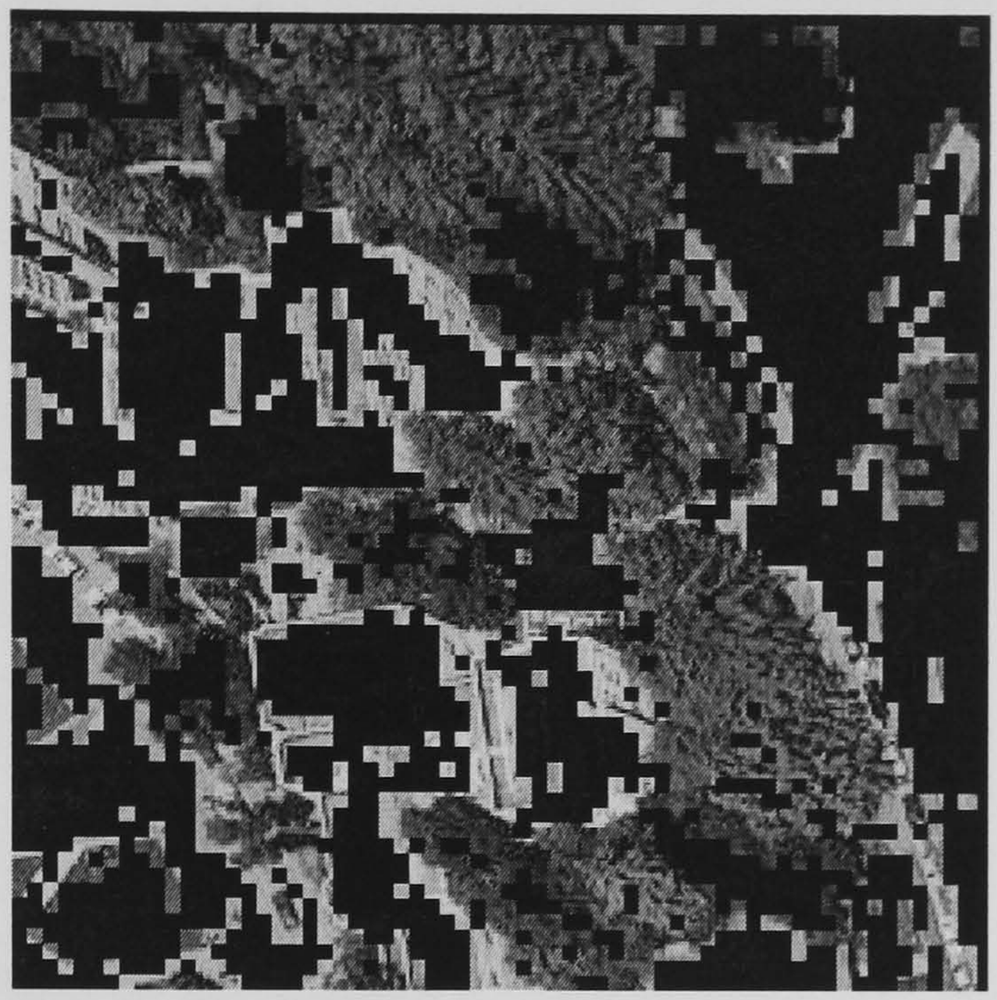


Figure 3.24 Aerial Image 1 with 4x4 pixel block segmentation

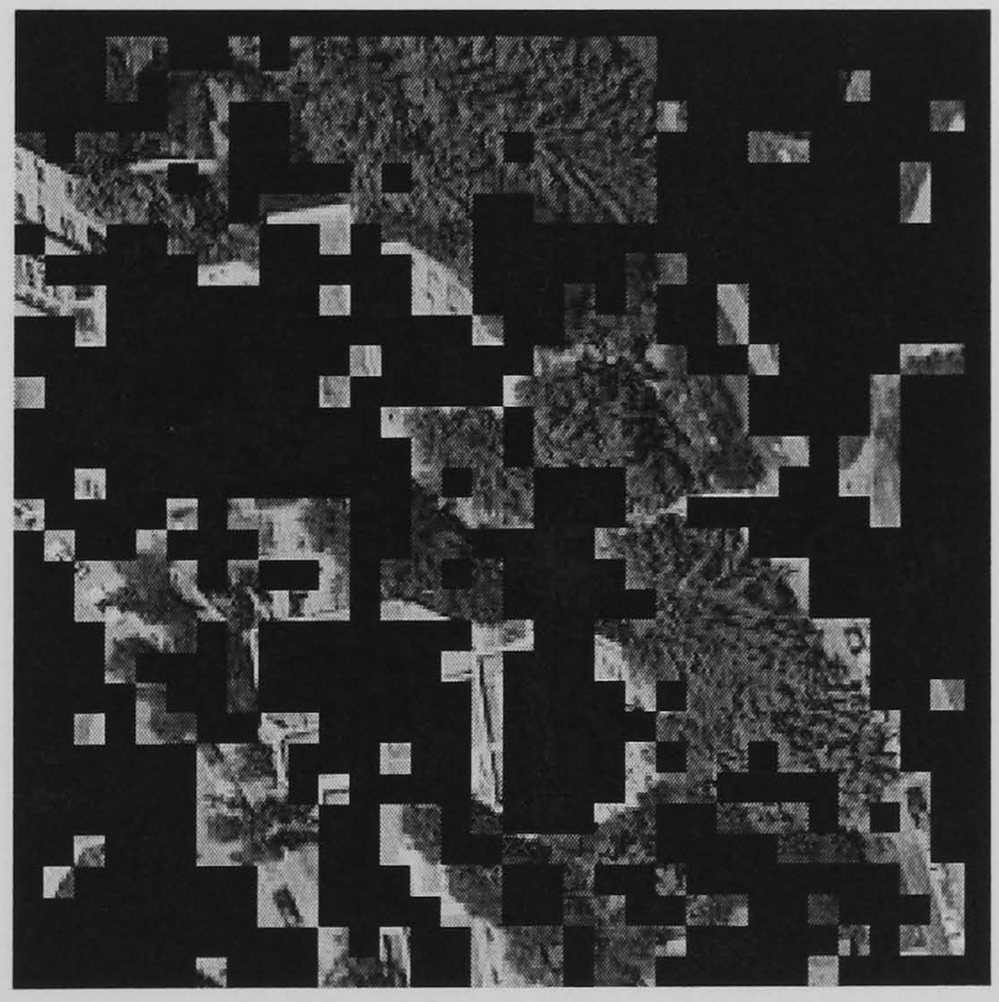


Figure 3.25 Aerial Image 1 with 8x8 pixel block segmentation

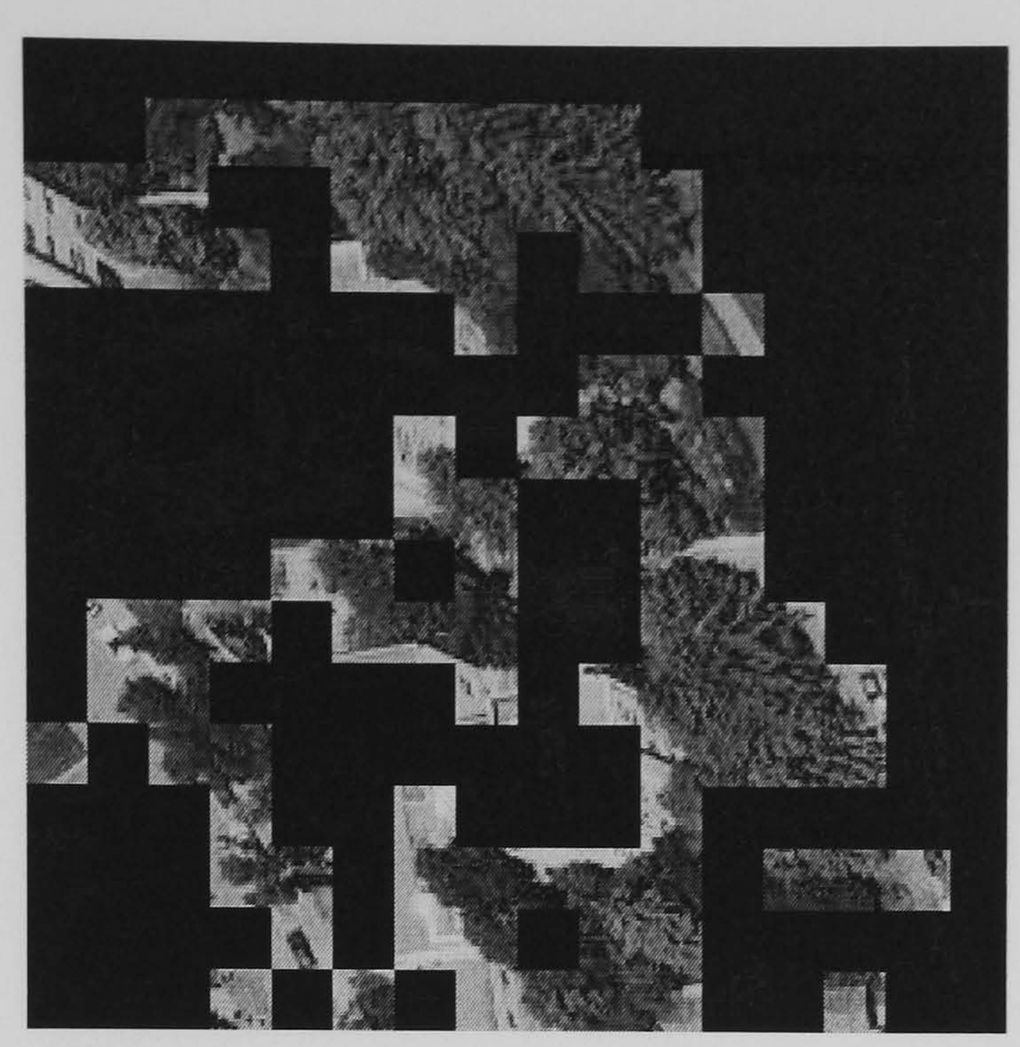


Figure 3.26 Aerial Image 1 with 16x16 pixel block segmentation

Visual inspection shows little misclassification for the building aspects in Figure 3.25 and Figure 3.26 as the size of the sample increases from which the SGLDM is created, unlike Figure 3.24.

To offer a comparison of performance with the other methodologies Figure 3.27 shows the output of a spatial grey level dependence matrix when operating on the Brodatz collage image Figure 3.7 using a 16x16 pixel square sample.

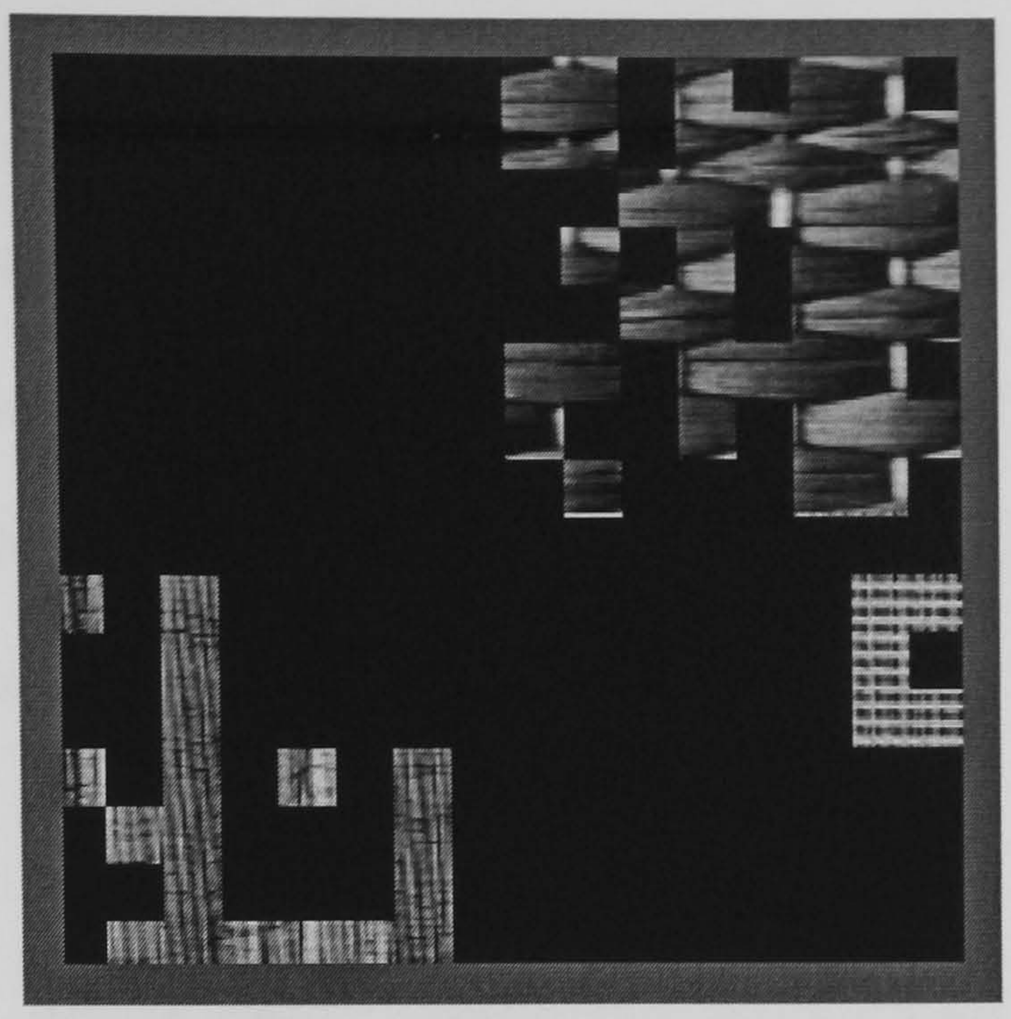


Figure 3.27 SGLDM Segmented Brodatz Collage

3.7 Summary

The four methods of texture analysis presented in sections 3.3, 3.4, 3.5 and 3.6 appear at a glance to offer the same capabilities. To offer a comparison of the performance of the different methods each was compared to an image that was hand segmented via a computer graphics package, Figure 3.28 and Figure 3.29.

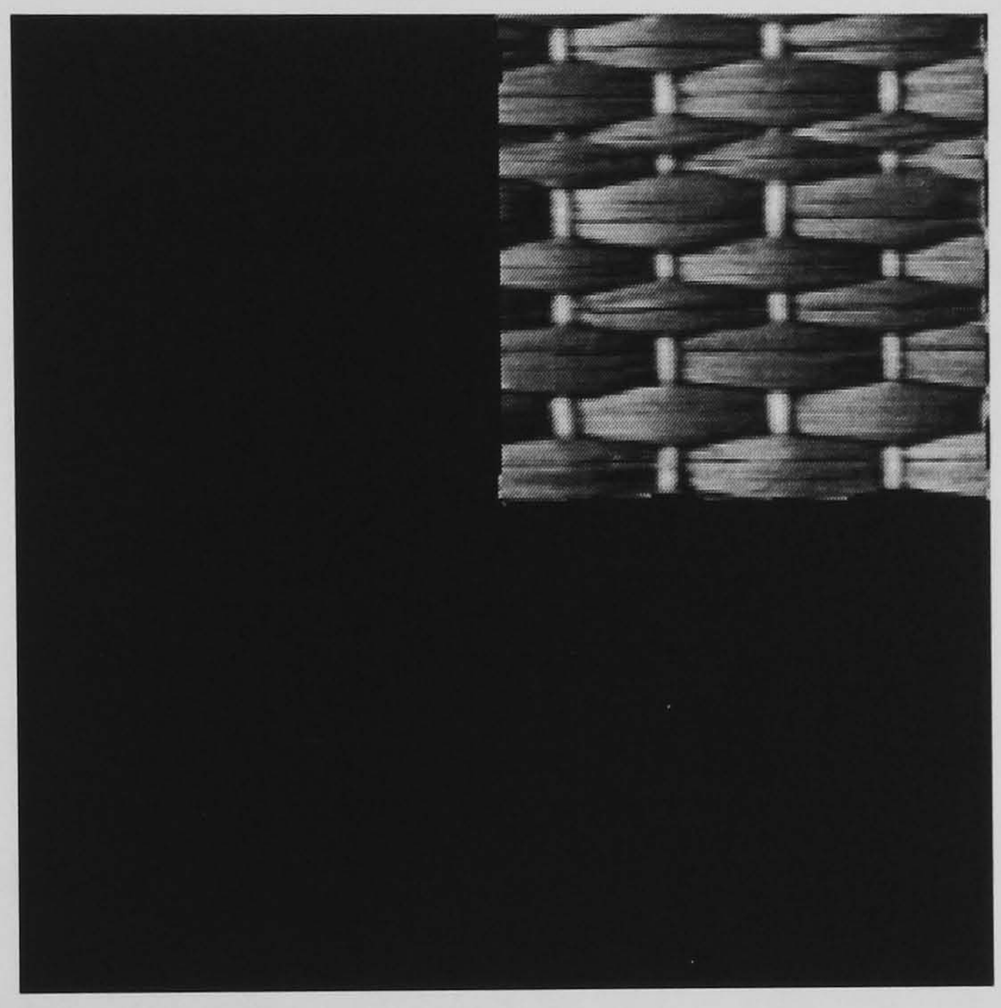


Figure 3.28 Hand Segmented Brodatz Collage Image

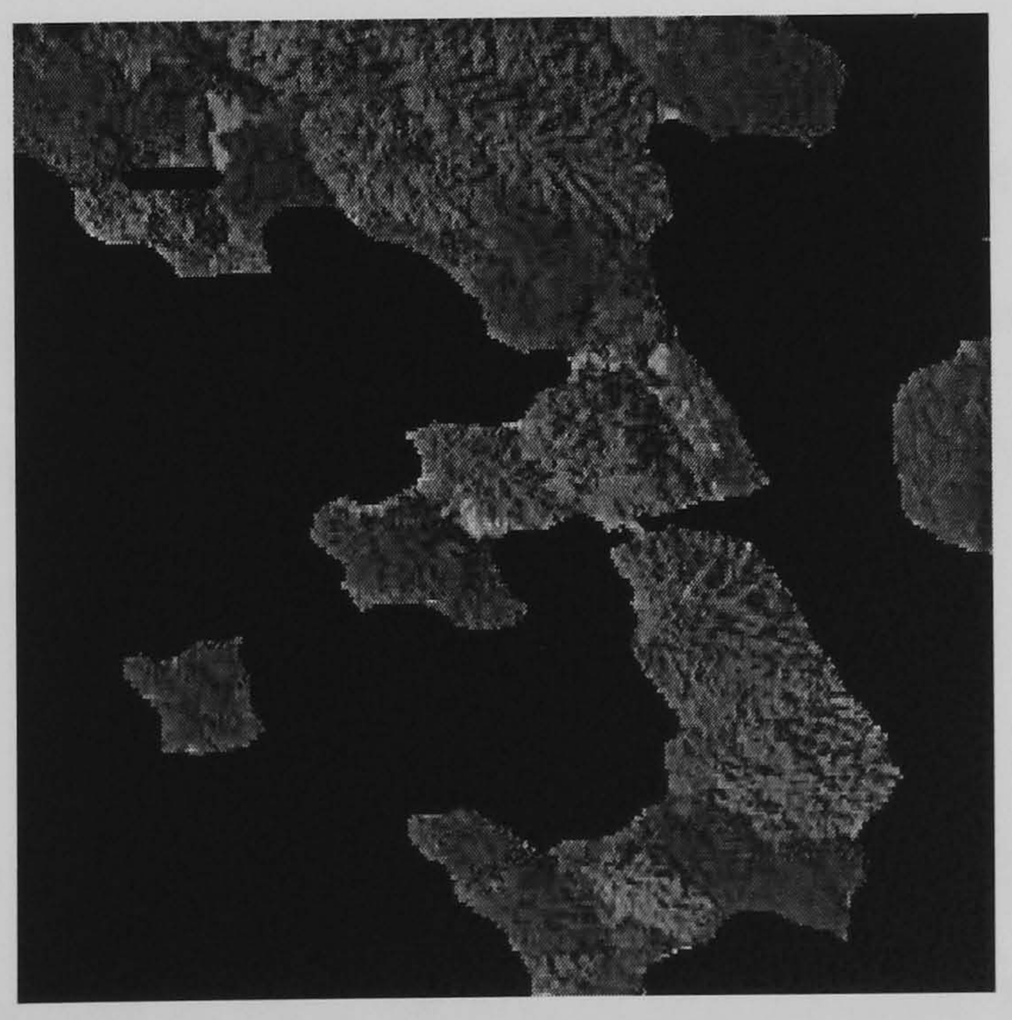


Figure 3.29 Hand Segmented Aerial Image 1

Table 3.5 below shows the percentage number of correctly classified pixels for each of the four methodologies examined. Assuming the hand segmented images are 100% correctly classified.

	Brodatz Collage	Aerial Image 1
Fourier	84.23%	70.42%
Wavelet	84.12%	58.86%
Cross-correlation	83.86%	51.50%
SGLDM	85.31%	65.23%

Table 3.5 Comparison of Texture Analysis Methods

The Fourier process gives a good overall performance for both Brodatz and real world textures. However when applied to a more complex image, for example an image with cars in it, the FFT process becomes difficult to use if the end result was to segment out all the cars and leave the remainder of the image. This is because the representation of a car in the Fourier domain can possess a large number of attributes.

With wavelets applied to the Brodatz series their ability to handle scale produces a very accurate classification when considering structured and deterministic textures. However real world textures are much more complex, a second layer of classification using wavelet features would be needed.

Cross-correlation upon deterministic and structured textures gives an accurate method of texture analysis providing an ideal method of segmentation. When cross-correlation is implemented on the aerial image shown in Figure 3.11, some limitations become

apparent as can be seen in Figure 3.23. There are no regular textures contained within it, this leads to some spurious triggers of the cross-correlation process. The setting of the threshold value of the correlation coefficient is also a limiting factor. The coefficient can change between the types of texture being sampled. The overall result indicates that this is the poorest of the four methods.

Although the initial appearance of the output from spatial grey level dependence matrices has a blocky look, the image has been segmented accurately. Conners and Harlow [Conners and Harlow 1980] demonstrate that spatial grey level dependence matrices are powerful classifiers of artificial and real world textures. With this and their popularity in mind, they were chosen to offer the basis of a benchmark against which the performance of the hybrid neural network based system is to be evaluated.

Chapter 4 Artificial Neural Networks

Sections 4.1 and 4.2 introduce the concept of biological and artificial neurons with aspects of their operation. Some basic training laws provided for the artificial neuron accompany this.

The back-propagation neural network model is discussed in detail in Section 4.3. The architecture of the network is examined and the algorithms required to train this model are documented. The benefits of multi-layer neural networks are also presented. This is followed by a practical demonstration of the back-propagation neural network being trained to act as an edge detector when working upon digital images.

Section 4.4 introduces another neural network architecture, the self-organising map. Again the algorithms required to train this model are reviewed and a practical example of the operation of this model working upon a digital image is included.

A brief summary of all the aspects of work discussed in this chapter is provided in Section 4.5.

4.1 Introduction

A neural network is a collection of highly parallel processing elements. Each element is known as a neuron and it can take many inputs to produce an output. The structure of a neuron in an artificial neural network is based on the biological equivalent found in all species of animals [Anderson 95]. The inputs to the cell body are fed from the synapses through the axon where some processing can take place, the output is then fed to other cells via the dendrites. This output process is often referred to as firing.

A typical operation of a neuron would be to fire when the summed inputs from other neurons on the synapses were greater than a threshold value held within the cell body. Haykin [Haykin 1994] gives an insight into the massive amount of neurons in the human cortex, with an estimate of $10x10^9$ neurons and $60x10^{12}$ connections via the synapses.

4.2 Artificial Neuron

The operation of an artificial neuron is analogous to that of the biological one. It takes inputs and performs a function and outputs the results. This model was first discussed by McCulloch and Pitts [McCulloch and Pitts 1943]. Figure 4.1 shows their model with two inputs.

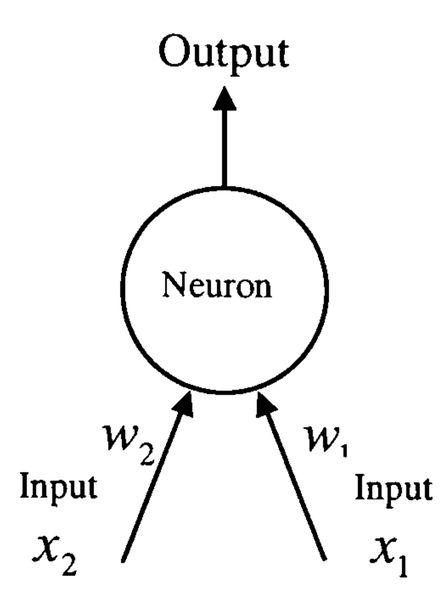


Figure 4.1 An Artificial Neuron

They defined that the neuron computes the weighted sum of its inputs (I) and then determines its output against a threshold T. Each input to the neuron (x) has a unique weight or value (w) against it. Equations (4.2.1, 4.3.2) show the neuron's state may only be +1 or -1.

$$I = \sum_{i=1}^{N} w_i x_i \tag{4.2.1}$$

$$\text{output} = \begin{cases} +1, & \text{if } I \geq T \\ -1, & \text{if } I < T \end{cases} \tag{4.2.2}$$

Where : N is the number of neuron inputs.

Rosenblatt (Rosenblatt 1958) took this concept further by introducing a training law given in equations (4.2.3, 4.2.4). This law allows the neuron to be trained to perform a function using a training set containing input patterns and required outputs. If the neuron's output is incorrect than the weight vectors applied to the inputs are modified and the neuron's output is re-assessed. This continues until the correct answer is achieved for all patterns. This process is referred to as training.

$$w_{new} = w_{old} + bnx \tag{4.2.3}$$

$$b = \begin{cases} 0, & \text{if the neuron's output is correct} \\ -1, & \text{if it is incorrect and the output should be negative} \\ +1, & \text{if it is incorrect and the output should be positive} \end{cases} \tag{4.2.4}$$

Where: n = learning rate, ranges $0 < n \leq 1$

Using these basic ideas, neurons can be connected together to produce a variety of neural network models. Lippmann [Lippman 1987] classified these models into binary input and continuous valued inputs, with sub-divisions for supervised and unsupervised learning. The very nature of image processing means that continuous valued networks offer an ideal architecture. The most popular and flexible method of continuous valued neural network using supervised learning is the error back-propagation algorithm [Rumelhart et al 1986].

4.3 Back-Propagation Neural Networks

The objective of training a back-propagation neural network is to adjust the weights of the neurons so their outputs match a desired set of results when training. This is achieved by using sets of data that have desired output data matching sets of input data. These training pairs or vectors are used to reduce the error of the network by means of a gradient descent system often referred to as the Delta Rule as shown in Figure 4.2.

This error of the back-propagation neural network can be found from the squared difference between the actual and wanted outputs of the neurons summed for all units. This error (E) can be found from (4.3.1).

$$E = \frac{1}{2}\sum_{x}(d_x - f_x)^2 \qquad (4.3.1)$$

Where:- d_x is the desired output for a neuron.

f_x is the actual output for the same neuron.

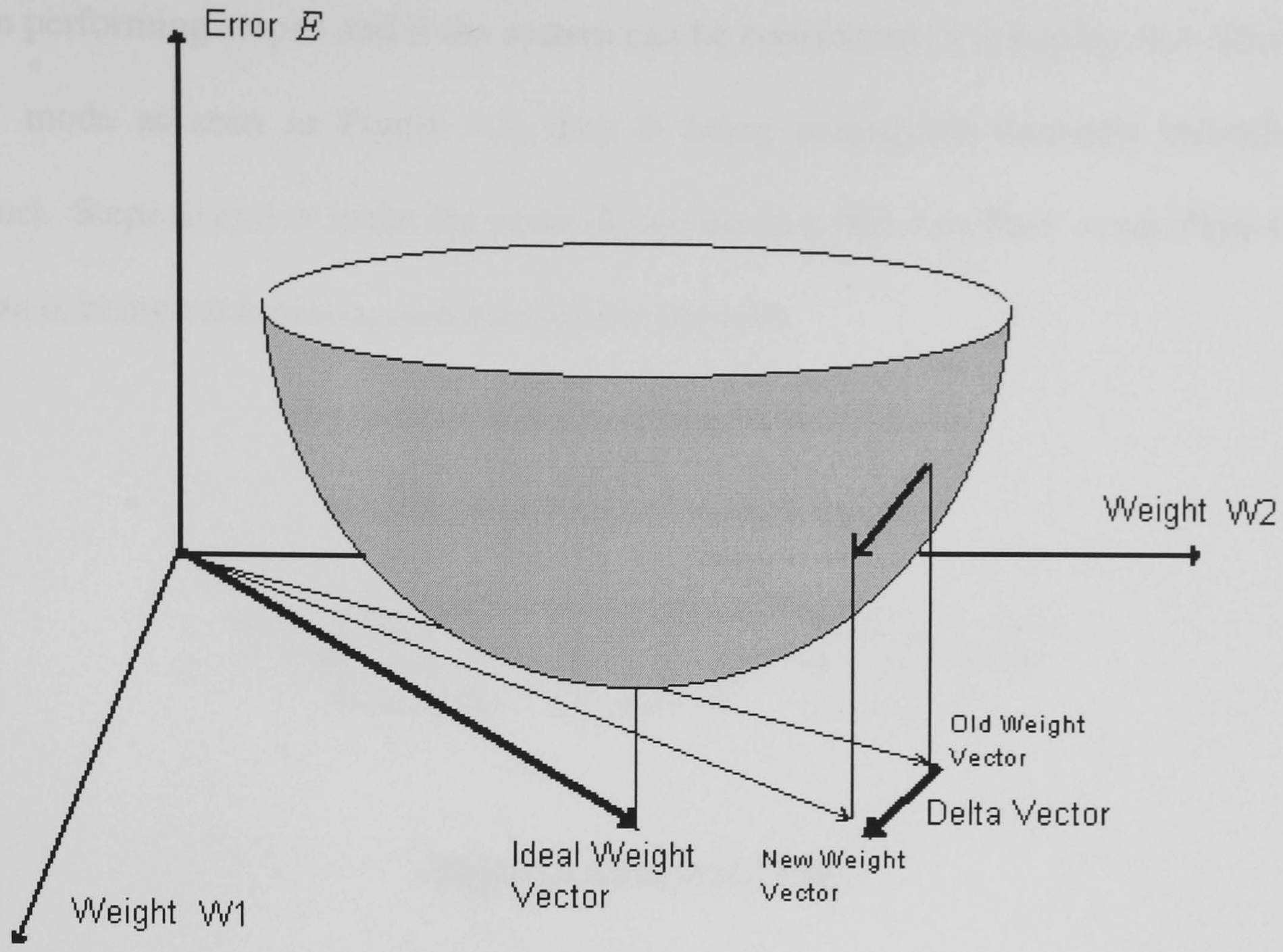


Figure 4.2 Delta Rule Gradient Descent System

This process of training can be broken down into five steps:-

i) Apply a training pair of vectors to the network.

ii) Calculate the output of the network

iii) Calculate the error of the network.

iv) Adjust the weights to reduce the error.

v) Repeat steps i through iv for all training vectors in the training set until the error is reduced to acceptable level.

When performing steps i and ii the system can be considered to operating in a 'Forward Pass' mode as seen in Figure 4.3, data is being propagated forwards through the network. Steps iii and iv make the network operate in a 'Reverse Pass' mode Figure 4.4, as data is being back-propagated through the network.

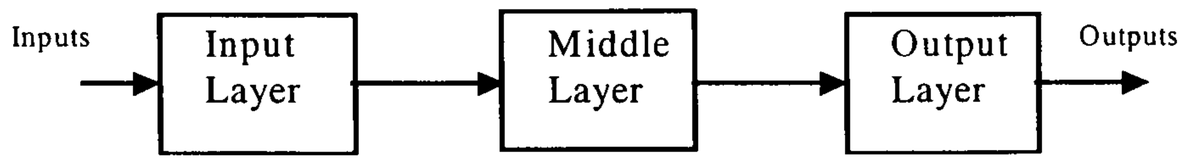


Figure 4.3 Forward Pass

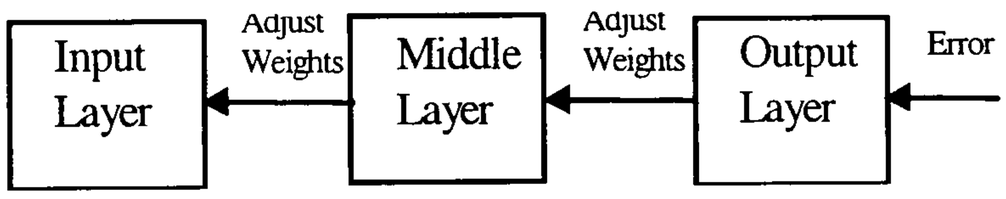


Figure 4.4 Reverse Pass

4.3.1 Error Back-Propagation Training Algorithm Forward Pass

The output of a neuron f is found by (4.3.1.1)

$$f = \frac{1}{1 + e^{-\alpha}} \tag{4.3.1.1}$$

Where: α is the sum of the products of the outputs of neurons of the preceding layer and the weights connecting them to the next layer.

The sum of products for neuron *y* in layer (*i*+1) is found using equation (4.3.1.2).

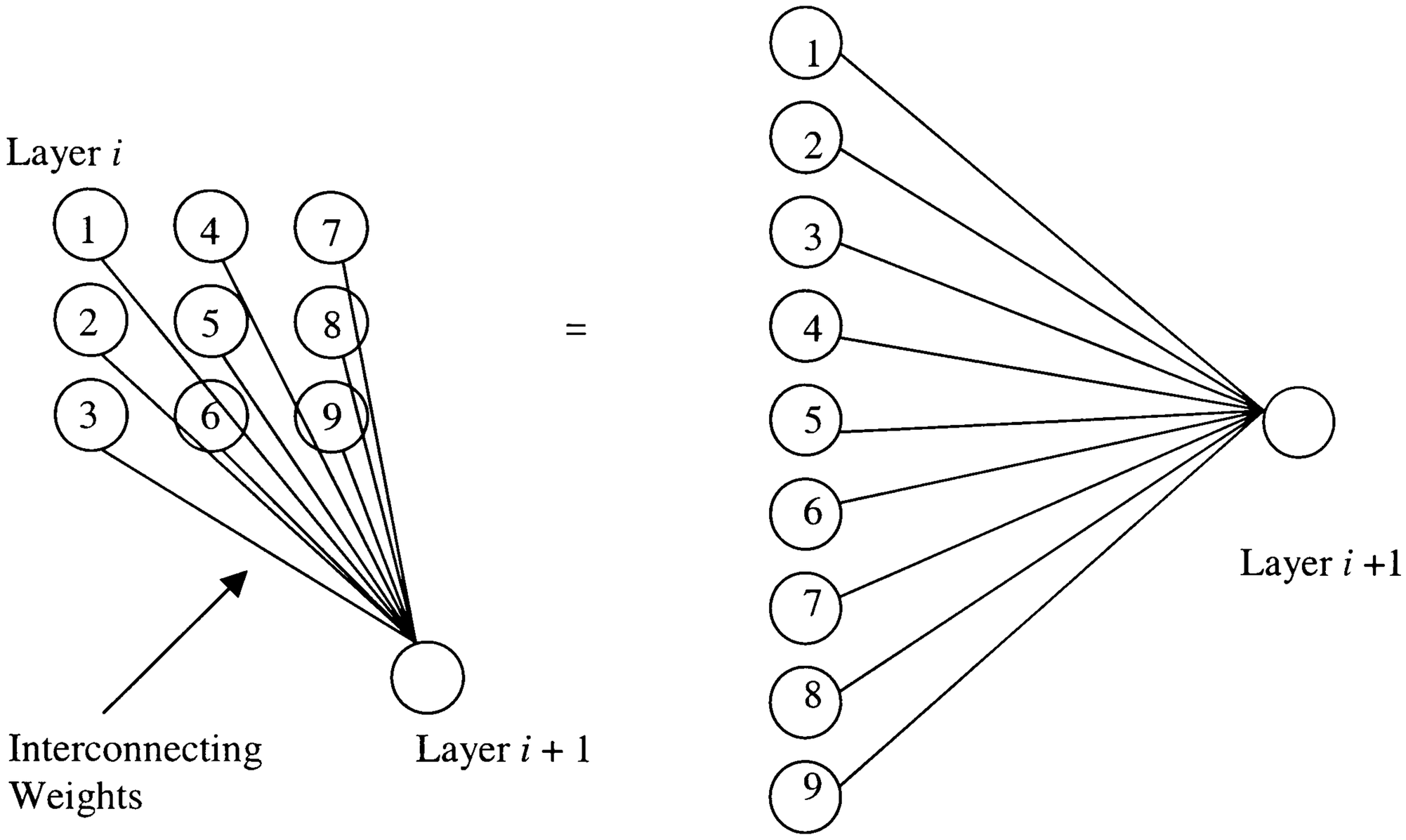
$$\alpha_y = \sum_x f_{ix} w_{ixy} \tag{4.3.1.2}$$

Where :- *i* identifies the layer (1 or 2)

x identifies neuron in layer *i*

y identifies the neuron in layer (*i*+1)

Figure 4.5 shows how a two dimensional network can be represented and indexed in a single dimension, each neuron has a unique number.



Layer *i*

Figure 4.5 Sample Neural Network Arrangement

After the outputs of the neurons have been calculated, they are used to produce α_y according to equation (4.3.1.2). This value is used to give the output of the neuron in the next layer according to equation (4.3.1.1), which limits the neuron outputs to the range 0 to 1. This function is referred to as the sigmoid function and its operation can be seen in Figure 4.6.

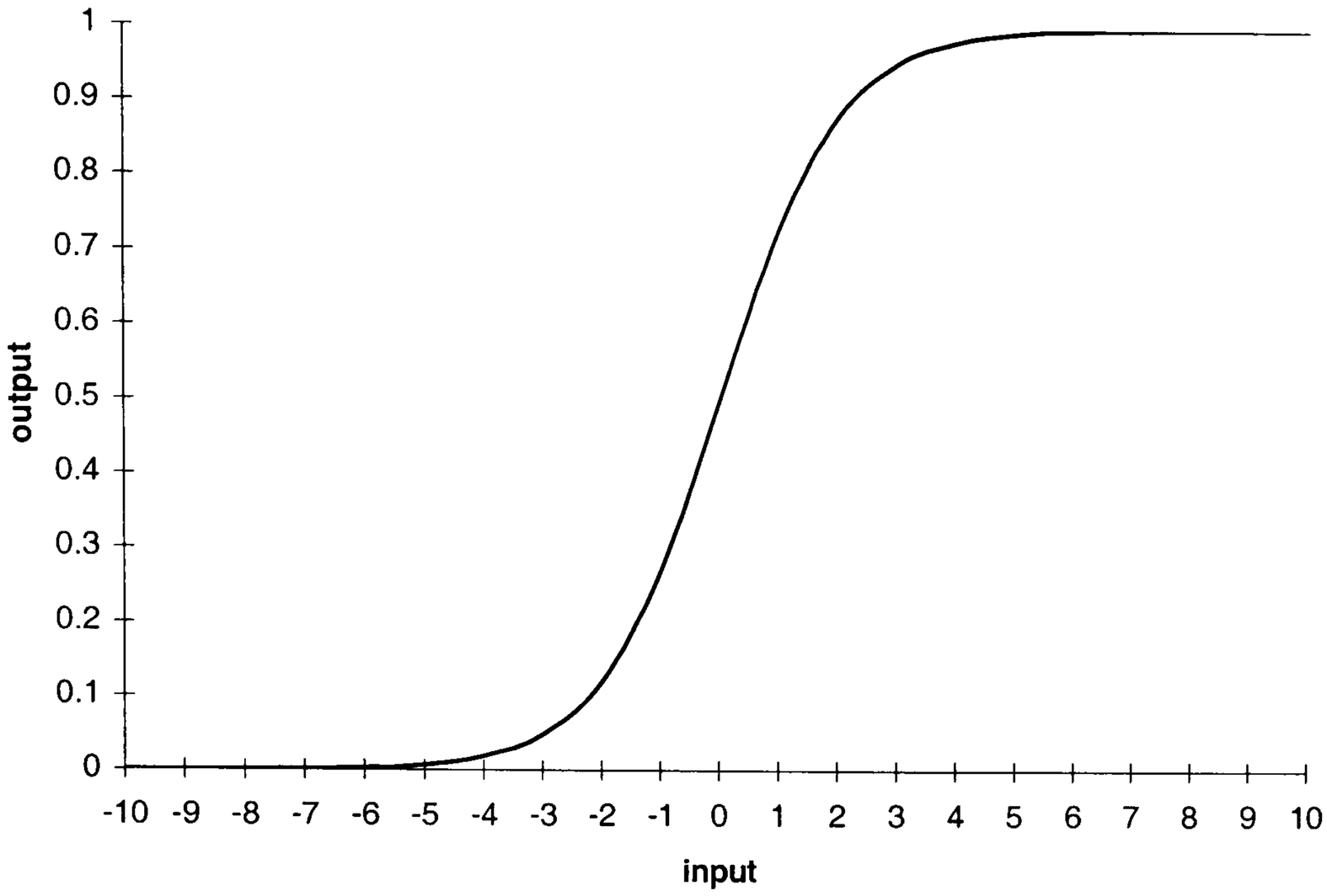


Figure 4.6 Sigmoid function

With all the outputs of the neurons calculated the reverse pass mode of operation can be implemented where the weights are adjusted to reduce the network's error.

4.3.2 Error Back-Propagation Training Algorithm Reverse Pass

The reverse pass back-propagation algorithms are used to calculate the error of the neurons in the output layer against the wanted output and adaptation of the weights feeding them from the previous layer.

The wanted output is derived from the training set. Each input applied to the network as a corresponding wanted output associated with it.

Using the neuron outputs calculated in (4.3.1.2) and the wanted output from the training set, the local gradient at the output layer δ_y can be found from (4.3.2.1).

$$\delta_y = f_y(1 - f_y)(d_y - f_y) \tag{4.3.2.1}$$

f_y is the output of a neuron in the output layer.

d_y is the wanted output for that neuron.

$f_y(1 - f_y)$ is the derivative of the neuron output to ensure a gradual error descent.

This term can be used in generating the gradient descent process for calculating changes to be applied to the interconnecting neuron weights.

Output Layer Weight Updates

The new weight $w_{2xy}(t+1)$ connecting the middle layer to output in the output layer can be found from equation (4.3.2.2)

$$w_{2xy}(t+1) = w_{2xy}(t) + \eta\delta_y f_x \tag{4.3.2.2}$$

f_x is the output of neuron in the middle layer.

δ_y is the local gradient generated from the error the neuron produced.

η is the learning rate and is a constant.

t is the current point in time in the training cycle.

x is the index of the neuron in the middle layer.

y is the index of the neuron in the output layer.

All the weights feeding the output layer can now be adapted using equation (4.3.2.2).

Hidden Layer Weight Updates

However for the hidden layer, the error term is now calculated by equation (4.3.2.3), because an error cannot be found with respect to the wanted output at this level of the neural network. The error term is made up from the sum of all errors connected to a neuron with all the weights connected to it.

$$\delta_x = f_x(1-f_x)\sum_y \delta_y w_{2xy} \tag{4.3.2.3}$$

Equation (4.3.2.4) is used to find the new weights from the input layer to the middle layer

$$w_{1xy}(t+1) = w_{1xy}(t) + n\delta_x f_x \tag{4.3.2.4}$$

Where:- f_x is the input value.

y is the index of the neuron in the middle layer.

x is the index of the neuron in the input layer.

4.3.3 Back-propagation Training Algorithm Modification

A variation on this rule is the inclusion of another term called momentum (Rumelhart et al 1986), equation (4.3.3.1). This is designed to reduce the length of time taken to train the neural network. Its purpose is the help the gradient descent training by adding momentum behind the weight vector to keep it moving towards achieving small error.

$$w_{2xy}(t+1) = w_{2xy}(t) + n\delta_y f_y + \beta\Delta w_{2xy} \tag{4.3.3.1}$$

β is the new momentum term.

Δw_{2xy} is the change applied to the neuron's weight on the last cycle.

4.3.4 Advantages of Back-Propagation

Multiple layer networks have the ability to solve linearly inseparable problems unlike single layer networks as declared by Minsky and Papert [Minsky and Papert 1969]. Single layer networks can solve problems that are linearly separable such as the AND or OR functions. Typically a hyperplane can separate decision regions as seen in Figure 4.7.

Figure 4.7 AND OR Functions

However single layer networks cannot solve linearly inseparable problems. The most common example of this can be seen in the exclusive-or problem (XOR).

If a network was to have two inputs a and b and a single output c then the following truth table (Table 4.1) would apply.

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1 XOR Truth Table

Lisboa and Haykin [Lisboa 1992, Haykin 1994] both demonstrate that creating a multiple layer neural network Figure 4.8 using a hidden layer can solve the linearly inseparable XOR problem.

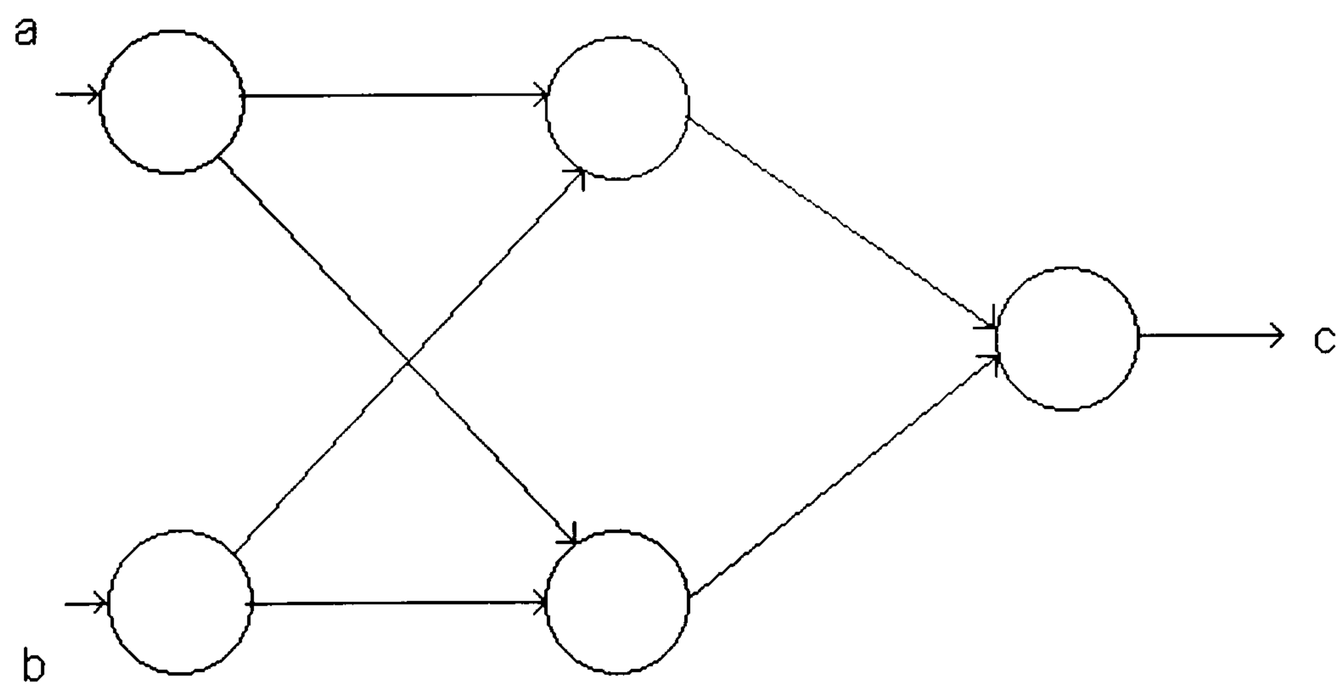


Figure 4.8 Simple XOR Back-Propagation Neural Network Architecture

With this simple network the XOR function can be implemented through a decision boundary generated by the multiple layers.

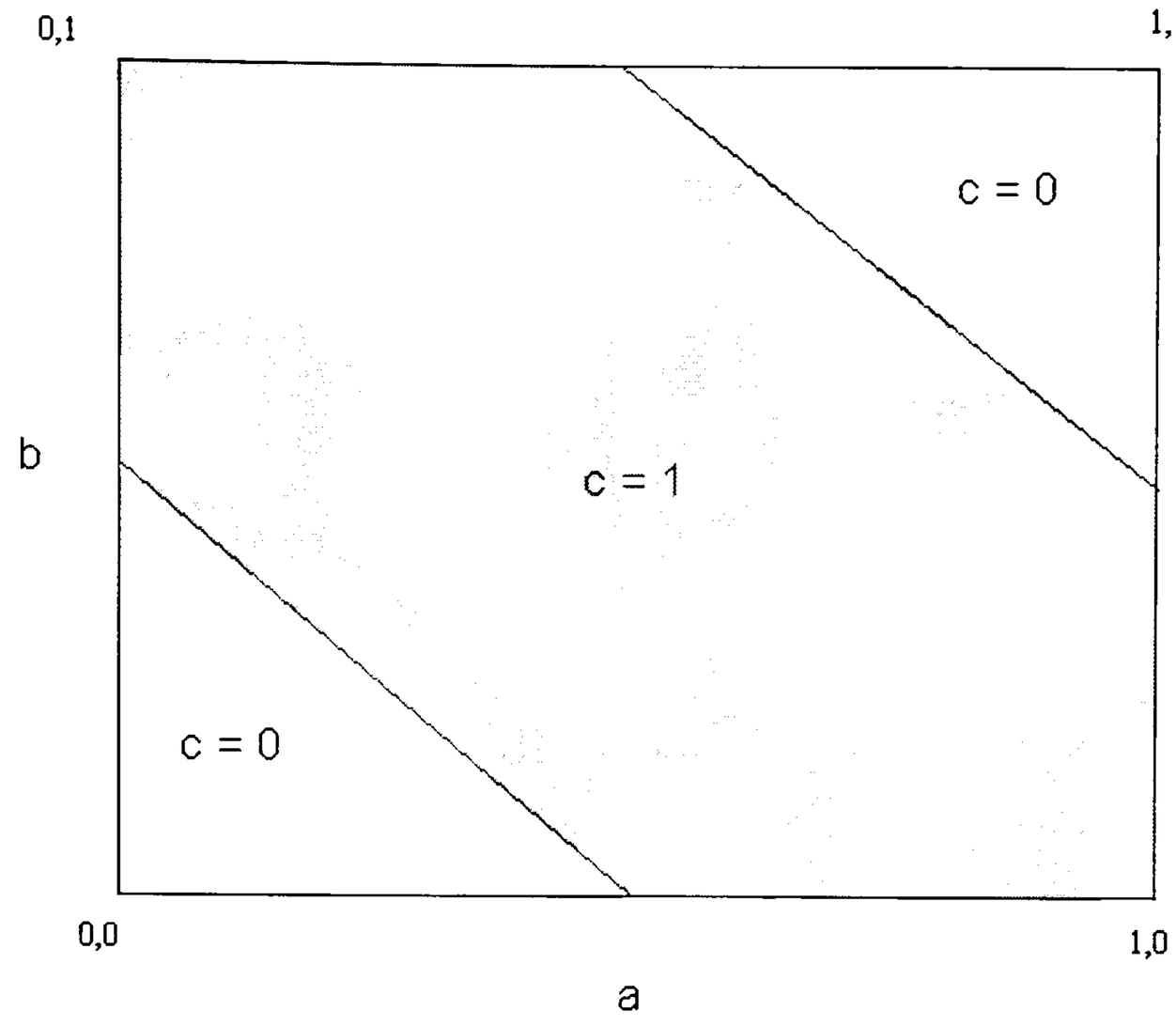


Figure 4.9 XOR Decision Boundaries

4.3.5 Use of Back-Propagation Neural Networks for Image Processing

The neural network consists of three layers:-

i) The input layer takes information from outside the network and prepares it for network operations. This may include some scaling of the data.

ii) The middle layer or hidden layer holds the bulk of the network. Training algorithms are applied to the interconnections either side of the middle layer. This layer itself may be sub-divided into more than one layer of neurons.

iii) The output layer interfaces with external elements, it also provides error information to be used in the training algorithms when training the network.

This error information provides the reference values for changing the weights between all layers of the network.

A typical arrangement of neurons can be seen in Figure 4.10.

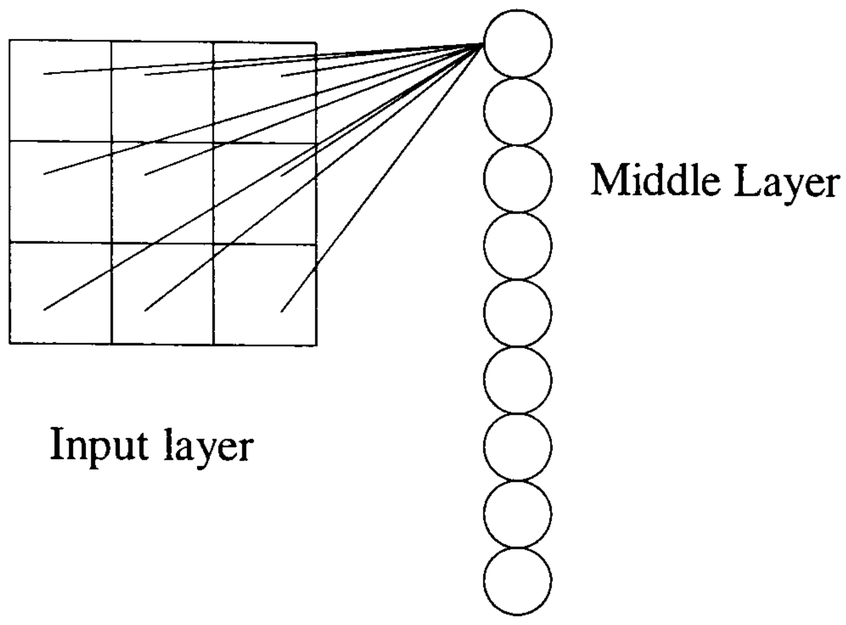


Figure 4.10 Input data arrangement for a neural network

The input values held in the mask map onto the input layer of the network. Note that in this figure only the first neuron (top) is shown as being connected to the input layer in the mask for clarity, in reality all the neurons are fully connected to all possible inputs. Since each neuron's mode of operation is between 0 and 1, the image data is scaled from 0 to 255 down to lie between 0 to 1.

Each of the associated weights to the neuron's inputs is intialised with a random number that lies between -0.5 and 0.5.

In this example the neural network requires two images, the input image which needs to altered in some way and a training image. The training image is the wanted output of the network. It is presented at the output layer and the network's output is compared against it. The network trains until its output approaches the training image.

A mask is applied to the image. The values held in the mask are fed into the neural network's input layer as depicted in Figure 4.10.

The outputs of the neurons in the middle layer are calculated and these outputs are fed into the inputs of the neurons in the output layer. The output layer is then computed and compared against the training image and the error calculated. The change in weights of the output layer and then the middle layer are calculated and then the cycle or epoch is complete. The mask is then advanced one position and the process is repeated. This carries on until the whole image is covered and one iteration is complete. A decision is then made as to embark on another iteration or to finish the training. This decision will be based upon the size of the error between the network's output against the desired training image. After training is complete the network can be run, this uses the same process as the forward pass but no error is calculated or applied to the network. The outputs of the network are copied to a temporary store that mirrors the dimensions of the input image. After the process is complete the image is updated with the contents of the temporary store.

To calculate the error between the training image and the network's output the Root Mean Square is taken after each cycle and this is used to produce an error value (4.3.5.1) at the end of a cycle when the whole image has been processed.

$$error = \left(\frac{\sqrt{(\text{Wanted output - Actual output})^2}}{\text{Total number of times the mask is used}} \right) \qquad (4.3.5.1)$$

The images shown in Figure 4.11 and Figure 4.12, were operated upon using the back-propagation training algorithm with the intention of simulating a high pass filter. The neural network architecture was implemented as follows:-

Input Layer : Nine neurons.

Middle Layer : Four neurons.

Output Layer : A single neuron.

Figure 4.11 Lena Image

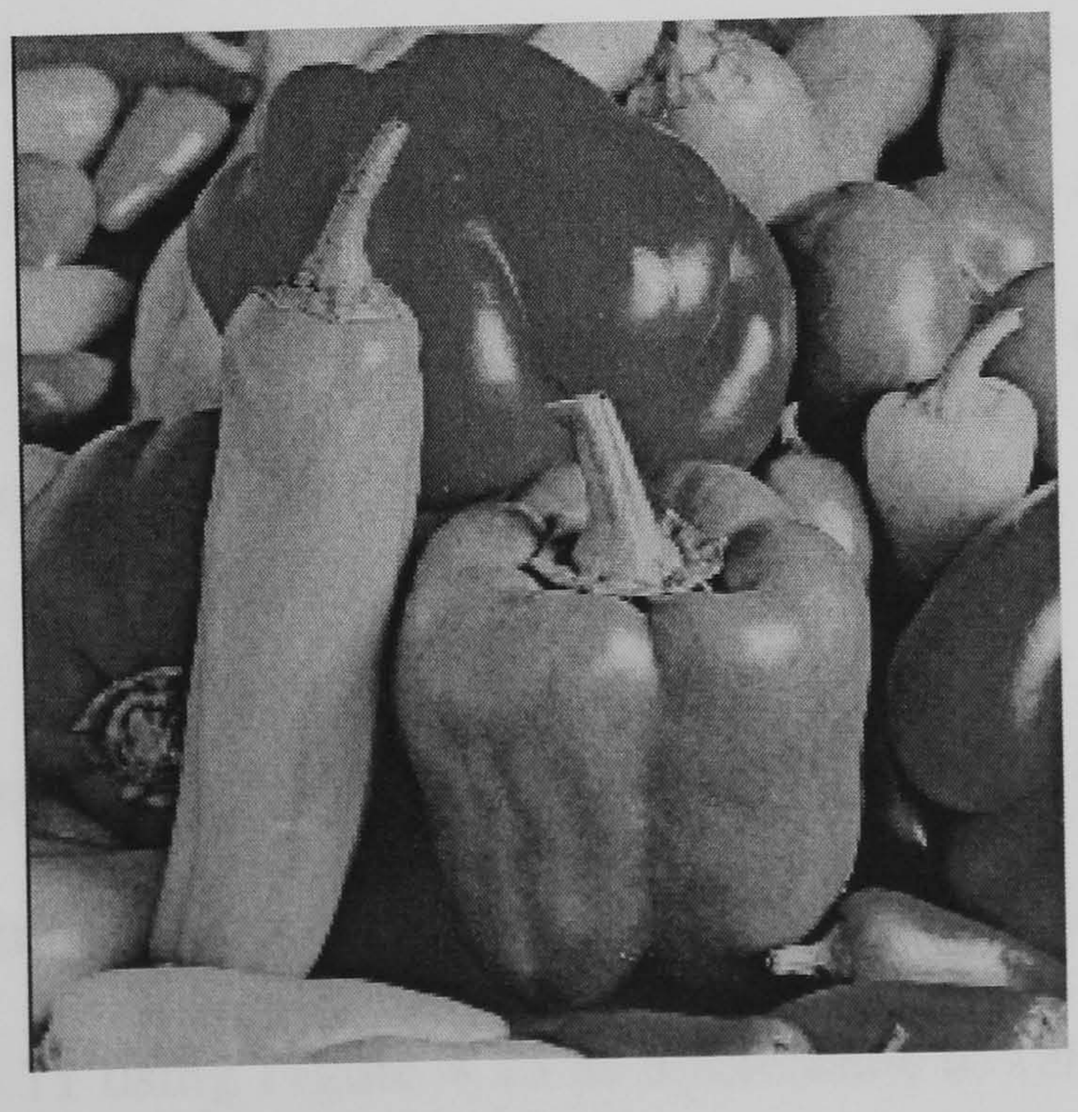


Figure 4.12 Peppers Image

The processes of training, was to drive a mask / kernel feeding the input layer across the image. The error term was calculated at the output layer by using an image that had a Laplacian 2D high pass filter applied to it to provide the wanted output through a process of convolution, Table 4.2.

-1	-1	-1
-1	8	-1
-1	-1	-1

Table 4.2 Laplacian Operation

Experimental testing found the optimum learning rate to be low (0.01) and with a momentum term of 0.5. Each image was processed for one hundred cycles. Each cycle consisting of complete coverage of the neural network input mask in the *x,y* plane.

As this Laplacian mask traverses the image, a second image is constructed from the output of the mask. The pixel values are multiplied by the constants assigned to each cell in the mask. The sum of all the cells is the new resulting pixel value for the new second image being constructed.

Figure 4.13 shows the output from the processing of Figure 4.11 with the standard Laplacian edge detecting function. Figure 4.14 is the result of the neural network training upon Figure 4.11 using Figure 4.13 as a reference to calculate the error term to back-propagate through the network.

Figure 4.13 Laplacian High Pass Filter Output for Lena Image

Figure 4.14 Neural Network Output for Lena Image

The network replicates the operation of the Laplacian filter by simulating the operation of a high pass filter upon images that have not been included as part of the training process. Figure 4.15 shows the output of the neural network when it is applied to Figure 4.12.

Figure 4.15 Neural Network Output for Peppers Image

Figure 4.16 Laplacian High Pass Filter Output for Peppers Image

4.4 Self-Organising Networks

Aleksander and Morton [Aleksander and Morton 1990] trace the origin of the self-organising map (SOM) to early work carried out by Von der Malsberg [Von Der Malsberg 1973]. The self-organising map does not have expected output results as in the case of back-propagation networks, but relies on a competitive inter-neuron relationship. When training, it behaves in an unsupervised fashion based upon competition amongst the neurons in the output layer, this is direct contrast to the back-propagation neural network, which operates in a supervised mode of training.

4.4.1 Self-Organising Map Network Architecture

Figure 4.17 shows a typical arrangement for a self-organising map with nine neurons in the output layer and twenty neurons in the input layer. For simplicity only the highlighted neuron is shown with all its connections. In reality, it and the rest of the neurons in the output layer have connections to all the neurons in the input layer.

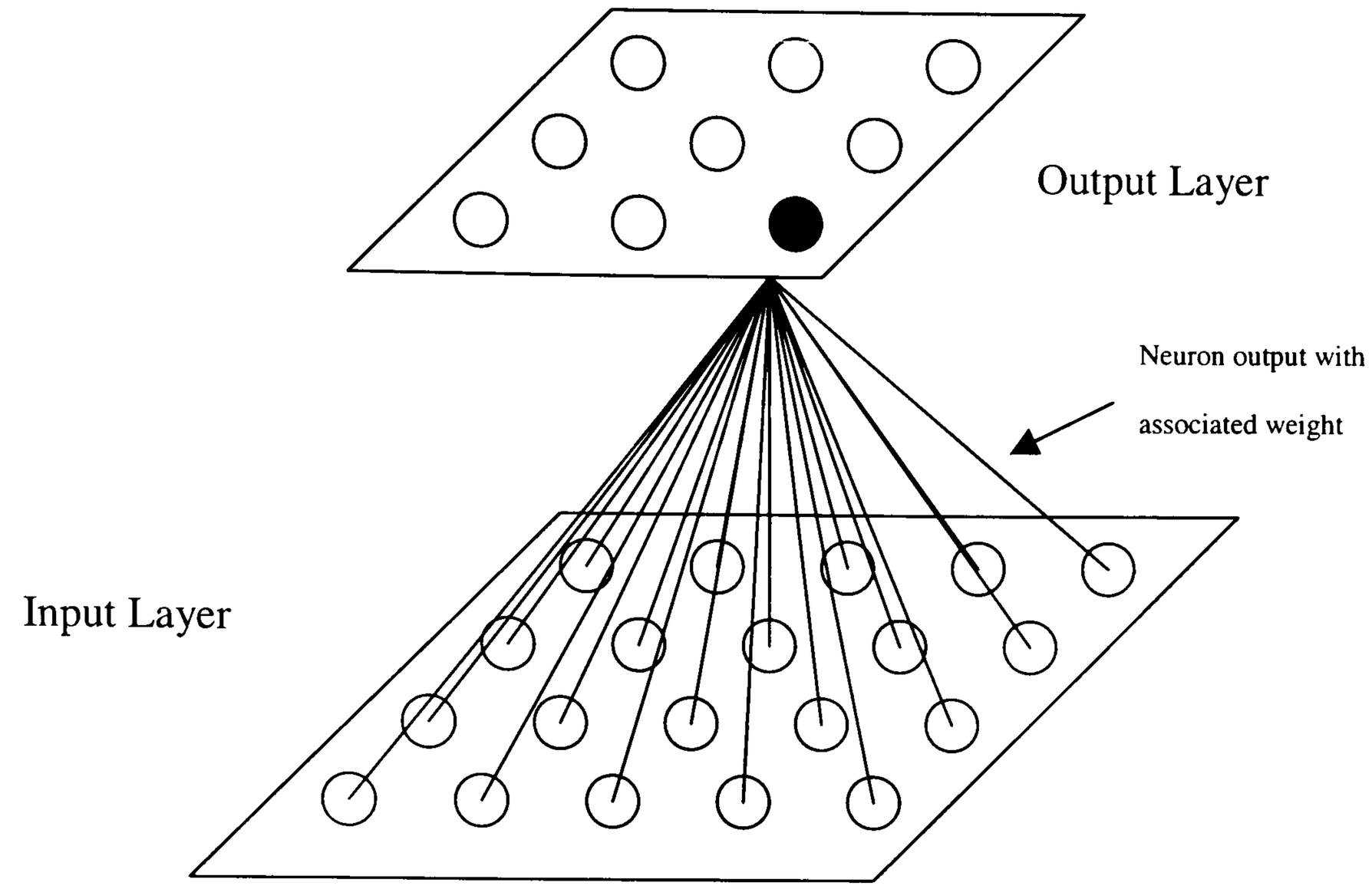


Figure 4.17 Self-Organising Map

The inter-neuron relationships are determined by competitive learning, the winning neurons in the training cycle have the largest updates to their weights. The weight updates that are applied to winning neuron are also applied to its neighbours, but these are fractional updates according to their distance from the winning neuron. This distance is known as the neighbourhood as depicted in Figure 4.18. The function to calculate the distance between neurons is known as the Box Distance or Chessboard Distance. As the network trains, the size of this neighbourhood decreases, until on the final cycle the neighbourhood has a size of one neuron, i.e. the winning neuron.

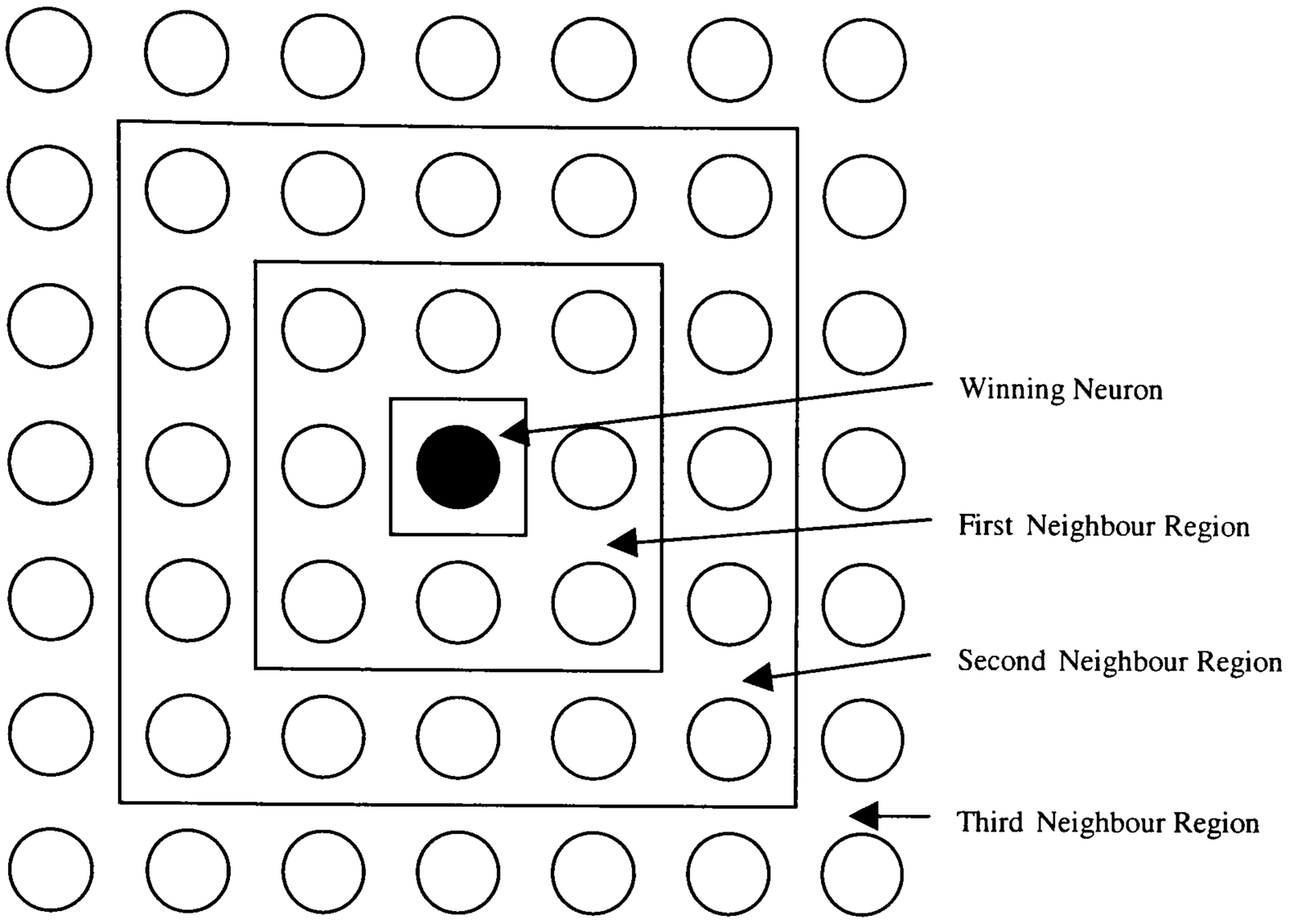


Figure 4.18 SOM Neighbourhood

4.4.2 Self-Organising Map Training Algorithm

The training of the self-organising map was popularised by Kohonen [Kohonen 1998]. It takes the form of evaluating the distance $d_y(t)$, at a point in time t, between the weights $w_{x,y}$ of a neuron and the applied input values $I_x(t)$ to the neuron.

The distance $d_y(t)$ for a neuron is calculated using equation (4.4.2.1).

$$d_y(t) = \sum_x (I_x(t) - w_{x,y}(t))^2 \qquad (4.4.2.1)$$

This distance $d_y(t)$ is evaluated for each of the neurons and the winning neuron j is then identified by finding the neuron which has the smallest distance value:-

$$j = \arg\min_{y}(d_y) \tag{4.4.2.2}$$

After the winning neuron j is found, its weights are updated according to the equation (4.4.2.3), where $\eta(t)$ is the learning rate at a specified moment in time t.

$$w_{x,y}(t+1) = w_{x,y}(t) + \eta(t)\left(I_x(t) - w_{x,y}(t)\right) \tag{4.4.2.3}$$

The learning rate gradually decreases in value over time as training of the SOM progresses. The weights of neurons situated in the neighbourhood of the winning neuron j are also updated. Fractional updates are applied to the winning neuron's neighbour's weights by equation (4.4.2.4).

$$w_{x,y}(t+1) = w_{x,y}(t) + \eta(t)\phi\left(I_x(t) - w_{x,y}(t)\right) \tag{4.4.2.4}$$

Where the scaling factor ϕ $(0 < \phi < 1)$ is determined according to the position of the neighbourhood neuron in relation to the winning neuron.

4.4.3 Use of Self-Organising Maps for Image Processing

Using a similar process to that seen in section 4.3.5 a mask is applied to an image. The mask feeds the input layer of the self-organising map. However this time there is no reference image to be considered, as the self-organising map is an unsupervised process. The inter-connecting weights are adjusted as the mask traverses the image for a pre-determined number of cycles. Again a cycle is deemed to be one complete pass of an image in the *x,y* plane.

The following architecture and settings were used to demonstrate the activity produced by a self-organising map:-

Input layer : Eighty one neurons, arranged in a nine by nine neuron grid.

Output layer : Twenty five neurons, arranged in a five by five neuron grid.

Learning rate was set to 0.3 and the initial neighbourhood size was 4.

This network was trained upon Aerial Image 1 shown in Figure 3.11. After the network had completed its training phase, it was re-applied to the image and a new image was created based upon the activity of the output layer.

Twenty-five grey scales (five by five neurons), ranging from black to white were assigned to map the activity of each neuron in the output layer. As the mask feeding the self-organising map traversed the image, a secondary image was created using the unique grey level assigned to the neurons in the output layer. That is when a winning neuron is found for a particular sample, its unique grey level is placed into the new image being constructed via a sliding window process.

Figure 4.19, shows that the training phase has allowed the neurons in the output layer to have different properties to stimulate them to becoming the most active neuron. In this case, the area within in the image that does not contain trees relies upon only a few neurons becoming the winners. Therefore identifying these neurons allows the image to be segmented.

This process takes the form of running the SOM across the source image again, this time acting as a filter. Again a secondary image is created via the output of the SOM. If the winning neuron is deemed to be representing a sample of non-trees then the grey scale placed in to the new image is black. If however the winning neuron is deemed to be not representing a non-trees sample then the original grey scale from the source image is passed into the image under construction. The designation of which neurons should represent desirable textures was made by the human operator in this case. These results are demonstrated in Figure 4.20.

Figure 4.19 Grey Level Output from SOM on Aerial Image 1

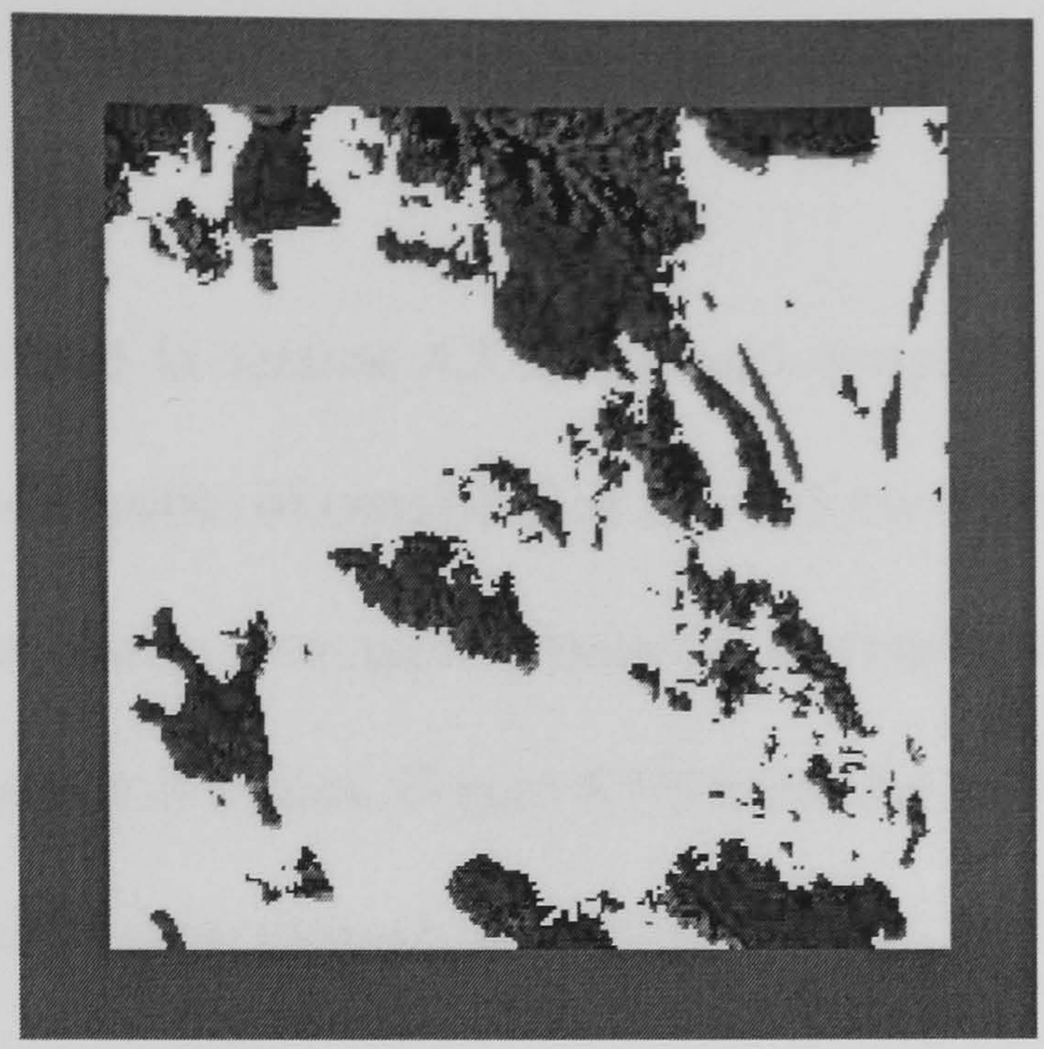


Figure 4.20 Segmented SOM Output from Aerial Image 1

White pixels are the ones that have been filtered out. The grey band around the image is the area the network cannot sample as the 9 x 9 mask will stray off the edge of the image.

4.5 Summary

With the example provided in section 4.3.5, the back-propagation neural network has been demonstrated to be capable of operating as a 'black box' system. It is able to take a desired output and train against it to approximate the operation of a Laplacian high pass filter. Comparing the output (Peppers, Figure 4.15) from the neural network against that of the equivalent Laplacian operation shown in Figure 4.16, the Neural Network output has highlighted all the edges along with removing some of the noise embedded within the image. This noise removal has occurred by the fact there was no noise in the original training image (Figure 4.11). The back-propagation neural network is ideally matched for applications that are strongly rule based.

Contrasting the back-propagation neural network is the self-organising map with its unsupervised training process. The self-organising neural network demonstrated in section 4.4.3 has strong abilities to provide unique and distinct outputs on its output layer for a wide range of possible inputs. This network works at its optimum when processing clusters of data that need to be classified.

Although the neurons themselves are simple processing elements, when constructed into a network they prove themselves to be powerful processors of data.

The following chapter presents a hybrid system which utilises both types of neural network for texture analysis in digital images.

Chapter 5 Hybrid Neural Network System for Texture Analysis

The idea behind the development of the hybrid neural network was to encapsulate the benefits of processing clusters of data from the self-organising map and the rule based capabilities of the back-propagation network.

The introduction in section 5.1 outlines the basis for the creation of the hybrid neural network system.

Section 5.2 examines the architecture of the hybrid system.

The training processes required to implement the hybrid system of neural networks are proposed in section 5.3. These processes are demonstrated against samples taken when training against the Brodatz texture series.

Section 5.4 concludes the hybrid developments with details on the operation of the hybrid architecture in an image processing environment.

A summary of the proposals for the hybrid neural network system are presented in Section 5.5.

5.1 Introduction

As seen in the previous chapter, the self-organising network exhibits the ability to be a powerful tool in the segmentation process of images by organising data into clusters. Unfortunately due to its nature of having an unsupervised training process, the neurons in the output layer will never have the same properties when the network is re-trained. This is as a consequence of applying random values to the inter-connecting weights before the training process starts. For example, a neuron that could indicate the presence of trees within an image, might only fire when roads are detected if the network was to be re-trained.

This can be seen graphically in Figure 5.1 and Figure 5.2. Figure 5.1 shows the output from the self-organising map when it was re-trained, Figure 5.2 shows a third re-training of the network. The same architecture and process was used to that which created the example in Figure 4.19. The same false grey scales were applied to the same neurons. However in this case it can be seen that the neurons now represent different aspects of texture within the image.

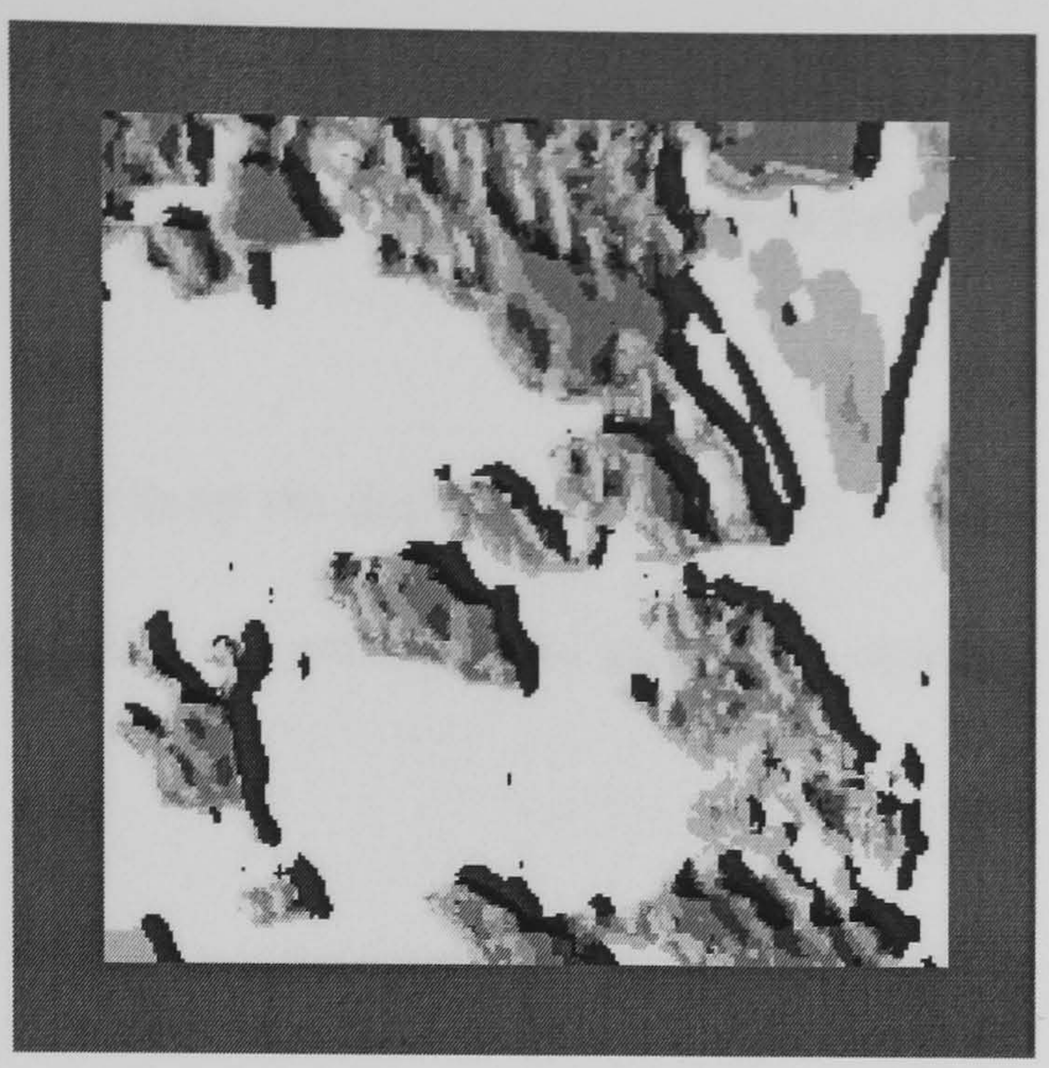


Figure 5.1 Artificial Output from SOM on Aerial Image 1, second training

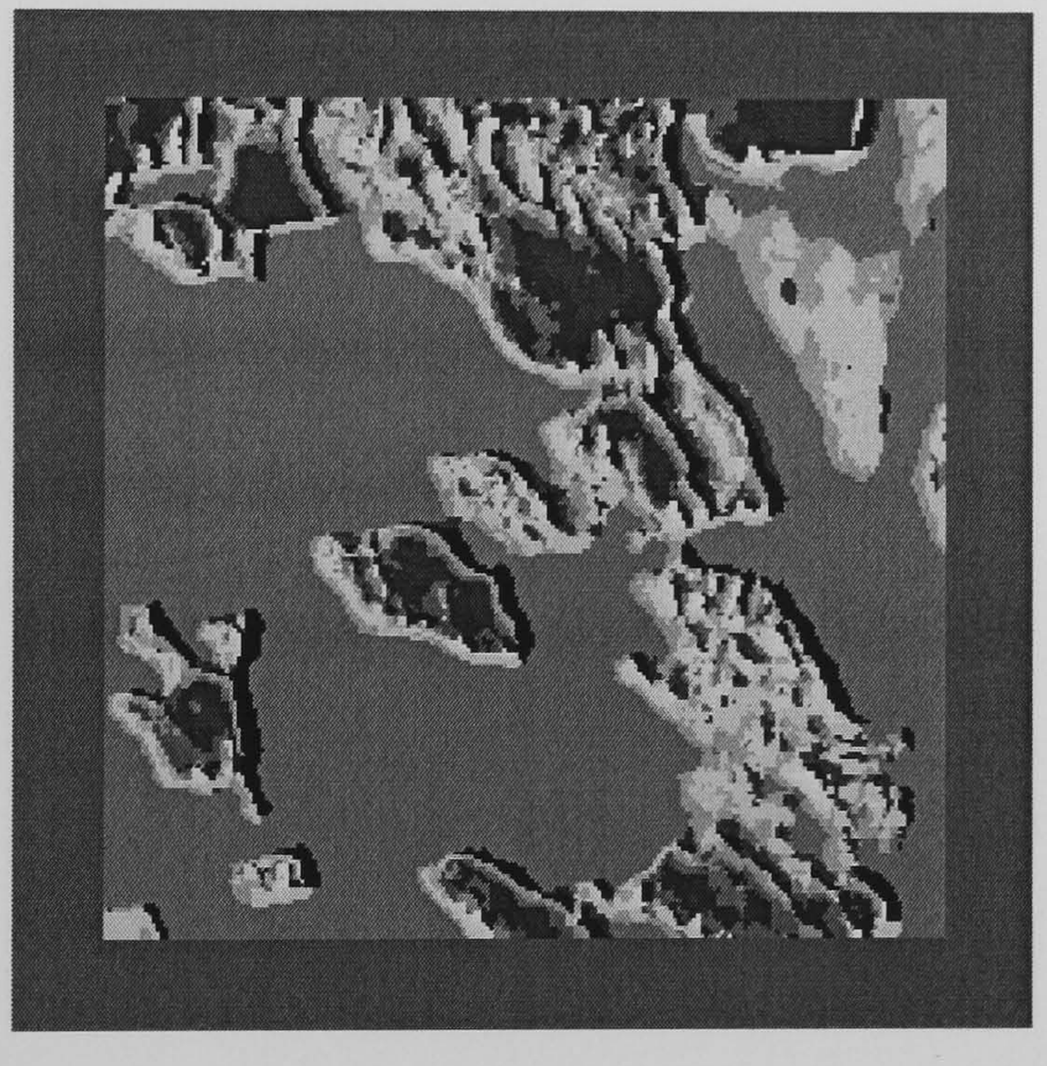


Figure 5.2 Artificial Output from SOM on Aerial Image 1, third training.

Even though the images can be seen to accurately represent the different textures within the image, there needs to be some supervised process applied to make sense of the data being created.

The process of classification of data that the self-organising map produces lends itself to that of the operation of the supervised back-propagation neural network.

Training the self-organising map on distinct textures, the back-propagation network can use these textures to calculate its error terms and train against the output layer of the self-organising map.

5.2 System Architecture

The hybrid neural network system is constructed from a self-organising neural network and a back-propagation neural network. The self-organising map traverses across images sampling pixel grey scale values from within them. This data is fed through the network to produce results upon the output layer. The data on its own is relatively meaningless unless it can be classified in some fashion. The back-propagation neural network does exactly this. Its sole purpose is to identify and classify the data being presented on the output layer of the self-organising map. The classification process may be to identify that a particular texture is present or to classify the types of texture within the image being sampled.

The output of the self-organising map is an array of neurons whose output values represent the products of the inputs attached to them. At a given point in time these values may not accurately represent the activity on the output layer of the self-organising map. In reality the recent history of the activity of the output layer needs to be recorded to avoid a spurious glitches with false triggering of neurons. To achieve the mapping of the output layer's history of activity, a cumulative two-dimensional histogram is created to record the most active neurons when working on a sample taken from an image.

The histogram transposes the self-organising map output layer activity data onto the input layer of the back-propagation network ready for classification. The self-organising map's output layer, inter-connecting histogram and the input layer of the back-propagation neural network all have the same dimensions.

The back-propagation network is constructed using three layers with the dimensions acquired through experimental data of speed versus accuracy. That is only enough neurons are allocated to complete the task in hand to keep the training times as low as possible.

The hybrid neural network depicted in Figure 5.3 is designed to classify textures by activating one of its neurons in the output layer when a particular texture is encountered by the self-organising section. In this case ten different types of texture can be catalogued (ten neurons in the output layer).

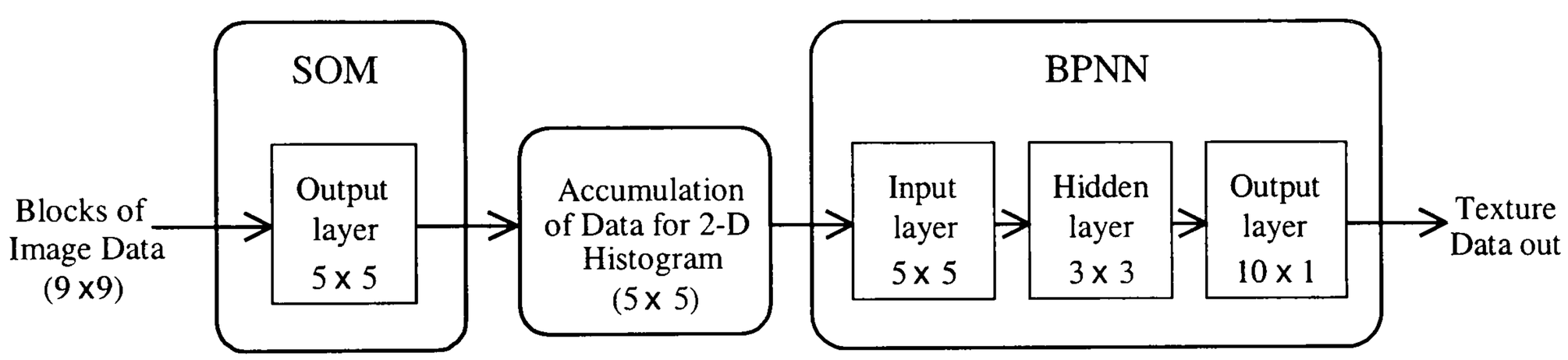


Figure 5.3 Hybrid Neural Network System for Texture Analysis

The dimensions of this architecture are derived through experimental analysis to determine the most computationally efficient with regards to accuracy. This is demonstrated in Section 6.2.2.

5.3 System Training

The training of the hybrid neural network takes the form of three distinct and separate stages :-

5.3.1 Self-Organising Map Training

To train the self-organising map, test images must be first created for the input data. An image is created for every individual texture that is to be classified. Data is extracted from each image in turn via a mask that is run across the image on a pixel by pixel basis. All images considered in this investigation are monochrome bitmaps, having a grey scale range of 256 (0 to 255) levels. As the maximum operating range of input for a neuron does not exceed one, the pixel grey scale information is scaled down accordingly. The mask extracting data from the image has a one to one mapping on to the input layer of the self-organising map. Each time the mask is moved across the image the data is presented to the SOM and a winning neuron is found. Its weights are then adjusted accordingly along with its neighbours' weights. The mask is applied to each image in turn until all the images in the training set have been accessed, then the mask position is incremented by one pixel in the x plane, and then again all the images are processed. This is repeated until the mask reaches the end of the x axis, now the mask is shifted by one pixel in the y direction and the x direction count is reset. After the mask has scanned the entire image a training cycle has been completed. At this point the neighbourhood and learning rate can be reduced with another training cycle beginning or the training phase can be deemed to have been completed.

An important part of the training cycle is the learning rate; if the learning rate is too low the network does not converge properly, if it is too high the network runs the risk of falling into local minima [Anderson 1995].

Figure 5.5 shows different learning rates self-organising map training on a Brodatz image D106 (Figure 5.4), a 9 by 9 pixel input layer was run across a sample region of 32 by 32 pixels for 10 cycles.

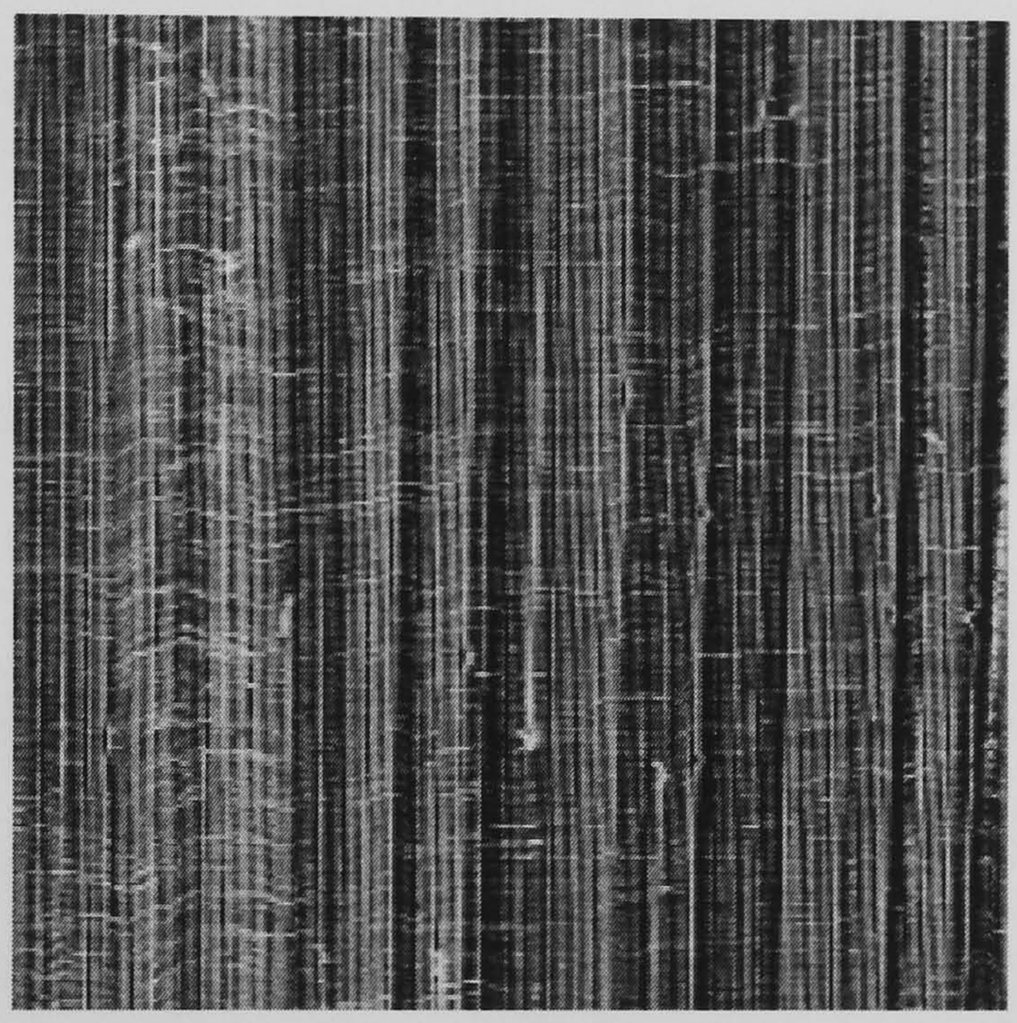


Figure 5.4 D106 (Cheesecloth)

The criteria for choosing a learning rate was having an optimum curve with a steady descent of the slope, levelling out at a low error value. The lowest final error value in conjunction with a steady slope was deemed to be the best relationship to overcome the risk of falling into local minima.

The learning rate shown is the initial learning rate at the start of the training process. Through experiments such as this on different images, it was found that the optimum curves were often obtained when the learning rate was in the region of 0.01.

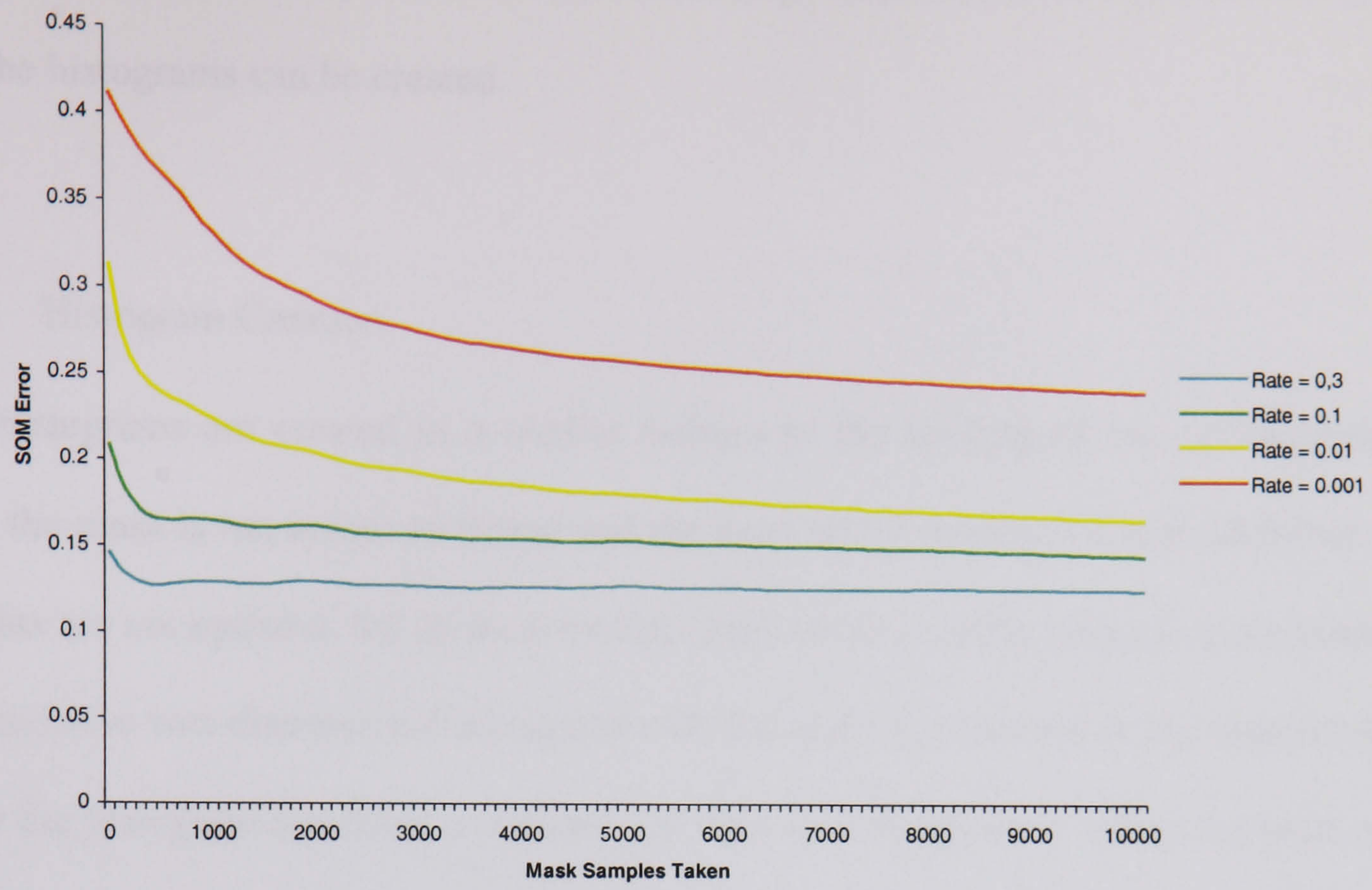


Figure 5.5 Comparison of SOM Learning Rates

The SOM error term (5.3.1) reflects the average error across all images in the training set.

$$error = \frac{1}{L}\sum_{k=1}^{L}\sum_{y=1}^{M}\sqrt{d_{y,k}} \qquad (5.3.1)$$

Where :- y is the index of the neuron in the output layer.

M is the number of neurons in the output layer.

k is the index of the image segment presented to the input layer.

L is the total number of image segments presented to the network.

(This is found from the total number of images multiplied by number of sample windows multiplied by the total number of input windows within the sample)

After training has taken place, the inter-connecting weights of all the neurons are saved and the histograms can be created.

5.3.2 Histogram Creation

The histograms are created in a similar fashion to the training of the self-organising map, the mask is run across an image and the most active neuron is found. However the weights are not updated, but an incremental count of the winning neurons is recorded in a cumulative two-dimensional histogram with the same dimensions as the output layer. After the histogram has been completed the data is normalised to values between zero and one. A typical example can be seen in Figure 5.6, there are different regions of activity (hotspots) on the output layer with some very dominant neurons.

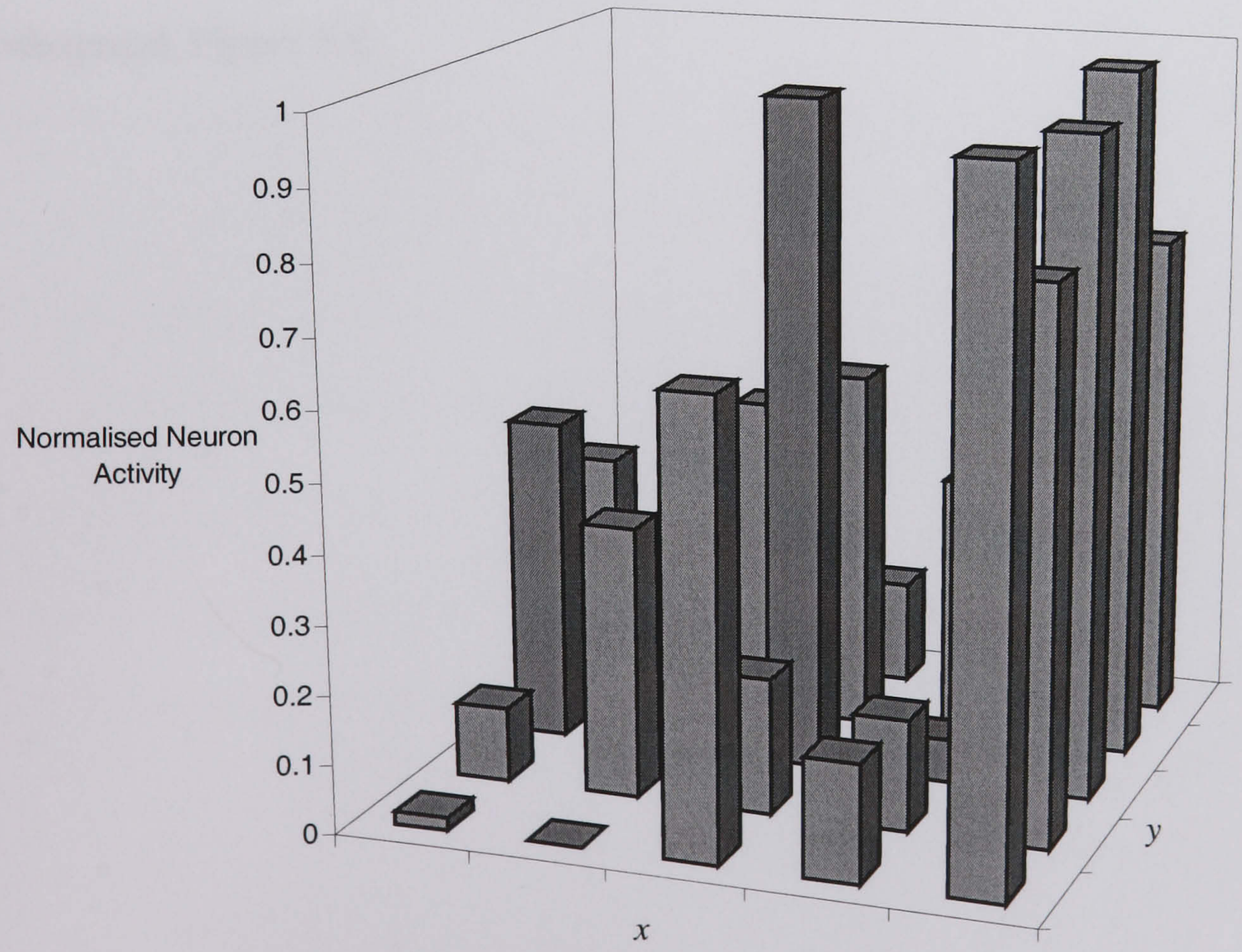


Figure 5.6 Histogram 1 for Brodatz Image D106 (Fig 4.4)

The activity in the output layer for a different image is considered by applying another Brodatz texture, Figure 5.7 D9 (Grass)

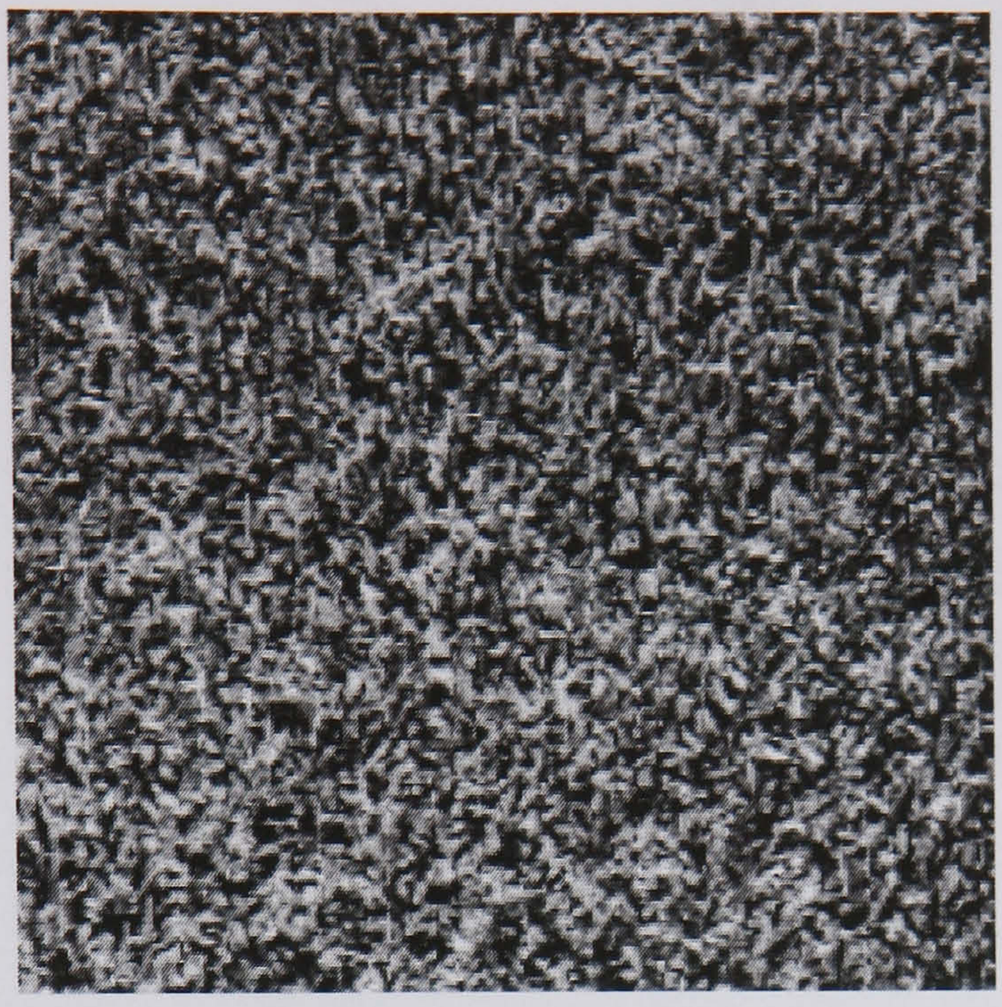


Figure 5.7 D9 (Grass)

The activity in the output layer when sampling Figure 5.7 can be seen to be more widespread, Figure 5.8.

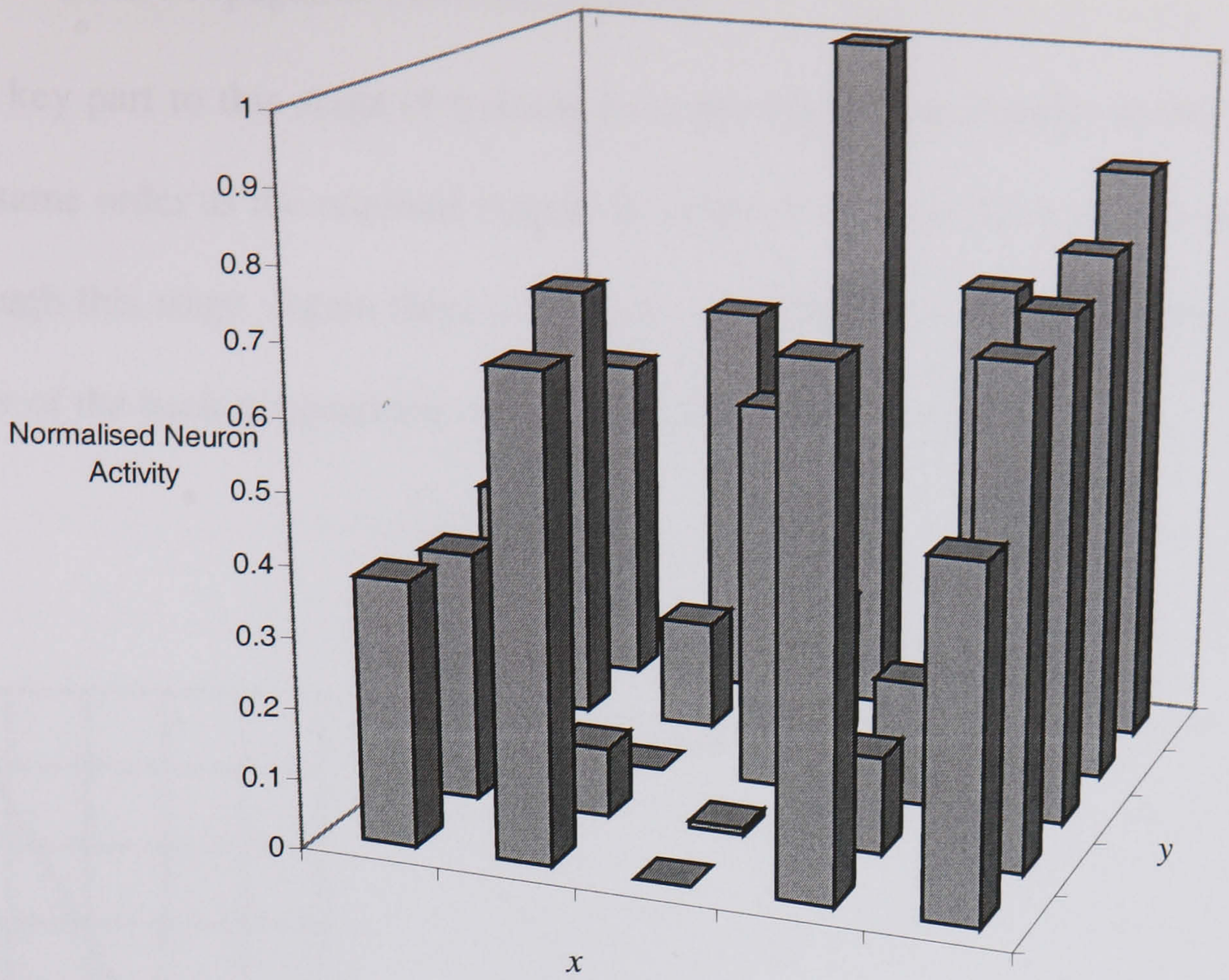


Figure 5.8 Histogram for Brodatz Image D9

As the inter-connecting histograms represent the activity on the output layer of the self-organising map, the sample size when creating these should be large enough to accurately represent the most active neurons. That is, if the sample region taken from the image to run the sampling mask over is too small, the histogram may not reflect the true activity of the output layer that would be otherwise seen in a larger image sample. The normal practice is to use the same image and size of sample that was used to train the network. e.g. if the sample image is 32 by 32 pixels then the sample region would be 32 by 32 pixels. However this need not be the case and is discussed in section 6.2.

5.3.3 Back-Propagation Network Training

The key part to this stage of training is to present the previously created histograms in the same order as the required outputs to create the correct error term to propagate back through this stage. Again there is a one to one mapping of the histogram onto the input layer of the back-propagation network as shown in Figure 5.9.

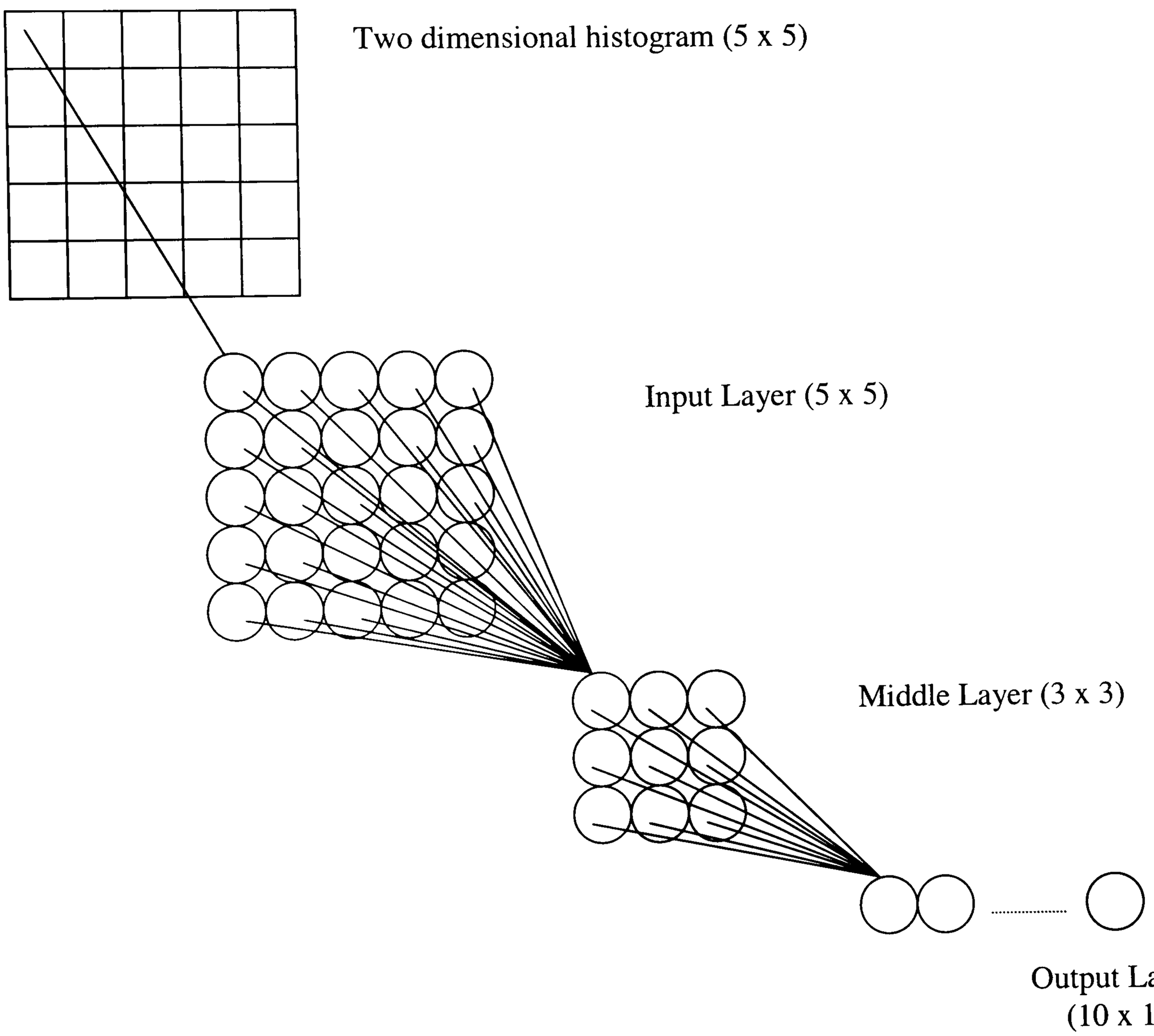


Figure 5.9 BPNN stage

As each histogram is presented to the input layer the appropriate look up is applied at the output layer to find the wanted output to calculate the error term.

As with self-organising map the learning rate influences the accuracy of the network, but this time there is also the momentum term to be taken into consideration. The following graphs explore the relationship between the learning rate and the momentum term to find the optimum training setting when training against the histogram created for Brodatz image D106.

Figure 5.10 shows a learning rate of 0.01 to be inadequate with the network failing to train correctly.

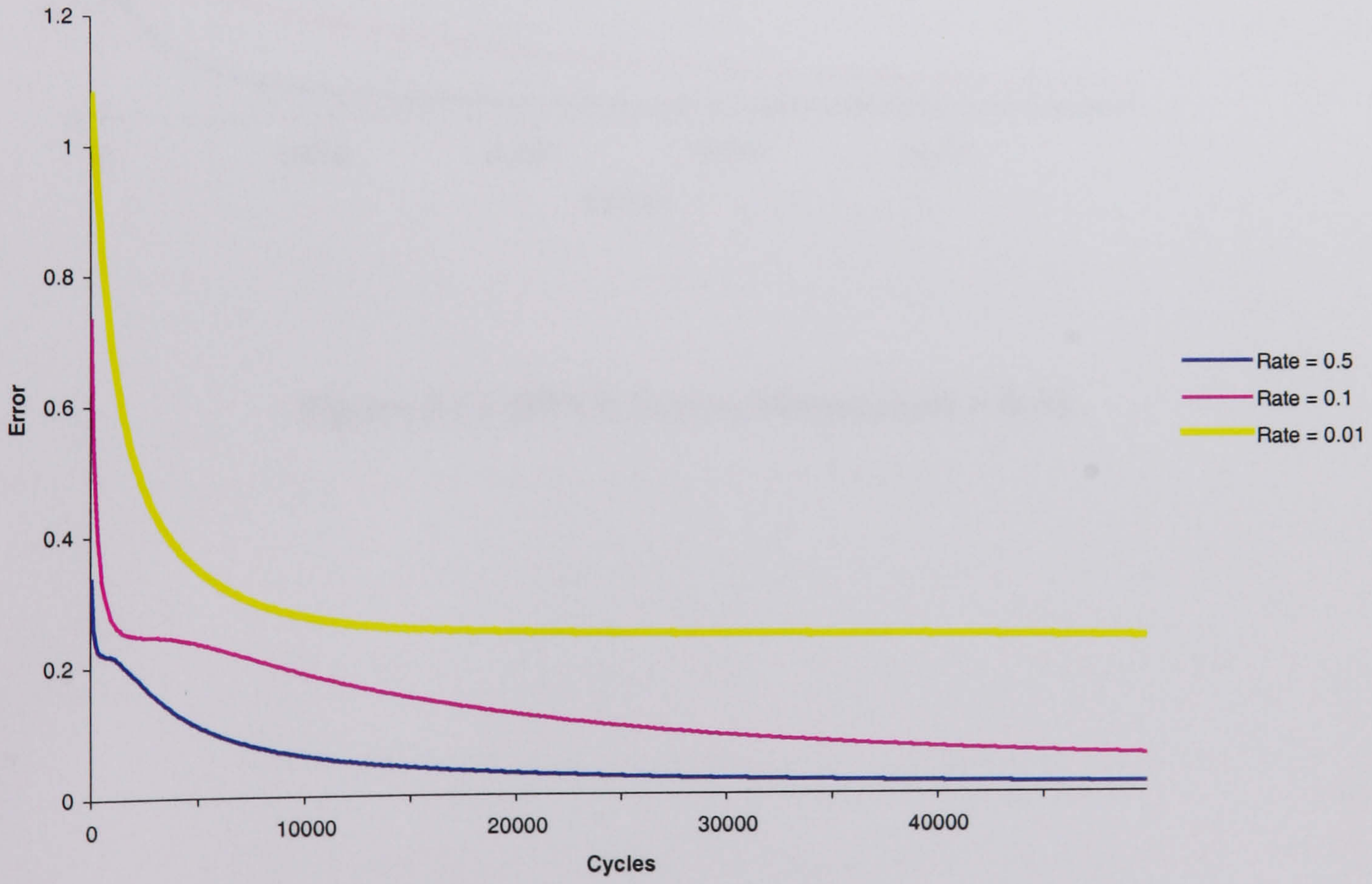


Figure 5.10 BPNN Error, Momentum = 0.125

As the momentum term increases to 0.25, a learning rate of 0.5 offers a steady curve that reduces to a small error quickly, Figure 5.11.

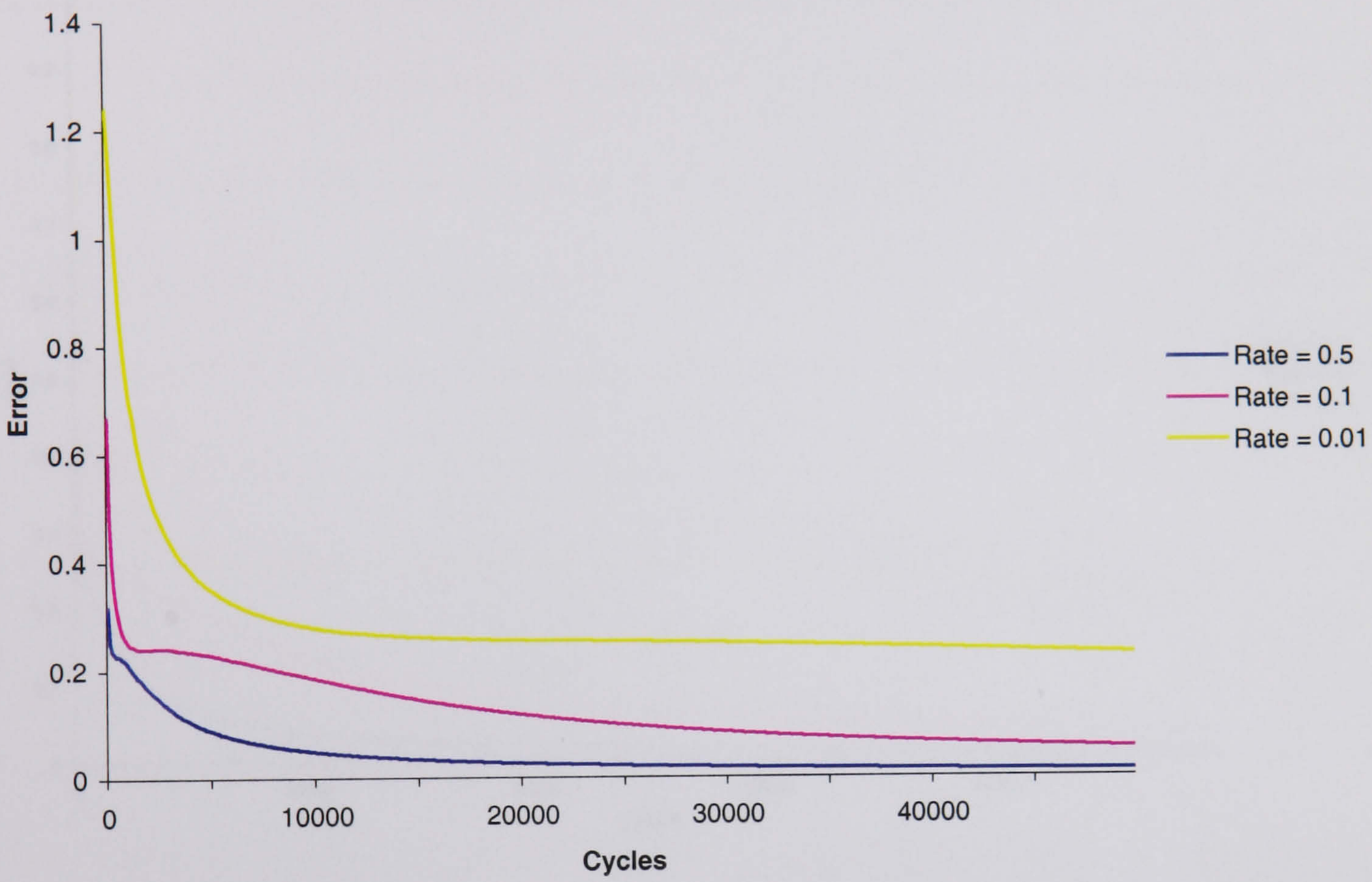


Figure 5.11 BPNN Error, Momentum = 0.25

Figure 5.12 shows a steady descent to a low error with a combination of momentum set at 0.5 and a learning rate of around 0.1.

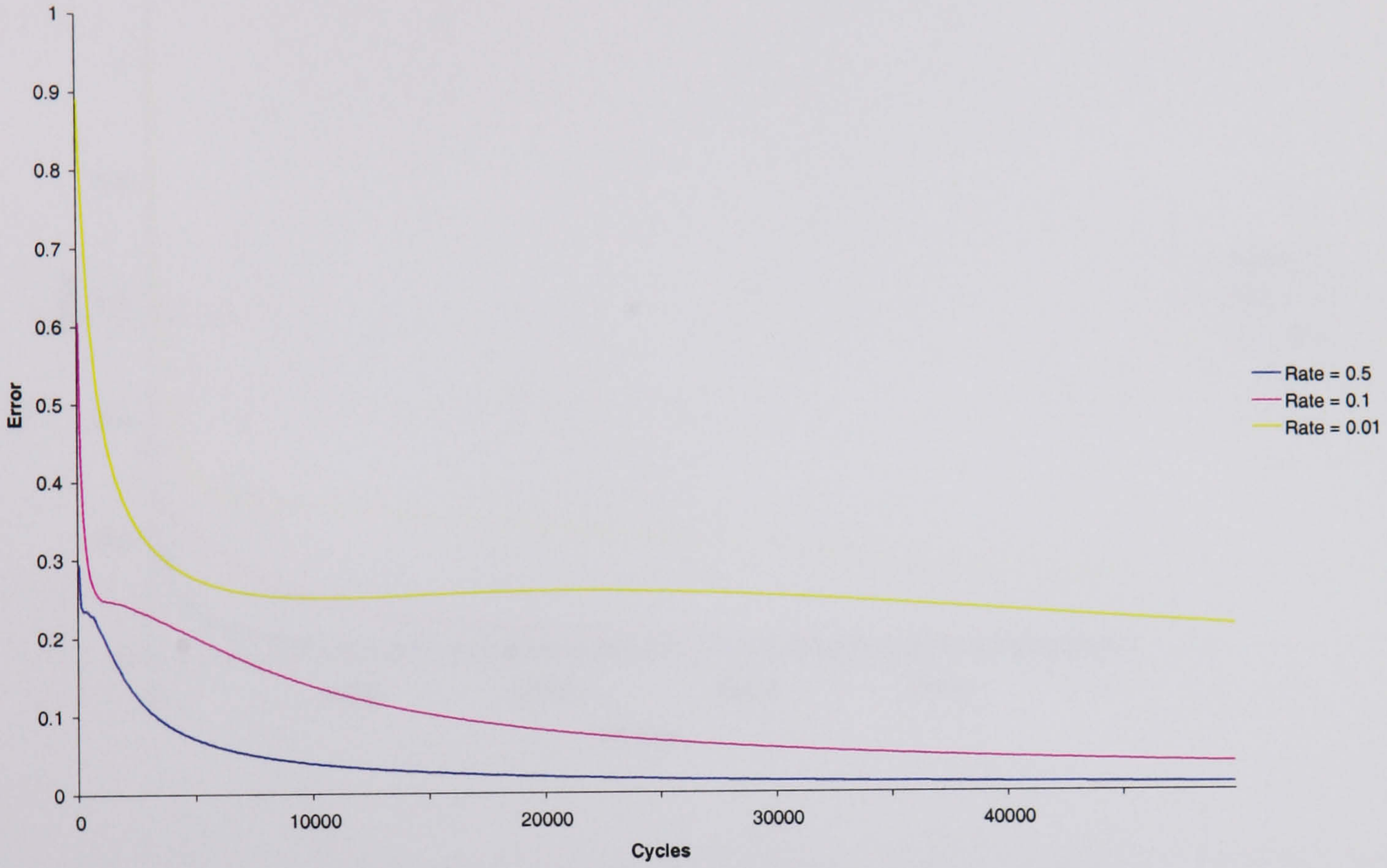


Figure 5.12 BPNN Error, Momentum = 0.5

With the larger momentum terms, the small rate of learning has its error forced down at a greater rate. However Figure 5.13 shows that if the momentum term is too large the initial error plunges too quickly with the risk of falling into local minima.

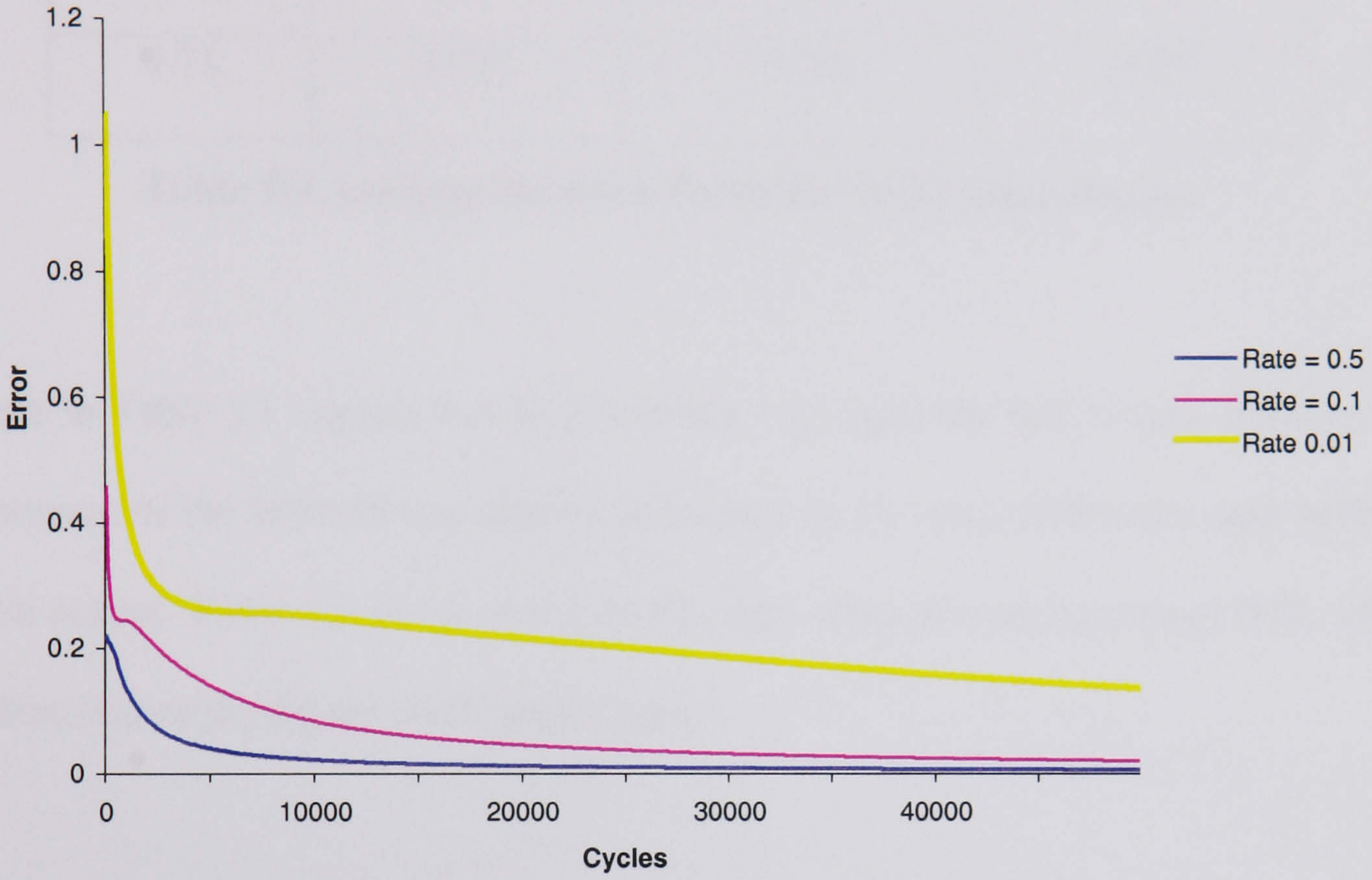


Figure 5.13 BPNN Error, Momentum = 0.75

After the network has trained to an acceptable error all the interconnecting weights between the neurons are saved to disk.

Another measure of the network error can be made when in a run time mode of operation. That is processing images that have not been part of the training set. Table 5.1 shows the average error of the network when considering ten Brodatz images, D4, D6, D9, D21, D24, D57, D73, D86, D93 and D106 from Section 6.2. The error being the percentage of the wanted output minus the actual output. 100% indicates total failure and 0% indicates complete success.

Momentum	Learning Rate		
	0.01	0.1	0.5
0.125	50.89	18.55	12.76
0.25	44.26	13.32	10.94
0.5	33.30	12.00	10.48
0.75	23.23	15.33	15.92

Table 5.1 Average Network Error for 10 Brodatz Images

The data in Table 5.1 implies that high learning rates give the best results, however the performance of the network can also be measured by the error of the test case with the weakest output. Table 5.2 shows the network error when processing image D73, which was identified as giving the worst performance.

Momentum	Learning Rate		
	0.01	0.1	0.5
0.125	76.66	93.49	85.93
0.25	71.20	75.01	81.45
0.5	59.88	30.33	60.62
0.75	77.43	86.60	95.10

Table 5.2 Percentage Network Error Brodatz Image D73

To achieve the best accuracy for the weakest case, Table 5.2 shows a learning rate of 0.1 used in conjunction with a momentum term of 0.5 gives the best results. This choice of parameters still gives an acceptable error for the ten Brodatz images (12.00) as shown in Table 5.1, even though a slightly lower error (10.48) could be achieved. However if the settings (learning rate of 0.5 and momentum 0.5) which gave this slightly lower average error (10.48 in Table 5.1) are used, then the error for the worst case (D73) increases significantly from 30.33 to 60.62 as seen in Table 5.2.

5.4 System Operation

During the training phases of the neural networks, the interconnecting weights are continuously adjusted to provide the smallest possible error term at each output layer. However in the run-time operation of the architecture, none of the weights are adjusted. Data from the image sample is propagated through both networks to produce a classification on the output layer of the back-propagation network.

This takes the form of :-

- Run a mask across a sample region in the image, mapping grey scale data on to the input layer of the self-organising map, hence creating the histogram of activity at the output layer. (To increase the accuracy of the system the sample region is bigger than the sample mask, typically the mask may be 9 by 9 pixels, whereas the sample region could be 25 by 25 pixels).
- Apply the histogram to the input layer of the back-propagation network.
- Monitor the output layer of back-propagation network for the neuron with the highest output, this neuron indicates which texture has been classified.

An example of this process was illustrated by Arrowsmith et al [Arrowsmith et al 1999], where the hybrid architecture was applied as a pre-processor for Spatial Grey Level Dependence Matrices. A training and a run-time (testing) set of textures were created. To calculate the wanted output, the repetitive elements embedded with the textures were found by means of cross-correlation and manual inspection. This training set was used in the network set up resulting in a system that was capable of classifying similar textures. To test the performance of the architecture, the run-time set of images was used the network's input.

The key elements of system operation were as follows:-

- The self-organising layer trained upon ten sample textures.
- Each of the ten samples was then processed again to produce ten histograms of activity of the self-organising map's output layer.
- The ten histograms were then used as inputs to train the back-propagation neural network. They were applied in the same order as the wanted output being expressed at the output layer of the back-propagation network for the calculation of the error term.
- After training, all the interconnecting weights from both networks are saved to disk.
- To analyse a texture, the mask feeding the input layer of the self-organising map is applied to a sample region of the image in question.
- As the mask traverses the sample, the activity upon the output layer is built up into the interconnecting histogram.
- This histogram is applied to the input layer of the back-propagation neural network and the output of the network is calculated.
- The most active neuron upon the output layer of the back-propagation neural network indicates the texture within the sample.

Note:- When considering the Brodatz image set, only one histogram is required per texture. However when processing other types of textures such as the ones found in the aerial imagery in the next chapter, multiple histograms are required to represent a single texture label.

5.5 Summary

The hybrid neural network takes the best elements of the back-propagation neural network and the self-organising map neural network to create a system with strong classification properties. These properties concentrate upon extracting meaning and understanding from the large data sets found in digital image processing environments.

The proposed architecture and training cycles in this chapter are capable of classifying texture features in images. The training set is the key to successful operation. It needs to contain enough samples of the textures it is required to classify. If the training set is constructed appropriately, the hybrid neural network is capable of a range of image digital processing applications, as demonstrated in Chapter 6.

Chapter 6 Experimental Results

The experimental results presented in this chapter aim to prove the versatility of the hybrid neural network when working in a digital image processing environment.

Section 6.1 outlines some of decisions taken when determining the imagery used for producing the experimental results, and to justify the images chosen.

The hybrid architecture is shown working as a classifier in Section 6.2, for which some sample imagery taken from the Brodatz series.

Some tuning of the architecture is also explored in Section 6.3.

Section 6.4 proposes another version of the hybrid neural network architecture capable of segmenting images. This section also evaluates the performance of hybrid neural network performance when segmenting real world images. A discussion of the performance of the hybrid architecture versus that of conventional spatial grey level dependence matrix feature generator classified by a back-propagation neural network is also provided.

A resume of the achievements presented in this chapter is concluded in Section 6.5.

6.1 Introduction

Since Haralick et al. [Haralick et al 1973] did his early work on texture analysis, the Brodatz series of textured images has been used a benchmark for researchers in the texture analysis field. Therefore their inclusion as images in the production of experimental data was deemed to be necessary. A hybrid neural network architecture, presented in Section 6.2.2, processes Brodatz images with the aim of classifying them based upon the periodic structures embedded within them.

However the main element of the work provided in this thesis is the hybrid neural network working upon real world images. The hybrid architecture is applied to real world images with the aim of deriving meaning from them via a segmentation process. In the work presented here, the real world images are taken from an aerial photographic mapping survey. The hybrid architecture is applied to them with the aim of identifying textures similar to those in its training set.

The same real world images also have spatial grey level dependence matrices applied to them which are in turn classified by a back-propagation neural network with the aim of producing a critical comparison of the two methodologies.

6.2 Brodatz Series of Texture Images

6.2.1 Training Images

Ten samples were identified in the Brodatz album [Brodatz 1966] that contained a wide variety of texture types.

The original Brodatz image series were labelled D1 to D112. To retain consistency the D label and number has been retained in this document. Each image is monochrome with the dimensions of 64 by 64 pixels using 256 grey scales and histogram equalised using the process from section 1.1.

Samples of the images used follow:-

- D4 (Pressed Cork), Figure 6.1
- D6 (Woven Aluminium), Figure 6.2
- D9 (Grass Lawn), Figure 6.3
- D21 (French Canvas), Figure 6.4
- D24 (Pressed Calf Leather), Figure 6.5
- D57 (Handmade Paper), Figure 6.6
- D73 (Soap Bubbles), Figure 6.7
- D86 (Ceiling Tile), Figure 6.8
- D93 (Fur), Figure 6.9
- D106 (Cheesecloth), Figure 6.10

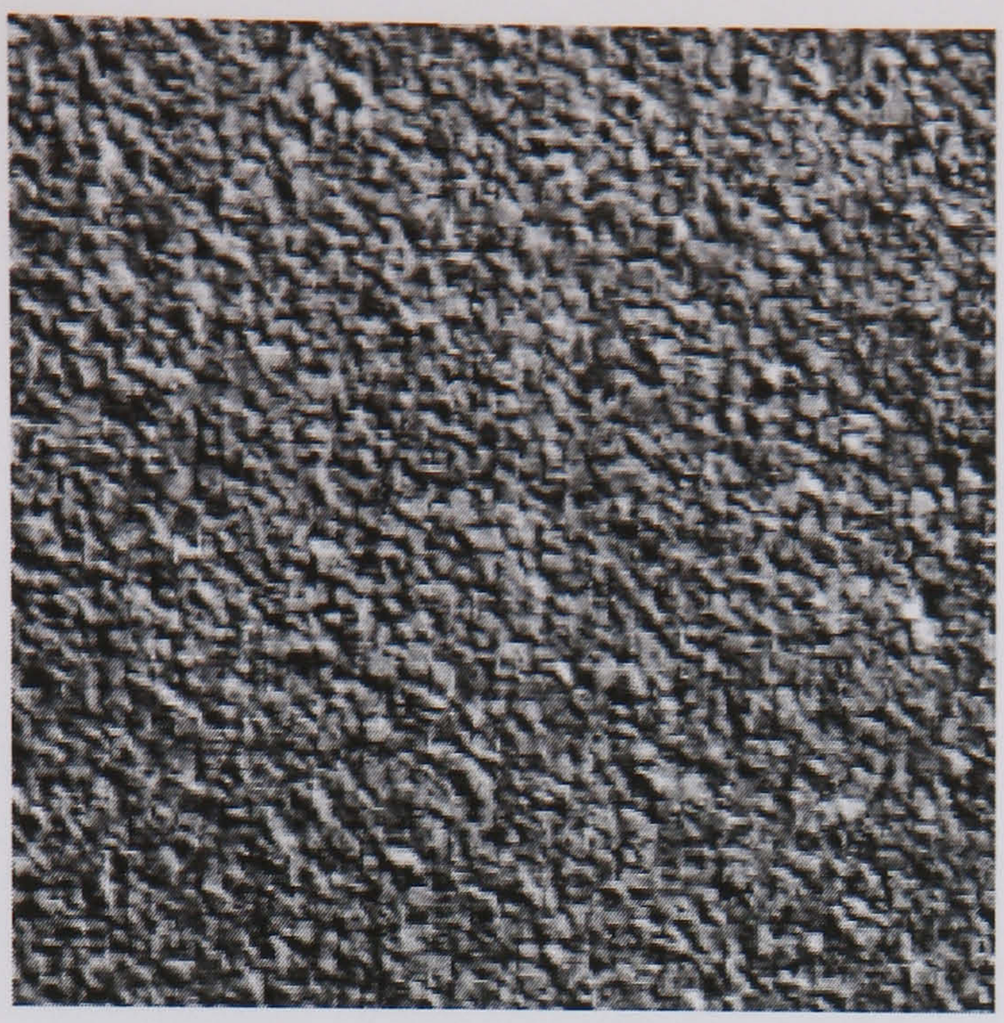


Figure 6.1 D4 (Pressed Cork)

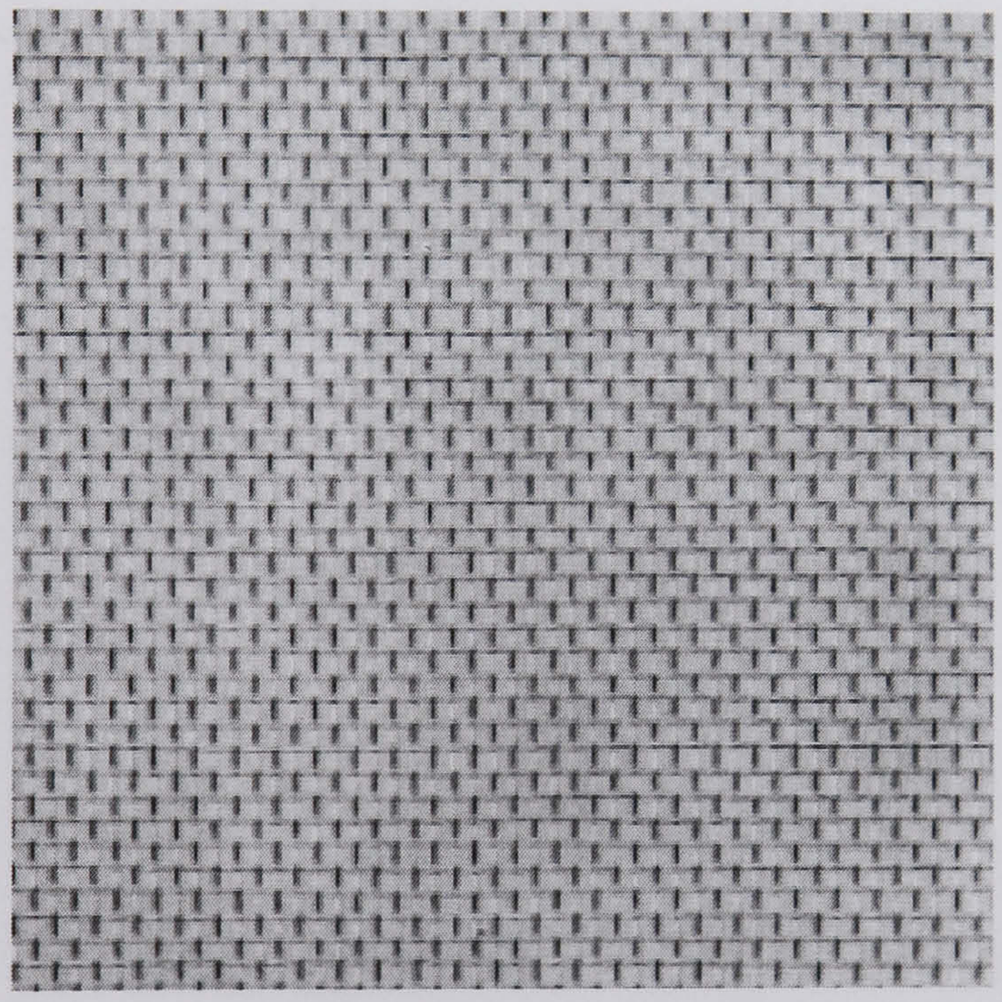


Figure 6.2 D6 (Woven Aluminium)

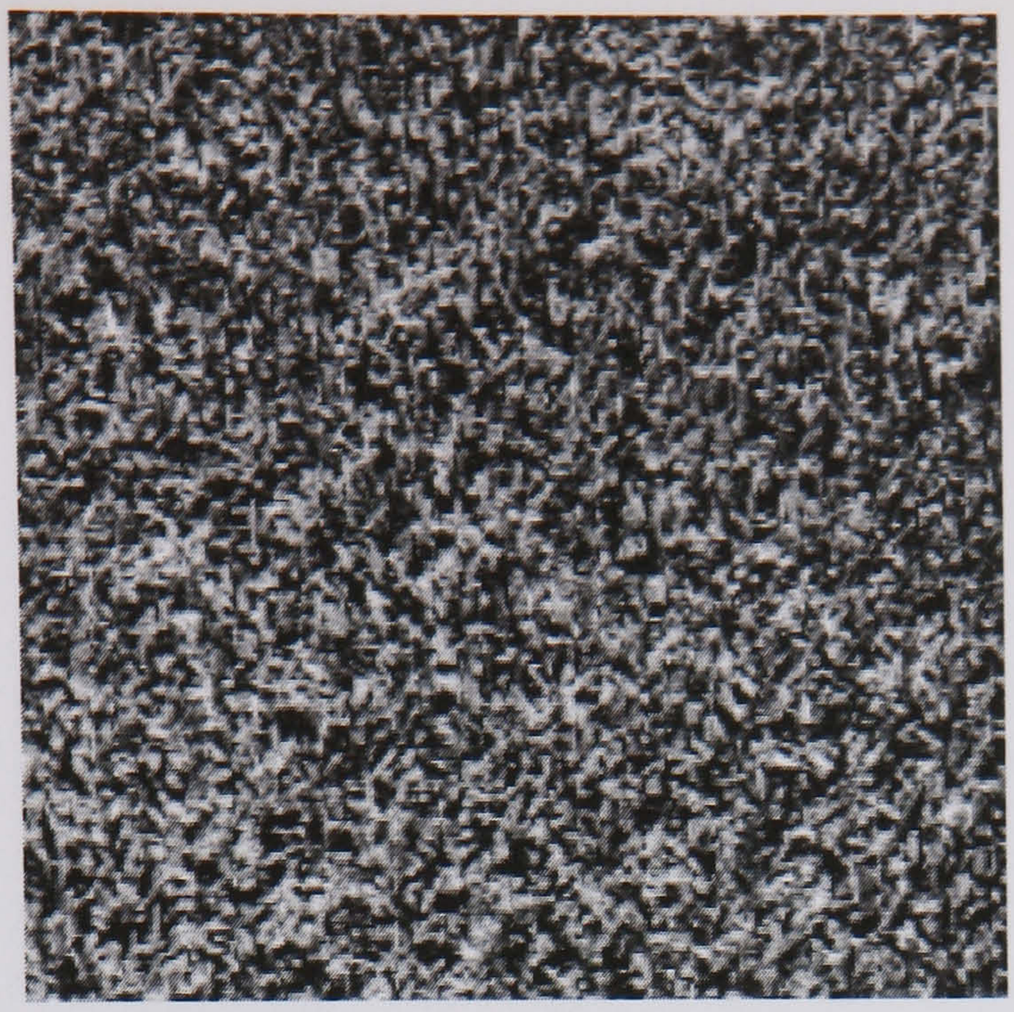


Figure 6.3 D9 (Grass Lawn)

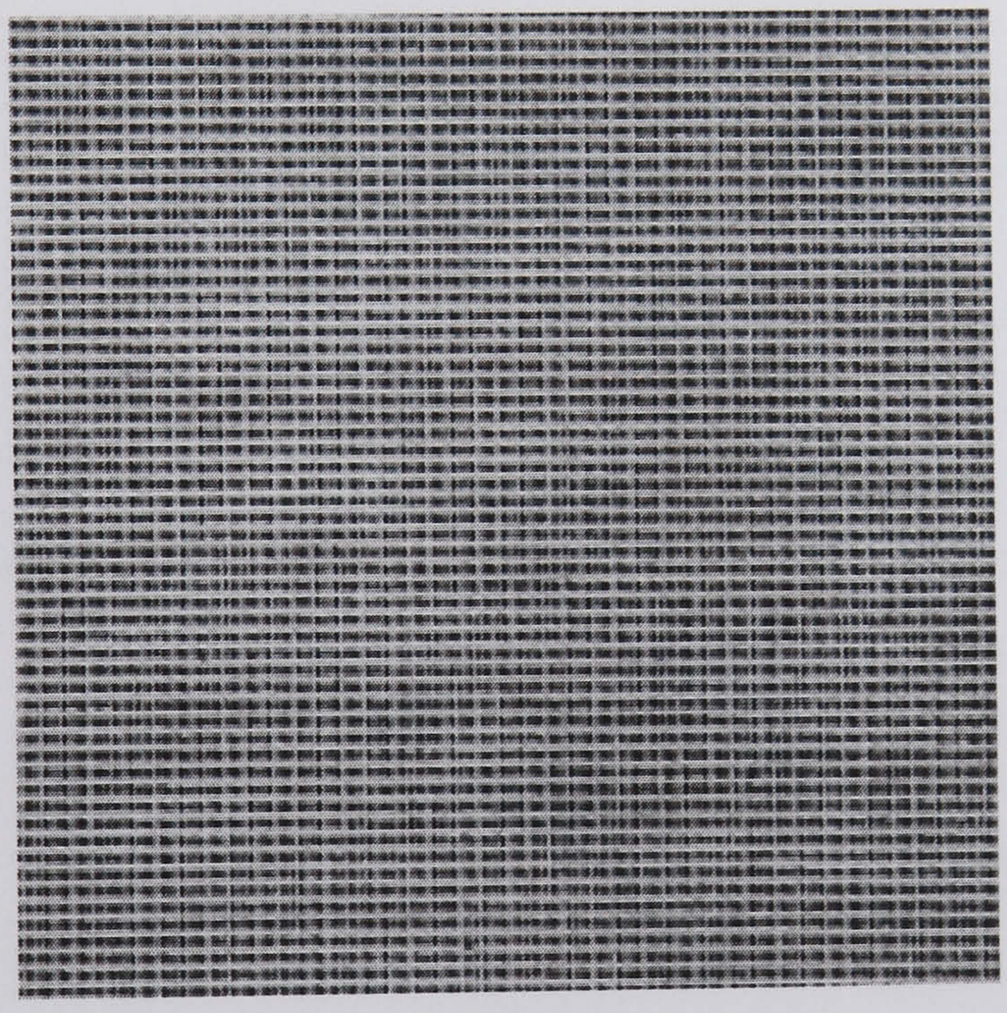


Figure 6.4 D21 (French Canvas)

Figure 6.5 D24 (Pressed Calf Leather)

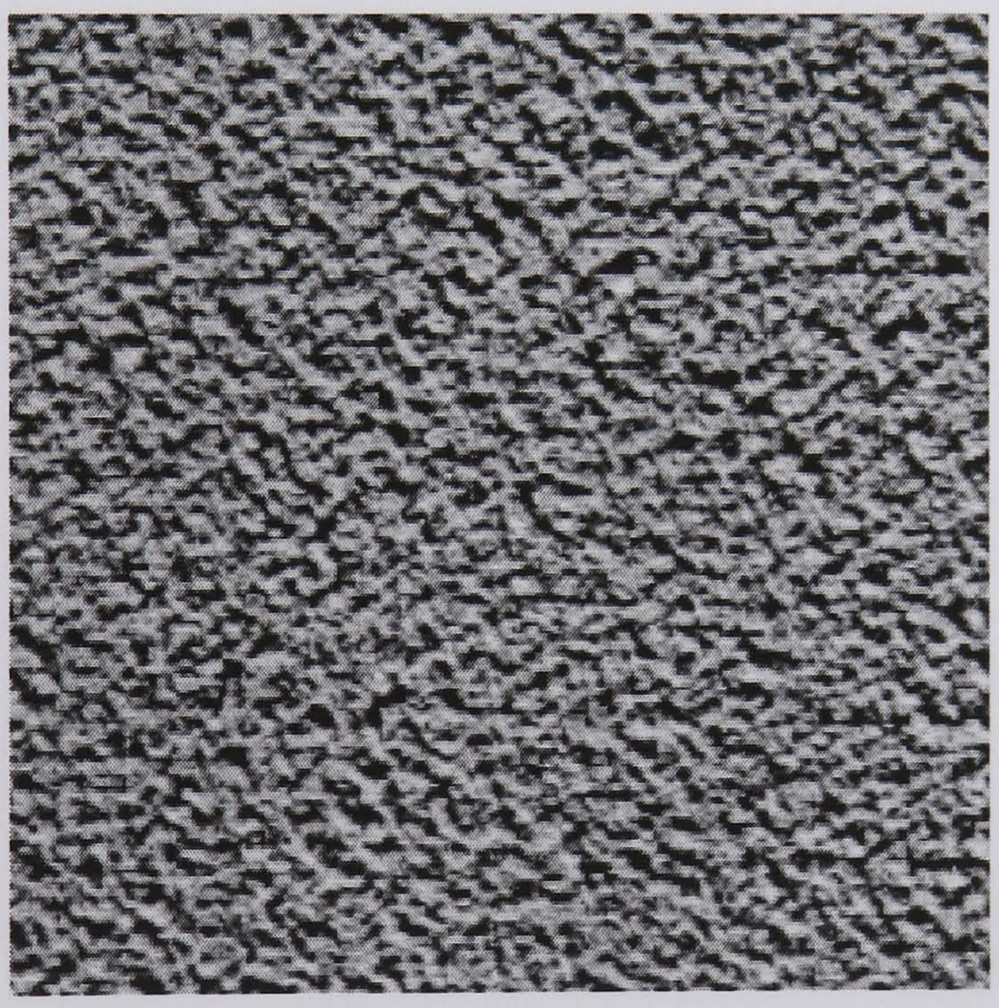


Figure 6.6 D57 (Handmade Paper)

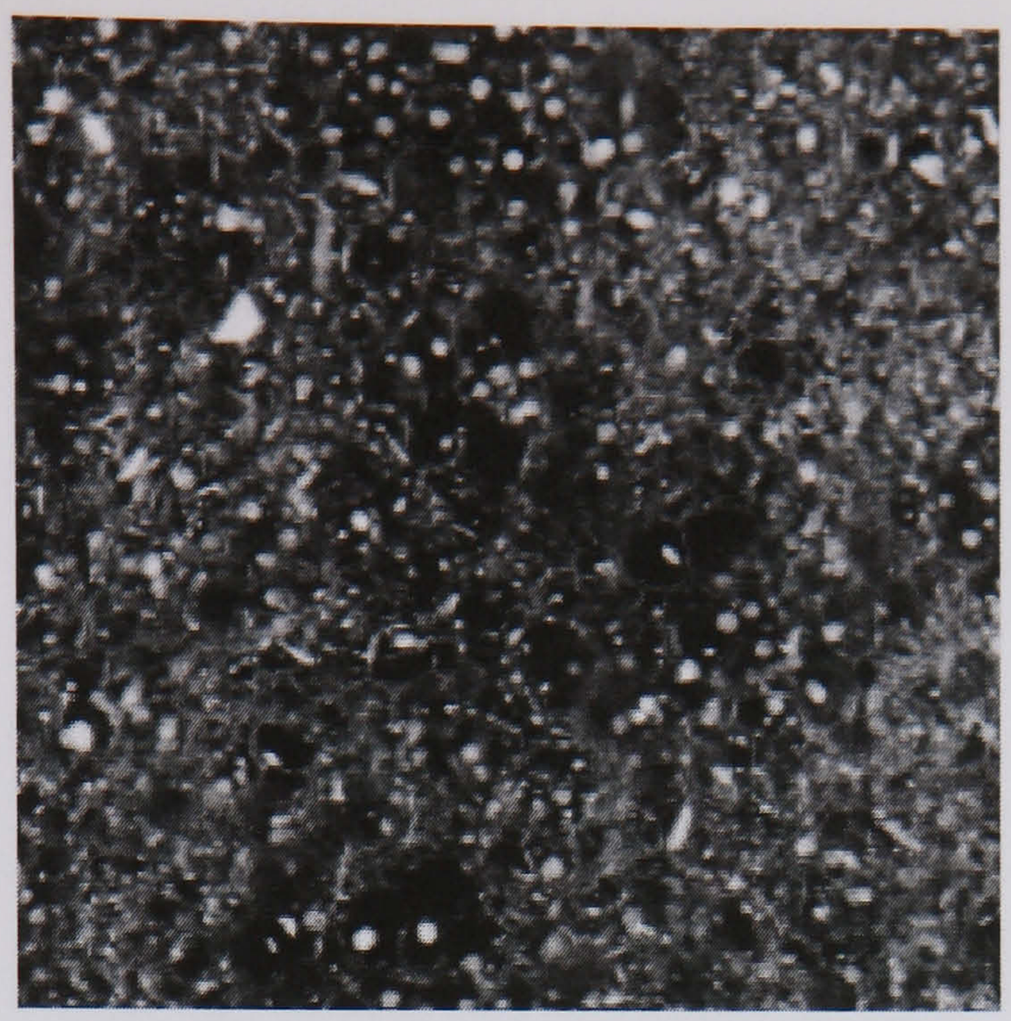


Figure 6.7 D73 (Soap Bubbles)

Figure 6.8 D86 (Ceiling Tile)

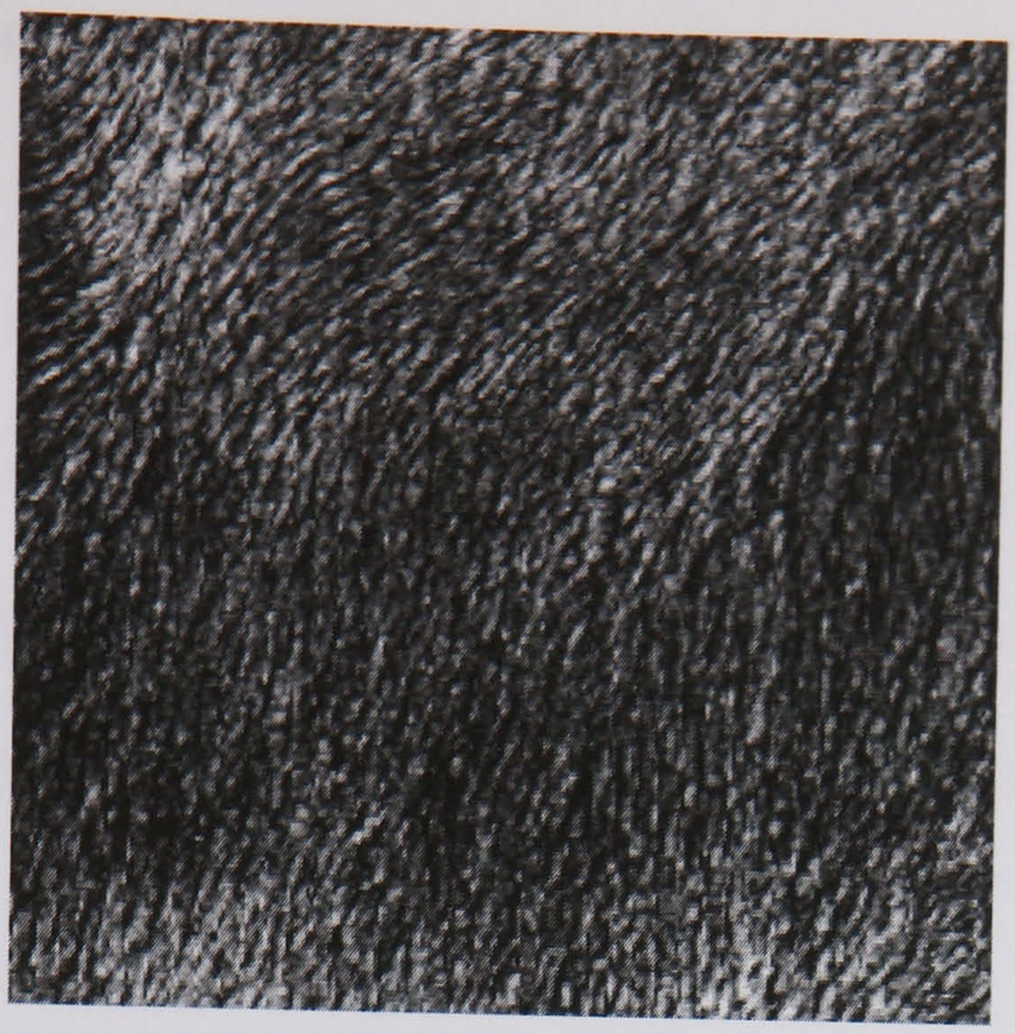


Figure 6.9 D93 (Fur)

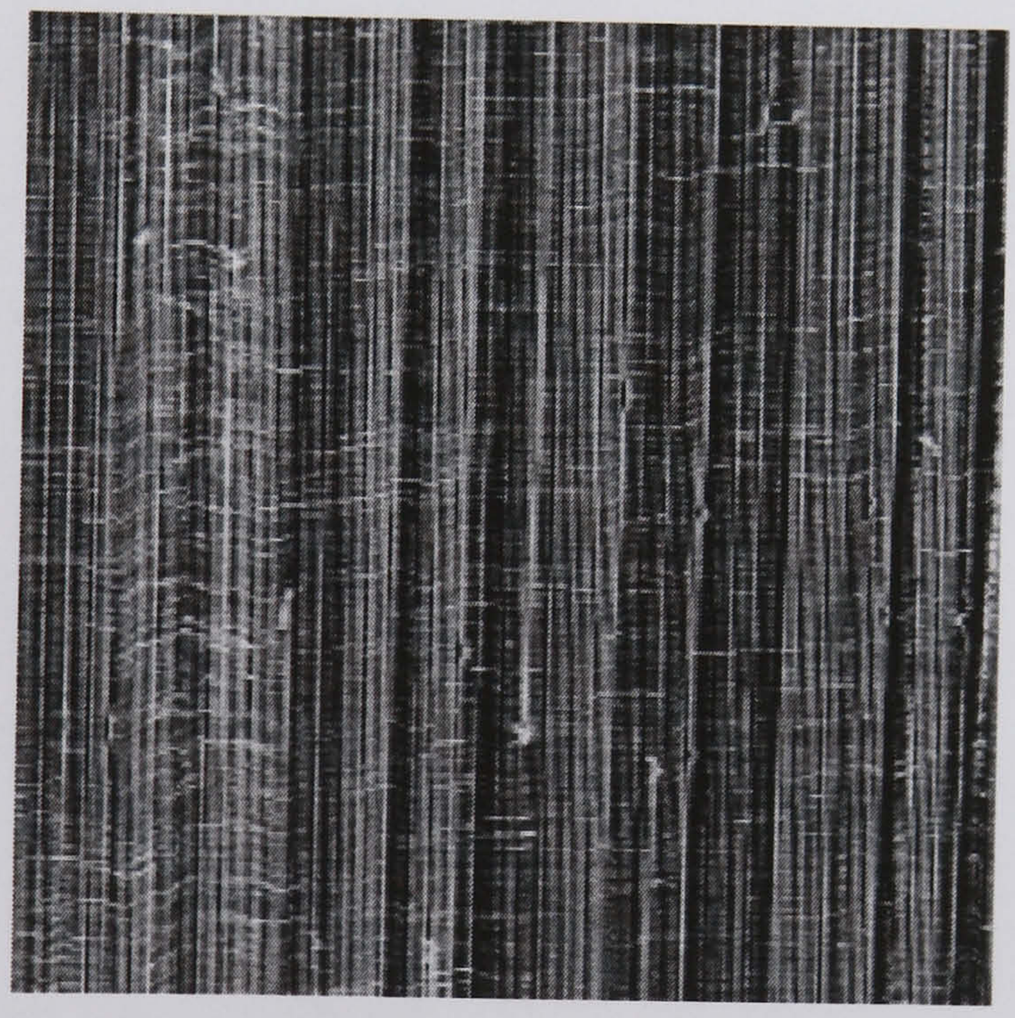


Figure 6.10 D106 (Cheesecloth)

6.2.2 Hybrid Network Architecture and Training

The Hybrid Neural Network dimensions were acquired through a process of experimental tests upon the Brodatz images with different hybrid network sizes. The optimum size is being the smallest architecture offering the highest accuracy. The smallest architecture is desirable to give quick processing times when implemented in software.

The parameters can be seen in below.

i) SOM Input Layer 3 by 3 neurons.

SOM Output Layer 2 by 2 neurons.

BPNN Input Layer 2 by 2 neurons.

BPNN Middle Layer 1 neuron.

ii) SOM Input Layer 5 by 5 neurons.

SOM Output Layer 2 by 2 neurons.

BPNN Input Layer 2 by 2 neurons.

BPNN Middle Layer 1 neuron.

iii) SOM Input Layer 5 by 5 neurons.

SOM Output Layer 3 by 3 neurons.

BPNN Input Layer 3 by 3 neurons.

BPNN Middle Layer 1 neuron.

iv) SOM Input Layer 7 by 7 neurons.

SOM Output Layer 3 by 3 neurons.

BPNN Input Layer 3 by 3 neurons.

BPNN Middle Layer 2 by 2 neurons.

v) SOM Input Layer 7 by 7 neurons.

SOM Output Layer 5 by 5 neurons.

BPNN Input Layer 5 by 5 neurons.

BPNN Middle Layer 3 by 3 neurons.

vi) SOM Input Layer 9 by 9 neurons.

SOM Output Layer 5 by 5 neurons.

BPNN Input Layer 5 by 5 neurons.

BPNN Middle Layer 3 by 3 neurons.

The BPNN Output layer was always ten neurons to represent the ten different images.

	i	ii	iii	iv	v	vi
D4	✗	✗	✓	✓	✓	✓
D6	✗	✗	✗	✗	✗	✓
D9	✗	✗	✗	✗	✓	✓
D21	✗	✗	✗	✓	✓	✓
D24	✗	✗	✓	✓	✓	✓
D57	✗	✗	✓	✓	✓	✓
D73	✗	✗	✗	✓	✓	✓
D86	✗	✗	✗	✓	✓	✓
D93	✗	✗	✓	✓	✓	✓
D106	✗	✗	✓	✓	✓	✓

Table 6.1 Experimental hybrid network sizes

Where:- ✓ indicates correct classification.

✗ indicates incorrect classification.

The optimum hybrid network uses the dimensions of :-

- Self-organising map input layer 9 by 9 neurons
- Self-organising map output layer 5 by 5 neurons
- Back-propagation network input layer 5 by 5 neurons
- Back-propagation network middle layer 3 by 3 neurons
- Back-propagation network output layer 10 by 1 neurons

For each sample two images were created, one for use in the training of the system and the other to test the performance of the system. Both images were created by sampling the original Brodatz image at two different and distinct positions.

6.2.3 System Performance

Table 6.2 shows the output from the hybrid neural network when trained upon the Brodatz images listed previously, Figure 6.1through to Figure 6.10. Each row contains the output of the ten neurons in the output layer of the back-propagation neural network.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0051	0.0000	0.0111	0.0126	0.0076	0.0154	0.0000	0.0000	0.0058	0.9678
D93	0.0252	0.0007	0.0021	0.0265	0.0008	0.0034	0.0415	0.0000	0.9513	0.0018
D4	0.0002	0.0275	0.0203	0.0000	0.0193	0.0220	0.0269	0.9591	0.0000	0.0015
D73	0.0015	0.0258	0.0003	0.0003	0.0050	0.0285	0.9427	0.0052	0.0413	0.0002
D24	0.0007	0.0018	0.0000	0.0001	0.0015	0.9662	0.0219	0.0235	0.0004	0.0255
D21	0.0181	0.0012	0.0167	0.0009	0.9726	0.0003	0.0000	0.0034	0.0000	0.0103
D86	0.014	0.0200	0.0118	0.9697	0.0018	0.0000	0.0000	0.0000	0.0177	0.0172
D57	0.0002	0.0194	0.9699	0.0162	0.0095	0.0000	0.0000	0.0079	0.0002	0.0105
D9	0.0000	0.9577	0.0171	0.0065	0.0094	0.0001	0.0214	0.0276	0.0000	0.0001
D6	0.9679	0.0000	0.0014	0.0088	0.0183	0.0039	0.0001	0.001	0.0218	0.0106

Table 6.2 Hybrid Output from the Brodatz Training Set

Figure 6.11 gives a pictorial view of the activity of the neurons on the output layer. A neuron that fires with a value close to one indicates a classified texture.

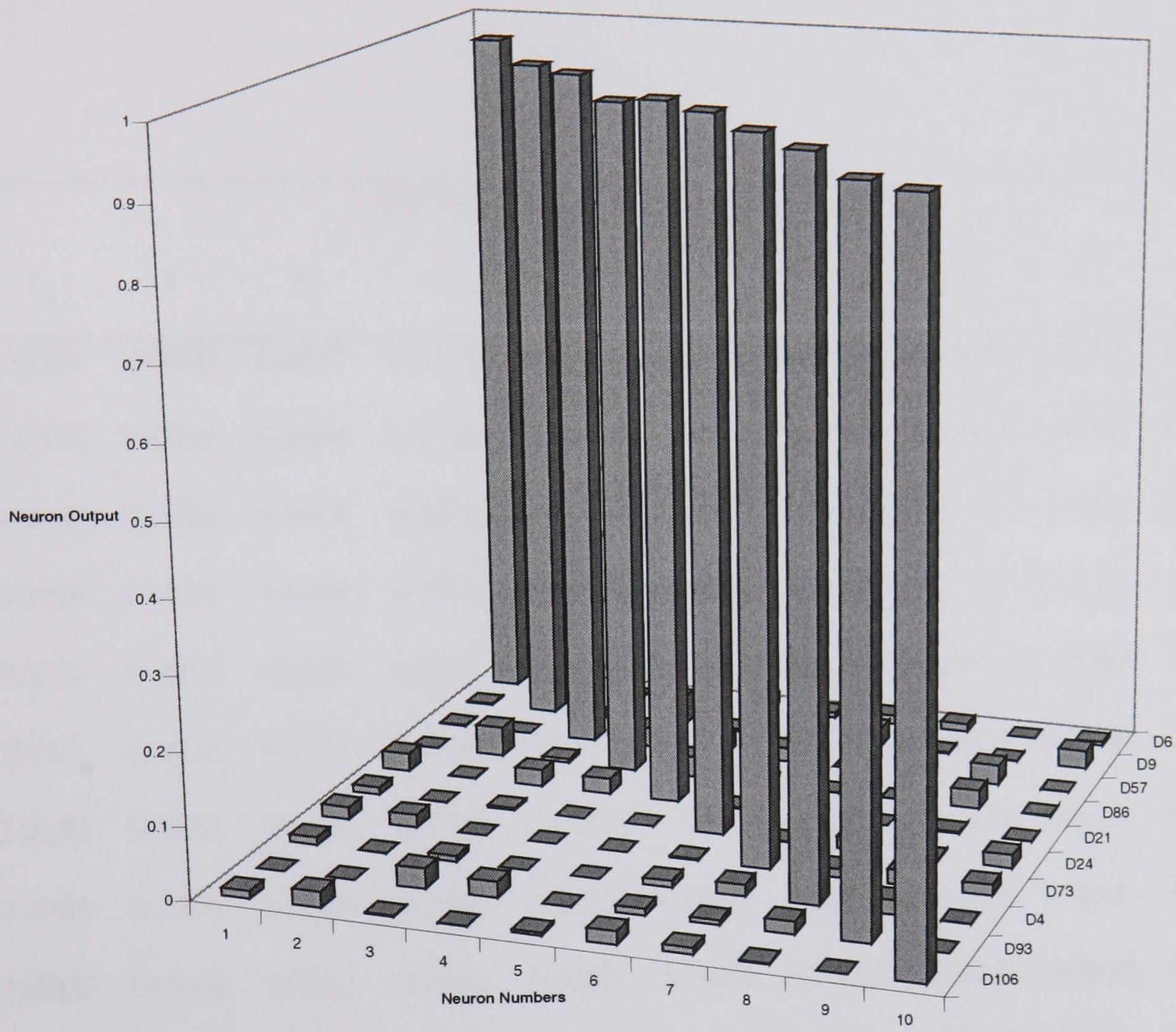


Figure 6.11 Hybrid Output from the Brodatz Training Set

A run-time set of images was constructed to test the performance of the hybrid architecture. These images contain the same type of texture to that of those used in the training phase, however they contain completely different pixels. When the run-time set of images are applied to the system, the following results are displayed in Table 6.3.

Images	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0026	0.0000	0.0197	0.0163	0.0024	0.0178	0.0000	0.0000	0.0767	0.9656
D93	0.0332	0.0012	0.0094	0.0769	0.0009	0.0006	0.0222	0.0000	0.9704	0.0013
D4	0.0000	0.3433	0.0953	0.0003	0.0201	0.0022	0.0290	0.9307	0.0000	0.0010
D73	0.0009	0.0220	0.0000	0.0002	0.0045	0.0931	0.9092	0.0027	0.0217	0.0004
D24	0.0026	0.0017	0.0000	0.0001	0.0017	0.9645	0.3930	0.0097	0.0069	0.0152
D21	0.017	0.0012	0.0170	0.0008	0.9724	0.0003	0.0000	0.0034	0.0000	0.0096
D86	0.0188	0.0184	0.0090	0.9793	0.0019	0.0000	0.0000	0.0000	0.0276	0.0266
D57	0.0000	0.1991	0.9365	0.0069	0.0391	0.0000	0.0001	0.0328	0.0000	0.0074
D9	0.0000	0.9224	0.0052	0.0051	0.0058	0.0002	0.0552	0.0162	0.0000	0.0000
D6	0.9679	0.0000	0.0013	0.0088	0.0183	0.0039	0.0001	0.0016	0.0218	0.0106

Table 6.3 Hybrid output from the Brodatz run-time set

The pictorial information presented in Figure 6.12 highlights the fact that although there is more 'background noise' from all the neurons, there are still strong outputs available for texture classification.

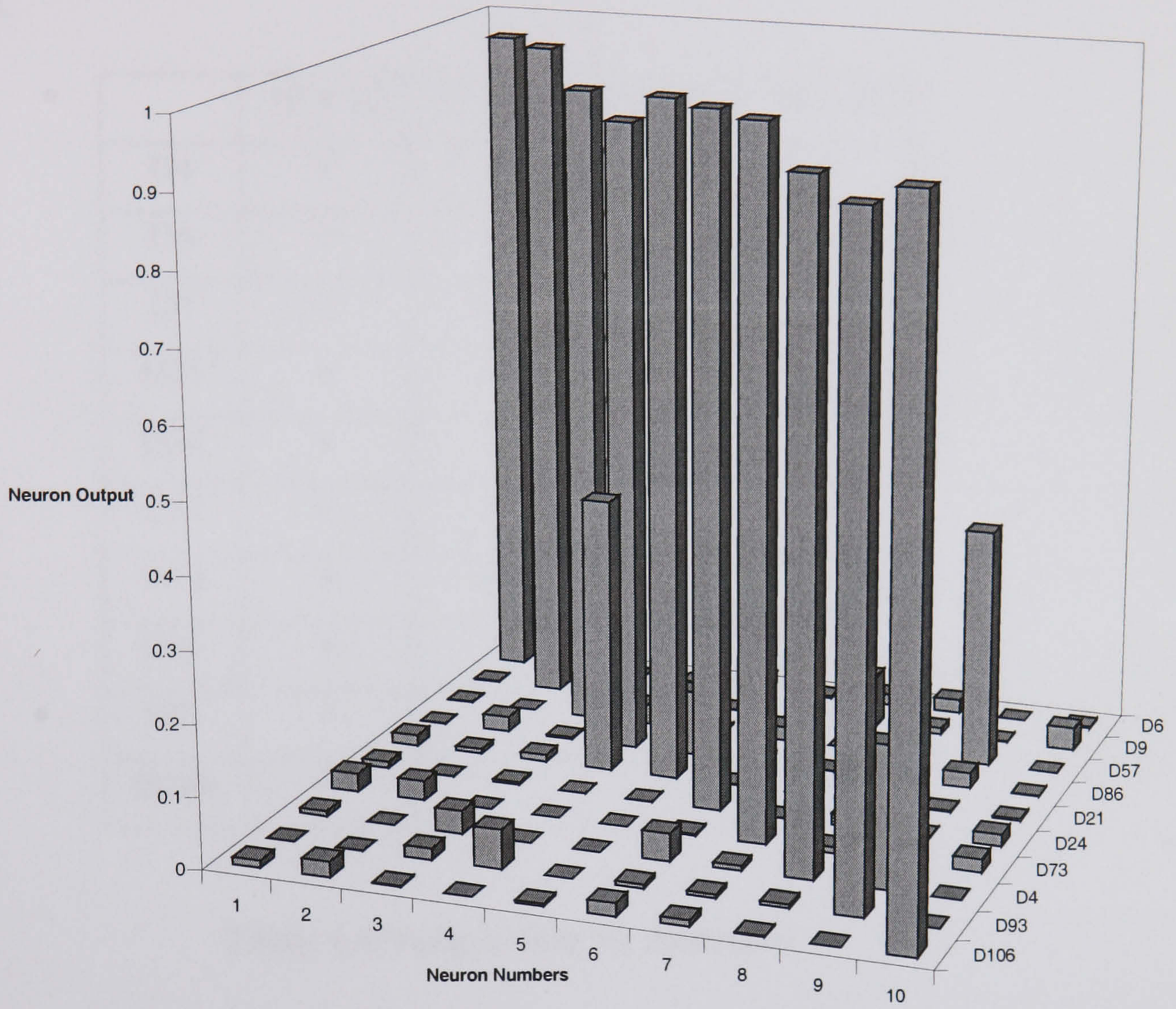


Figure 6.12 Hybrid output from the Brodatz run-time set

The additional noise that is presented when working on the run-time images is not considered to be a problem. By searching through all the neurons' output values, the neuron with the highest value indicates which texture has been sampled.

Although the test images are 64 by 64 pixels in size, the sample does not have to be of the same size. Using the same system architecture from section 6.2 the following results in Table 6.4 were achieved using the same run-time set of ten images using variable sample sizes. The columns show the sample size and the rows represent the Brodatz image titles.

	10 x 10	15 x 15	20 x 20	25 x 25
D4	✓	✓	✓	✓
D6	✓	✓	✓	✓
D9	✓	✓	✓	✓
D21	×	✓	✓	✓
D24	✓	✓	✓	✓
D57	✓	✓	✓	✓
D73	×	×	✓	✓
D86	×	×	✓	✓
D93	✓	✓	✓	✓
D106	×	×	×	✓

Table 6.4 Sample Size vs. Accuracy

Where:- ✓ indicates correct classification. × indicates incorrect classification.

Small samples are adequate for textures which are strongly ordered with a periodicity covering a small number of pixels such as Figure 6.4 D21 (French Canvas). However to encompass all the textures in the training / run-time set a large general-purpose sample region must be used. This is of the order of 25 by 25 pixels.

Note:- The sample size is the region that the mask feeding the input layer of the self-organising map traverses, it is not the size of the mask.

6.2.4 Comparison of Hybrid Architecture Against a "Glue Less" Interface

The accuracy of the histograms of the hybrid architecture can be seen when comparing the results from section 6.2.3 to the same neural network architecture but without the histogram interface. In this case the interface between the Self-organising map and the Back-propagation network could be said to be "glue less". The training of the network is the same as the hybrid network with regards to the self-organising layer, all images are trained upon and the interconnecting weights are saved after the training phase is complete. However instead of re-applying the self-organising network to the images to generate the interconnecting histograms, its output is used to directly feed the input layer of the back-propagation network for its training phase. This training process takes the form of running of the self-organising network across all the images in the training set applying the self-organising output layer to the input layer of the back-propagation network.

Using exactly the same suite of images used in training the hybrid network, the "glue less" network was trained as described above to give the results presented in Table 6.5.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0000	**0.2194**	0.0224	0.0299	0.0003	0.0431	0.0048	0.1270	0.032	0.0915
D93	0.0000	0.0017	0.0000	0.0000	0.0000	0.0019	0.2048	0.0440	**0.8251**	0.0000
D4	0.0000	0.1613	0.0204	0.0231	0.0003	0.0404	0.0343	0.1546	**0.2176**	0.0169
D73	0.0000	0.0187	0.0003	0.0176	0.0000	0.0076	0.3427	0.0582	**0.5733**	0.0003
D24	0.0000	0.0543	0.0007	0.0019	0.0000	0.1532	0.0428	0.1353	**0.2235**	0.0065
D21	0.0331	0.0138	0.0004	0.0017	**0.9564**	0.0002	0.0000	0.0007	0.0008	0.0000
D86	0.0011	0.0940	0.0101	**0.6954**	0.0023	0.0064	0.0351	0.0000	0.0012	0.0052
D57	0.0002	**0.3161**	0.1650	0.0442	0.0139	0.0071	0.0000	0.0865	0.0093	0.0185
D9	0.0000	**0.1822**	0.0354	0.0206	0.0024	0.0568	0.0233	0.1475	0.1425	0.0152
D6	**0.9866**	0.0386	0.0008	0.0055	0.0106	0.0009	0.0000	0.0066	0.0208	0.0000

Table 6.5 "Glue Less" Network Training Results

The boxes highlighted indicate which neuron should be the highest firing for each individual texture in the training set. Values in bold type indicate the highest neuron output for that particular image under test. There is a 50% failure in the training phase with the network failing to produce a distinct neuron for textures D106, D4, D73, D24 and D57.

A graphical representation of this data presented in Figure 6.13.

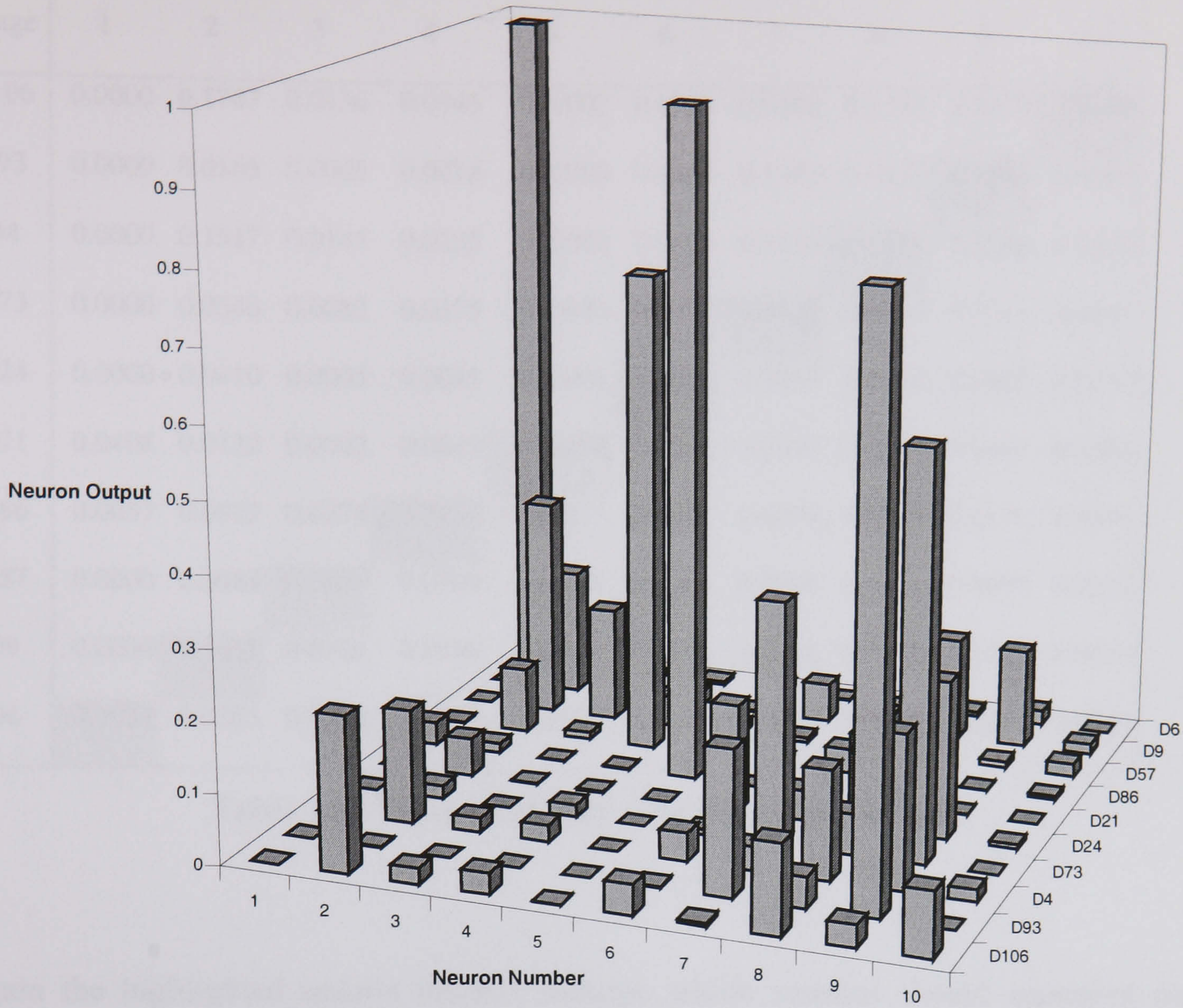


Figure 6.13 "Glue Less" Network Training Results

Unlike the training phase of the hybrid architecture there is considerably more noise in the neurons output. Only textures D93, D21 and D6 have near maximum (1.0) neuron outputs.

Applying the run time set of images used previously with the hybrid architecture gives the results in Table 6.6.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0000	**0.1787**	0.0136	0.0245	0.0002	0.0750	0.0102	0.1188	0.0473	0.0885
D93	0.0000	0.0165	0.0005	0.0018	0.0000	0.0045	0.1952	0.0537	**0.7383**	0.0003
D4	0.0000	0.1517	0.0145	0.0235	0.0002	0.0420	0.0374	0.1561	**0.2546**	0.0156
D73	0.0000	0.0348	0.0082	0.0375	0.0020	0.0062	**0.2822**	0.0578	0.5721	0.0047
D24	0.0000	0.0410	0.0005	0.0015	0.0001	0.1660	0.0453	0.1110	**0.1967**	0.0112
D21	0.0408	0.0122	0.0002	0.0019	**0.9468**	0.0002	0.0000	0.0006	0.0007	0.0000
D86	0.0057	0.0822	0.0274	**0.7151**	0.0115	0.0019	0.0024	0.0324	0.0136	0.0035
D57	0.0000	**0.3084**	0.0659	0.0704	0.0019	0.0141	0.0006	0.1326	0.0096	0.0215
D9	0.0000	**0.1669**	0.0308	0.0190	0.0030	0.0658	0.0288	0.1350	0.1479	0.0210
D6	**0.9837**	0.0383	0.0008	0.0055	0.0129	0.0009	0.0000	0.0065	0.0205	0.0000

Table 6.6 "Glue Less" Network Run Time Results

Again the highlighted neuron outputs indicate which neurons should represent each individual texture. As with the training data there is a 50% failure in identification of the appropriate texture.

Figure 6.14 shows a large amount of noise generated by the network when the run time set of images are presented to the “glue less” network.

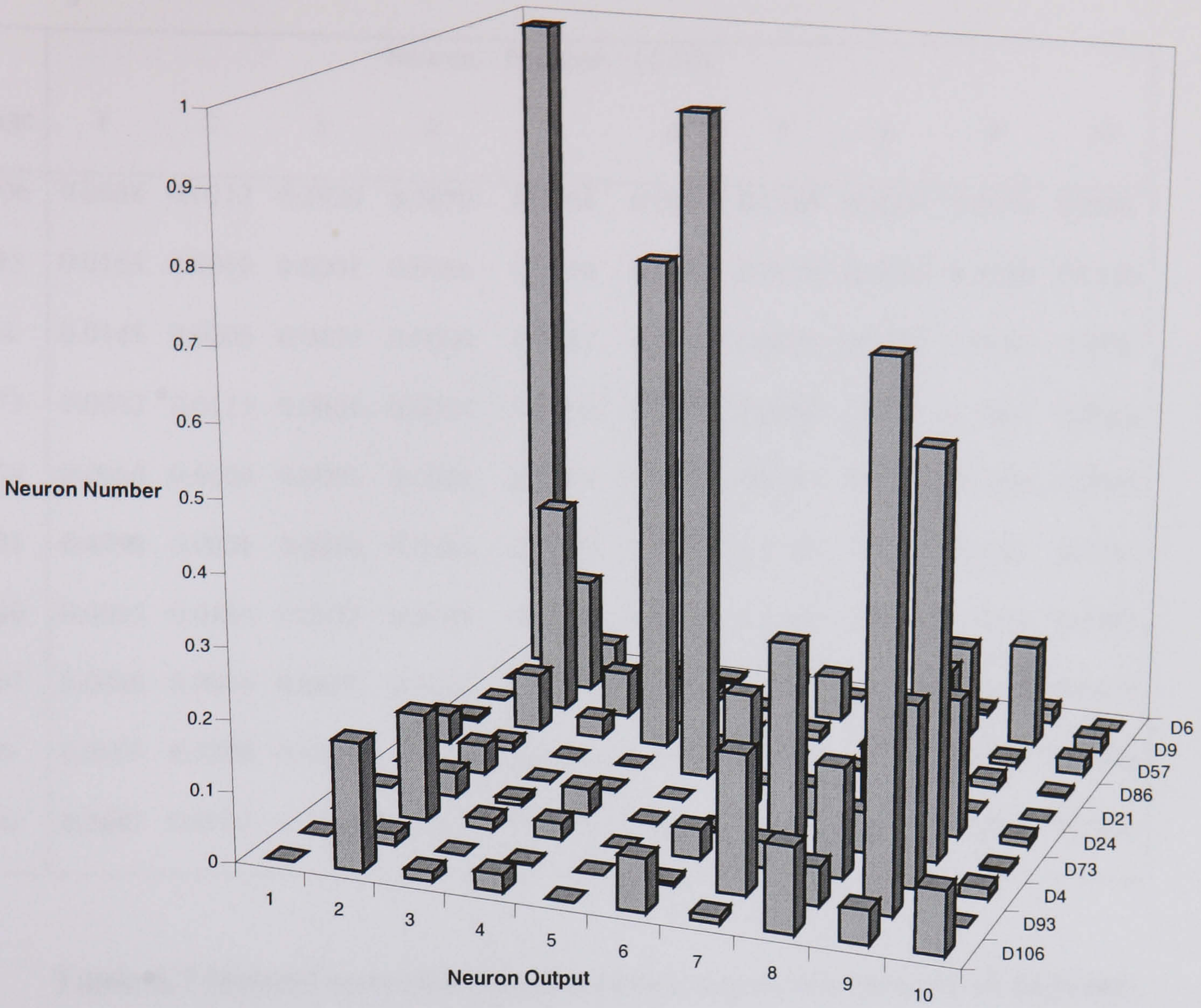


Figure 6.14 "Glue Less" Network Run Time Results

6.2.5 Vulnerability of the training process.

The accuracy of the hybrid neural network is directly related to the quality of the images in its training set. This is borne out of the next two examples. If the textures are rotated through forty five and ninety degrees the failure rate increases to thirty and sixty percent respectively as shown in **Table 6.7** and **Table 6.8**. To overcome these failures extra images would have to be added to the training set that typically would represent all types of image that may be encountered when running.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0038	0.0213	0.3930	**0.7670**	0.0064	0.0001	0.0165	0.0004	0.0024	0.0091
D93	0.0163	0.0019	0.0001	0.0366	0.0159	0.0014	0.0118	0.0000	**0.9790**	0.0148
D4	0.0145	0.0205	0.0429	0.0000	0.0187	0.0040	0.0039	**0.9601**	0.0000	0.004
D73	0.0032	0.0113	0.0001	0.0054	0.0009	0.0099	**0.5659**	0.0001	0.3867	0.0023
D24	0.0060	0.0109	0.0001	0.0024	0.0009	0.0295	**0.8557**	0.0003	0.1284	0.0016
D21	**0.4798**	0.0001	0.0036	0.0046	0.2970	0.0003	0.0008	0.0037	0.0125	0.0081
D86	0.0025	0.0183	0.0232	**0.9669**	0.0102	0.0001	0.0080	0.0001	0.0064	0.0107
D57	0.0046	0.0094	**0.8607**	0.2918	0.0058	0.0000	0.0163	0.0023	0.0009	0.0122
D9	0.0005	**0.7299**	0.0886	0.0568	0.0028	0.0009	0.0256	0.0038	0.0005	0.0006
D6	**0.5687**	0.0001	0.0038	0.0017	0.1931	0.0015	0.0008	0.0095	0.0132	0.0201

Table 6.7 Hybrid output when run time images are rotated 45 degrees.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0051	0.0745	0.0086	**0.6115**	0.0009	0.0012	0.1093	0.0000	0.0772	0.0029
D93	0.0202	0.0013	0.0001	0.0564	0.0151	0.0016	0.0081	0.0000	**0.9747**	0.0219
D4	0.0015	**0.2724**	0.0545	0.0004	0.0097	0.0004	0.0080	0.2682	0.0000	0.0004
D73	0.0022	0.0398	0.0002	0.0027	0.0006	0.0148	**0.8008**	0.0002	0.0888	0.0010
D24	0.0025	0.0206	0.0001	0.0021	0.0014	0.0165	**0.5546**	0.0001	0.3569	0.0024
D21	0.0065	0.0000	0.0015	0.0288	0.0252	0.0077	0.0000	0.0043	0.0084	**0.9737**
D86	0.0025	0.0127	0.0222	**0.9509**	0.0119	0.0001	0.0080	0.0001	0.0052	0.0122
D57	0.0033	0.0182	**0.6268**	0.5268	0.0061	0.0000	0.0151	0.0010	0.0014	0.0095
D9	0.0029	0.0271	0.0002	0.0009	0.0003	0.0870	**0.6512**	0.0007	0.0152	0.0010
D6	0.1487	0.0002	0.0030	0.0107	**0.4769**	0.0003	0.0006	0.0039	0.0167	0.0201

Table 6.8 Hybrid output when run time images are rotated 90 degrees.

Another possible failure is the encountering of textures that have not been part of the network's training set. Five new textures to the training set were applied: D11 (Home spun wool cloth), D38 (Water), D49 (Straw Screening), D67 (Plastic Pellets) and D102 (Cane, shadow graph).

Figure 6.15 D11 Home spun wool cloth

Figure 6.16 D38 Water

Figure 6.17 D49 Straw Screening

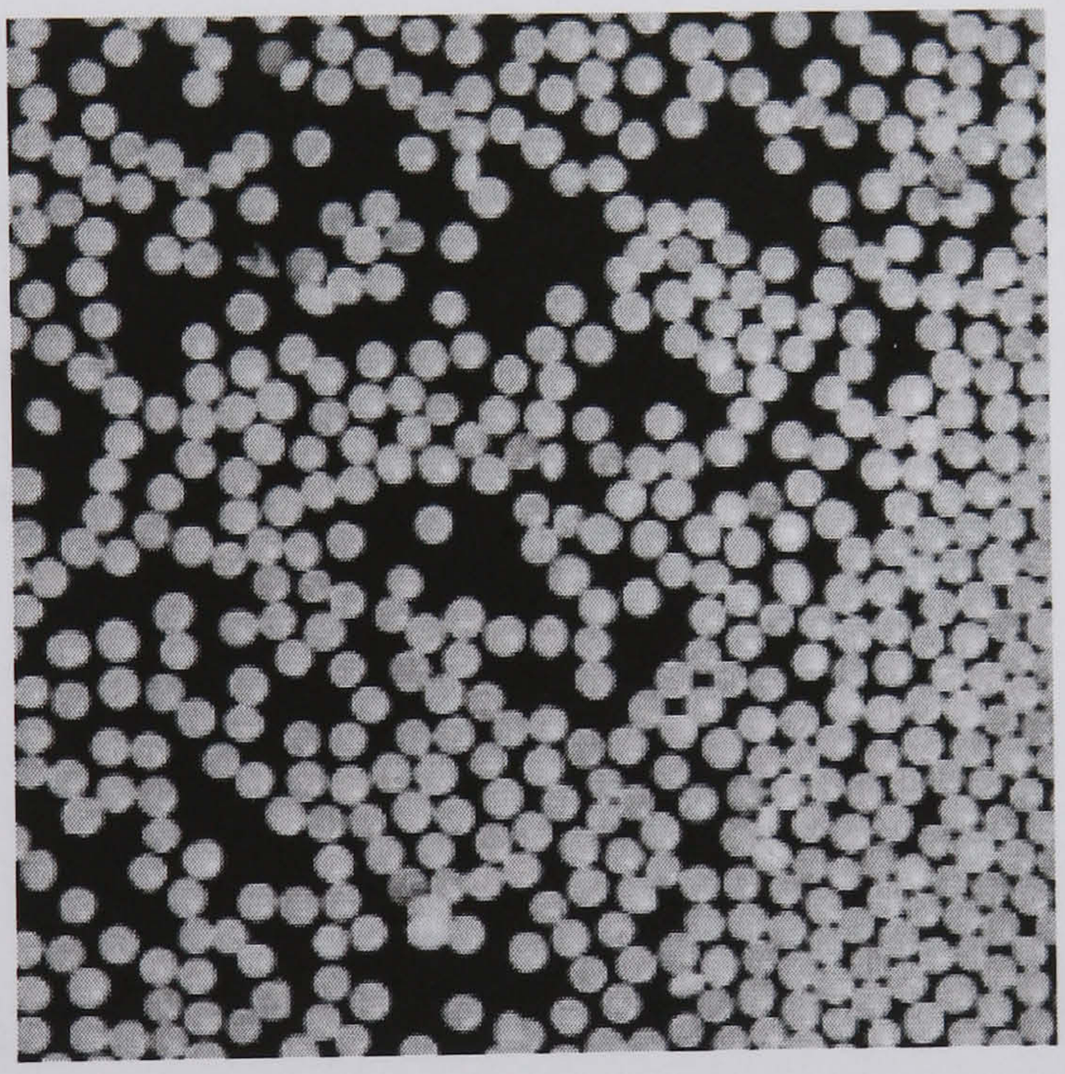


Figure 6.18 D67 Plastic Pellets

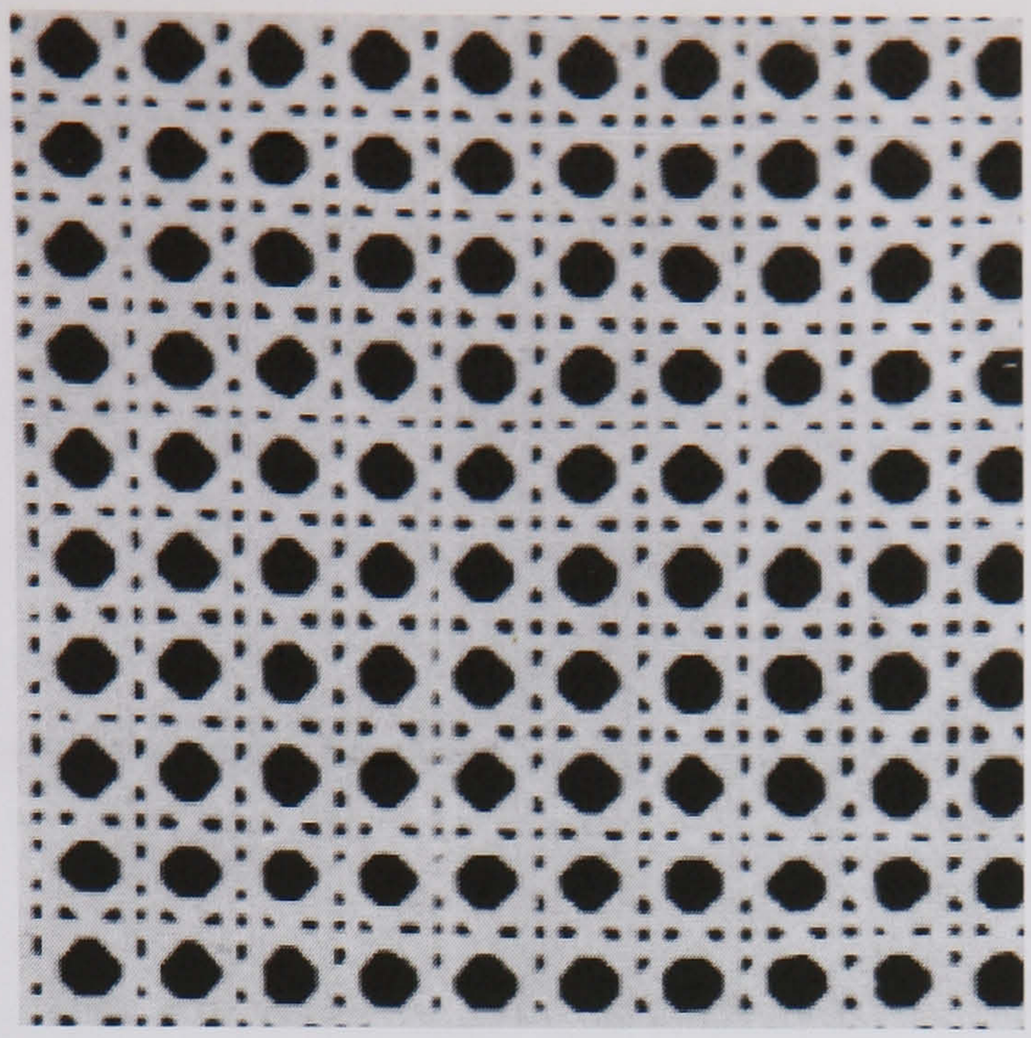


Figure 6.19 D102 Cane Shadow Graph

The network's output is recorded in Table 6.9. There are two strong outputs against unknown textures. The neurons that gave the high outputs were responsible for classifying textures with similar grey scales in them. Neuron nine classified D93 (Fur) originally a dark image with little contrast, whereas neuron five classified D21 (French Canvas) a high contrast image.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D11	0.0014	0.0335	0.0001	**0.3862**	0.0800	0.0000	0.0996	0.0001	0.3388	0.0119
D38	0.0074	0.0001	0.0000	0.0071	0.0117	0.0932	0.0323	0.0002	**0.8334**	0.0121
D49	0.0073	**0.2147**	0.0049	0.0038	0.0001	0.0238	0.0207	0.0003	0.0339	0.0018
D67	0.0010	0.0194	0.0048	0.0812	**0.0085**	0.0000	0.0007	0.0004	0.0008	0.0045
D102	0.0497	0.0015	0.0002	0.0204	**0.6290**	0.0000	0.0000	0.0009	0.0014	0.0005

Table 6.9 System Peformance for Unknown Textures

6.3 Variations of the Hybrid Neural Network Architecture

Some variations of the hybrid neural network architecture presented here to explore the resolving power of the two-dimensional network.

6.3.1 Flat Self-Organising Network Input Layer

The input layer of the self-organising network was made to be one-dimensional, the ability of the architecture to classify the textures is shown in Table 6.10 and Figure 6.20. The architecture is similar to that as presented before but the input layer to the self-organising neural network is flat and is nine neurons wide. This is the smallest dimension that results in all the images in the training set being classified by the back-propagation neural network.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0006	0.0184	0.0010	0.0057	0.0000	0.0268	0.0001	0.0192	0.0002	**0.9655**
D93	0.0004	0.0001	0.1353	0.0170	0.0076	0.0002	0.0109	0.0114	**0.5533**	0.0004
D4	0.0052	0.0000	0.014	0.0024	0.0000	0.0001	0.0262	**0.9013**	0.0556	0.0044
D73	0.0146	0.0000	0.0019	0.0142	0.0000	0.0128	**0.9028**	0.2154	0.0319	0.0002
D24	0.0112	0.0062	0.0000	0.0007	0.0000	**0.9235**	0.0156	0.0027	0.0039	0.0730
D21	0.0045	0.0287	0.0414	0.0332	**0.8140**	0.0005	0.0014	0.0005	0.0104	0.0007
D86	0.0001	0.0607	0.0473	**0.8693**	0.0161	0.0004	0.0039	0.0005	0.0013	0.0130
D57	0.0003	0.0001	**0.6596**	0.0159	0.0222	0.0000	0.0162	0.0046	0.1703	0.0004
D9	0.0045	**0.9476**	0.0011	0.0476	0.0077	0.0219	0.0008	0.0001	0.0003	0.0210
D6	**0.9573**	0.0287	0.0000	0.0000	0.0182	0.0249	0.0043	0.0077	0.0007	0.0001

Table 6.10 Flat SOM Input Layer (9 Neurons Wide)

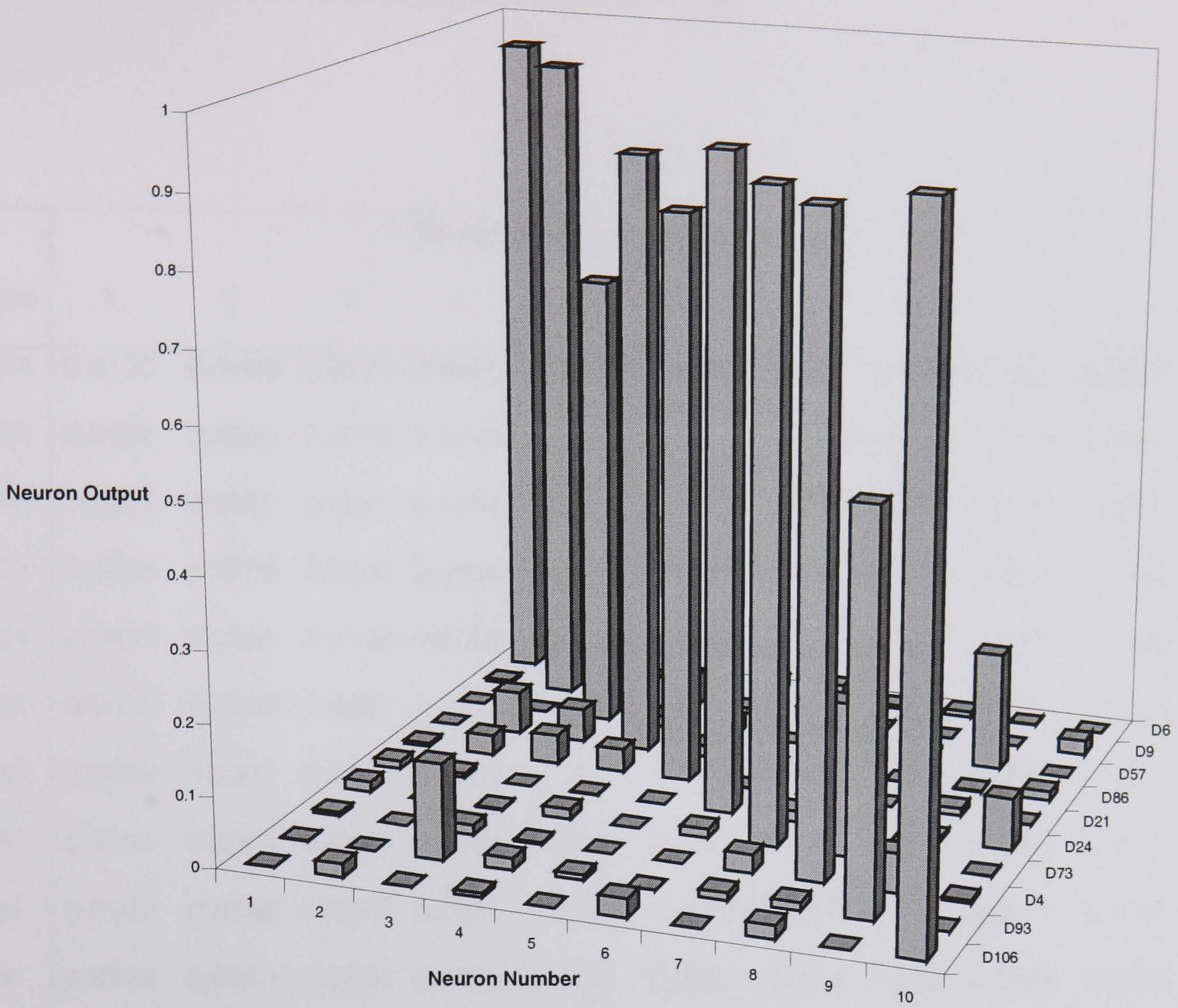


Figure 6.20 Flat SOM Input Layer (9 Neurons Wide)

The flat input layer to the self-organising map is capable of producing data that results in successful classification of all the images albeit with a lesser accuracy to that of a two dimensional network upon certain textures that contain stochastic or weakly ordered attributes.

Interestingly if the flat input layer is made wider to try and increase the performance of the system the accuracy is actually reduced. Table 6.11 shows the results when the input layer to be doubled in size to be eighteen neurons wide.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0130	0.0196	0.0173	0.0000	0.0092	0.0287	0.0001	0.0005	0.0000	**0.9638**
D93	0.0002	0.0000	0.0470	0.0098	0.0845	0.0056	0.0031	0.0221	**0.9418**	0.0000
D4	0.0031	0.0001	0.0079	0.0014	0.0000	0.0032	0.0717	**0.3152**	0.0651	0.0000
D73	0.0025	0.0750	0.0000	0.0039	0.0000	0.0084	**0.8507**	0.1751	0.0837	0.0000
D24	0.0087	0.0002	0.0126	0.0000	0.0001	**0.9526**	0.0107	0.0139	0.0001	0.0483
D21	0.0001	0.2810	0.0027	0.0095	**0.9204**	0.0000	0.0002	0.0082	0.0067	0.0265
D86	0.0032	0.0351	0.0014	**0.5768**	0.1406	0.0000	0.0051	0.0016	0.0385	0.0017
D57	0.1214	0.0000	**0.6470**	0.4560	0.0088	0.0011	0.0049	0.0023	0.0961	0.0015
D9	0.0002	**0.9886**	0.0000	0.0050	0.0241	0.0000	0.0190	0.0102	0.0001	0.0251
D6	**0.9745**	0.0000	0.7801	0.4072	0.0001	0.0036	0.0067	0.0002	0.0004	0.0392

Table 6.11 Flat SOM Input Layer (18 Neurons Wide)

Again it is the textures that contain stochastic or weakly ordered attributes that cause problems for the network. There is a general increase in background noise in the classification of the textures, as seen in Figure 6.21. The number of neurons in the input layer of the self-organising map is not the key to accurate classification, but the dimensions of the input layer is seen to be more important.

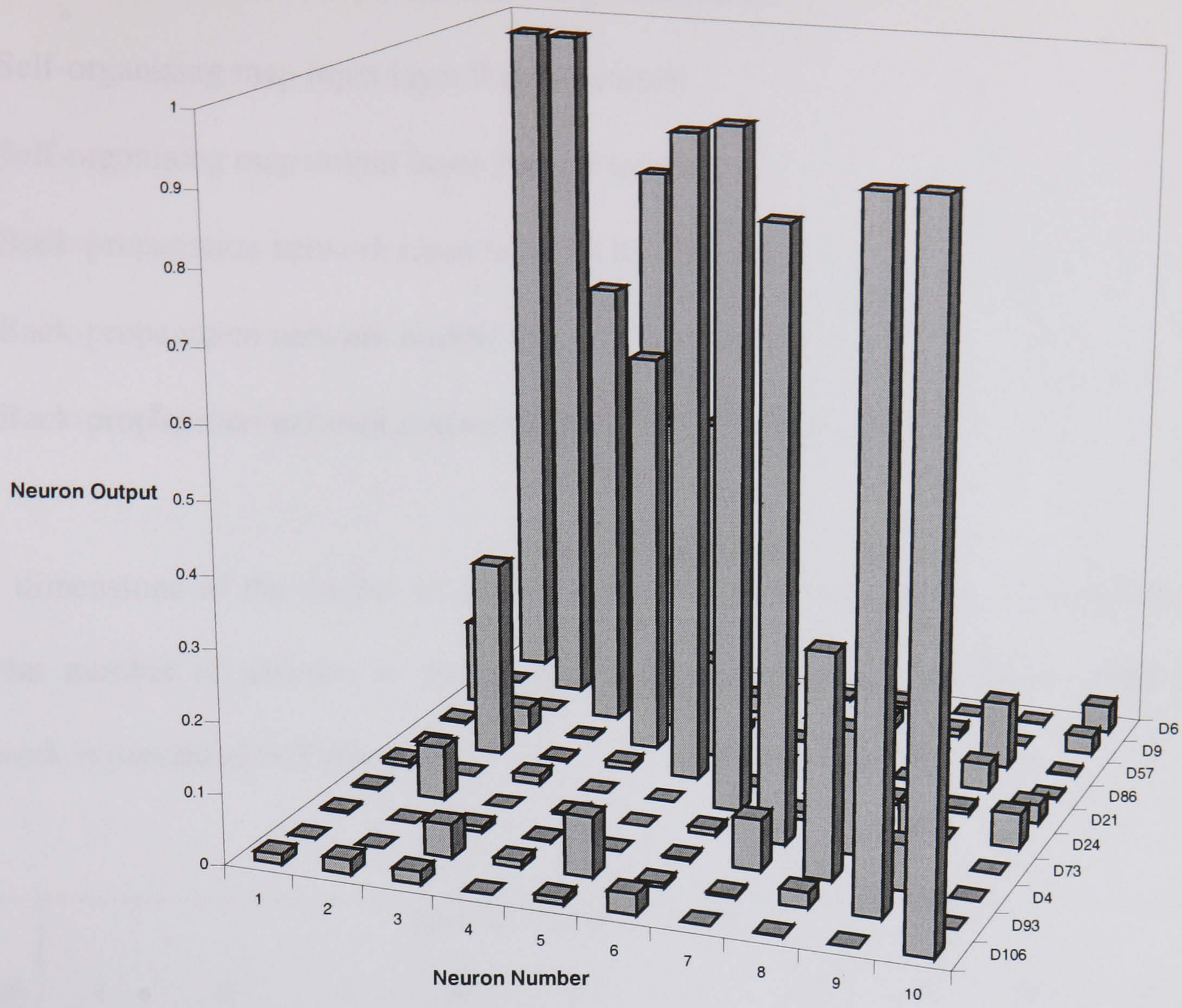


Figure 6.21 Flat SOM Input Layer (18 Neurons Wide)

6.3.2 Flat Self-Organising Network Output Layer

As well as a flat input layer, other investigations examined the effects of a flat output layer for the self-organising map. If this layer is changed then the histogram dimensions and the input layer to the self-organising map must also be changed.

The experimental network had the following dimensions:-

- Self-organising map input layer 9 by 9 neurons
- Self-organising map output layer 20 by 1 neurons
- Back-propagation network input layer 20 by 1 neurons
- Back-propagation network middle layer 3 by 3 neurons
- Back-propagation network output layer 10 by 1 neurons

The dimensions of the output layer was made to be twenty neurons wide to allow a similar number of neurons to the original architecture. The classification from this network is presented in Table 6.12.

Image	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
D106	0.0183	0.0199	0.0000	0.0015	0.0002	0.0224	0.0130	0.0156	0.0000	**0.9679**
D93	0.0092	0.0000	0.0580	0.0533	0.0001	0.0005	0.0017	0.0088	**0.7479**	0.0000
D4	0.0002	0.0000	0.0044	0.0400	0.0000	0.0047	0.0157	**0.9674**	0.0712	0.0005
D73	0.0022	0.0002	0.0002	0.0000	0.0000	0.0164	**0.9704**	0.0083	0.0027	0.0388
D24	0.0080	0.0000	0.0095	0.0000	0.0107	**0.9695**	0.0032	0.0132	0.0000	0.0214
D21	0.0083	0.1354	0.0112	0.0003	**0.9767**	0.0082	0.0015	0.0000	0.0002	0.0001
D86	0.0147	0.0068	0.0258	**0.1706**	0.0448	0.0004	0.0003	0.0005	0.0081	0.0001
D57	0.0064	0.0000	**0.5563**	0.1113	0.0008	0.0021	0.0009	0.0088	0.1835	0.0000
D9	0.0068	**0.9615**	0.0000	0.0493	0.0114	0.0000	0.0219	0.0002	0.0016	0.0388
D6	**0.9709**	0.0004	0.0007	0.0030	0.0145	0.0031	0.0032	0.0000	0.0076	0.0157

Table 6.12 Flat SOM Output Layer (20 Neurons wide)

Increasing or decreasing this number had negligible performance on the network until very low values are used such as ten neurons. At this point the system fails to classify all the textures staring with the stochastic or weakly ordered textures.

6.3.3 Adjustments to the Self-Organising Maps Training Process

Apart from the box distance measure between neurons employed in section 4.4.1, other distance metrics can be employed such as Euclidean and City Block distance measures.

The Euclidean distance metric D calculates the distance between two neurons (x_1, y_1) and (x_2, y_2) from equation (6.3.3.1).

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{6.3.3.1}$$

Figure 6.22 shows several distances to neighbouring neurons calculated from the equation above assuming the centre neuron is the winning neuron.

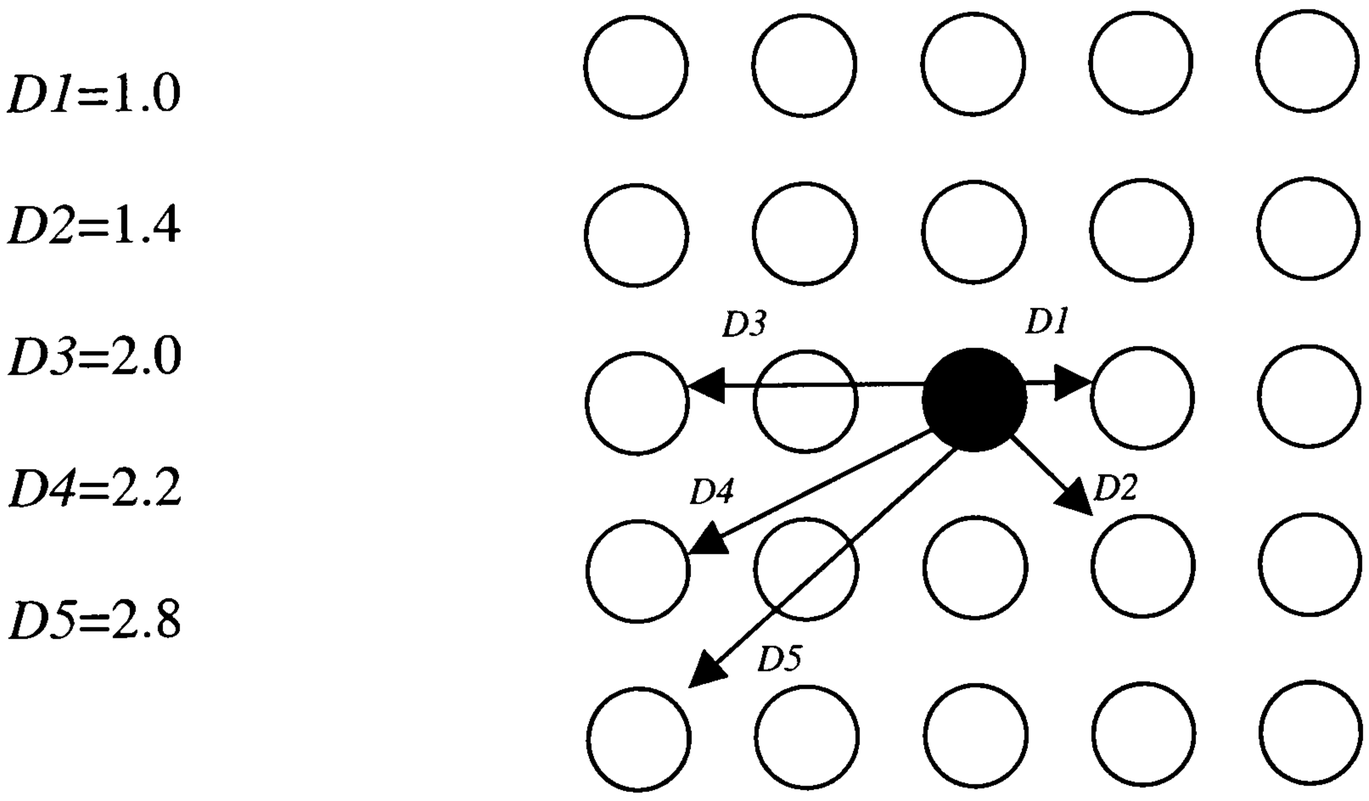


Figure 6.22 Euclidean Neighbourhood

The City Block distance metric provides a computationally very simple distance measure. This metric assumes that is only possible to 'travel' between neurons along grid lines and diagonal moves are not allowed. With only straight lines used this measure sometimes also called the Manhattan or Taxi Driver metric. Figure 6.23 shows paths taken by this metric to three neurons, again assuming the centre neuron to be the winner. Only one step is need to get to neuron *A*, however two steps via an intermediate neuron are required to get to neuron *B*. Neuron *C* lies four steps away.

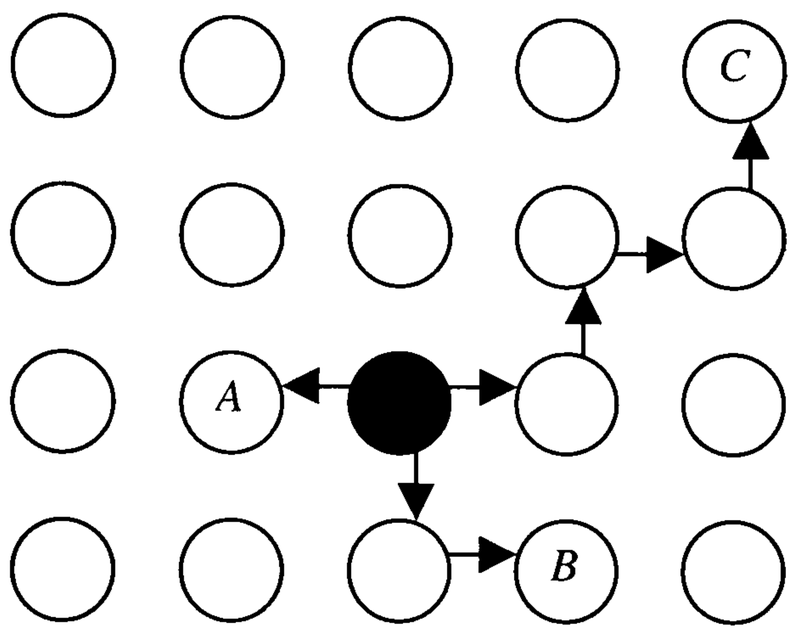


Figure 6.23 City Block Neighbourhood

The results presented in the following ten tables show the performance of the three distance metrics Box, Euclidean and City Block when operating upon the ten Brodatz images from section 6.2. Each distance measure was used in both training and run time operation with the same training parameters identified in section 6.2.2. The highest outputs are highlighted in bold text to indicate the best performing metric for the particular texture being sampled.

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0026	0.0000	0.0197	0.0163	0.0024	0.0178	0.0000	0.0000	0.0767	0.9656
Euclid.	0.0105	0.0103	0.0004	0.0089	0.0030	0.0069	0.0051	0.0087	0.0000	0.9781
City B.	0.0083	0.0104	0.0002	0.0000	0.0056	0.0081	0.0081	0.0014	0.0001	**0.9791**

Table 6.13 SOM metric comparison for D106

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0332	0.0012	0.0094	0.0769	0.0009	0.0006	0.0222	0.0000	0.9704	0.0013
Euclid.	0.0001	0.0000	0.0178	0.0631	0.0013	0.0004	0.0032	0.0038	**0.9707**	0.0002
City B.	0.0002	0.0001	0.0093	0.0686	0.0011	0.0002	0.0055	0.0027	0.7417	0.0013

Table 6.14 SOM metric comparison for D93

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0000	0.3433	0.0953	0.0003	0.0201	0.0022	0.0290	0.9307	0.0000	0.0010
Euclid.	0.0013	0.0000	0.0649	0.0032	0.0000	0.0127	0.0080	**0.9855**	0.0265	0.0126
City B.	0.0016	0.0000	0.0197	0.0001	0.0000	0.0051	0.0063	0.9671	0.0170	0.0020

Table 6.15 SOM metric comparison for D4

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0009	0.0220	0.0000	0.0002	0.0045	0.0931	0.9092	0.0027	0.0217	0.0004
Euclid.	0.0082	0.0131	0.0000	0.0000	0.0000	0.0130	**0.9881**	0.0099	0.0093	0.0100
City B.	0.0002	0.0002	0.0002	0.0058	0.0001	0.0130	0.9788	0.0149	0.0087	0.0158

Table 6.16 SOM metric comparison for D73

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0026	0.0017	0.0000	0.0001	0.0017	0.9645	0.3930	0.0097	0.0069	0.0152
Euclid.	0.0058	0.0116	0.0041	0.0000	0.0002	0.9748	0.0077	0.0125	0.0002	0.0050
City B.	0.0104	0.0060	0.0081	0.0000	0.0001	**0.9751**	0.0032	0.0100	0.0000	0.0063

Table 6.17 SOM metric comparison for D24

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0170	0.0012	0.0170	0.0008	**0.9724**	0.0003	0.0000	0.0034	0.0000	0.0096
Euclid.	0.0072	0.0022	0.0077	0.0381	0.9697	0.0010	0.0001	0.0000	0.0025	0.0052
City B.	0.0070	0.0024	0.0020	0.2020	0.9655	0.0032	0.0043	0.0000	0.0001	0.0070

Table 6.18 SOM metric comparison for D21

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0188	0.0184	0.0090	**0.9793**	0.0019	0.0000	0.0000	0.0000	0.0276	0.0266
Euclid.	0.0018	0.0183	0.0001	0.7167	0.0519	0.0001	0.0002	0.0000	0.0041	0.0046
City B.	0.0002	0.0176	0.0001	0.9606	0.0315	0.0020	0.0232	0.0001	0.0190	0.0003

Table 6.19 SOM metric comparison for D86

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0000	0.1991	0.9365	0.0069	0.0391	0.0000	0.0001	0.0328	0.0000	0.0074
Euclid.	0.0067	0.0000	0.8719	0.0859	0.0008	0.0046	0.0010	0.2104	0.5097	0.0012
City B.	0.0020	0.0000	**0.9537**	0.0006	0.0027	0.0017	0.0002	0.0489	0.3249	0.0046

Table 6.20 SOM metric comparison for D57

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.0000	0.9224	0.0052	0.0051	0.0058	0.0002	0.0552	0.0162	0.0000	0.0000
Euclid.	0.0081	0.9623	0.0000	0.0436	0.0041	0.0114	0.0782	0.0000	0.0029	0.0378
City B.	0.0121	**0.9757**	0.0000	0.0279	0.0016	0.0210	0.0086	0.0001	0.0008	0.0122

Table 6.21 SOM metric comparison for D9

Metric	Neuron Number (1-10)									
	1	2	3	4	5	6	7	8	9	10
Box	0.9679	0.0000	0.0013	0.0088	0.0183	0.0039	0.0001	0.0016	0.0218	0.0106
Euclid.	**0.9903**	0.1701	0.0008	0.0032	0.0049	0.0196	0.0041	0.0005	0.0000	0.0064
City B.	0.9692	0.1275	0.0001	0.0000	0.0027	0.0061	0.0001	0.0034	0.0000	0.0044

Table 6.22 SOM metric comparison for D6

All three metrics provide accurate data to the back–propagation classifier section to enable one hundred percent classification of the ten Brodatz textures. In this application the choice of distance metric does not impact significantly on the accuracy of the system.

6.4 Real World Images

6.4.1 Training Images

To evaluate the hybrid neural network architecture against real world problems, some images from the aerial photographic survey [USB 1994] of the campus and surrounding neighbourhood of University of Santa Barbara were used to construct a data set for testing purposes. Again all images are histogram equalised.

When applying the hybrid network to real world images, certain flaws in the training philosophy became immediately apparent that were not evident when using the Brodatz data set. The Brodatz images are essentially artificial textures as they have moderately the same characteristics across the image.

Consider the aerial image Figure 6.24 explored earlier in Section 3.2. Again if the segmentation of the trees is desired, then a single example of a tree would not suffice. Closer examination of the trees within the image shows a wide variation of foliage density and a broader range of grey scales used.

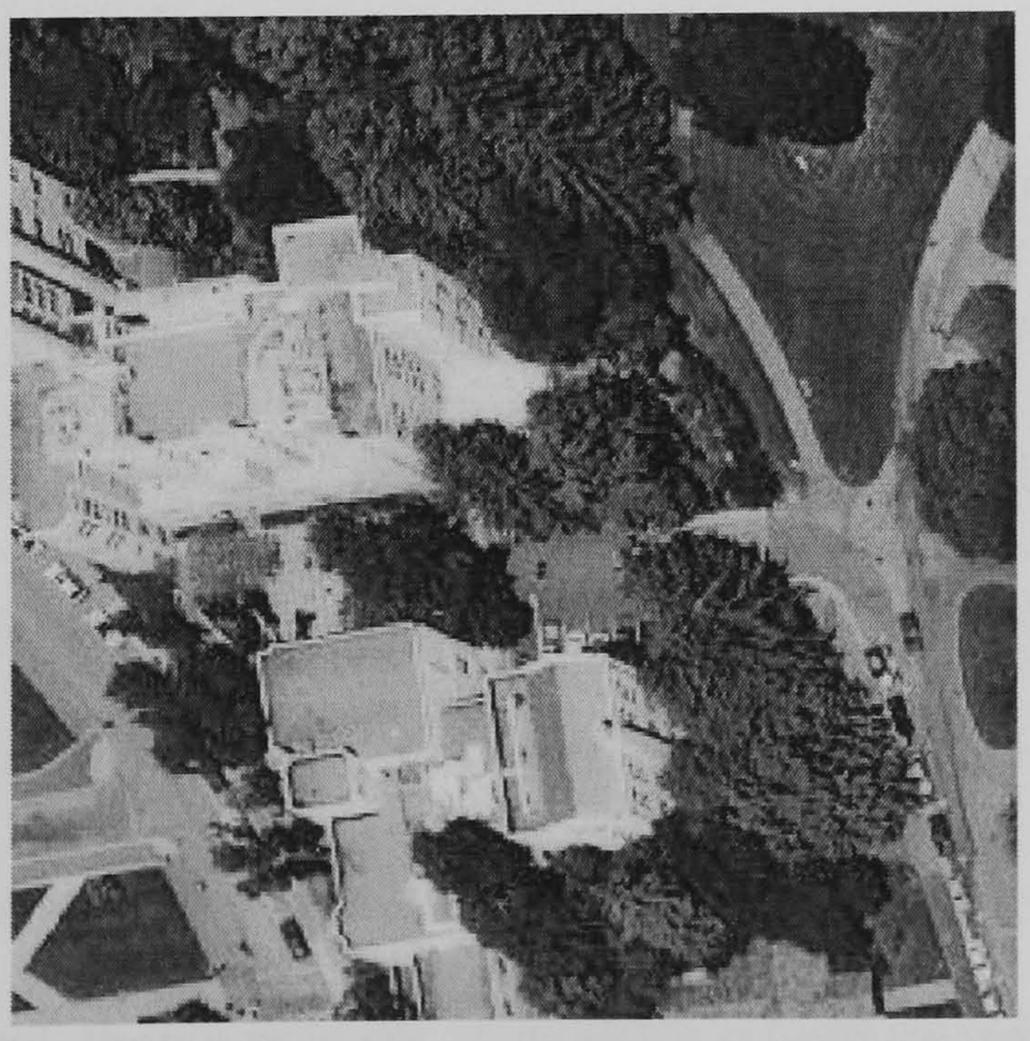


Figure 6.24 Aerial Image 1

When considering the other elements within the image such as roads and buildings, there can not be a generic sample for each type of texture as used in the Brodatz experiments.

The training set had to be expanded to encompass a broad range of textures that would have to be processed when segmenting desired features out of aerial images. Again the image in the training set would be sampled from different images to those used in the testing of the hybrid architecture.

Four labels were defined to the samples created:-

- **Tree**: these samples would contain a wide variety of trees and bushes.
- **Road**: wide range of samples containing anything automotive such as cars, highways, freeways and car parks.
- **Building**: from residential to commercial properties.
- **Grass**: covers sports fields, gardens and wasteland.

6.4.2 Hybrid Architecture and Training

The architecture and operation of the hybrid architecture were also changed to optimise the performance of the system. Altering the output layer of the back-propagation neural network to contain just two neurons, changed the system to act as a binary classifier, that is it would be trained to look for just one type of texture. The interconnecting weights of the neurons would now be developed for specialist texture classification. That is a set of weights would be created for each texture to be classified. If a different texture is to be processed, then a new set of weights would be loaded and used.

When sampling an image for a texture, neuron 1 would fire if the target texture has been identified, otherwise neuron 2 would fire indicating an unwanted texture has been identified. No threshold boundary is applied to the neurons' output, the neuron with the highest output indicates the detected texture.

Another change to the operation of the hybrid network was identified when segmenting an image rather than just classifying it. Samples would be taken in small blocks with sizes ranging from the size of the self-organising map's input layer, up to larger samples for histogram accuracy. Therefore the resultant segmented image becomes very blocky.

This is demonstrated in Figure 6.25 by using the following network architecture and training conditions which demonstrated the best results from Section 6.2.2.

- Self-organising map input layer 9 by 9 neurons
- Self-organising map output layer 5 by 5 neurons
- Back-propagation network input layer 5 by 5 neurons
- Back-propagation network middle layer 3 by 3 neurons
- Back-propagation network output layer 2 by 1 neurons
- SOM Learning rate 0.01 for 3 cycles* starting with a neighbourhood of 2 pixels.
- BPNN Learning rate of 0.1 for 50000 samples, momentum term of 0.5

*A SOM cycle is deemed to be a complete pass over every image in the training set with the entire coverage of every image in the set. At the end of a cycle the neighbourhood term is decremented.

The training set consisted of sixty images, thirty images for the desired feature and the remaining thirty comprised of the three unused labels. The self-organising stage processed the images in an order of desired texture then an undesired texture followed again by desired textures until all the images have processed. The sample mask is moved across by one pixel and the whole processed is repeated. The intermediate histograms are presented to the back-propagation neural network in a random order.

Figure 6.25 Aerial Image 1 Segmented by blocks of 15 x 15 pixel samples

In the example of Figure 6.25 the trees were designated as the desired texture in the training set applied to the network. The image was sampled in square blocks of 15 pixels, each sample was processed and a decision made as to whether or not the desired texture was present. A black block indicates that the sample was not the desired texture.

The grey border around the image indicates that region could not be processed by the mask because it would sample data that was off the edge of the image. Therefore if 15 x 15 pixel mask was used, a 7 pixel buffer / indent is used around the edge of the image.

6.4.3 System Performance

To overcome the blocky effect, a sliding window Figure 6.26 was employed by means of constructing a new image of segmented data as the original image is being sampled.

The image is sampled in blocks of pixels. Within each sample a mask feeding the input layer of the self-organising map was run over the entire sample generating the intermediate histograms. After each sample is processed the back-propagation section provides an indication as whether or not the desired texture has been sampled. The centre pixel of the sample region is then either masked out or left remaining at its original grey scale value depending upon the output of the hybrid network. The sample region is incremented by one pixel and the process repeated until the entire image has been covered.

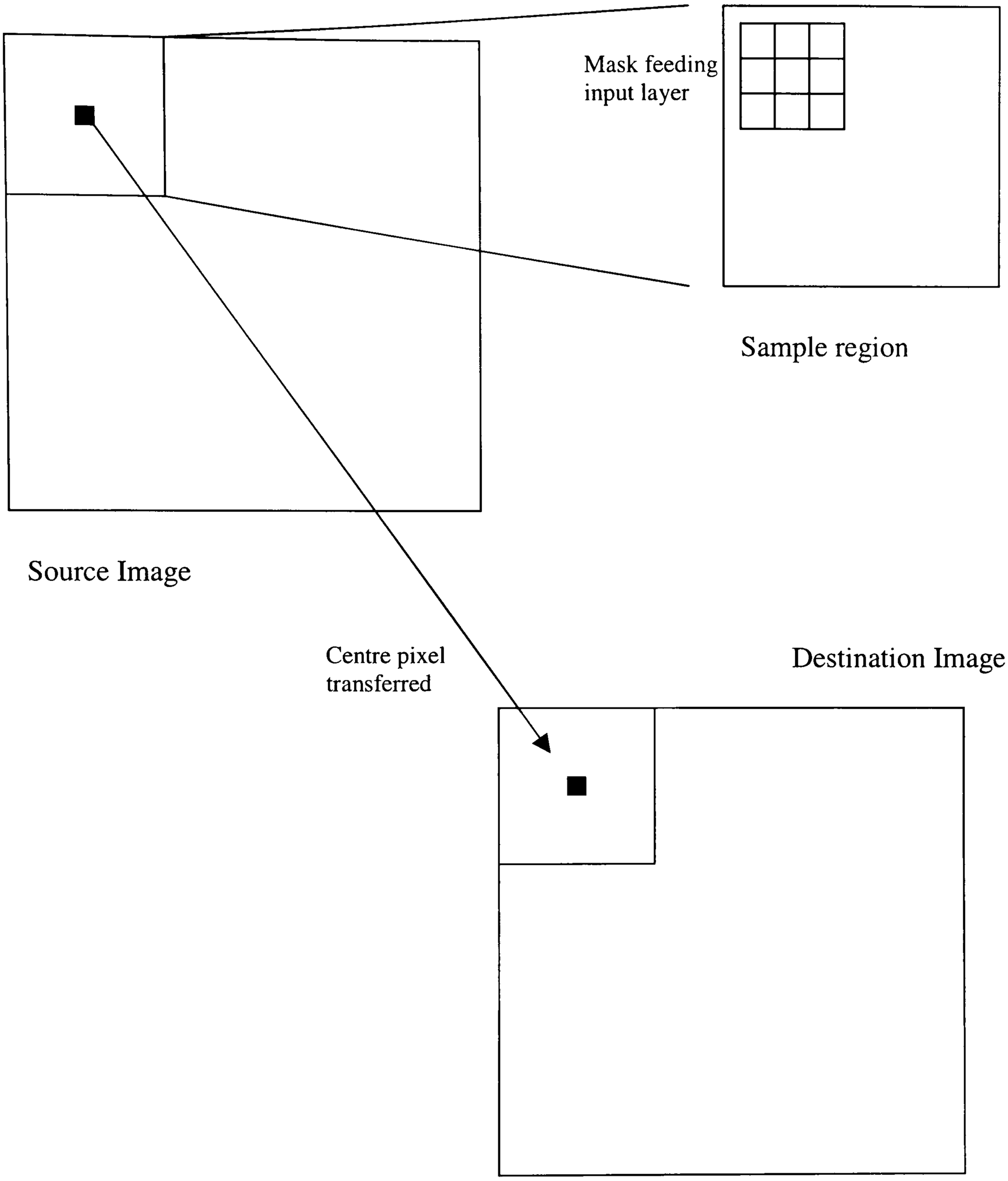


Figure 6.26 Sliding window sampling

Figure 6.27 demonstrates a sliding window upon Aerial Image 1 from Figure 6.24 using a sample size of 20 x 20 pixel blocks.

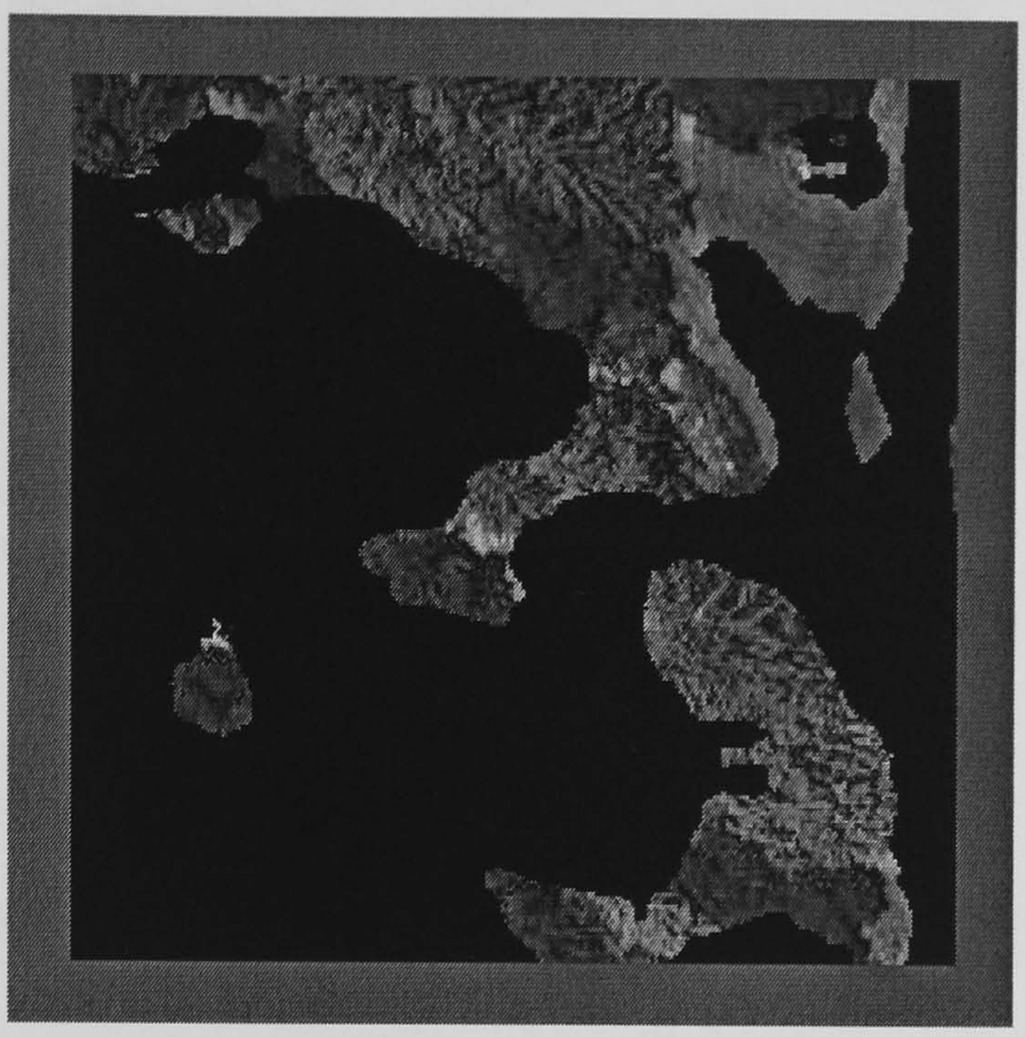


Figure 6.27 Aerial Image 1 segmented by a sliding window in 20 x 20 pixel square samples.

The size of sample taken when performing this process is critical to the accuracy achieved. This is demonstrated when the performing a segmentation process on Figure 6.28, in this case the hybrid network was trained to classify roads and cars.

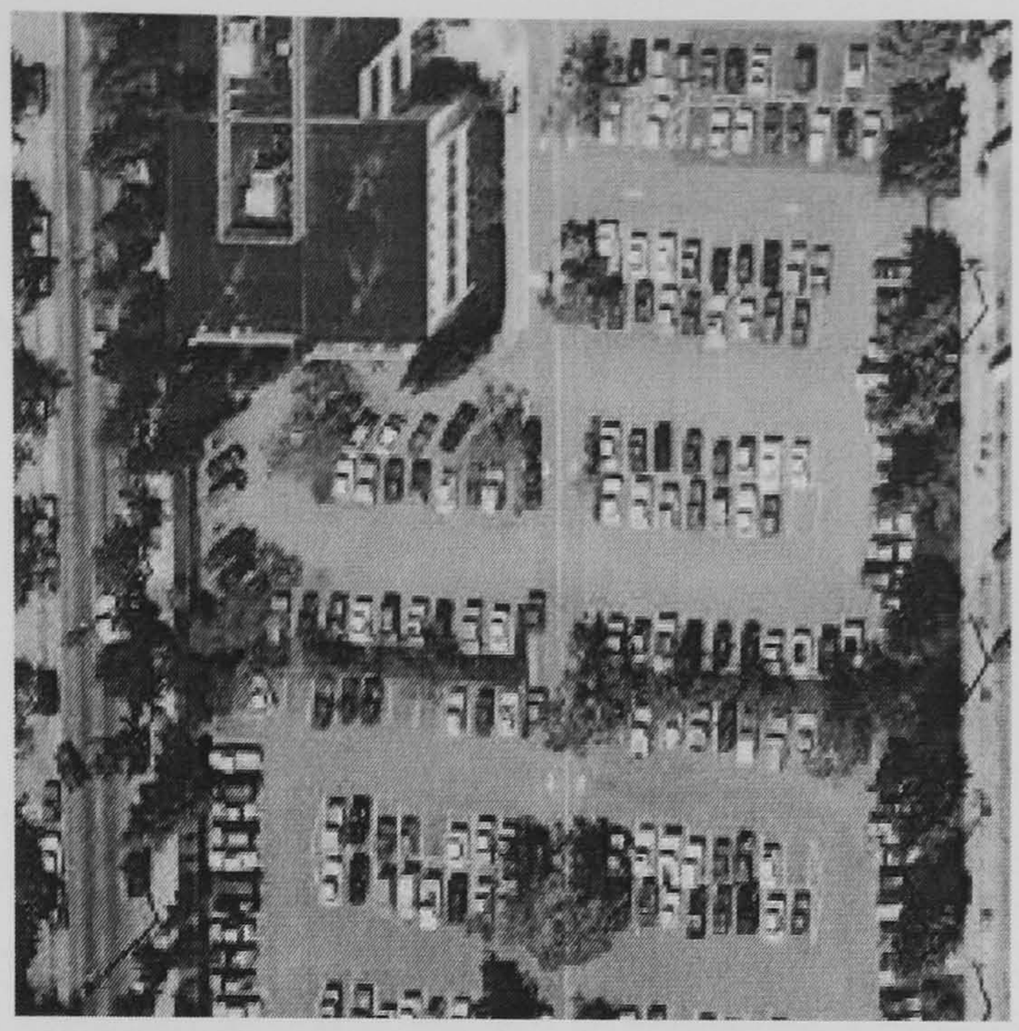


Figure 6.28 Aerial Image 2, Car Park

The first sample size was made to be nine by nine pixels which is the same size as the input layer mask. This size of sample does not allow the intermediate histogram to be built up, it therefore only represents a local region of the image and cannot give an accurate representation of the most active neurons. Figure 6.29 shows that the system has produced results that have isolated the cars and road regions albeit with a large amount of corruption.

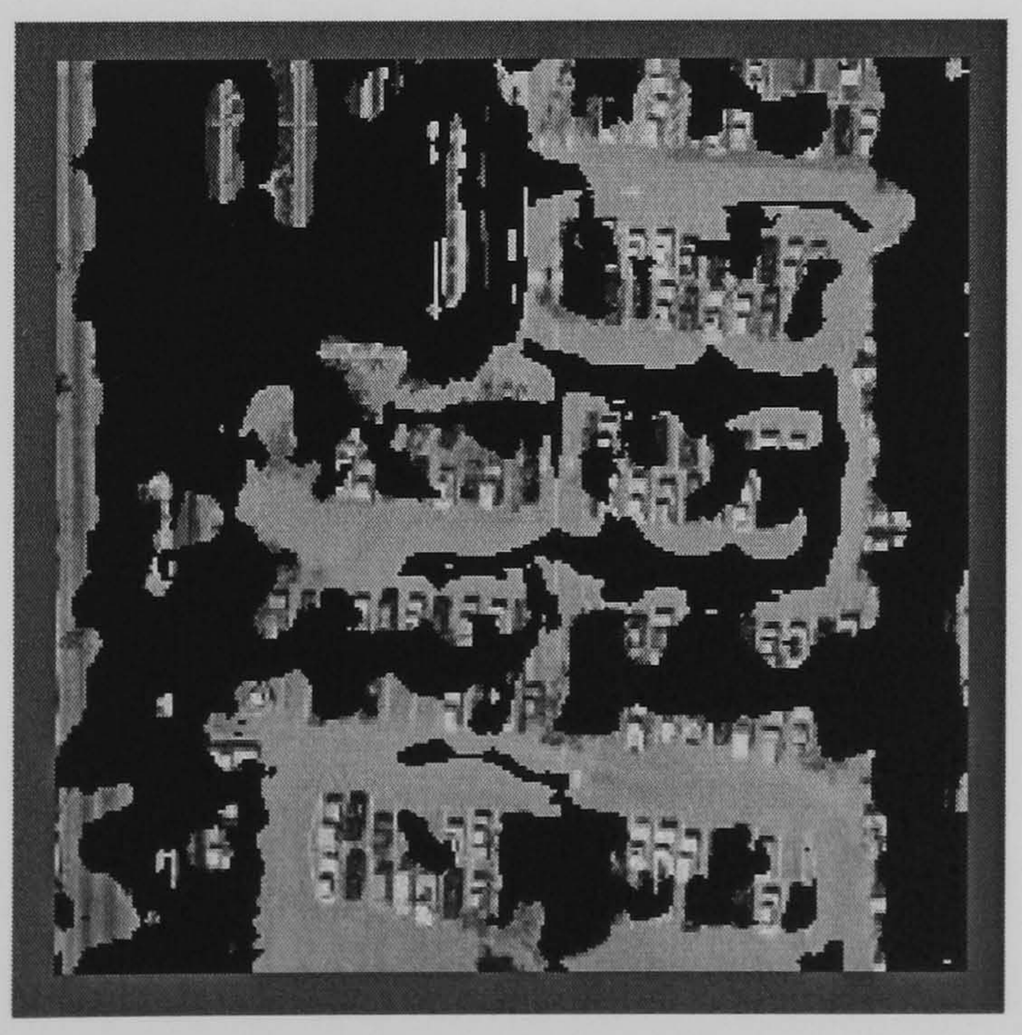


Figure 6.29 Aerial Image 2, Car Park, 9 x 9 pixel sample

Increasing the sample size to a 15 x 15 pixel squares gives better results, although there are still regions that have not been segmented correctly, Figure 6.30.

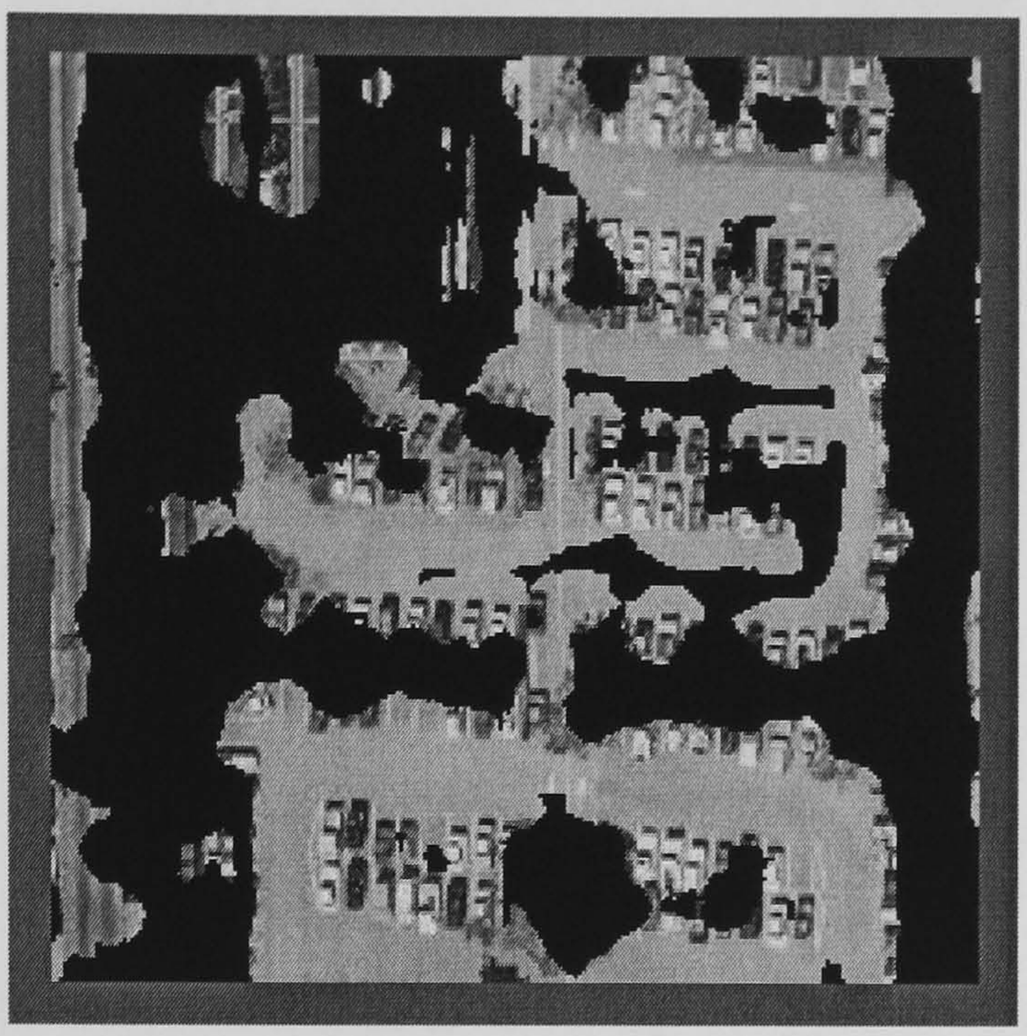


Figure 6.30 Aerial Image 2, Car Park, 15 x 15 pixel sample

With a sample size of 21 x 21 pixels, Figure 6.31 shows that the error is becoming acceptable in terms of the segmentation highlighting roads and cars.

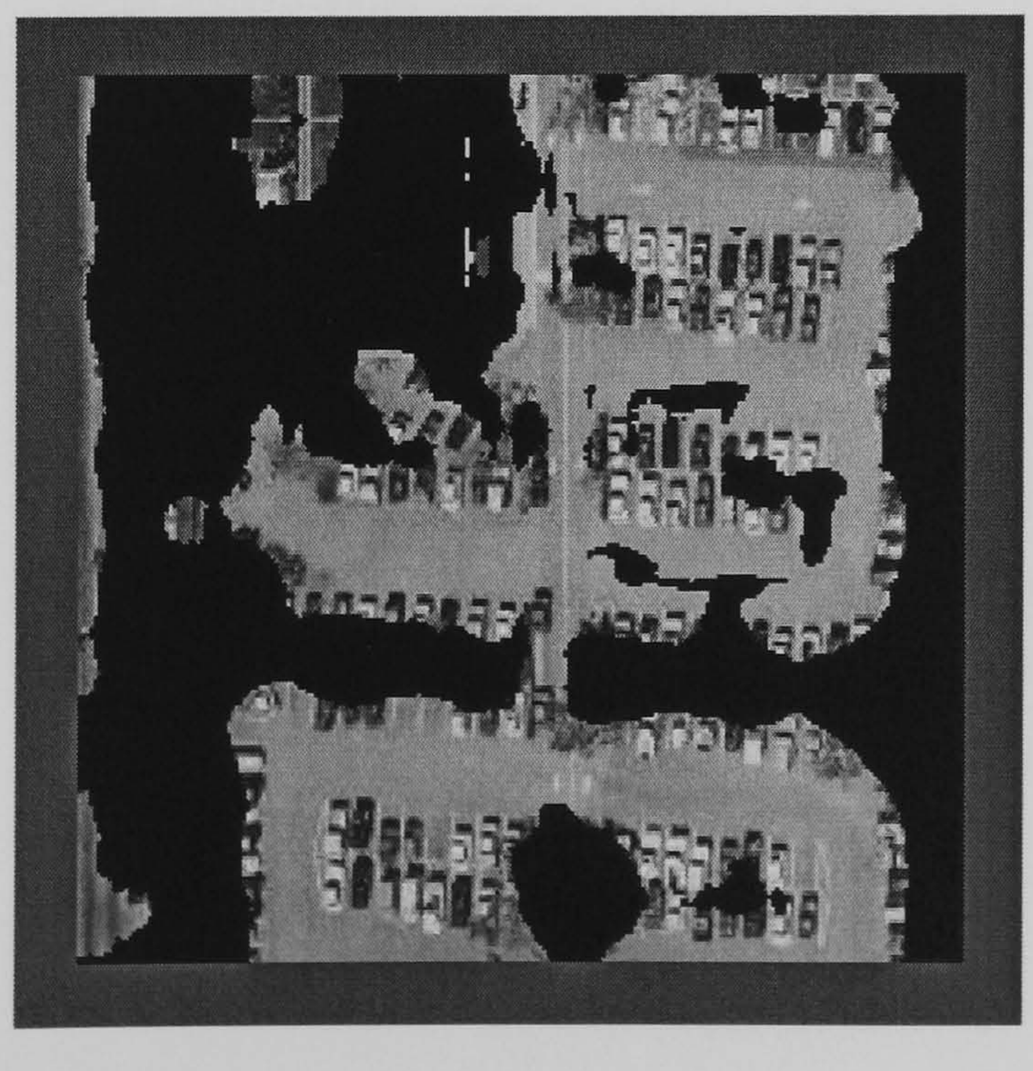


Figure 6.31 Aerial Image 2, Car Park, 21 x 21 pixel sample

With a sample size of 30 x 30 pixels, accurate results are achieved with very little spurious triggering, Figure 6.32.

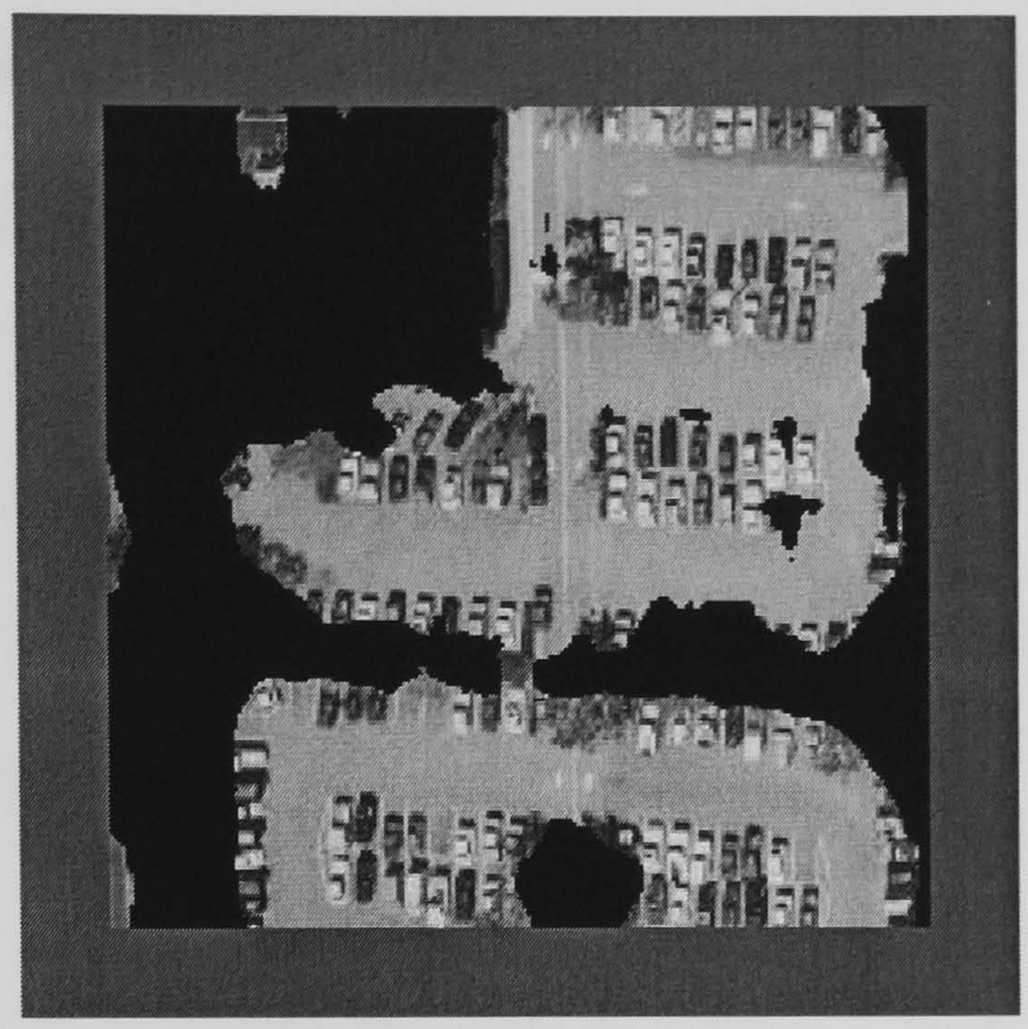


Figure 6.32 Aerial Image 2, Car Park, 30 x 30 pixel sample

The 30 x 30 pixel square mask was judged to offer the best accuracy against system performance. With a sample size of 40 x 40 (Figure 6.33), the negligible increase in accuracy does not warrant the extra burden upon the processor.

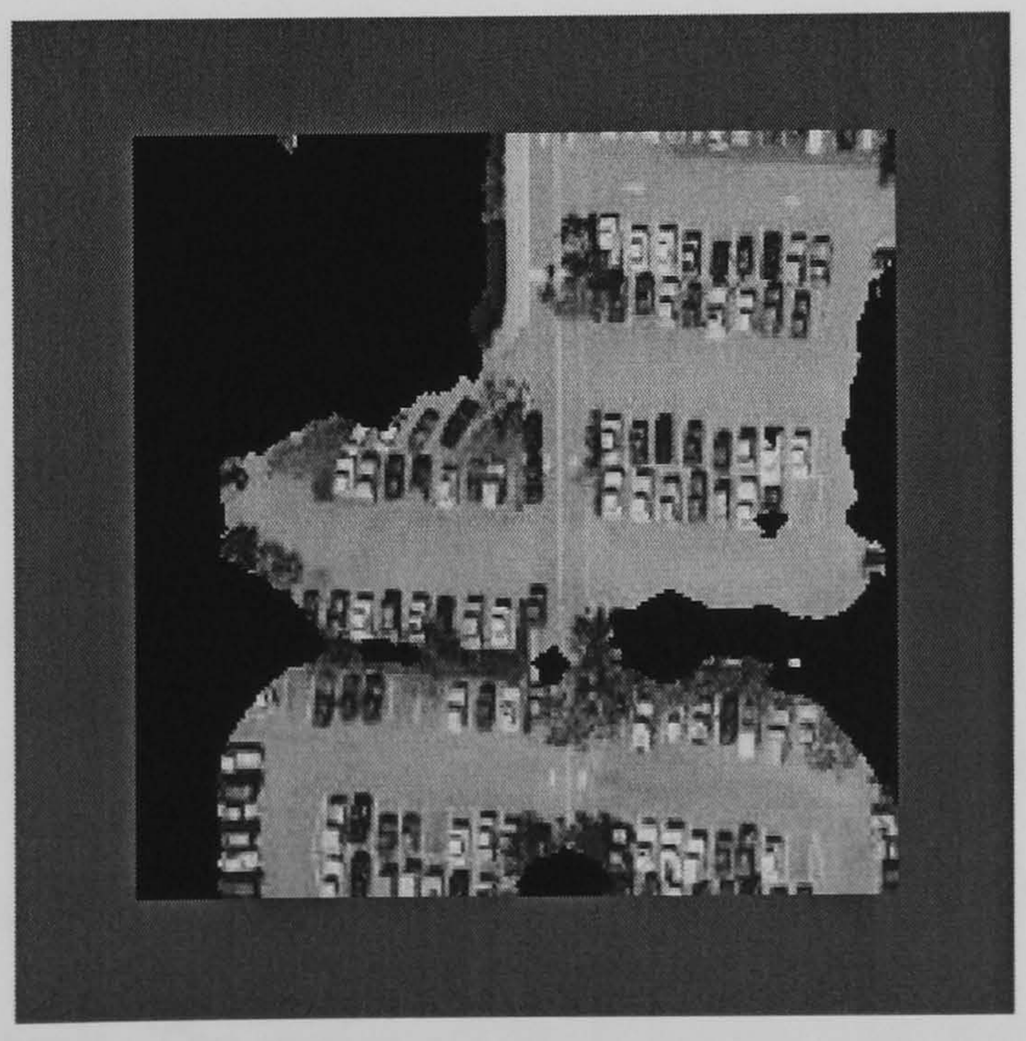


Figure 6.33 Aerial Image 2, Car Park, 40 x 40 pixel sample

6.4.4 Real World Image Results

To give an indication of how the hybrid neural network architecture performance relates to other machine segmentation techniques a benchmark method is introduced. It offers the feature generation properties of the spatial grey level dependence matrices and the classification power of the back-propagation neural network as proposed by Muhamad and Deravi [Muhamad and Deravi 1993]. Appendix C gives an overview of the operation of this benchmark process in relation to image segmentation for the results presented in this section.

To evaluate the performance of the hybrid neural network and benchmark systems when processing real world images, test images have to be created where they are segmented by hand via a computer graphics package. The assumption being made here is that a human can perform the segmentation process to 100% accuracy. These test images can be used to generate an error term against the output of the both systems. This is achieved by overlaying the hand segmented image on top of the output of the system under test and each pixel is cross checked between the two images. From this comparison the number of correctly classified pixels can be summed and an average calculated for that particular image.

Sixty aerial images are presented to the two systems with thirty images being nominated to have trees as their desired texture segmentation output and thirty having automotive elements to be segmented out. Table 6.23 overleaf shows the results gathered from both systems when segmenting for automotive elements. Values in bold type indicate which system gave the best result. The entire pictorial results and original source images are presented in Appendix E.

Image Name	Hybrid Neural Network	SGLDM / BPNN Classifier
Aerial Image 3	**91.18**	89.18
Aerial Image 4	**83.03**	77.72
Aerial Image 5	**78.79**	78.17
Aerial Image 6	**84.55**	83.80
Aerial Image 7	85.58	**88.47**
Aerial Image 8	83.85	**90.63**
Aerial Image 9	**85.59**	83.08
Aerial Image 10	**79.60**	62.78
Aerial Image 11	**84.55**	77.17
Aerial Image 12	**85.89**	76.91
Aerial Image 13	**86.73**	86.09
Aerial Image 14	**84.12**	78.48
Aerial Image 15	**85.68**	80.77
Aerial Image 16	**82.73**	79.245
Aerial Image 17	**77.93**	73.54
Aerial Image 18	**82.75**	77.89
Aerial Image 19	**84.75**	82.61
Aerial Image 20	**83.05**	76.85
Aerial Image 21	**86.49**	85.85
Aerial Image 22	**95.95**	75.10
Aerial Image 23	86.00	**95.28**
Aerial Image 24	83.60	**90.90**
Aerial Image 25	**90.90**	80.83
Aerial Image 26	**85.70**	84.48
Aerial Image 27	79.89	**86.16**
Aerial Image 28	90.55	**94.84**
Aerial Image 29	**88.02**	87.46
Aerial Image 30	**83.74**	81.63
Aerial Image 31	76.17	**81.94**
Aerial Image 32	**76.24**	75.55

Table 6.23 Percentage of Correctly Classified Pixels for Automotive Elements

Table 6.23 gives the percentage number of pixels correctly classified for both the hybrid neural network and the spatial grey dependence matrix / back-propagation neural network benchmark method. This percentage is calculated with reference to the hand segmented images. The system with the most accurate output is highlighted in bold type.

Considering automotive segmentation the overall average performance of correctly classified pixels is: -

- Hybrid Neural Network 84.45%
- SGLDM / BPNN Benchmark 82.11%

The performance of both systems is close with the hybrid neural network giving a higher accuracy overall when considering all thirty images that need to be segmented for automotive elements. This can be attributed to the ability of the hybrid neural network to handle scenes that contain urban and residential content. Quite often the hybrid architecture will encounter and segment out trees whereas SGLDM / BPNN benchmark will fail and leave them in the final image thus lowering its score. Aerial images 10 and 22 (Figures E8 and E20) are prime examples of this.

However the SGLDM / BPNN benchmark does have an advantage when processing scenes with limited textures particularly with a high content of the desired texture within in them. Aerial images 23 and 24 (Figures E21 and E22) show this with a large amount of automotive elements within them. False triggering on behalf of the hybrid neural network lowers its accuracy in these scenarios.

Image Name	Hybrid Neural Network	SGLDM / BPNN Classifier
Aerial Image 33	**86.89**	81.01
Aerial Image 34	**84.95**	78.99
Aerial Image 35	**83.05**	70.16
Aerial Image 36	**83.36**	66.67
Aerial Image 37	**91.21**	87.30
Aerial Image 38	**84.03**	80.85
Aerial Image 39	**88.20**	80.31
Aerial Image 40	**83.42**	77.12
Aerial Image 41	**84.59**	81.46
Aerial Image 42	**85.01**	74.62
Aerial Image 43	**85.55**	70.67
Aerial Image 44	**87.76**	62.92
Aerial Image 45	**88.09**	67.64
Aerial Image 46	**87.44**	68.76
Aerial Image 47	**87.10**	67.35
Aerial Image 48	**87.40**	62.85
Aerial Image 49	**89.17**	82.53
Aerial Image 50	**86.88**	60.27
Aerial Image 51	**91.06**	67.32
Aerial Image 52	**90.83**	77.88
Aerial Image 53	**83.07**	65.96
Aerial Image 54	**83.15**	68.58
Aerial Image 55	**90.19**	84.66
Aerial Image 56	**88.66**	72.29
Aerial Image 57	**85.94**	68.12
Aerial Image 58	**85.17**	72.85
Aerial Image 59	**86.93**	82.06
Aerial Image 60	**83.19**	74.04
Aerial Image 61	**87.18**	75.82
Aerial Image 62	**86.80**	77.66

Table 6.24 Percentage of Correctly Classified Pixels for Tree Elements

Table 6.24 gives the percentage performance of both systems when considering segmenting tree like textures.

Considering tree segmentation, the overall average performance of correctly classified pixels is: -

- Hybrid Neural Network 86.54%
- SGLDM / BPNN Benchmark 73.62%

Surprisingly since both systems were trained on the same images the performance of the hybrid neural network is noticeably higher to that of the SGLDM / BPNN benchmark unlike the previous examples of segmenting automotive elements. The difference in performance can be attributed to the hybrid neural network often over segmenting an image by removing large amounts of detail, sometimes even some of the desired texture. Whereas the SGLDM / BPNN benchmark under segments the image by leaving the desired texture intact but also a large amount of the undesired texture. An typical example of this can be seen in aerial images 51 and 52 (Figures E49 and E50), where a large amount of data needs to be segmented out.

The SGLDM / BPNN benchmark again gives its best performance when there is little data to be segmented out of the image, with a large amount of the wanted texture existing within the image which can be seen in aerial images 33 and 37 (Figures E35 and E31).

6.4.5 System Timings

All the experiments presented in this thesis were performed upon an Intel Pentium III™ processor with a clock speed of 500 MHz running the Windows98™ operating system.

The architecture of the neural network impacts significantly upon the performance of the system in the training phase of the self-organising map. This is demonstrated in Figure 6.34, timings were recorded with different sizes of input layer, the output layer remained static with a fixed dimension of 9 by 9 neurons. The times taken were to process a sample of 32 by 32 pixels within an image.

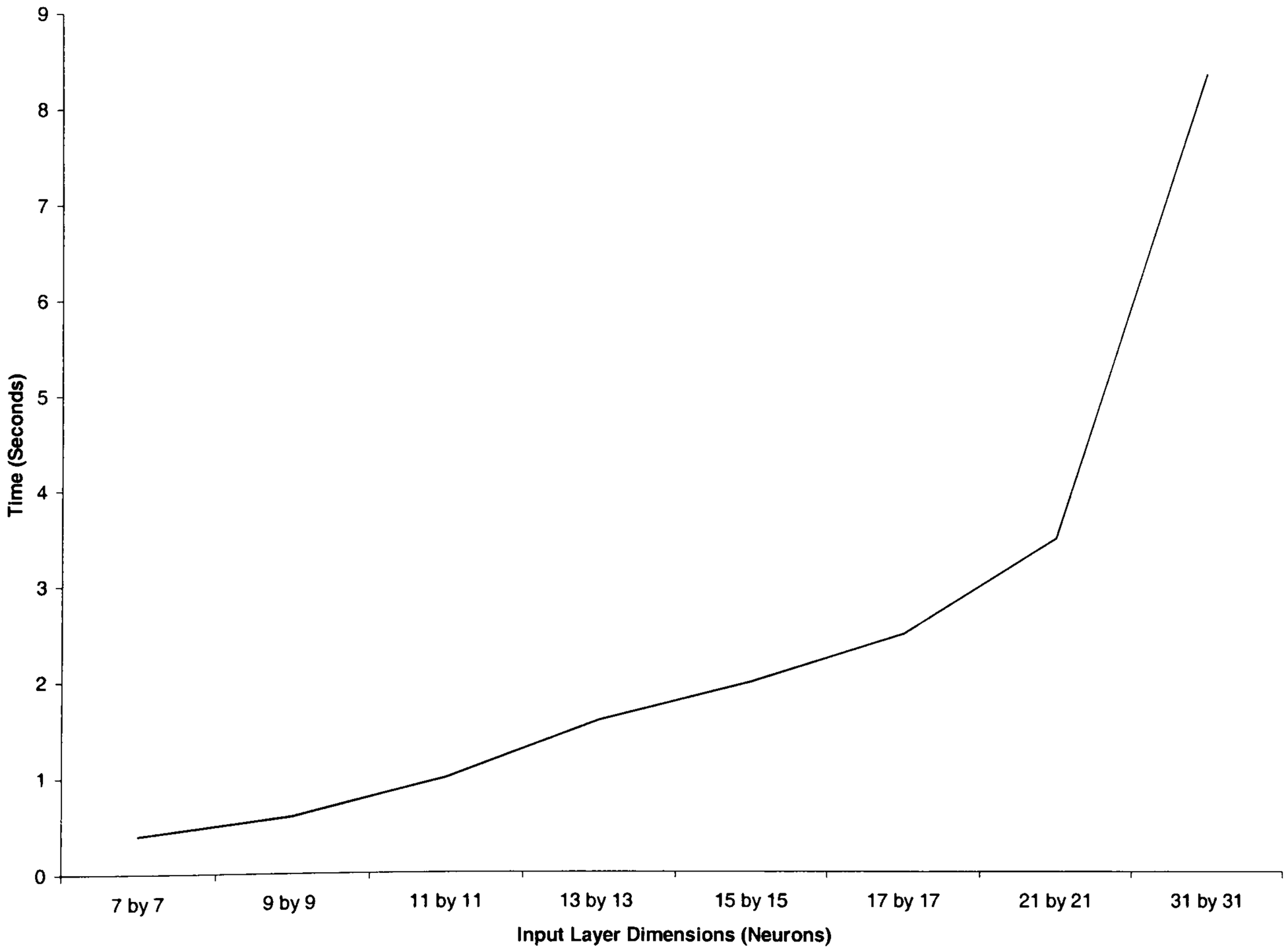


Figure 6.34 Variable input layer, fixed output layer

Again in the training phase, the process was repeated by varying the size of the output layer and keeping the input layer static with a fixed dimension of 9 by 9 pixels. Figure 6.35 shows that the dimensions of the output layer are far less critical to performance to those of the input layer. The growth in processing time is reasonably linear.

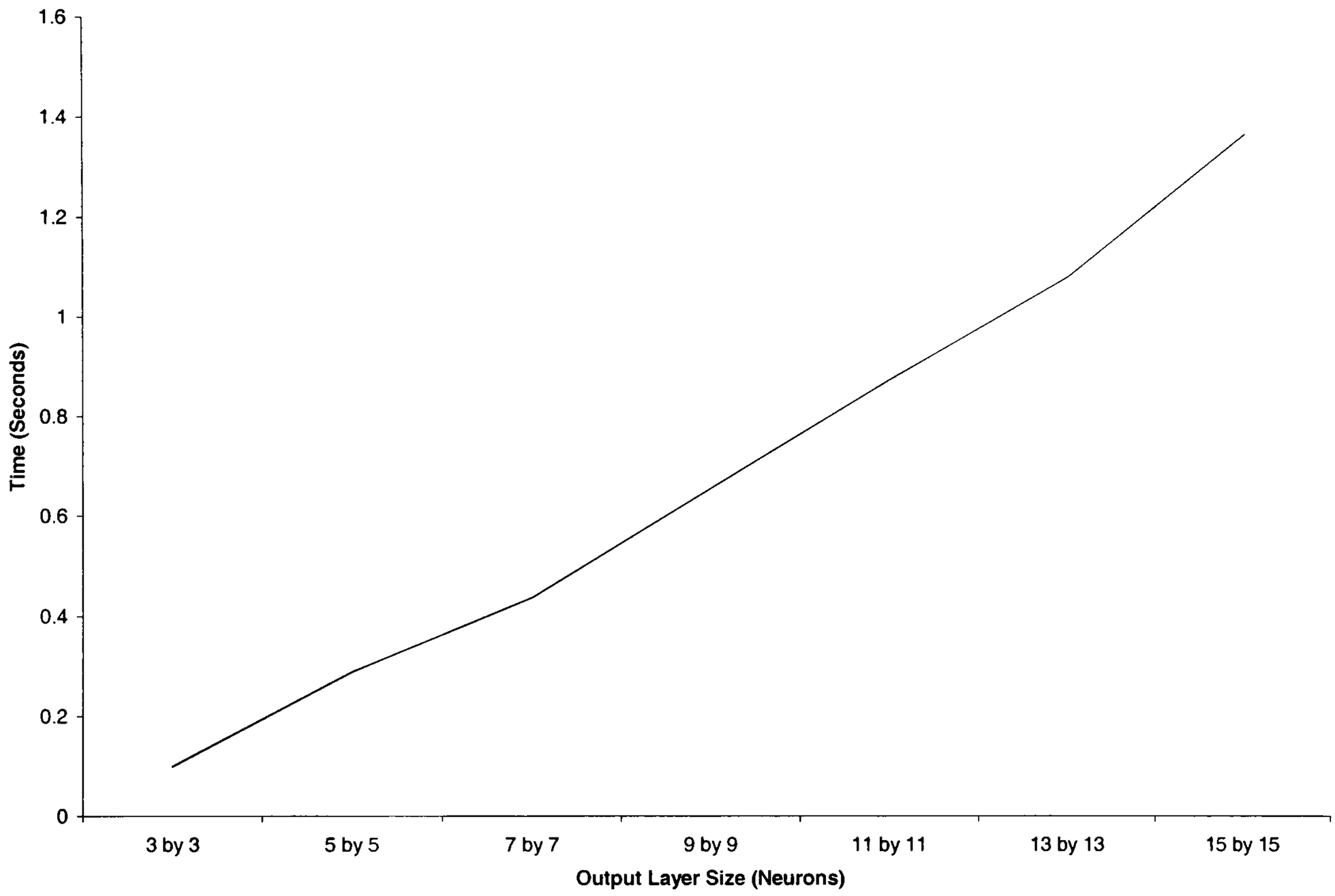


Figure 6.35 Variable output layer, fixed input layer

Another important factor for performance considerations when using the sliding window described in section 6.4.2 is the sample size used in the self-organising layer. Figure 6.36 depicts the relationship between sample size and performance. The dimensions of the network is again a 9 by 9 neuron input layer and a 5 by 5 neuron output layer for the self-organising map. Only one pass / cycle was applied to the image.

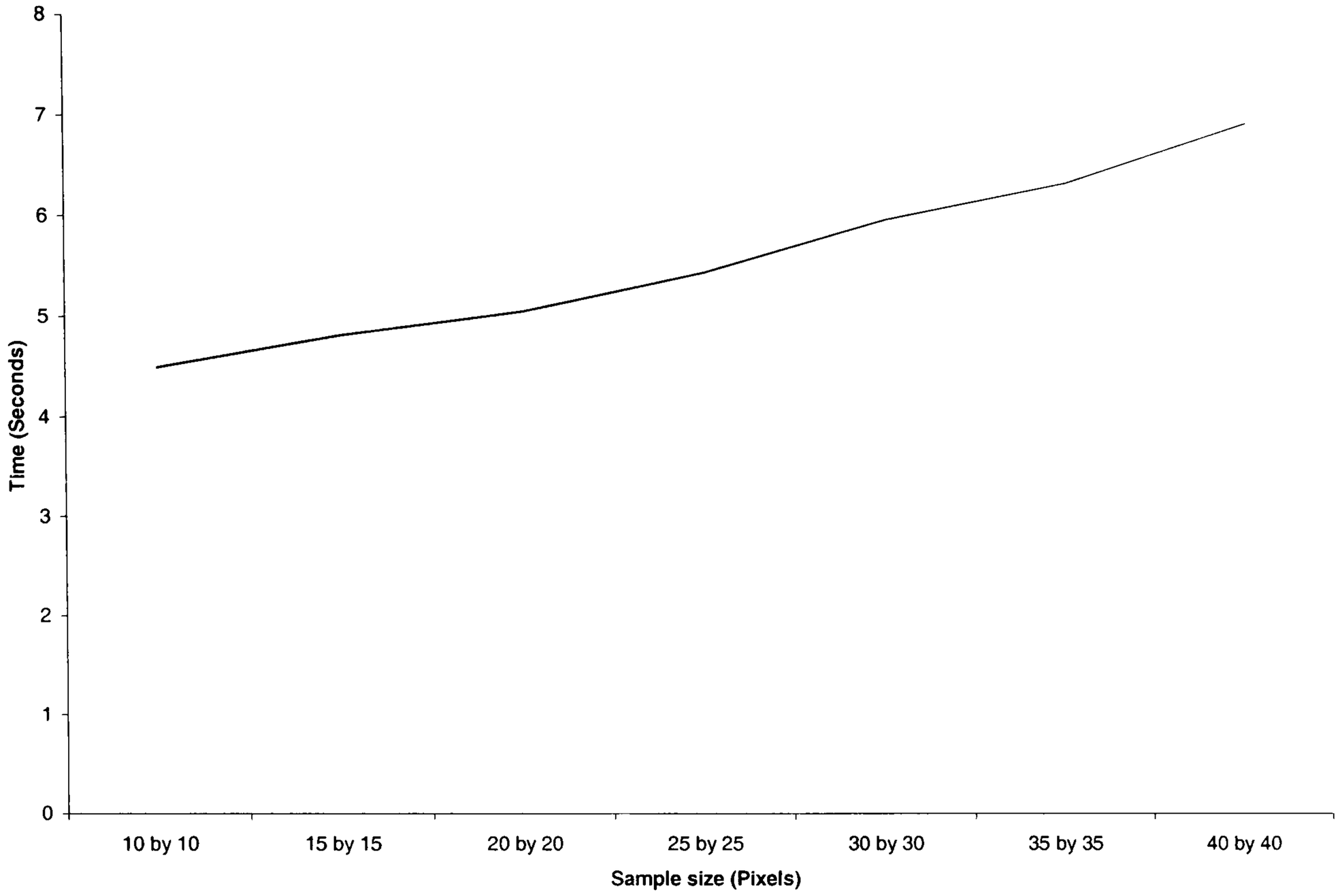


Figure 6.36 Hybrid performance upon 128 by 128 pixel image

The processing time of the back-propagation section of the hybrid neural network can be considered to be negligible due to the fact there are relatively small number of samples to be processed, each sample being a histogram produced by the self-organising map. Using the same architecture that produced the results in section 6.4.2 the back-propagation neural network section was timed in the training phase to give the results depicted in Figure 6.37. The number of images in the training set used to produce the histograms was 60.

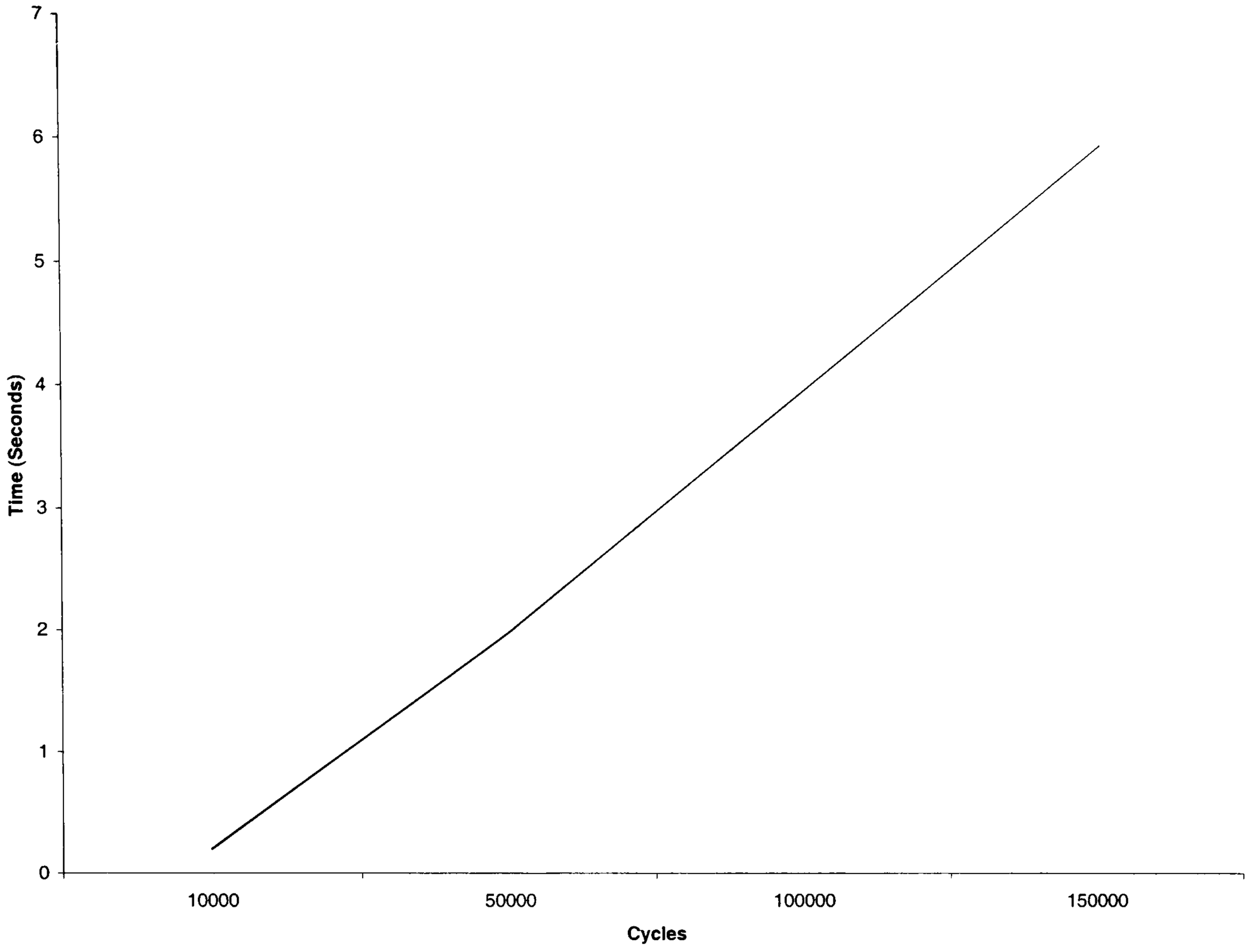


Figure 6.37 BPNN Training Performance

The typical number of cycles needed to train this section of the hybrid neural network is of the order of 50000 iterations. A cycle is deemed to be the presentation of a histogram sample to the input layer of the back-propagation neural network.

6.5 Summary

This chapter has provided the optimal training patterns and architecture dimensions that allow the hybrid neural network to successfully classify images taken from the Brodatz texture series.

This model has been demonstrated to give accurate results with different dimensions and training philosophies. The only common theme being the inter-connecting histograms which accumulate texture data being fed from the self-organising neural network.

A second variant of the hybrid neural network has also been proposed along with experimental to show it to be capable of segmenting aerial images depending upon their texture content.

The hybrid neural network architecture is demonstrated to give a higher accuracy than the spatial grey level dependence matrices benchmark methodology when segmenting aerial images for automotive and tree like textures.

The overall performance of correctly classified pixels when considering both textures across all sixty images is:-

- Hybrid Neural Network 85.49%
- Spatial Grey Level Dependence Matrix / Back-Propagation Neural Network Benchmark 77.85%

With regards to the segmentation of automotive elements there is little to choose from the two systems, with both of them to be capable with little false triggering. The spatial grey level dependence matrix / back-propagation neural network gives a good score of 82.11% with the hybrid neural network marginally ahead with 84.45%.

The overall performance advantage of the hybrid neural network arises due to its ability to cope textures with little contrast; it gives consistent results looking at segmenting tree like textures (86.54%). These textures, which have little contrast, cause problems for the spatial grey level dependence matrix / back-propagation neural network benchmark with a lower score of (73.62%). This unfortunately is the prime cause of lowering the overall score of an otherwise respectable benchmark.

Chapter 7 Conclusions and Recommendations for Further Work

A concise review of the work submitted in this thesis is presented in this the final chapter. This chapter extracts the key elements of this research project with aim of documenting the research activities resulting in the hybrid neural network architecture.

Section 7.1 presents the original contributions to knowledge contained within the thesis with documentary evidence against existing knowledge in this field of research.

Some conclusions are drawn together in Section 7.2 from the work that was necessary to generate this thesis.

Finally, some recommendations are made with regards to further work that can be applied to the hybrid neural network architecture in Section 7.3.

7.1 Original Contributions to Knowledge

The work presented in this thesis can be summarised as follows with regards to the original contributions to knowledge in this field of research: -

- Known and established neural network models have been taken and integrated together to produce a new and novel method of data extraction and classification within a digital image processing environment.

- Via the use of interconnecting histograms between the two neural network models, unique fingerprints can be assigned to individual textures in the training set.

- The hybrid neural network architecture has been demonstrated to be capable of classifying a series of images from the Brodatz series of textures. This has resulted in a publication [Arrowsmith et al 1999].

- The experimental results produced in this thesis have shown the hybrid neural network model to capable of segmenting real world aerial images.

- A critical comparison presented in this thesis has also shown the hybrid neural network architecture to have a higher performance to that of a spatial grey level dependence matrix / back-propagation neural network.

7.2 Conclusions

With the hybrid neural network model, the training set needs to include the textures that are likely to be encountered when sampling an image and some label that can be assigned to the texture classes in the training set. When working with the hybrid neural network model, it was discovered that a broad range of images was required to be included in the training set to obtain a reasonable level of data extraction..

The accuracy of the hybrid neural network is only as good as its training data, which was demonstrated in Section 6.2.5. If a texture is encountered that was not part of the training set then the network will often classify the texture to one that had similar properties in its training set.

When implementing the hybrid neural network in a segmentation mode working upon aerial images, some reference to the altitude of the aircraft must be known, e.g. textures representing trees at a low altitude would be coarse differing from finer textures at high altitudes. If the altitude is to be over a wide range, then different training models must be applied to the interconnecting weights of the neurons. Therefore this range of altitudes must be included in creating multiple training sets, that can be dialled in when a change of altitude is detected. Unfortunately due to lack of imagery, this function was not included in this thesis.

Although at times, particularly in the early stages of the project the amount of work carried out almost became a software engineering project in its own right. Implementing neural networks in a image processing environment requires a large of amount of

complicated source code. However this investment was extremely fruitful when integrating the two neural models together, as it allowed complete control and visibility of the inner working of the neural networks. The author was not limited in anyway by the tool set, any new ideas could be immediately implemented and the results monitored.

7.3 Recommendations for Further Work

The realm of digital image processing and its possible applications are directly linked to the computing power of the day. When this project was first embarked upon, the fastest personal computer of the day had a clock speed of 33 MHz and with 4 Mbytes of RAM, and today as the project draws to a close, machines with clocks speeds of over 1 GHz with 512 Mbytes RAM are becoming common place. It is with this vast increase of computing power available that continues to drive new and ever expanding areas of digital image processing. This is particularly relevant to neural networks. Processing that took hours to implement a neural model can now be achieved in minutes and with the machines of tomorrow these models will be able to run in real time using software emulation rather than hardware implementations such as the RAM node [Aleksander and Morton 1990].

As well as the growth in computing capabilities, the standard of image acquisition and image resolutions has changed vastly across the life of this project. Images with a resolution of 512 by 512 pixels were considered to be of a very high resolution at the outset. However comparing this to the capability of modern machines with display resolutions of 1280 by 1024 pixels, these images could be considered to be almost low resolution. Image acquisition has advanced from video digitisers reliant upon A/D converters through to CCD devices providing true colour images (32bit).

The whole process of off-line manipulation of images is starting to look dated, the ever increasing performance of computing will allow real time processing of images. It is this direction that a great deal of future work can be applied to the hybrid neural network model.

With slight alterations to the hybrid model, a system could be constructed to perform target recognition task. The process of target recognition upon a video feed can be broken down into sub-units. If for instance the video image was digitised to a resolution of 256 by 256 pixels, then 64 hybrid neural networks sampling in 32 by 32 pixel squares could be mapped on to the video feed, Figure 7.1.

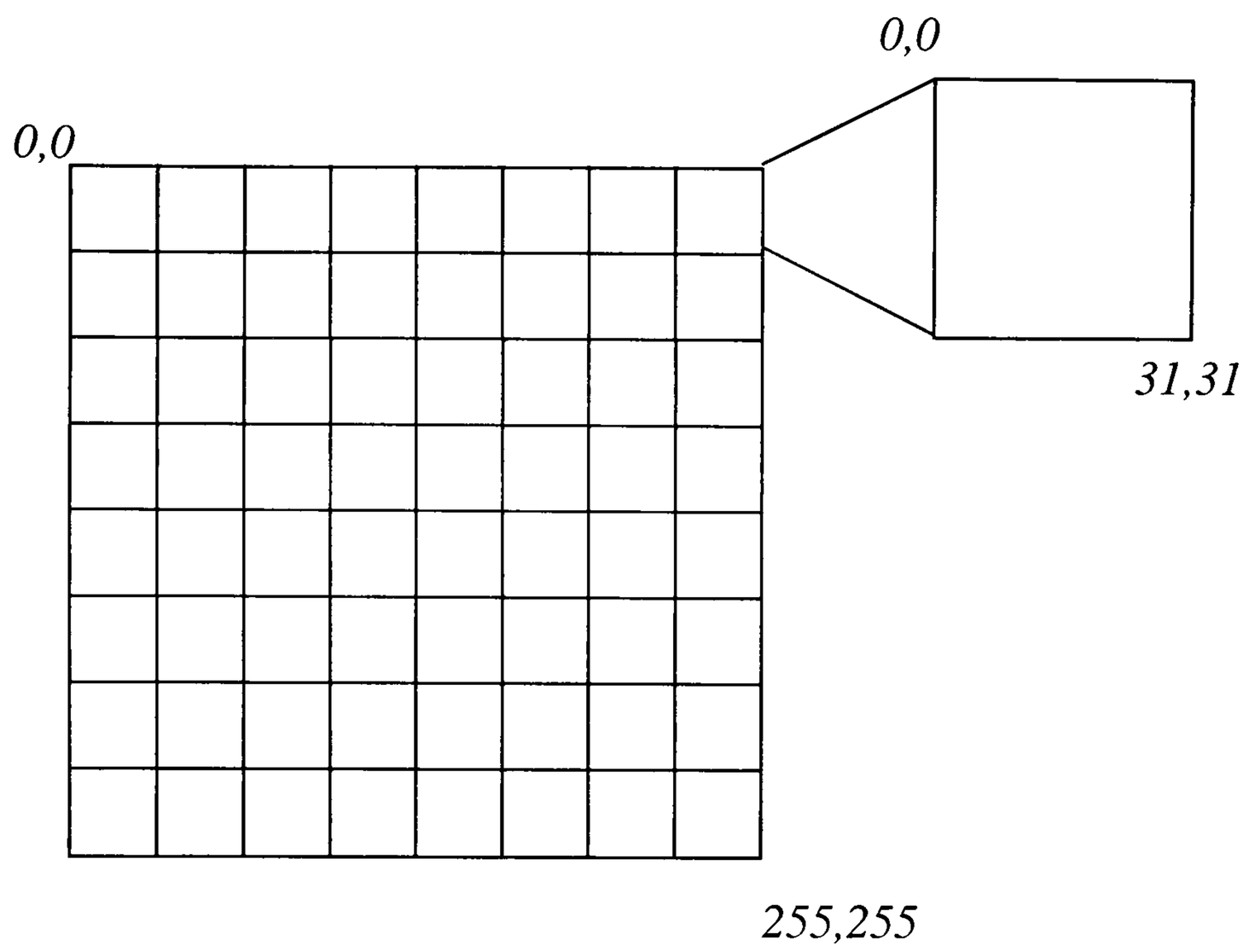


Figure 7.1 Proposed Image Acquisition Architecture for Hybrid Network

Instead of a mask scanning the sample region, the whole sample region could map directly on to an enlarged input layer of the self-organising map. The inter-connecting histogram would now be constructed over time, rather than be built up from samples by the mask traversing the sample region. In practice the inter-connecting histograms would have a certain depth to them, organised as a stack. That is after *n* number of results from the self-organising layer, the first inputs into the histogram will removed to make way for new results being entered into the stack.

The second stage of the hybrid architecture would remain the same. The back-propagation neural network would still take the histogram of self-organising output layer activity as its input and would fire, producing a classification upon its output layer. This result would not be the final stage of processing. The system is now modular, numerous results will be produced by the individual hybrid networks. A third stage would have to be implemented. This stage would take as its input all the results produced by the hybrid modules, Figure 7.2.

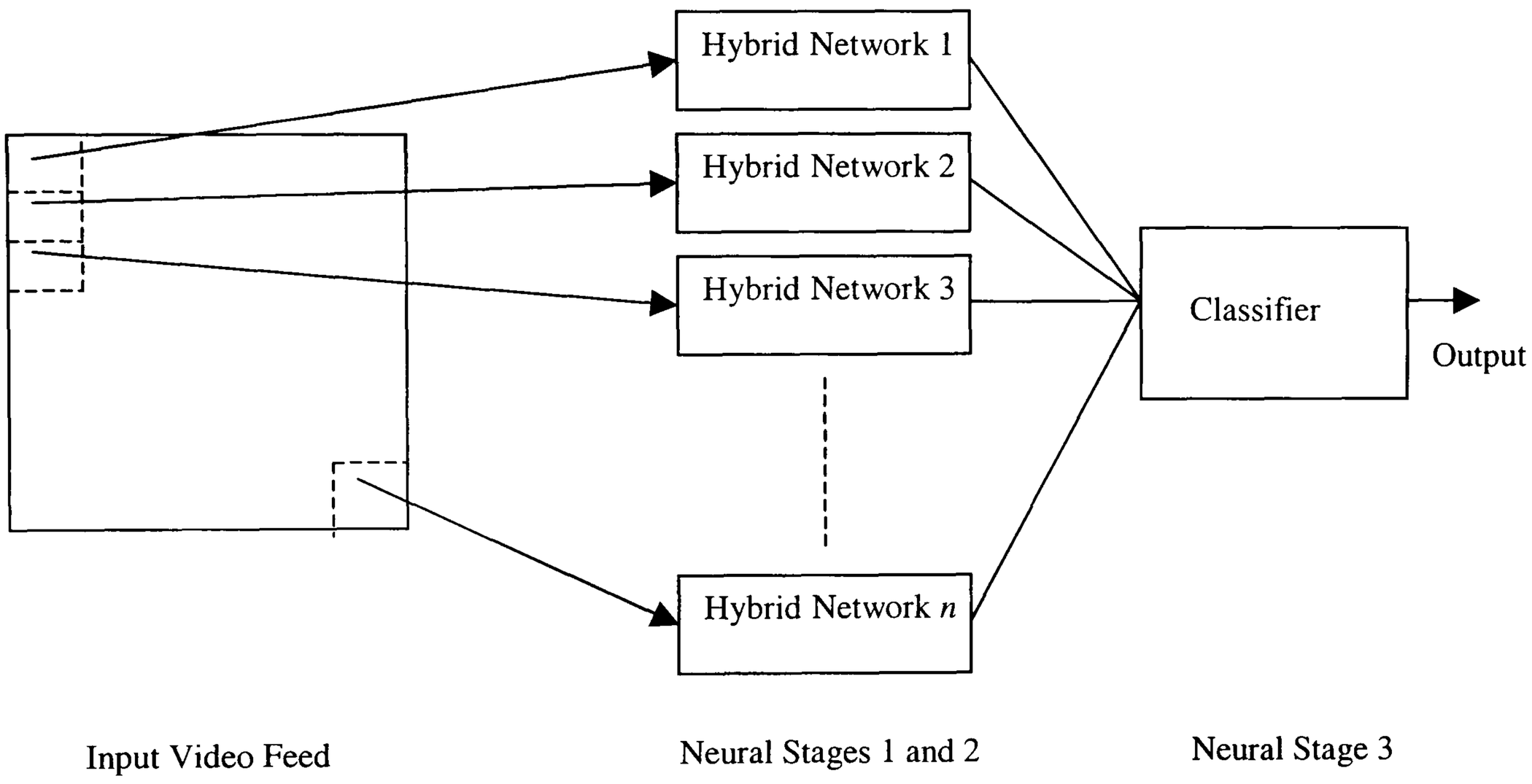


Figure 7.2 Three Stage Neural Network Classifier

Depending on the mode of operation, the third stage could act as majority voter, producing a classification result based upon the most active modules. A different mode of operation might be track objects with regards to speed and angle of motion.

A variation of the three stage classifier might be to replace the third stage with a reconstructed image. This would be constructed by enlarging the output layer from the back-propagation stage of the hybrid neural network. The output layer size would have the same dimensions as the input layer of the self-organising stage. As data propagates through the network, a filtering process could be carried out before the image is reconstructed, Figure 7.3.

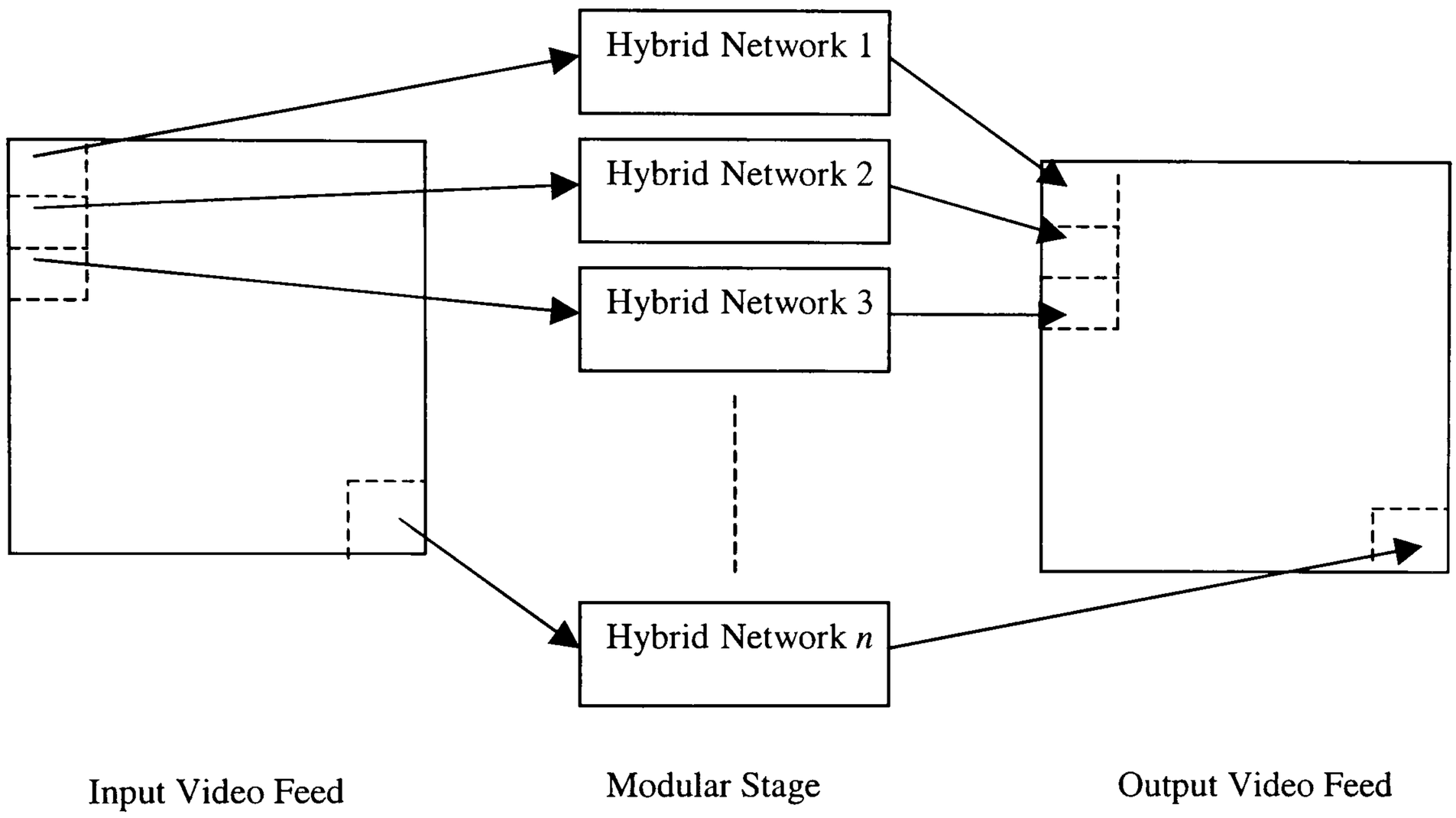


Figure 7.3 Modular Hybrid Neural Network Video Filter

With current advance in processor speeds, the model proposed in Figure 6.2, could be implemented on the next generation of GHz processors if the following assumptions are made. When the hybrid model is in run-time mode, the time taken to execute a self-organising map with an input layer of 32 by 32 neurons and an output layer of 9 by 9 neurons is 4.8 ms on a Pentium III™ 500 MHz processor. Therefore a complete frame could be processed in 0.31 seconds (3.2Hz), resulting in three frames per second which is not very practical.

However with optimised code running in processor cache and increased processor clock speed, frame rates of ten to fifteen frames per second should be achievable on a 1 GHz processor.

The hybrid neural network filter proposed in Figure 6.3, is for the time being not imminently achievable. If a model was to be implemented with the dimensions of a 32 by 32 neuron input layer feeding a 9 by 9 neuron output layer for the self-organising map. A 9 by 9 input layer feeding a 5 by 5 middle layer driving a 32 by 32 neuron output layer the back-propagation neural network would take 0.24 seconds for one firing. Therefore a complete frame made up from 256 by 256 pixels would take 15.36 pixels.

This may seem impracticable, but referring back to the description of the pace of technology progress in the beginning of this section, this need not be the case. If another part time research project were to be undertaken to investigate hybrid neural network models operating on video images, it is most probable that in six years when the project draws to a close, the hardware of the day will more than capable of achieving full motion video using the future work proposed here. It has to be remembered that when this research was first undertaken, the World Wide Web (WWW) was only just beginning to be implemented through the Mosaic™ software from CERN and the fastest modem speed was 2400 bps. Technology will often provide surprising advances in short timescales.

One aspect of training that was not explored in this thesis but might well lend itself to the scenario described above is that of continuous training. Real world video feed rarely matches exactly the quality or content of laboratory image sequences. Allowing the self-organising layer to perform minor updates upon its inter-connecting weights would tailor the application to its surroundings. This would be possible, as there is no supervision of this stage in the training phase.

References

ALEKSANDER Igor, MORTON Helen

An Introduction to Neural Computing

Chapman Hall, London. 1990

ANDERSON James

An Introduction to Neural Networks

MIT Press 1995, ISBN 0-262-01144-1

AL-JANOBI Abdulrahman

Performance Evaluation of Cross Diagonal Texture Matrix Method of Texture Analysis

Pattern Recognition

2001, 34, pp 171 - 180

ARROWSMITH MJ, VARLEY MR, PICTON PD, HEYS JD

Hybrid Neural Network System for Texture Analysis

IEE 7th International Conference on Image Processing and its Applications, Manchester

1999, Volume 1, pp 339 – 343

AUGUSTEIJN Marijke F, CLEMENS Laura E, SHAW Kelly A

Performance Evaluation of Texture Measures for Ground Cover Identification In Satellite Images by Means of a Neural Network Classifier

IEEE Transactions on Geoscience and Remote Sensing

May 1995, Volume 33, Number 3, pp 616 – 626

BAHLMANN Claus, HEIDEMANN Gunther, RITTER Helge

Artificial Neural Networks for Automated Quality Control of Textile Seams

Pattern Recognition

1999, 32, pp 1049 - 1060

BARALDI Andrea, PARMIGGIANI Flavio

An Investigation of the Textural Characteristics Associated with Grey Level Cooccurrence Matrix Statistical Parameters

IEEE Transactions on Geoscience and Remote Sensing

March 1995m Volume 33, Number 2, pp 293 – 304

BHATT MR, DESAI UB

Robust Image Restoration Algorithm Using Markov Random Field Model

Graphical Models and Image Processing

1994, Vol.56, No 1, pp 61 - 74

BHATTACHARYA U, CHAUDHURI BB, PARUI SK

An MLP-Based Texture Segmentation Method Without Selecting a Feature Set

Image and Vision Computing

1997, 15, 937 – 948

BIEBELMANN E, K☐PPEN M, NICKOLAY B

Practical Applications of Neural Networks in Texture Analysis

Neurocomputing

1996, 13, pp 261- 279

BISCHOF H, SCHNEIDER W, PINZ AJ

Multispectural Classification of Landsat Images Using Neural Networks

IEEE Transactions on Geoscience and Remote Sensing

May 1992, Volume 30, Number 3, pp 482 - 490

BRODATZ Phil

Textures – A Photographic Album for Artists and Designers

Dover, New York 1966

BRUZZONE L, CONESE C, MASELLI F, ROLI F

Multisource Classification of Complex Rural Areas by

Statistical and Neural Network Approaches

Photogrammetric Engineering and Remote Sensing

May 1997, Volume 63, Number 5, pp 523 – 533

CHANG T, KUO C J

Texture Analysis and Classification with tree structured wavelet transform

IEEE Transactions on Image Processing

1993, 2, pp 429 – 441

CHAUDHURI BB, KUNDU Pulak, SARKAR Nirupam

Detection and Gradation of Oriented Texture

Pattern Recognition Letters

1993, 14, pp 147 - 153

CLAUSI David A, JERNIGAN Ed

A Fast Method to Determine Co-Occurrence Texture Features

IEEE Transactions on Geoscience and Remote Sensing

1998, Vol. 36, No 1, pp 298 - 300

CLARK M, BOVIK A C

Texture Segmentation Using Gabor Modulation / Demodulation

Pattern Recognition Letters

1987, 6, pp 261 - 267

CLAUSI David A, JERNIGAN Ed

Designing Gabor Filters for Optimal Texture Separability

Pattern Recognition

2000, 33, pp 1835 - 1849

COGGINS J M, JAIN A K

A Spatial Filtering Approach to Texture Analysis

Pattern Recognition Letters

1985, 3, pp 195 - 203

CONNERS Richard, HARLOW Charles

A Theoretical Comparison of Texture Algorithms

IEEE Transactions on Pattern Analysis and Machine Vision

May 1980, Volume PAMI-2, Number 3, pp 204 – 222

CONNERS Richard, TRIVEDI Mohan, HARLOW Charles

Segmentation of a High-Resolution Urban Scene Using Texture Operators

Computer Vision, Graphics and Image Processing

1984, 25, pp 273 – 310

CROSS G C, JAIN A K,

Markov Random Field Texture Models

IEEE Transactions on Pattern Analysis and Machine Intelligence

1983, 5, pp 25 – 29

CZIRIA Balazs, DERGANC Joze, OLLE Krisztian, PRINCE Simon,

PETKOVIC Tomislav

Texture Analysis Using Wavelet Transform

9th Summer School on Image Processing

12th-21st July 2001, Szeged, Hungary

DANIELL Cindy E, KEMSLEY David H, LINCOLN William P, TACKETT Walter A, BARAGHIMIAN Gregory A

Artificial Neural Networks for Automated Target Recognition

Optical Engineering

1992, Vol. 31 No 12, pp 2521 - 2531

DAUBECHIES I

Ten Lectures on Wavelets

Capital City Press

1992, Montpellier, Vermont.

DAUGMAN JG

Networks for Image Analysis: Motion and Texture

Proc. of International Joint Conference on Neural Networks, Washington DC

1989, Vol. 1, pp 189 - 193

DE NATALE Francesco GB

Rank Order Functions for the Fast Detection of Texture Faults

International Journal of Pattern Recognition and Artificial Intelligence

1996, Vol, 10, No 8, pp 971 - 984

DE RIDDER Dick, DUIN Robert, VERBEEK Piet, VAN VLIET Lucas

On the Application of Neural Networks to Non-linear Image Processing Tasks

The Fifth International Conference on Neural Information Processing, Japan

October 1998, pp 161 – 165

DRIMBAREAN A, WHELAN PF

Experiments in Colour Texture Analysis

Pattern Recognition Letters

2001, 22, pp 1161 – 1167

FRYE Richard E, LEDLEY, Robert S

Texture Discrimination Using Discrete Cosine Transformation Shift-Intensive (DCTSIS) Descriptors

Pattern Recognition

2000, 33, pp 1585 - 1598

FUKUE Kiyonari, SHIMODA Haruhisa, SAKATA Toshibumi

Spatial Landcover Classification Using a Neural Network Driven by Co-occurrence Matrix for Land Cover Elements.

International Geoscience and Remote Sensing Symposium , Seattle WA.

6^{th} – 10^{th} July 1998, v15, pp 1137-1139.

GHINELLI BMG, BENNETT JC

Feasibility of Employing Artificial Neural Networks for Emergent Crop Monitoring in SAR systems

IEE Processing of Radar, Sonar Navigation

1998, Vol. 145, No 5, pp 291 – 296

GOLTSEV Alexander, WUNSCH Donald C

Inhibitory Connections in the Assembly Neural Network for Texture Segmentation

Neural Networks

1998, 11, pp 951 - 962

GOTLIEB Calvin C, KREYSZIG Herbert E

Texture Descriptors Based on Co-Occurrence Matrices

Computer Vision and Image Processing

1990, 51, pp 70 – 86

GREENBERG Shlomo, GUTERMAN Hugo

Neural Network Classifiers for Automatic Real World Aerial Image Recognition

Applied Optics

1996, Vol. 35, No 23, pp 4599 - 4609

HADDON JF, BOYCE JF

Co-occurrence Matrices for Image Analysis

Electronics and Communication Engineering Journal

April 1993, Volume 5, Part 2, pp 71 – 83

HARALICK RM

Statistical and Structural Approaches to Texture

Proceedings of the IEEE

May 1979, Volume 67, Number 5, pp 786 – 804

HARALICK Robert, SHANMUGAM K, DINSTEIN Its'hak.

Textural features for Image Classification

IEEE Transactions on Systems, Man and Cybernetics

November 1973, Volume SMC-3, Number 6, pp 610 - 621

HAYKIN Simon

Neural Networks, A Comprehensive Foundation

Macmillan, New York. 1994

HE Dong-Chen, WANG Li

Texture Features Based on Texture Spectrum

Pattern Recognition

1991, Vol. 24, No 5, pp 391 – 399

HEALEY Christopher G, ENNS James T

Building Perceptual Textures to Visualize Multidimensional Datasets

IEEE Visualization '98"

1998 , pp 111-118

HSU Tao-I, KUO Jiann Ling, WILSON R

A Multiresolution Texture Gradient Method for Unsupervised Segmentation

Pattern Recognition

2000, 33, pp 1819 - 1833

INGGS MR, ROBINSON AD

Ship Target Recognition Using Low Resolution Radar and Neural Networks

IEEE Transactions on Aerospace and Electronic Systems

1999, Volume 35, Number 2, pp 386 – 393

JAHNE Bernd

Digital Image Processing: Concepts, Algorithms and Scientific Applications

Springer Verlag, New York, 1995 3rd Edition ISBN 3-540-59298-9

JAN Sen-Ren, HSUEH Yuang-Cheh

Window Size Determination for Granulometrical Structural Texture

Classification

Pattern Recognition Letters

1998, 19, pp 439 - 446

JAMES Mike

Pattern Recognition

BSP, Oxford. 1987 ISBN 0-632-01885-2

JAWAHAR CV, RAY AK

Incorporation of Gray Level Imprecision in Representation and Processing of Digital Images

Pattern Recognition Letters

1996, 17, pp 541 – 546

JULESZ B

Texton Gradients: The Texton Theory Revisited

Biological Cybernetics

1986, 54, pp 245 – 251

KARRAS DA, KARKANIS SA, MERTZIOS BG

Supervised and Unsupervised Neural Network Methods Applied to Textile Quality Control Based on Improved Wavelet Feature Extraction Techniques

International Journal of Computer Mathematics

1998, Vol. 67, pp 169 - 181

KASPARIS T, CHARALAMPIDIS D, GEORGIOPOULOS M, ROLLAND J

Segmentation of Textured Images Based on Fractals and Image Filtering

Pattern Recognition

2001, 34, pp 1963- 1973

KOHONEN Teuvo

The Self-organizing Map

Neurocomputing

1998, Volume 21, pp 1 - 6

KULKARNI AD

Artificial Neural Networks for Image Understanding

VNR Computer Library, New York. 1994

LAINE Andrew, FAN Jian

Texture Classification by Wavelet Packet Signatures

IEEE Transactions on Pattern Analysis and Machine Intelligence

1993, Vol. 15, No 11, pp 1186 - 1197

LAMPINEN Jouko, SMOLANDER Seppo

Self-Organizing Feature Extraction in Recognition of Wood Surface Defects and Color Images

International Journal of Pattern Recognition and Artificial Intelligence

1996, Volume 10, Number 1

LAWS Keith I

Textured Image Segmentation

Ph.D. Thesis, University of Southern California, 1980

LIN Hsin-Chih, WANG Ling-Ling, YANG Shi-Nine

Extracting Periodicity of a Regular Texture Based on Autocorrelation Functions

Pattern Recognition Letters

1997, 18, pp 433 - 443

LIPPMANN Richard P

An Introduction to Computing with Neural Nets

IEEE ASSP Magazine,

April 1987, pp 417 – 435

LISBOA PGJ.

Neural Networks, Current Applications

Chapman & Hall, London. 1992

LIU F, PICARD RW

Periodicity, Directionality and Randomness: Wold Features for Image Modelling and Retrieval

IEEE Transactions on Pattern Analysis and Machine Intelligence

1996, Vol. 18, No 7, pp 722 - 733

LORETTE A, DESCOMBES X, ZERUBIA J

Texture Analysis Through a Markovian Modelling and Fuzzy Classification: Application to Urban Area Extraction from Satellite Images

International Journal of Computer Vision

2000, 36(3), pp 221 - 236

LOW Adrian

Introductory Computer Vision and Image Processing

McGraw-Hill, UK. 1991 ISBN-0-07-707403-3

LU Chun-Shein, CHUNG Pau-Choo

Wold Features for Unsupervised Texture Segmentation

IEEE International Conference on Pattern Recognition

August 1998, pp 1689 - 1693

McCULLOCH Warren, PITTS Walter

A Logical Calculus of the Ideas Immanent in Nervous Activity

Bulletin of Mathematical Biophysics

1943, Volume 5, pp 115 – 133

MA WY, MANJUNATH BS

Texture Features and Learning Similarity

Proceedings of IEEE International Conference on Computer Vision and Image Processing, San Francisco, CA. June 1996

MALLAT Stephane G

A Theory for Multiresolution Signal Decomposition: The Wavelet Representation

IEEE Transactions on Pattern Analysis and Machine Intelligence

1989, Vol. 11, No 7, pp 674 - 693

MARANA AN, COSTA L da F, VELASTIN SA, LOTUFO RA

Oriented Texture Classification Based on Self-Organizing Neural Network and Hough Transform

IEEE International Conference on Acoustics, Speech and Signal Processing, Munich

April 1997, pp 2773 – 2775

MARCEAU Danielle, HOWARTH Philip, DUBOIS Jean-Marie, GRATTON Denis

Evaluation of the Grey-Level Co-Occurrence Matrix Method for Land-Cover Classification Using SPOT Imagery

IEEE Transactions on Geoscience and Remote Sensing

July 1990, Volume 28, Number 4, pp 513 – 519

MATHER Paul M, BRANDT Tso, KOCH Magaly

An Evaluation of Landsat TM Spectral Data and SAR- Derived Textural Information for Lithological Discrimination in the Red Sea Hills, Sudan

International Journal of Remote Sensing

1998, Volume 19, Number 4, pp 587 - 604

MATLAB

http://www.mathworks.com/

MINSKY Marvin, PAPERT Seymour

Perceptrons : An Introduction to Computational Geometry

MIT Press, 1969. ISBN 0-262-63111-3

MUHAMAD Anwar K, DERAVI Farzin

Neural Network Texture Classifiers Using Direct Input Cooccurrence Matricies

Proc. International Conference on Acoustic Speech and Signal Processing

(ICASSP 93)

Minneapolis 1993 pp v117-120

MURTAGH F,

Interpreting the Kohonen Self-Organizing Feature Map Using Contiguity-

Constrained Clustering

Pattern Recognition Letters

1995, 16, pp 399 - 408

OH Gyuhwan, LEE Seungyong, SHIN Sung Yong

Fast Determination of Textural Periodicity Using Distance Matching Function

Pattern Recognition Letters

1999, 20, pp 191 - 197

OJA Erkki, VALKEALAHTI Kimmo

Co-occurrence Map: Quantizing Multidimensional Texture Histograms.

Pattern Recognition Letters

1996, 17, pp 723-730

OJALA Timo, PIETIK ¡ INEN Matti

Unsupervised Texture Segmentation Using Distributions

Pattern Recognition

1999, 32, pp 477 - 486

PAL Nikhil, PAL Sankar

A Review on Image Segmentation Techniques

Pattern Recognition

1993, Volume 26, Number 9, p 1277 – 1294

PALA P, SANTINI S,

Image Retrieval by Shape and Texture

Pattern Recognition

1999, 32, pp 517 - 527

PARKKINEN J, SELK ǂ INAHO K

Detecting Texture Periodicity from the Co-Occurrence Matrix

Pattern Recognition Letters

1990, 11, pp 43 - 50

PASCHOS George

Chromatic Correlation Features for Texture Recognition

Pattern Recognition Letters

1998, 19, pp 643 - 650

PASQUARIELLO G, SATALINO G, LA FORGIA V, SPILOTROS F

Automatic Target Recognition for Naval Traffic Control using Neural Networks

Image and Vision Computing

1998, 16, pp 67 – 73

PENTLAND A

Fractal Based Description of Natural Scenes

IEEE Transactions on Pattern Analysis and Machine Intelligence

1984, 9, pp 661-674

PORTILLA Javier, SIMONCELLI Eero P

A Parametric Texture Model Based on Joint Statistics of Complex Wavelet

Coefficients

International Journal of Computer Vision

2000, 40, 1, pp 49 – 71

RAO Ravishankar

A Taxonomy for Texture Description and Identification

Springer-Verlag, 1990, ISBN 0-387-97302-8

RAGHU PP, YEGNANARAYANA B

Multispectural Image Classification Using Gabor Filters and Stochastic

Relaxation Neural Network

Neural Networks

1997, Volume 10, Number 3, pp 561-572.

RAGHU PP, POONGODI R, YEGNANARAYANA B

A Combined Neural Network Approach for Texture Classification

Neural Networks

1995, Volume 8, Number 6, pp 975-987.

REED TR, WECHSLER H

Segmentation of Textured Images and Gestalt Organization Using Spatial /

Spatial Frequency Representations

IEEE Transactions on Pattern Analysis and Machine Intelligence

1990, 12, pp 1-12.

ROSENBLATT Frank.

The Perceptron : A probabilistic Model for Information Storage and

Organisation in the Brain

Psychological Review

1958, pp 92 – 114, ISSN 0033-295X

RUIZ-DEL-SOLAR Javier

TEXSOM: Texture Segmentation using Self-Organizing Maps

Neurocomputing

1998, 21, pp 7 - 18

RUMELHART DE, HINTON GE, WILLIAMS RJ

Learning Internal Representations by Error Propgation Chapter 8

Parallel Distributed Processing Vol 1

Edited by RUMELHART DE, MCCLELLAND JL

MIT Press, Massachusetts, 1986

SERPICO SB, BRUZZONE L, ROLI F

An Experimental Comparison of Neural and Statistical Non-parametric

Algorithms for Supervised Classification of Remote-Sensing Images

Pattern Recognition Letters

1996, 17, pp 1331 – 1341

SHEN HC, BIE CYC

Feature Frequency Matrices as Texture Image Representation

Pattern Recognition Letters

1992, 13, pp 195 - 205

SKLANSKY Jack

Image Segmentation and Feature Extraction

Transactions on Systems, Man and Cybernetics,

April 1978, Volume 8, Number 4.

SONKA Milan, HLAVAC Vaclav, BOYLE Roger

Image Processing, Analysis and Machine Vision

Chapter 13 : Texture

1993, Chapman & Hall. ISBN 0-412-45570-6

STAROVOITOV Valery V, JEONG Sang-Yong, PARK Rae-Hong

Texture Periodicity Detection: Features, Properties and Comparisons

IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans

1998, Vol. 28, No 6, pp 839 - 849

STEWART Clayton, LU Yi-Chuan, LARSON Victor

A Neural Clustering Approach for High Resolution Radar Target Classification

Pattern Recognition

1994, Volume 27, Number 4, pp 503 – 513

STRAND Jarle, TAXT Torfinn

Local Frequency Features for Texture Classification

Pattern Recognition

1994, Vol. 27, No 10, pp 1387 - 1406

SUEN Pei-Hsiu, HEALEY Glenn

Modeling and Classifying Color Textures Using Random Fields in a Random Environment

Pattern Recognition

1999, 32, pp 1009 – 1017

TSAI DM, HSIEH CY

Automated Surface Inspection for Directional Textures

Image and Vision Computing

1999, 18, pp 49 - 62

UNSER Michael

Local Linear Transforms for Texture Measurements

Signal Processing

1986, 11, pp 61 - 79

USB Campus 1994 Aerial Photography Series

http://www.lib.berkeley.edu//EART/campus/

VALKEALAHTI Kimmo, OJA Errki

Texture Classification with Single and Multiresolution Co-occurrence Maps.

International Journal of Pattern Recognition and Artificial Intelligence

1998, Volume 12, Number 4, pp 437-452.

VAN GOOL L, DEWAELE P, OOSTERLINCK A

Survey – Texture Analysis Anno 1983

Computer Vision, Graphics and Image Processing

1983, 29, pp 336 – 357

VAN HULLE MM, TOLLENAERE T

A Modular Artificial Neural Network for Texture Processing.

Neural Networks

1993, Volume 6, pp 7-32.

VISA Ari

Identification of Stochastic Textures with Multiresolution Features and

Self-Organizing Map

Proceedings of the 10th International Conference on Pattern Recognition,

Atlantic City

June 1990, pp 518 – 522

VISA Ari

Unsupervised Image Segmentation Based on a Self-organizing Feature Map and

a Texture Measure.

Proc. Of the 11th IAPR International Conference of Pattern Recognition,

Netherlands.

1992, Volume 3 , pp 101-104

VAN DE WOUWER G, SCHEUNDERS P, LIVENS S, VAN DYCK D

Wavelet Correlation Signatures for Color Texture Characterization

Pattern Recognition

1999, 32, pp 443 - 451

VON DER MALSBERG Chris

Self-Organizing of orientation sensitive cells in the striate cortex.

Kybernetik, 14, pp 85-100, 1973

WALKER Ross F, JACKWAY Paul, LONGSTAFF I D,

Improving Co-Occurrence Matrix Feature Discrimination

Proc. of 3rd Conf. On Digital Image Computing: Tecnhiques and Applications.

1995m 6-8th December, pp 643 – 648

WANG Jing-Wein, CHEN Chin-Hsing, CHIEN Wei-Ming, TSAI Chih-Ming

Texture Classification Using Non-Separable Two-Dimensional Wavelets

Pattern Recognition Letters

1998, 19, pp 1225 - 1234

WANG Zhiling, GUERRIERO A, DE SARIO M

Comparison of Several Approaches for the Segmentation of Texture Images

Pattern Recognition Letters

1996, 17, pp 509 - 521

WESZKA Joan, DYER Charles, ROSENFELD Azriel

A Comparative Study of Texture Measures for Terrain Classification

IEEE Transactions on Systems, Man and Cybernetics

1976, Volume SMC-6, Number 4, pp 269 – 285

YOSHIMURA Motohide, OE Shunichiro

Evolutionary Segmentation of Texture Image Using Genetic Algorithms

Towards Automatic Decision of Optimum Number of Segmentation Areas

Pattern Recognition

1999, 32, pp 2041 - 2054

ZUCKER Steven W, TERZOPOULOS Demetri

Finding Structure in Co-Occurrence Matrices for Texture Analysis

Computer Graphics and Image Processing

1980, 12, pp 286 - 308

Appendix A Software

A.1 Software Development

Upon embarkation on this project a conscious decision was made to develop all the tools used for generating neural networks via custom software rather than relying upon third party software vendors. Some of the factors influencing the decision were the ability to model any type of structure and the author's software background. The only commercial neural modelling software available at the time (MATLAB™) did not offer a function to customise the neural structures or the ability to access the structures to log data or monitor operation.

The initial software was developed in MSDOS™, using the programming language C++. It performed small neural network architectures quickly and accurately, however specific device drivers had to be written to view the result on the VDU. Because these used software calls to the machines video card, it quickly became un-portable to other machines. Another limitation encountered was the limit on array declarations, MSDOS™ applications had a memory limit of 640K. This meant that as the neural networks grew in size, less memory was becoming available to implement them. A graphic example of this can be seen in Appendix C. An array of integers was created the same size as an input image, 512 pixels by 512 pixels. With a word representing an integer, the memory allocation for this structure is over 500 Kbytes, which would be a non-starter as far as MSDOS is concerned.

At this point in the research programme decision was taken to move up to developing software for the newly released operating system Windows 95™. This operating system offered several advantages, the primary one being the memory model. There were no limits on using memory, and if the machine ran out of physical RAM it would create virtual memory by using hard disk space. A new compiler was purchased by the department specifically aimed at Windows 95™ development, this being Borland Visual C++ Version 5.0™. However being a different system, new software had to be developed to make use of the Windows operating system. The new software allowed images to be cut and pasted between other windows applications such as Microsoft Word™ and Paint Shop Pro™. It also did not tie up the system's resources because Windows 95™ is a multi-tasking environment unlike MSDOS™. Tasks could be therefore running in the background of the operating system.

Figure A.1 shows a snapshot of the image processing tool developed to implement neural networks and conventional methods of image processing, with a set of images in the Brodatz training set stacked up ready for processing. When experimenting with different parameters such as sample size and learning rate it is useful to have the ability to compare immediately the results to those achieved previously, depicted in Figure A.2.

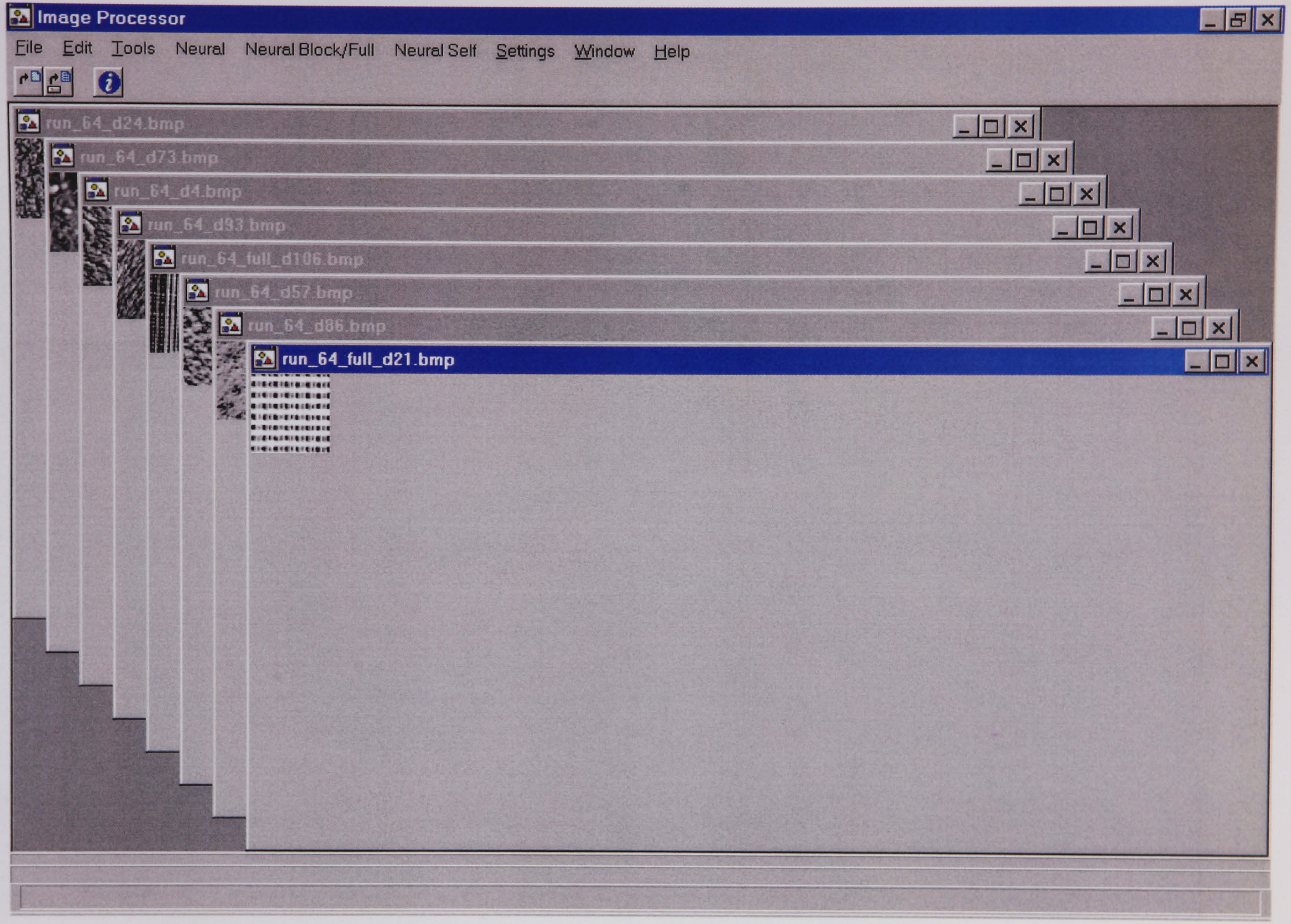


Figure A.1 Image Processor in Training Mode

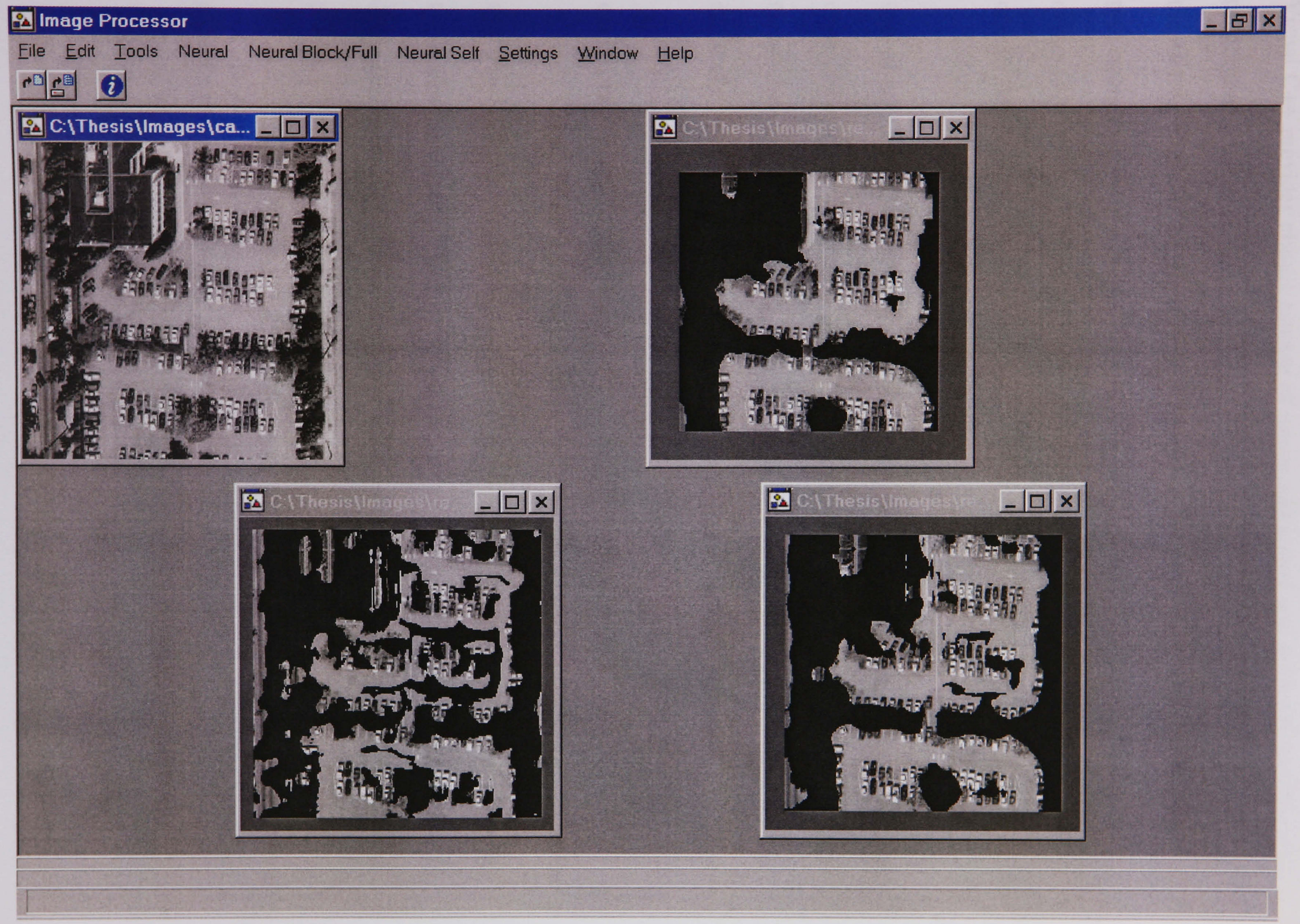


Figure A.2 Image Processor Results Evaluation

A.2 Data Structures

Visual C++ provides classes capable of image addressing and manipulation. The C++ programming language also offers some very useful data structure declarations that lend themselves to image processing and neural networks. To give a flavour of the image processing environment developed for this research project, a few of the structures are explored in the next two sections. This is however is a very small fraction of the amount of code developed to meet the requirements of the project.

One of the most useful classes provided by Borland Visual C++ is that of the Tdib, this class gives access to device independent bitmaps (DIB) as discussed in Section 1.1. Once a bitmap has been displayed in the image processing environment, a copy can be made of it by using the GetBits function to set up a pointer in memory :-

```
unsigned char *imagecopy= (unsigned char*)Dib->GetBits();
```

Variable imagecopy now points to the beginning of memory when the bitmap is stored. Unsigned characters are used as they represented a single byte, 8 bit grey scales with 256 grey levels. To address individual pixel locations, offsets are used against this pointer :-

```
pixel=*(imagecopy+(x+ width * y));
```

Variable pixel contains the grey scale value at position *x,y*. The DIB class also has another useful function that it can return the height and width of an image :-

```
int width  = Dib->Width();

int height = Dib->Height();
```

Variables width and height are given integer values to cater for images larger the 256 pixels (maximum value for unsigned characters).

An important note regarding the representation of bitmaps by the DIB class is that they are constructed the opposite way to those of conventional television pictures. The source point is the bottom left hand corner, with the image terminating at the top right hand corner, illustrated by Figure A.3 for a square image of 256 pixels.

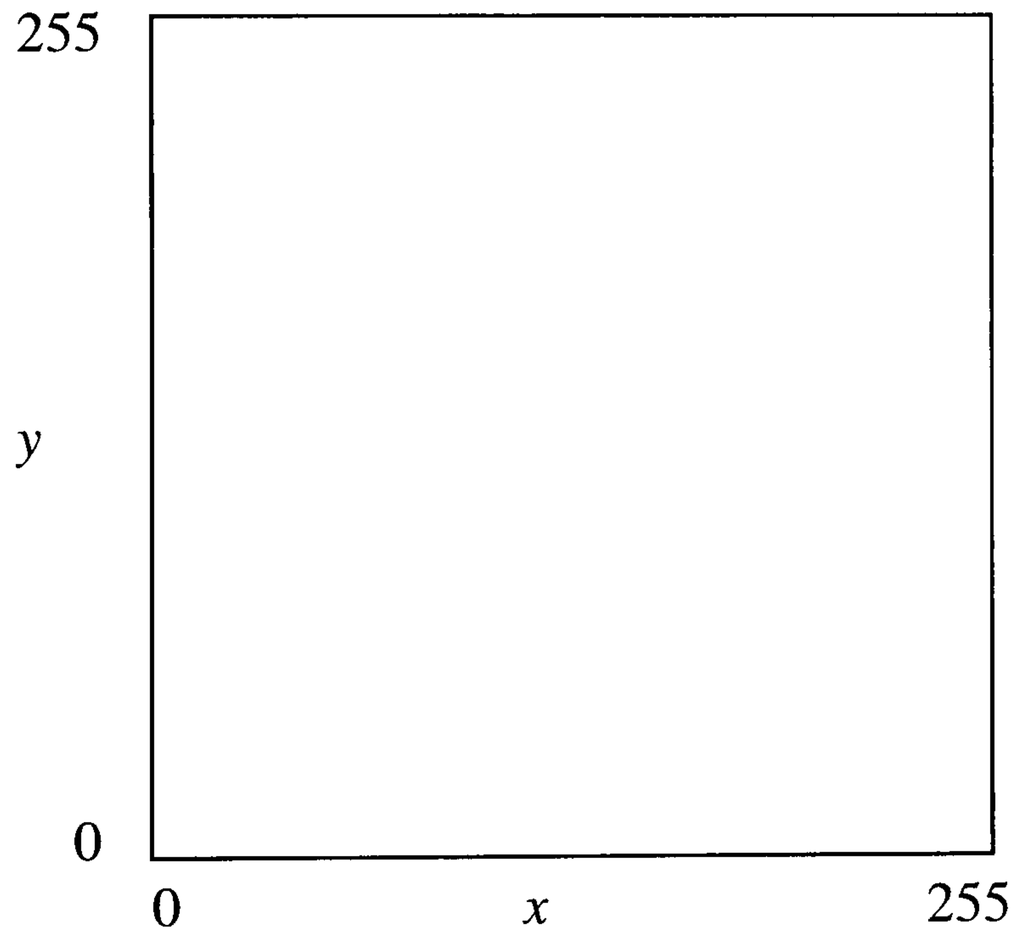


Figure A.3 DIB Representation

The following excerpt of code sets up the three layers of the back-propagation network, a five by five square input layer, a three by three square middle layer and a flat output layer consisting of 2 neurons.

```
const middle_x=3;

const middle_y=3;

const output_x=2;

const output_y=1;

const input_x=5;

const input_y=5;

float input[input_x][input_y];

float middle[middle_x][middle_y];

float output[output_x][output_y];
```

The connections between the layers can be modelled by :-

```
struct

middle_layer_weights{float wght[input_x][input_y];};

struct

output_layer_weights{float wght[middle_x][middle_y];};


struct middle_layer_weights middle_w[middle_x][middle_y];

struct output_layer_weights output_w[output_x][output_y];
```

To reference the weights for example connecting the middle layer to the input layer then the following can be used.

```
middle_w[x][y].wght[xx][yy]
```

Where X and Y reference the position of the neuron in the middle layer that is going to have its interconnecting weights addressed. XX and YY address the individual weight by reference to a particular neuron in the input layer.

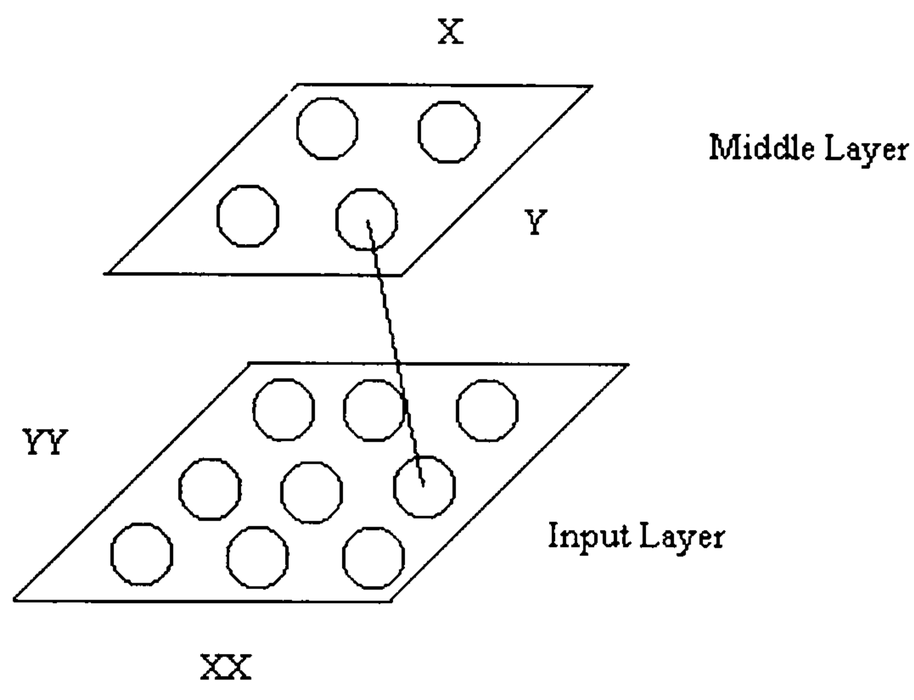


Figure A.4 Data Structure Example

If the array numbering scheme starts top left (0,0), then Figure A.4 shows that neuron 1,1in the middle layer is connected to neuron 2,1 in the input layer. The code used to address the interconnecting weight would be:-

```
middle_w[1][1].wght[2][1]
```

Appendix B Cross-Correlation

The cross-correlation process presented in Section 3.5 produces a coefficient that indicates periodicity/sample match within the image sample being processed. Close examination of Brodatz Image D55 (Straw Matting) Figure B.1, shows there are regular repetitions in the horizontal and vertical planes. The repetitions in the vertical plane appear to have twice the frequency to those in vertical plane.

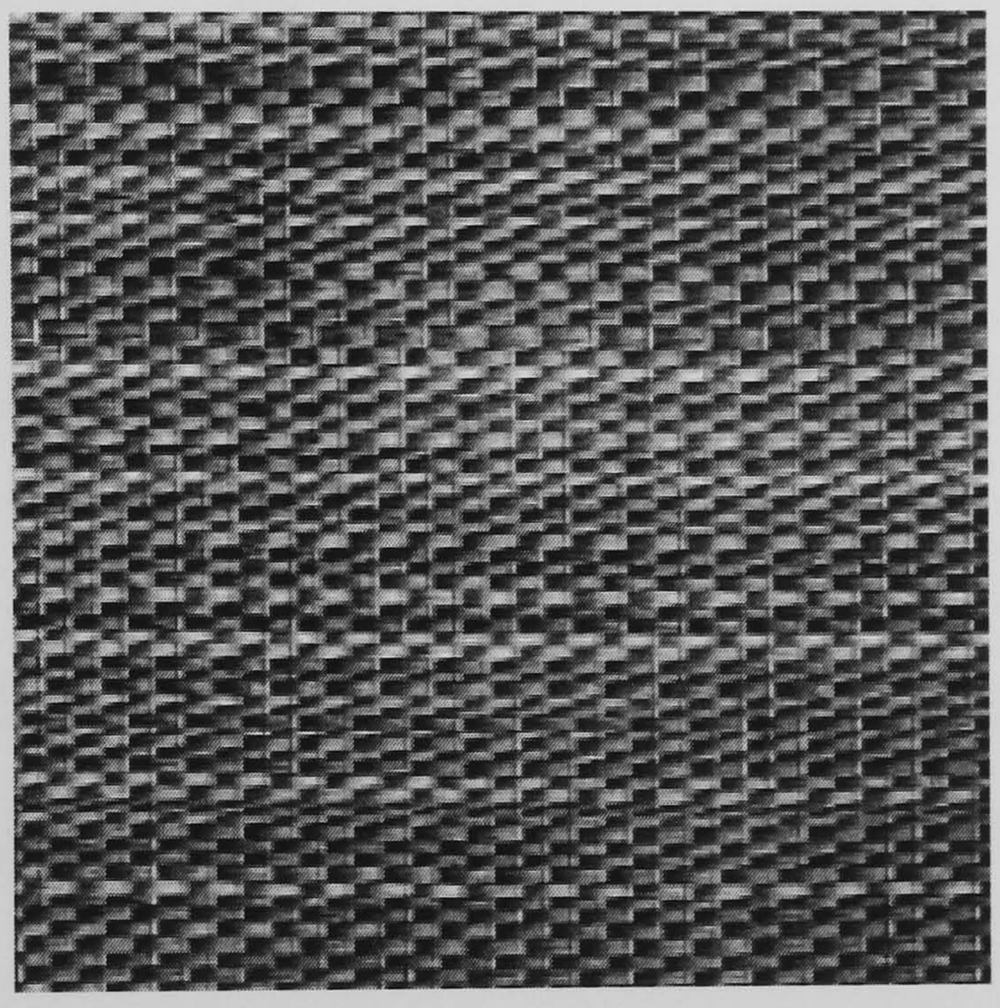


Figure B.1 D55

If the cross-correlation algorithm is applied to Figure B.1, the periodic elements within the image become visible via correlation coefficients. A 16-element mask produces the results in Figure B.2 and Figure B.3 applied in horizontal and vertical directions.

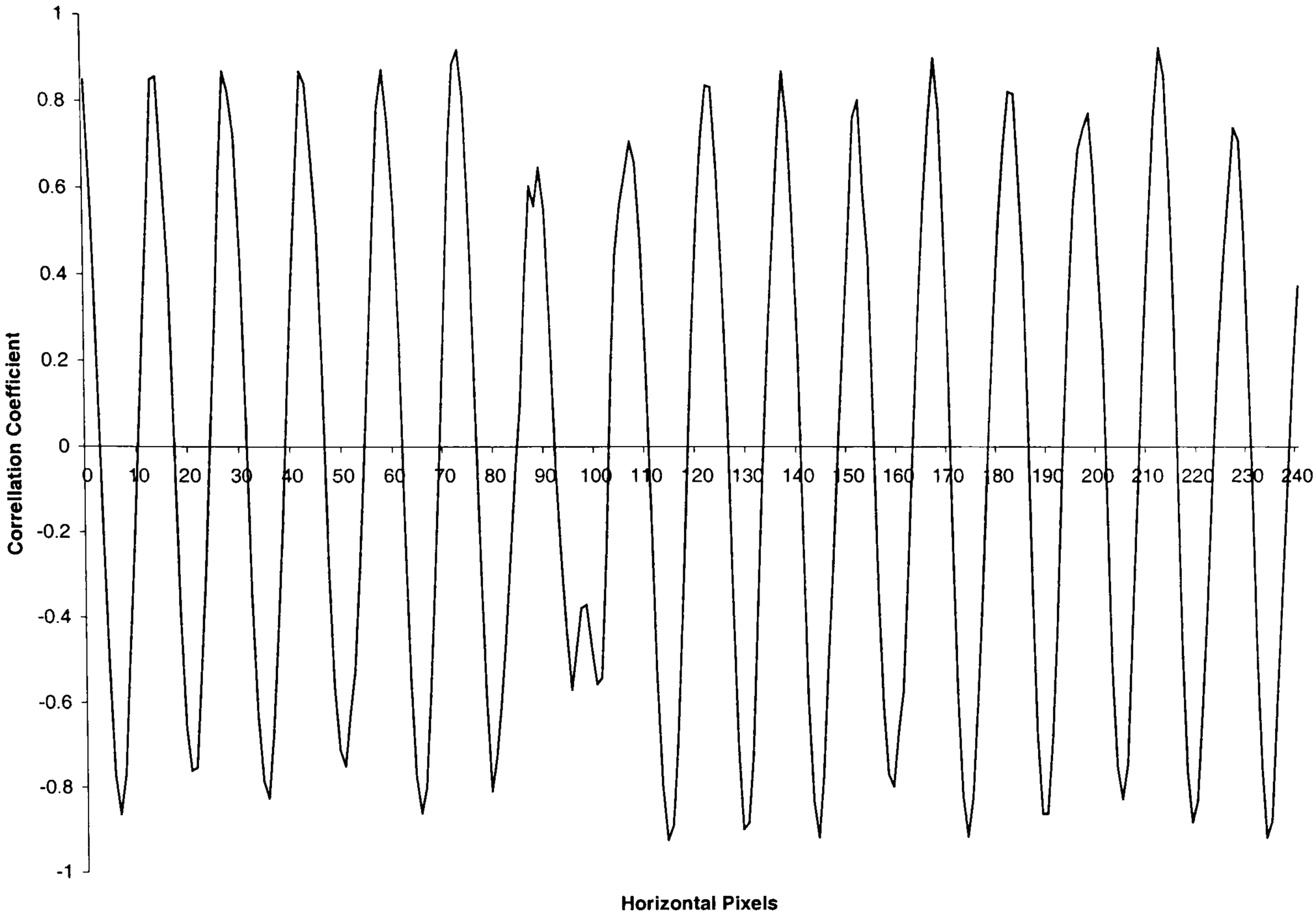


Figure B.2 Correlation Coefficient in the Horizontal Plane

Figure B.2 gives an indication of a sample match of every 13 or 14 pixels, setting a threshold value of 0.6 would ensure the segmentation of this particular texture.

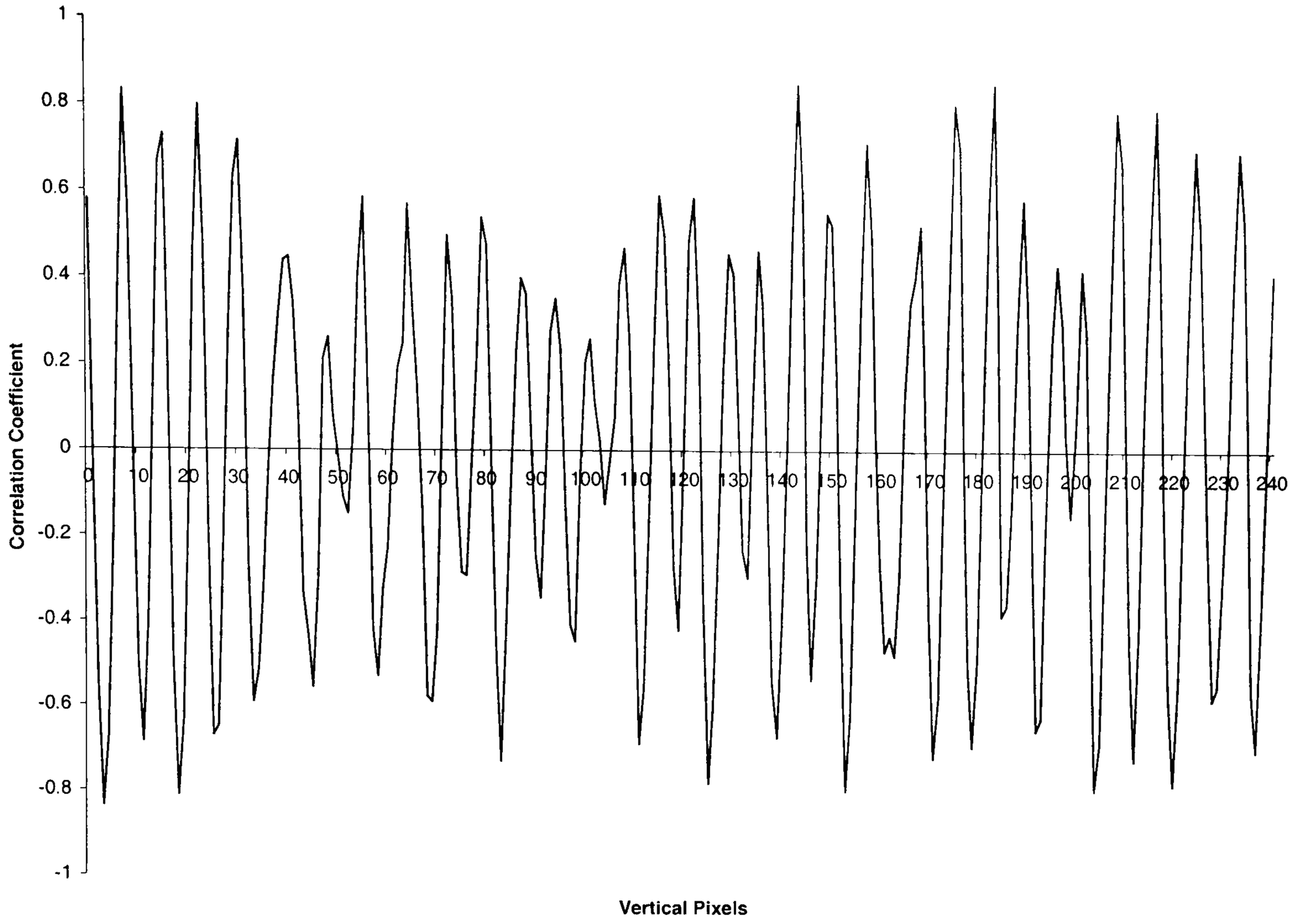


Figure B.3 Correlation Coefficient in the Vertical Plane

Figure B.3 gives an indication of sample match of every 7 pixels. However in this case setting a threshold value of 0.6 would not ensure the complete segmentation of this particular texture. To capture the entire texture, a lower threshold value would have to be used.

The setting of the threshold limit is a severe limitation of the cross-correlation process. This value will differ from texture to texture and from different sampling planes, depending upon the strength of the texture, e.g. deterministic versus structured textures.

When considering real world images such as Aerial Image 1 Figure 3.11, the size of the sample mask and the sampling point for the sample mask become critical. When constructing Figure 3.23 numerous mask sizes were tested in constructing the sample mask. The whole process was considered to be somewhat hit and miss depending on the type of texture to segmented out. A large mask in the order of 30 pixels produced the best results for tree segmentation.

Appendix C Spatial Grey Level Dependence Matrix / Back-Propagation Neural Network

Although the theory of spatial level dependence matrices appears to relatively straightforward, the actual implementation of them in software is far from easy. There are a number of variables to be considered when producing the matrices such as :-

1) Which are the best functions (energy, contrast) to use on the matrices?

2) What thresholds should be applied to the functions?

3) Should more than one function be applied?

These considerations can be removed with one stroke by employing a neural network to act as a classifier operating upon the features extracted from the spatial grey level dependence matrices. Muhamad and Deravi [Muhamad and Deravi 1993] successfully demonstrated this approach by classifying Brodatz textures by placing a back-propagation neural network upon spatial grey level dependence matrices. Fukue et al [Fukue 1998] also use the same method to classify Landsat imagery. The spatial grey level dependence matrices generate features from the image being processed and the neural network acts as a classifier, Figure C.1.

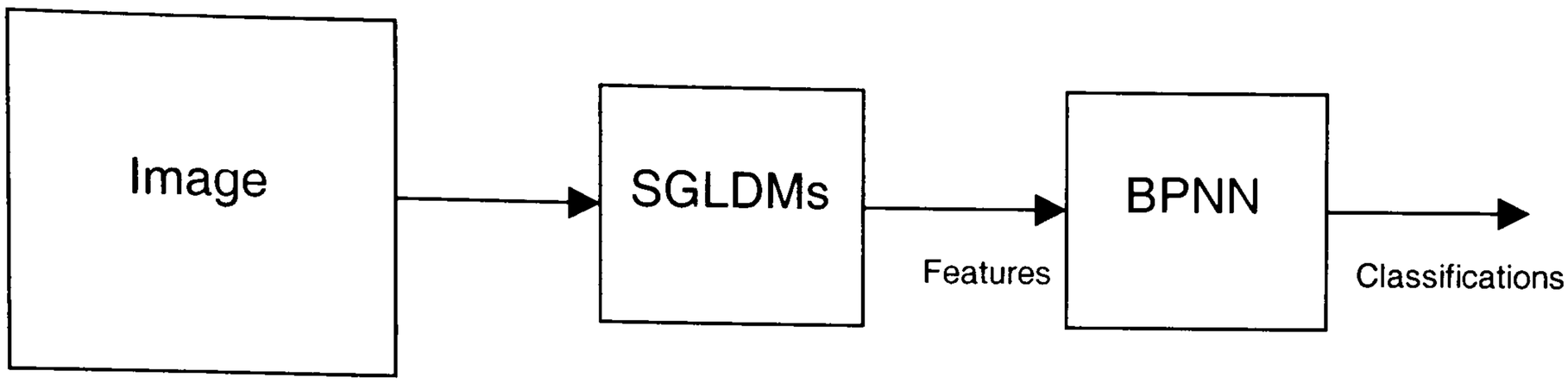


Figure C.1 SGLDM/BPNN Classifier

The training process employed takes the form of two stages:-

i) Generate spatial grey level dependence matrices for all the sample textures in the training set over multiple distance measures and sample angles.

ii) Apply the data acquired from step one to the input of the back-propagation neural network and train the network against labels of known textures assign to each spatial grey level dependence matrix.

The system can now classify textures it encounters that have been included in its training set. The method of operation is similar to that of the hybrid neural network as shown in Section 6.3. That is instead of segmenting the image in blocks, a new image is created on a pixel by pixel basis depending upon the output of the back-propagation neural network driven by the spatial grey level dependence matrices. However before the second image is constructed, a record of the results of the matrices functions applied to the back-propagation network is recorded. As multiple passes of the image for multiple distances d and angles θ are required, a majority vote is used to determine if a pixel is to be segmented based on the networks output, Figure C.2.

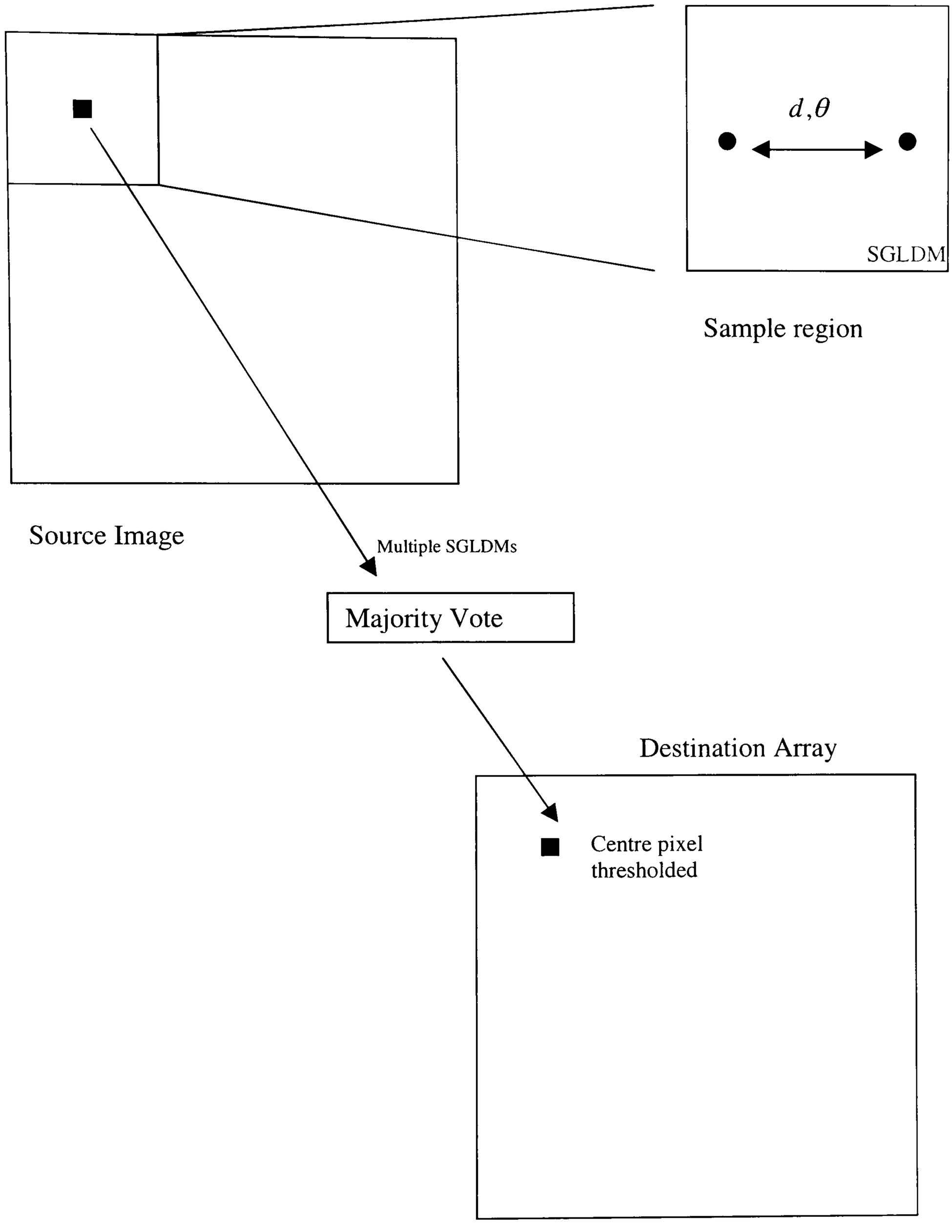


Figure C.2 SGLDM Sliding Window

Multiple spatial grey level dependence matrices are created at different displacement distance and angles as shown by Conners et al [Conners et al 1984] to try and enable the matrices to capture a wide range of textures. The values used are 1,2,4,6,8 and 12 for distances of d and 0,45,90 and 135 degree angles for θ. The images are quantized down to contain 8 grey scales from 256 to give the matrices a dimension of 8 by 8 elements. This is done to allow this benchmark architecture to possess similar dimensions to that of the hybrid neural network to give meaningful comparisons of the two methodologies.

After a result has been determined for the spatial grey level dependence matrix created for the sample window, the sample window is stepped across the image by one pixel in the x plane and the process is repeated. The sample window is traversed across the entire image on a pixel by pixel basis until all the x,y plane is processed.

Some key points of this method are:-

- All the data used to train it are exactly the same as used with the hybrid neural network.
- The sample size is the same.
- The dimensions are 8 by 8 SGLDM. With a back-propagation neural network having an 8 by 8 input layer, 3 by 3 middle later and a 2 by 1 output layer.
- The output layer operates in the same way as the hybrid neural network with one neuron indicating a positive match and the other indicating a negative match.

Appendix D Published Work

This appendix contains a paper illustrating the hybrid neural network architecture presented by the author of the thesis at the IEE Seventh International Conference on Image Processing and its Applications, Manchester, England, 13th – 15th July 1999.

HYBRID NEURAL NETWORK SYSTEM FOR TEXTURE ANALYSIS

M.J. Arrowsmith[1], M.R. Varley[1], P.D. Picton[2], and J.D. Heys[1]

[1] University of Central Lancashire, United Kingdom
[2] University College Northampton, United Kingdom

ABSTRACT

Texture classification and segmentation in digital images is commonly achieved using spatial grey level dependence matrices (SGLDMs), often referred to as co-occurrence matrices. This involves the computation of many matrices over a range of different spatial separations and orientations. The approach proposed in this paper uses a hybrid neural network system, consisting of a self-organising map followed by a backpropagation network, to restrict the number of SGLDMs that need to be computed. The system is trained in two phases on images with known texture content. The trained system is able to provide information, in the form of pixel spacing and orientation, on the texture content of unseen images. This information may be used to select appropriate SGLDMs for further texture classification. Experimental results are presented which demonstrate the effective performance of the system.

INTRODUCTION

In this paper, the authors propose a new neural network based system aimed at simplifying the use of spatial grey level dependence matrices (SGLDMs). SGLDMs, commonly referred to as co-occurrence matrices, are a widely used tool for texture classification and segmentation in the realm of digital image processing [1]. Their operation, however, is highly dependent on the sampling distance between pixels and the relative orientations of textures in the image. In most applications, many SGLDMs would have to be calculated over several different spatial distances between pairs of pixels, and through numerous orientations [2]. This typically introduces large computational costs, when using SGLDMs for texture analysis, if no prior information regarding spatial distances and orientation is available.

The new methodology presented in this paper reduces these computational costs by acquiring the pixel spacing relationship and orientation prior to use of the SGLDM, thus reducing the number of SGLDMs that are to be calculated. This is achieved by the use of two neural network models, the self-organising map (SOM) and the backpropagation neural network (BPNN), working in conjunction. The output of the SOM is used to construct a cumulative two-dimensional histogram, which gives a representation of the texture content of the input image. This information is presented to the BPNN, which is trained to output information on the pixel spacing and orientation of the texture. Previous work in this field has shown neural networks to be capable of classifying types of textures from co-occurrence matrices [3, 4], and has also applied SOMs to classify Gabor texture features [5, 6].

Promising results have been achieved using this approach. The trained system is able to correctly identify texture parameters in all images in its training set. Additionally, successful identification of textures in previously unseen images (i.e. images not included in the training set) has been achieved. The paper presents results from a system trained using synthetic texture images, and images with real-world textures from the Brodatz series [7].

SYSTEM ARCHITECTURE AND TRAINING

The hybrid system comprises two neural network models:

- A self-organising map (SOM) with a 9 x 9 array of neurons fed from 17 x 17 inputs obtained from a region of the image. The SOM outputs are used to form a 9 x 9 cumulative histogram representing the input image.

- A backpropagation neural network (BPNN) with 81 input nodes fed from the 9 x 9 histogram, 49 hidden neurons and an 14 x 4 output layer.

The sizes of the two networks have been determined by trial and error following several tests involving one-dimensional and two-dimensional structures.

A block diagram of the system is shown in Fig. 1.

Training of the hybrid system is a two-step process which uses a set of images containing different textures, with the pixels that comprise these textures having different orientations and spatial relationships.

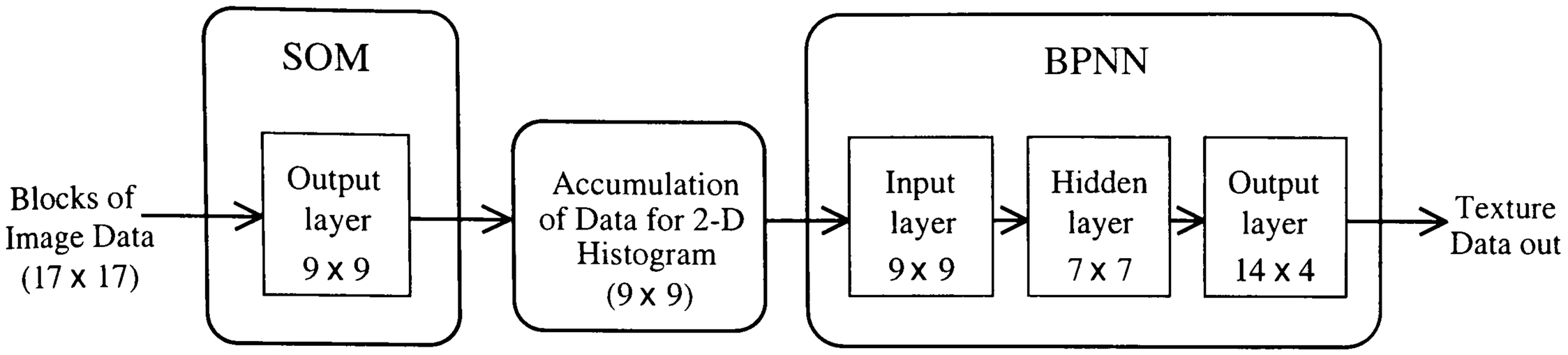


Fig. 1 Hybrid Neural Network System for Texture Analysis

During the first step, a two-dimensional 17 x 17 mask is used to map pixel grey levels, via a linear scaling function (converting grey scales of 0 through 255 to 0 through 1.0), onto the output layer of the SOM, the interconnecting weights of which have previously been assigned random values in the range -1.0 to 1.0.

The processing of image data, in blocks of 17 x 17 pixels to the SOM output layer is depicted in Fig. 2.

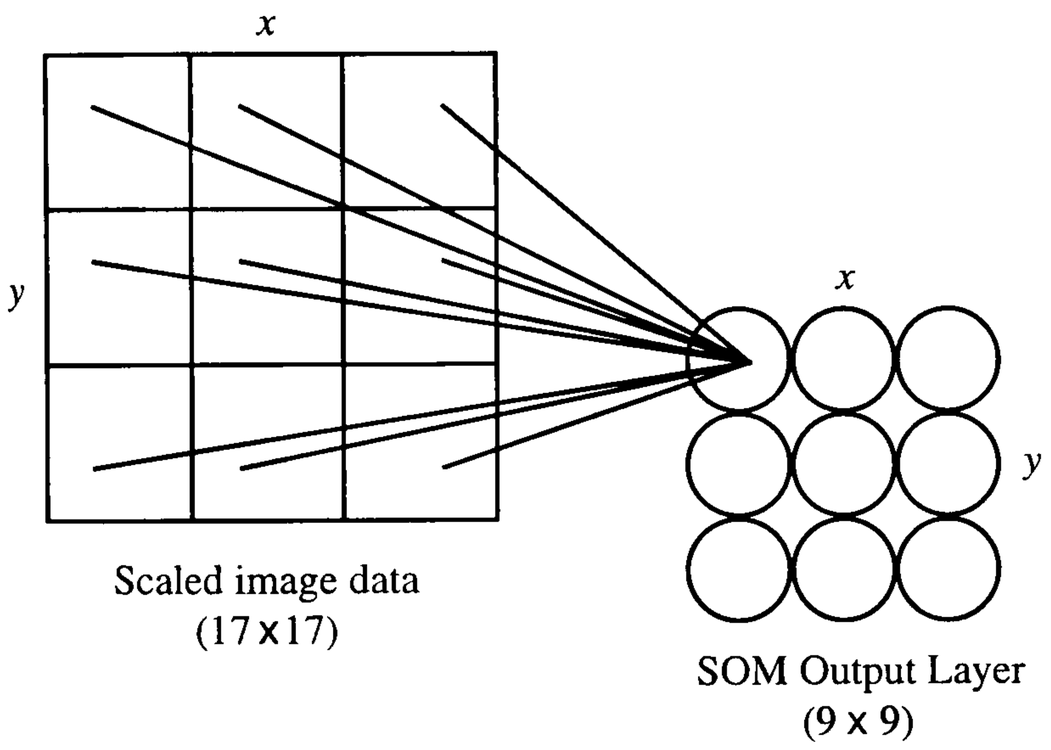


Fig. 2 Presentation of Image Data to SOM

Note that in Fig. 2, for clarity only the first neuron (top left) is shown as being fully connected to the input data held in the 17 x 17 mask - in reality all the neurons are fully connected to all possible inputs.

The training of the SOM [8] takes the form of evaluating the distance d_k, at a point in time t, between the weights w_{ik} of neuron k and the applied input values I_i to neuron k, with the inputs taken over all positions on the x and y axes.

The distance d_k for neuron k is calculated using

$$d_k = \sum_x \sum_y \left(I_i(t) - w_{ik}(t) \right)^2$$

This distance d_k is evaluated for each of the 81 neurons in the SOM (i.e. for all values of k), and the winning neuron j is then identified by finding the neuron which has the smallest difference value d_j, i.e.

$$d_j < d_k \qquad \text{for all } k,\ k \neq j$$

After the winning neuron j is found, its weights are updated according to the equation

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(I_i(t) - w_{ij}(t) \right)$$

where $\eta(t)$ is the learning rate at a specified moment in time t. The learning rate gradually decreases in value over time as training of the SOM progresses. The weights of neurons situated in the neighbourhood of the winning neuron j are also updated. Fractional updates are applied, i.e.

$$w_{ik}(t+1) = w_{ik}(t) + \phi_k \eta(t)\left(I_i(t) - w_{ik}(t) \right)$$

where the scaling factor ϕ_k $(0 < \phi_k < 1)$ is determined according to the position of the neighbourhood neuron in relation to the winning neuron. The neighbourhood used in this application is square and centred on the winning neuron.

After this training step has been taken, a new input is presented to the SOM via the 17 x 17 mask, until all pixels in the training images have been processed. As training progresses, the size of the neighbourhood is gradually decreased after each pass through all the images in the training set, until the neighbourhood is only one pixel wide for the final pass.

The values representing the interconnecting weights between the neurons are saved, and the intermediate data between the two neural networks is calculated. This is achieved by running the SOM over an image in the training set and constructing a 9 x 9 two-dimensional cumulative histogram, which gives a numerical representation of the most active neurons within the

SOM for that image. The histograms are linearly scaled such that the values lie in the range 0 to 1.0.

Each image now has its own unique identity via the contents of its own two-dimensional histogram at the output of the SOM. An example of a scaled 9 x 9 two-dimensional cumulative histogram is shown in Fig. 3.

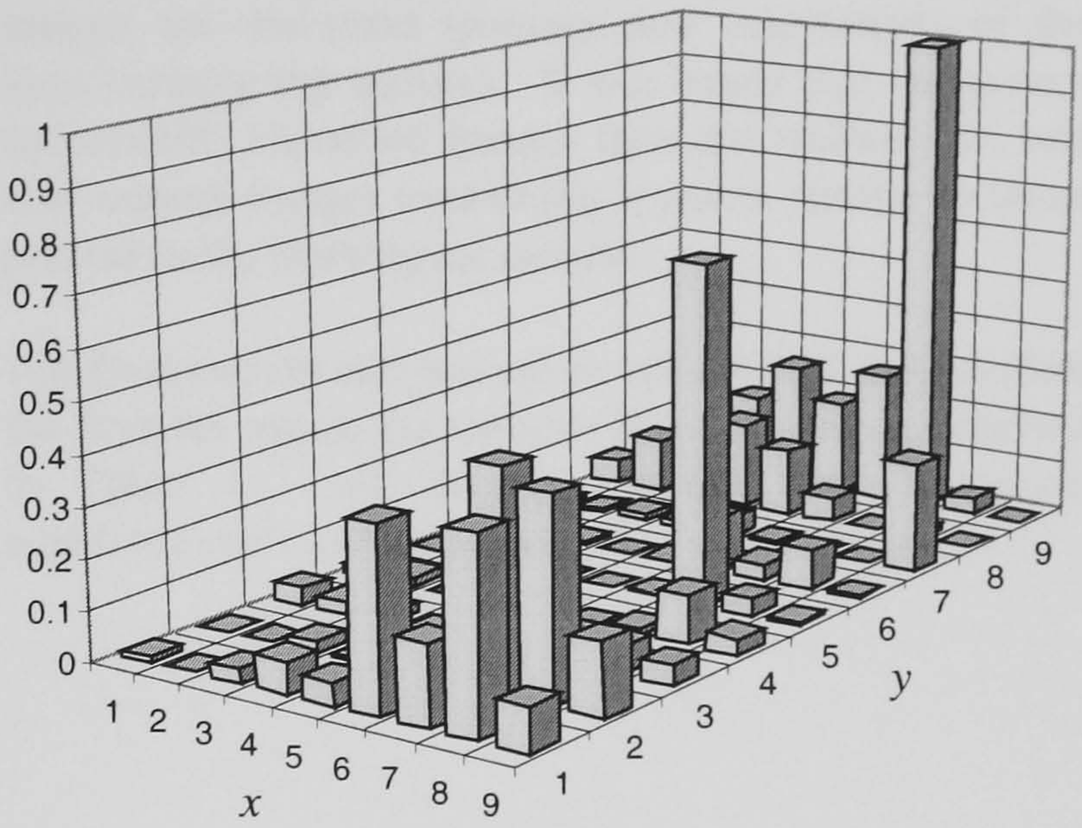


Fig. 3 Scaled 9 x 9 Two-Dimensional Cumulative Histogram obtained from SOM Output Layer

The values in the 9 x 9 histogram have a one-to-one mapping to the input layer of the BPNN. The second step of the training process is to train the BPNN to classify the histogram. For each histogram the known pixel spatial relationship, obtained using the autocorrelation function [9], and orientation, obtained by visual inspection, of the corresponding training image are presented as a target output. The backpropagation training algorithm [10] is used to train the BPNN to classify histograms according to different pixel separations from 2 pixels to 15 pixels (corresponding to 14 columns in the output layer) and different texture orientations (0°, 90°, 45° and 135°, corresponding to 4 rows in the output layer). Again, after training is complete, the weights between the interconnecting neurons of the BPNN are saved for the run-time application of this model.

TEXTURE ANALYSIS USING THE TRAINED SYSTEM

The run-time mode of operation is to select an area from an image that is to be classified using an SGLDM. This selected area is then applied to the SOM input layer in groups of 17 x 17 pixels, and a cumulative 2-D histogram is generated which represents that area. The scaled histogram is then presented to the BPNN, which provides an indication of the most likely pixel separation and orientation that forms the texture in the image. This data can be used to select the most appropriate SGLDMs for subsequent texture classification. It should be noted that no training or updating of the weights in either network takes place during this run-time process.

The system was initially trained using 64 x 64 images containing synthetic textures and superimposed random noise. The run-time set of images for testing this system also consists of lines superimposed with random noise; however the lines are of a different grey scale value. The only common components between the two sets of images are the pixel spacings and orientations of the lines forming the textures. It was found that the system successfully classified images from the training set, and also unseen images containing textures similar to those present in the training set images.

The final system was trained on ten 64 x 64 images from the Brodatz series, the textures in which are summarised in Table 1. All Brodatz images were histogram equalised prior to training and testing the system.

EXPERIMENTAL RESULTS

Name	Texture Content	Pixel spacing at 0° (pixels)	Pixel spacing at 90° (pixels)
D6	Woven Aluminium Wire	7	9
D14	Woven Aluminium Wire	5	6
D20	French Canvas x4	5	10
D21	French Canvas ½ size	9	9
D53	Oriental Straw Cloth	12	4
D55	Straw Matting	7	9
D78	Oriental Straw Cloth	6	7
D84	Raffia	8	8
D104	Loose Burlap	8	8
D106	Cheesecloth	4	4

Table 1 Texture classifications of images from the Brodatz series [7] used for training and testing the system

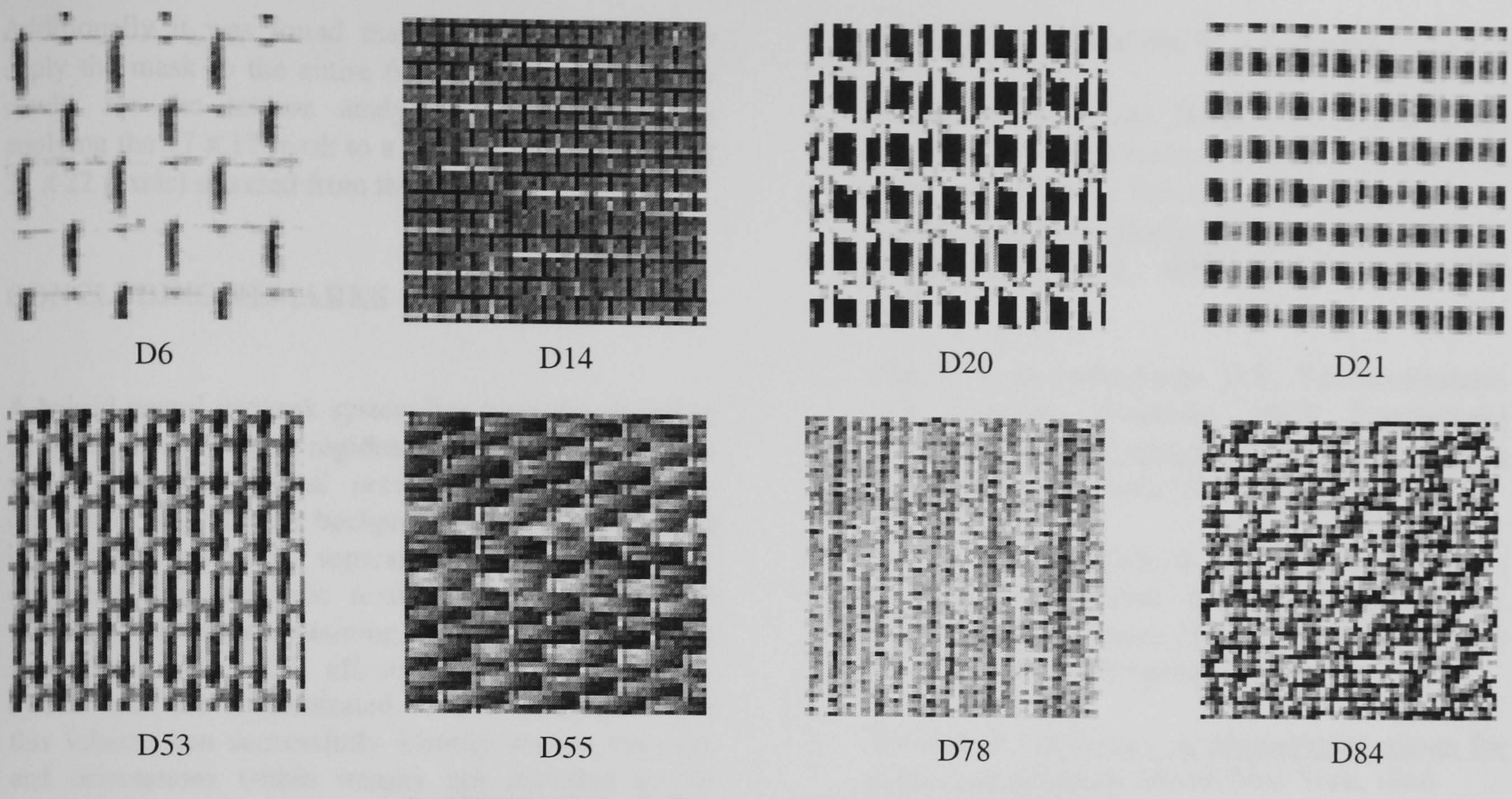


Fig. 4 Brodatz Images used as the Training Set

For each type of texture, two images were created by sampling the original Brodatz image in different locations. One image was used as part of the training set for the system, and the other was used to test the network in its run-time mode. The images used in the training set are shown in Fig. 4.

The system was trained using the following parameters:

- The SOM used an initial neighbourhood size of eight neurons and an initial learning rate $\eta(0)$ of 0.1, with each image being presented 10 times. The input layer to the SOM was moved across the image on a pixel-by-pixel basis.

- The scaled cumulative two-dimensional histograms produced were applied to the BPNN. The error backpropagation training algorithm was used with a learning rate of 0.3, and the network was trained for 10000 cycles. The error decreases to give 100% classification.

The run time images were applied to the network to test the effectiveness of training. Again, these were all correctly classified.

The data in Fig. 5 shows sample results from the output layer of the BPNN for test image D55, for which the periodicity is correctly determined to be every 7 pixels at an orientation of 0°, and 9 pixels at 90° orientation. There is no significant texture content for orientations of 45° and 135°.

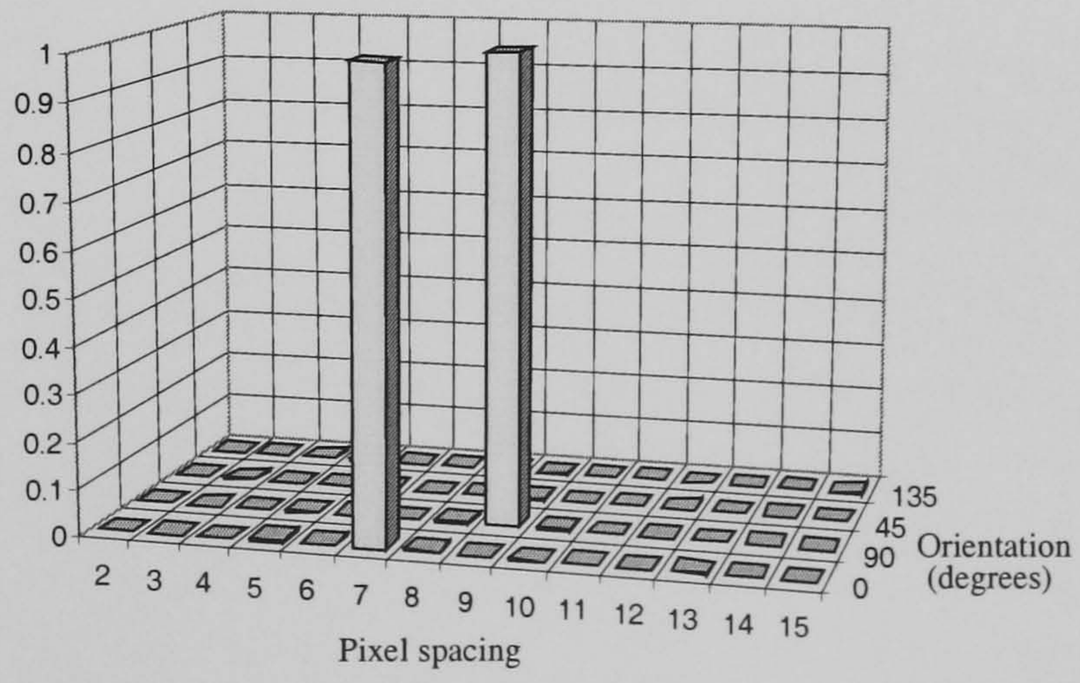


Fig. 5 System Output for Brodatz Test Image D55

Additionally it was found that it was unnecessary to apply the mask to the entire 64 x 64 image. Accurate results for the texture analysis were achieved by applying the 17 x 17 mask to a smaller region (typically 22 x 22 pixels) selected from the image.

CONCLUDING REMARKS

A hybrid neural network system has been developed to analyse textures within regions of images. The system incorporates two neural network models: the self-organising map and the backpropagation network. The system has been trained separately on two training sets: one containing synthetic textures with superimposed noise, and the other containing real-world textures from Brodatz images. The effective performance of the system has been demonstrated. It has been shown that this scheme can successfully identify texture spacings and orientations within images not included in the training set.

The hybrid neural network system is intended as a pre-processing stage, and may be used to select a reduced number of SGLDMs for subsequent texture classification, thus decreasing the computational load.

REFERENCES

1 Haralick, R.M., Statistical and Structural Approaches to Texture, Proceedings of the IEEE, vol. 67, no. 5, May 1979, pp786-804

2 Conners, R.W., Trivedi, M.M., and Harlow, C.A., Segmentation of a High-Resolution Urban Scene Using Texture Operators, Computer Vision, Graphics, and Image Processing, vol. 25, 1984, pp273-310

3Oja, E. and Valkealahti, K., Co-occurrence map: Quantizing t

4 Muhamad, A.K. and Deravi, F., Cooccurrence Matrices for Automatic Texture Classification using a Neural Network, 2nd International Conference on Automation Robotics and Computer Vision (ICARV92), ppCV-19.3.1-4, Singapore 1992

5 Ma, W.Y. and Manjunath, B.S., Texture Features and Learning Similarity, IEEE International Conference on Computer Vision and Pattern Recognition, California USA, June 1996

6 Raghu, P.P., Poongodi, R. and Yegnanarayana, B., A Combined Neural Network Approach for Texture Classification, Neural Networks, vol. 8, no. 6, pp975-987, Pergamon 1995

7 Brodatz, P, Textures – A photographic album for artists and designers, Dover, New York, 1966

8 Kohonen, T., Self-Organization and Associative Memory, Springer Verlag, Berlin, 1984

9 Low, A., Introductory Computer Vision and Image Processing, McGraw-Hill, 1991

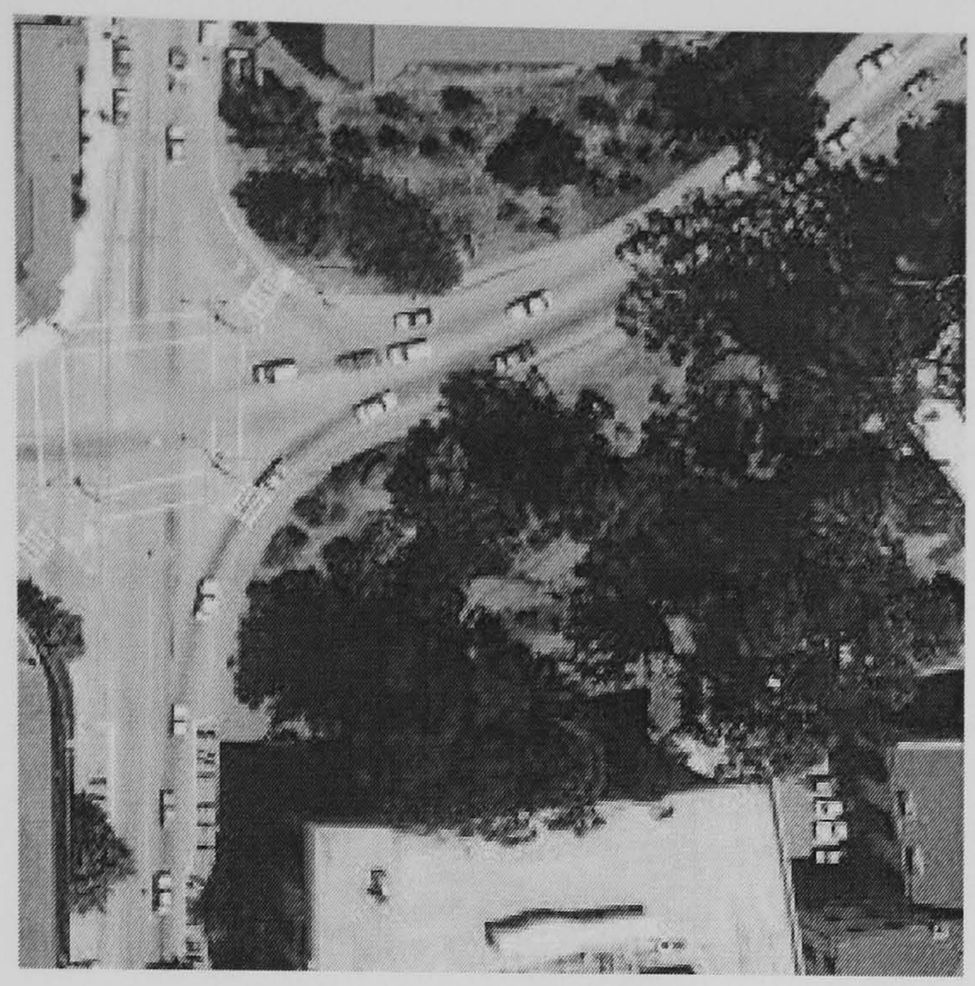
10 Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 'Learning internal representations by error propagation', in Parallel distributed processing: Explorations in the microstructure of cognition, D. E. Rumelhart and J.L. McClelland (Eds.), MIT Press, 1986

Appendix E Pictorial Aerial Image Results

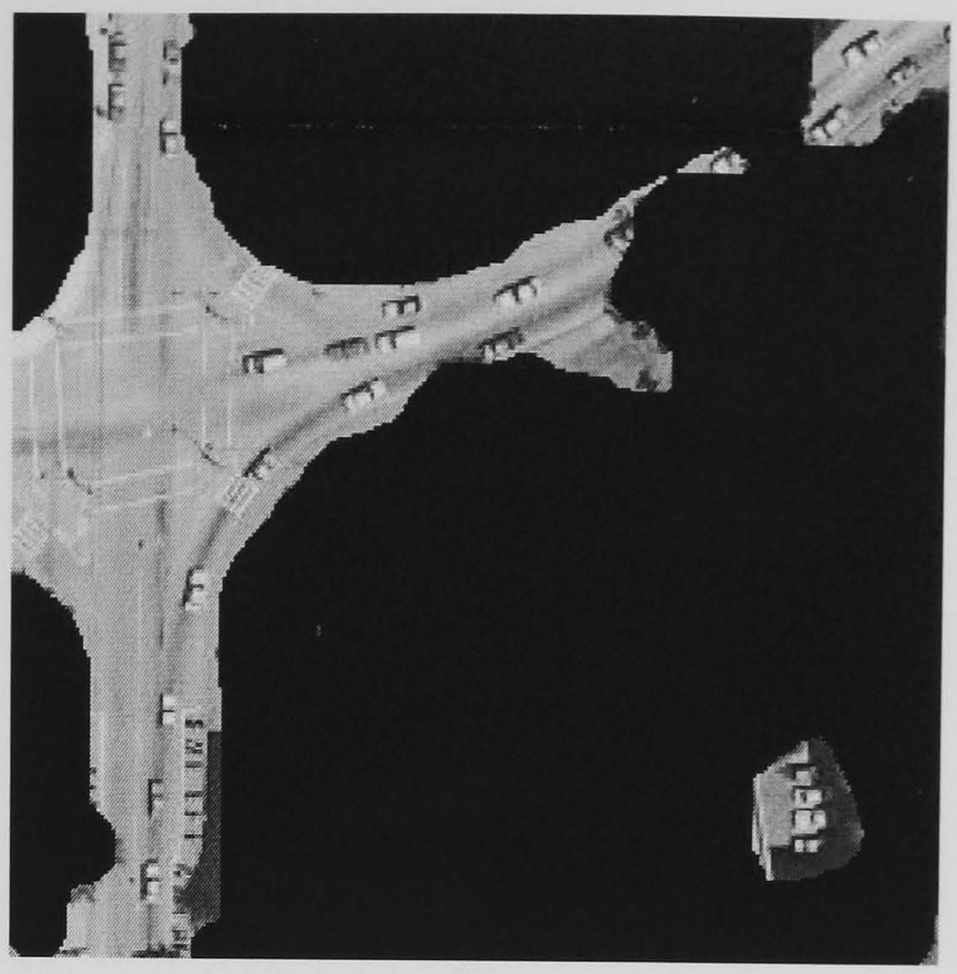
This appendix consists of the real world aerial images used to generate the results presented in chapter six. Accompanying each aerial image is a hand segmented image along with a hybrid neural network segmented image and a spatial grey level dependence matrix classified by a back-propagation neural network segmented image.

A single page is dedicated to each image. The positioning of the images takes the form of: -

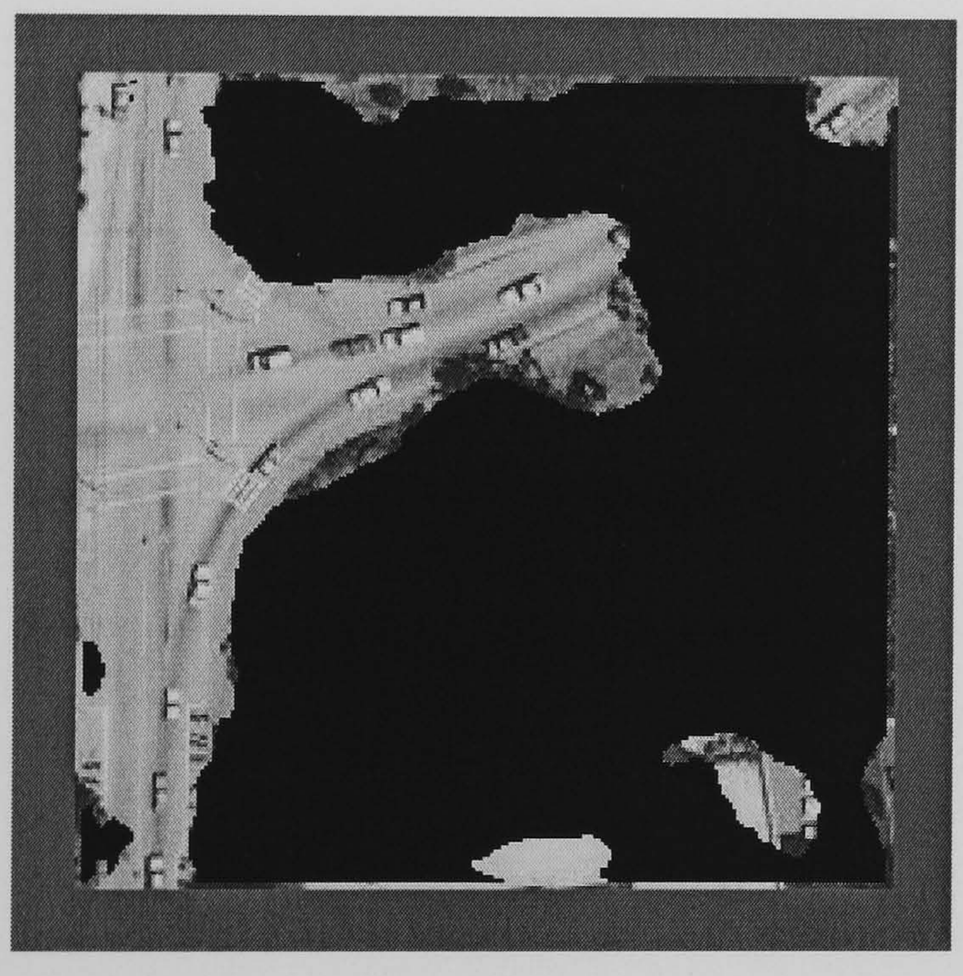
- Top left, the original aerial image.
- Top right, the hand segmented image. This image was generated by a human via a computer graphics package to indicate desired segmentation properties.
- Bottom left, hybrid neural network segment image. As documented in Chapter 5.
- Bottom right, spatial grey level dependence matrix classified by a back-propagation neural network image. As documented in Appendix C.



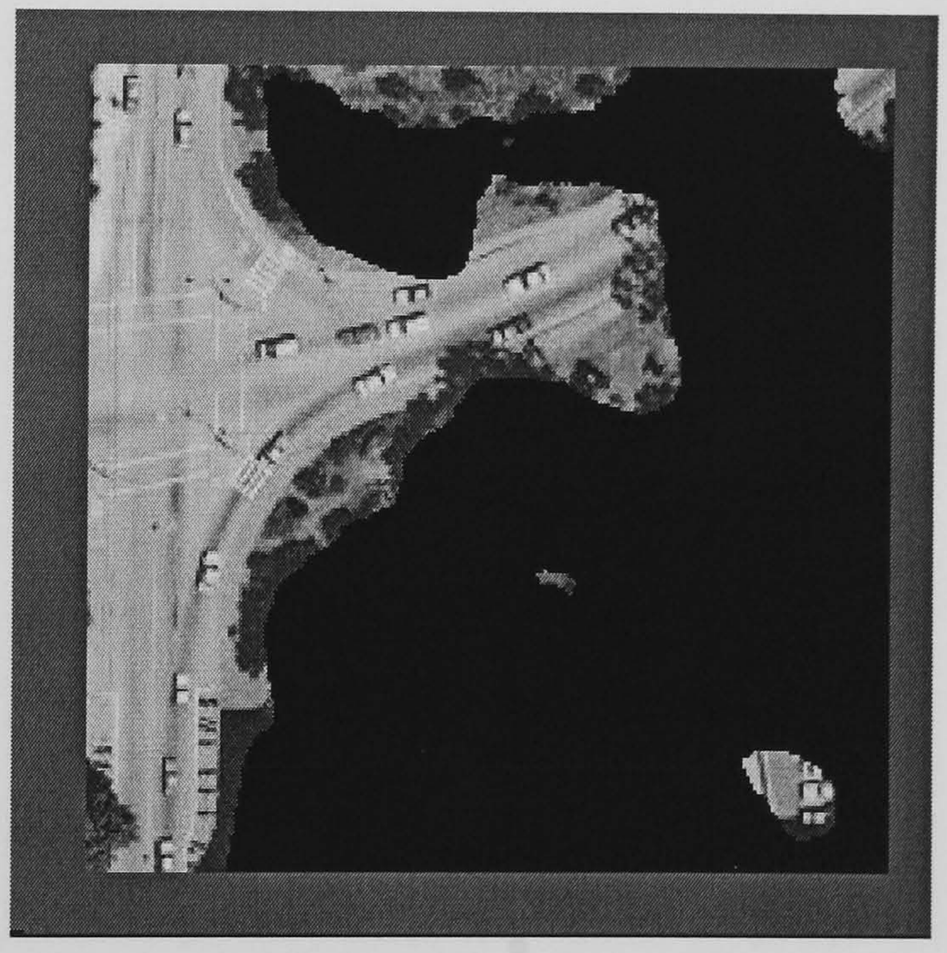
Aerial Image 3



Hand Segmented Image

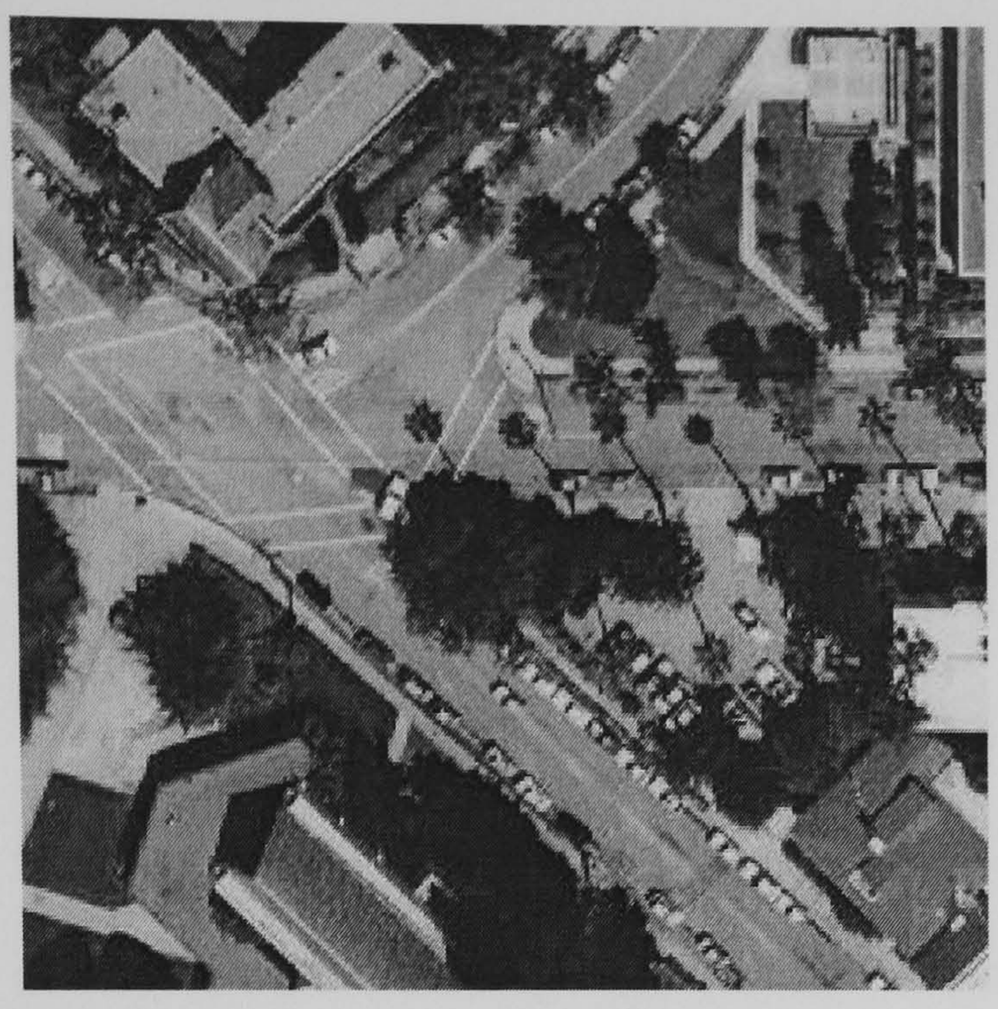


Hybrid Neural Network (91.18%)



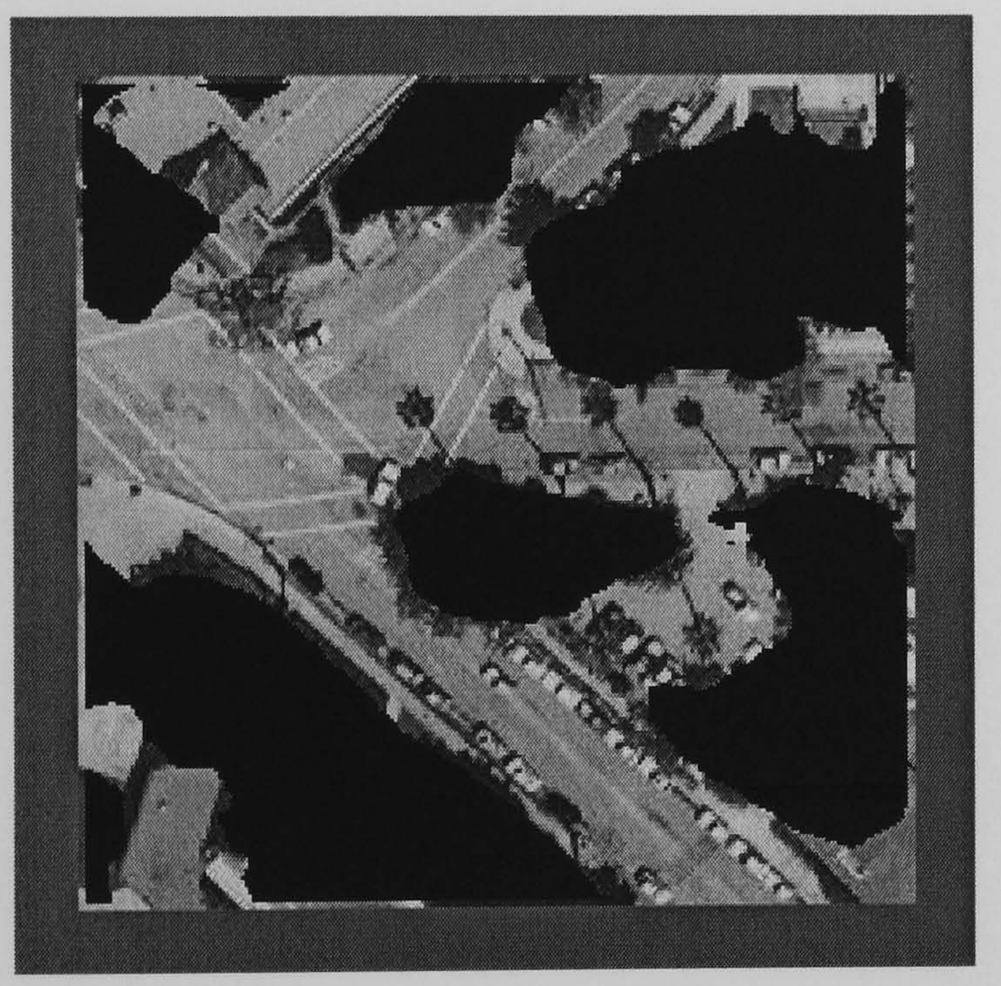
SGLDM /BPNN (89.18%)

Figure E.1 Aerial Image 3 Results

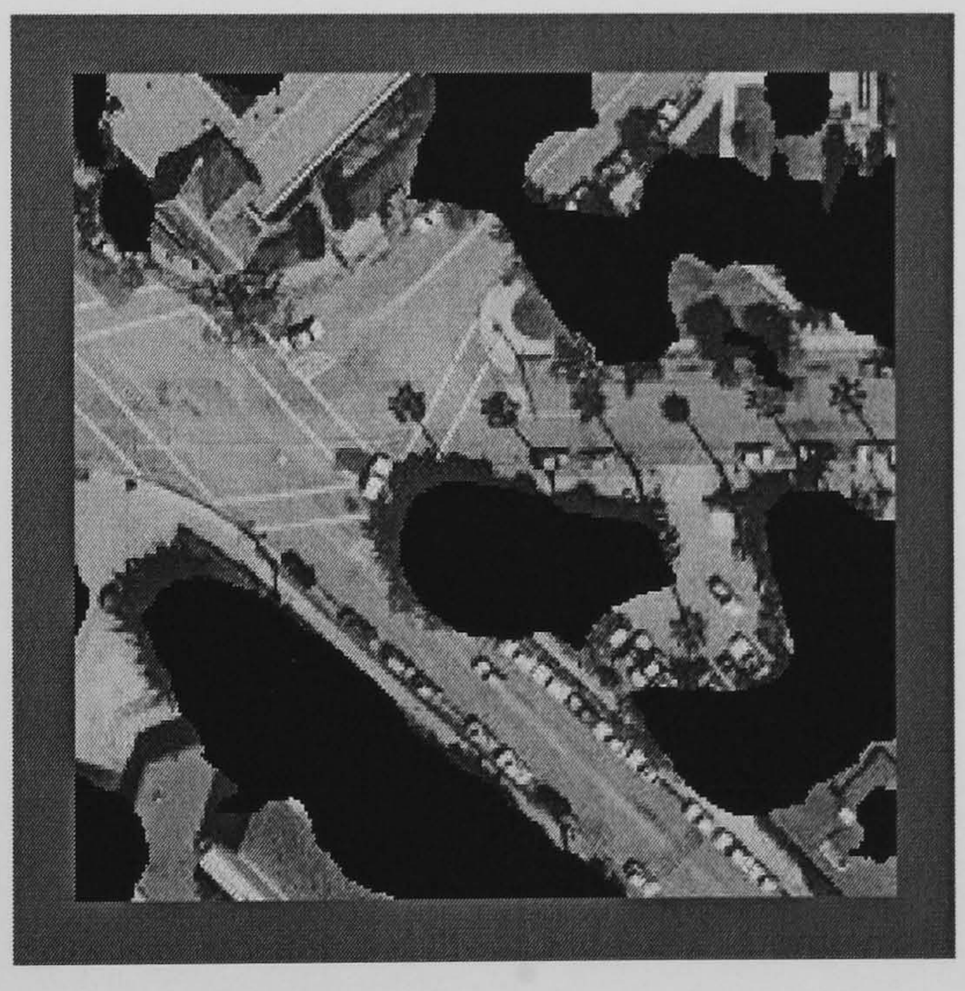


Aerial Image 4

Hand Segmented Image

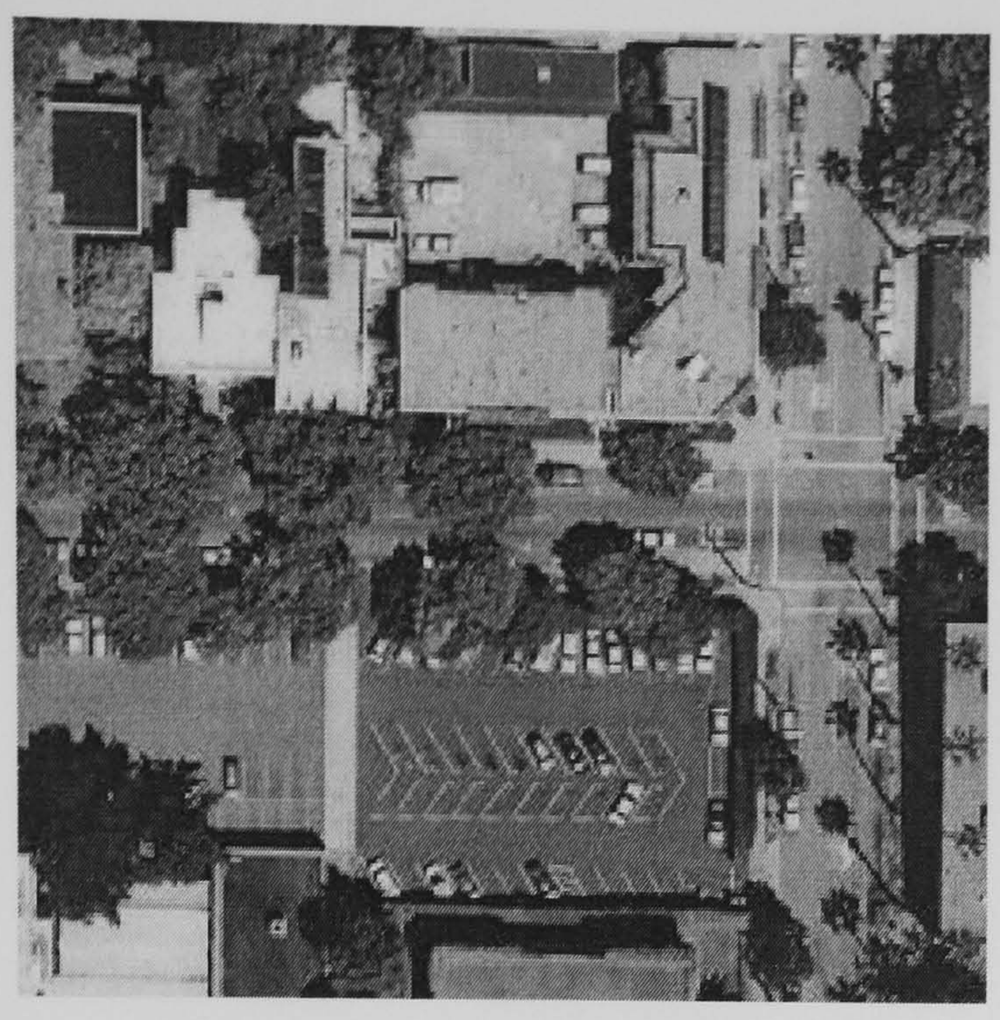


Hybrid Neural Network (83.03%)

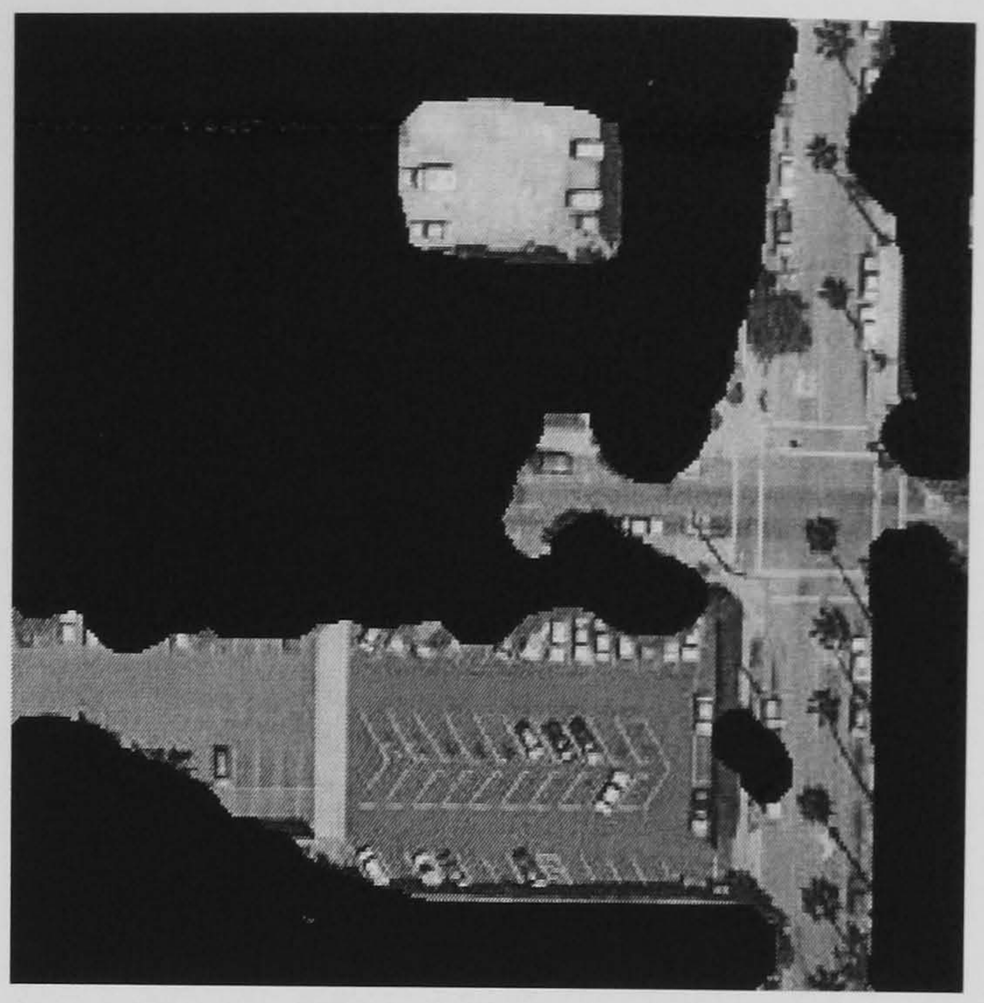


SGLDM /BPNN (77.72%)

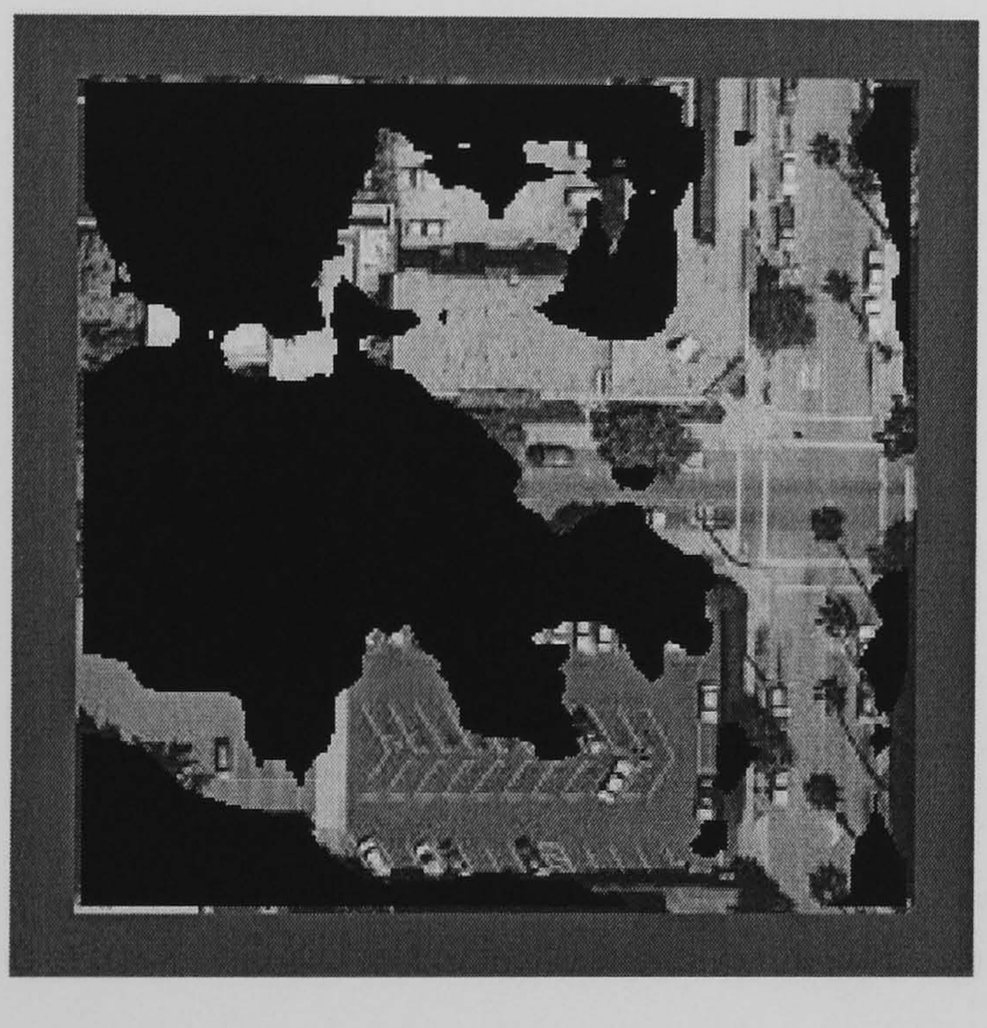
Figure E.2 Aerial Image 4 Results



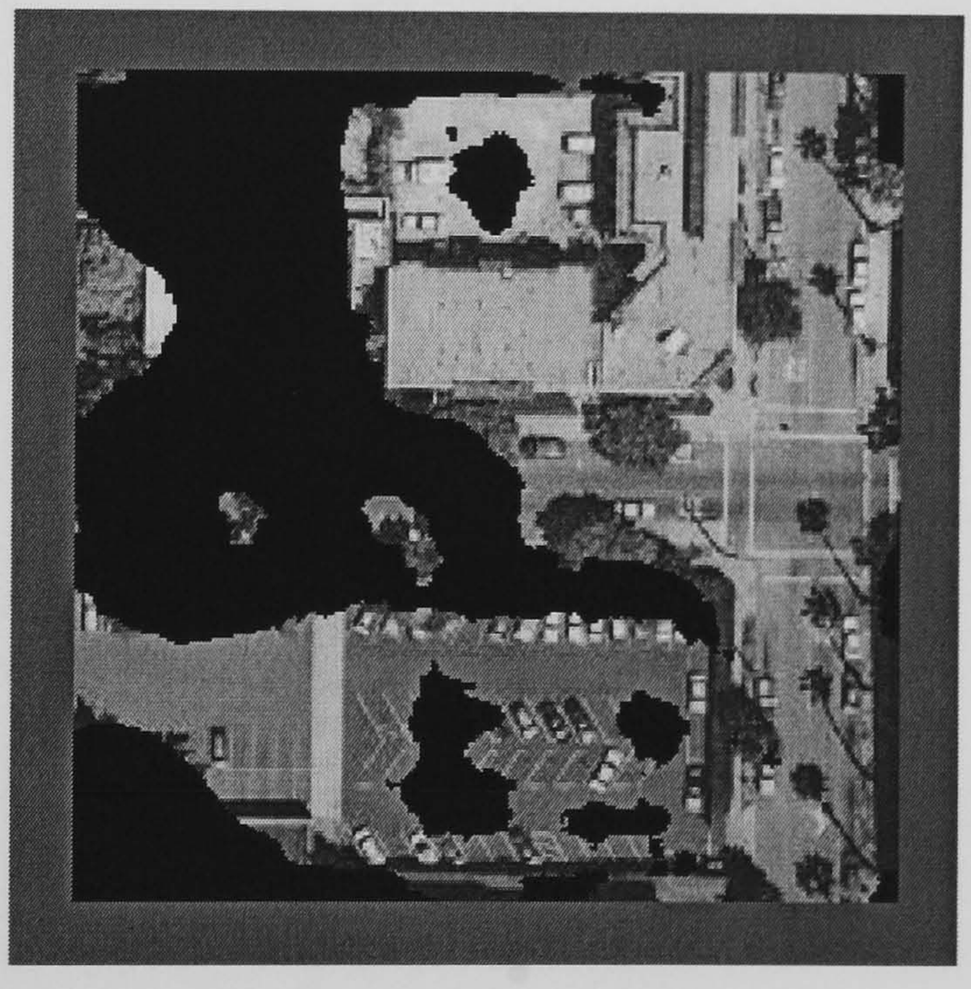
Aerial Image 5



Hand Segmented Image



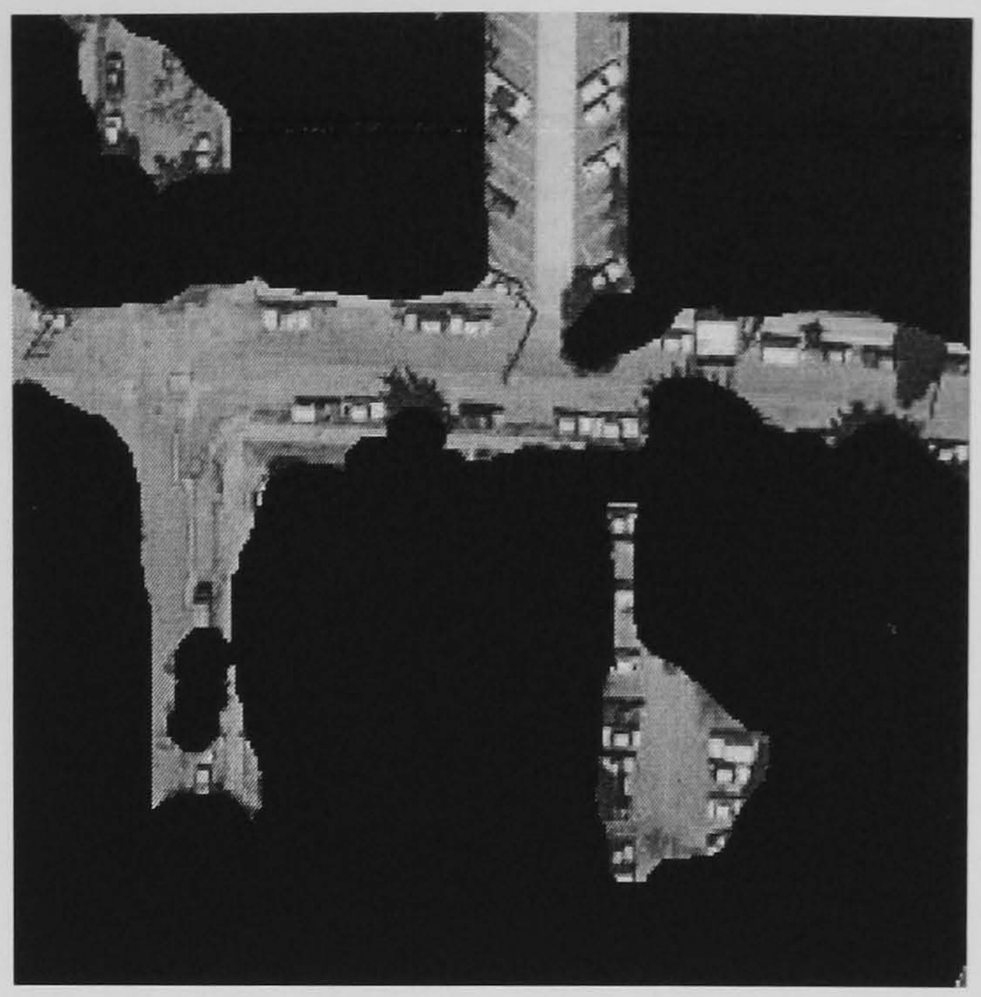
Hybrid Neural Network (78.79%)



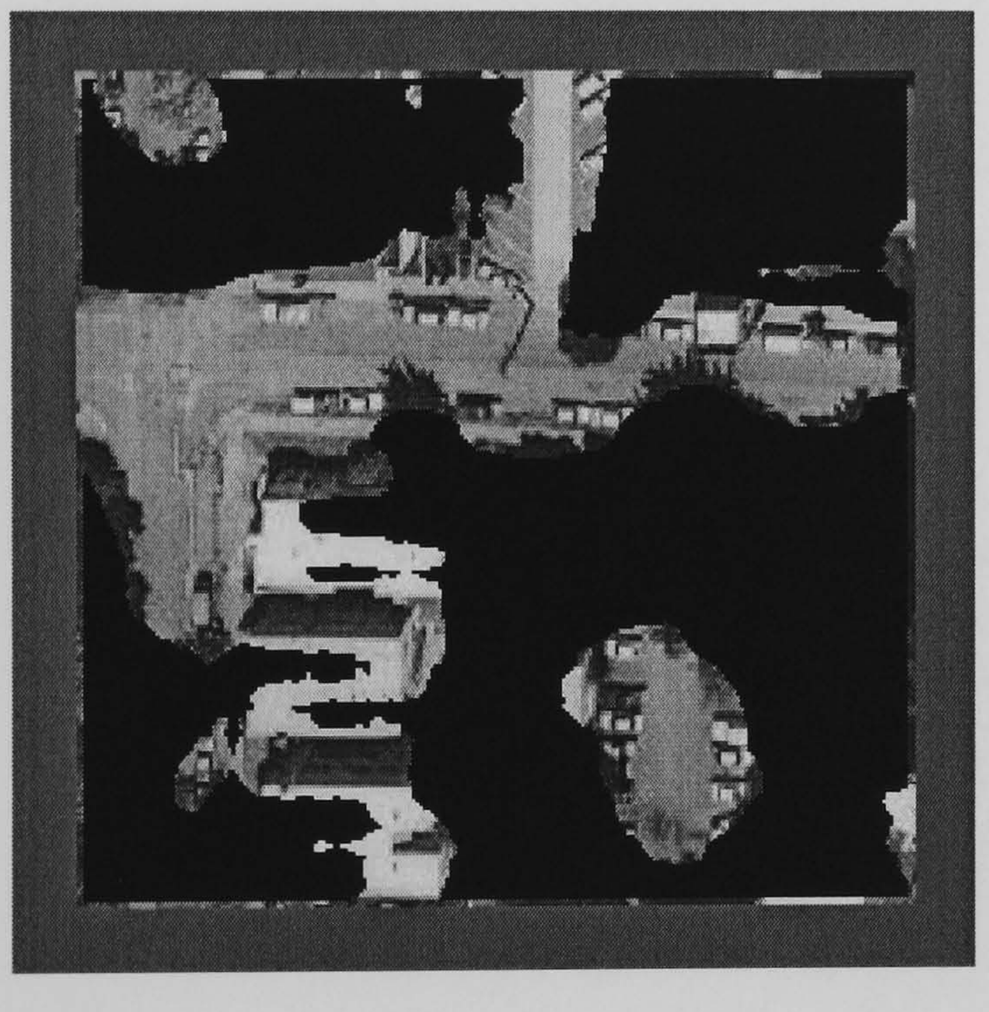
SGLDM /BPNN (78.17%)

Figure E.3 Aerial Image 5 Results

Aerial Image 6



Hand Segmented Image

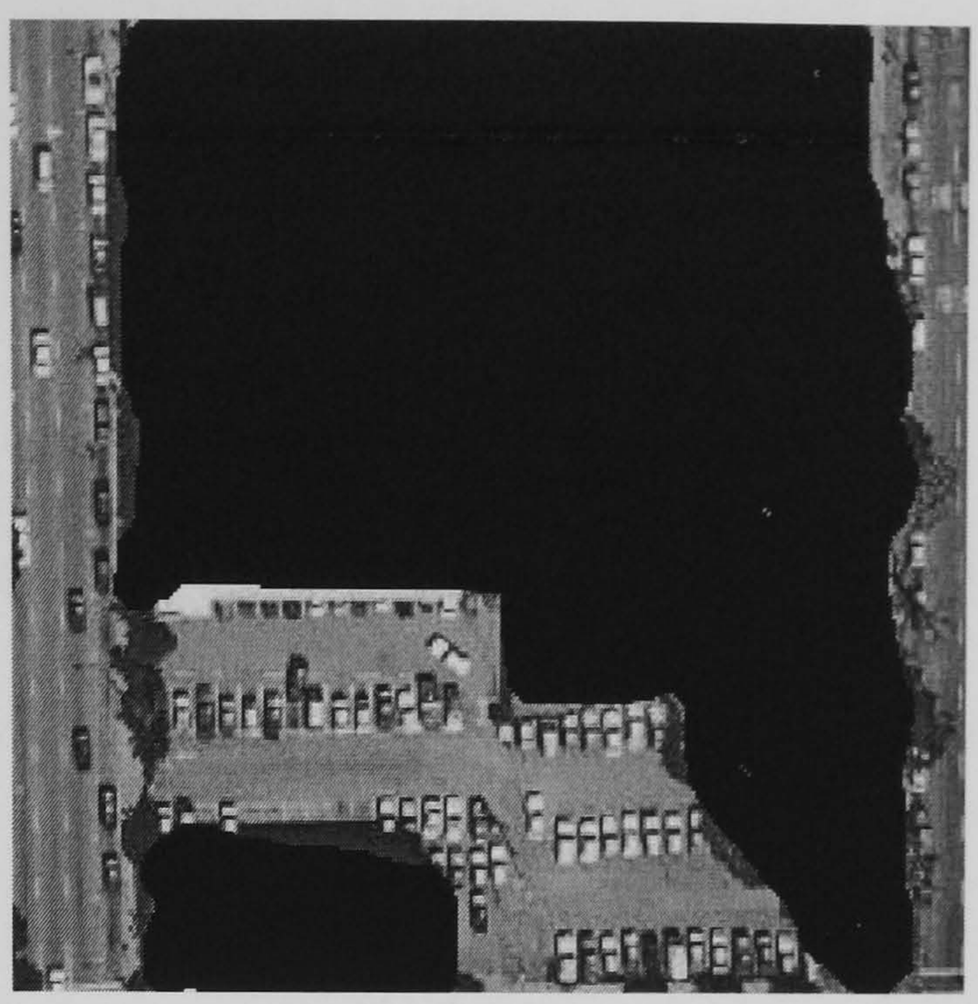


Hybrid Neural Network (84.55 %)

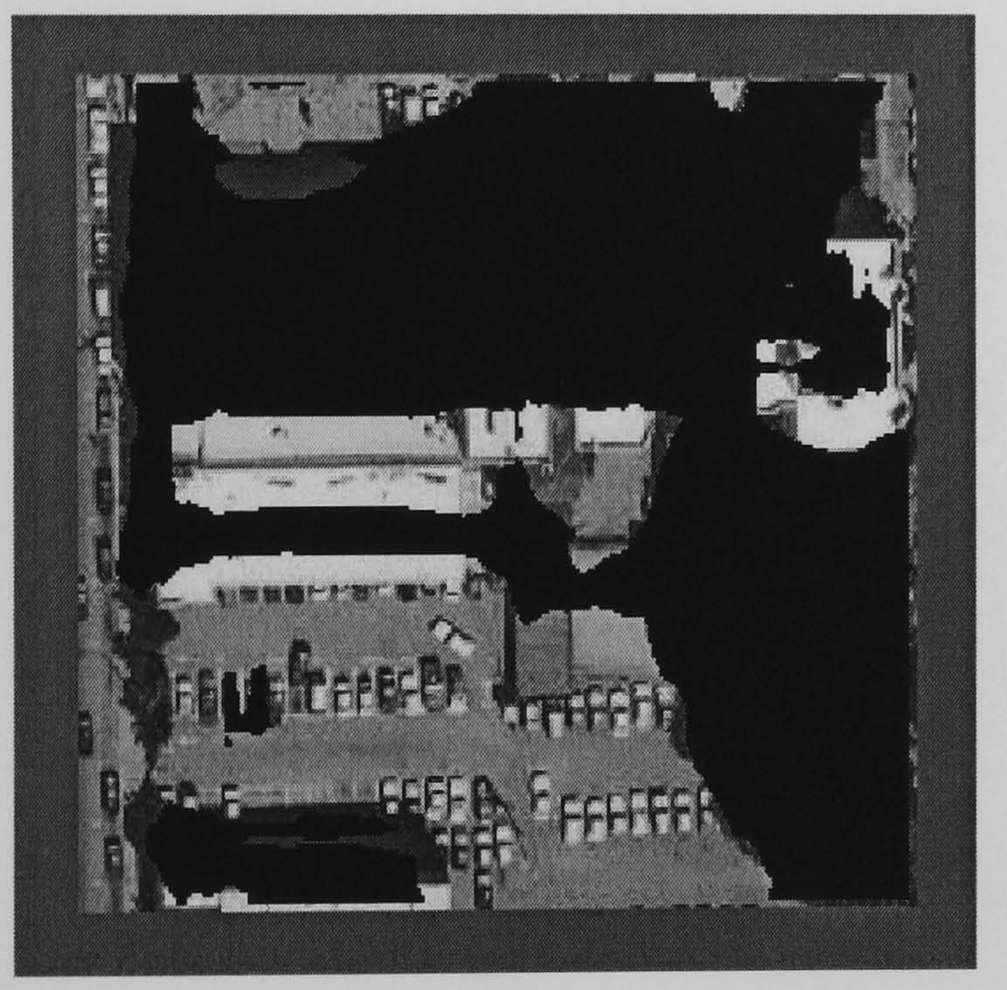
SGLDM /BPNN (83.80 %)

Figure E.4 Aerial Image 6 Results

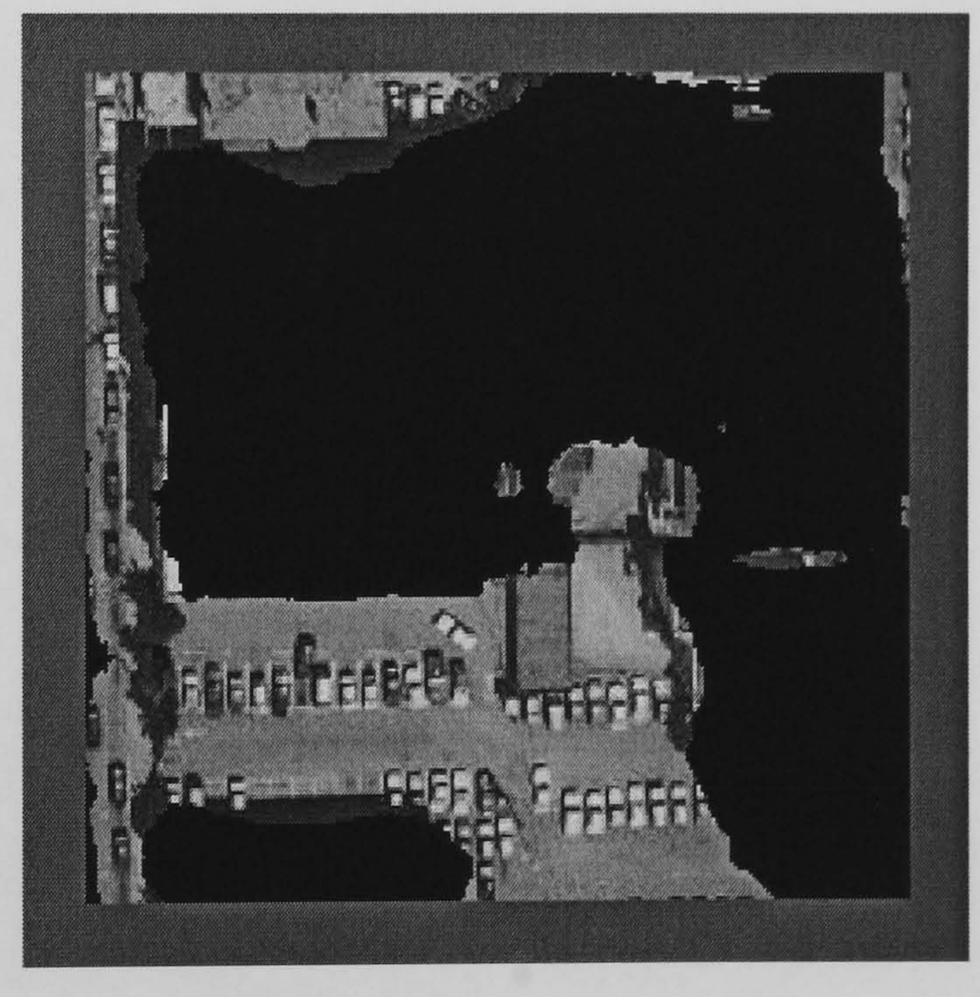
Aerial Image 7



Hand Segmented Image

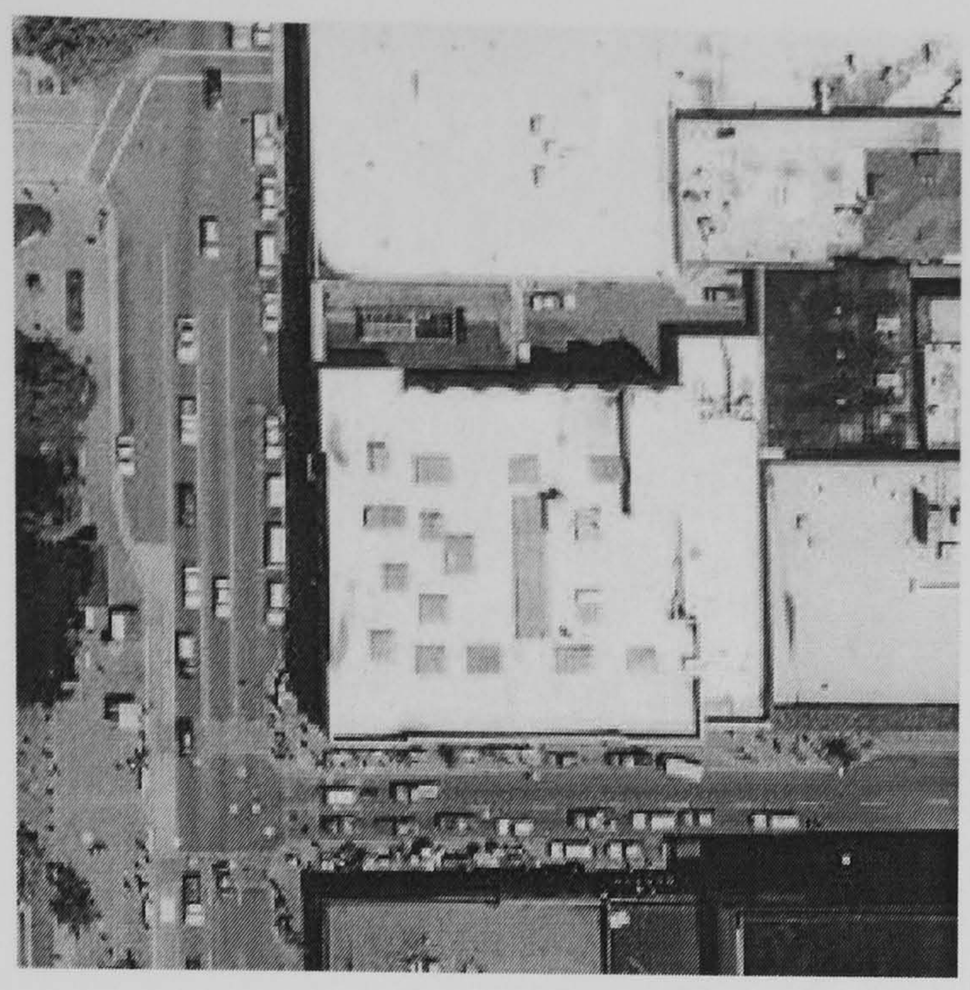


Hybrid Neural Network (85.58 %)

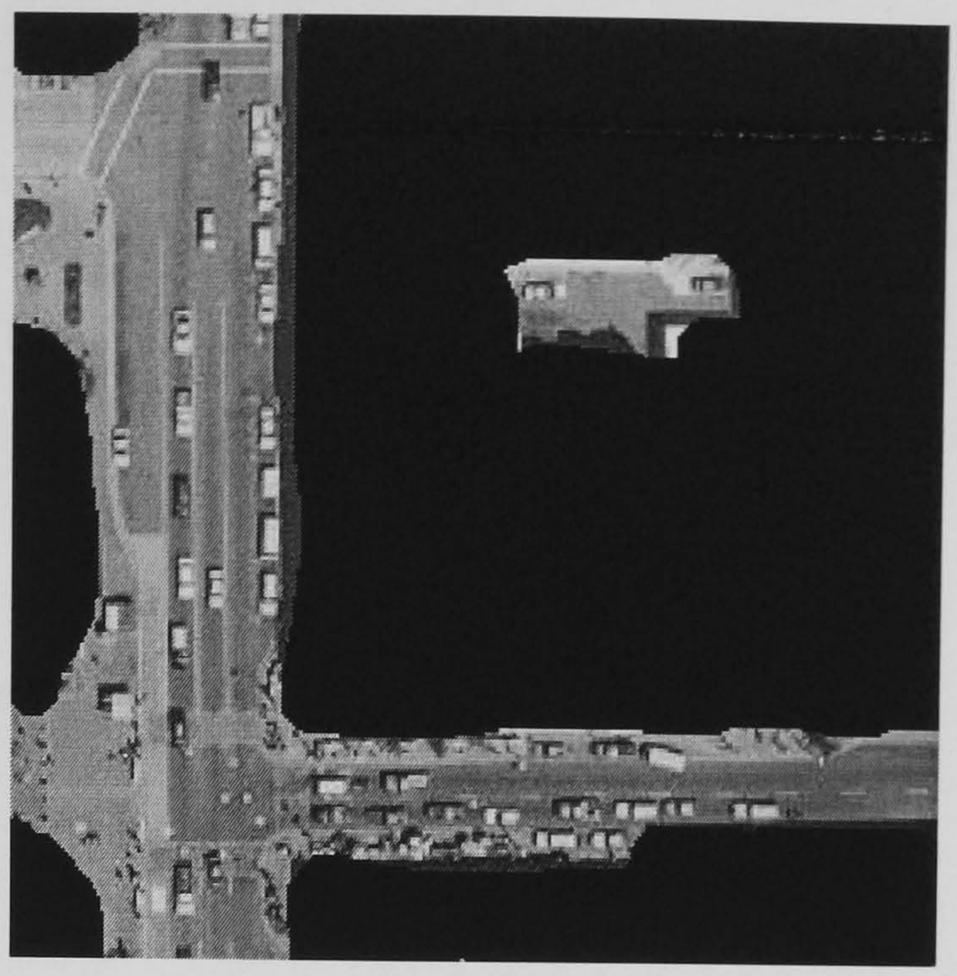


SGLDM /BPNN (88.47 %)

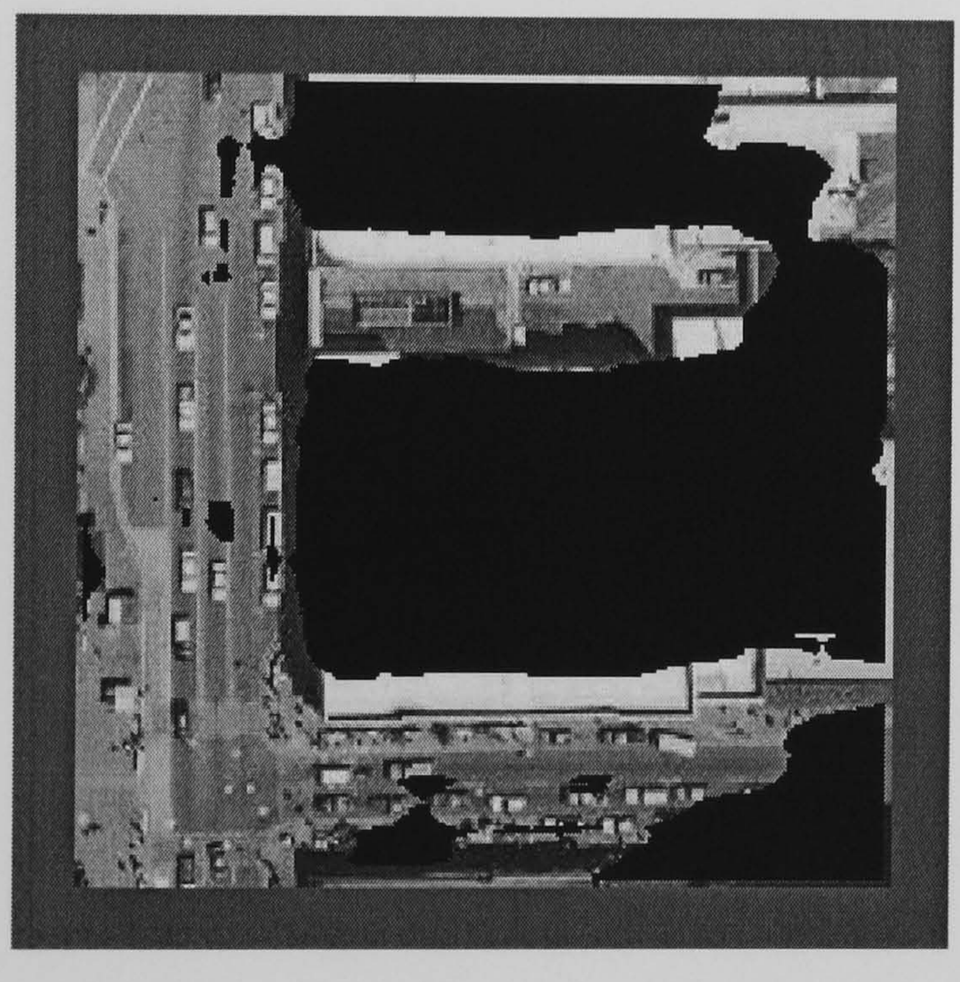
Figure E.5 Aerial Image 7 Results



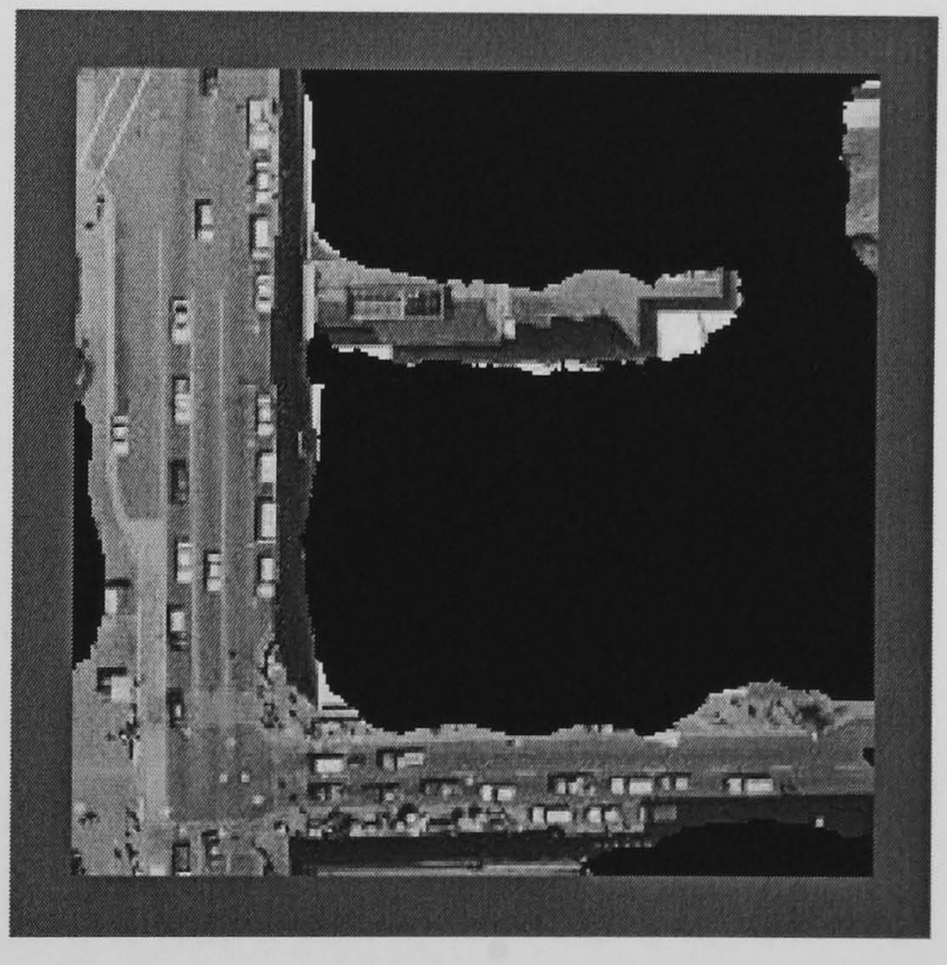
Aerial Image 8



Hand Segmented Image

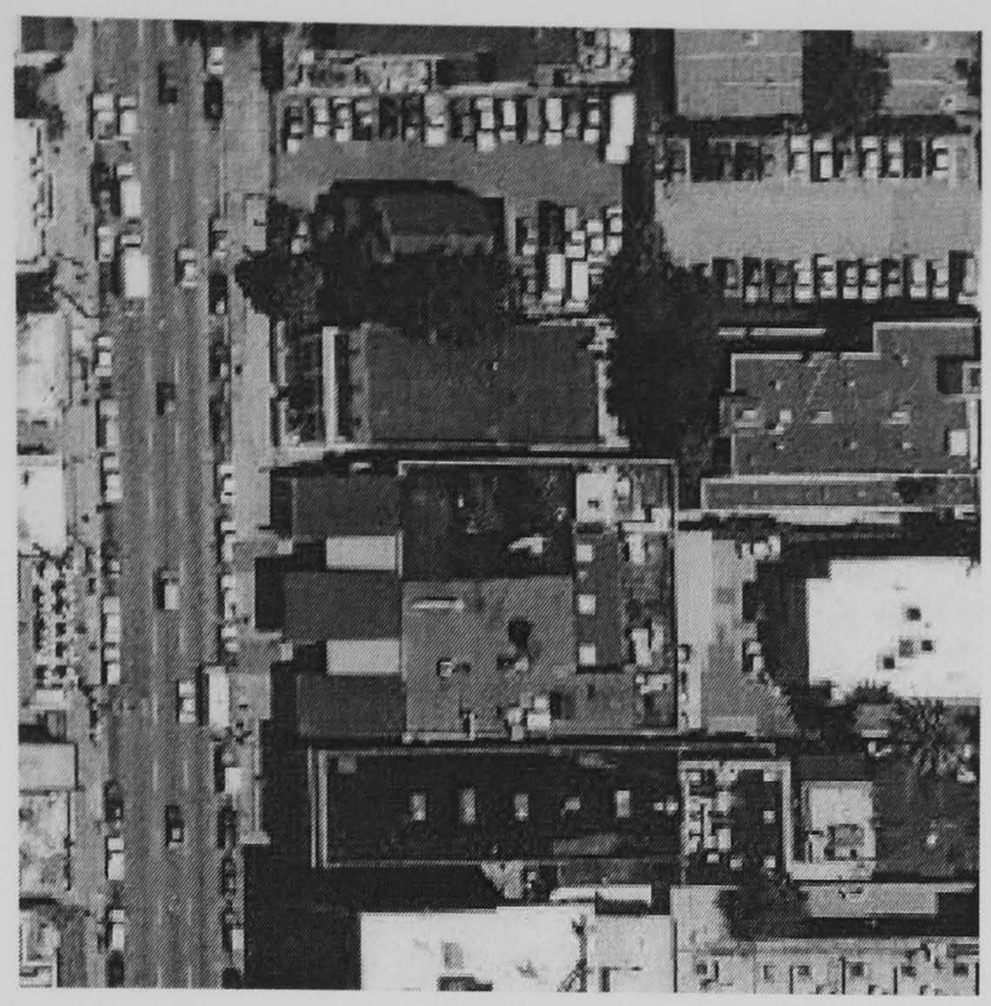


Hybrid Neural Network (83.85 %)

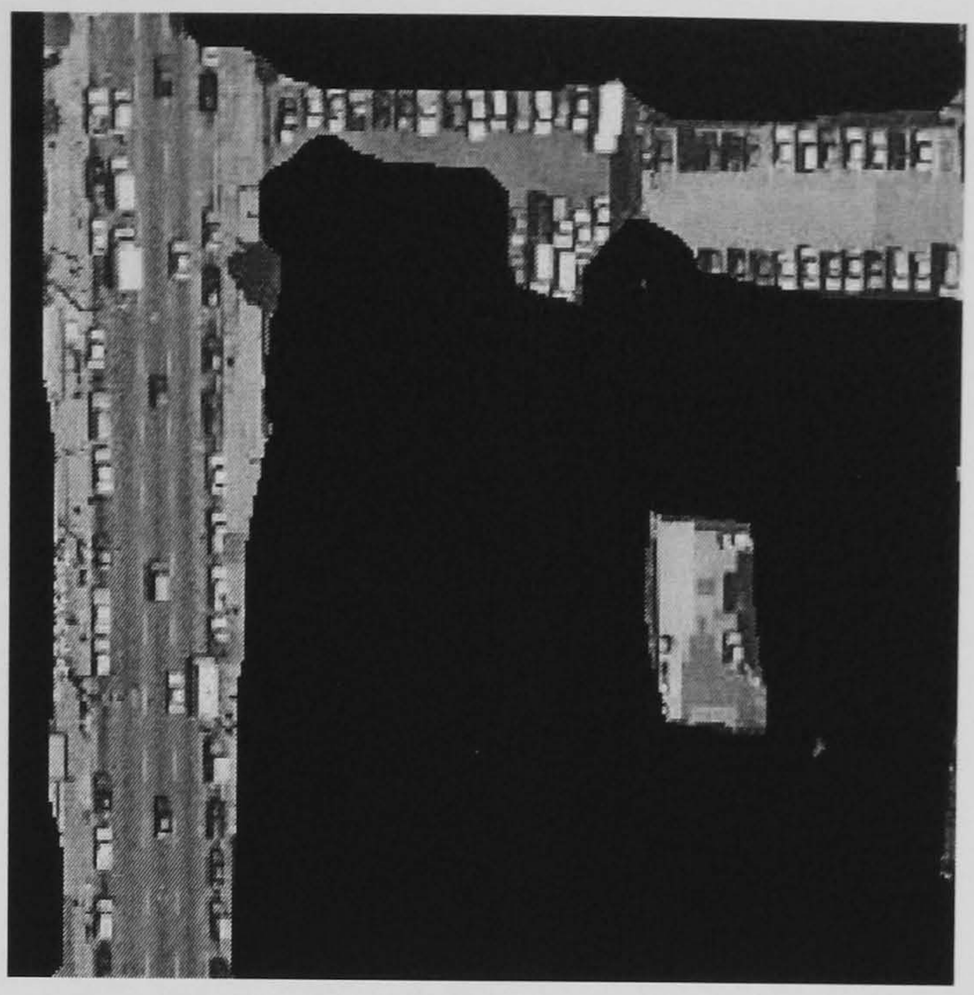


SGLDM /BPNN (90.63 %)

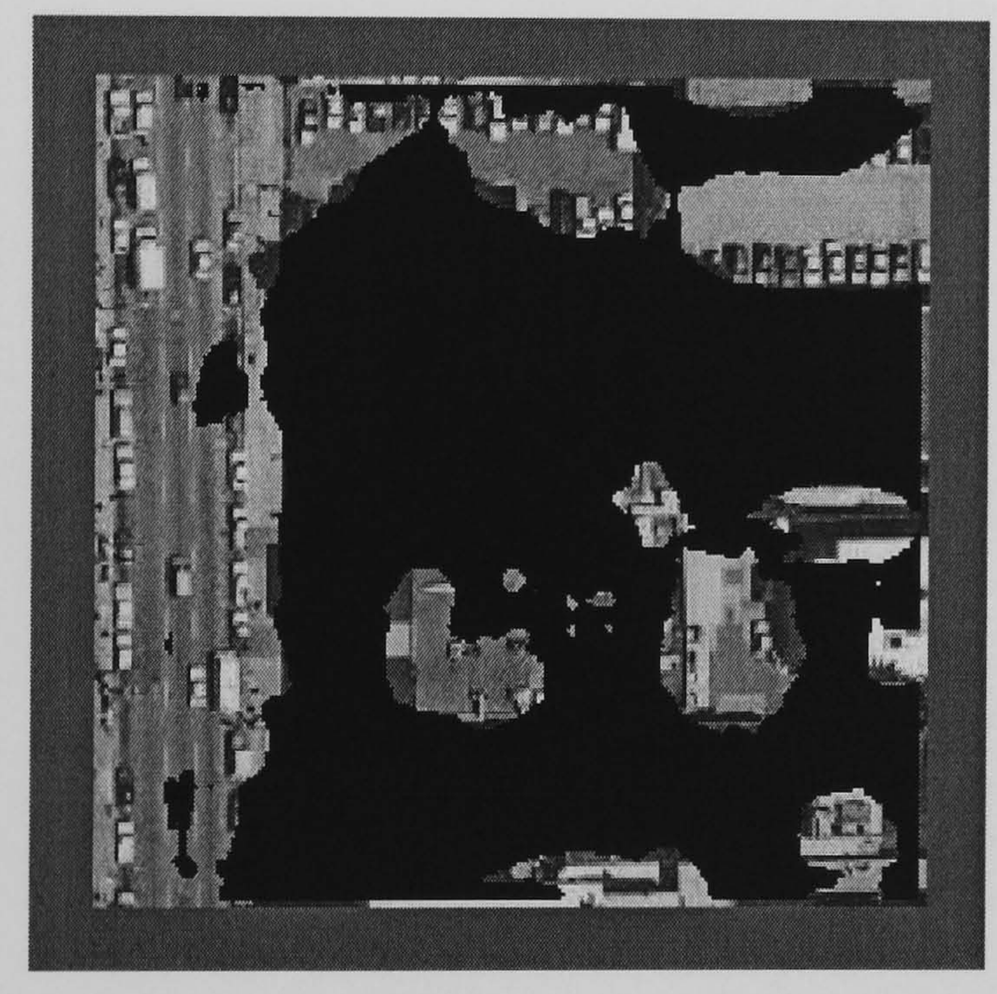
Figure E.6 Aerial Image 8 Results



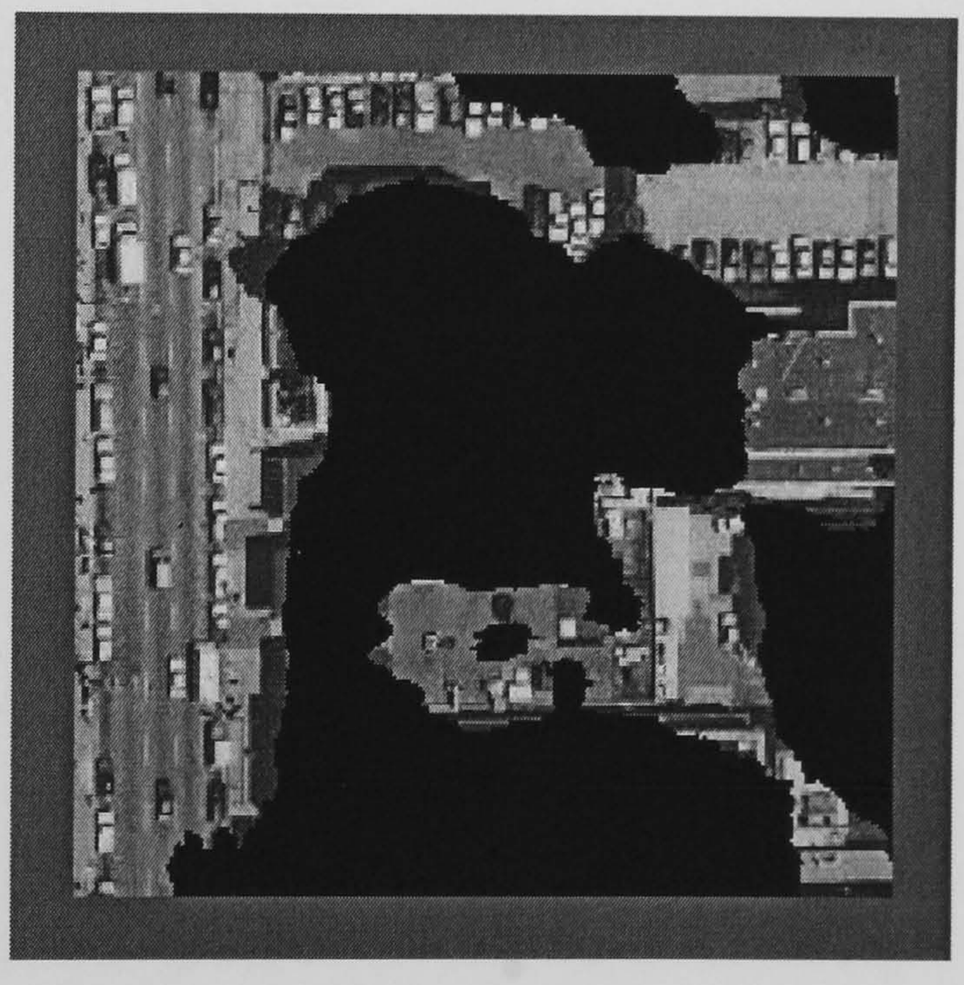
Aerial Image 9



Hand Segmented Image

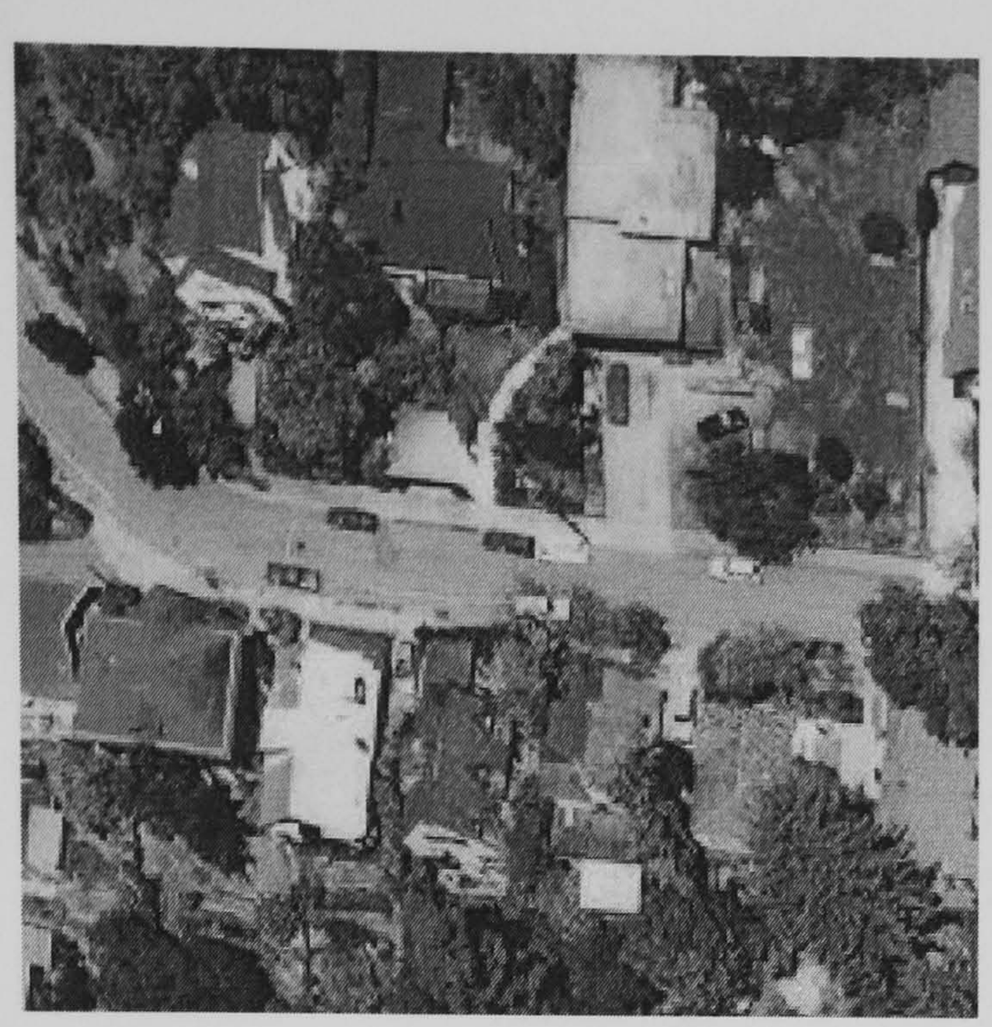


Hybrid Neural Network (85.59 %)



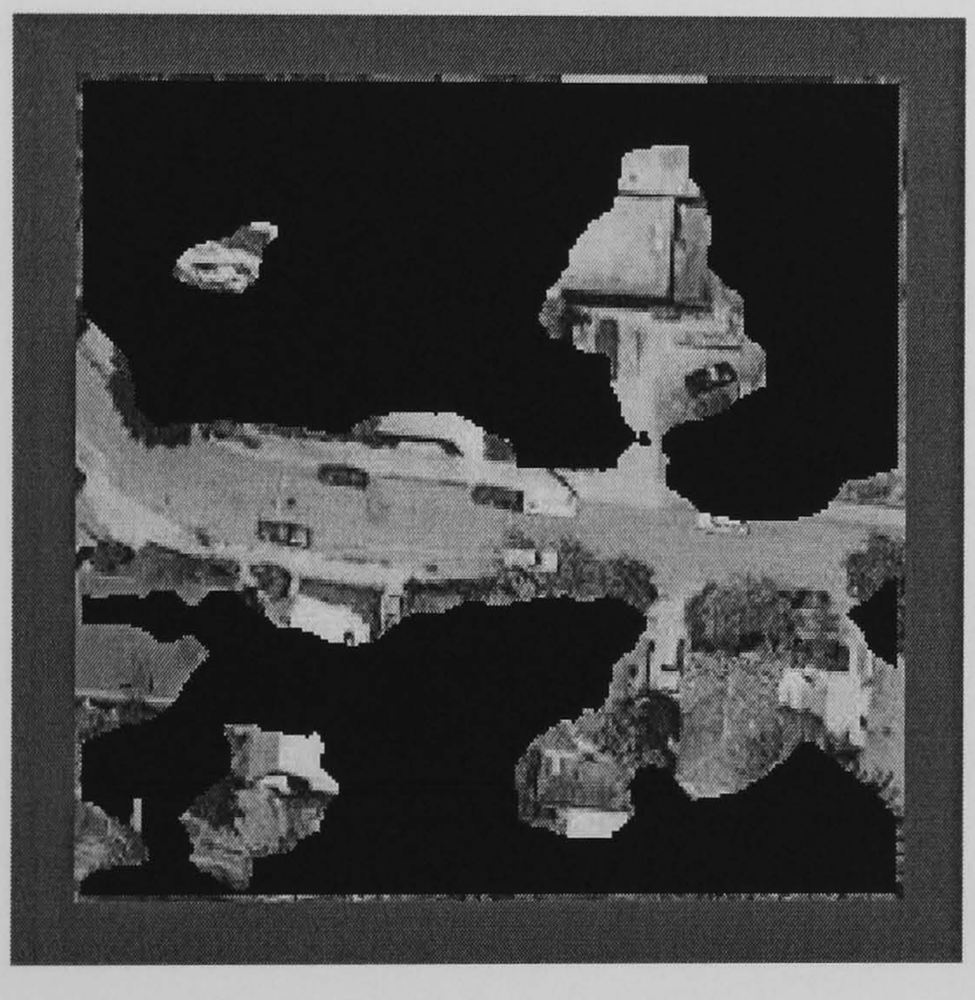
SGLDM /BPNN (83.08 %)

Figure E.7 Aerial Image 9 Results

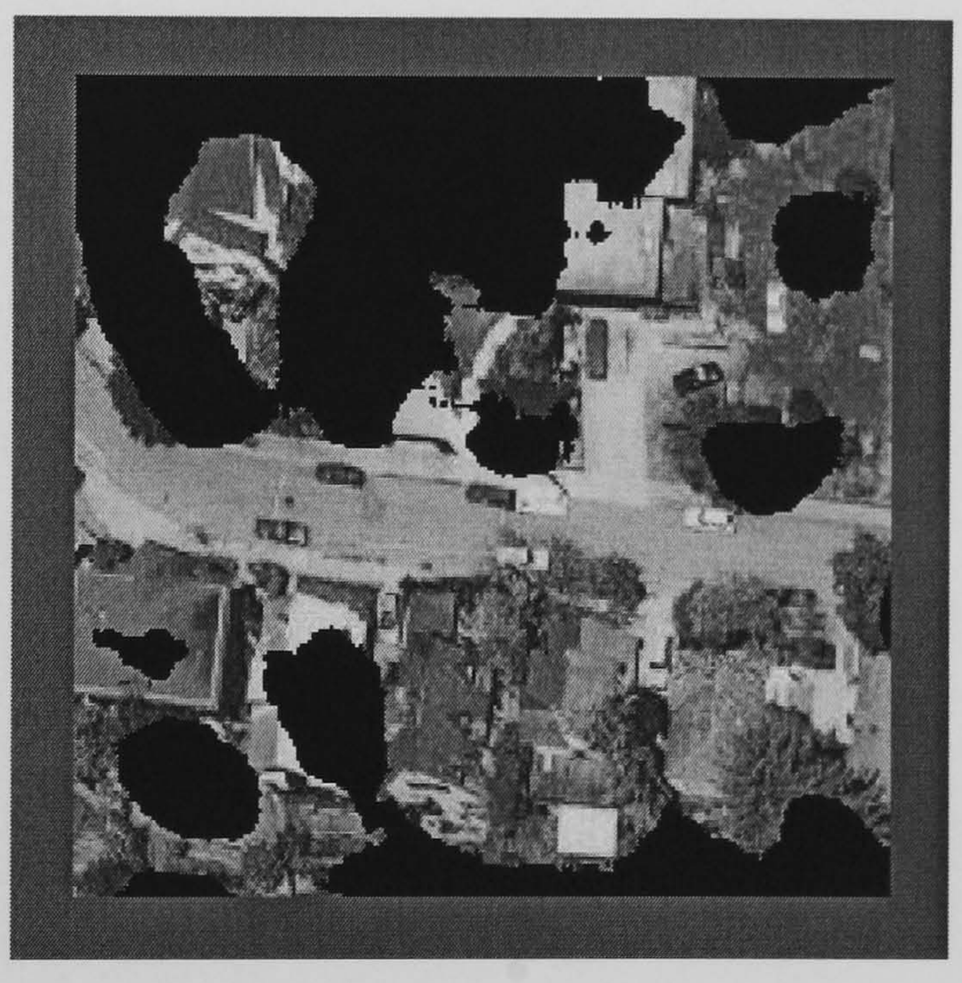


Aerial Image 10

Hand Segmented Image



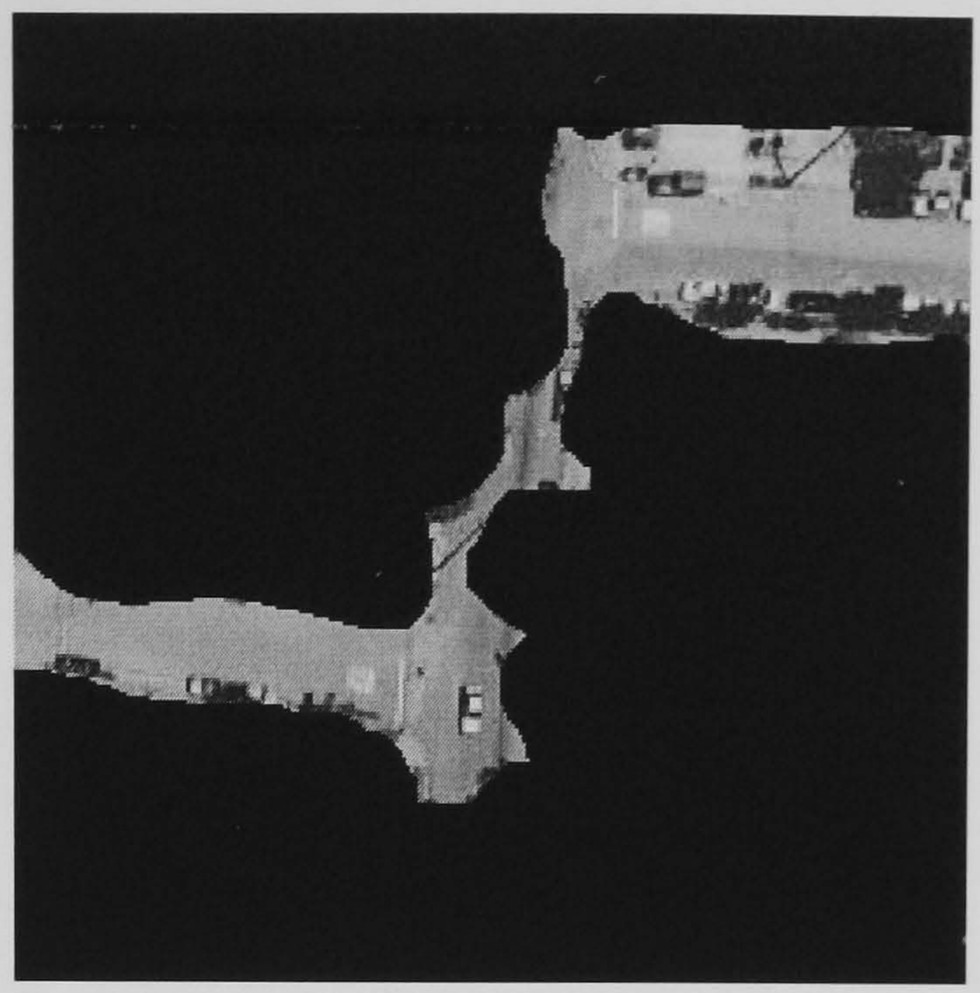
Hybrid Neural Network (79.6 %)



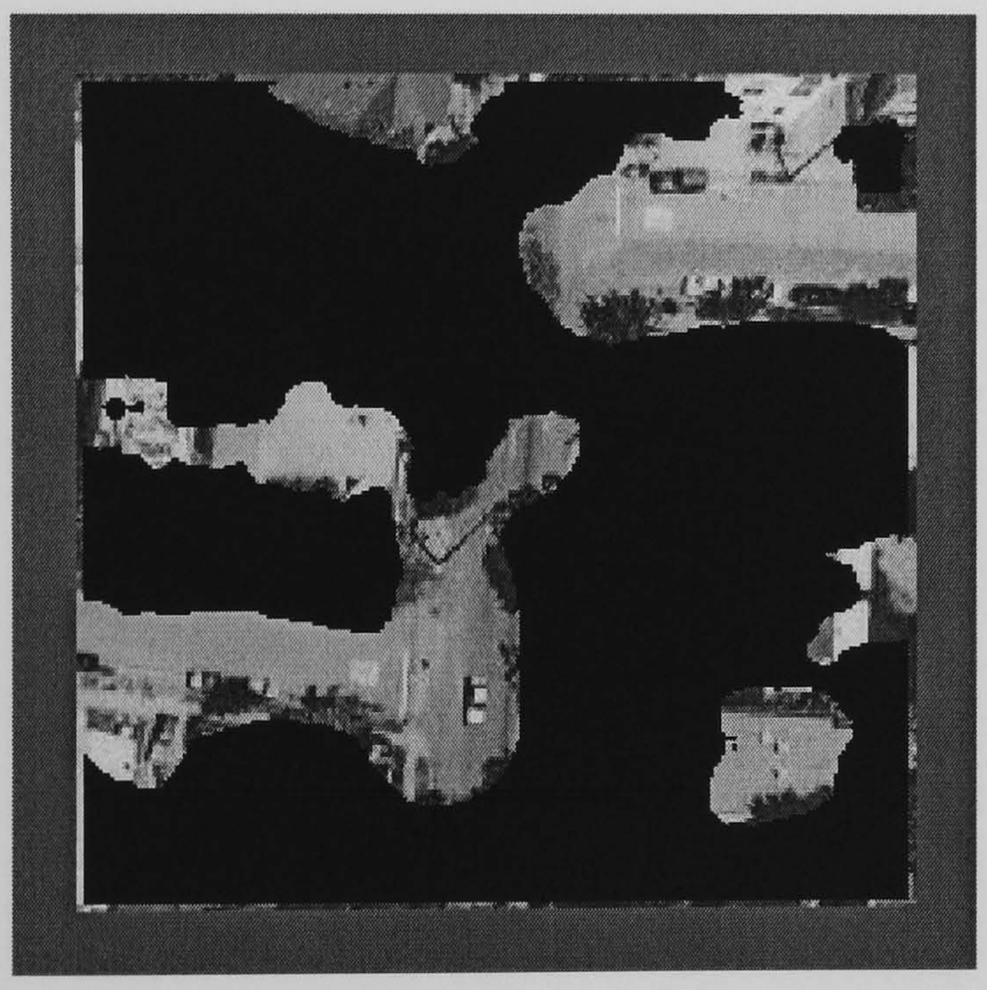
SGLDM /BPNN (62.78 %)

Figure E.8 Aerial Image 10 Results

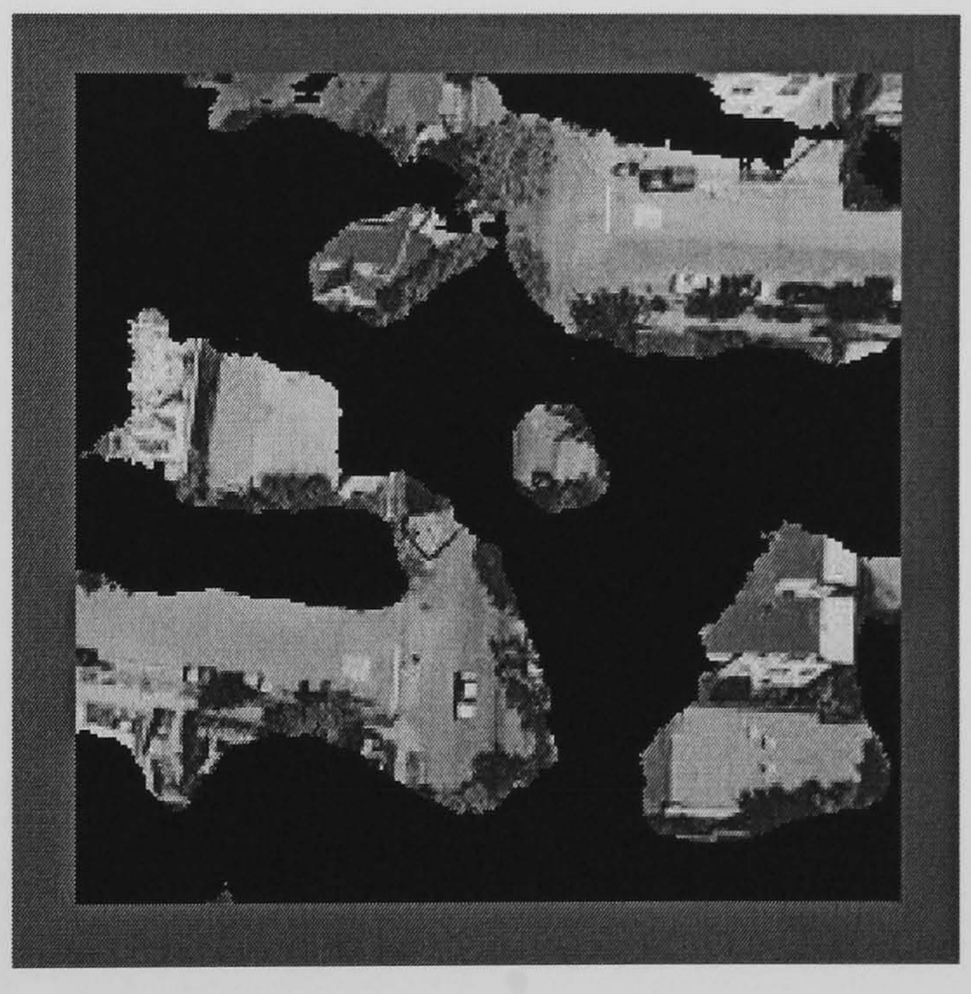
Aerial Image 11



Hand Segmented Image



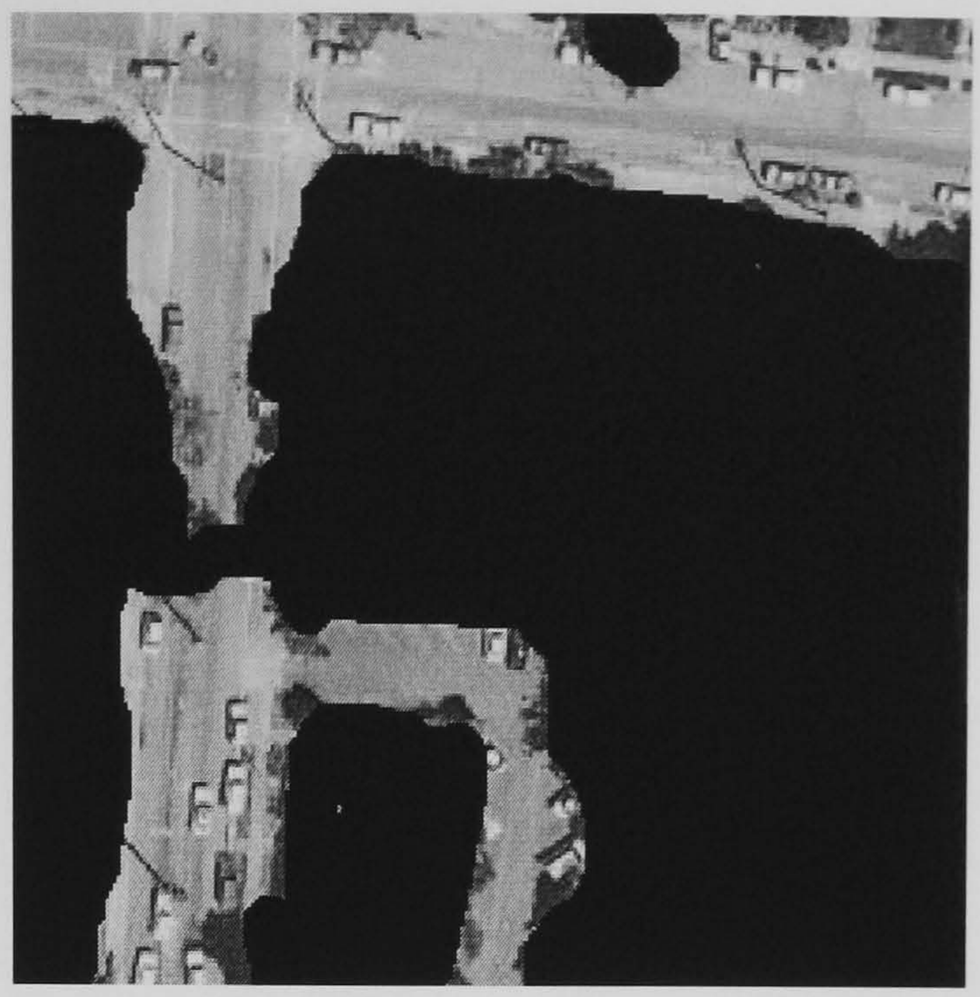
Hybrid Neural Network (84.55 %)



SGLDM /BPNN (77.17 %)

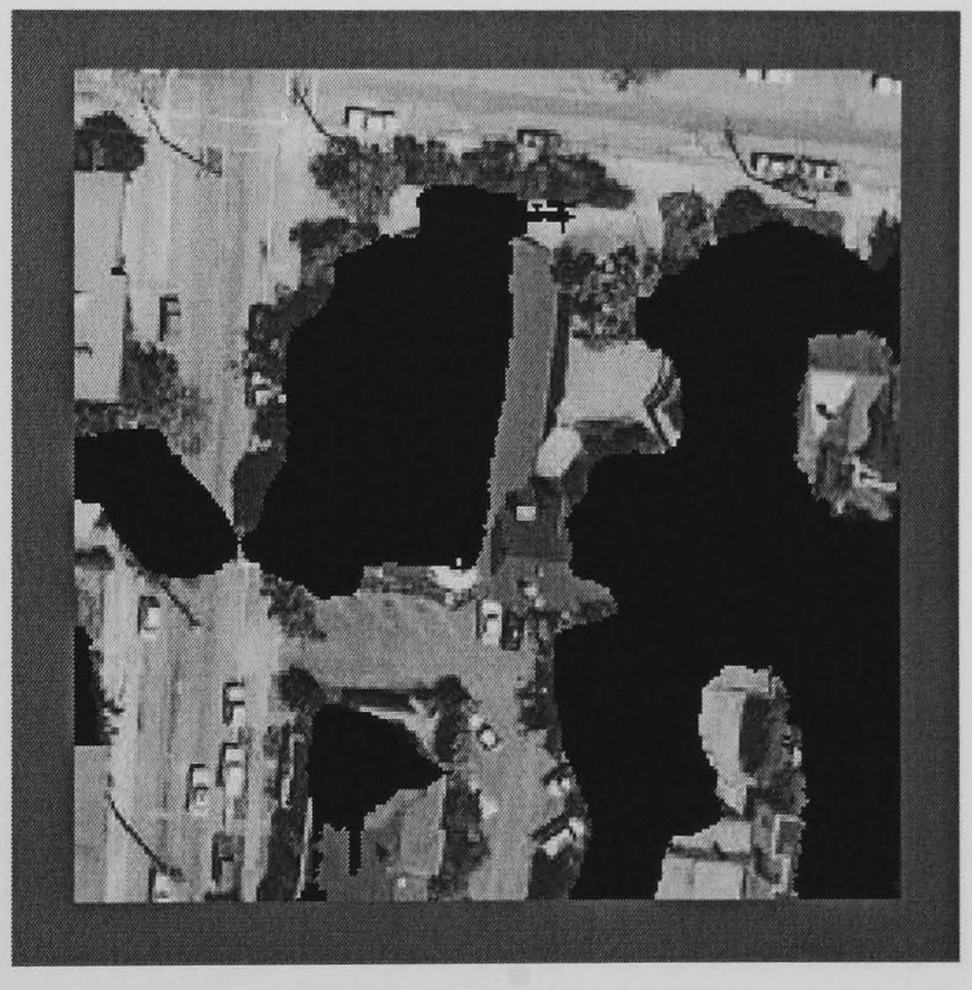
Figure E.9 Aerial Image 11 Results

Aerial Image 12



Hand Segmented Image

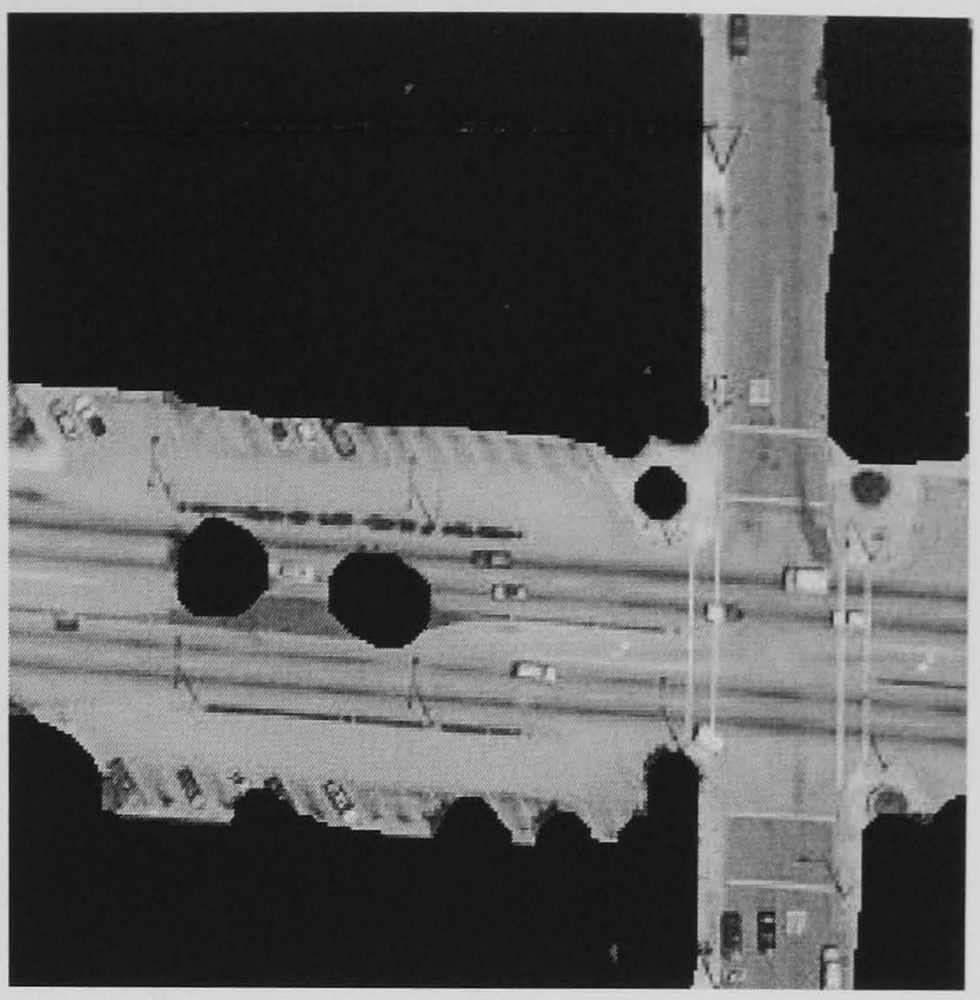
Hybrid Neural Network (85.89 %)



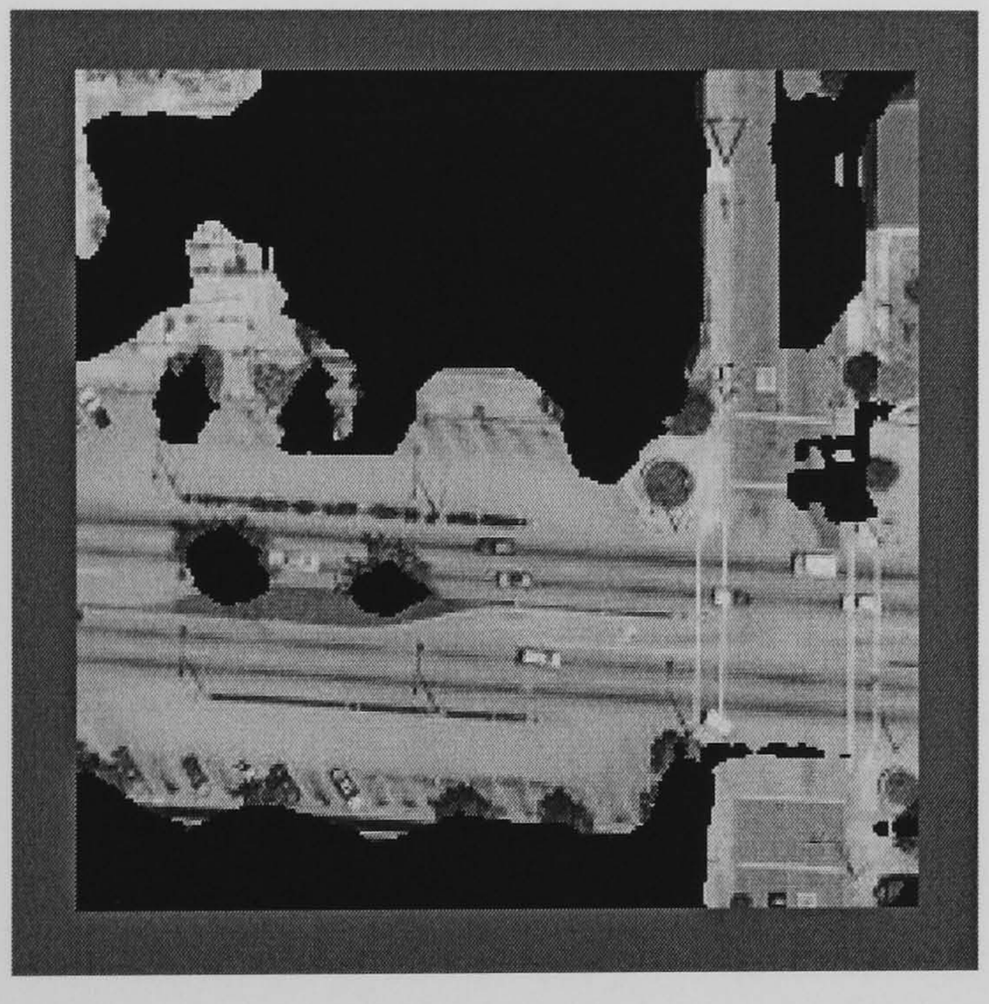
SGLDM /BPNN (76.91 %)

Figure E.10 Aerial Image 12 Results

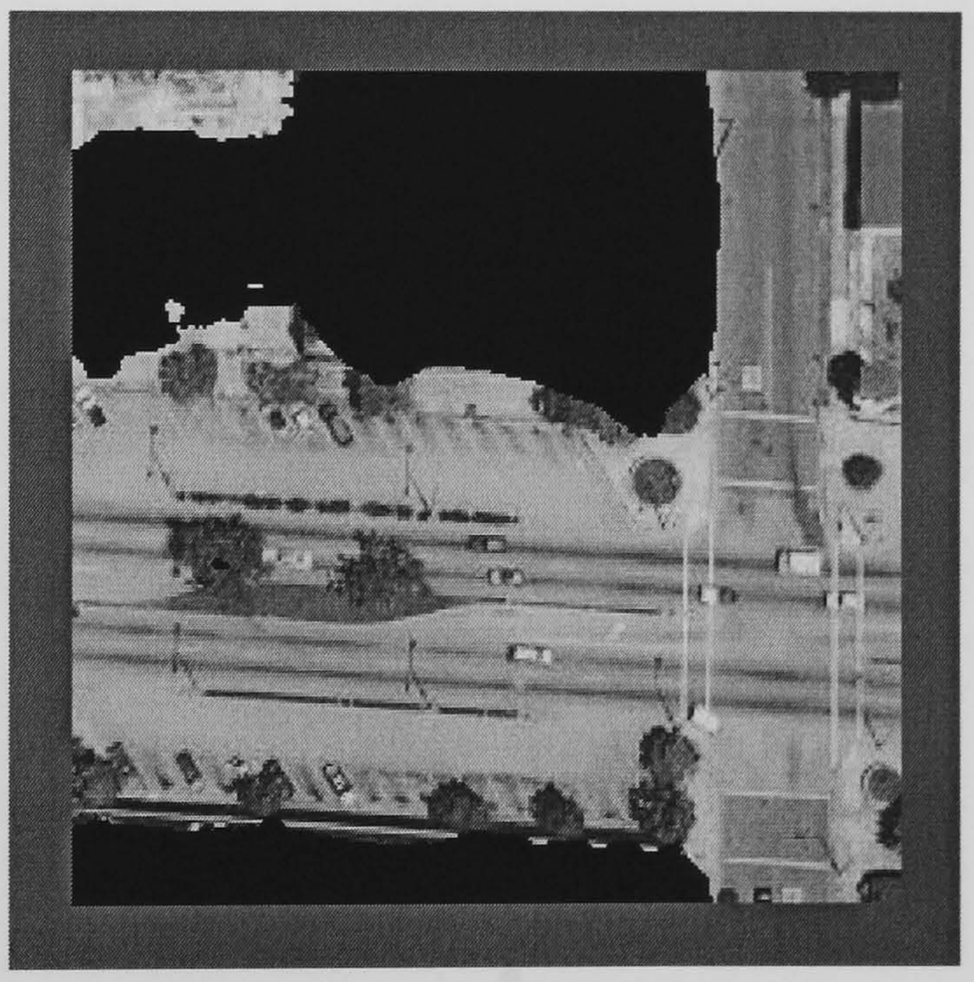
Aerial Image 13



Hand Segmented Image

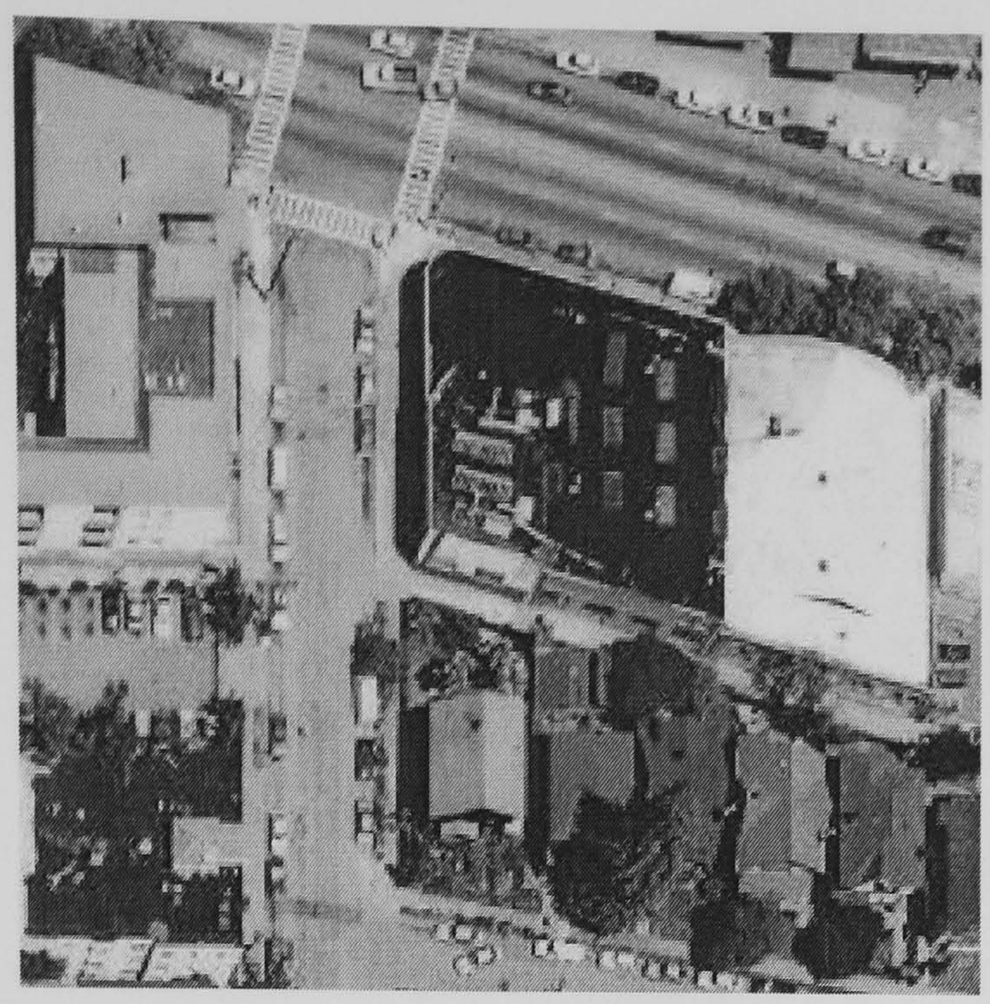


Hybrid Neural Network (86.73 %)

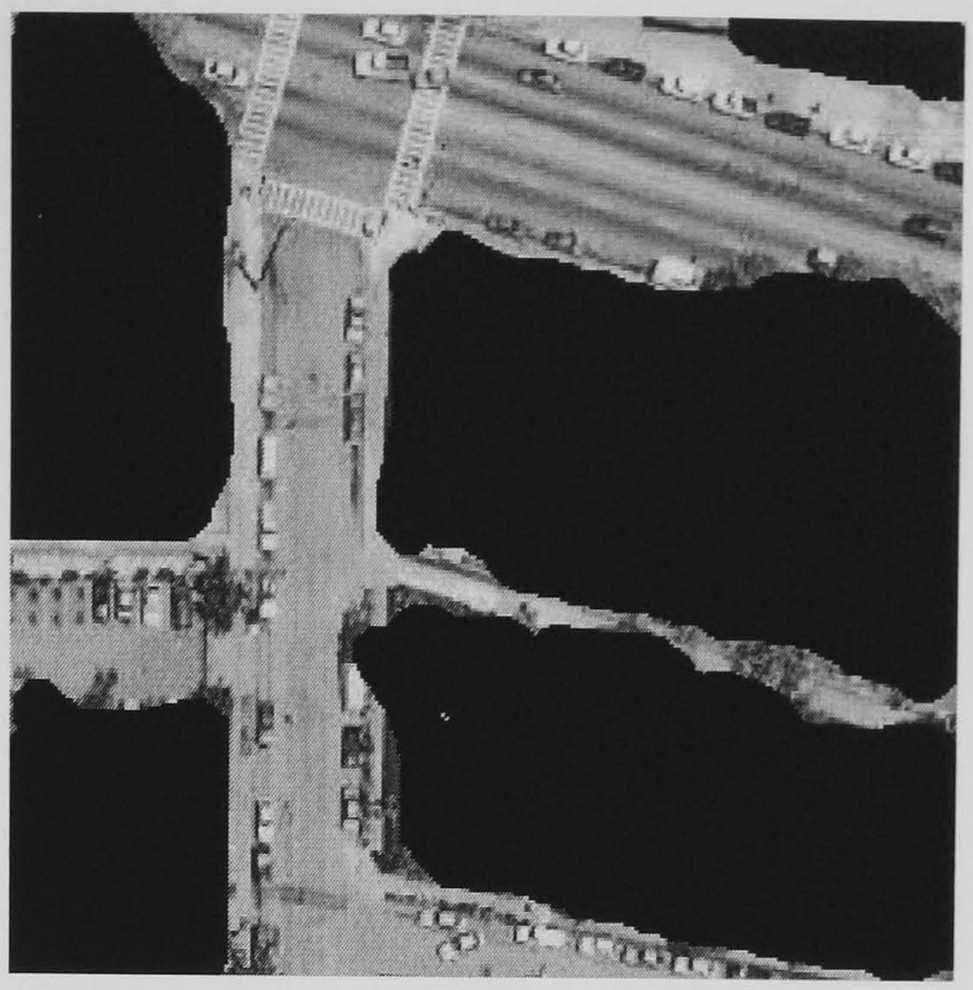


SGLDM /BPNN (86.09 %)

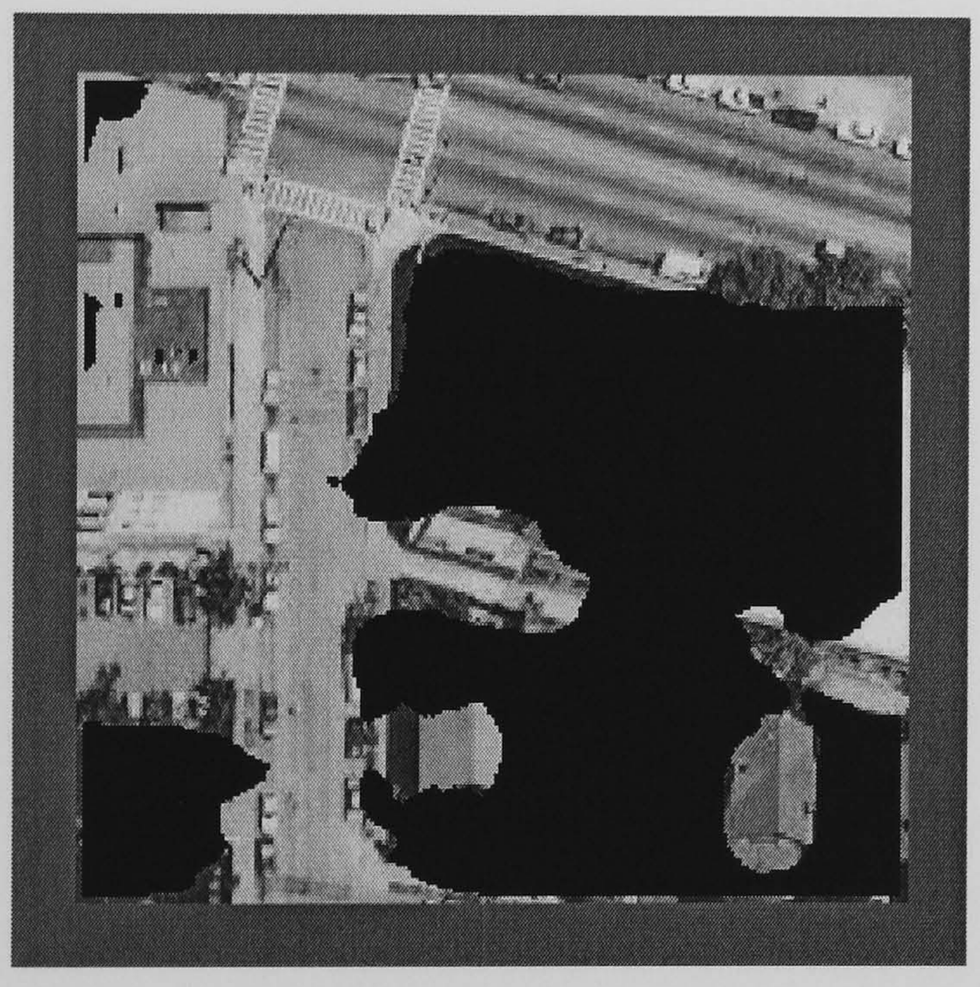
Figure E.11 Aerial Image 13 Results



Aerial Image 14



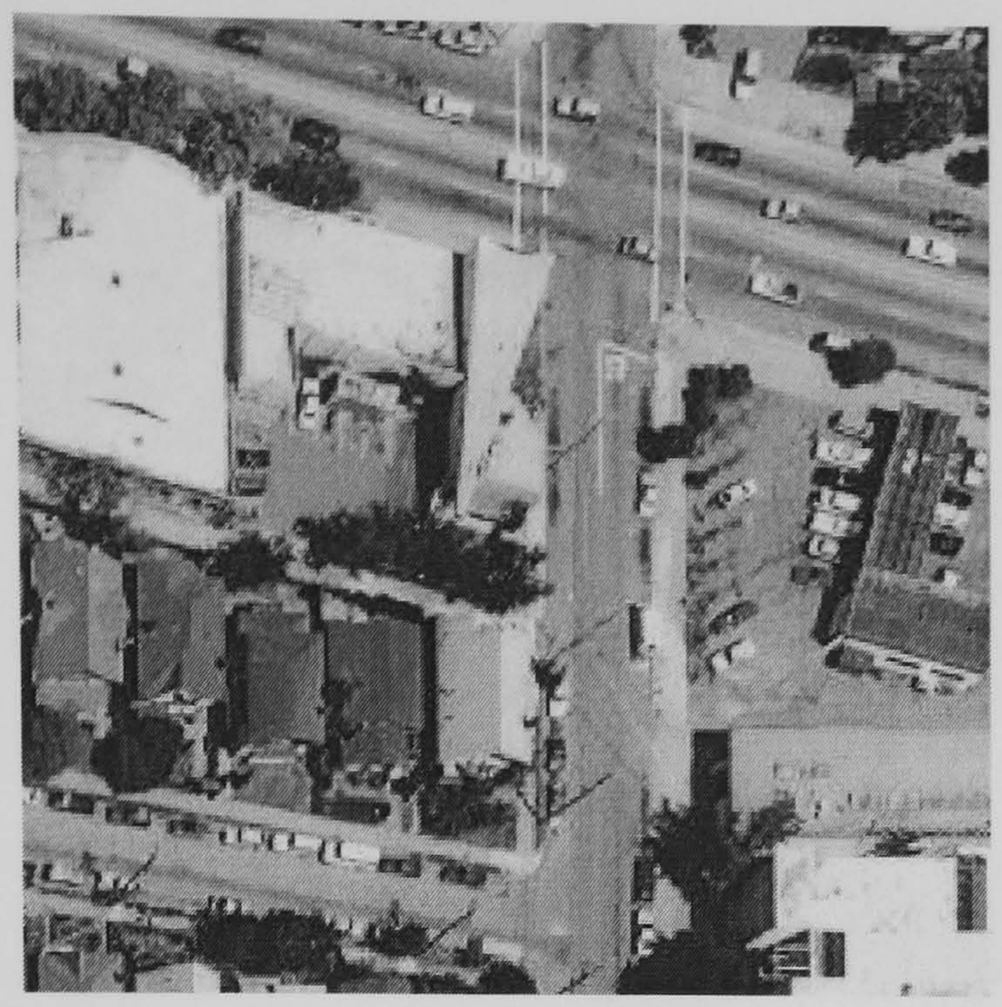
Hand Segmented Image



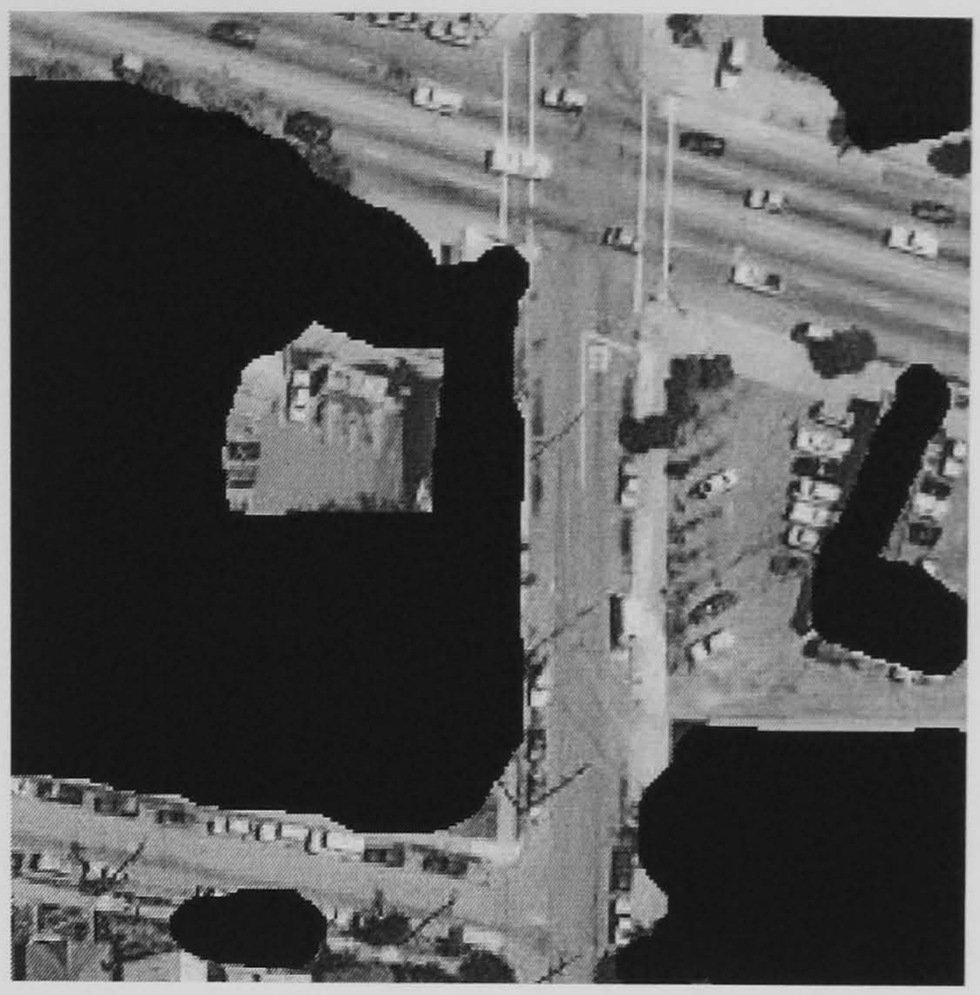
Hybrid Neural Network (84.12 %)

SGLDM /BPNN (78.48 %)

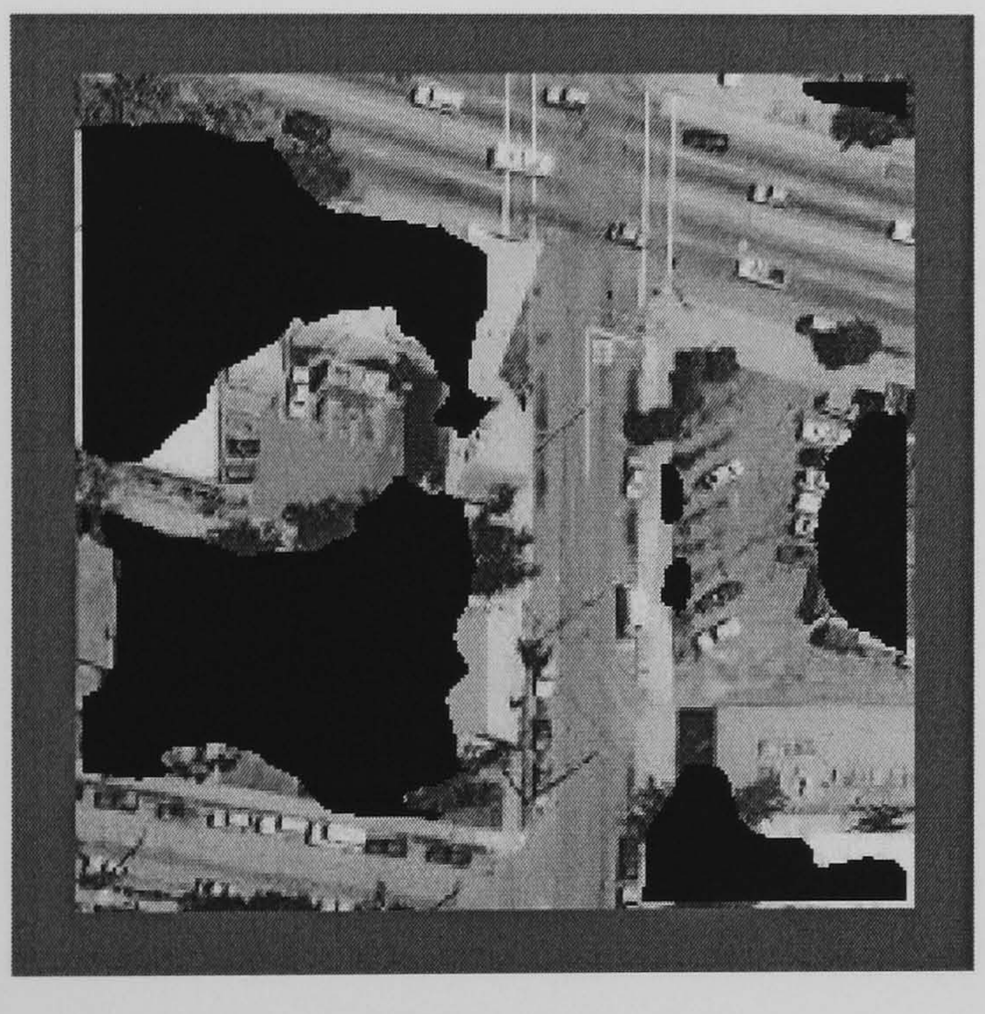
Figure E.12 Aerial Image 14 Results



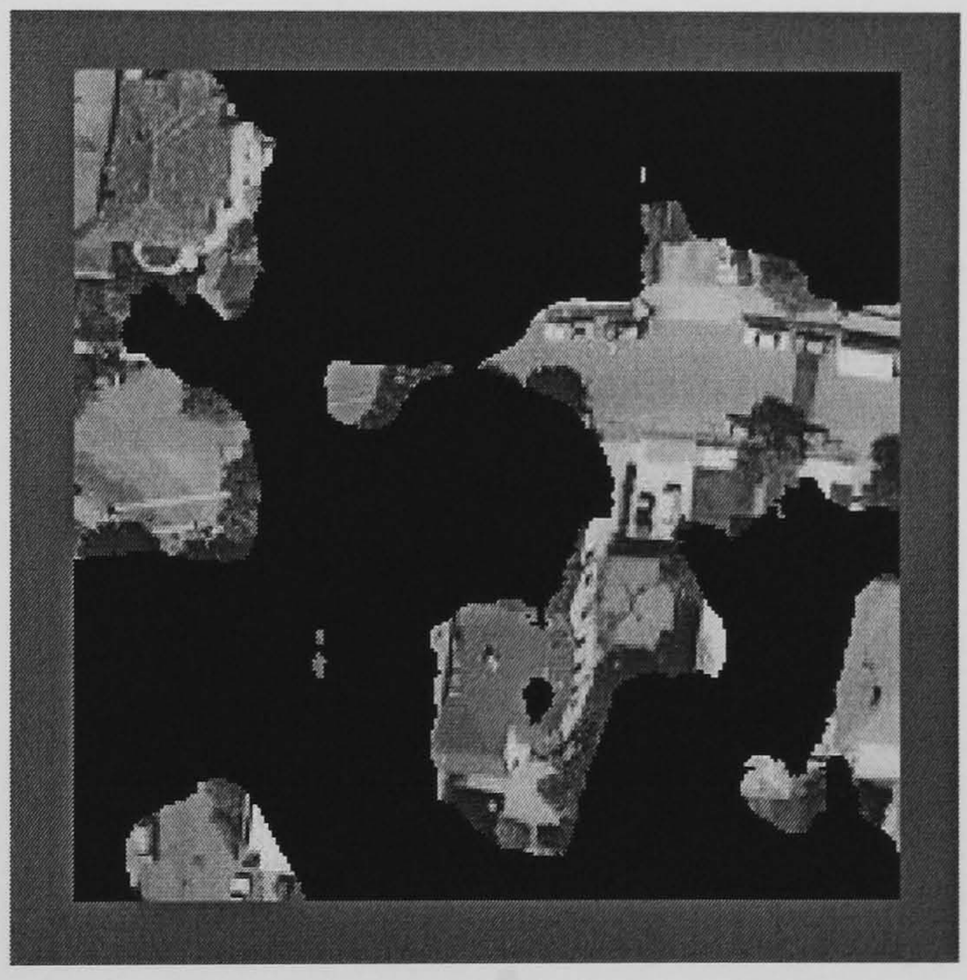
Aerial Image 15



Hand Segmented Image



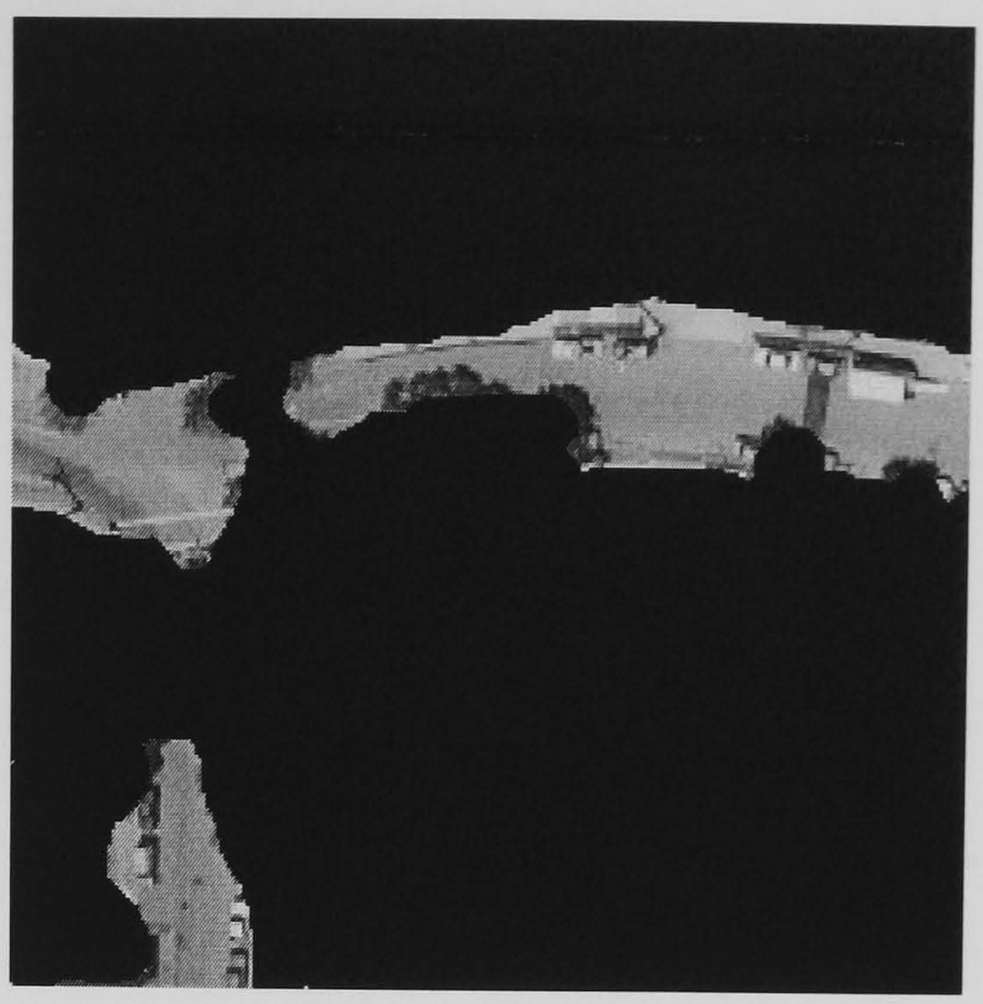
Hybrid Neural Network (85.68 %)



SGLDM /BPNN (80.77 %)

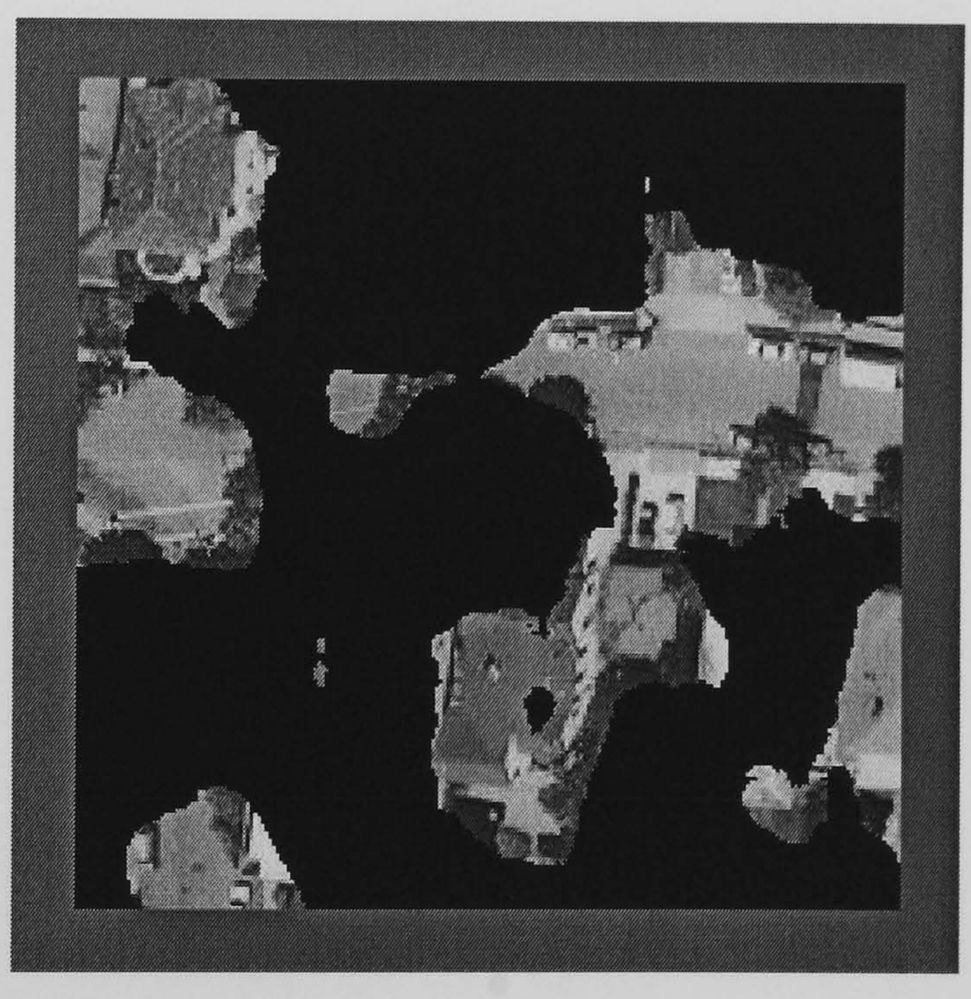
Figure E.13 Aerial Image 15 Results

Aerial Image 16



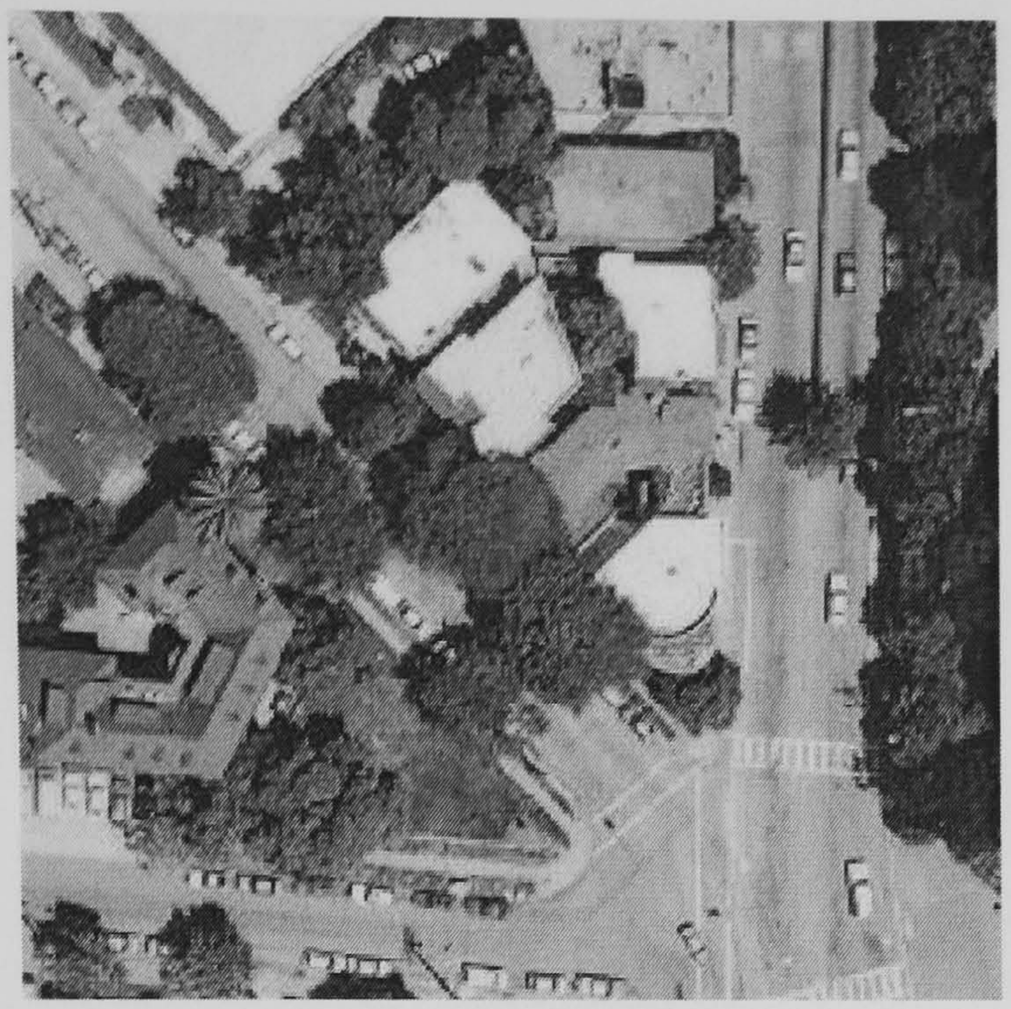
Hand Segmented Image

Hybrid Neural Network (82.73%)



SGLDM /BPNN (79.24 %)

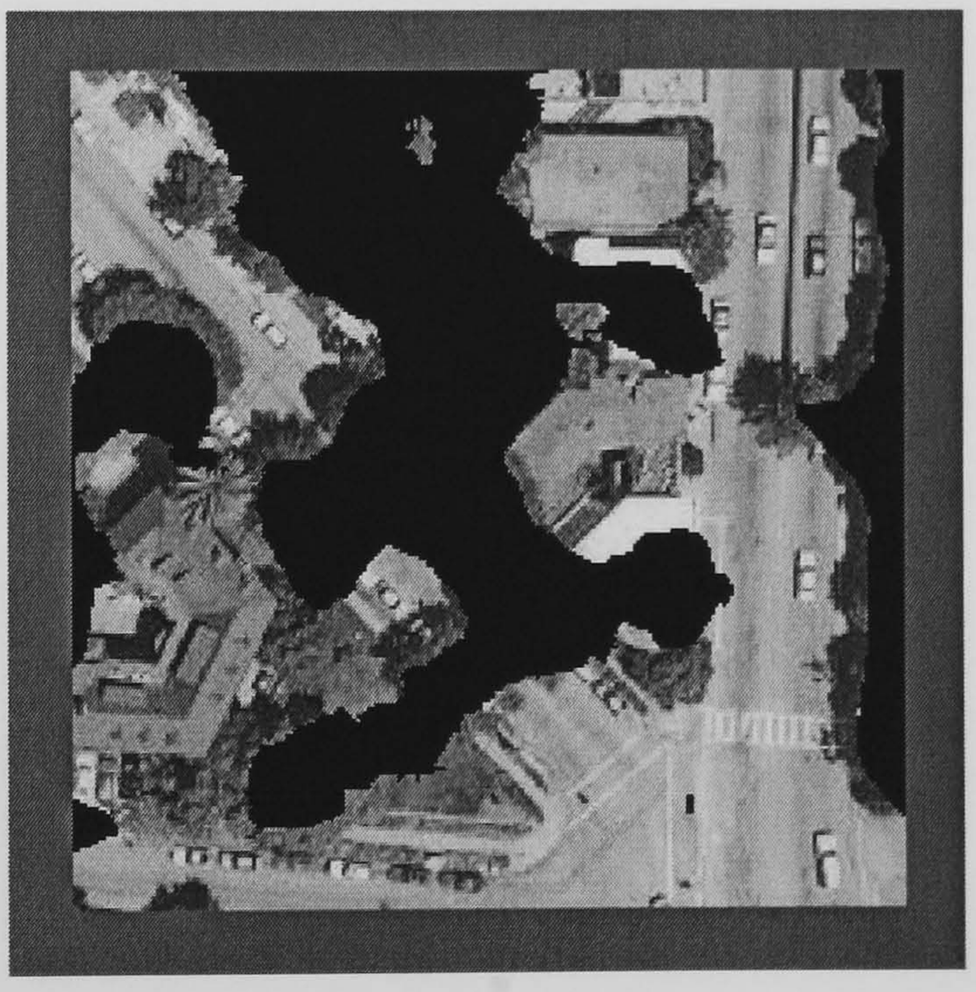
Figure E.14 Aerial Image 16 Results



Aerial Image 17

Hand Segmented Image

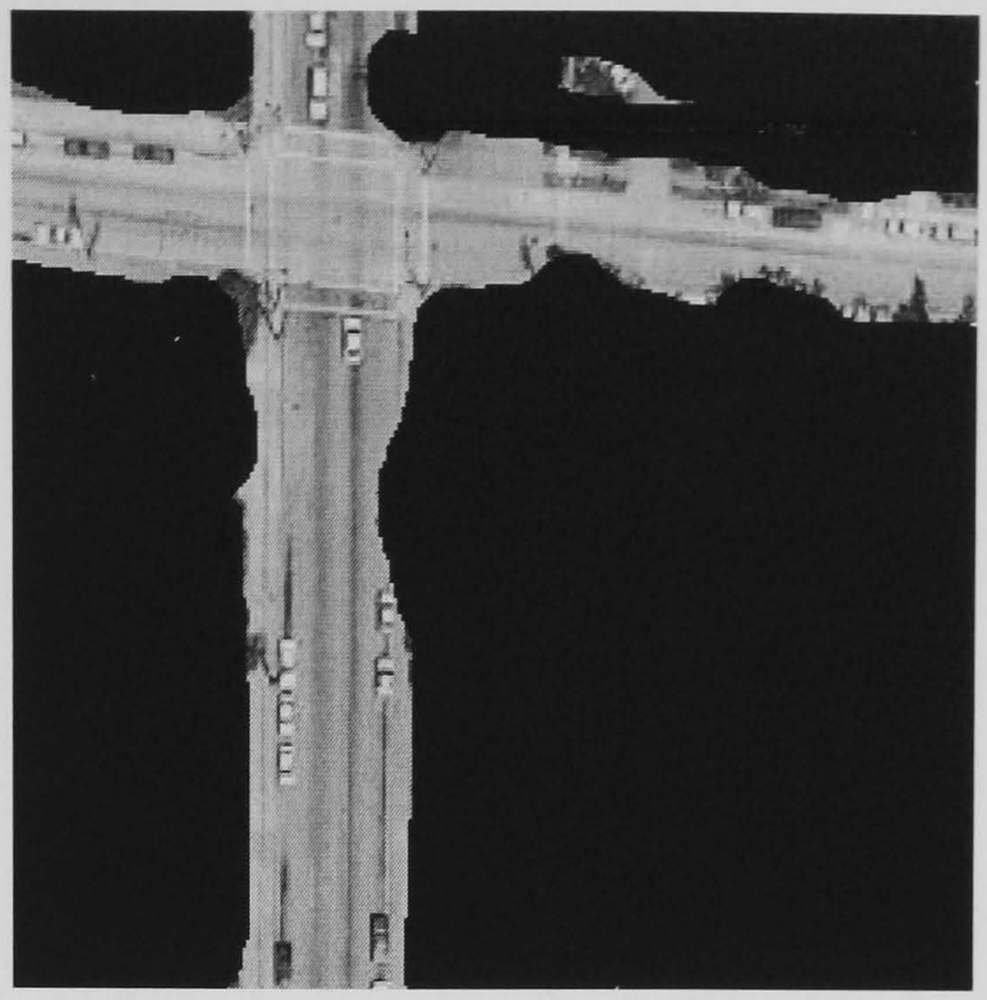
Hybrid Neural Network (77.93 %)



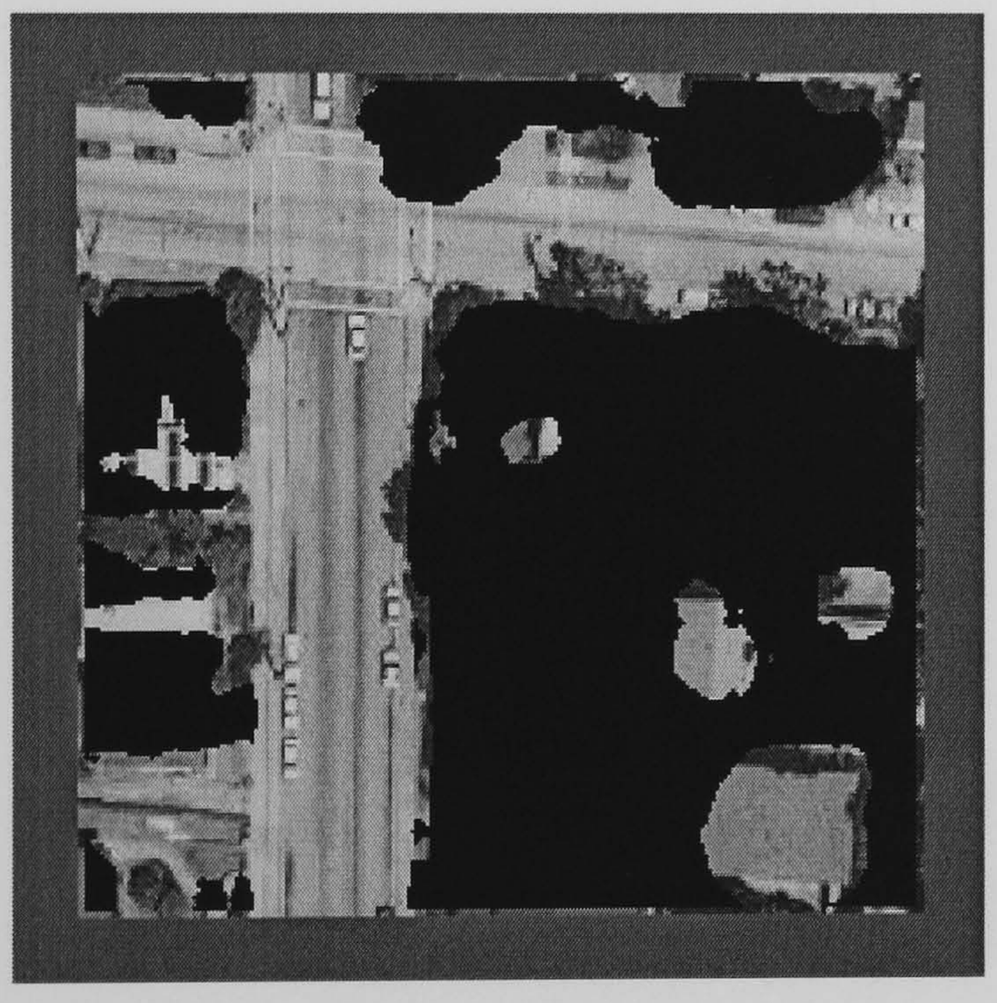
SGLDM /BPNN (73.54 %)

Figure E.15 Aerial Image 17 Results

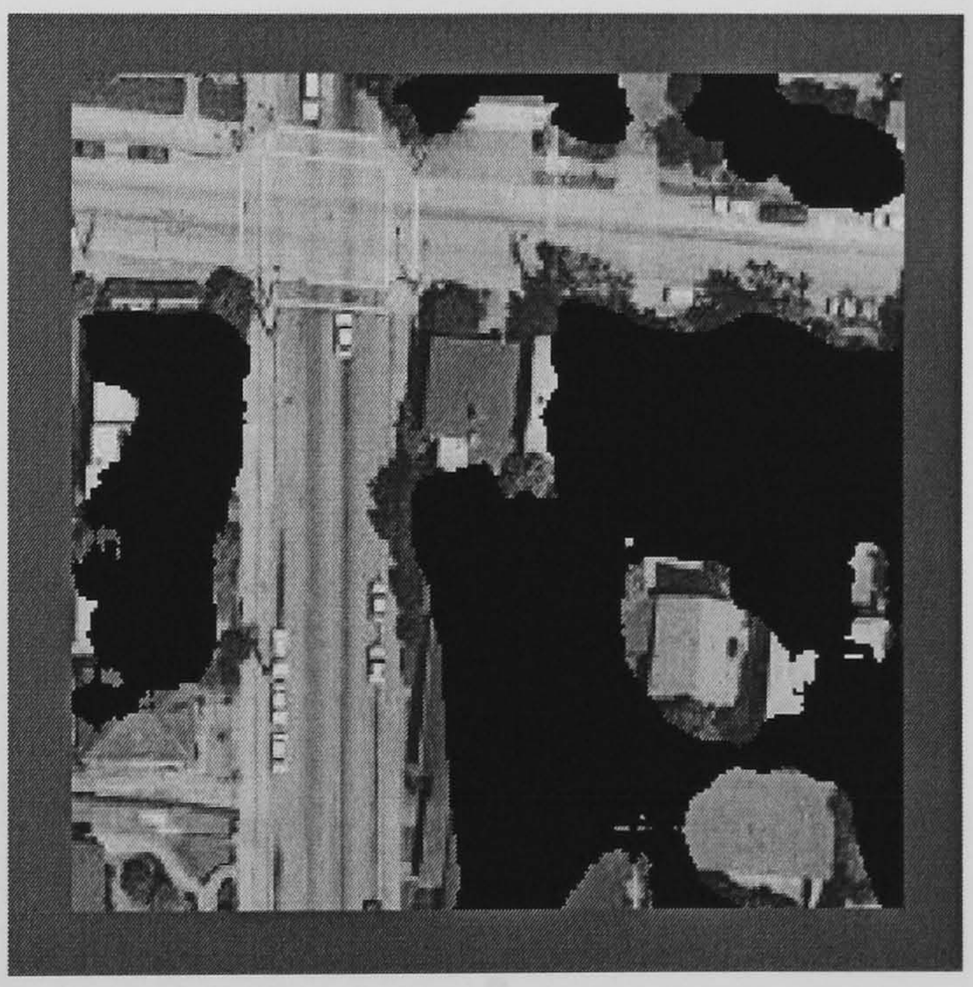
Aerial Image 18



Hand Segmented Image

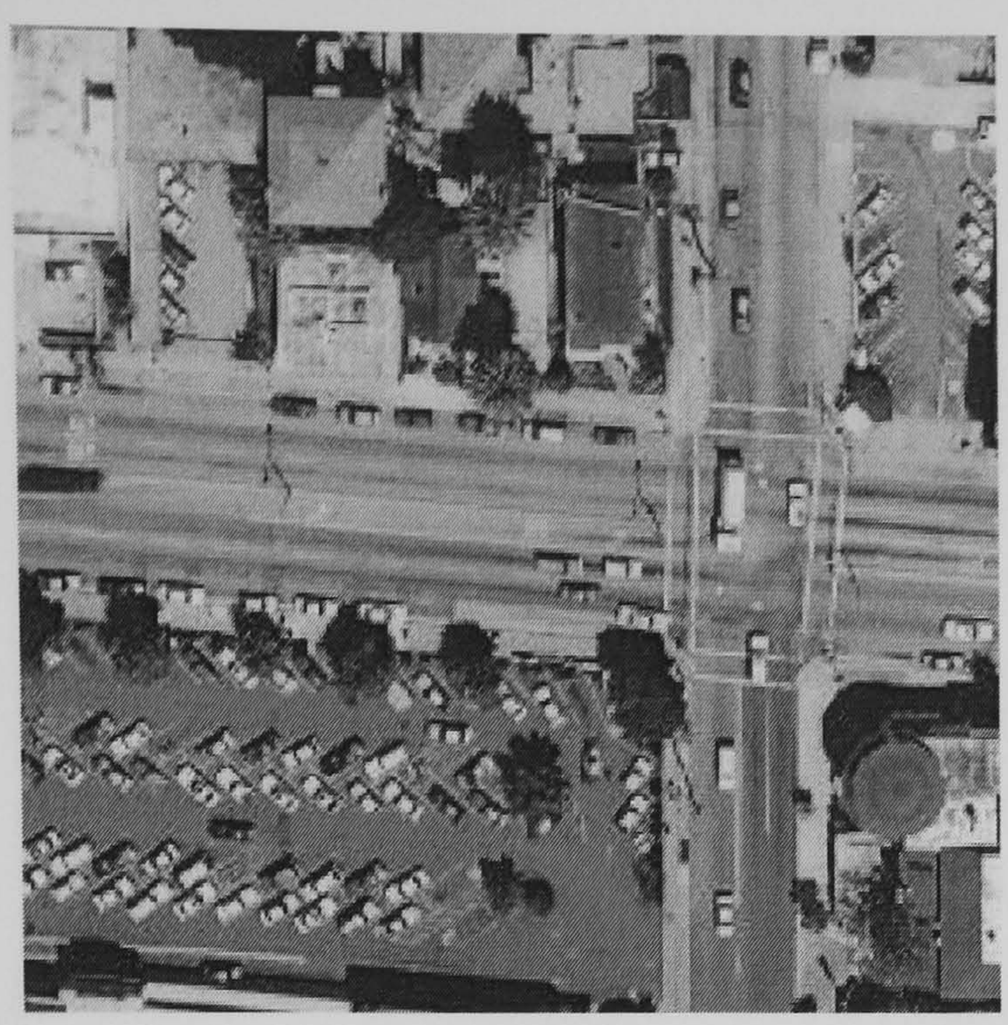


Hybrid Neural Network (82.75 %)



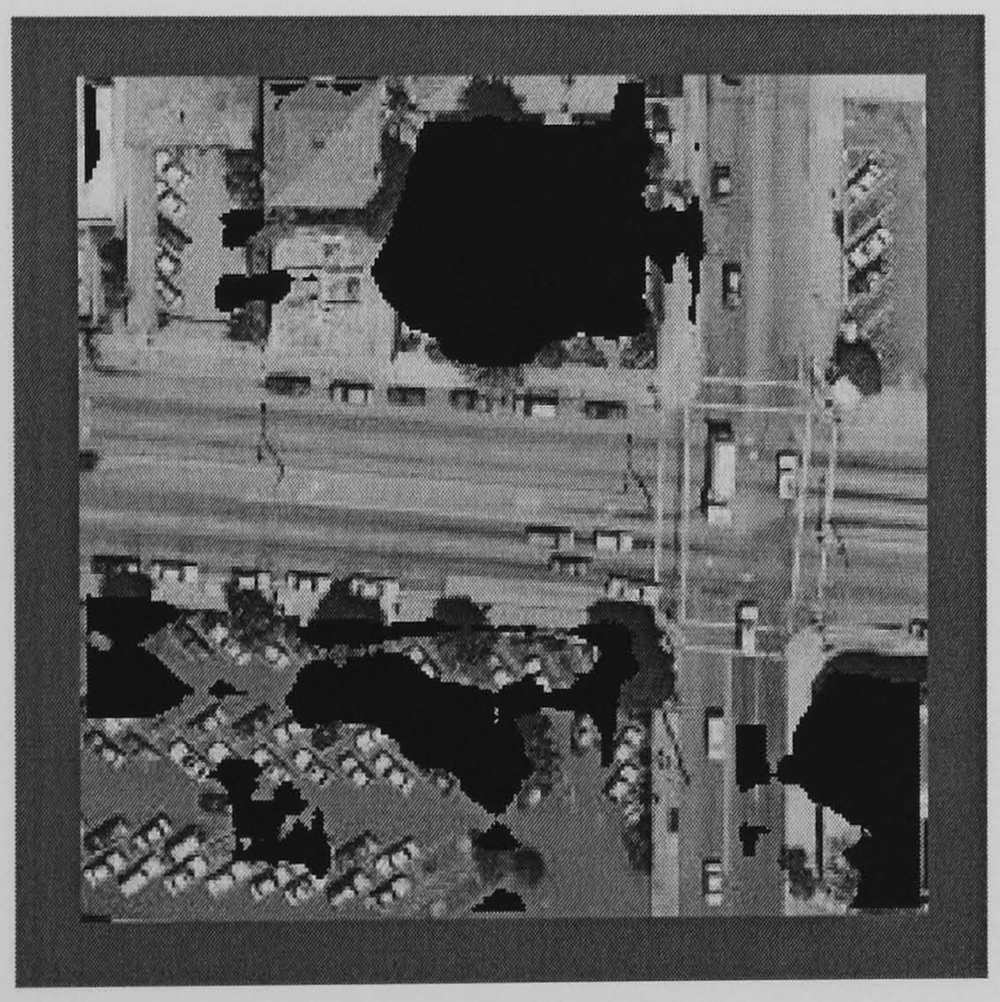
SGLDM /BPNN (77.89 %)

Figure E.16 Aerial Image 18 Results

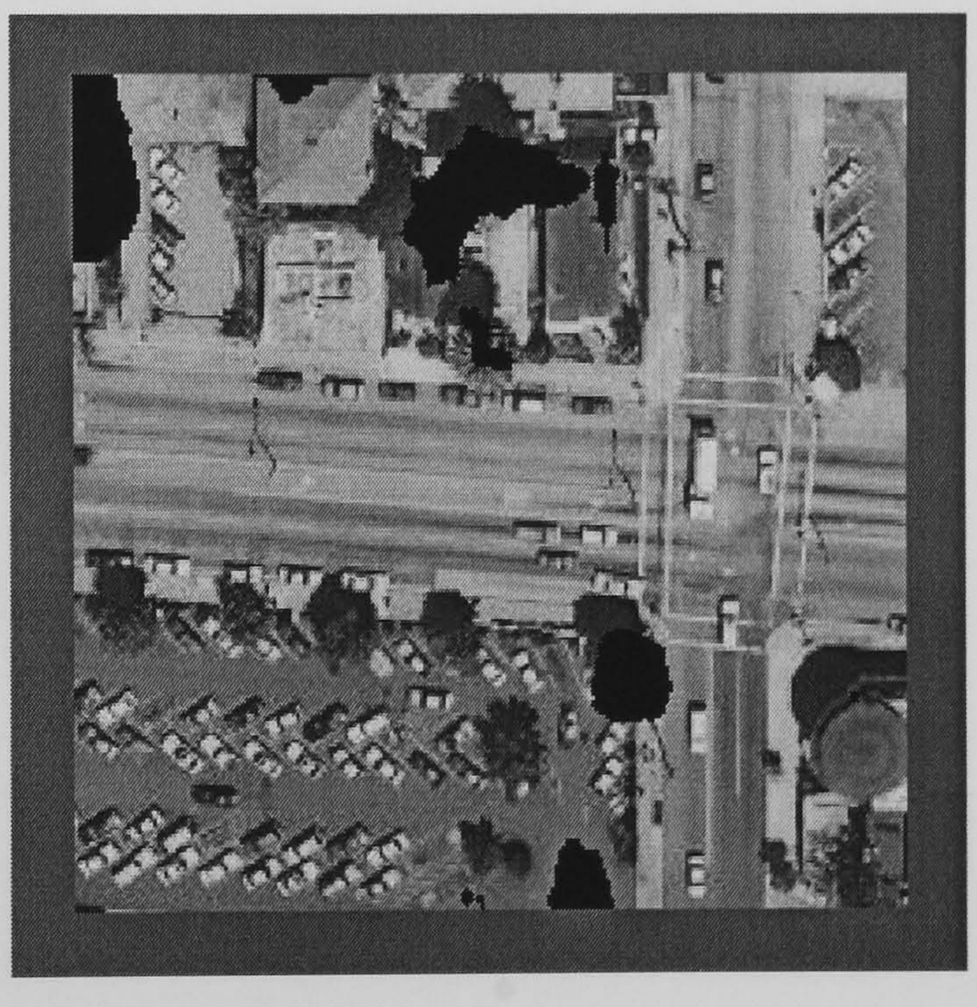


Aerial Image 19

Hand Segmented Image



Hybrid Neural Network (84.75 %)

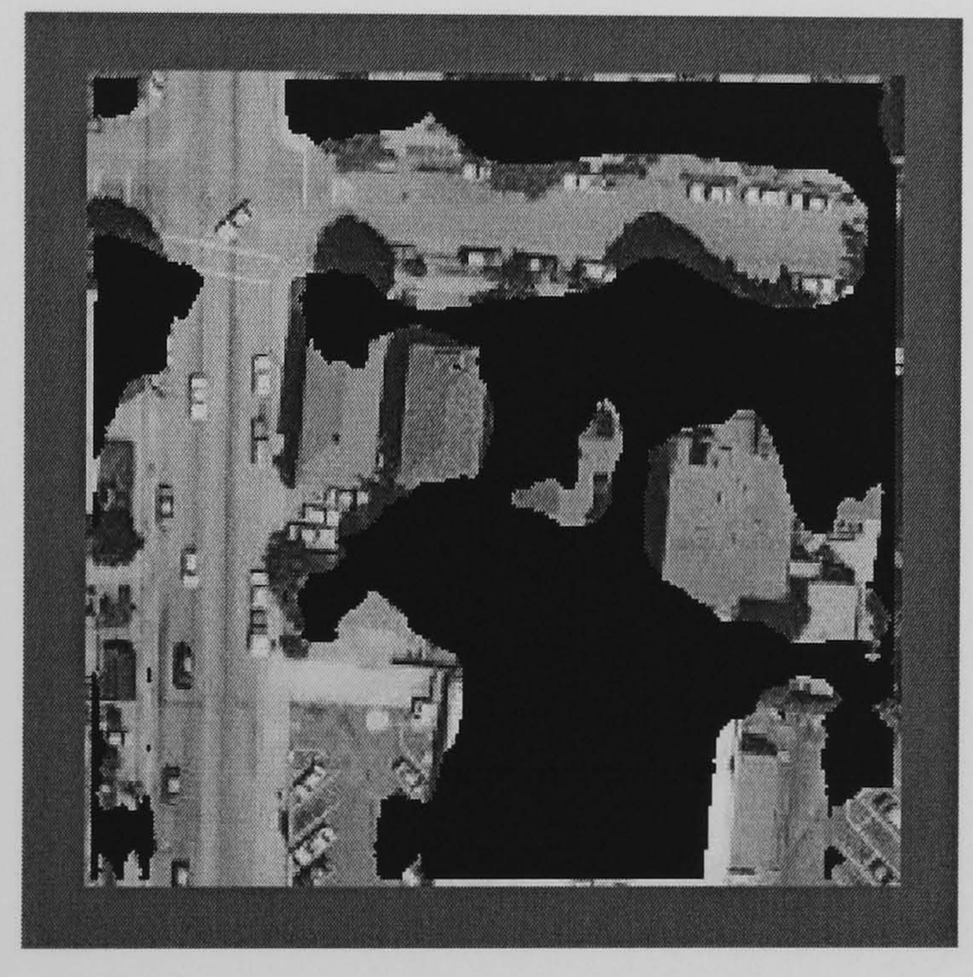


SGLDM /BPNN (82.61 %)

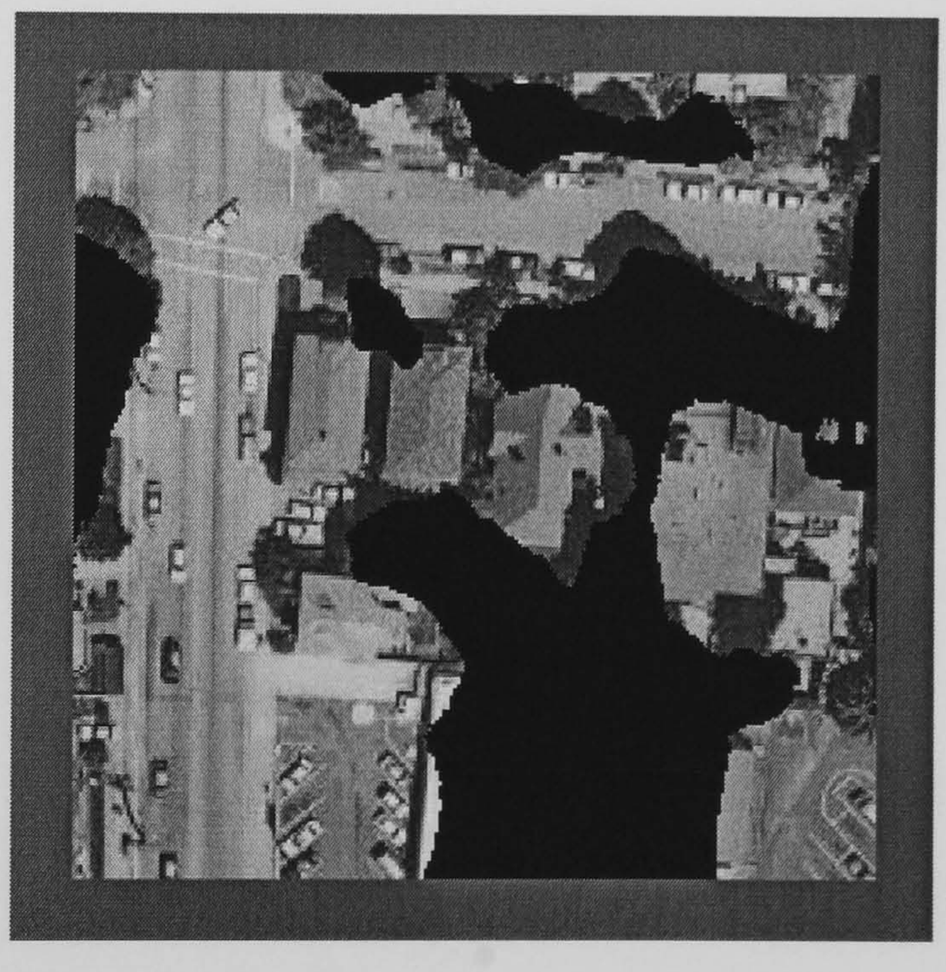
Figure E.17 Aerial Image 19 Results

Aerial Image 20

Hand Segmented Image



Hybrid Neural Network (83.05 %)



SGLDM /BPNN (76.85 %)

Figure E.18 Aerial Image 20 Results

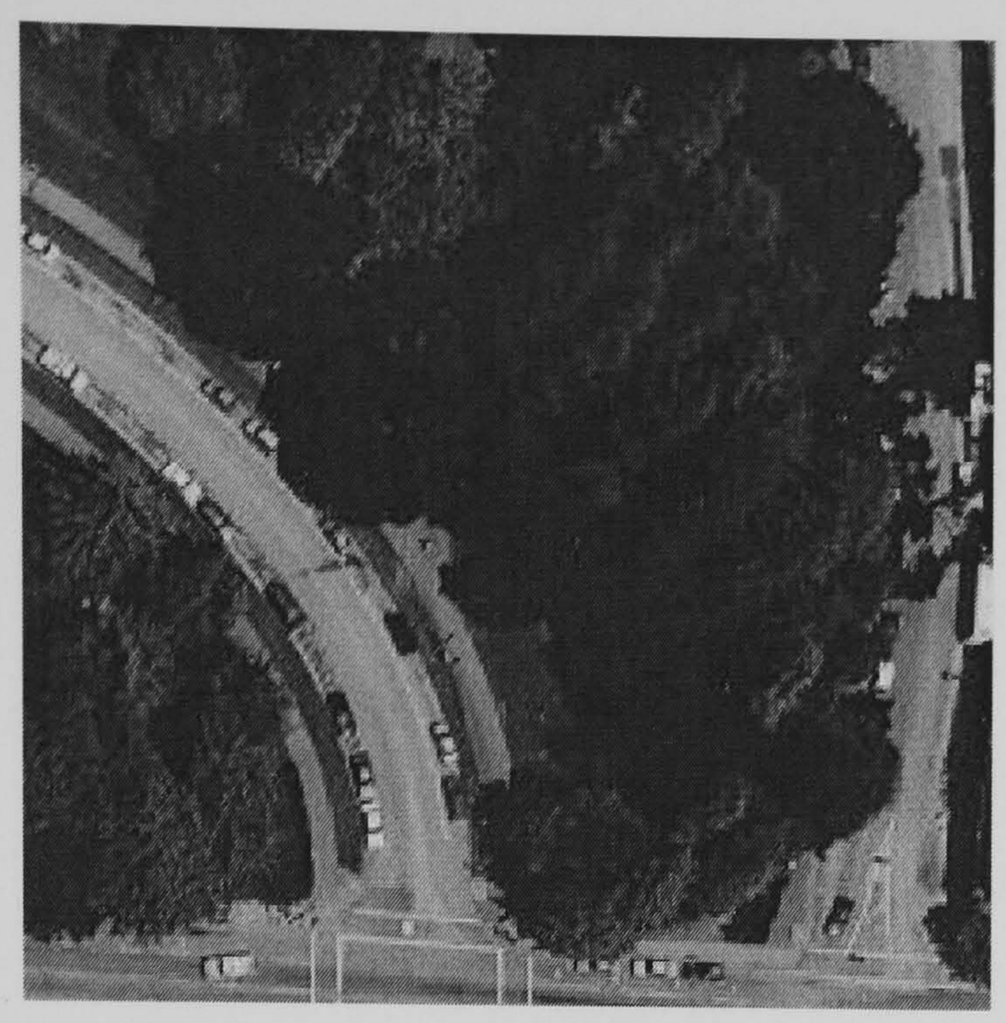
Aerial Image 21

Hand Segmented Image

Hybrid Neural Network (86.49 %)

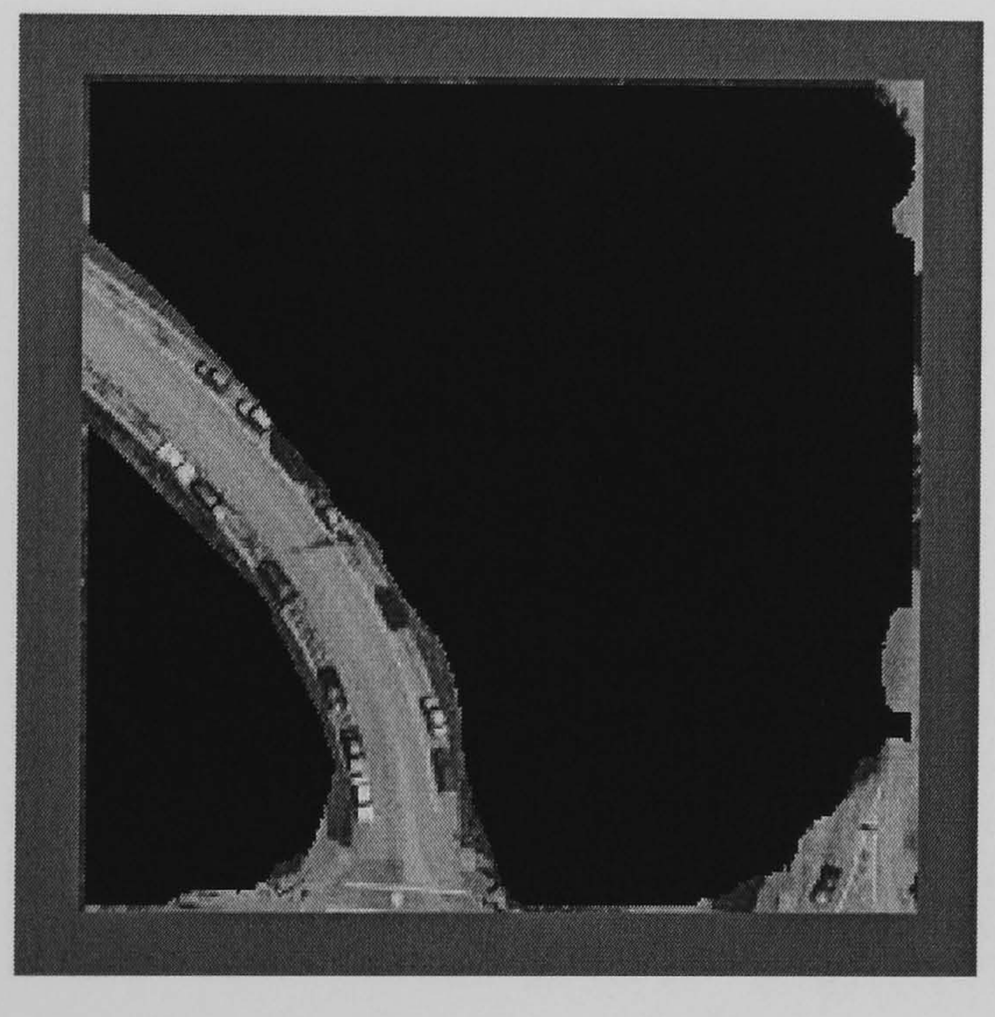
SGLDM /BPNN (85.85 %)

Figure E.19 Aerial Image 21 Results

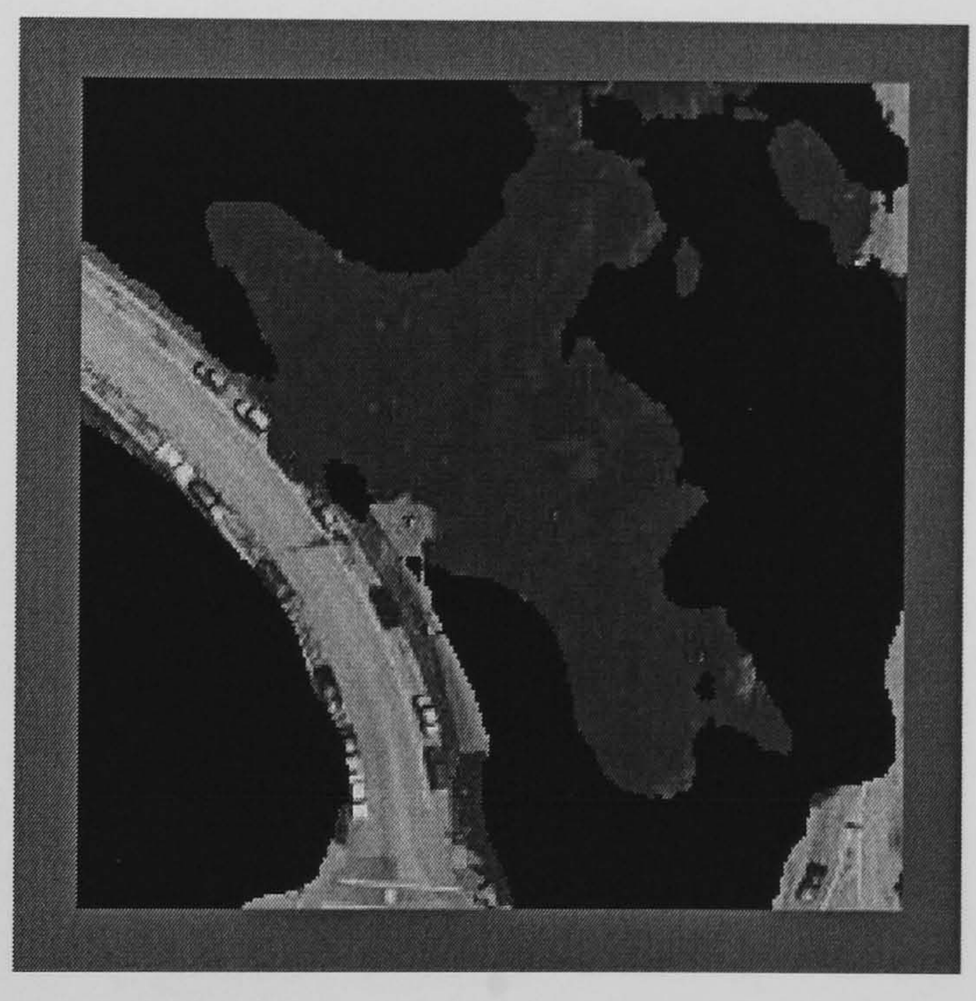


Aerial Image 22

Hand Segmented Image

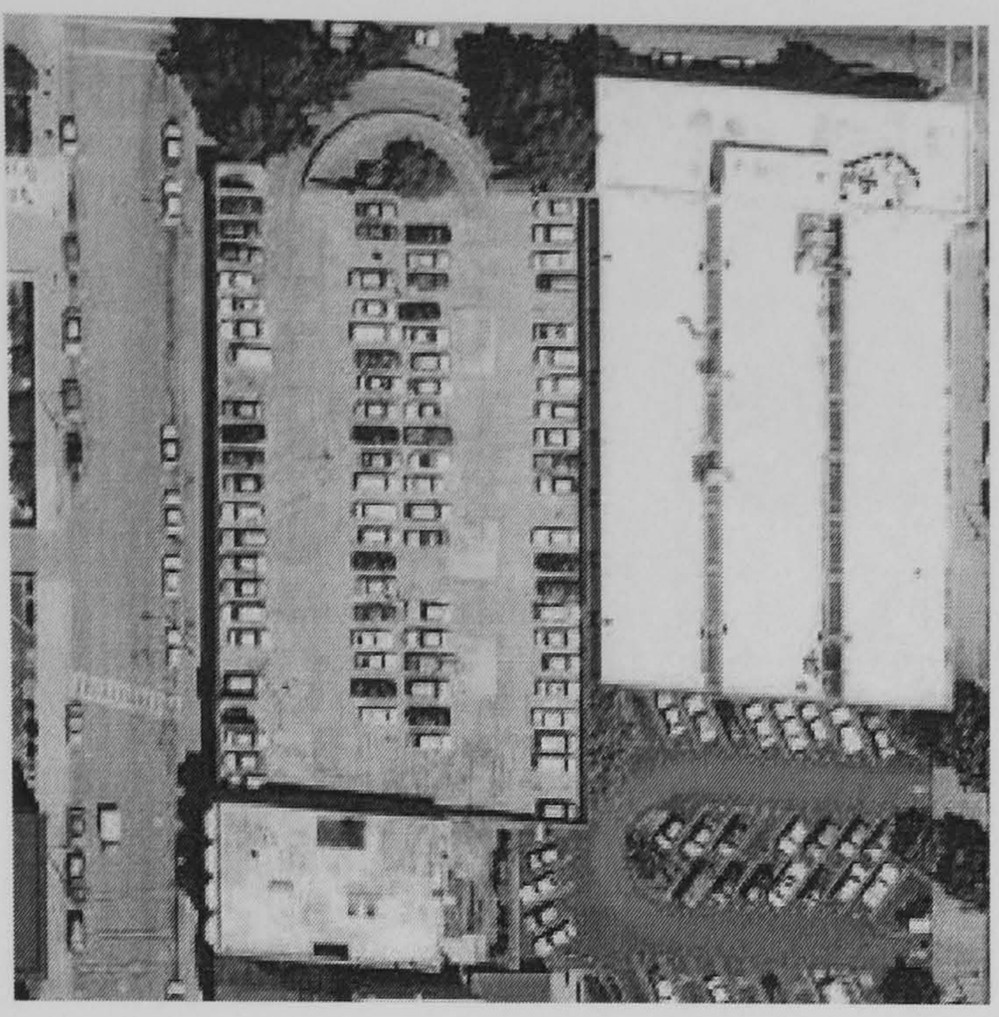


Hybrid Neural Network (95.95 %)



SGLDM /BPNN (75.10 %)

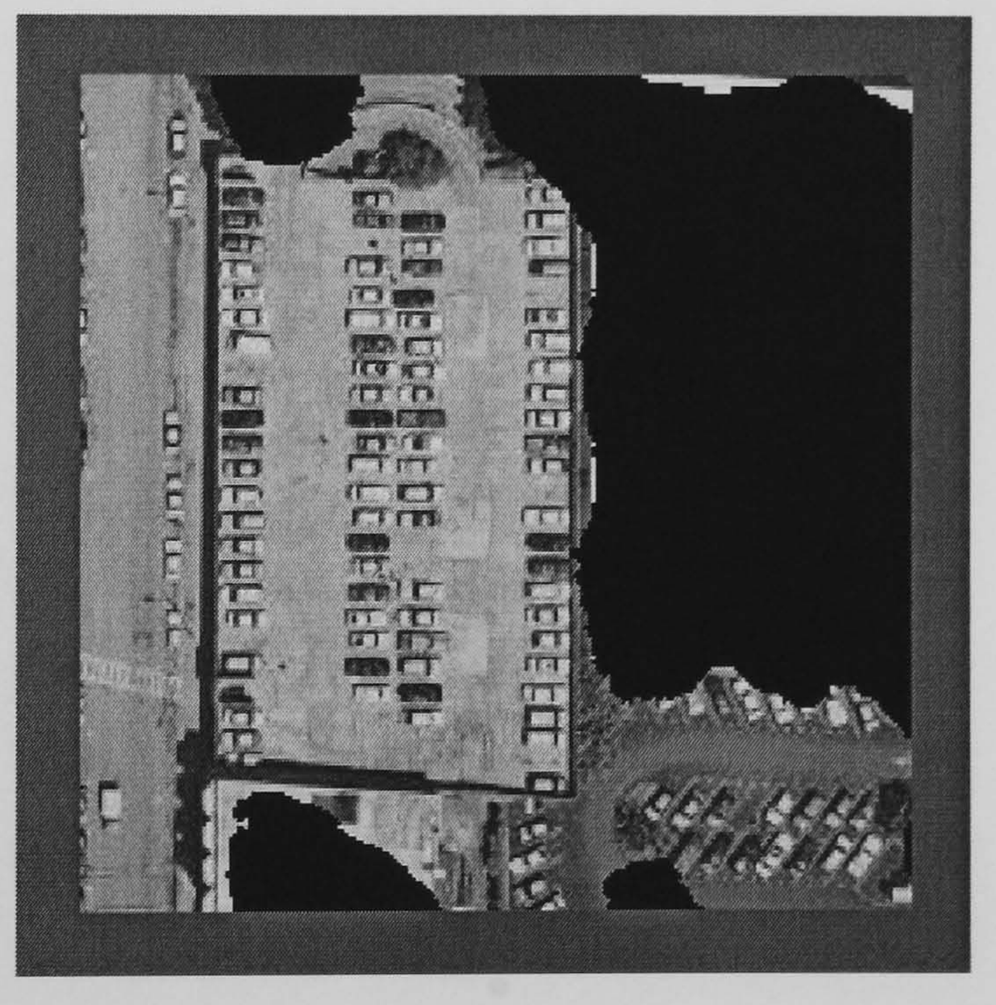
Figure E.20 Aerial Image 22 Results



Aerial Image 23

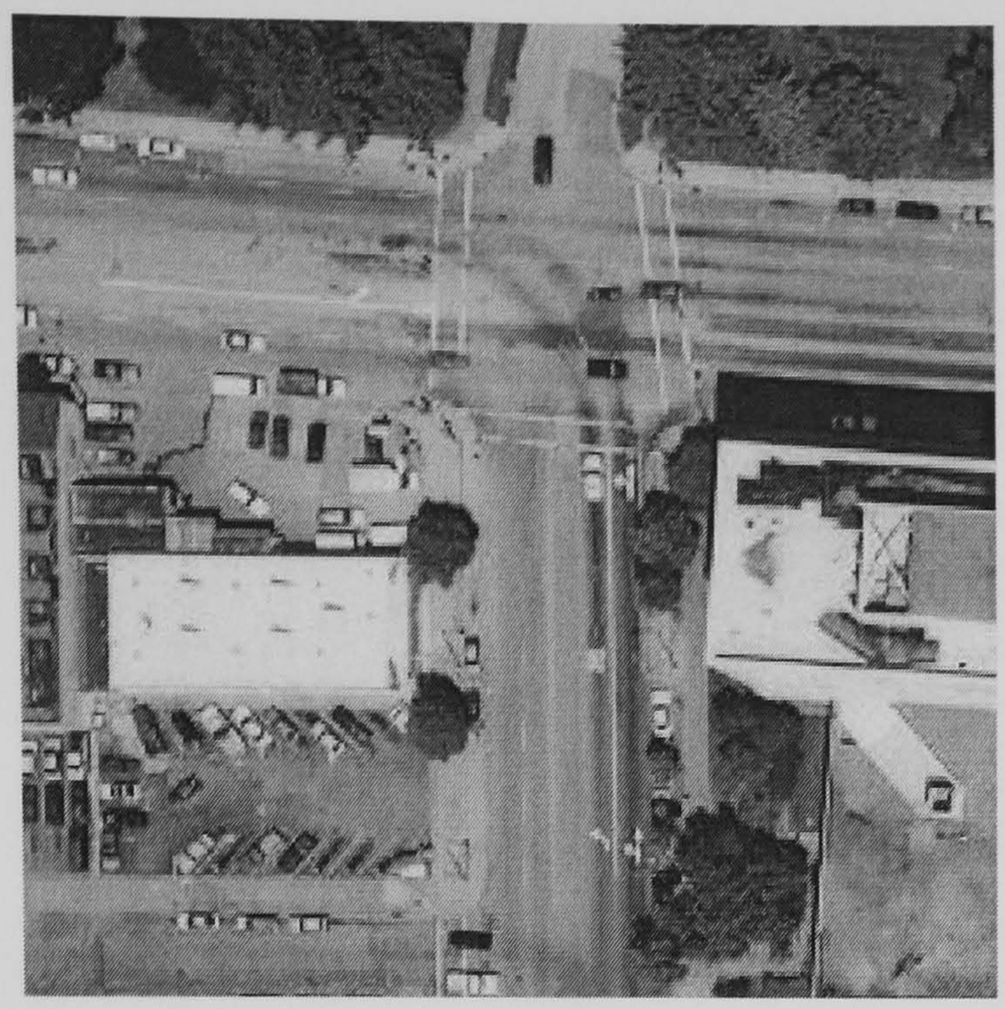
Hand Segmented Image

Hybrid Neural Network (86.00 %)



SGLDM /BPNN (95.28 %)

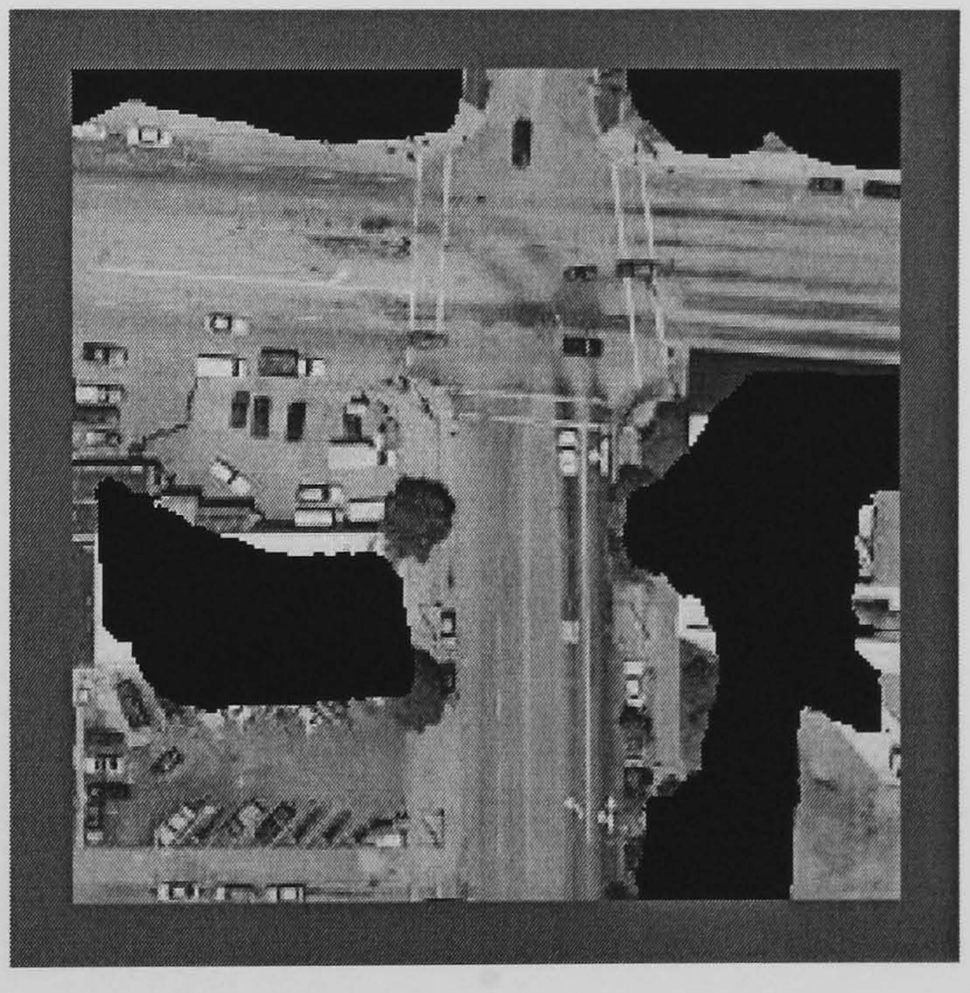
Figure E.21 Aerial Image 23 Results



Aerial Image 24

Hand Segmented Image

Hybrid Neural Network (83.60 %)

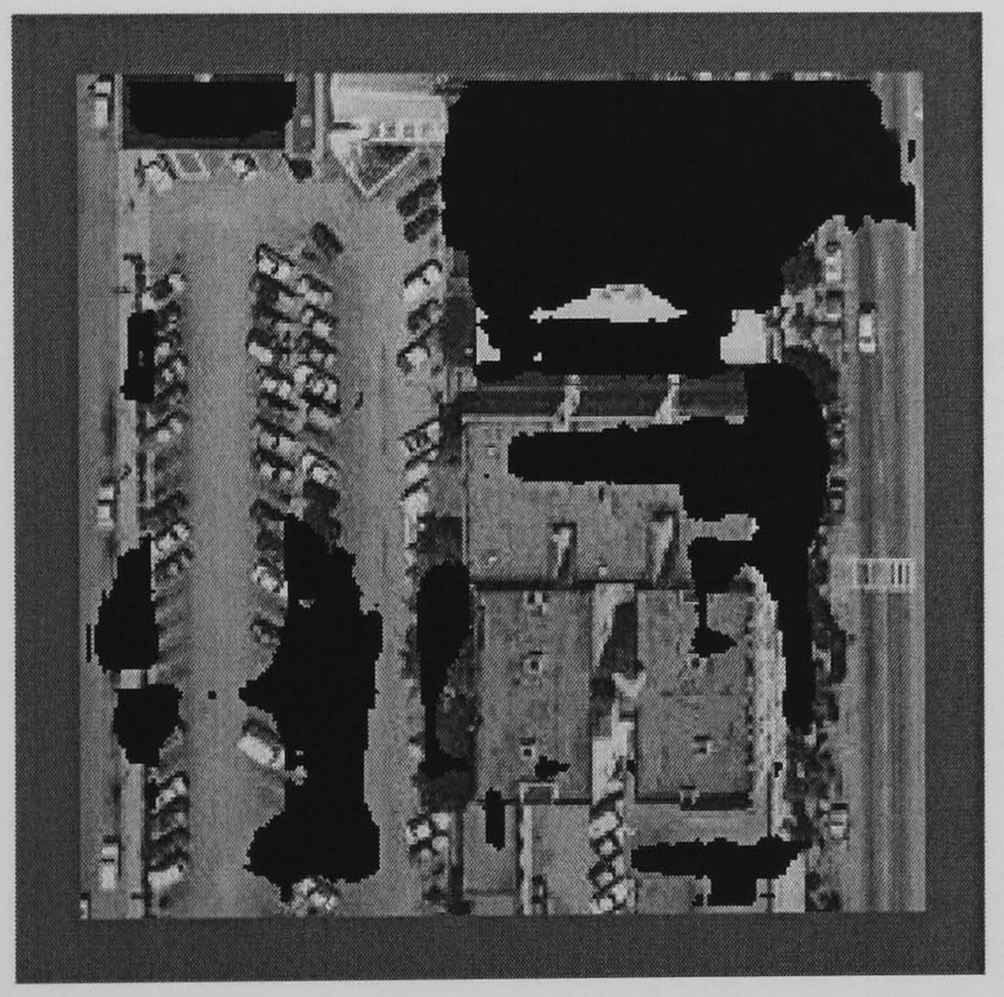


SGLDM /BPNN (90.90 %)

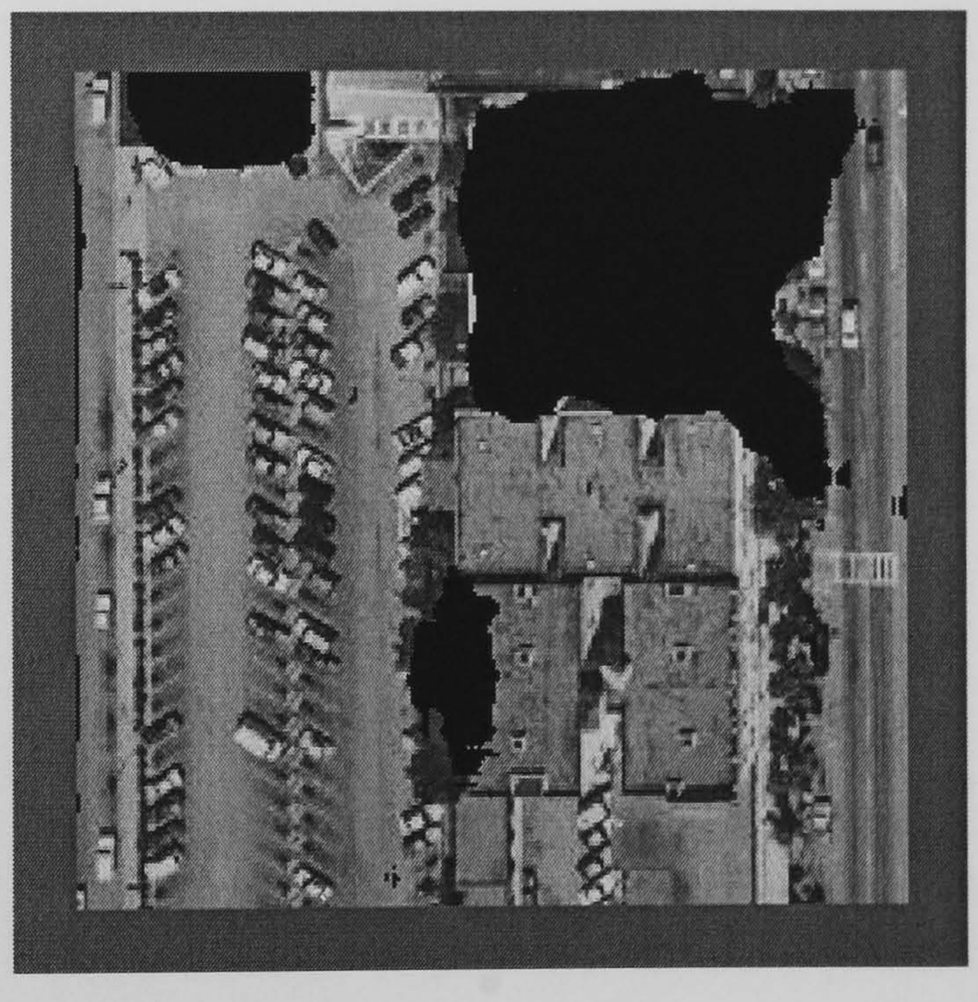
Figure E.22 Aerial Image 24 Results

Aerial Image 25

Hand Segmented Image

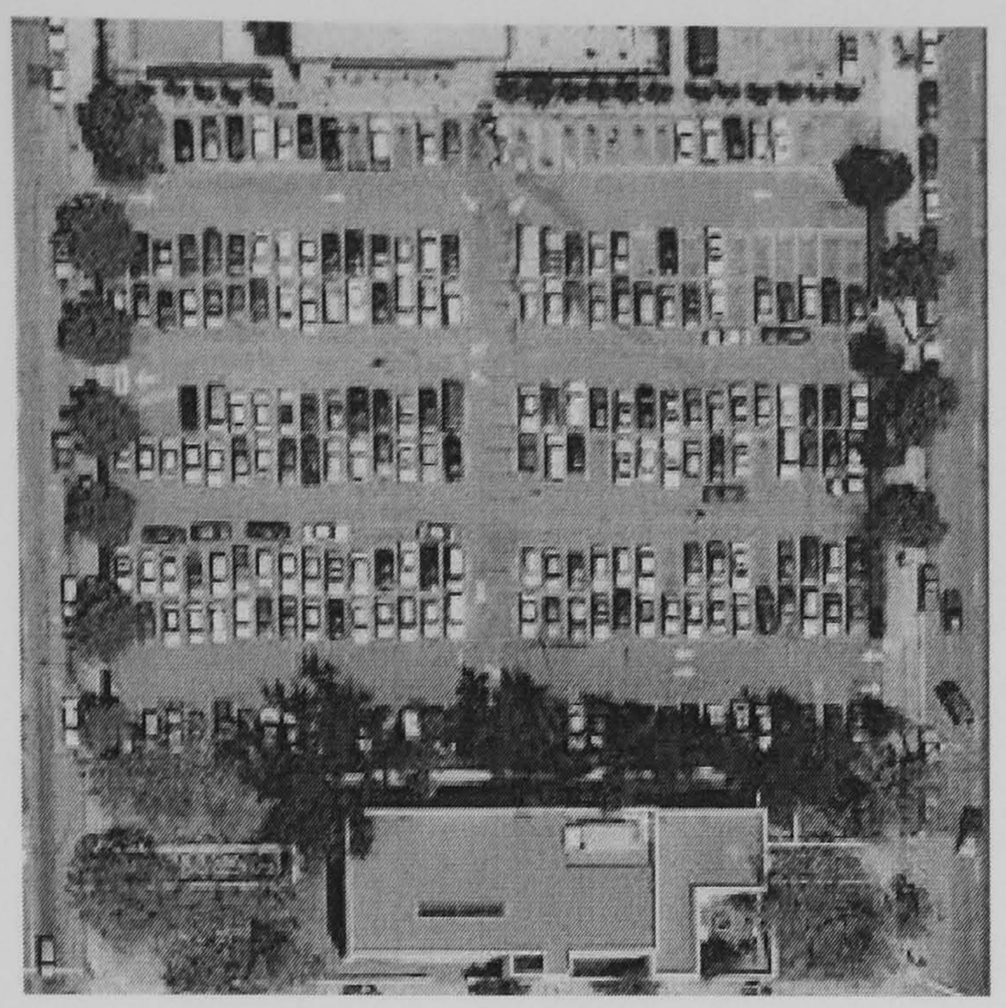


Hybrid Neural Network (90.90 %)

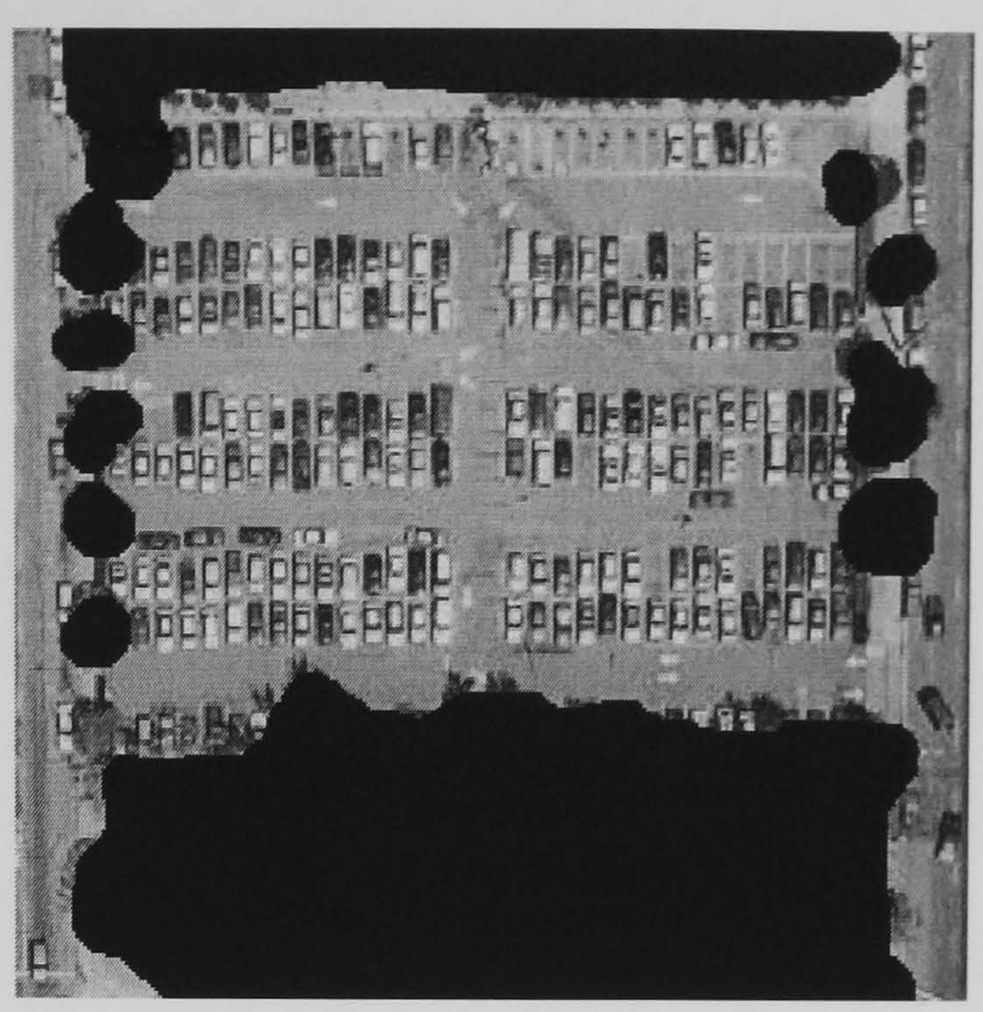


SGLDM /BPNN (80.83 %)

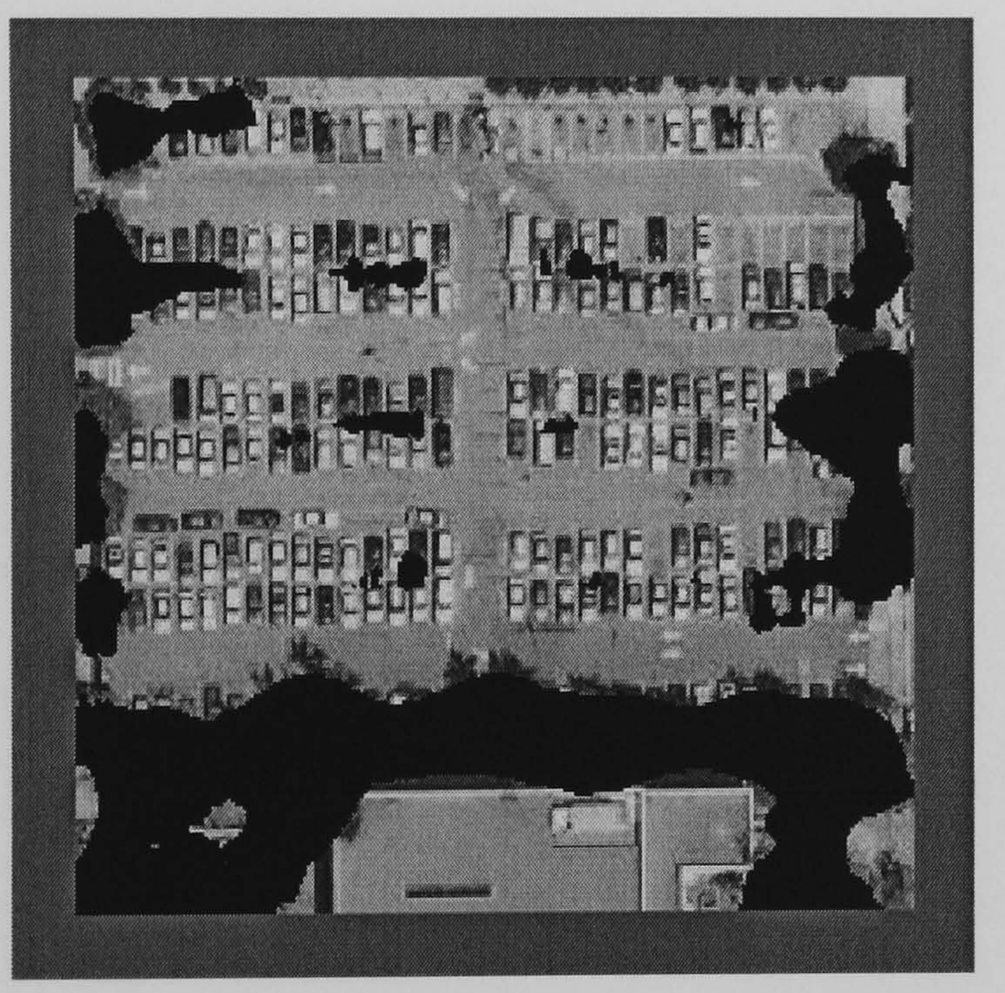
Figure E.23 Aerial Image 25 Results



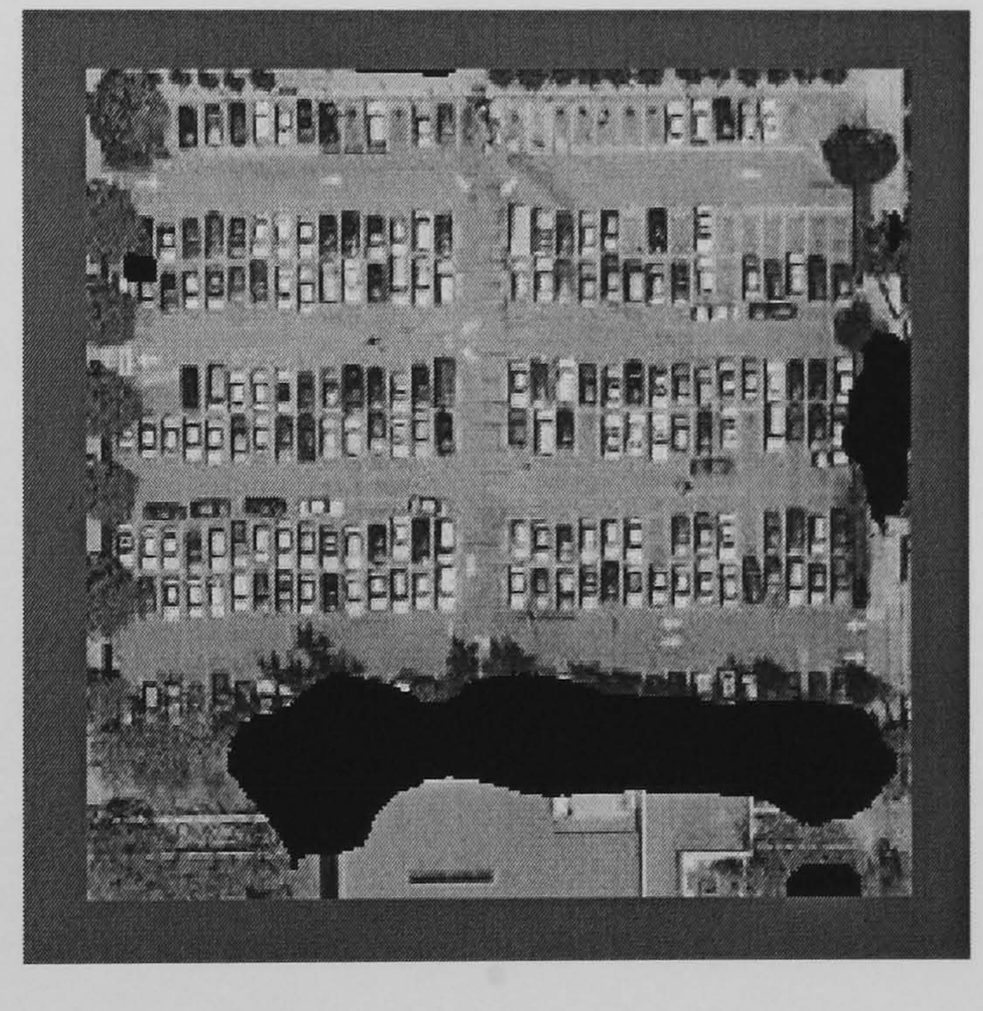
Aerial Image 26



Hand Segmented Image

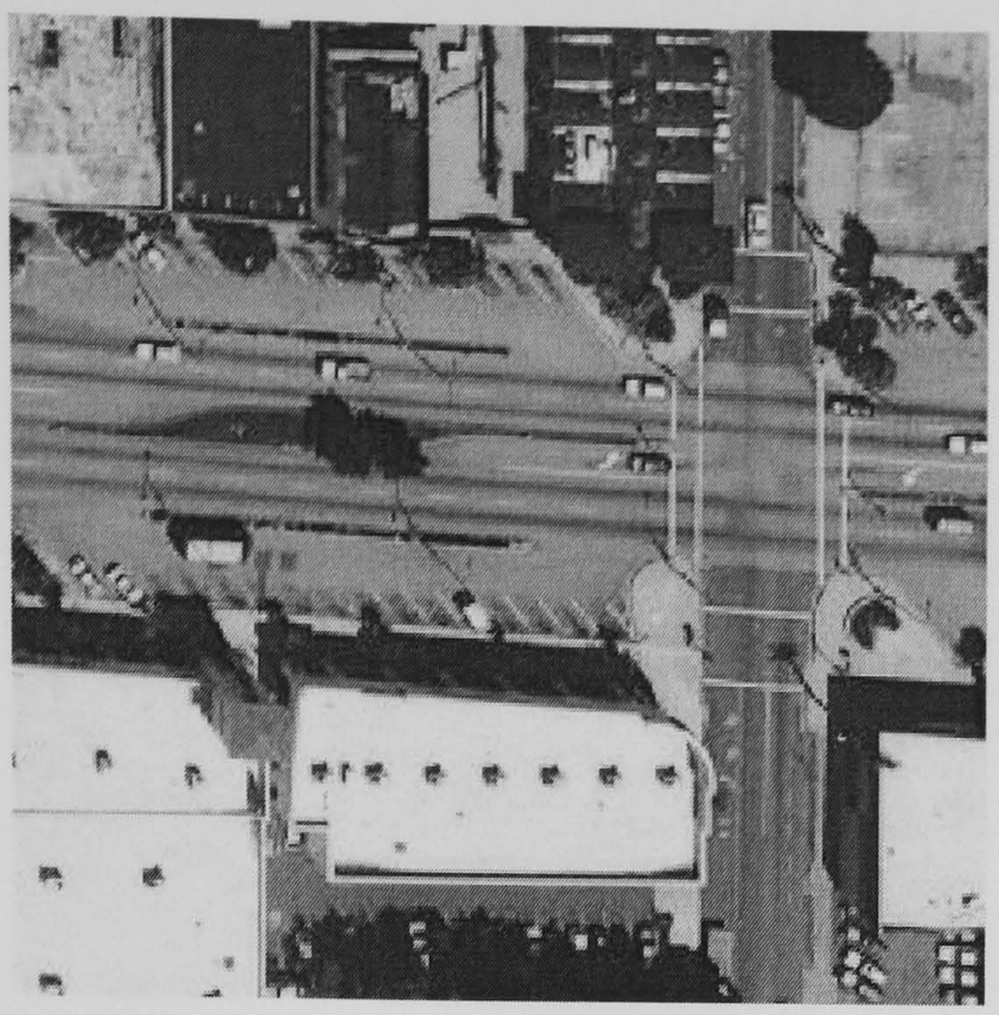


Hybrid Neural Network (85.70 %)

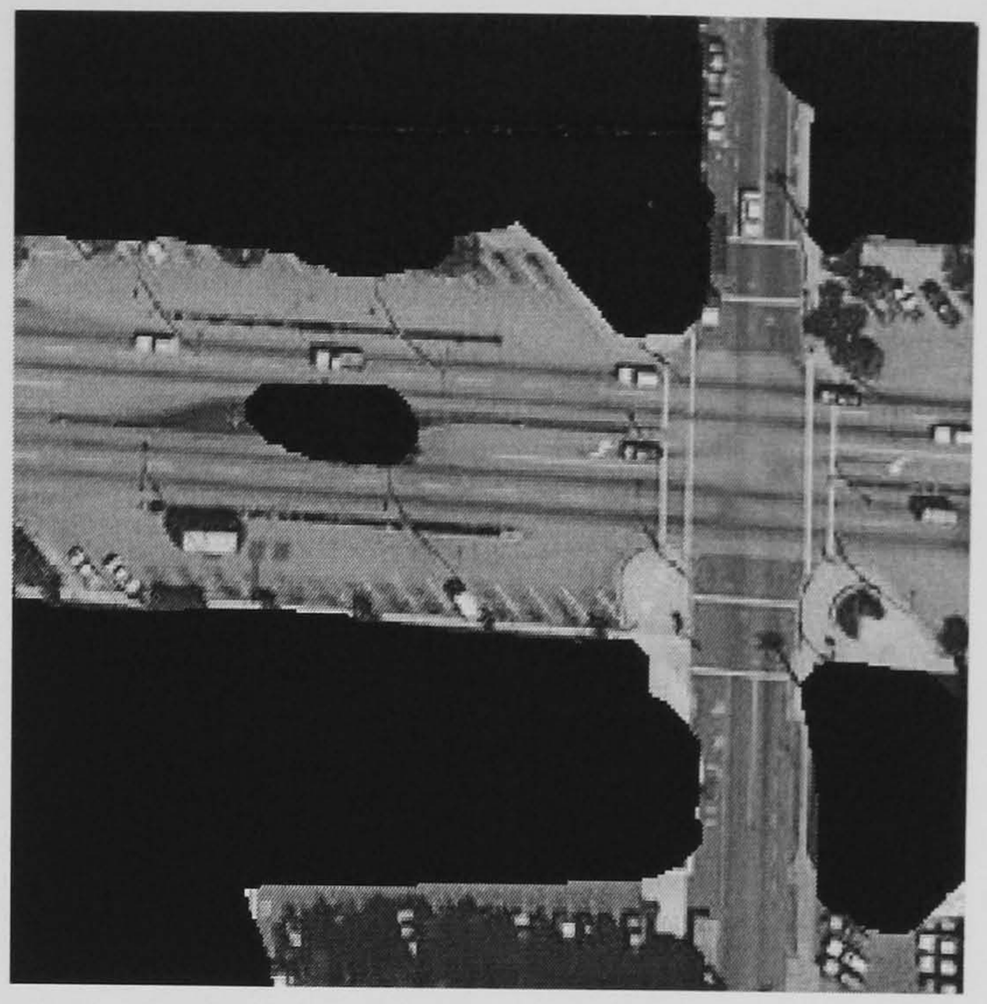


SGLDM /BPNN (80.83 %)

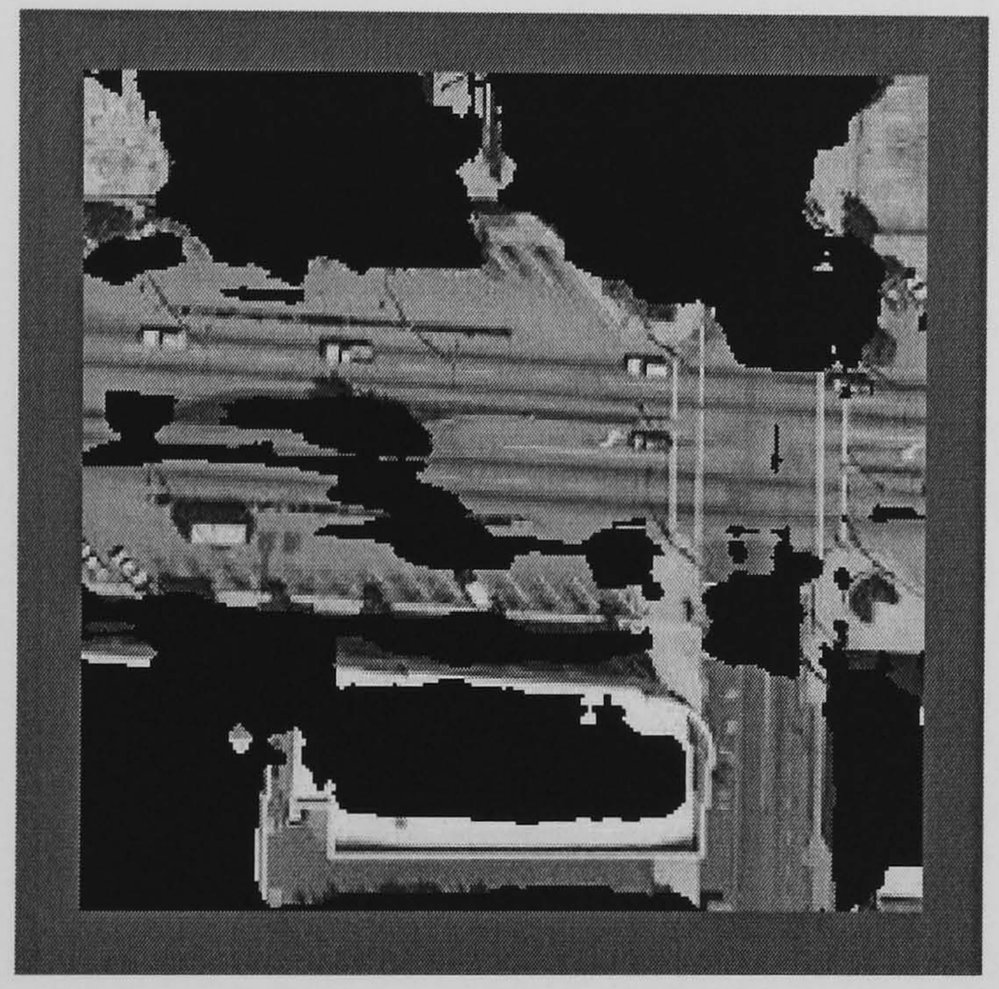
Figure E.24 Aerial Image 26 Results



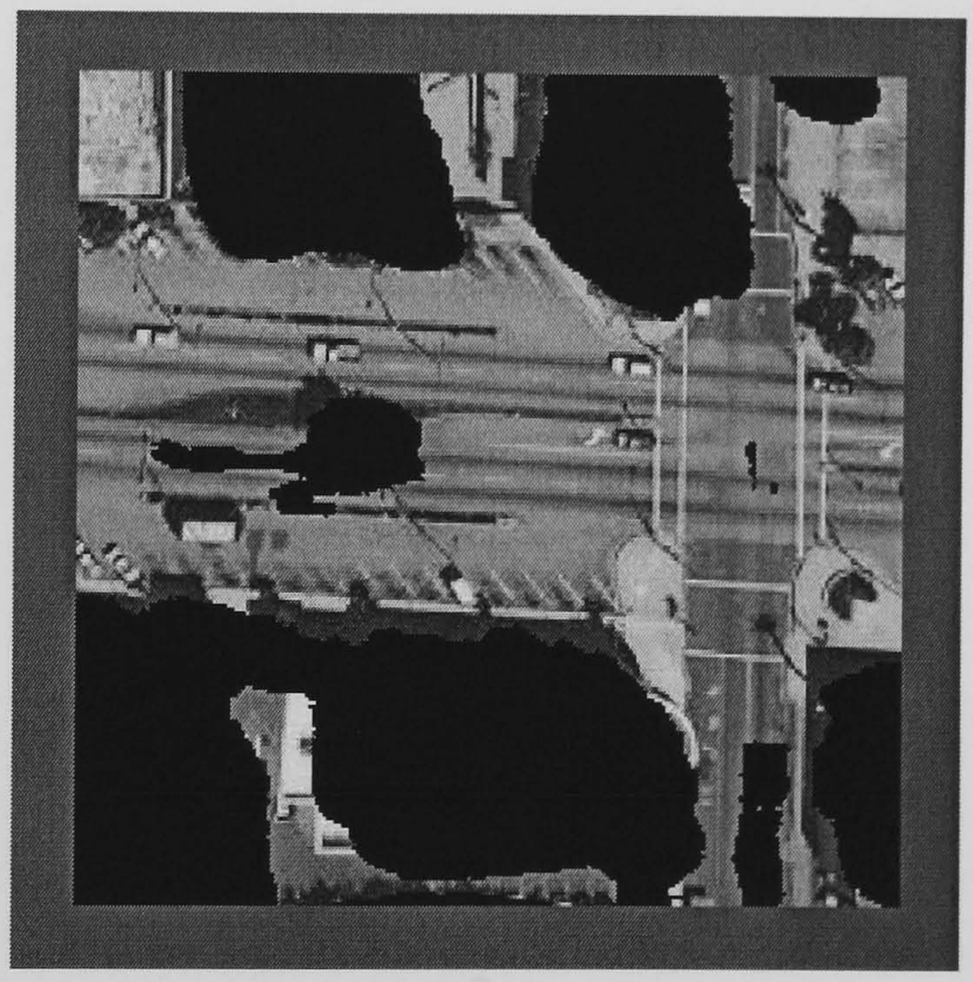
Aerial Image 27



Hand Segmented Image

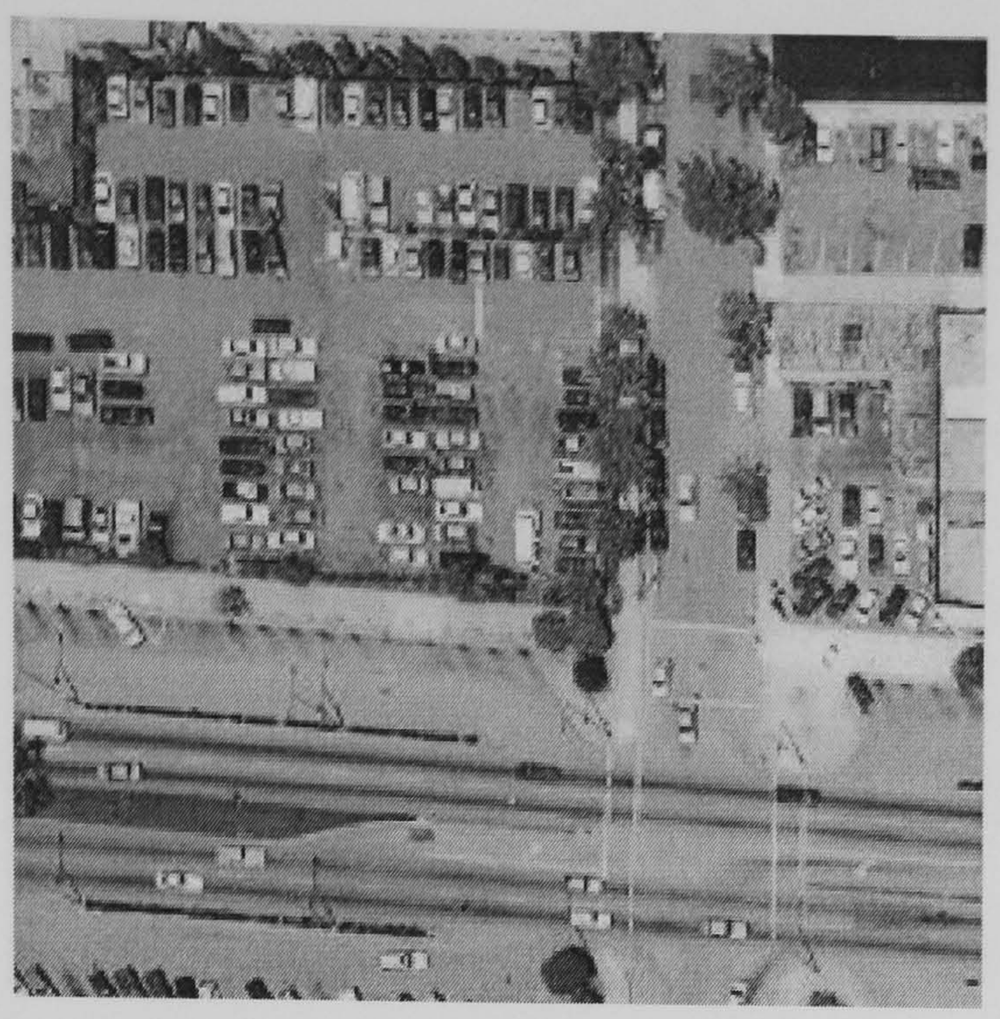


Hybrid Neural Network (79.89 %)

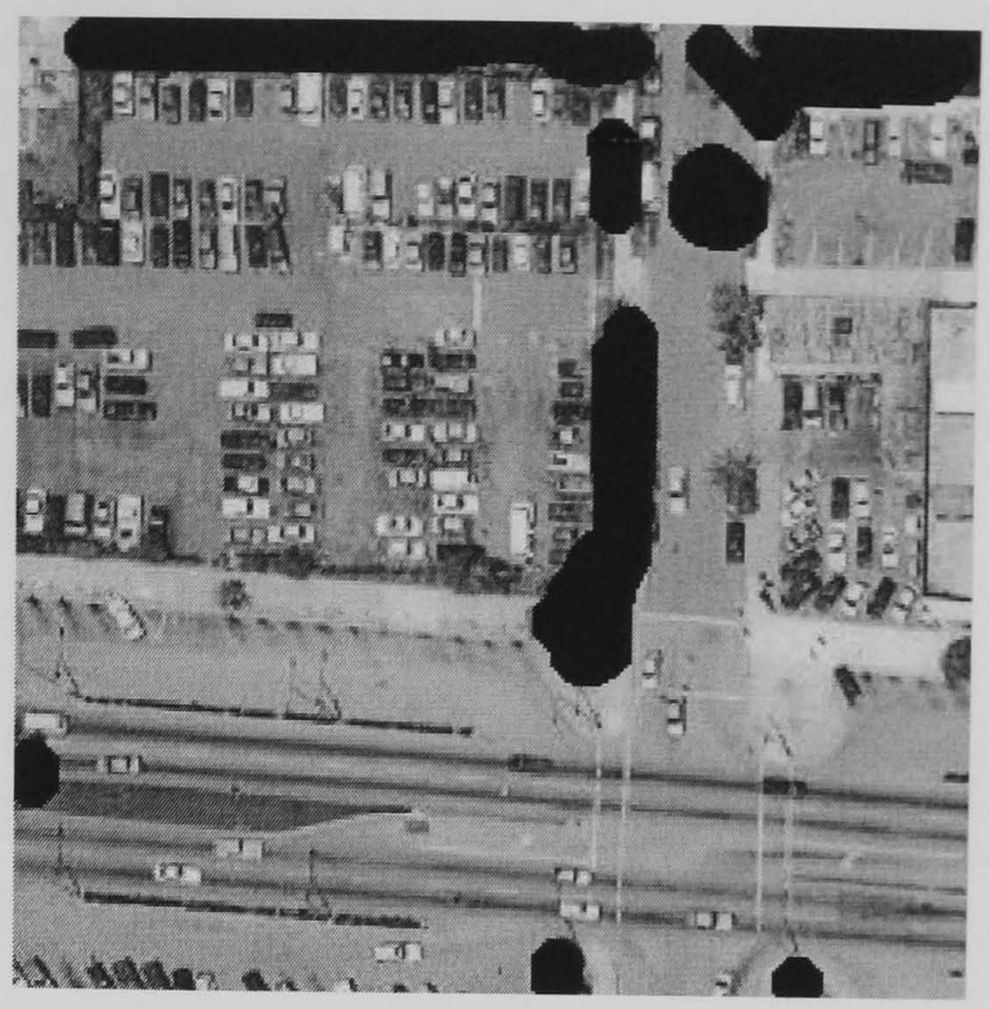


SGLDM /BPNN (86.16 %)

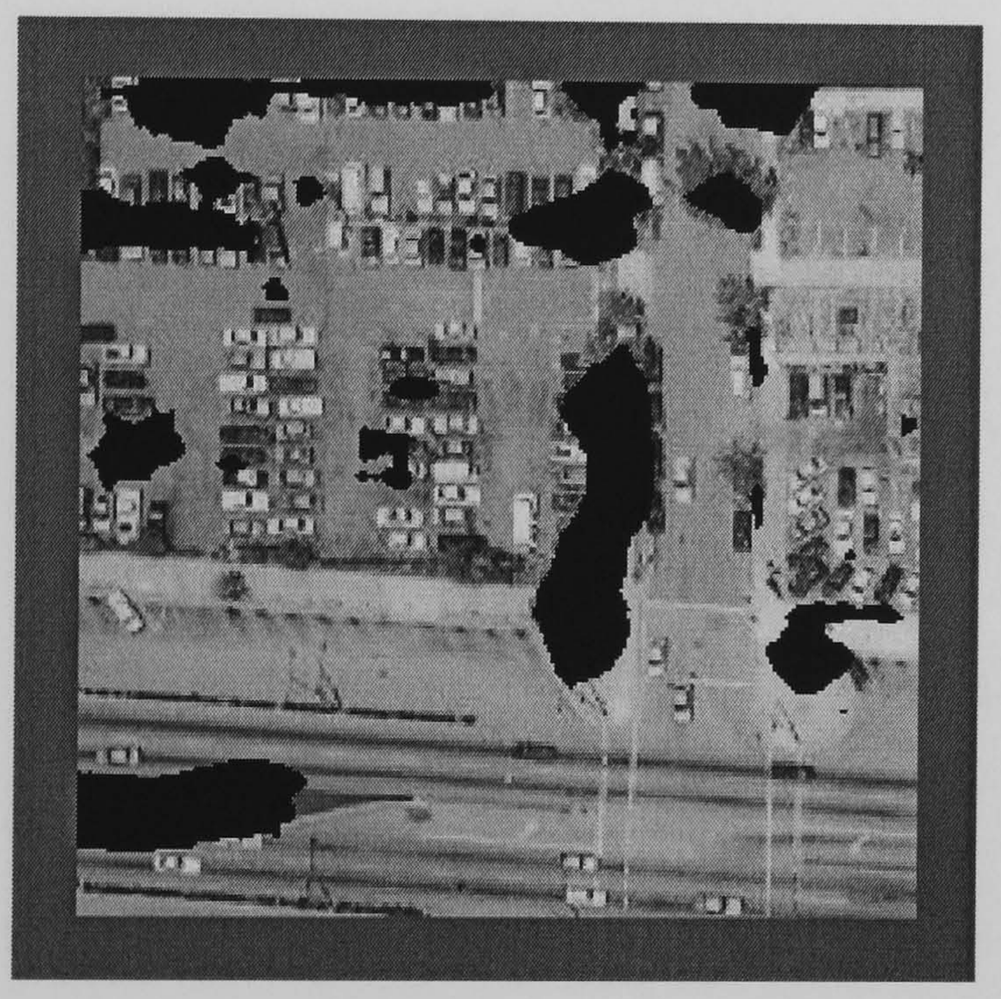
Figure E.25 Aerial Image 27 Results



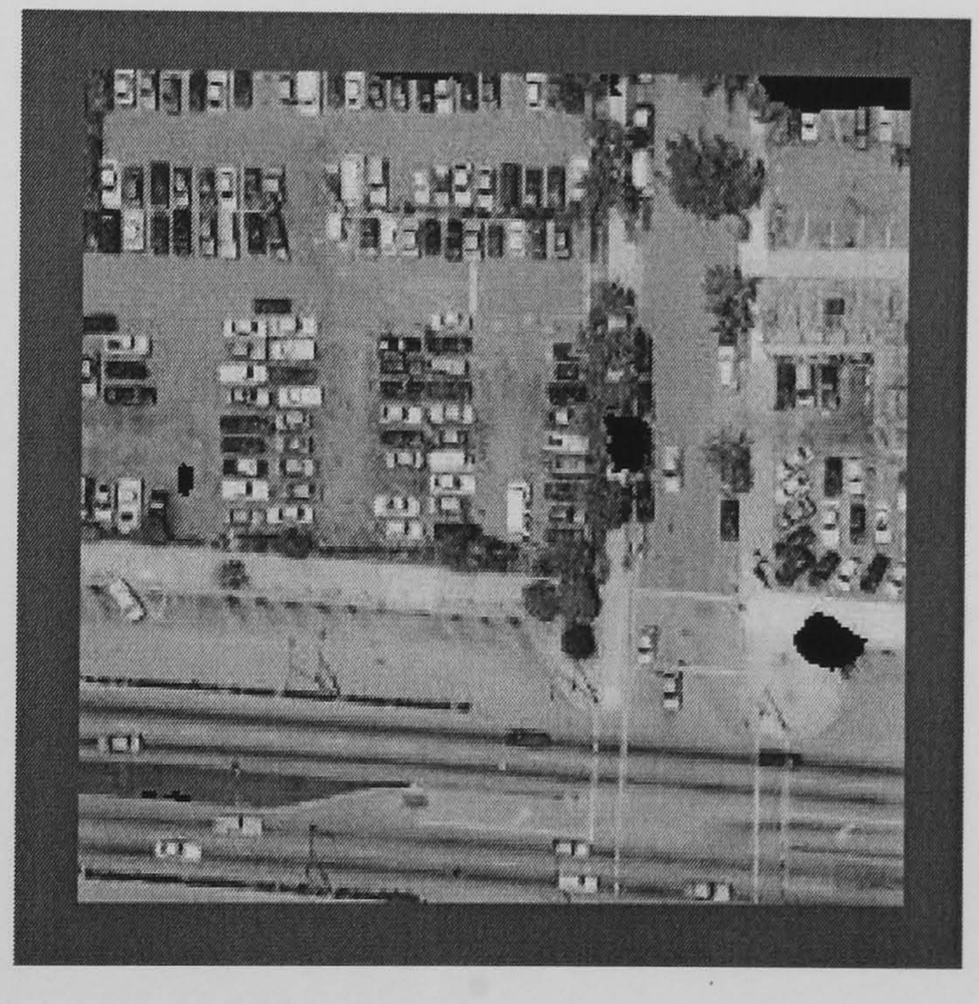
Aerial Image 28



Hand Segmented Image



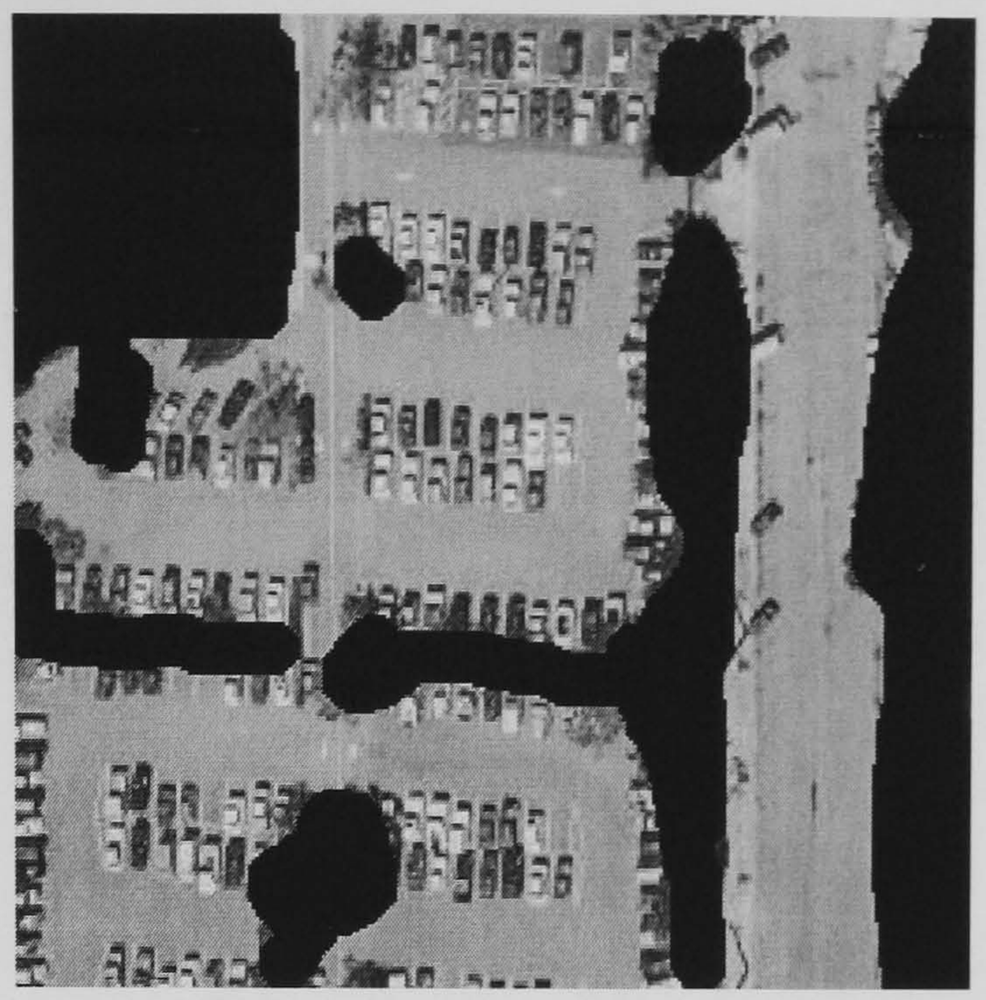
Hybrid Neural Network (90.55 %)



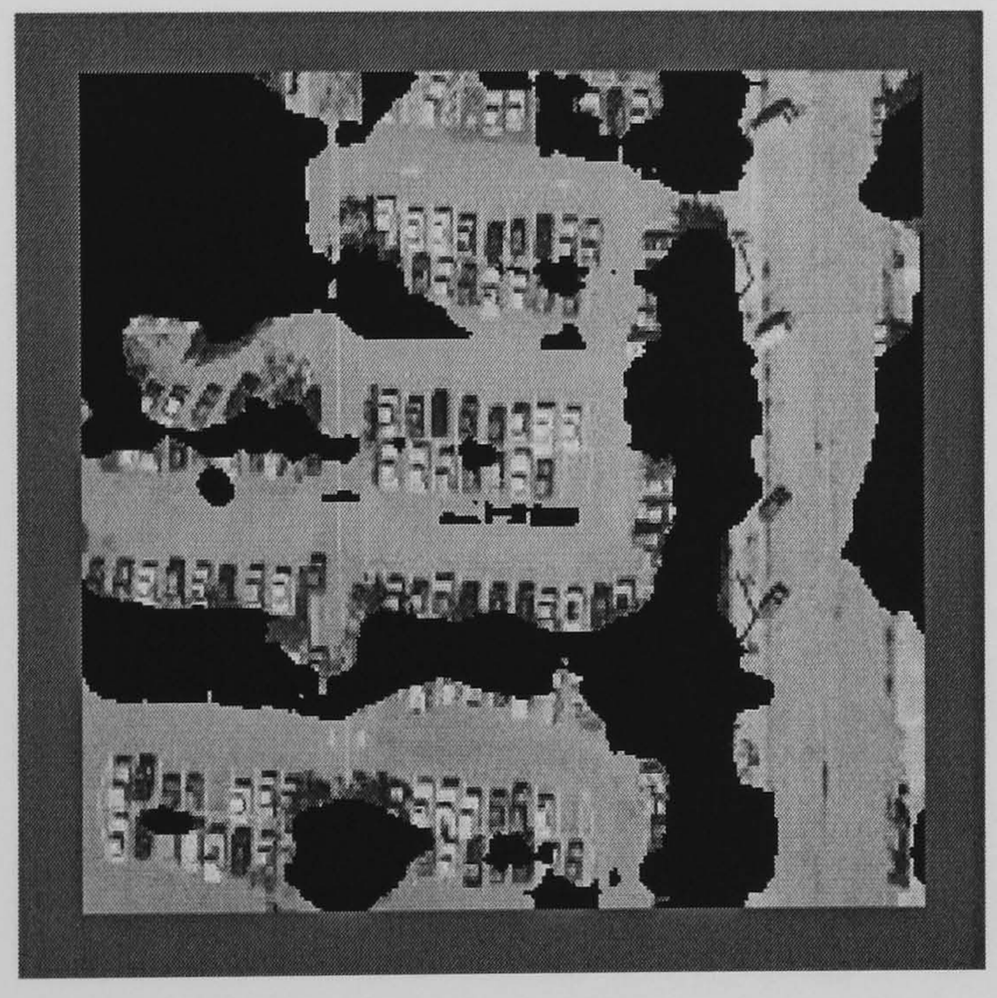
SGLDM /BPNN (94.84 %)

Figure E.26 Aerial Image 28 Results

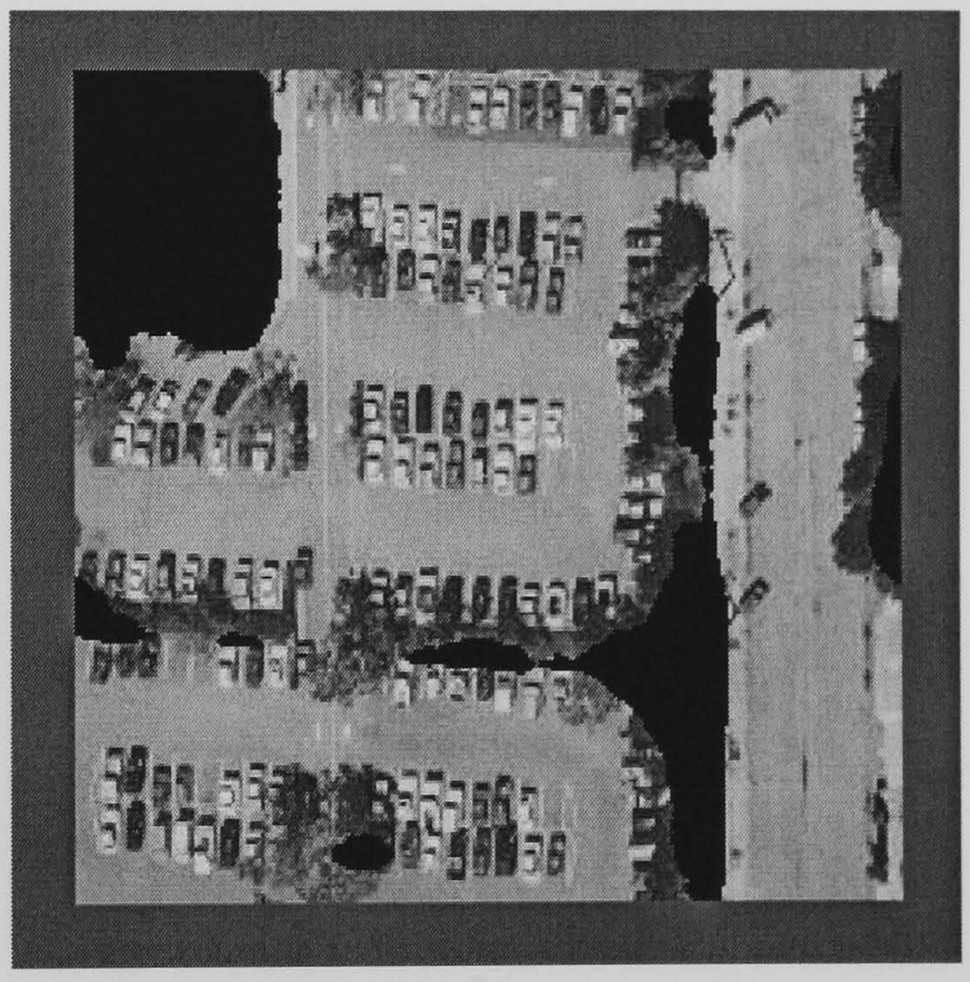
Aerial Image 29



Hand Segmented Image

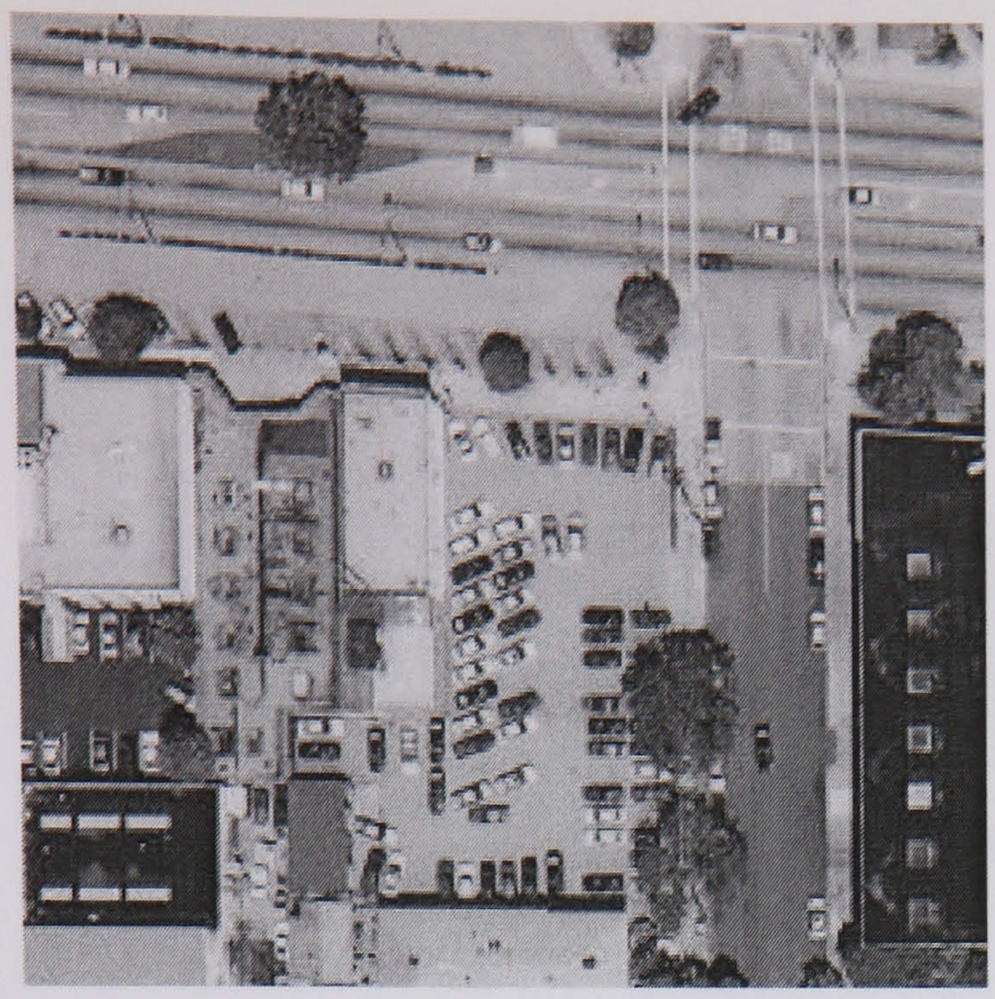


Hybrid Neural Network (88.02 %)

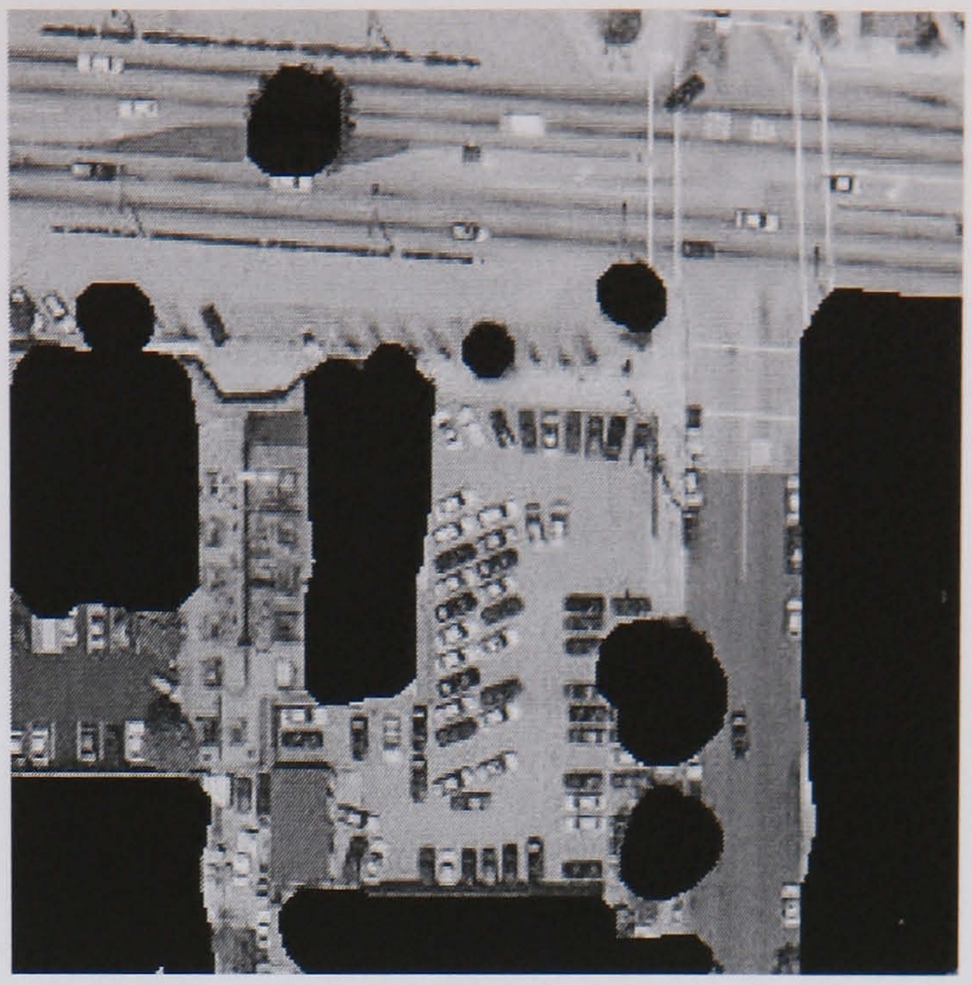


SGLDM /BPNN (87.46 %)

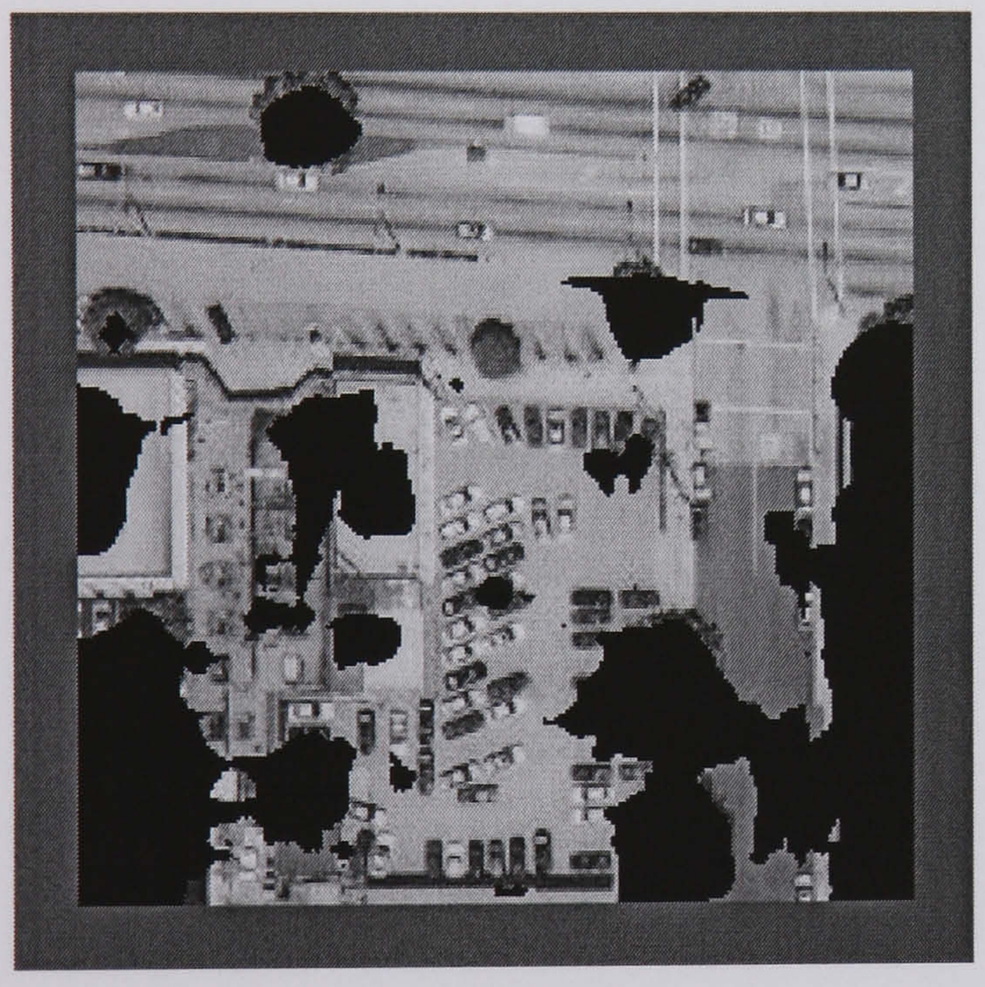
Figure E.27 Aerial Image 29 Results



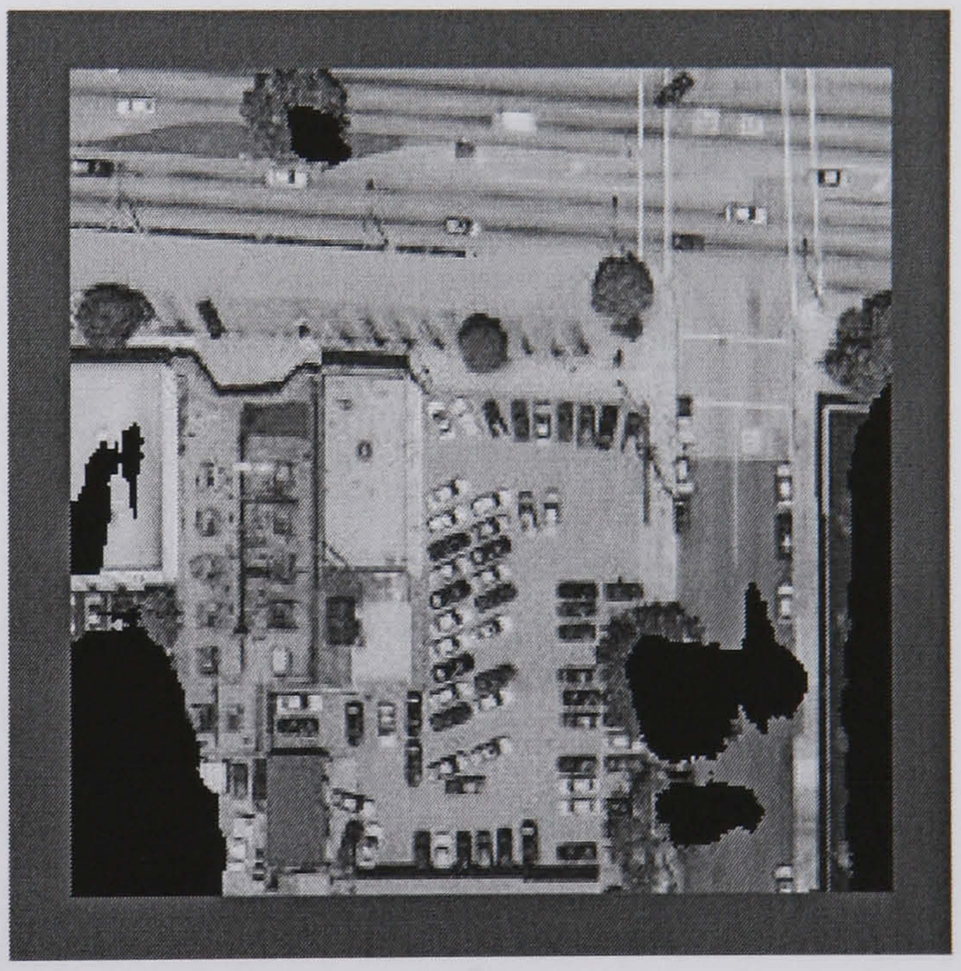
Aerial Image 30



Hand Segmented Image



Hybrid Neural Network (83.74 %)

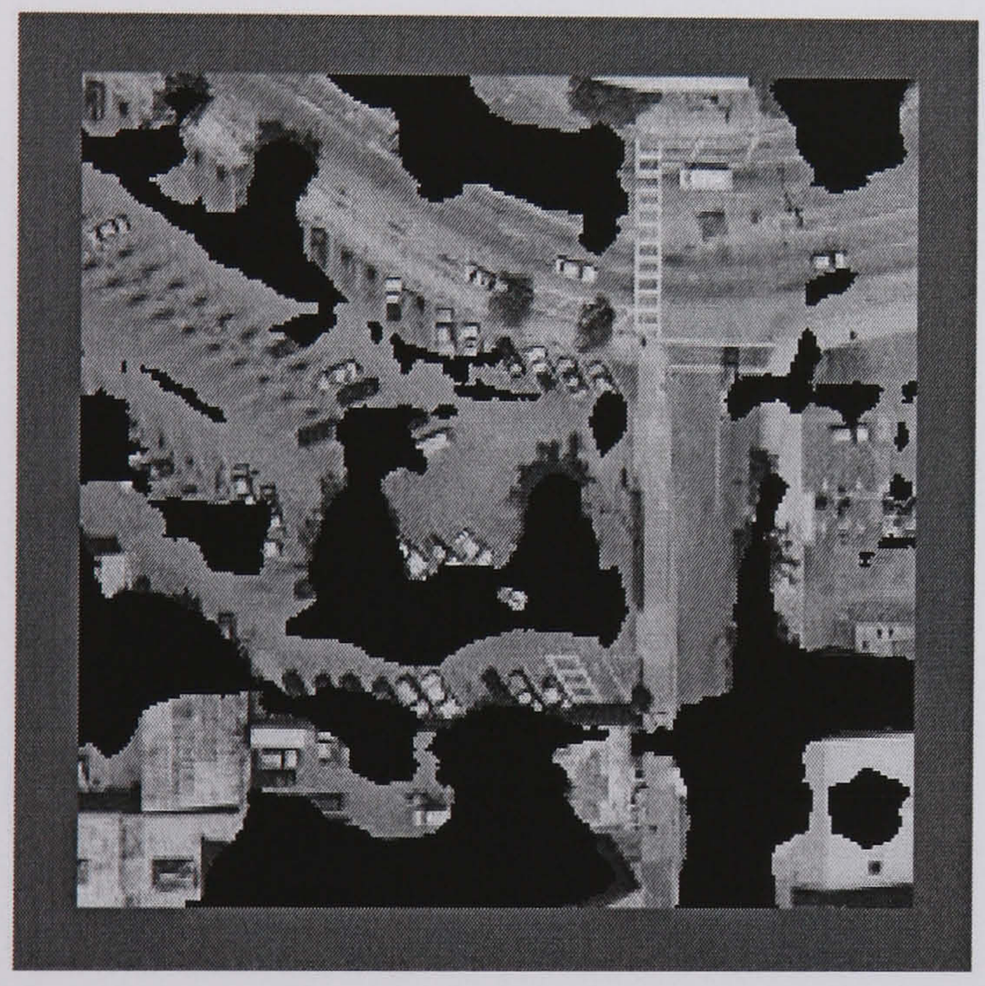


SGLDM /BPNN (81.63 %)

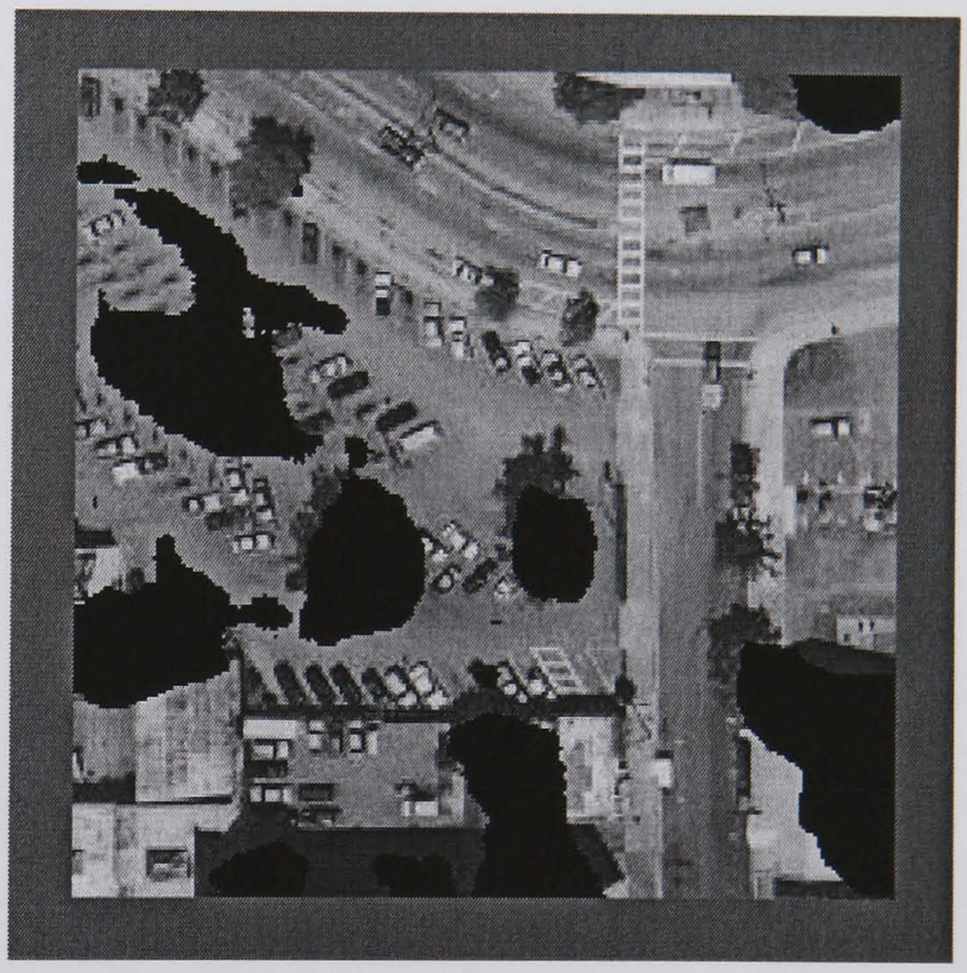
Figure E.28 Aerial Image 30 Results

Aerial Image 31

Hand Segmented Image



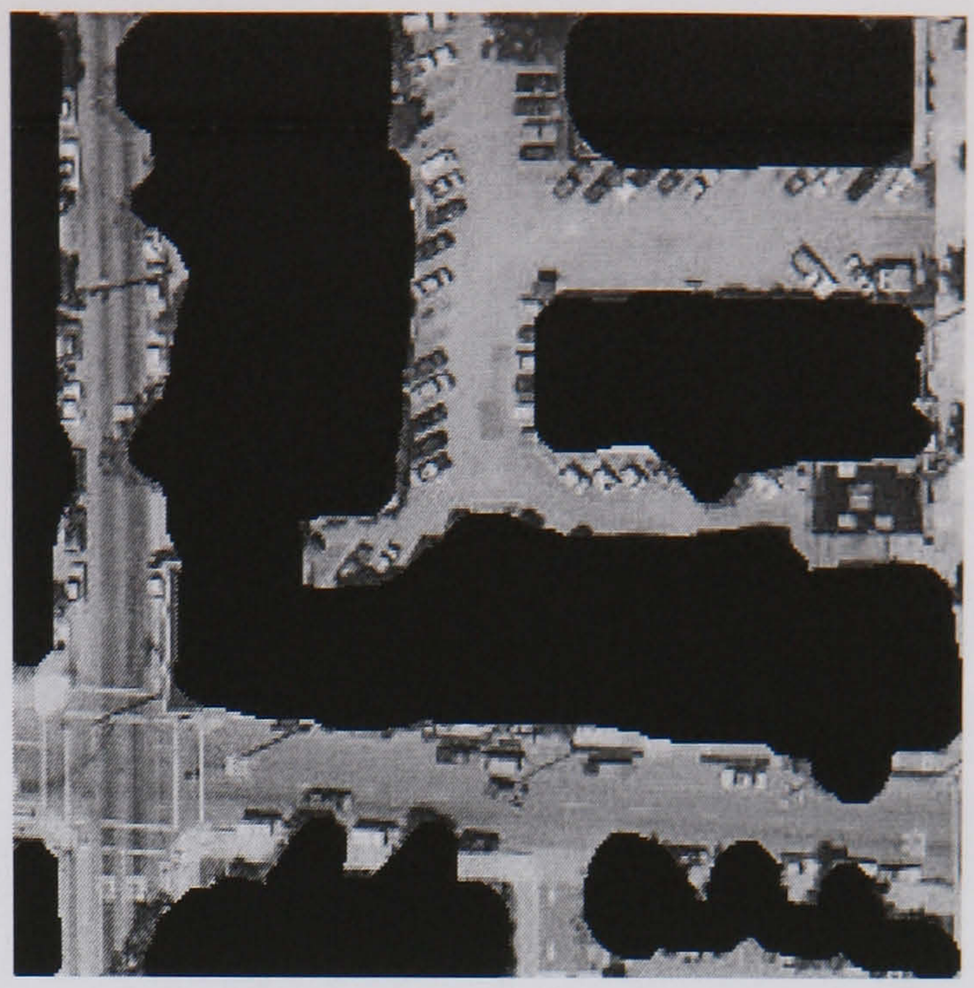
Hybrid Neural Network (76.17 %)



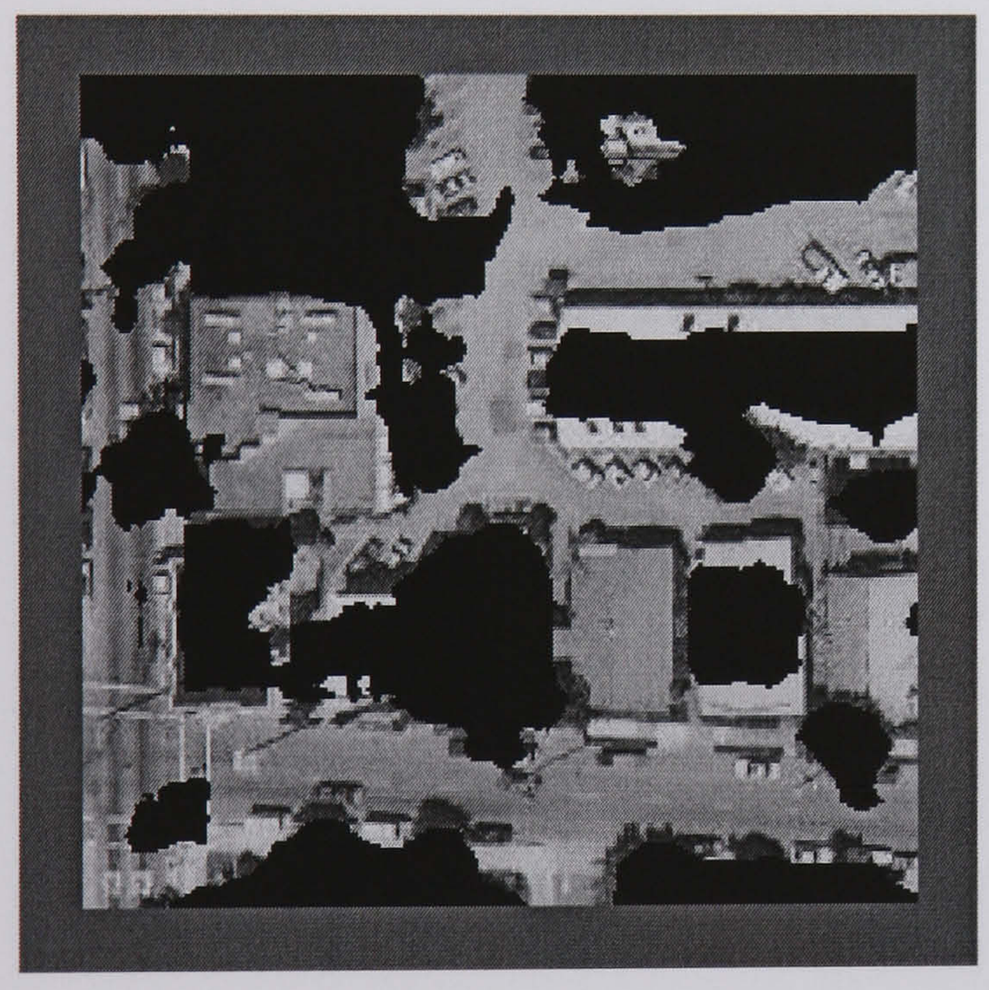
SGLDM /BPNN (81.94 %)

Figure E.29 Aerial Image 31 Results

Aerial Image 32



Hand Segmented Image

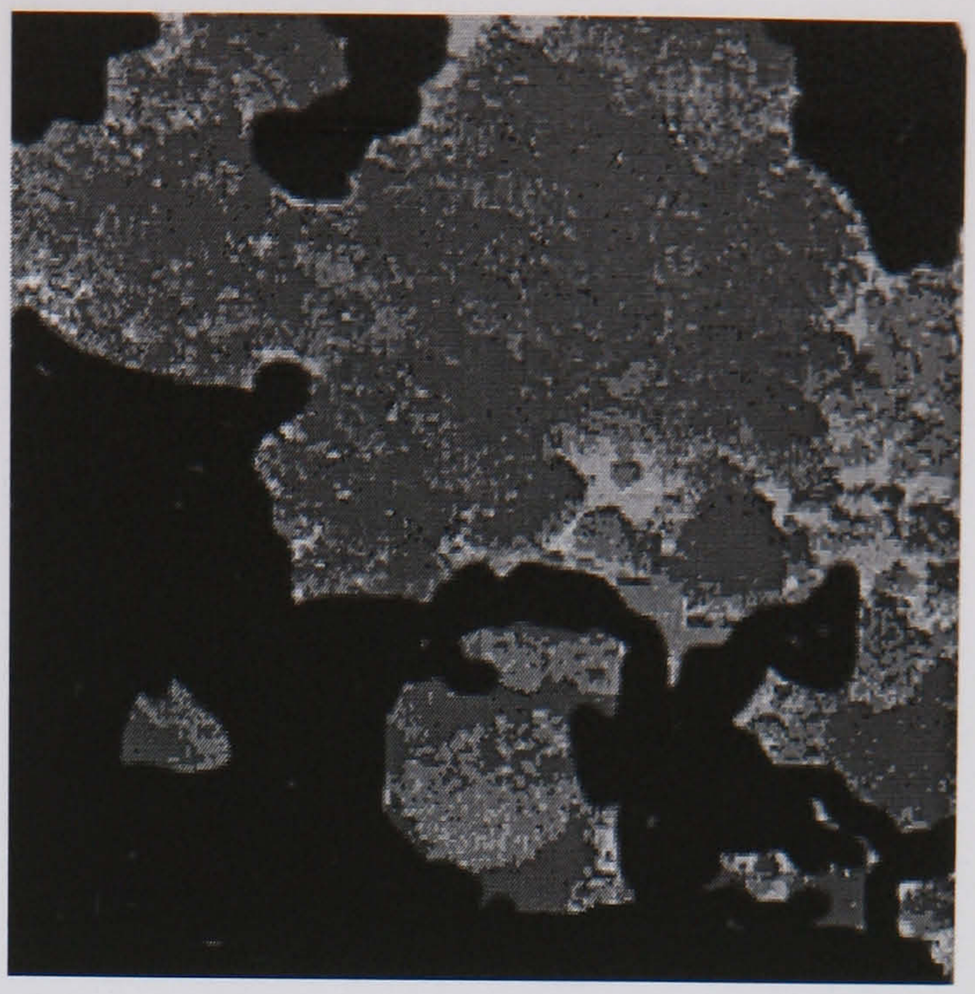


Hybrid Neural Network (76.24 %)

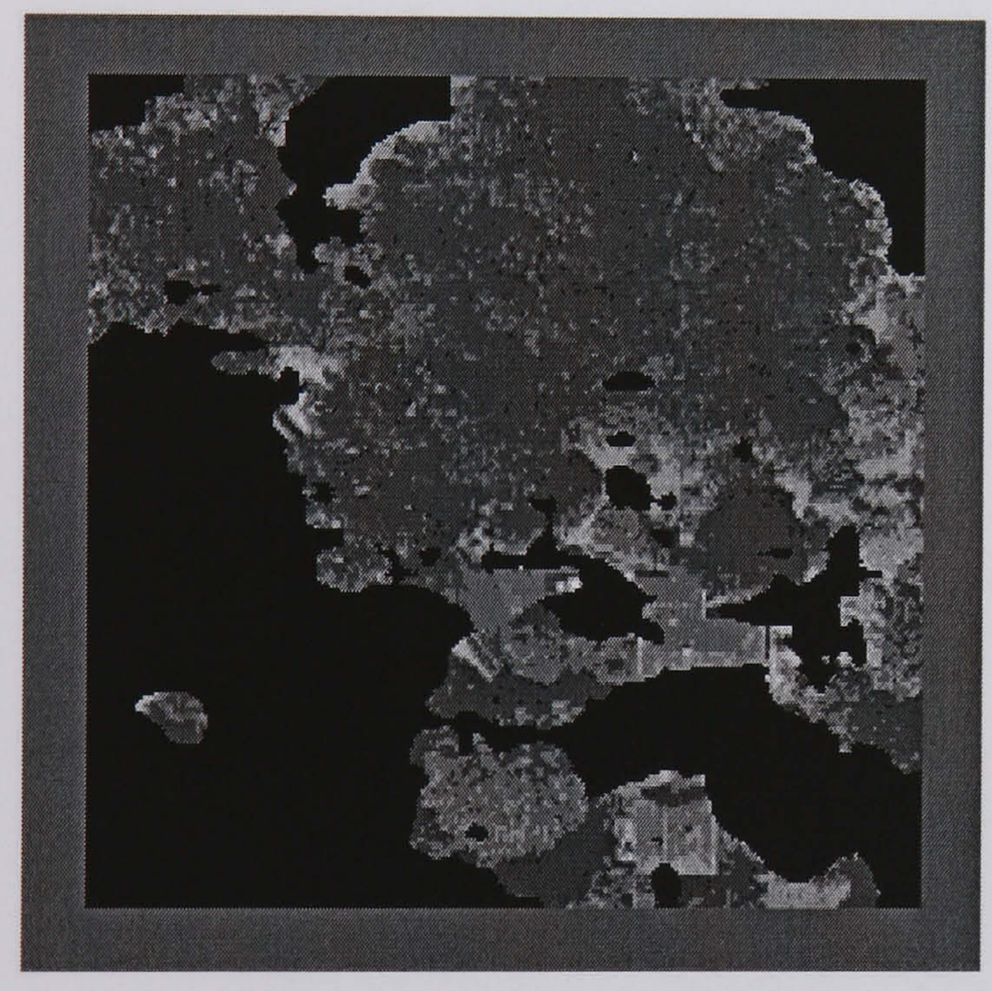
SGLDM /BPNN (75.55 %)

Figure E.30 Aerial Image 32 Results

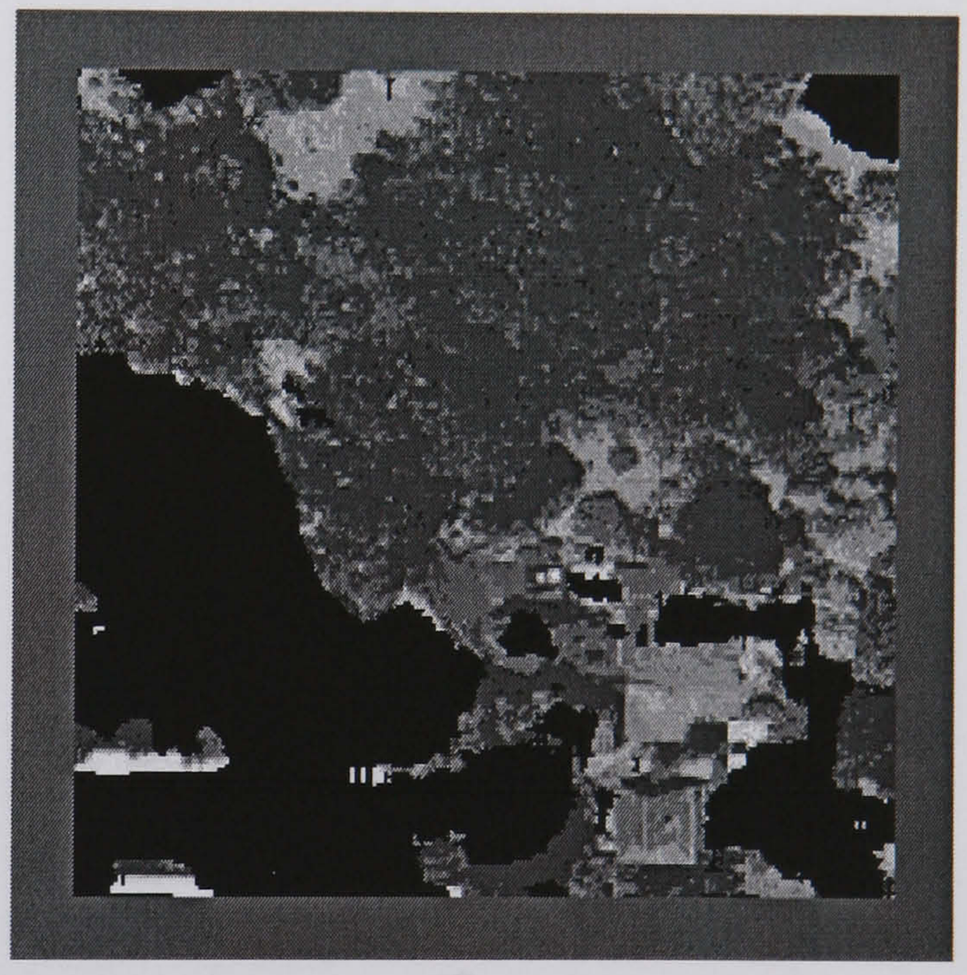
Aerial Image 33



Hand Segmented Image

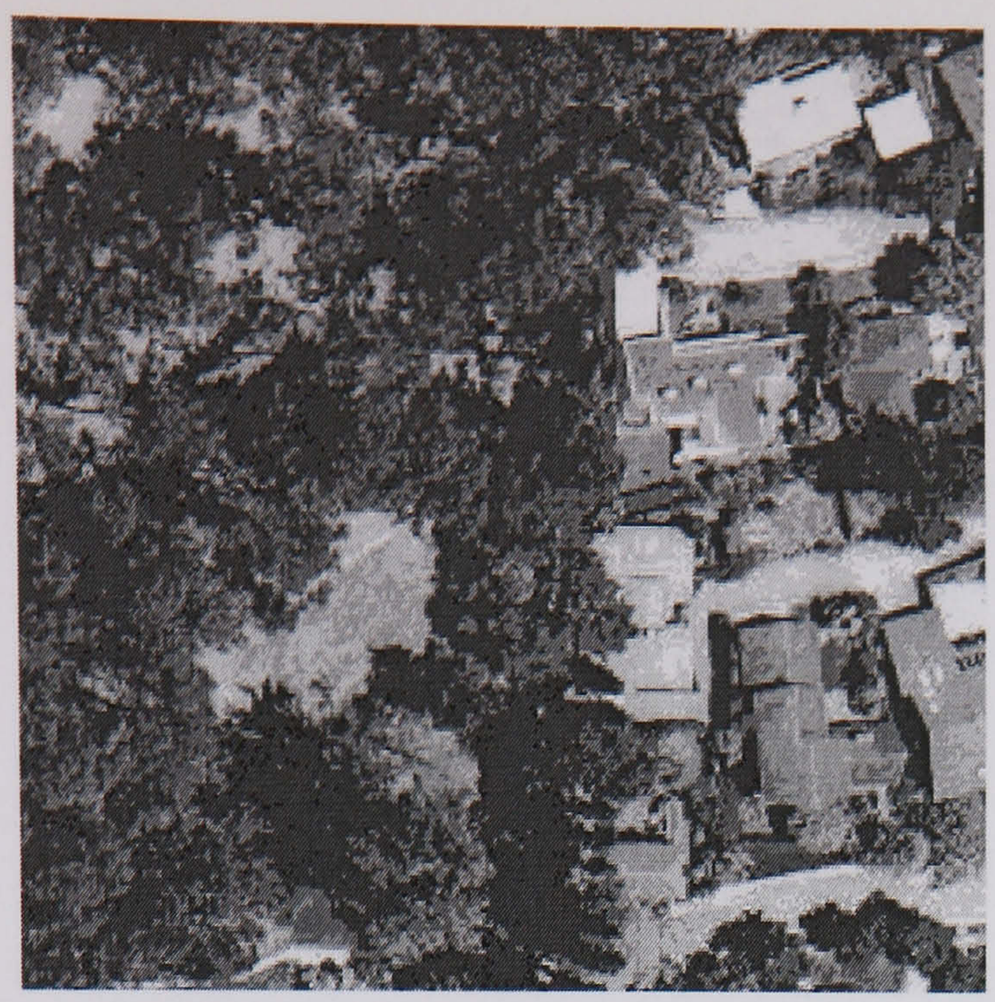


Hybrid Neural Network (86.89 %)

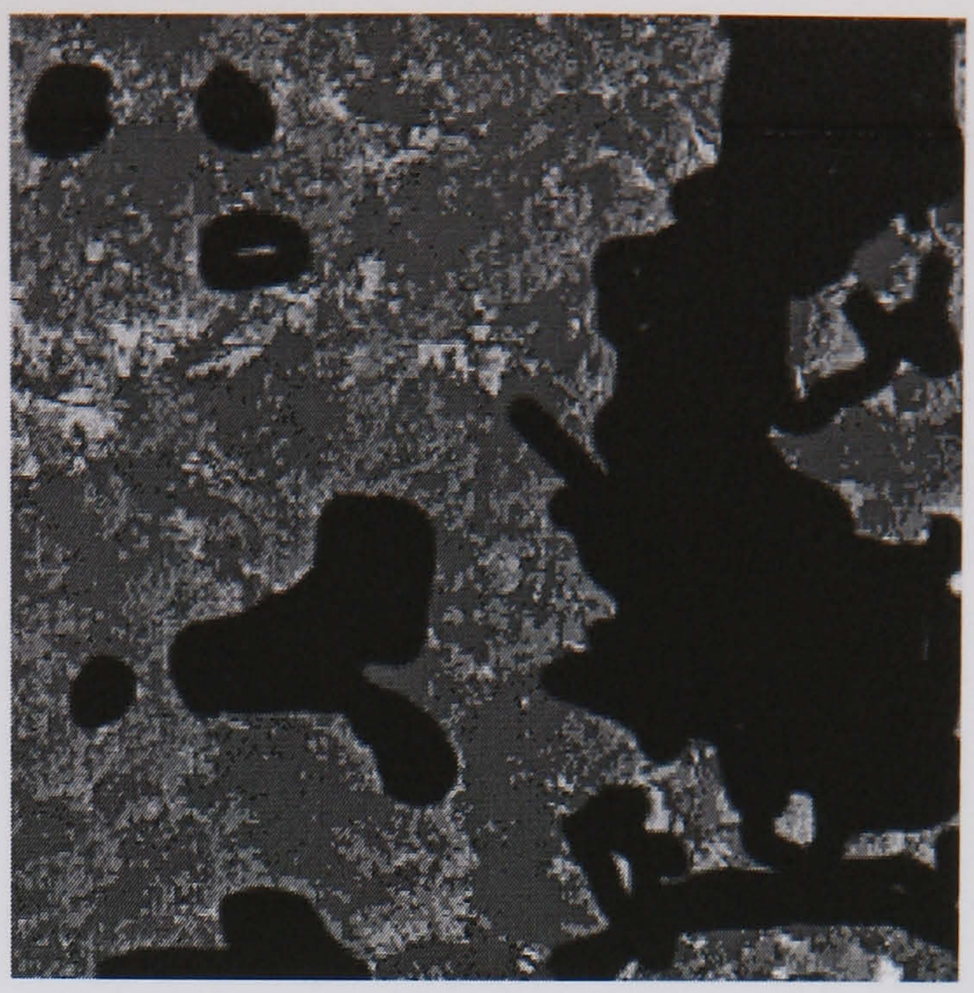


SGLDM /BPNN (81.01 %)

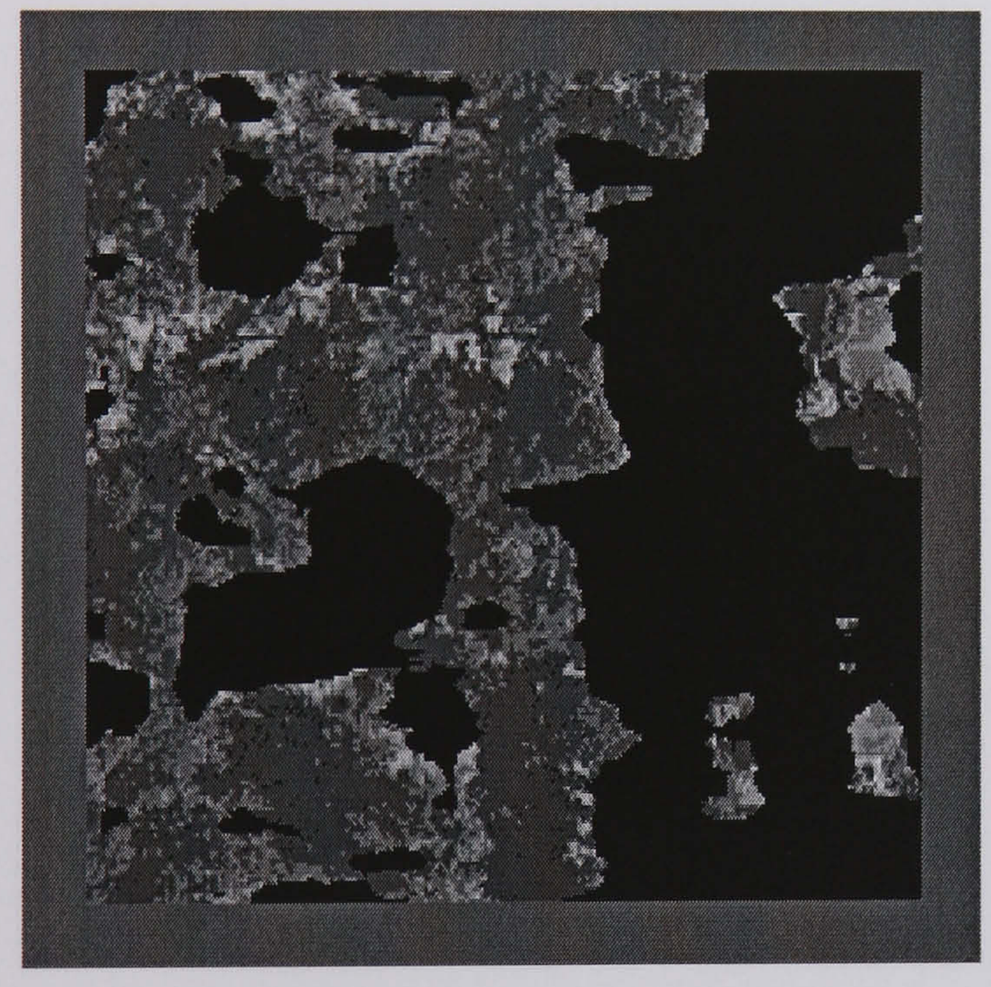
Figure E.31 Aerial Image 33 Results



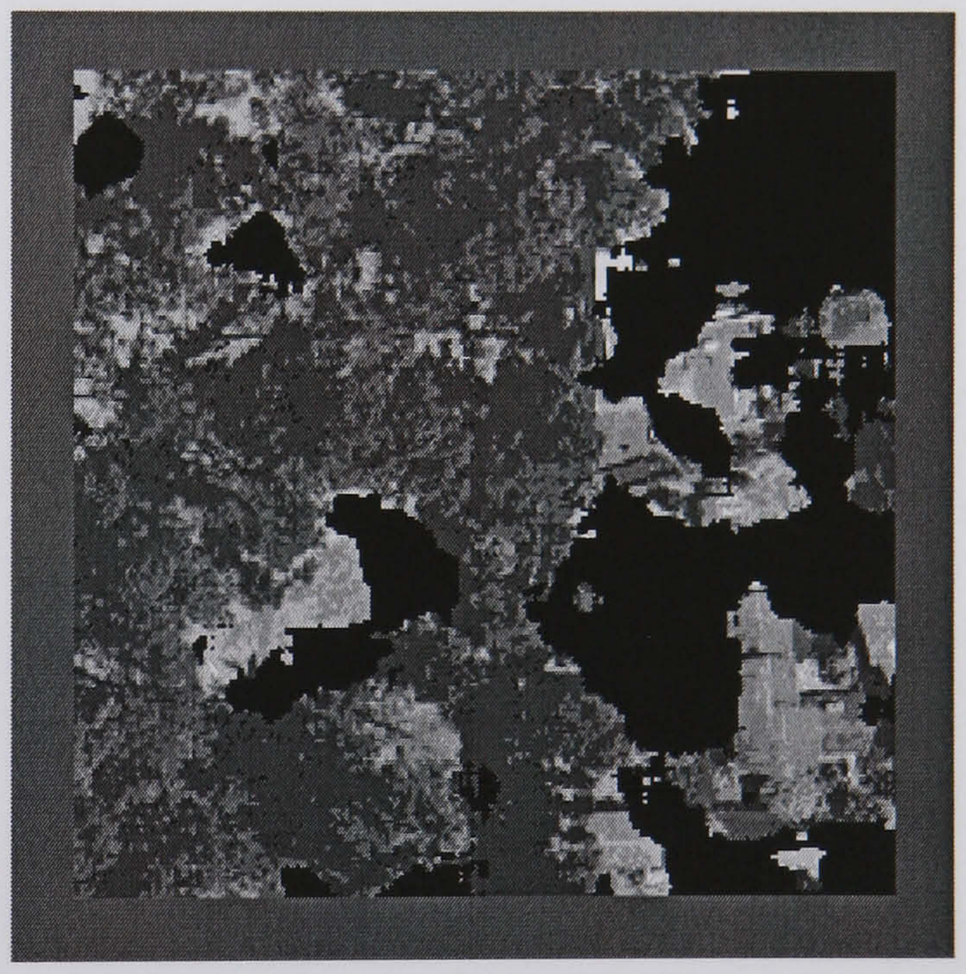
Aerial Image 34



Hand Segmented Image

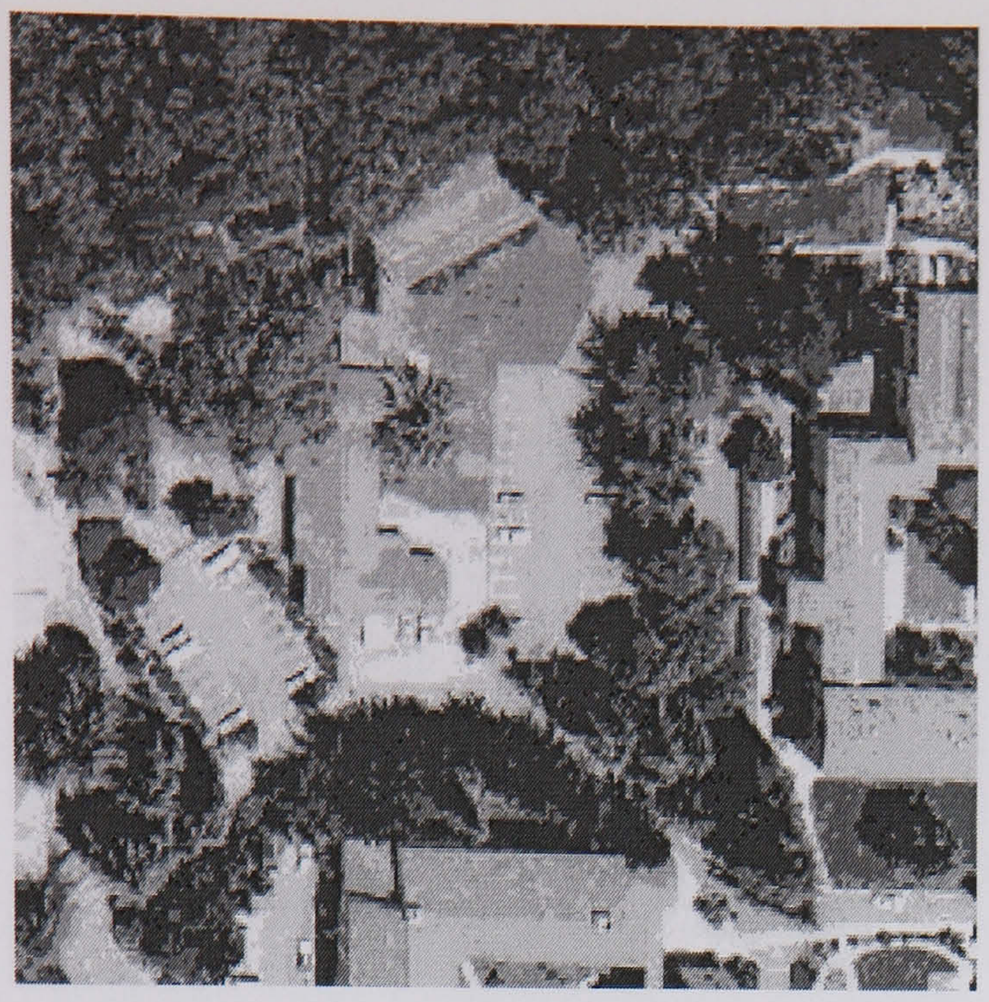


Hybrid Neural Network (84.95 %)

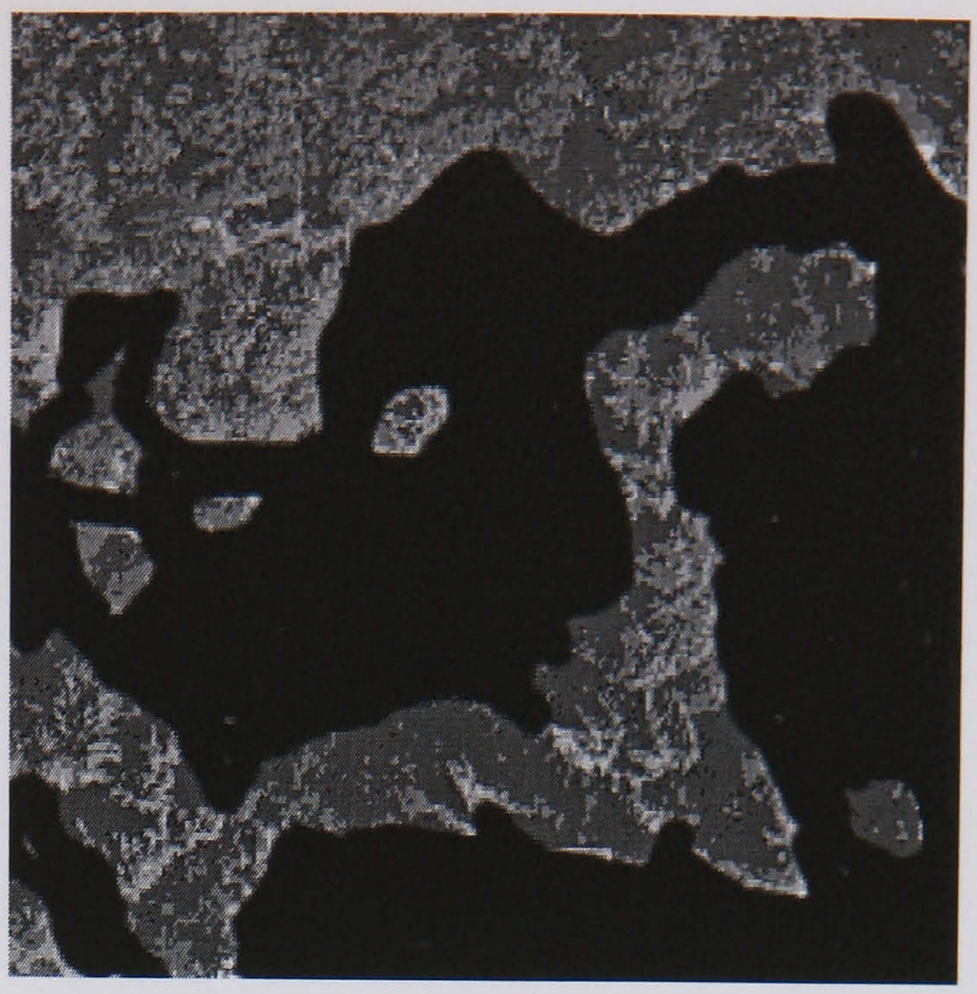


SGLDM /BPNN (78.99 %)

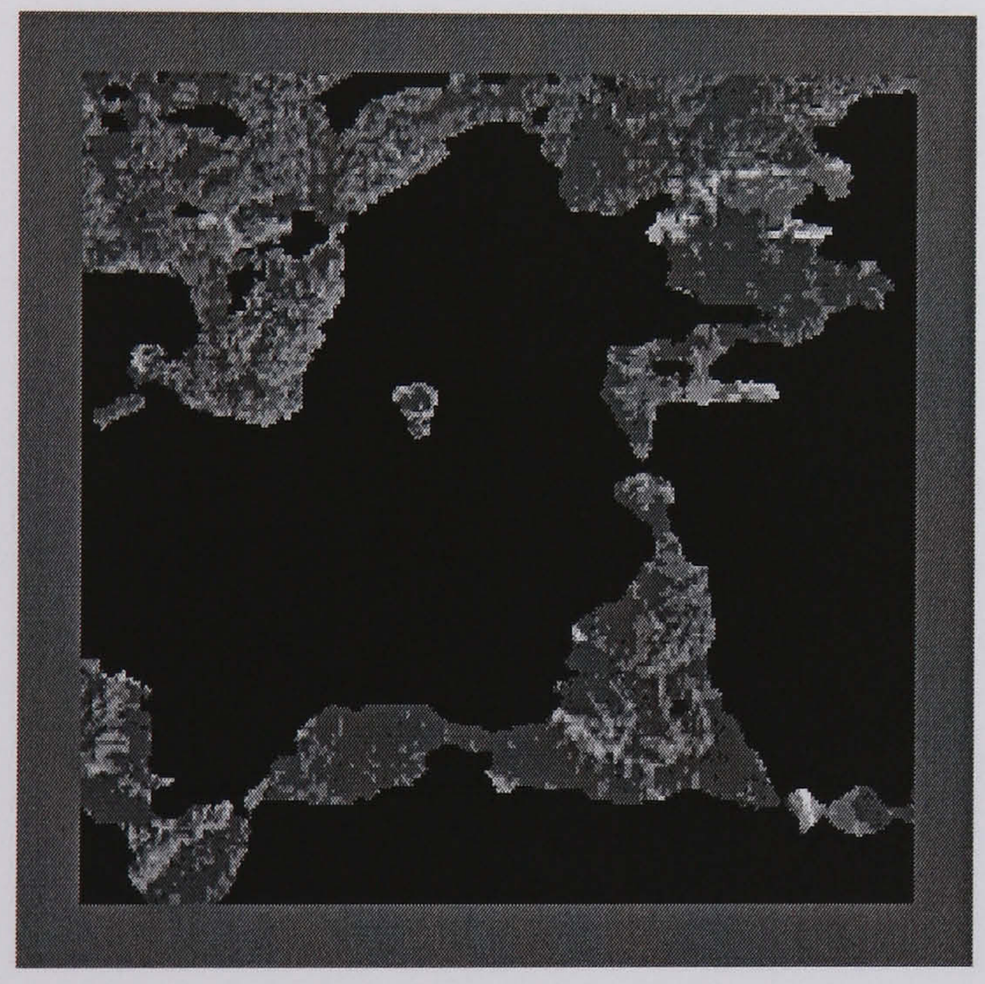
Figure E.32 Aerial Image 34 Results



Aerial Image 35



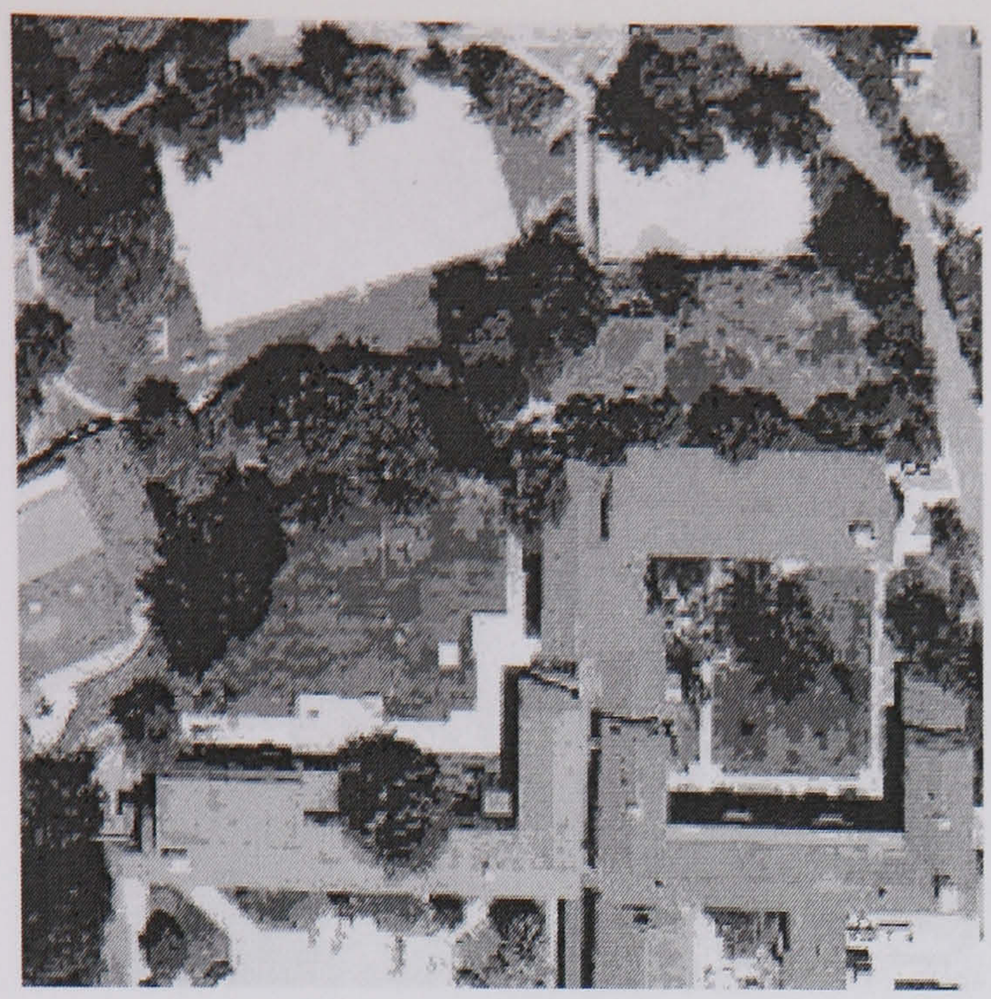
Hand Segmented Image



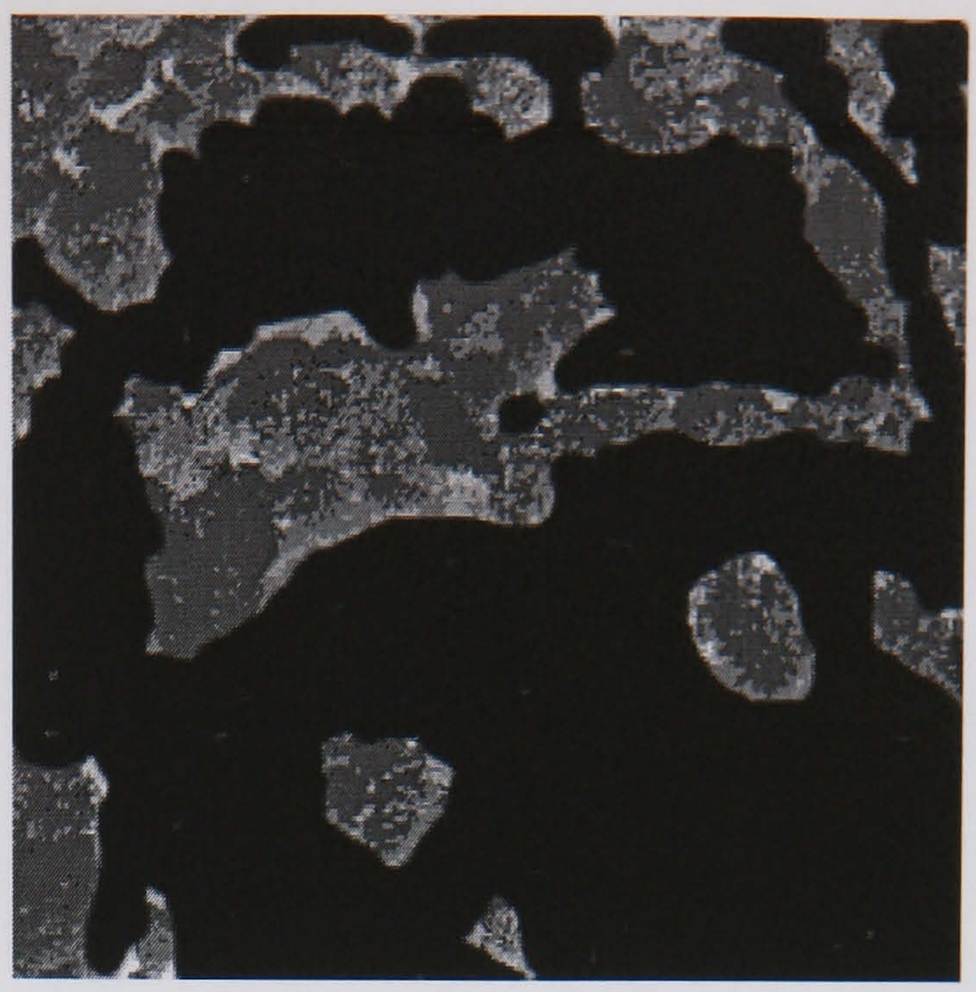
Hybrid Neural Network (83.05 %)

SGLDM /BPNN (70.16 %)

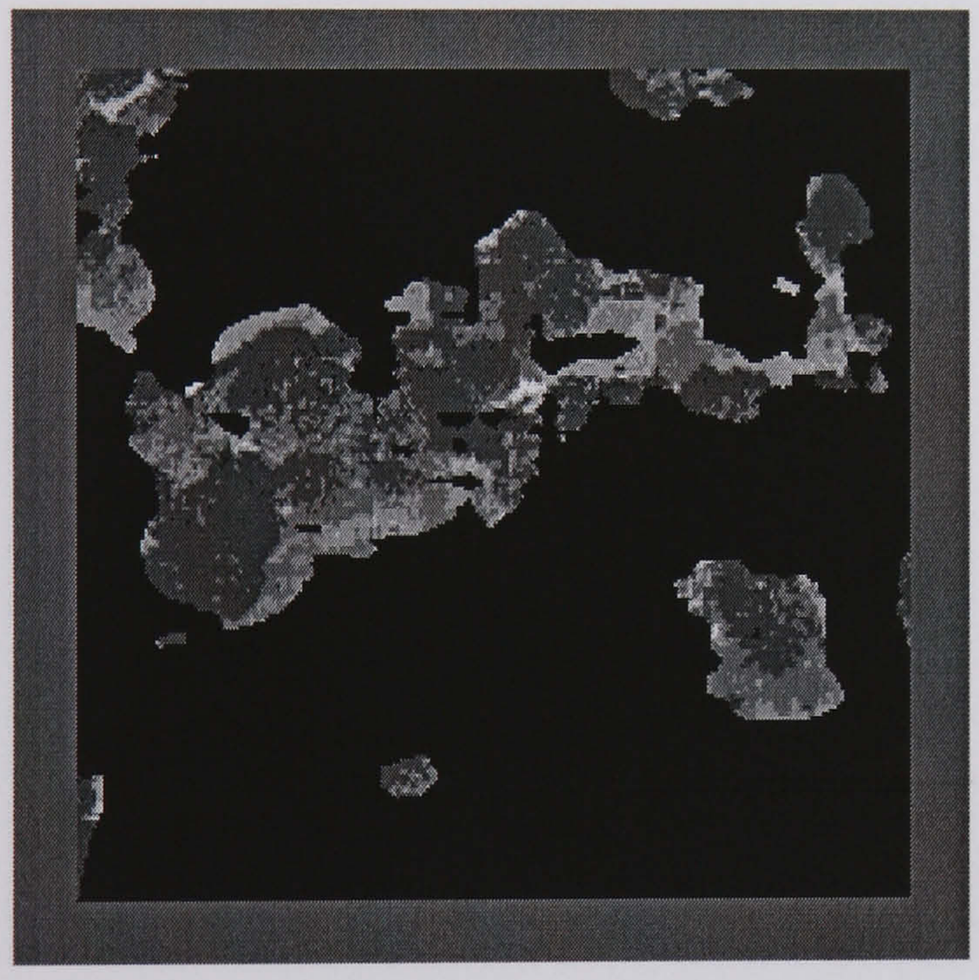
Figure E.33 Aerial Image 35 Results



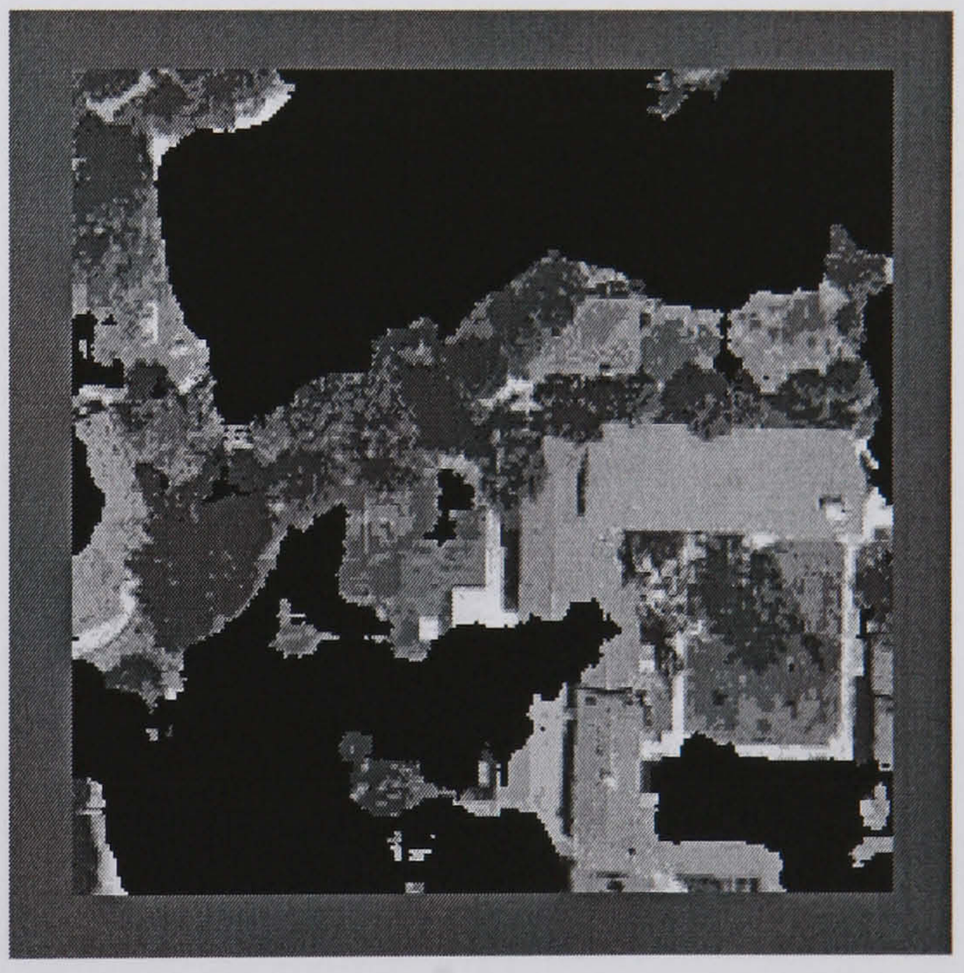
Aerial Image 36



Hand Segmented Image



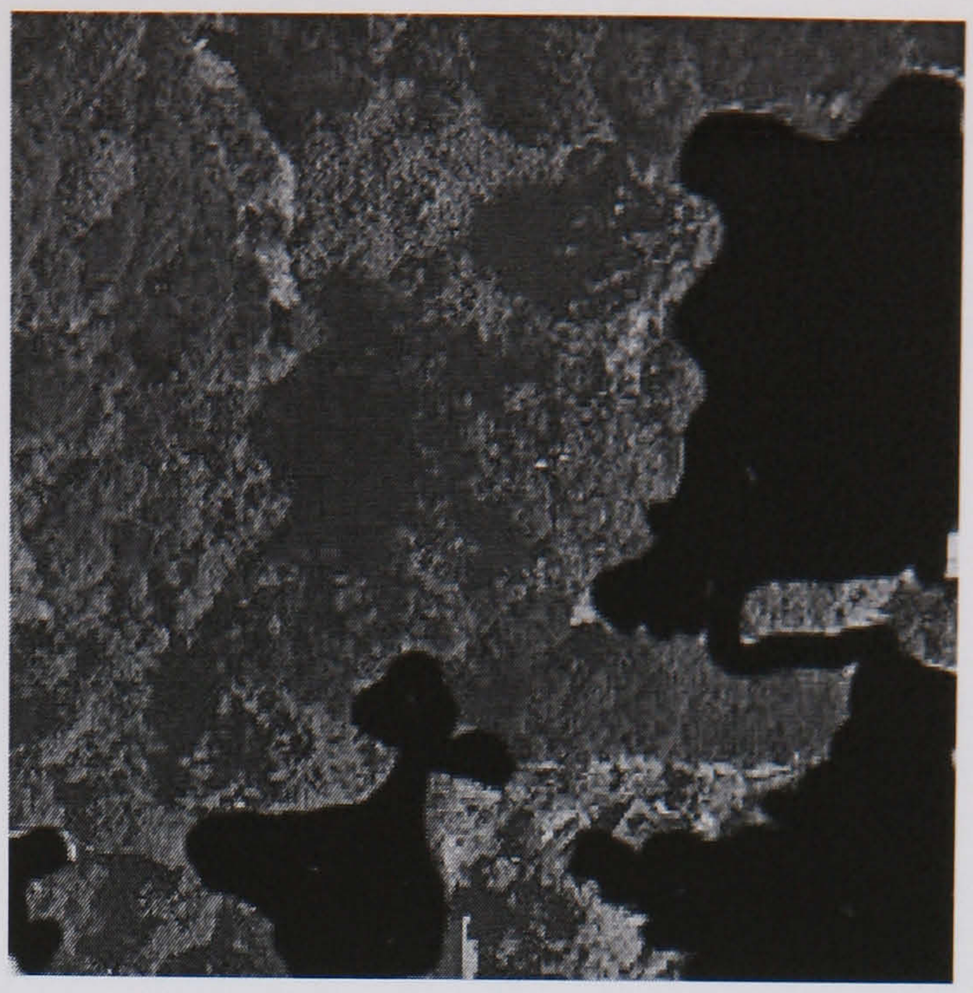
Hybrid Neural Network (83.36 %)



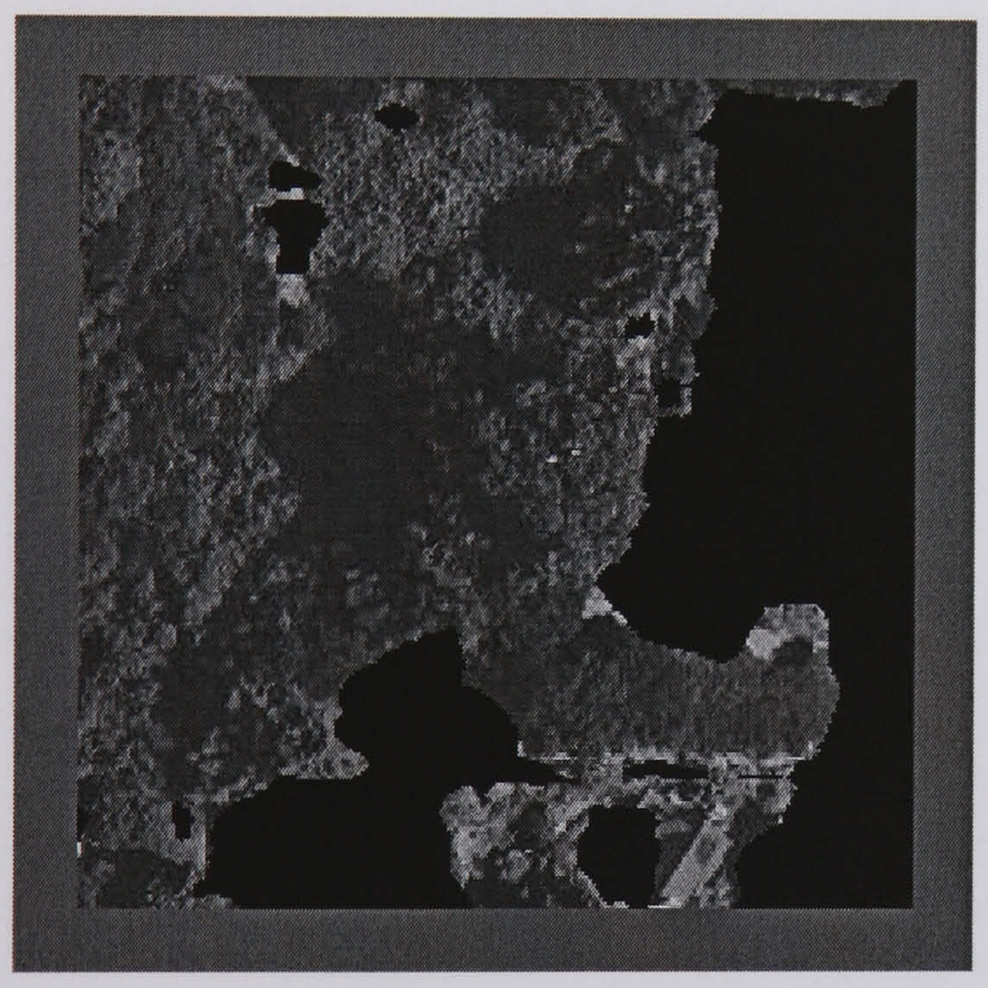
SGLDM /BPNN (66.67 %)

Figure E.34 Aerial Image 36 Results

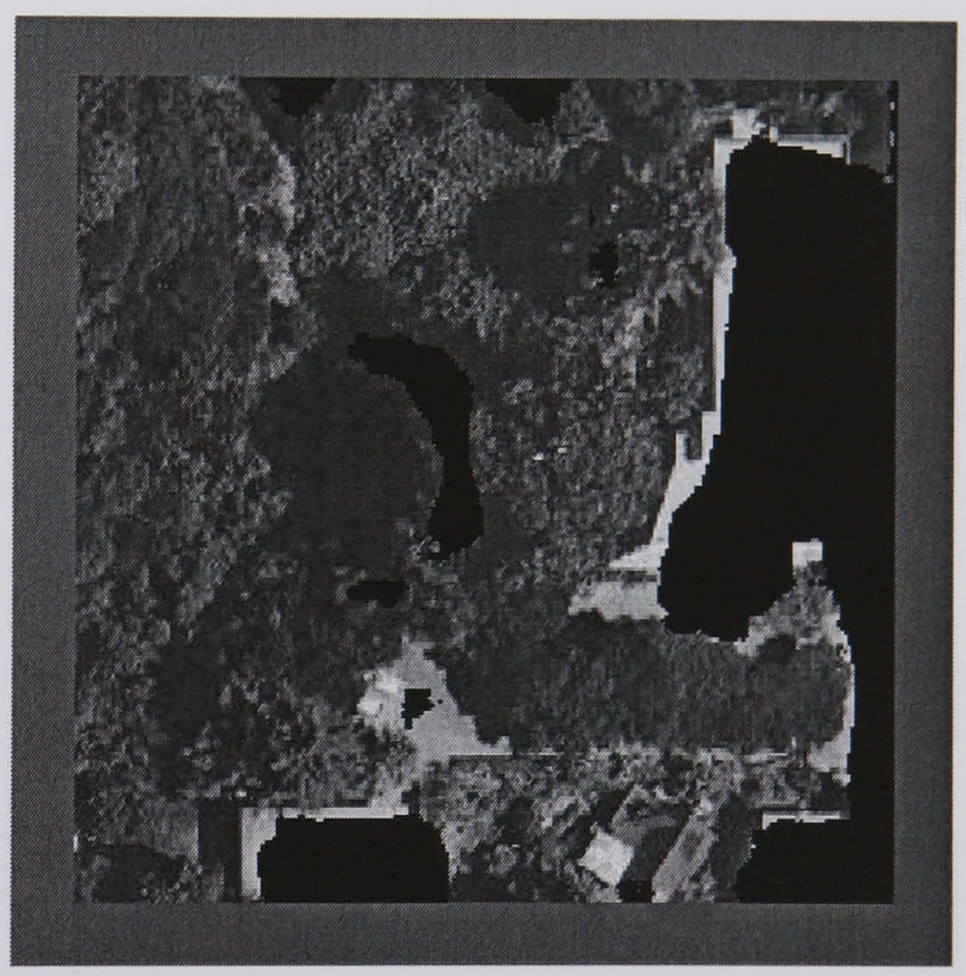
Aerial Image 37



Hand Segmented Image



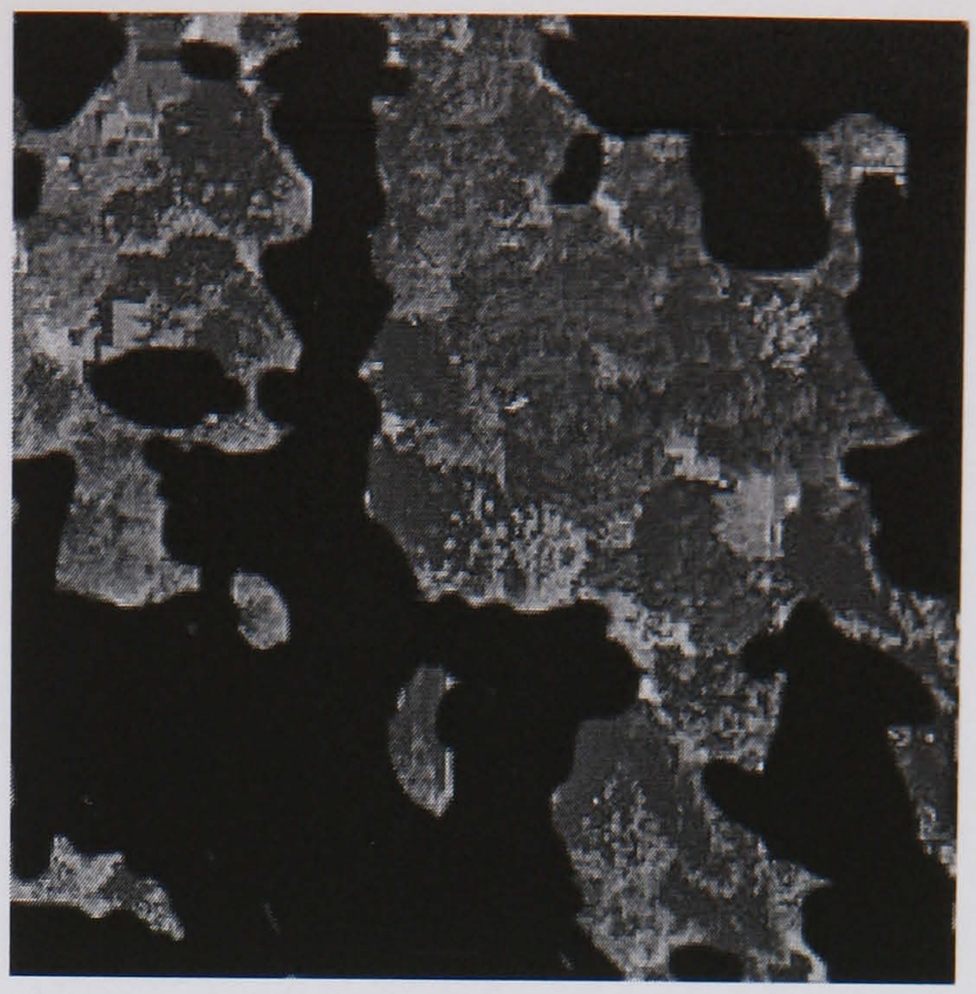
Hybrid Neural Network (91.21 %)



SGLDM /BPNN (87.30 %)

Figure E.35 Aerial Image 37 Results

Aerial Image 38



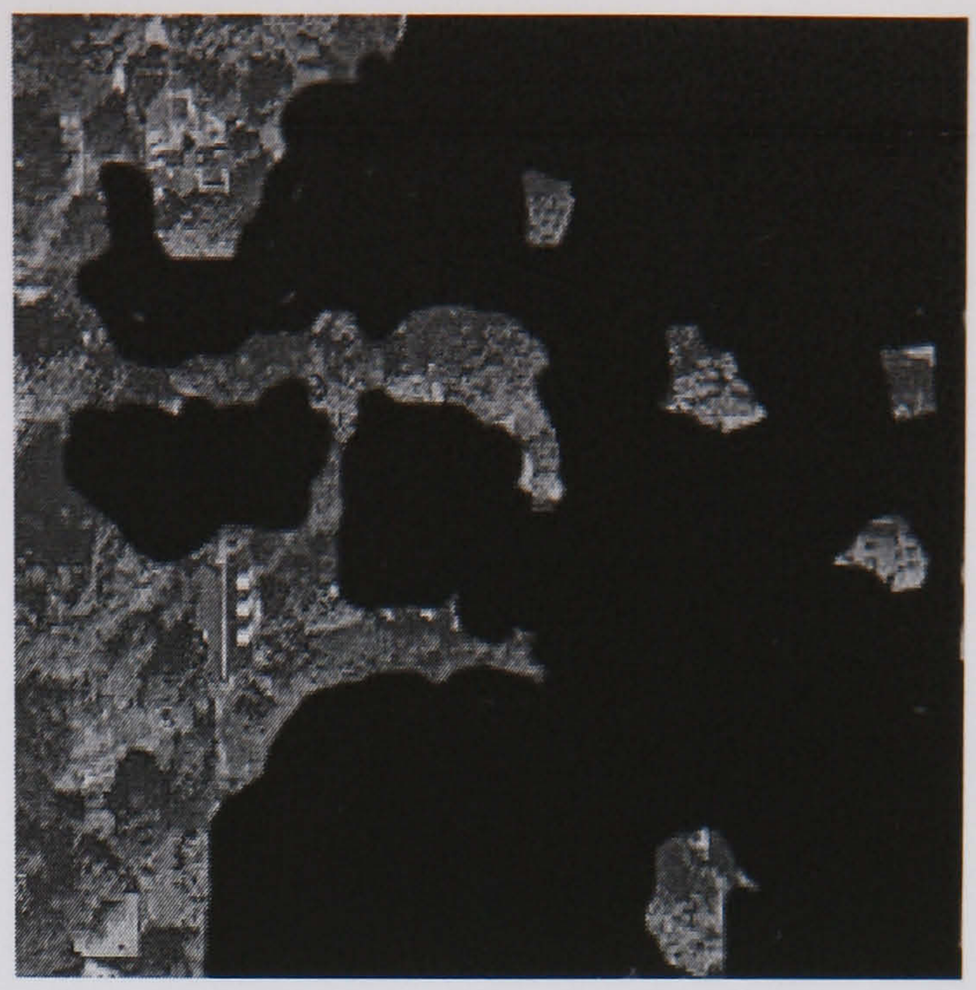
Hand Segmented Image

Hybrid Neural Network (84.03 %)

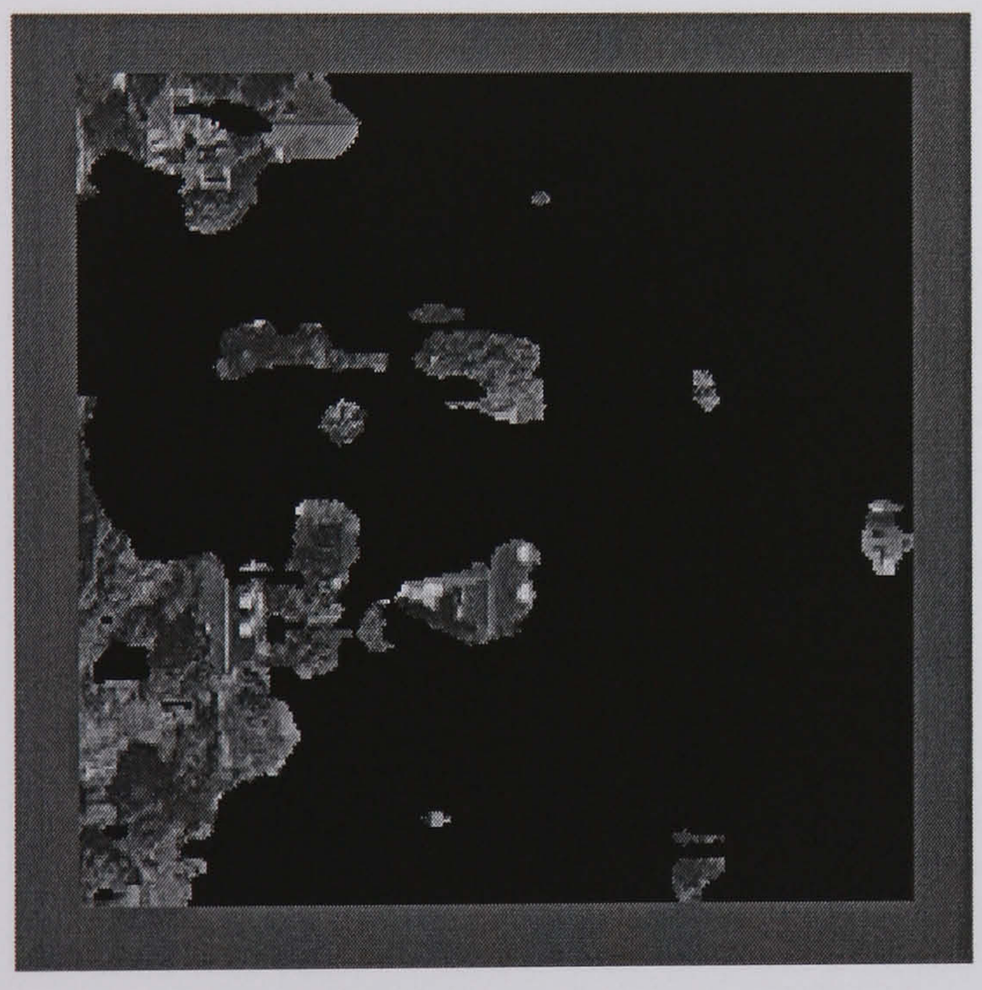
SGLDM /BPNN (80.85 %)

Figure E.36 Aerial Image 38 Results

Aerial Image 39



Hand Segmented Image

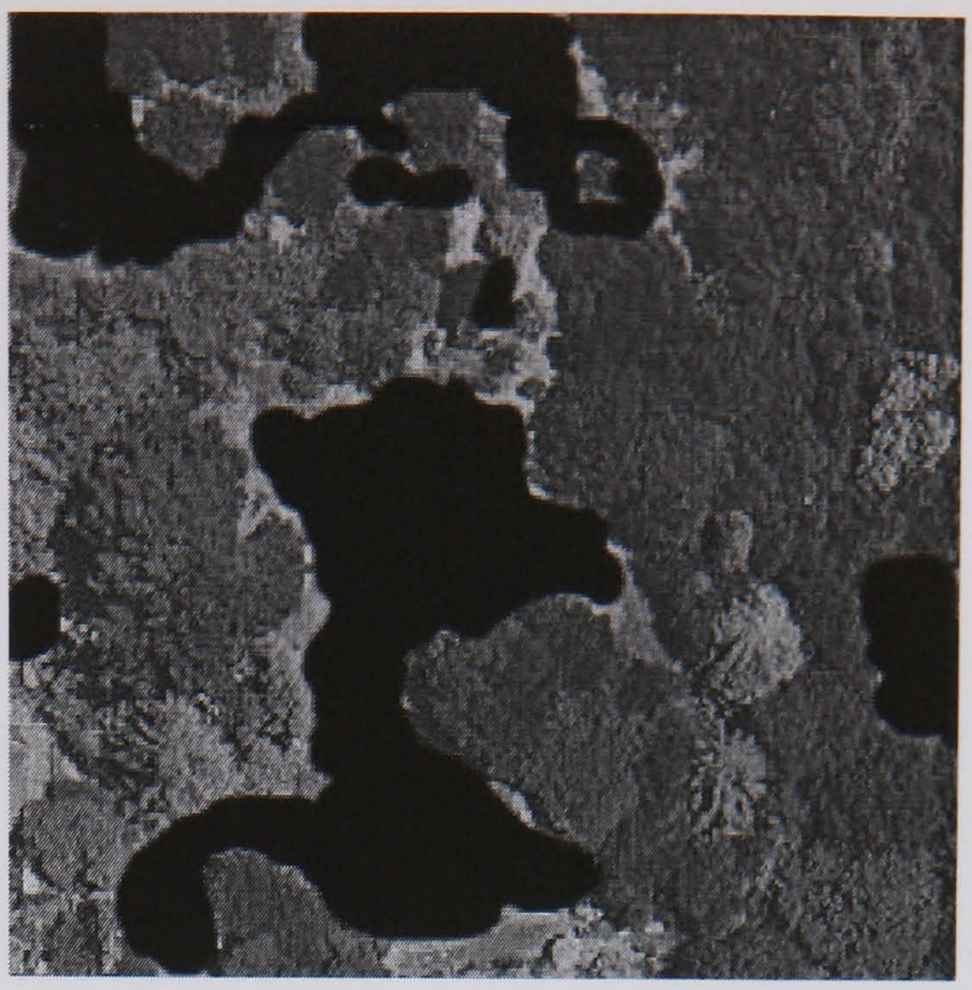


Hybrid Neural Network (88.20 %)

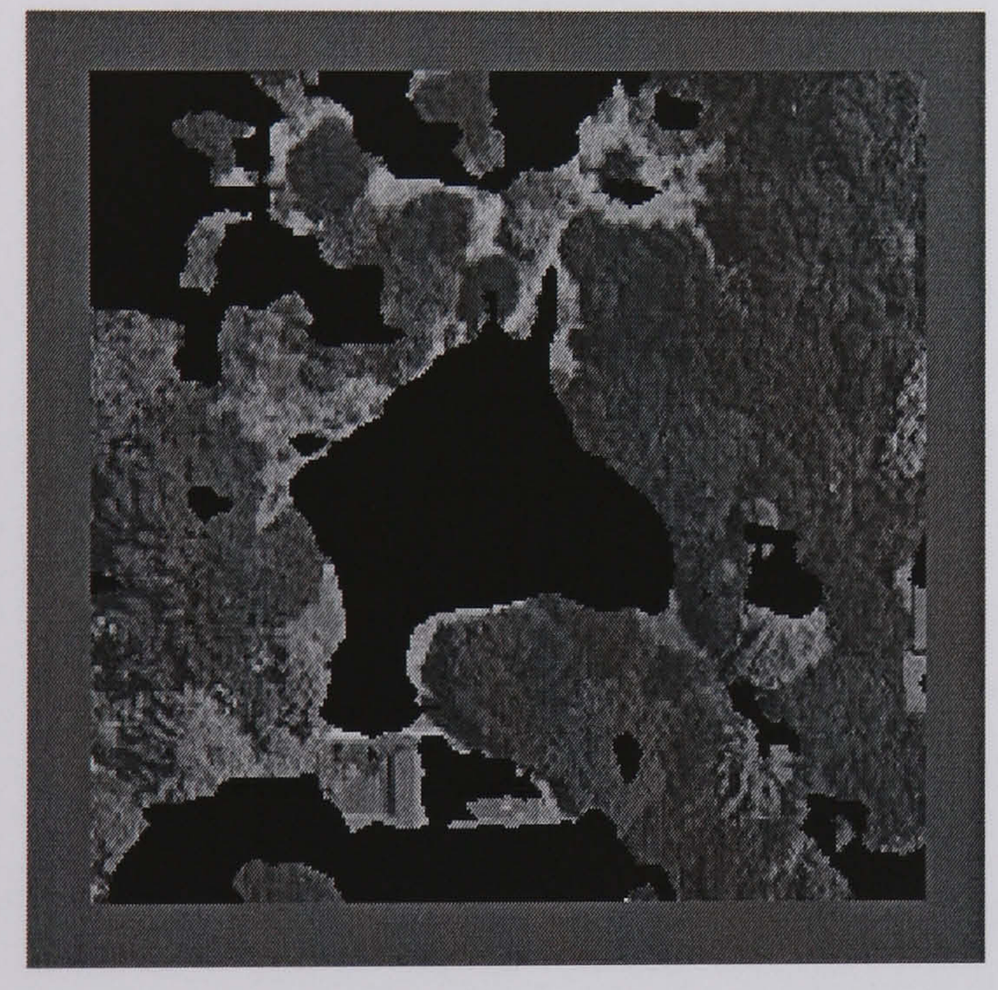
SGLDM /BPNN (80.31 %)

Figure E.37 Aerial Image 39 Results

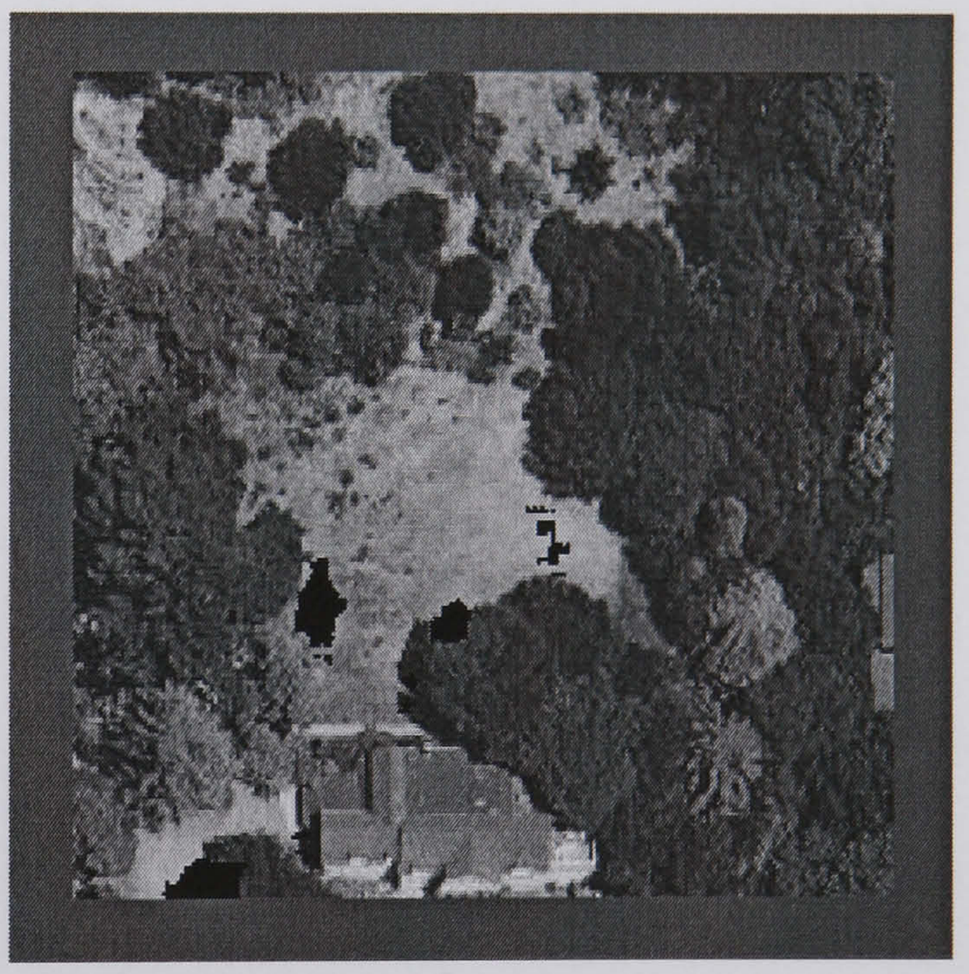
Aerial Image 40



Hand Segmented Image



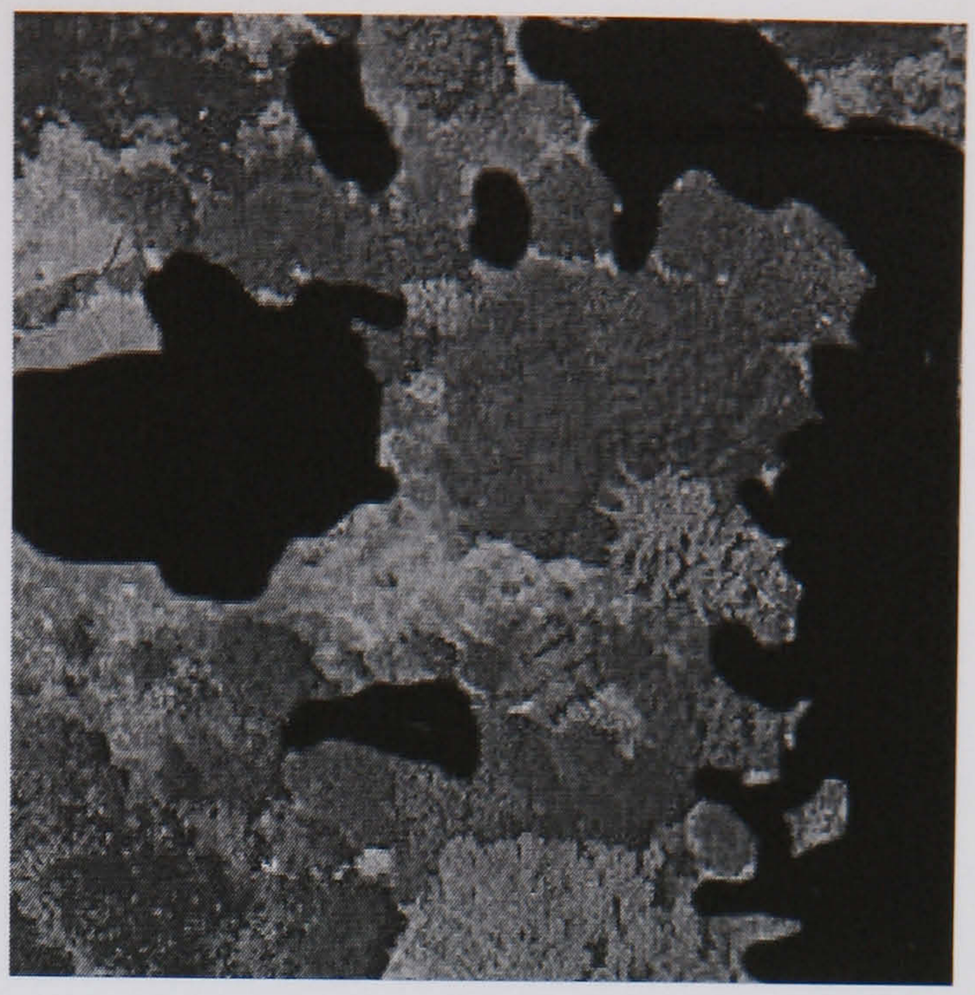
Hybrid Neural Network (83.42 %)



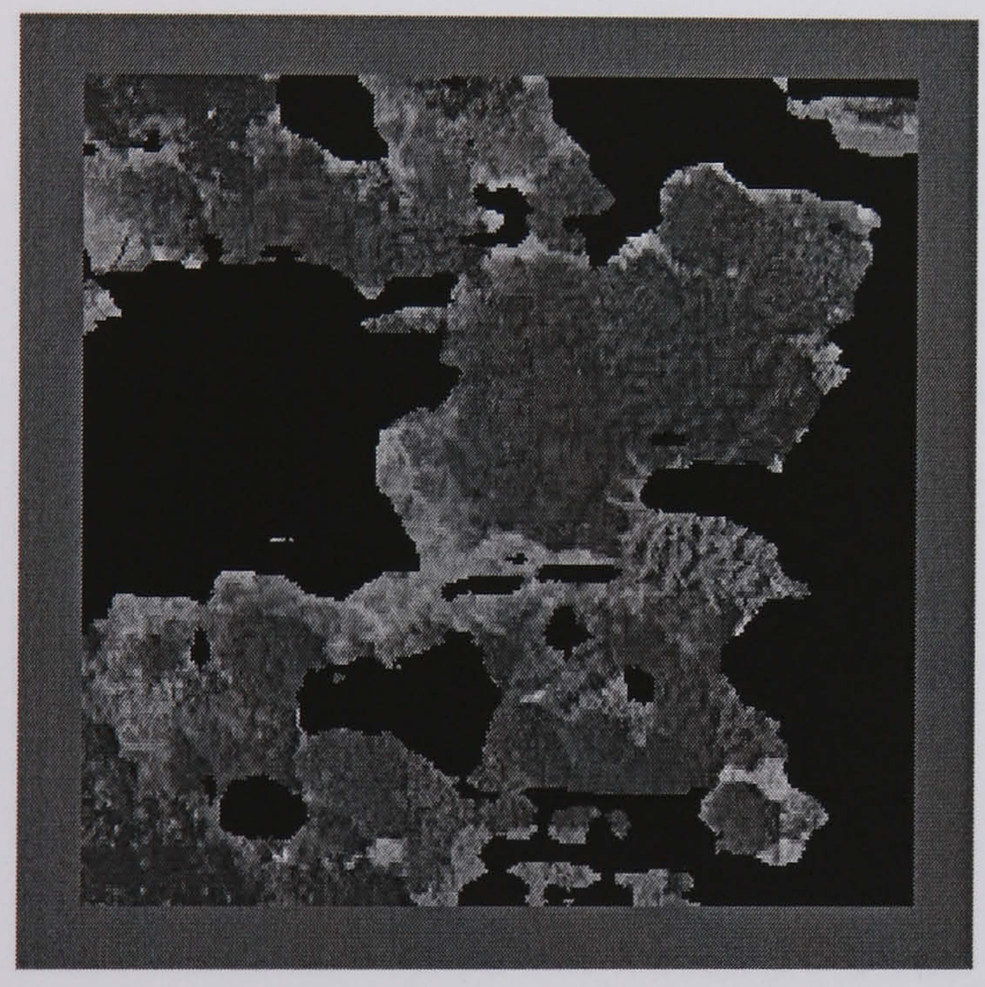
SGLDM /BPNN (77.12 %)

Figure E.38 Aerial Image 40 Results

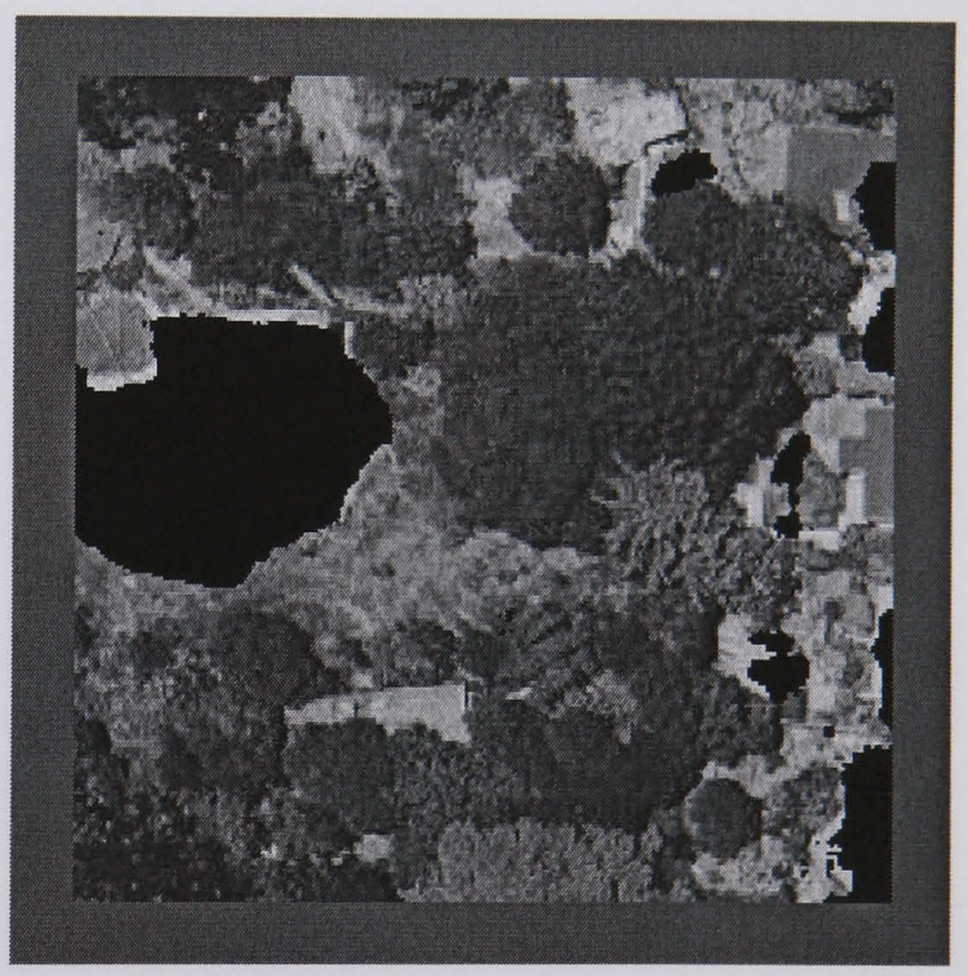
Aerial Image 41



Hand Segmented Image



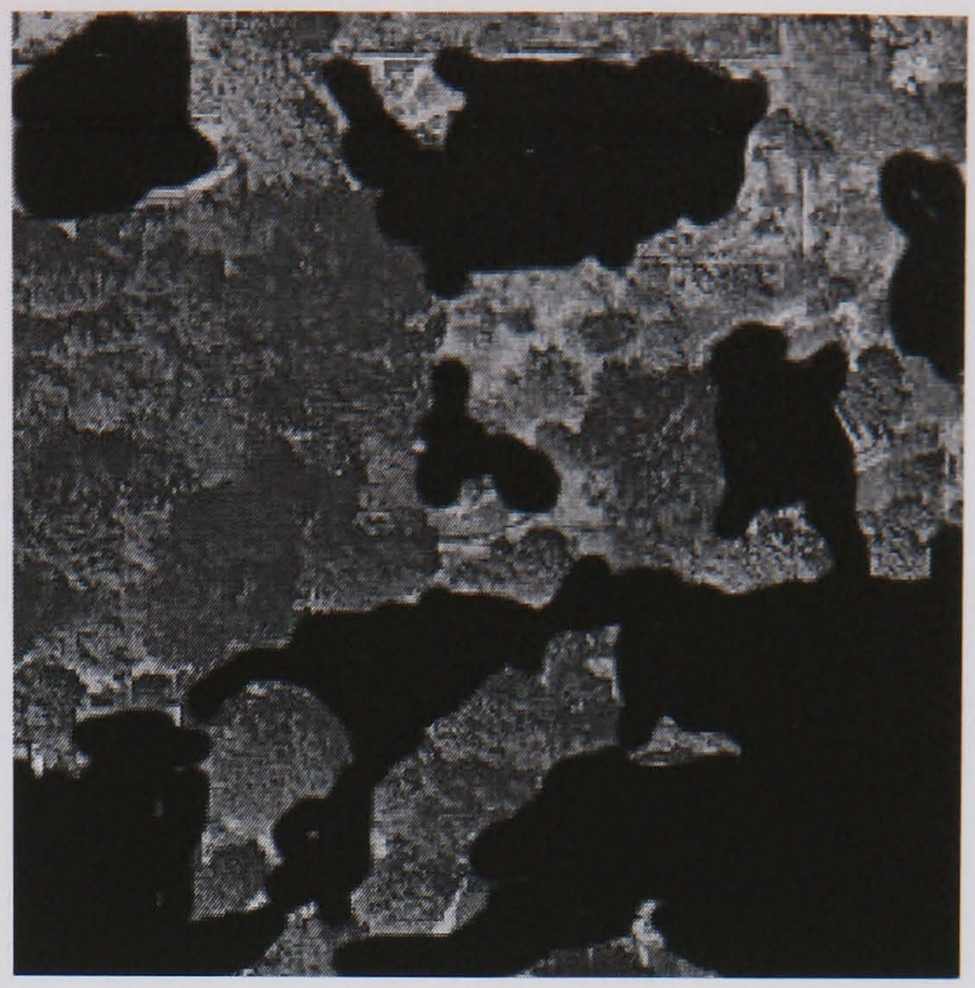
Hybrid Neural Network (84.59%)



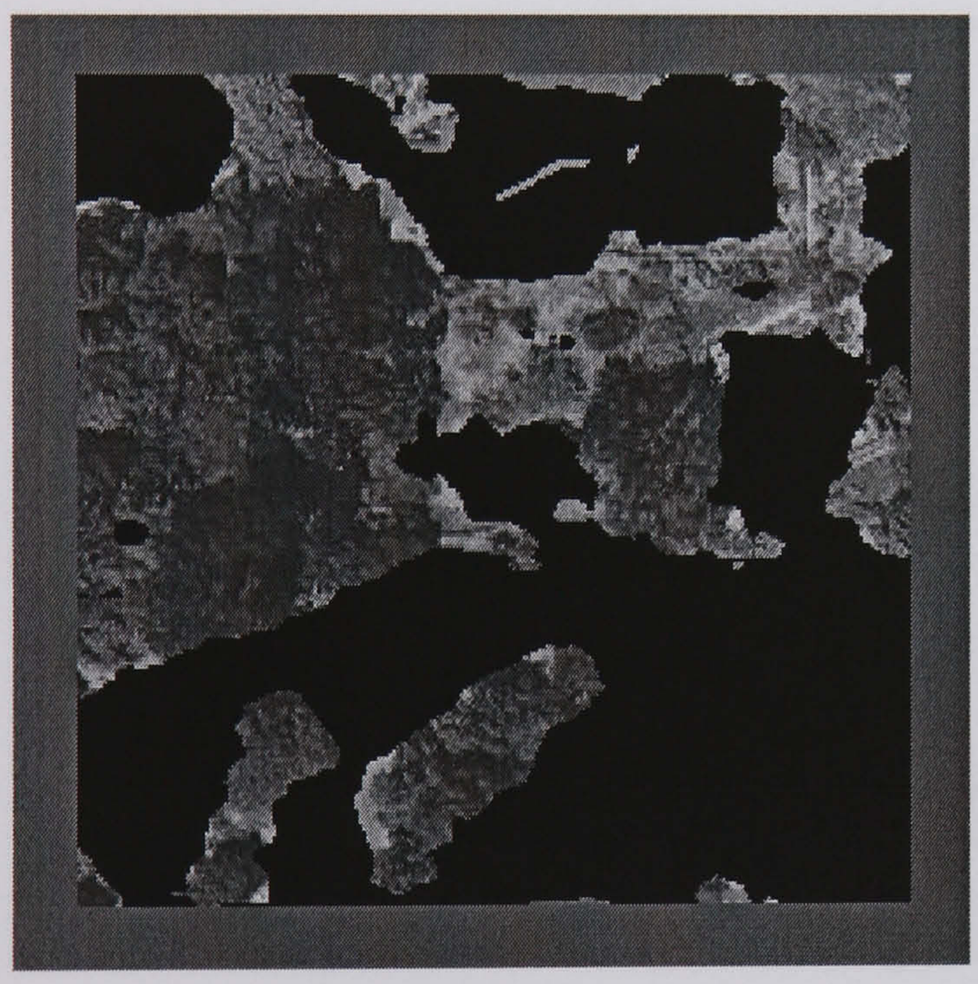
SGLDM /BPNN (81.46 %)

Figure E.39 Aerial Image 41 Results

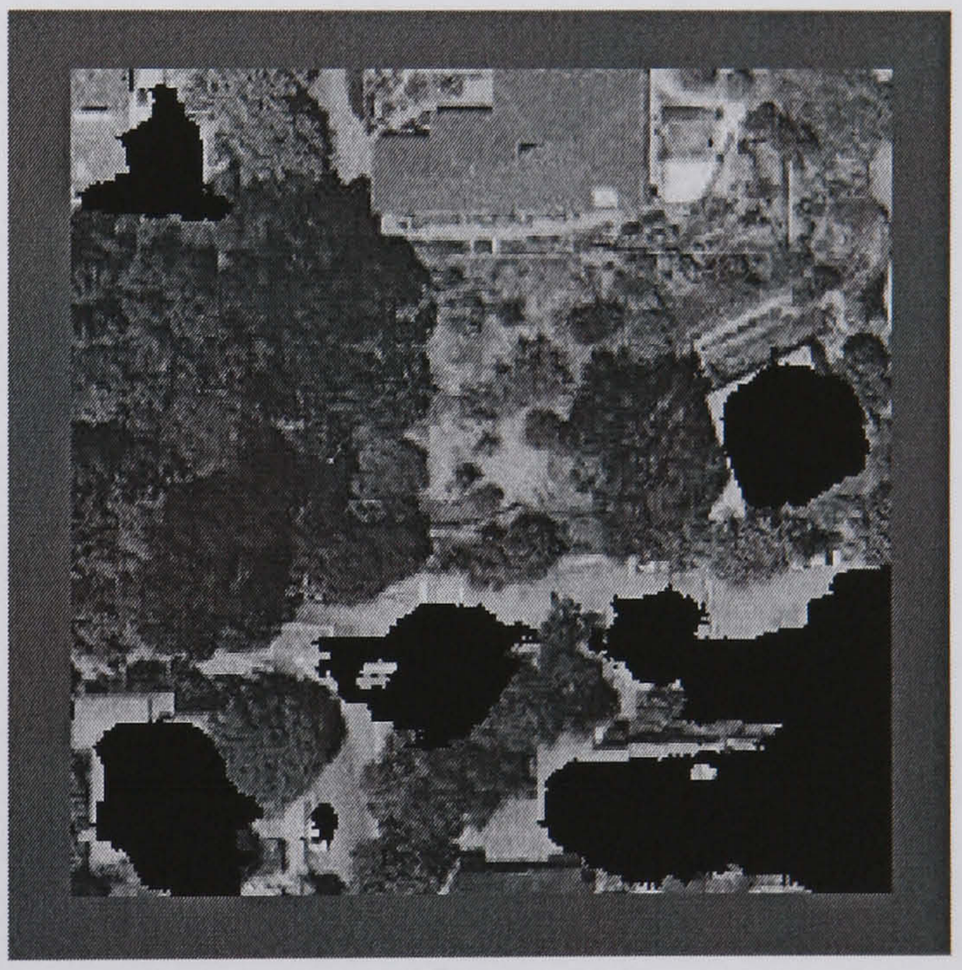
Aerial Image 42



Hand Segmented Image



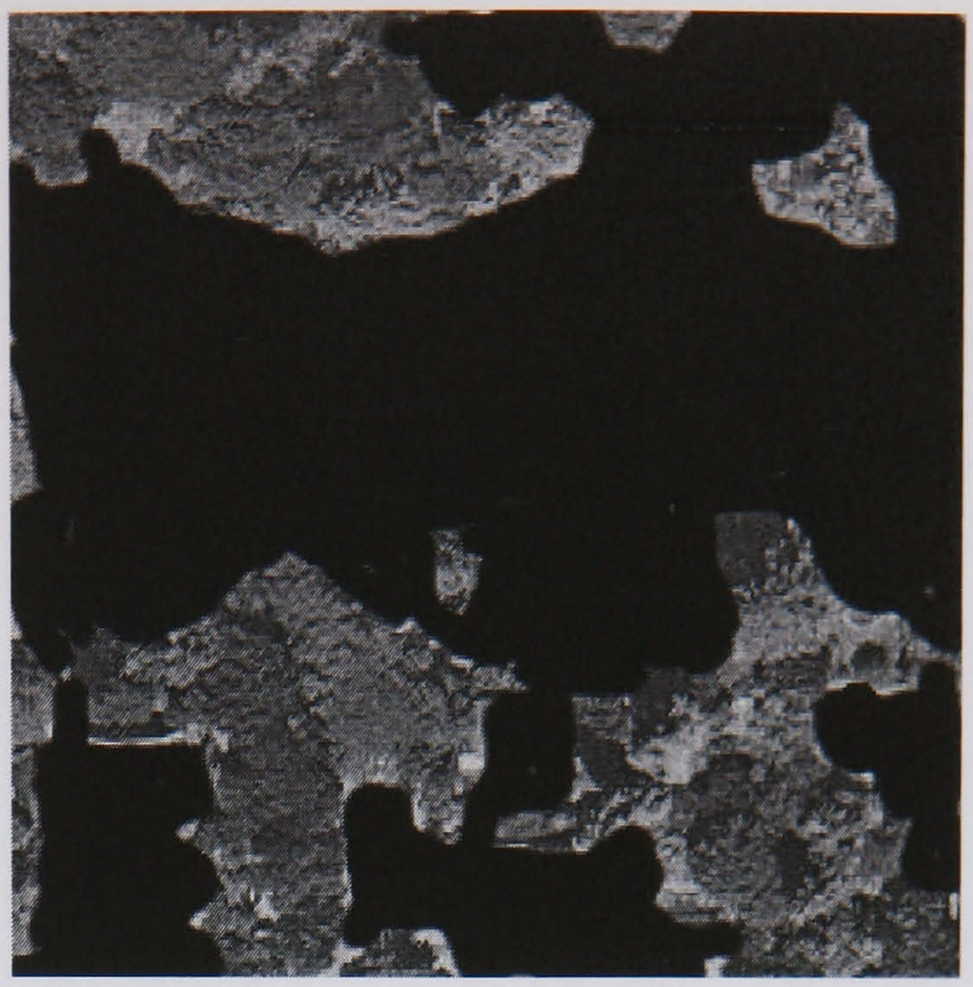
Hybrid Neural Network (85.01 %)



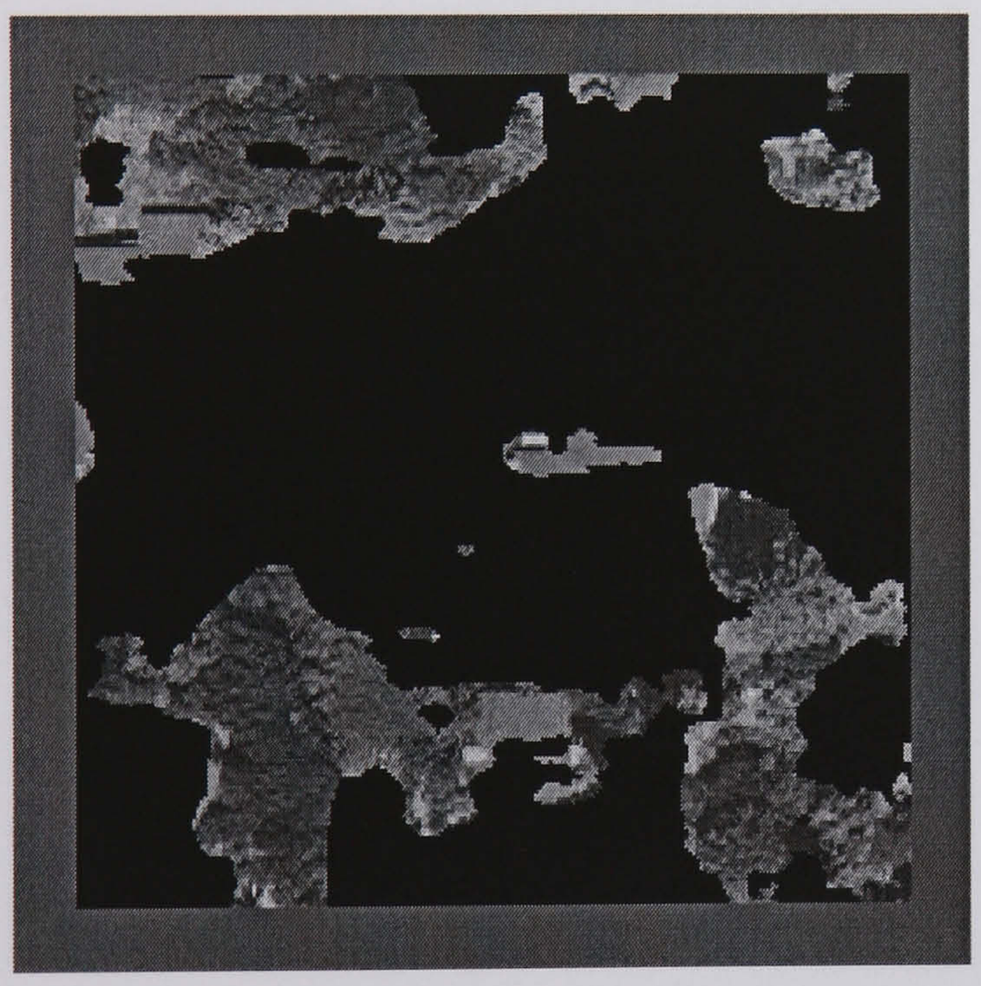
SGLDM /BPNN (74.62 %)

Figure E.40 Aerial Image 42 Results

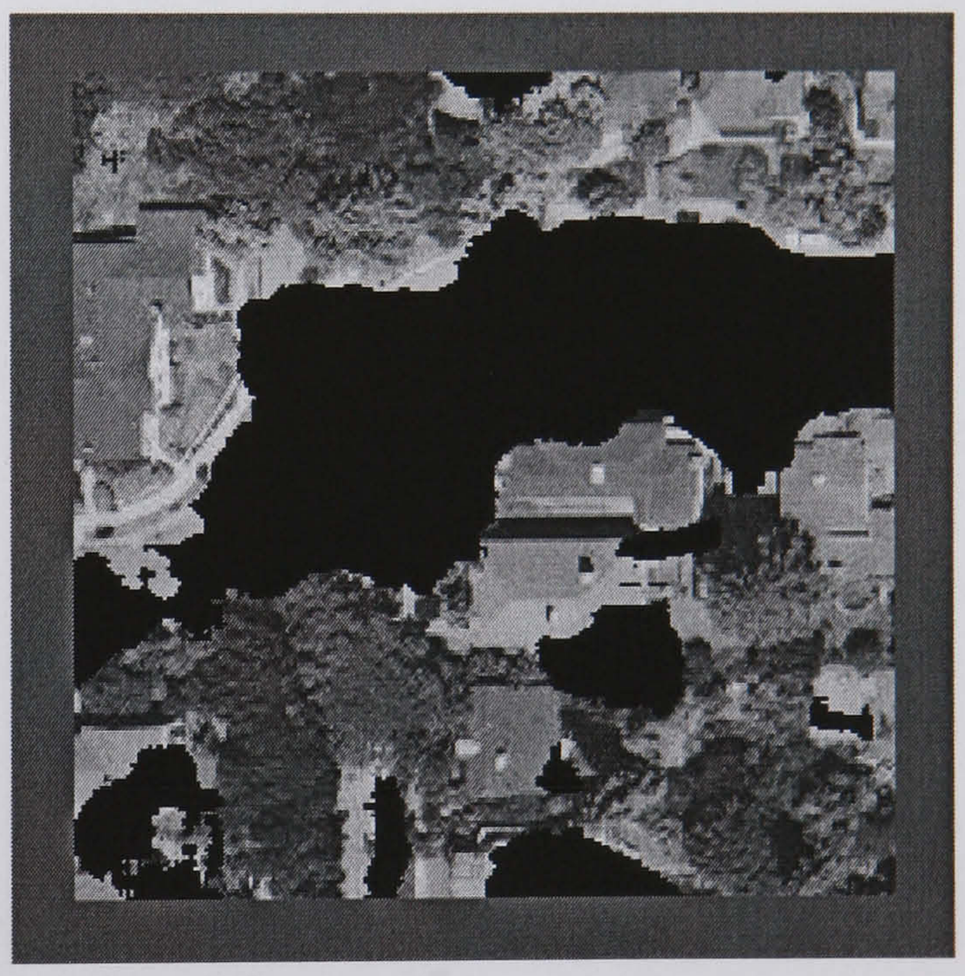
Aerial Image 43



Hand Segmented Image



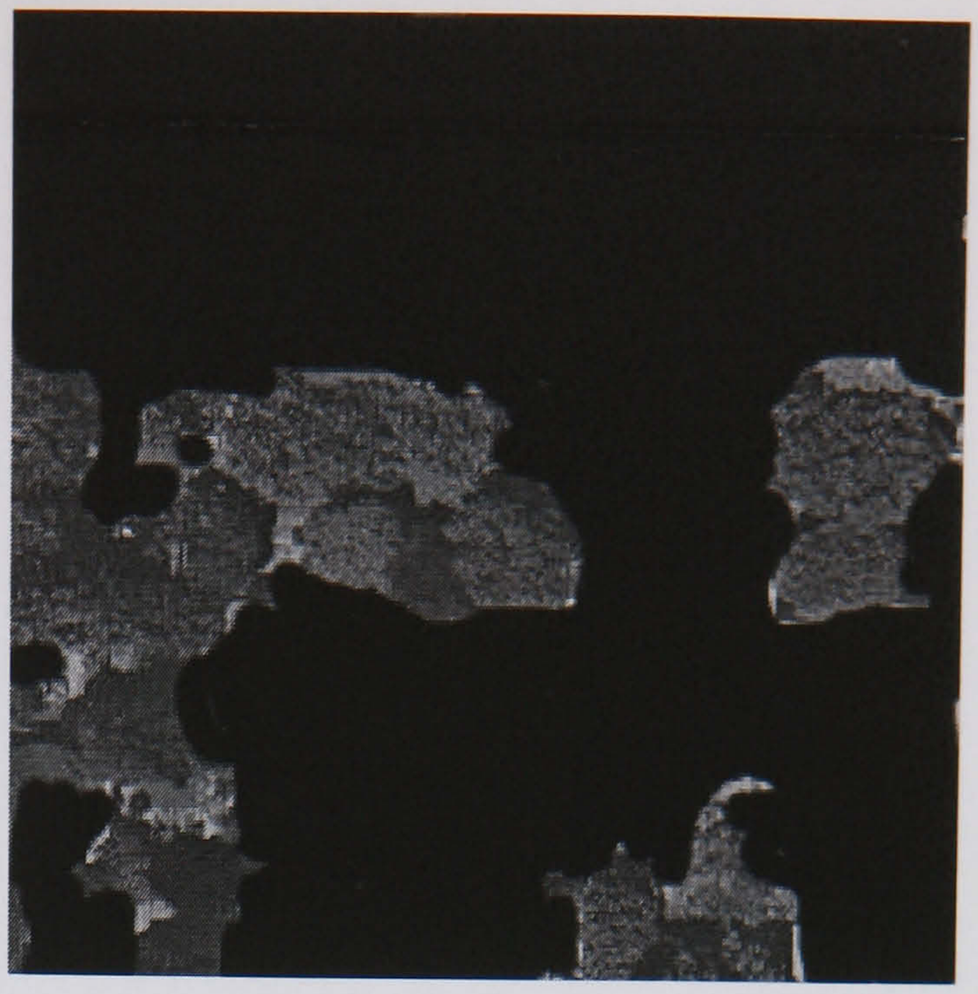
Hybrid Neural Network (85.55 %)



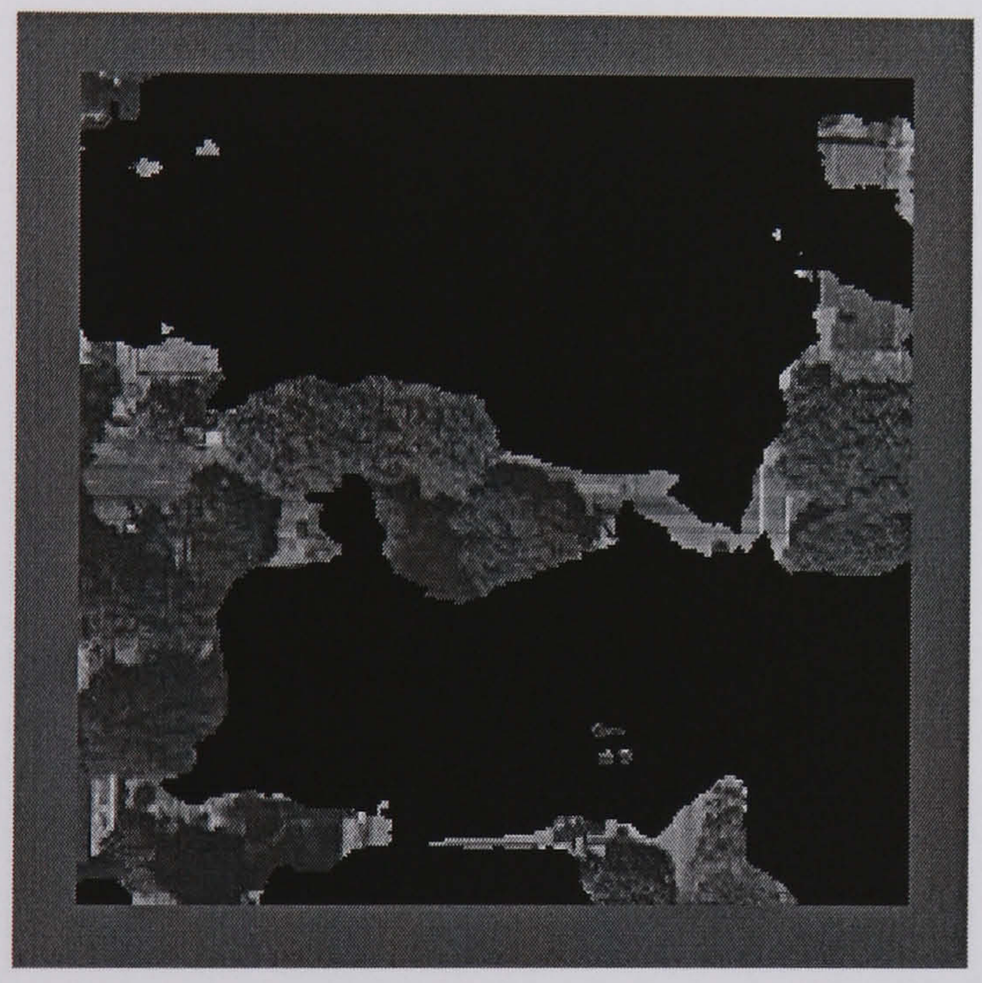
SGLDM /BPNN (70.67 %)

Figure E.41 Aerial Image 43 Results

Aerial Image 44



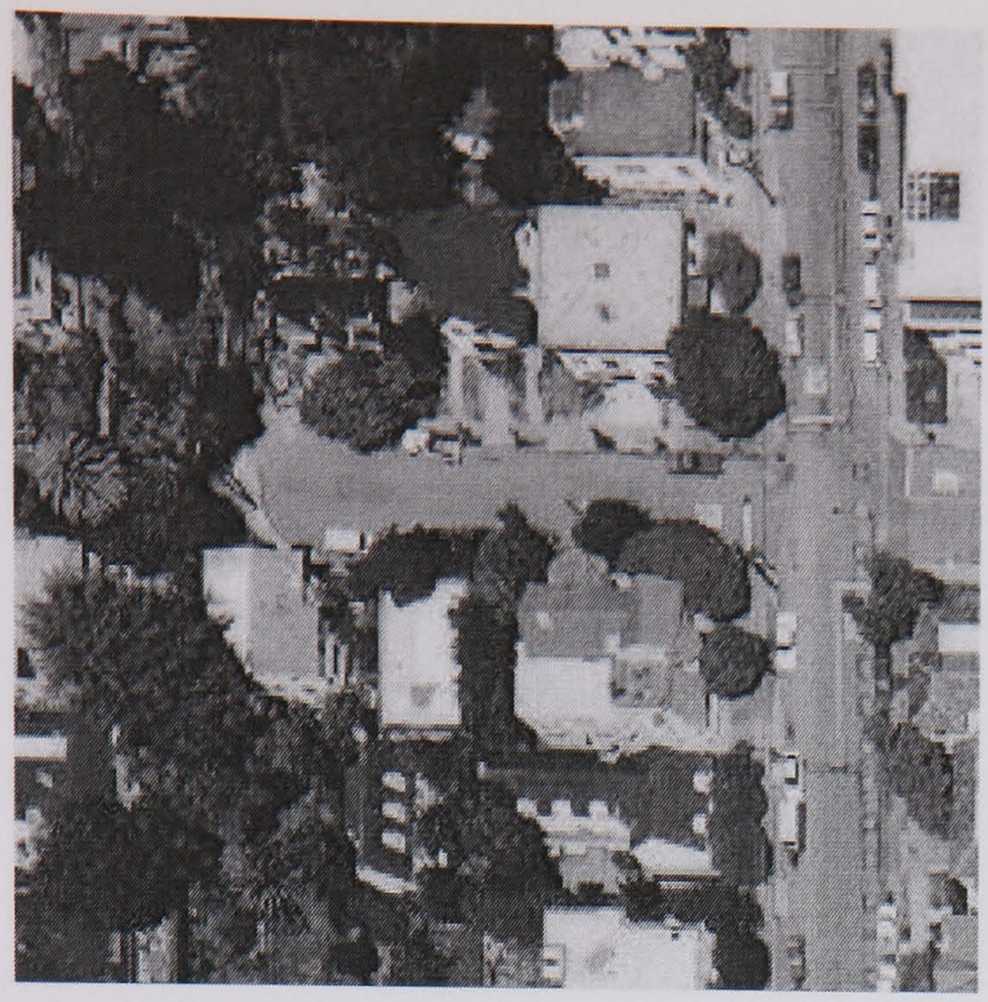
Hand Segmented Image



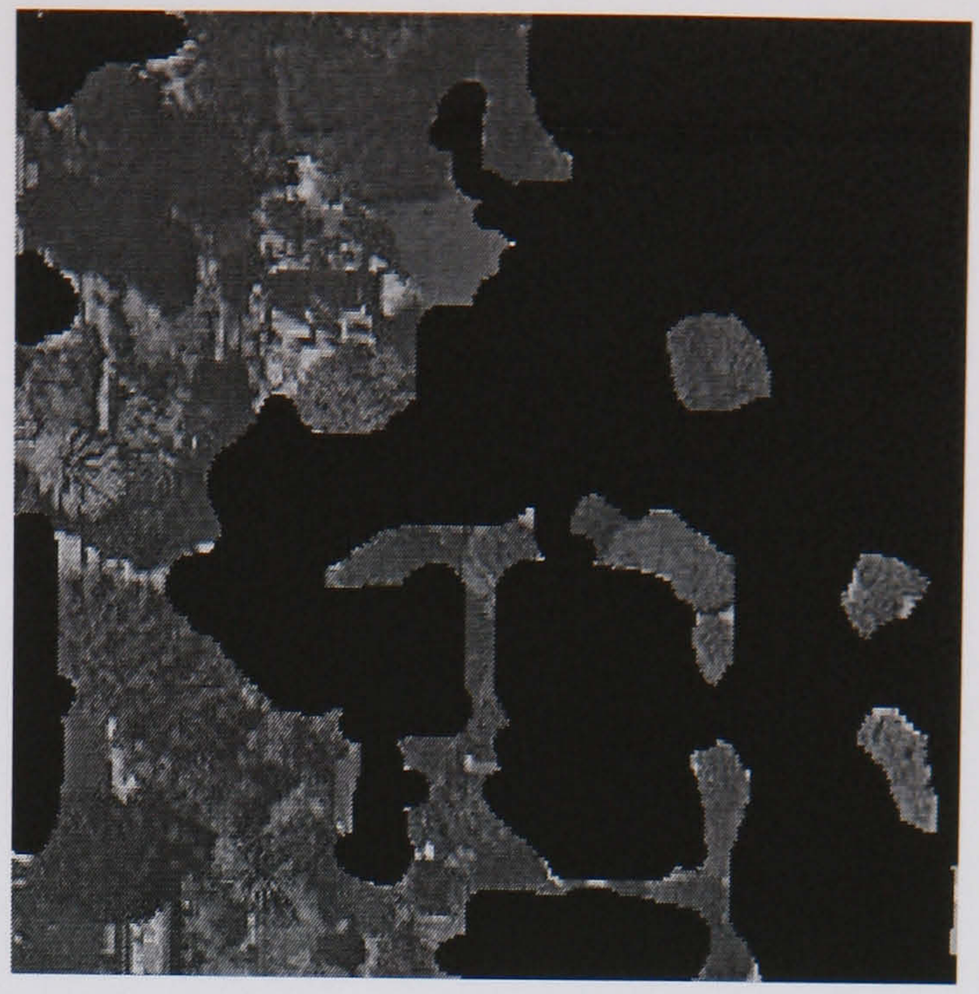
Hybrid Neural Network (87.76 %)

SGLDM /BPNN (62.92 %)

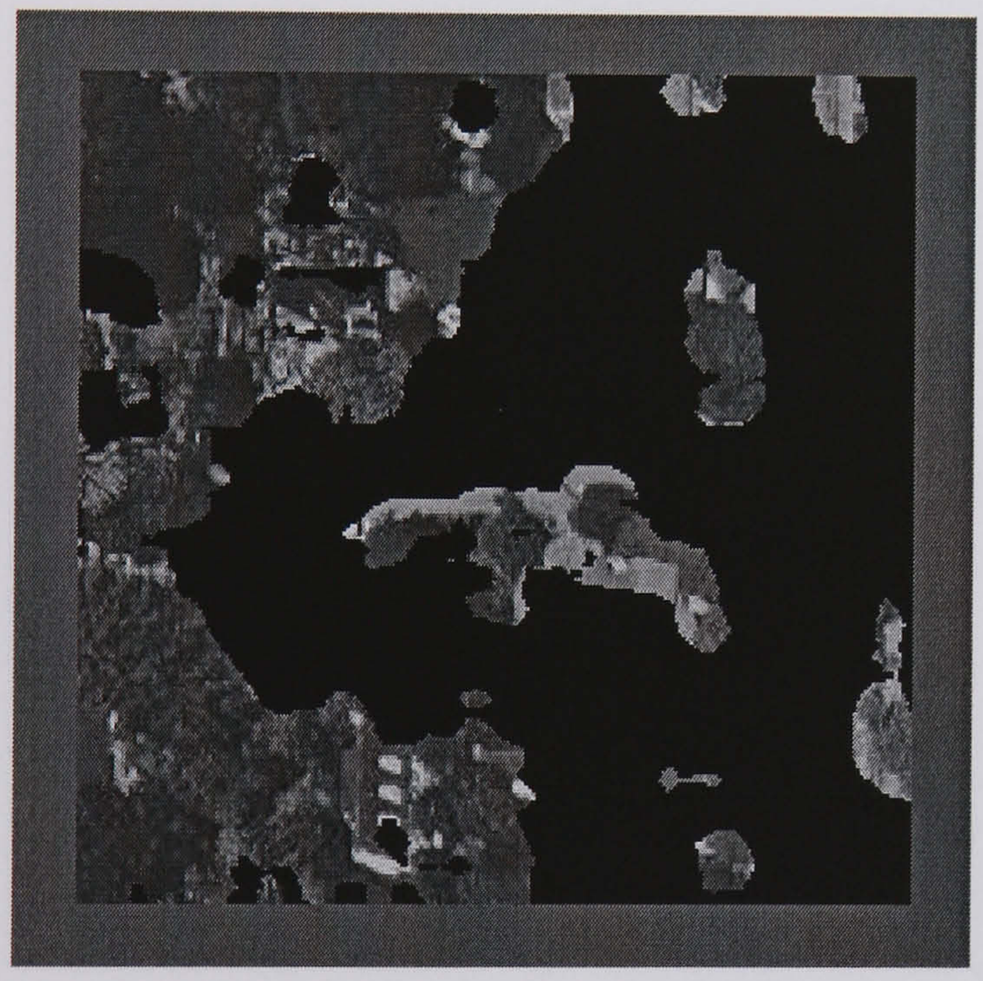
Figure E.42 Aerial Image 44 Results



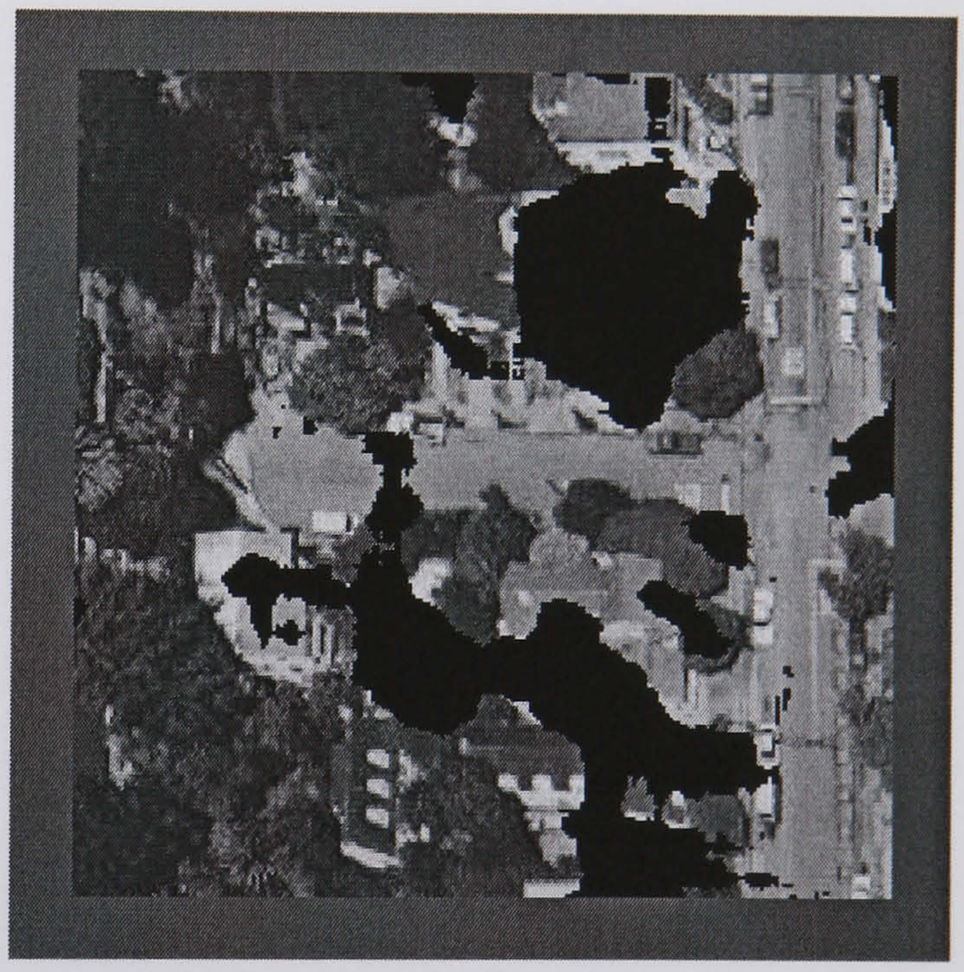
Aerial Image 45



Hand Segmented Image



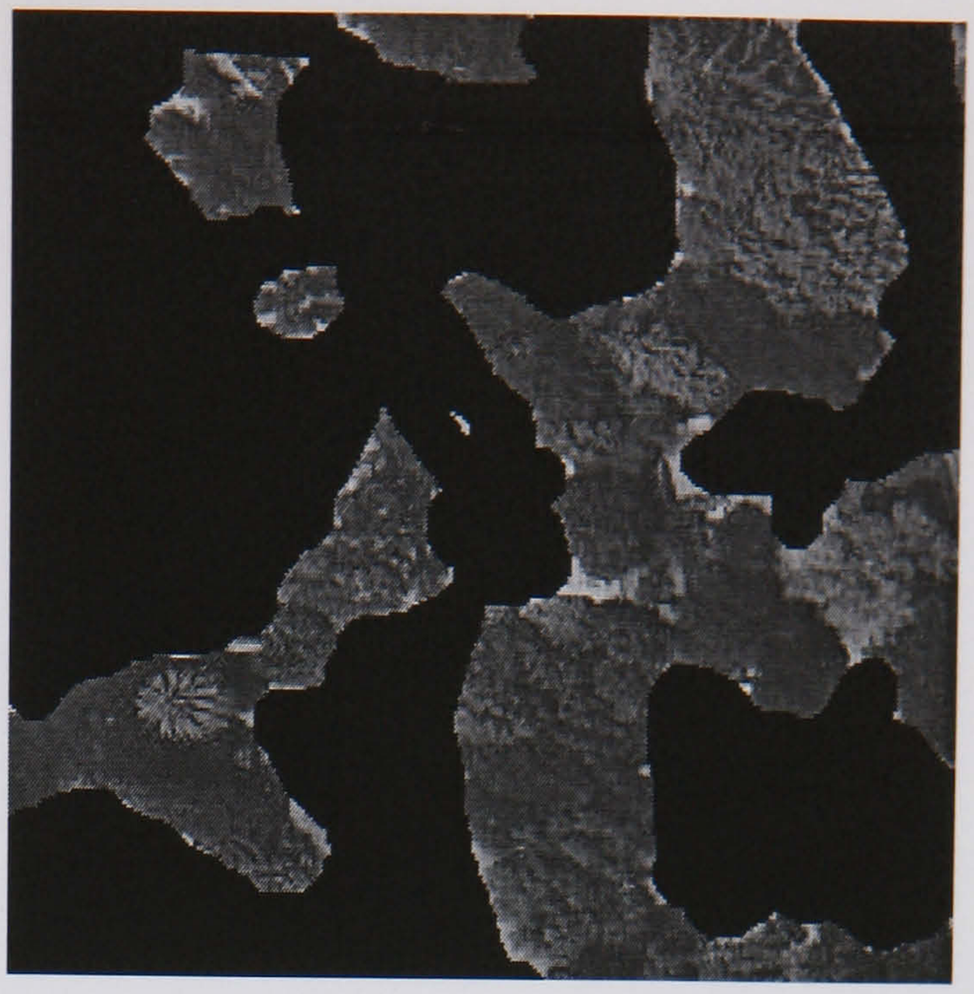
Hybrid Neural Network (88.09 %)



SGLDM /BPNN (67.64%)

Figure E.43 Aerial Image 45 Results

Aerial Image 46



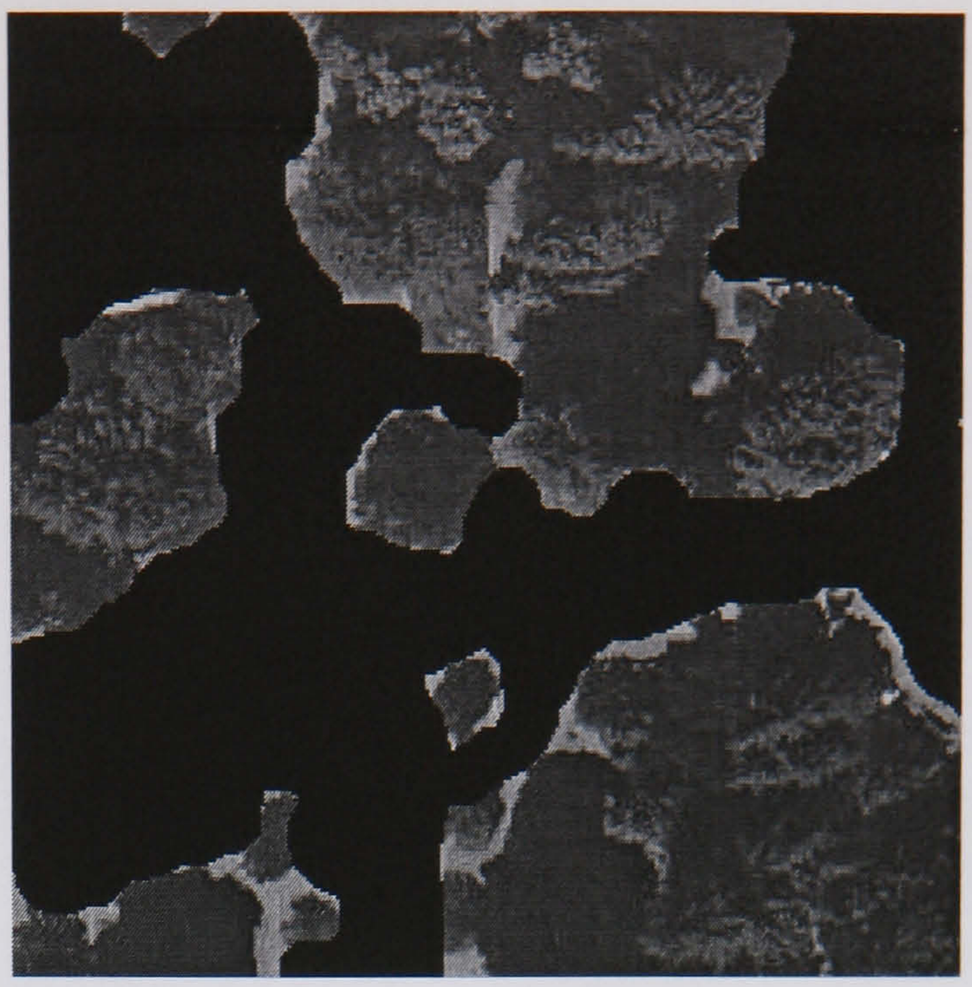
Hand Segmented Image

Hybrid Neural Network (87.44 %)

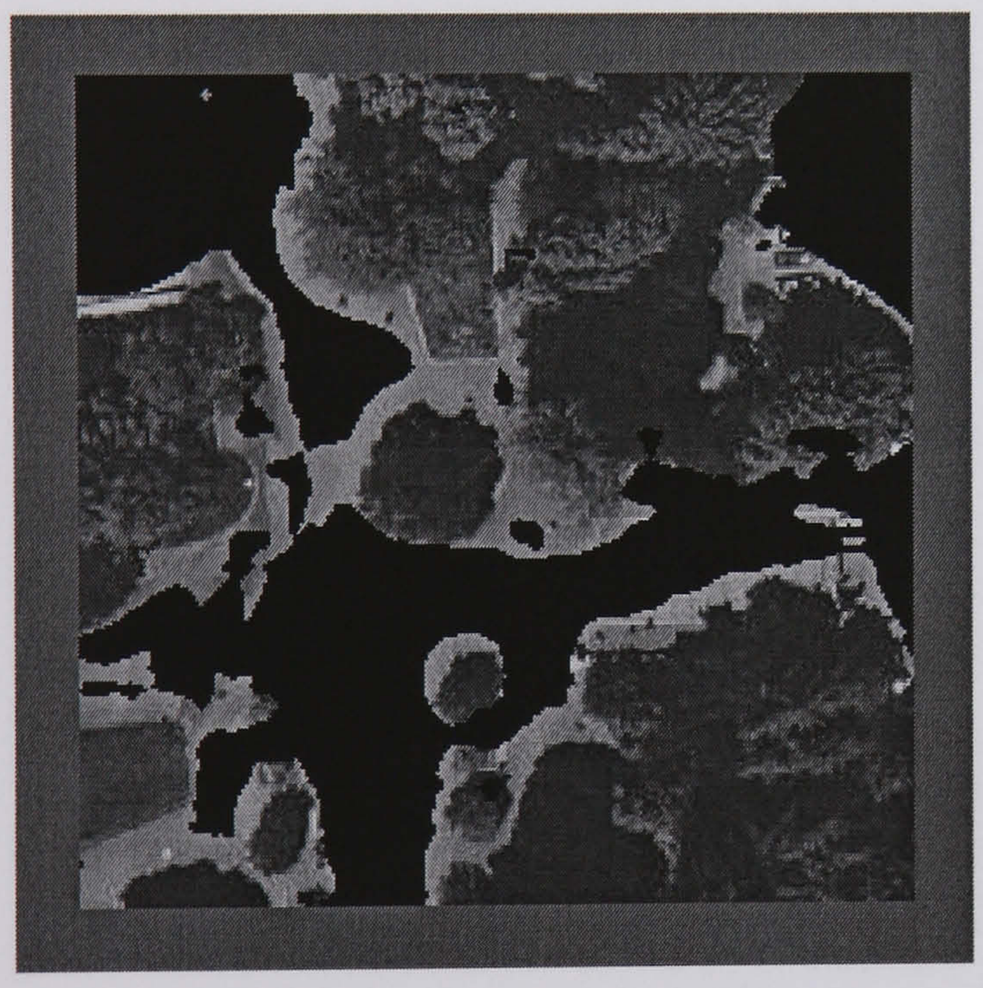
SGLDM /BPNN (68.76%)

Figure E.44 Aerial Image 46 Results

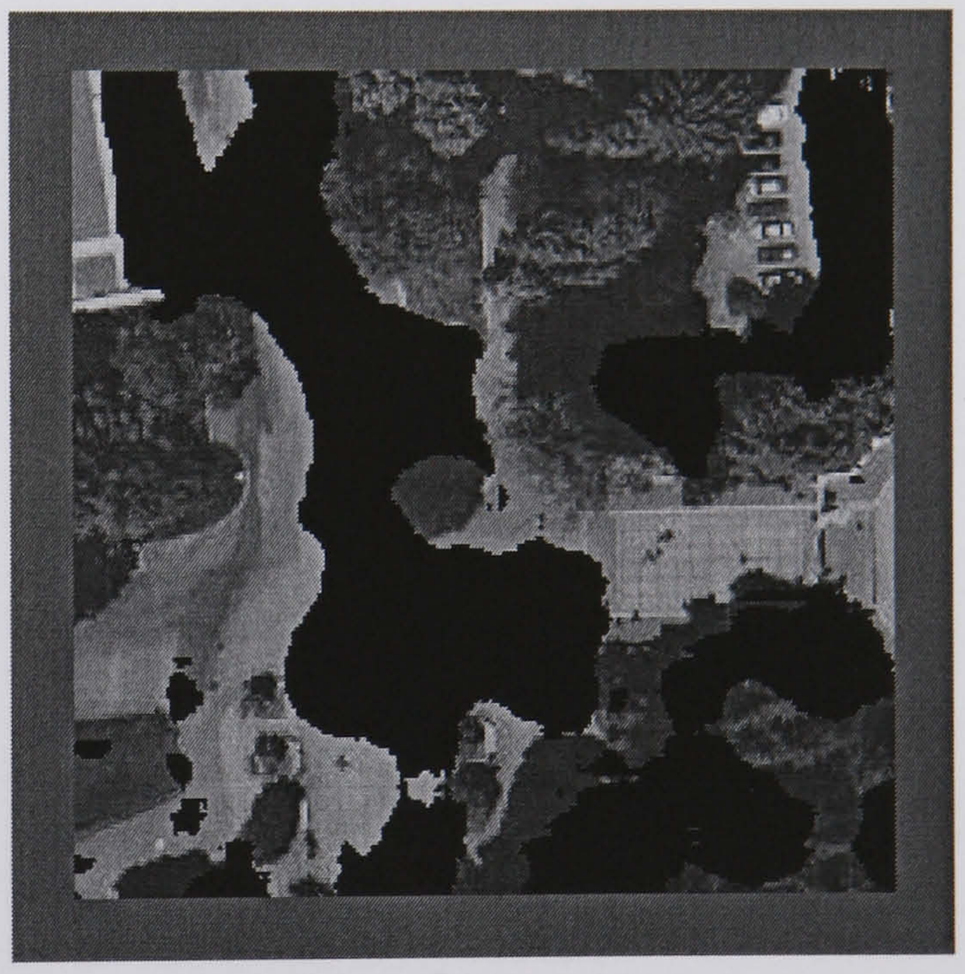
Aerial Image 47



Hand Segmented Image

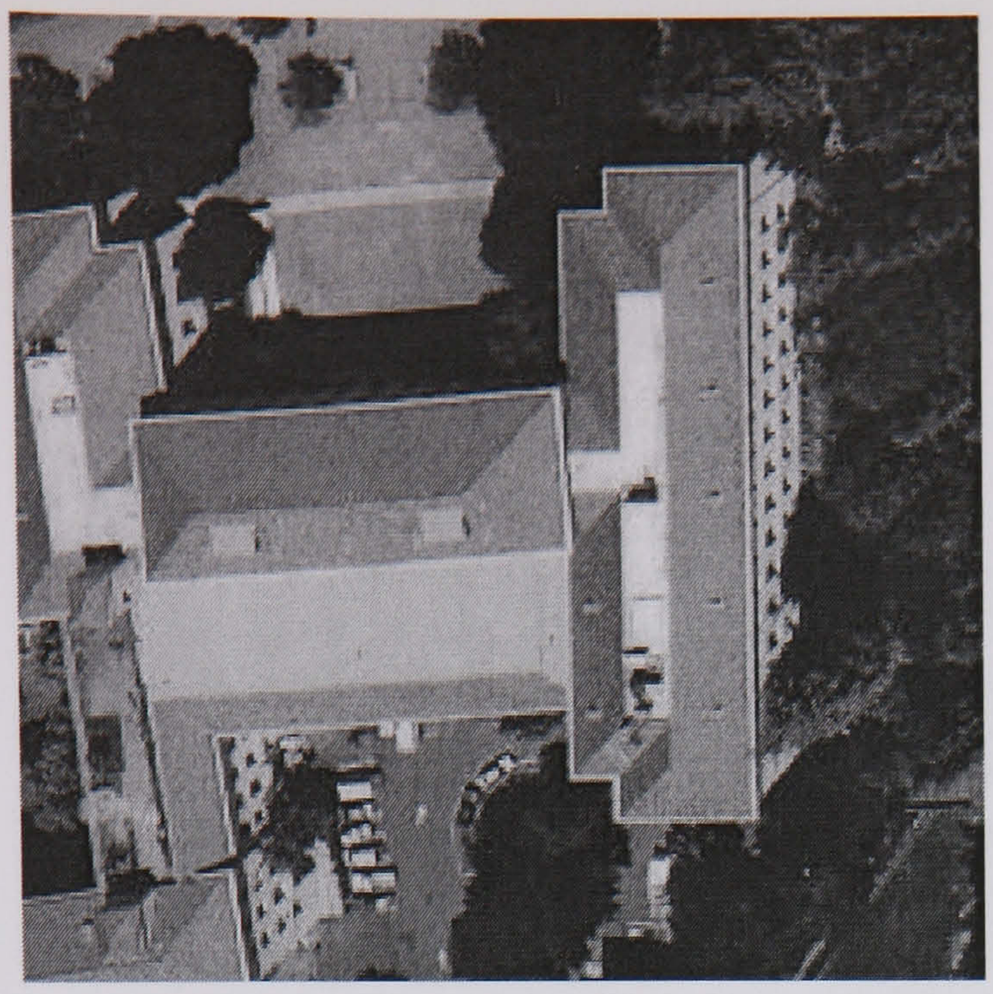


Hybrid Neural Network (87.10 %)

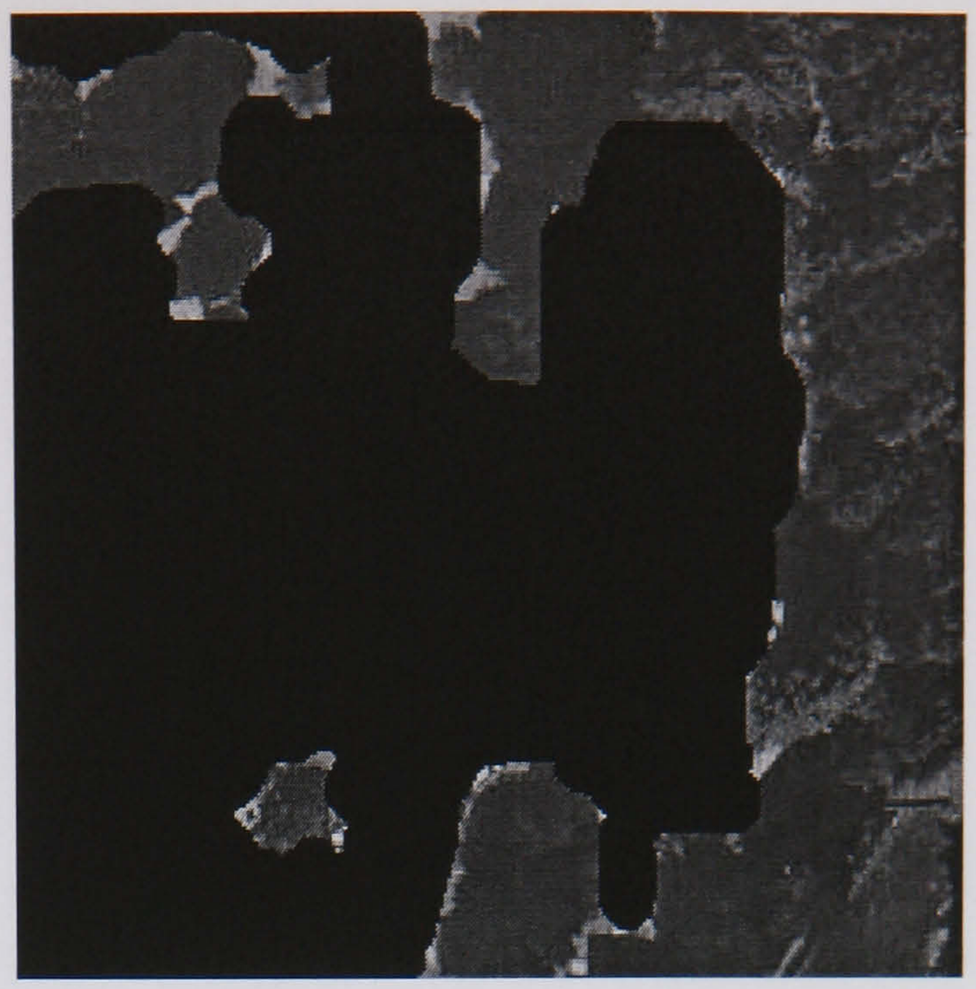


SGLDM /BPNN (67.35 %)

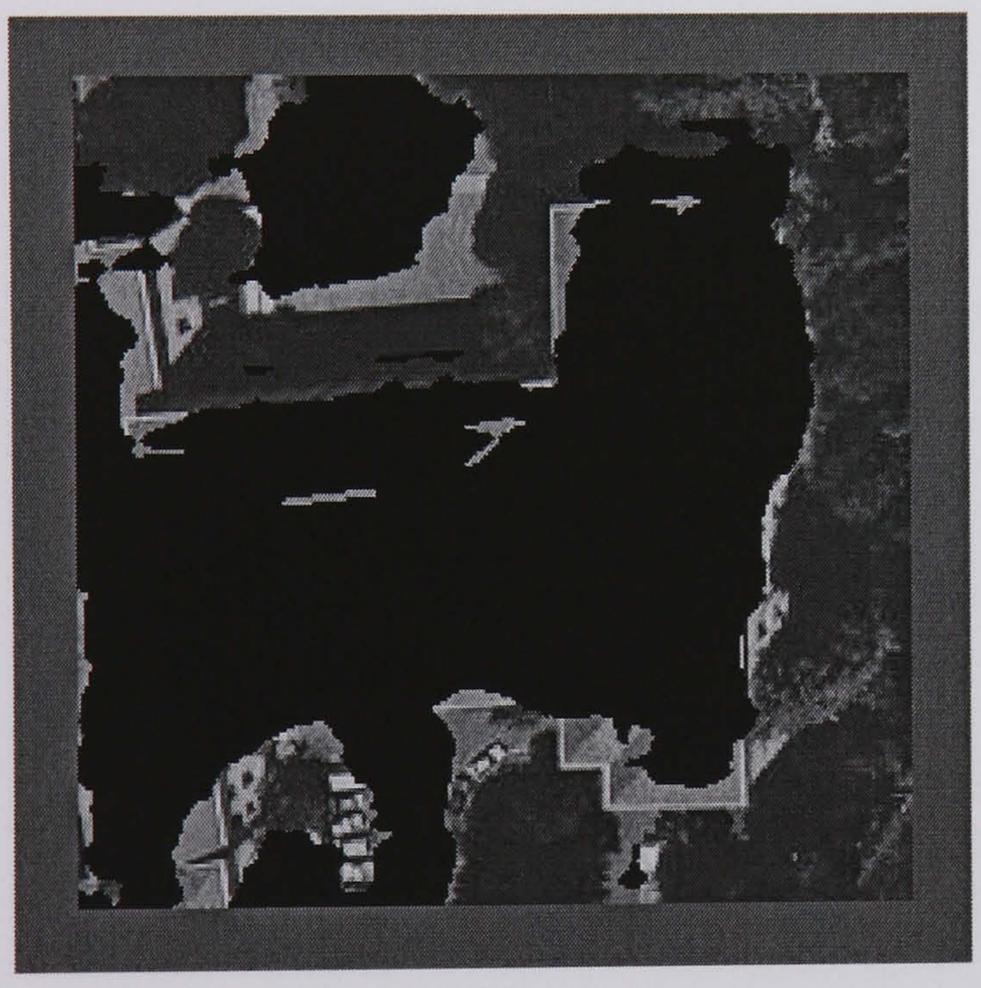
Figure E.45 Aerial Image 47 Results



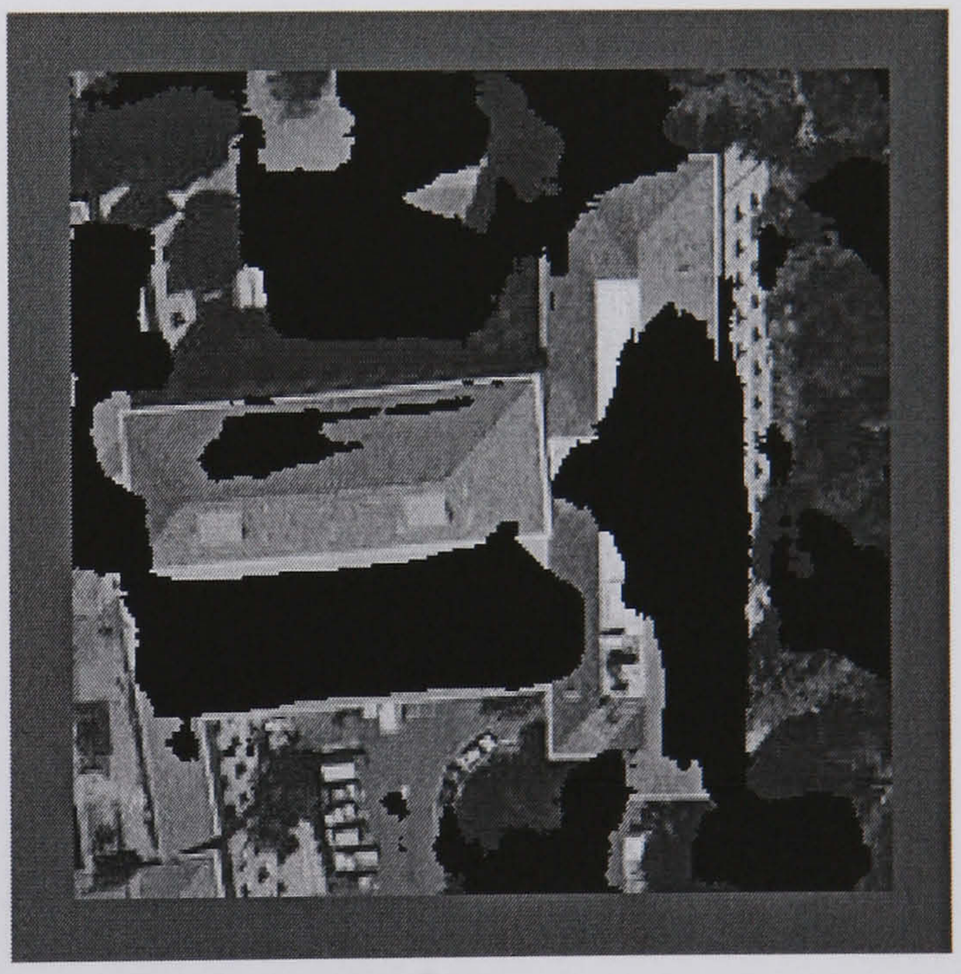
Aerial Image 48



Hand Segmented Image

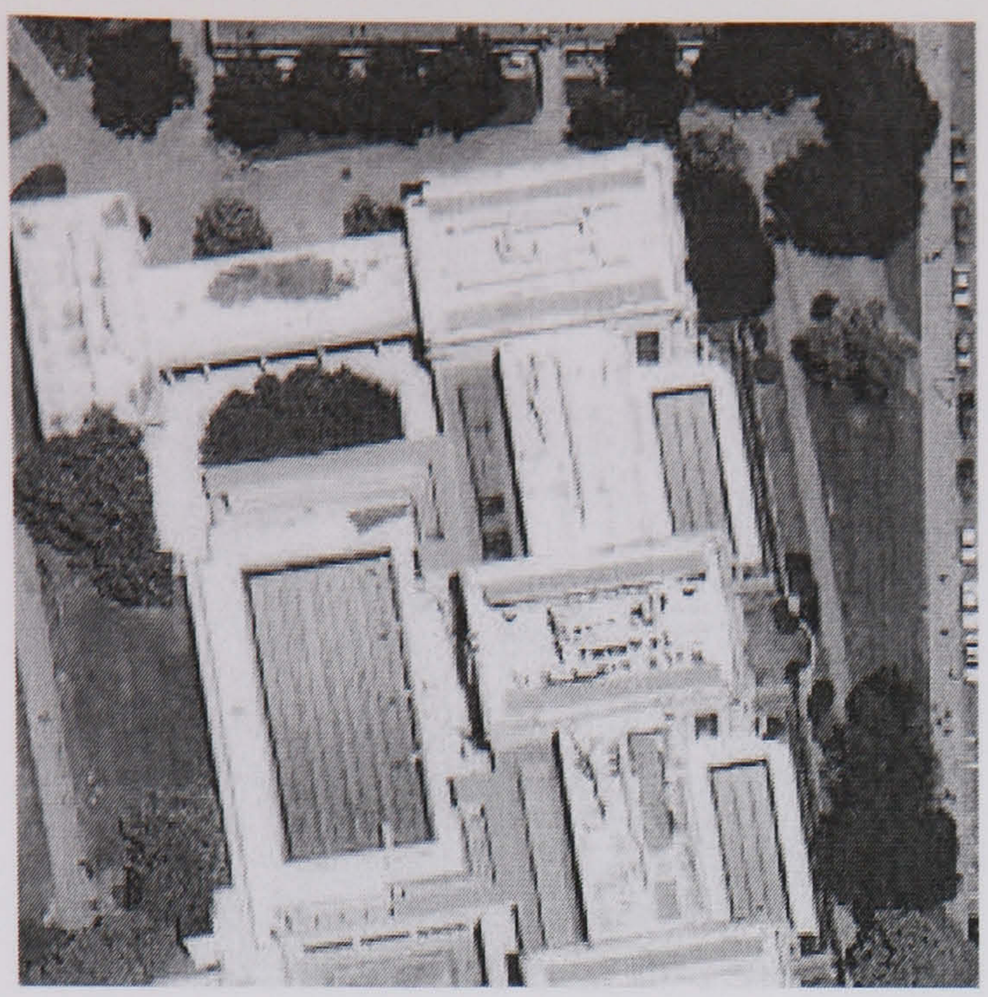


Hybrid Neural Network (87.40 %)

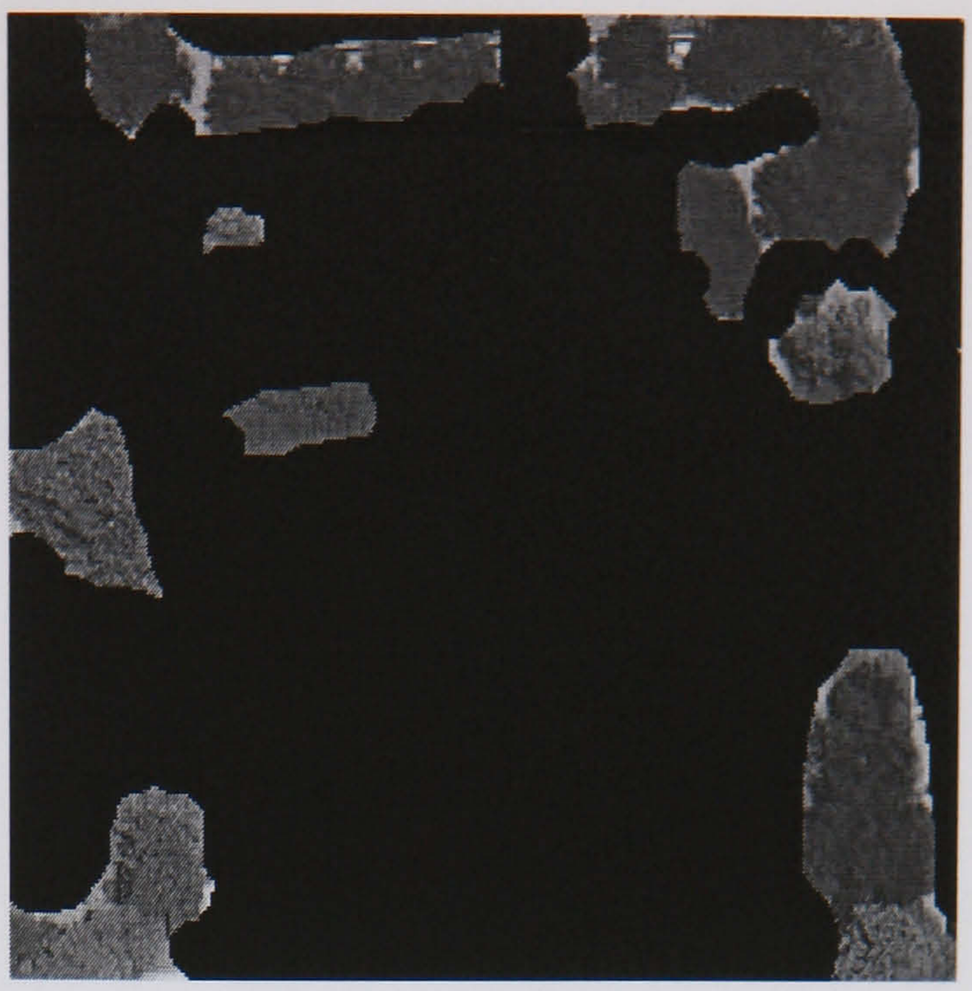


SGLDM /BPNN (62.85 %)

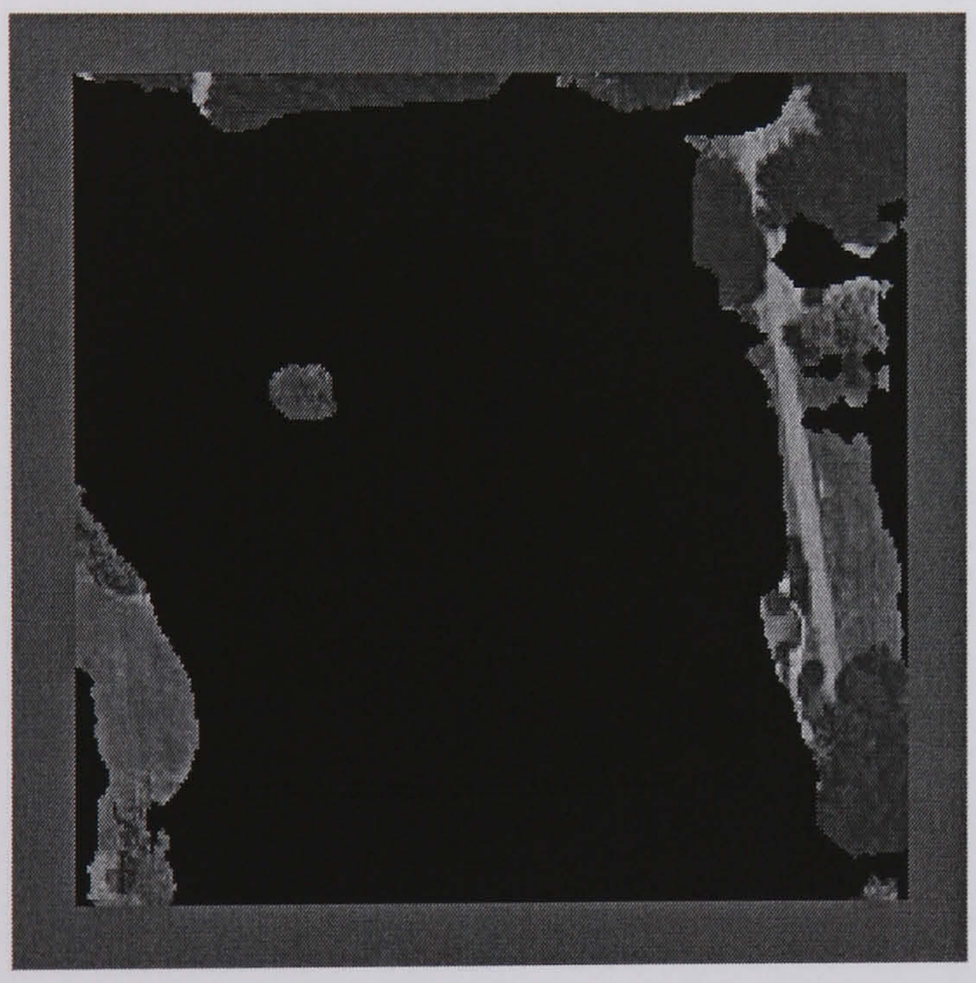
Figure E.46 Aerial Image 48 Results



Aerial Image 49



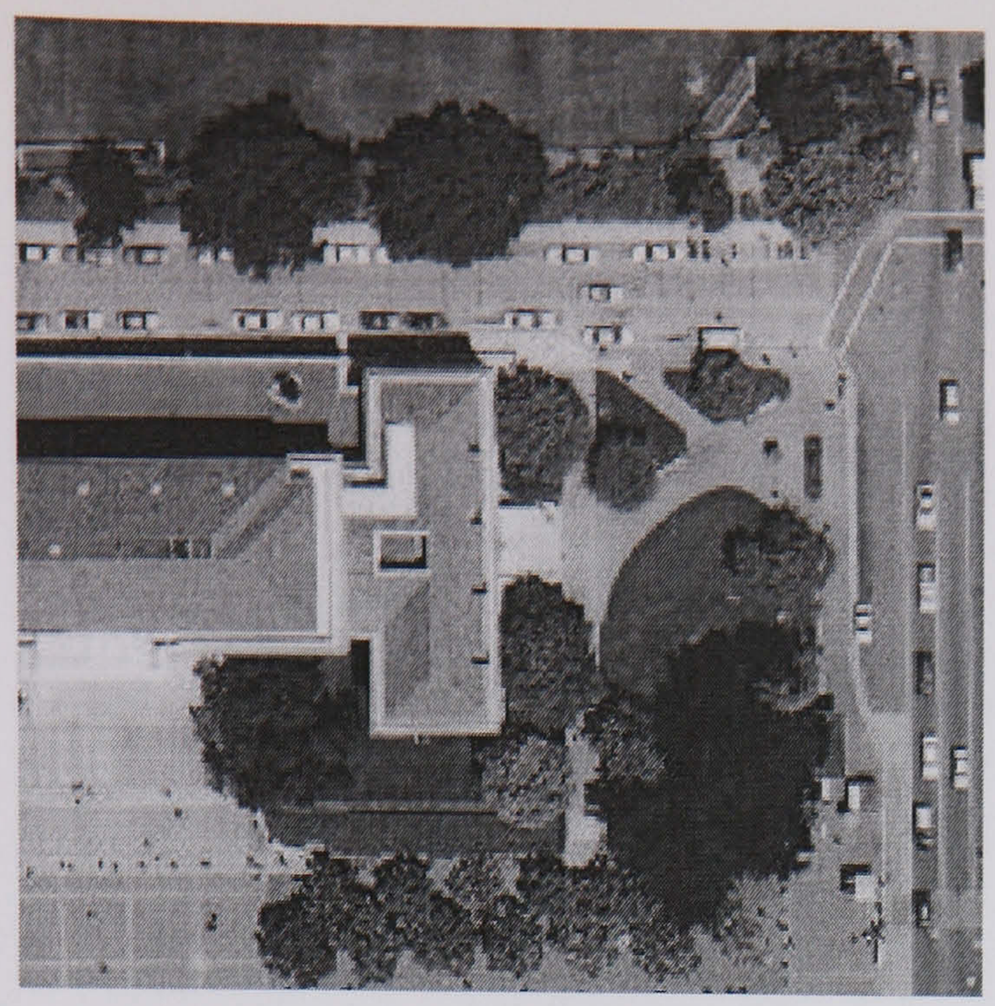
Hand Segmented Image



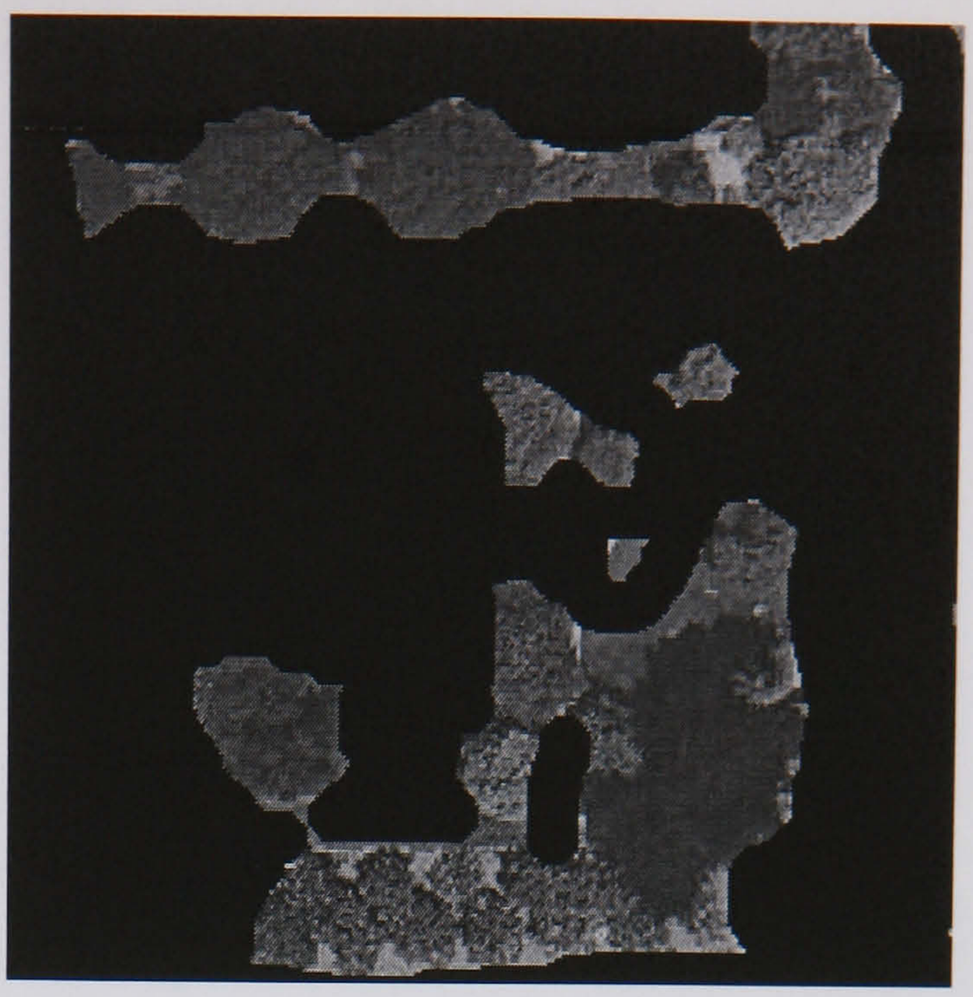
Hybrid Neural Network (89.17 %)

SGLDM /BPNN (82.53 %)

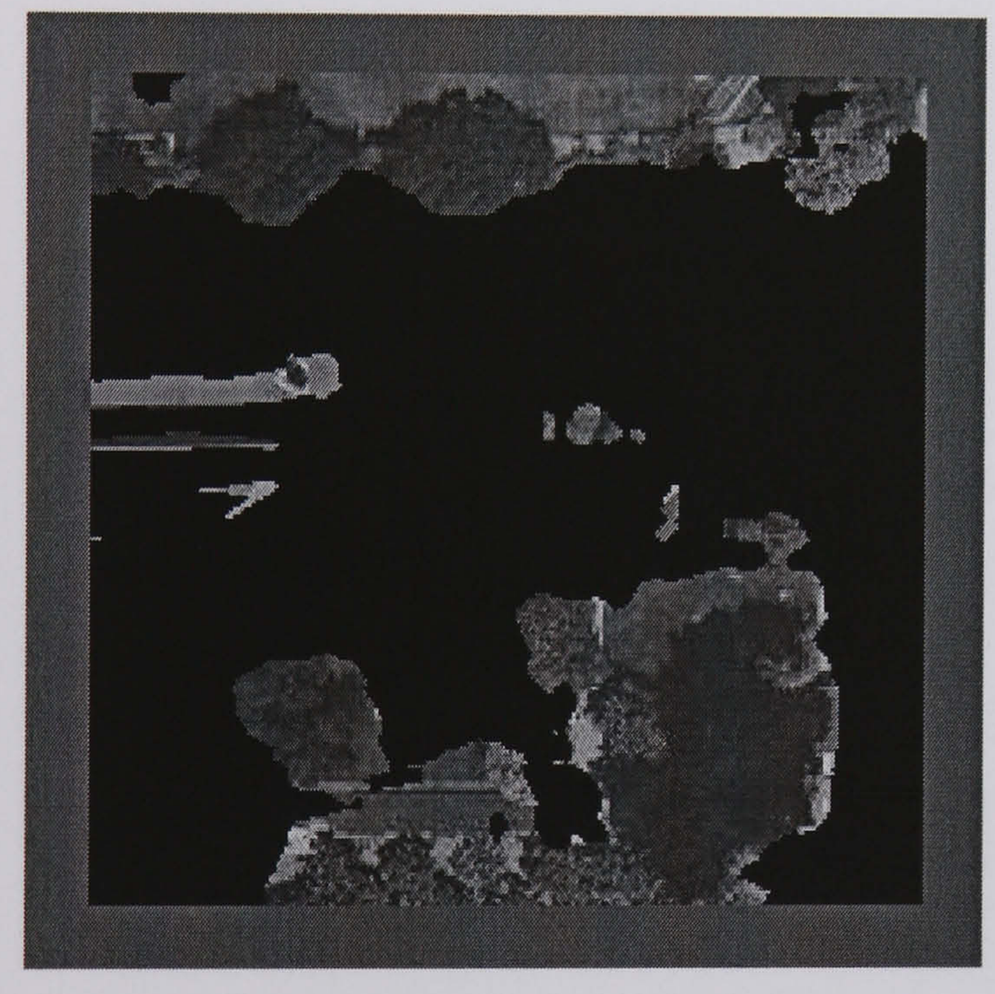
Figure E.47 Aerial Image 49 Results



Aerial Image 50



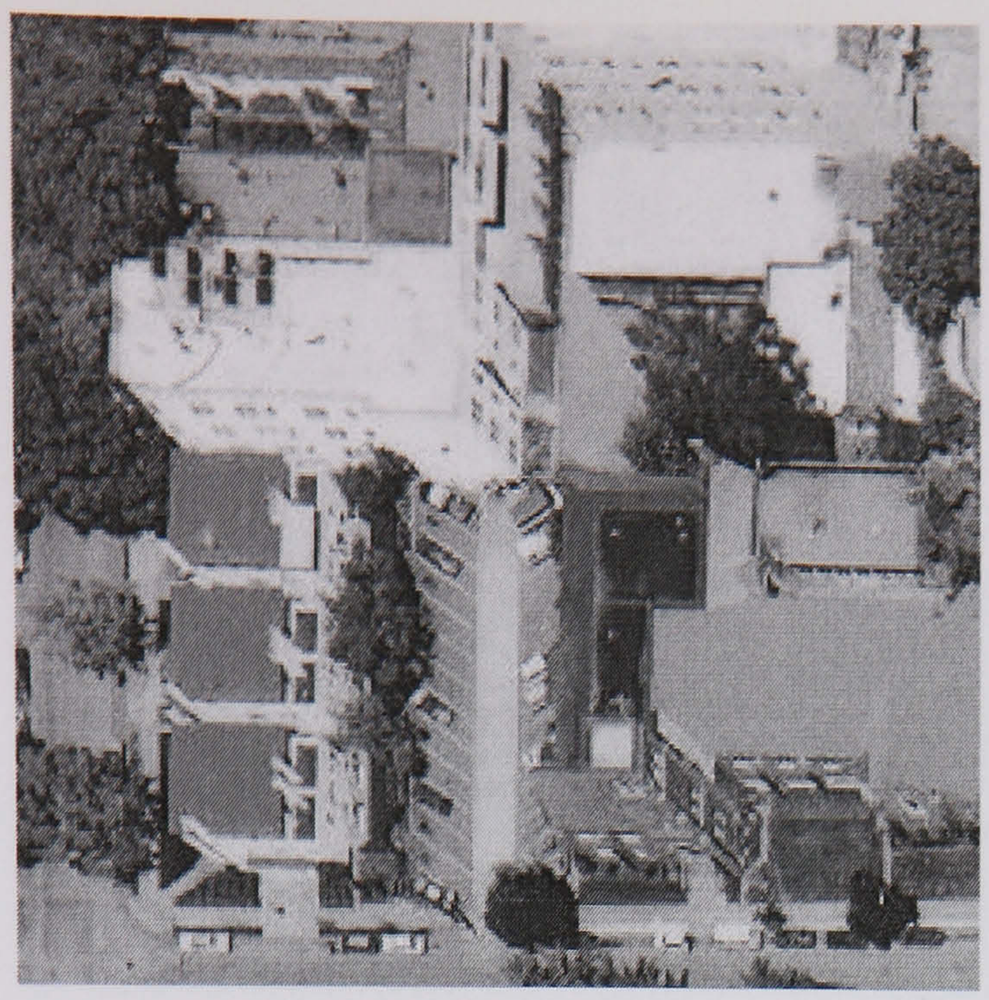
Hand Segmented Image



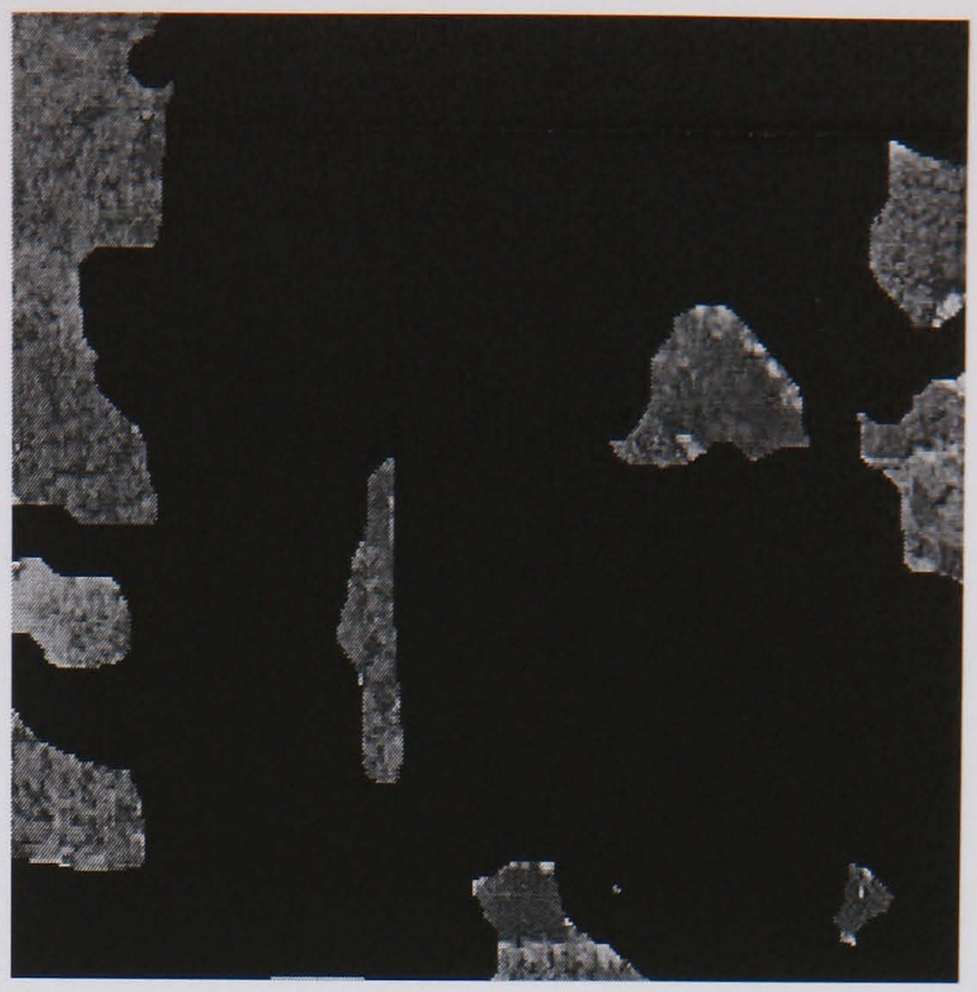
Hybrid Neural Network (86.88%)

SGLDM /BPNN (60.27 %)

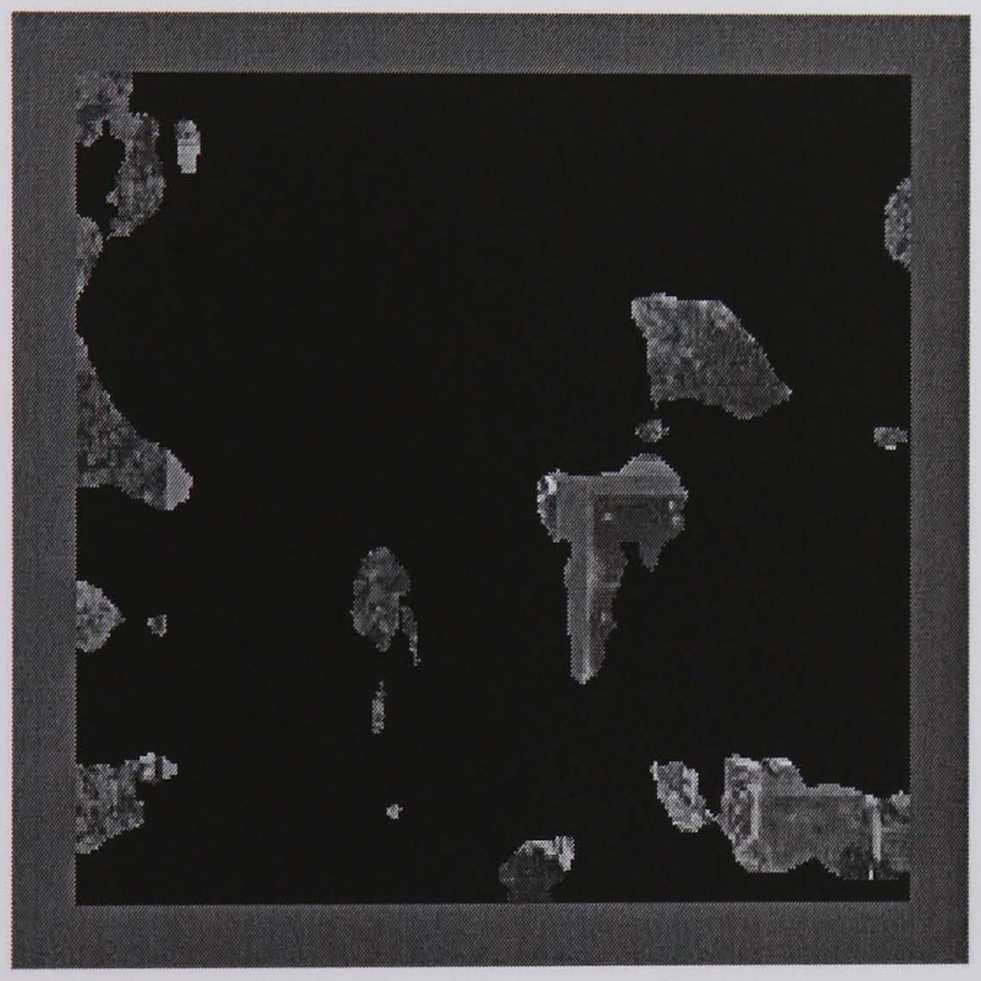
Figure E.48 Aerial Image 50 Results



Aerial Image 51



Hand Segmented Image

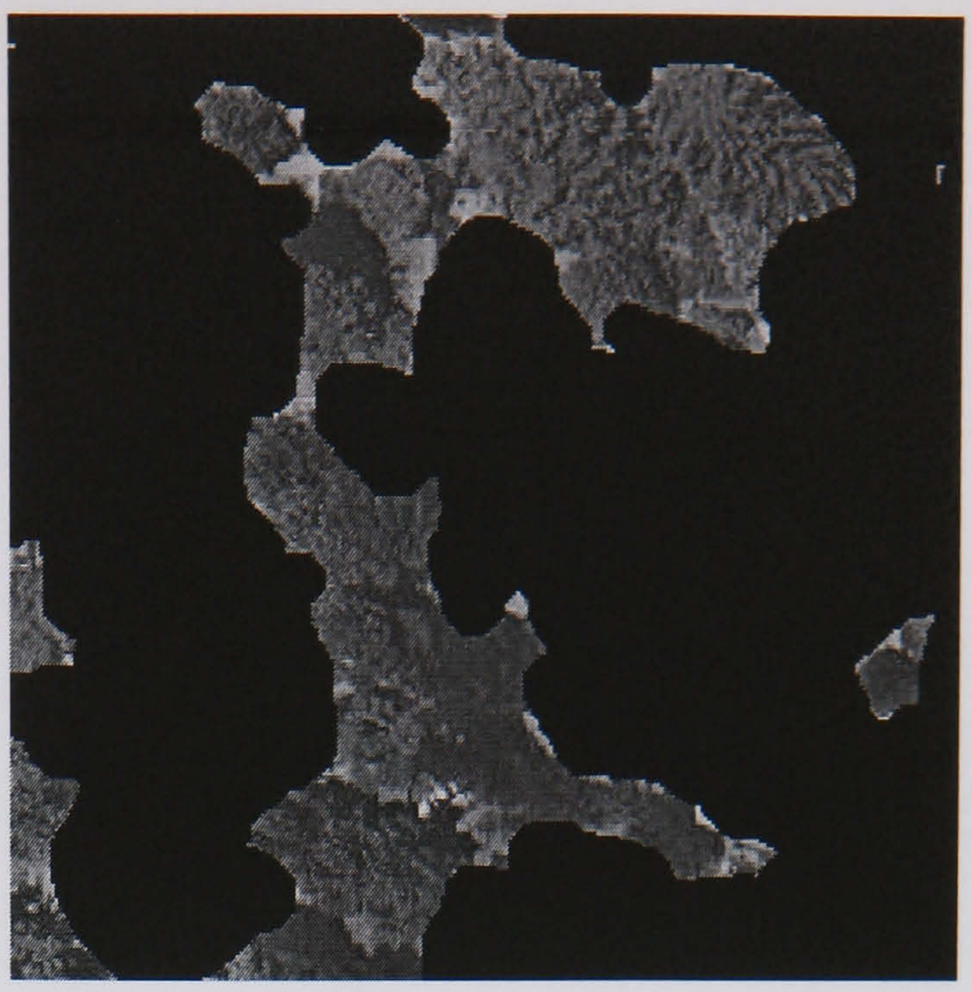


Hybrid Neural Network (91.06 %)

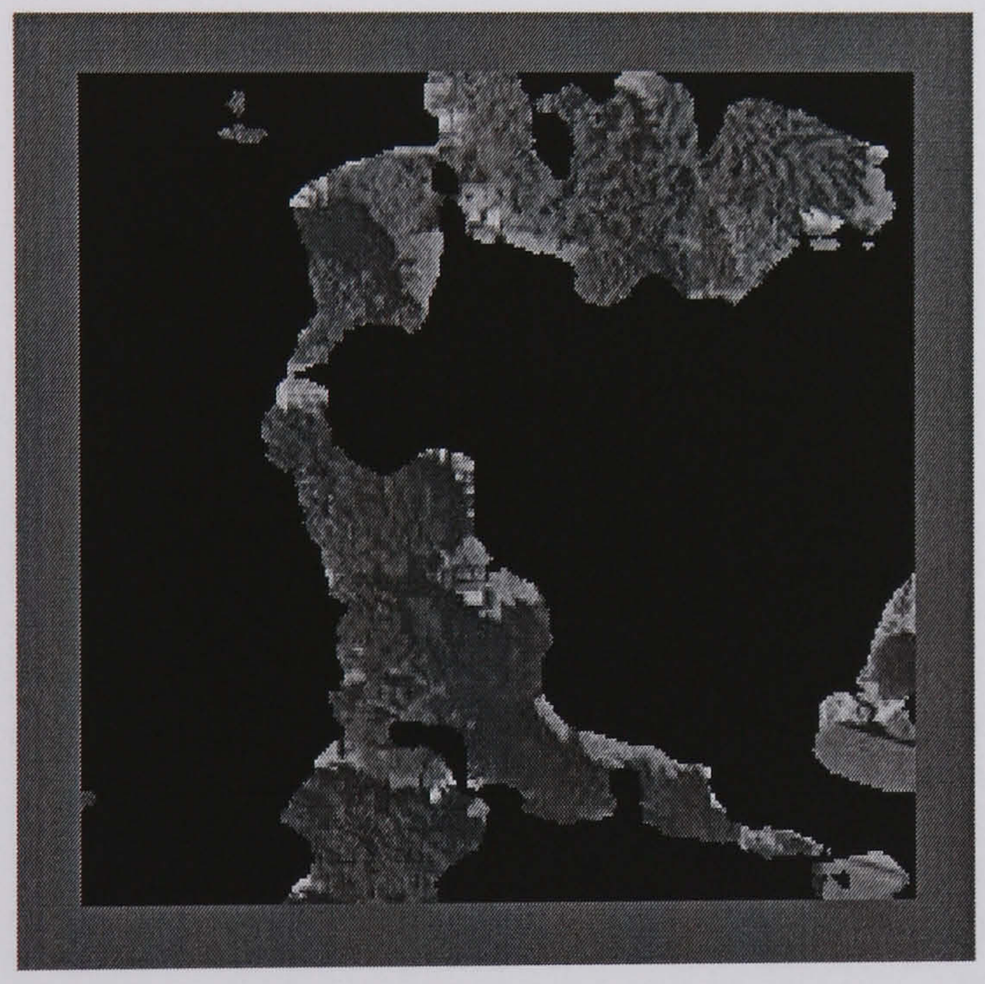
SGLDM /BPNN (67.32 %)

Figure E.49 Aerial Image 51 Results

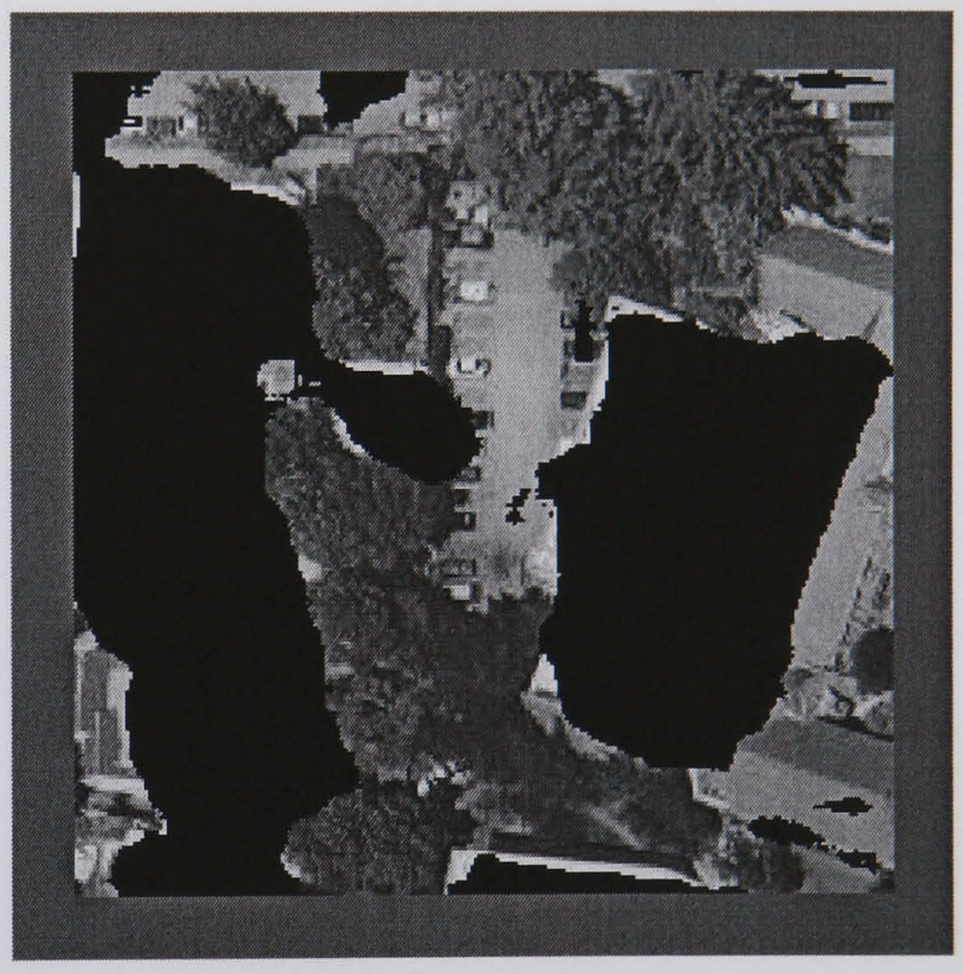
Aerial Image 52



Hand Segmented Image



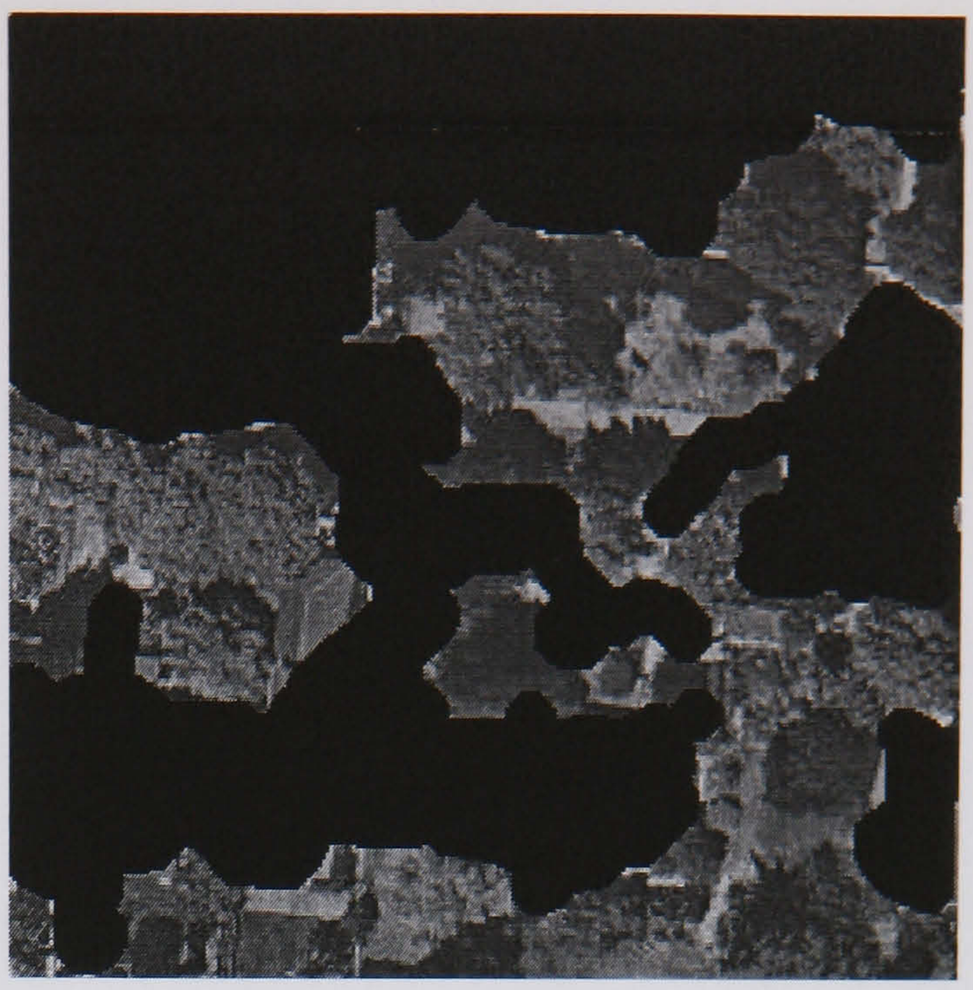
Hybrid Neural Network (90.83 %)



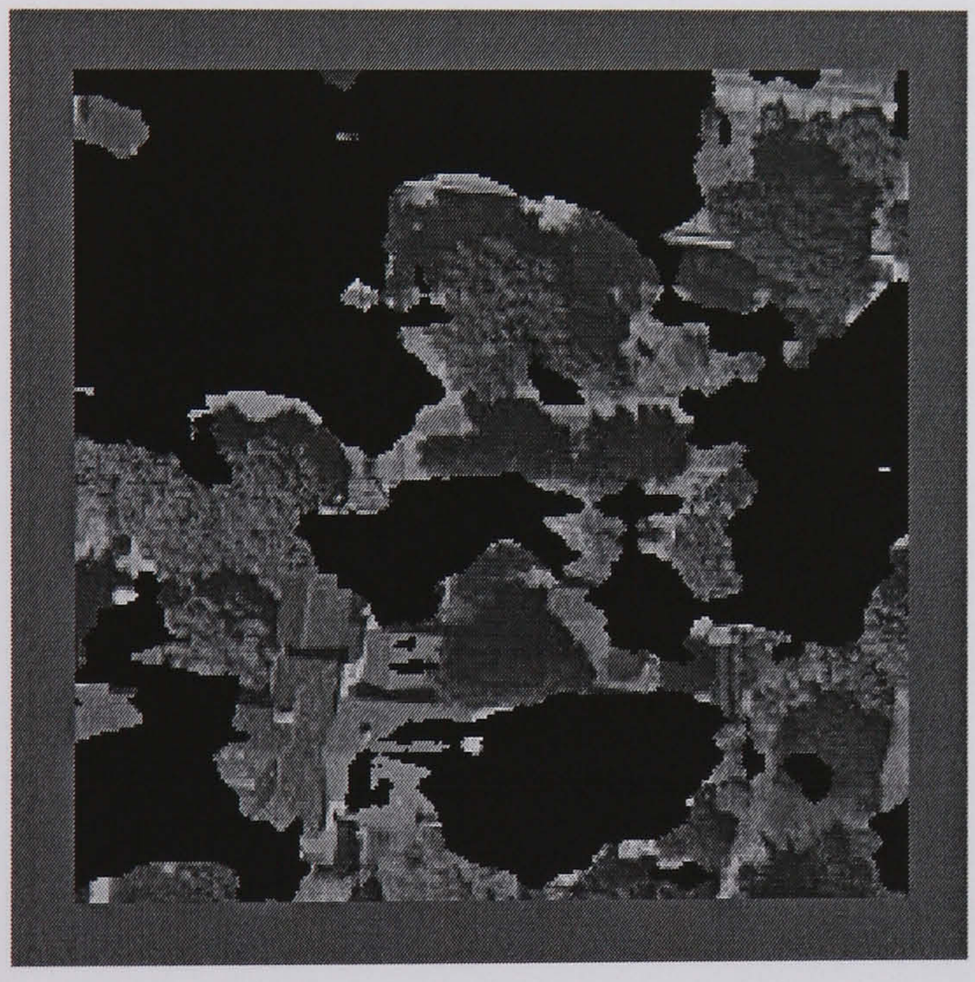
SGLDM /BPNN (77.88 %)

Figure E.50 Aerial Image 52 Results

Aerial Image 53



Hand Segmented Image

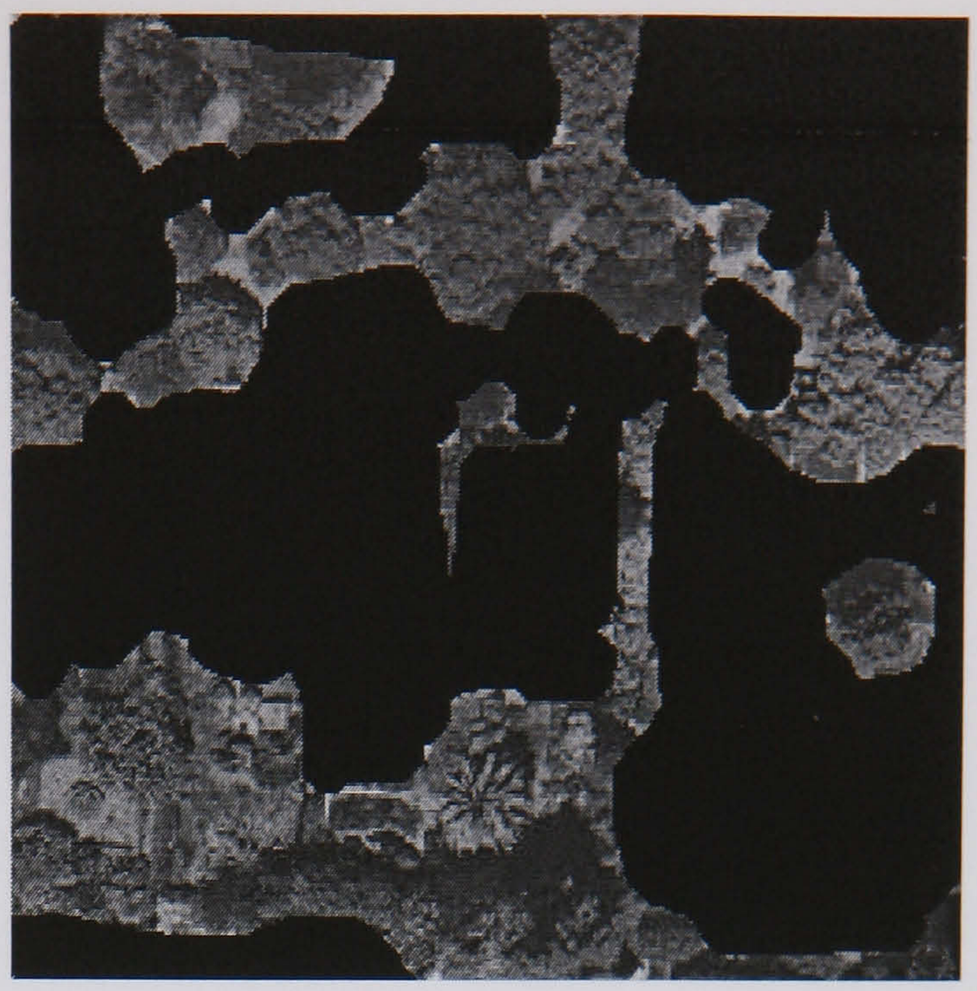


Hybrid Neural Network (83.07 %)

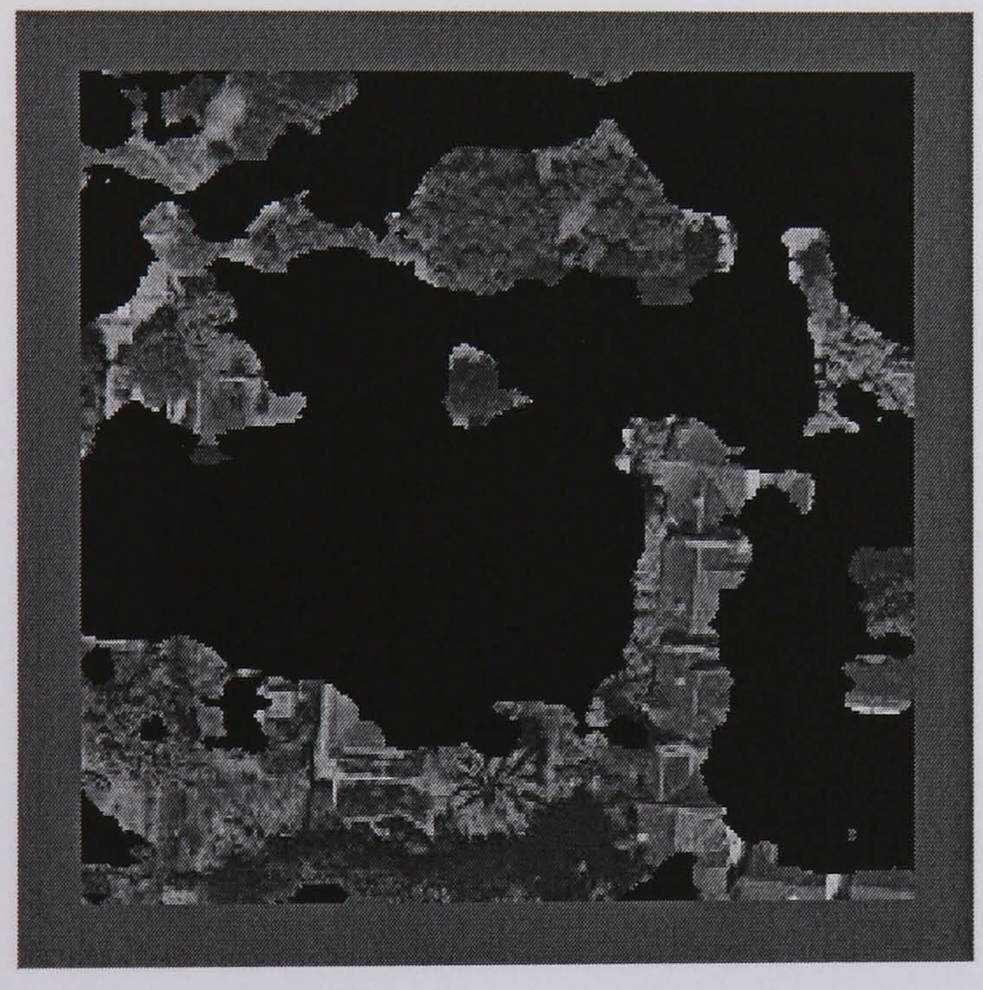
SGLDM /BPNN (65.96 %)

Figure E.51 Aerial Image 53 Results

Aerial Image 54



Hand Segmented Image

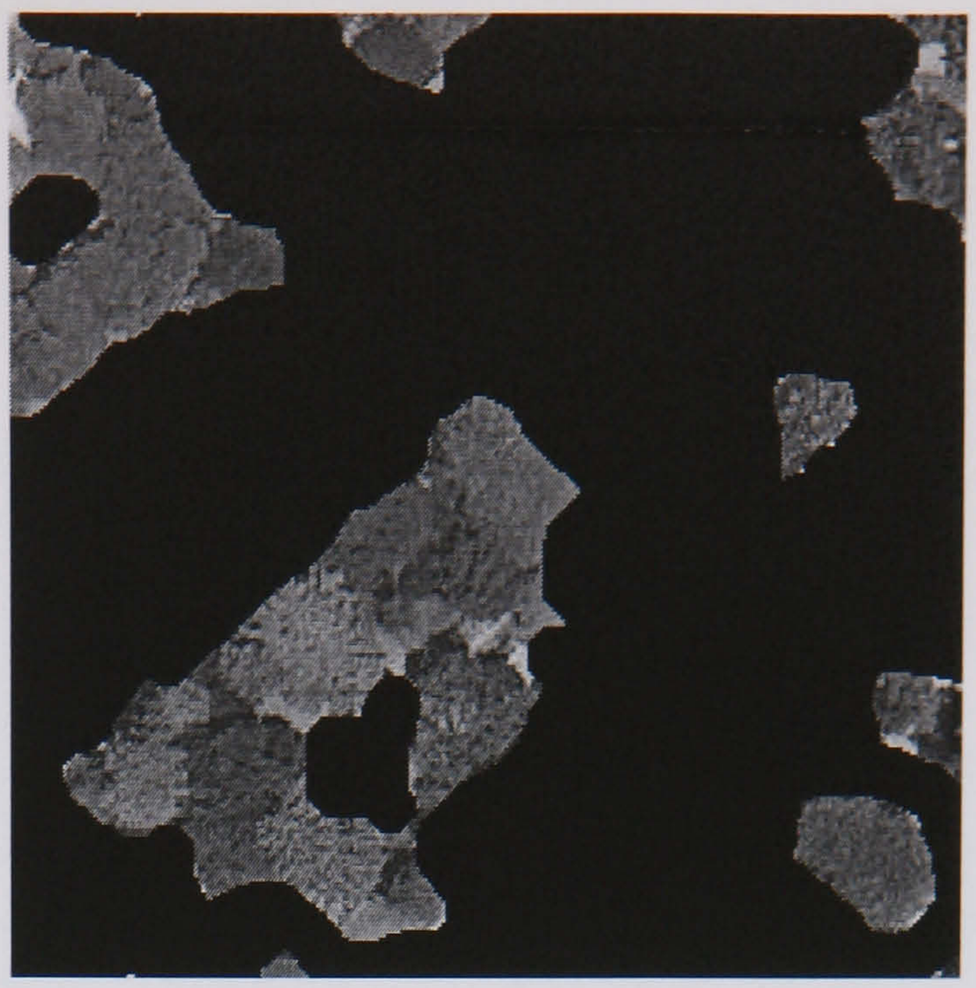


Hybrid Neural Network (83.15 %)

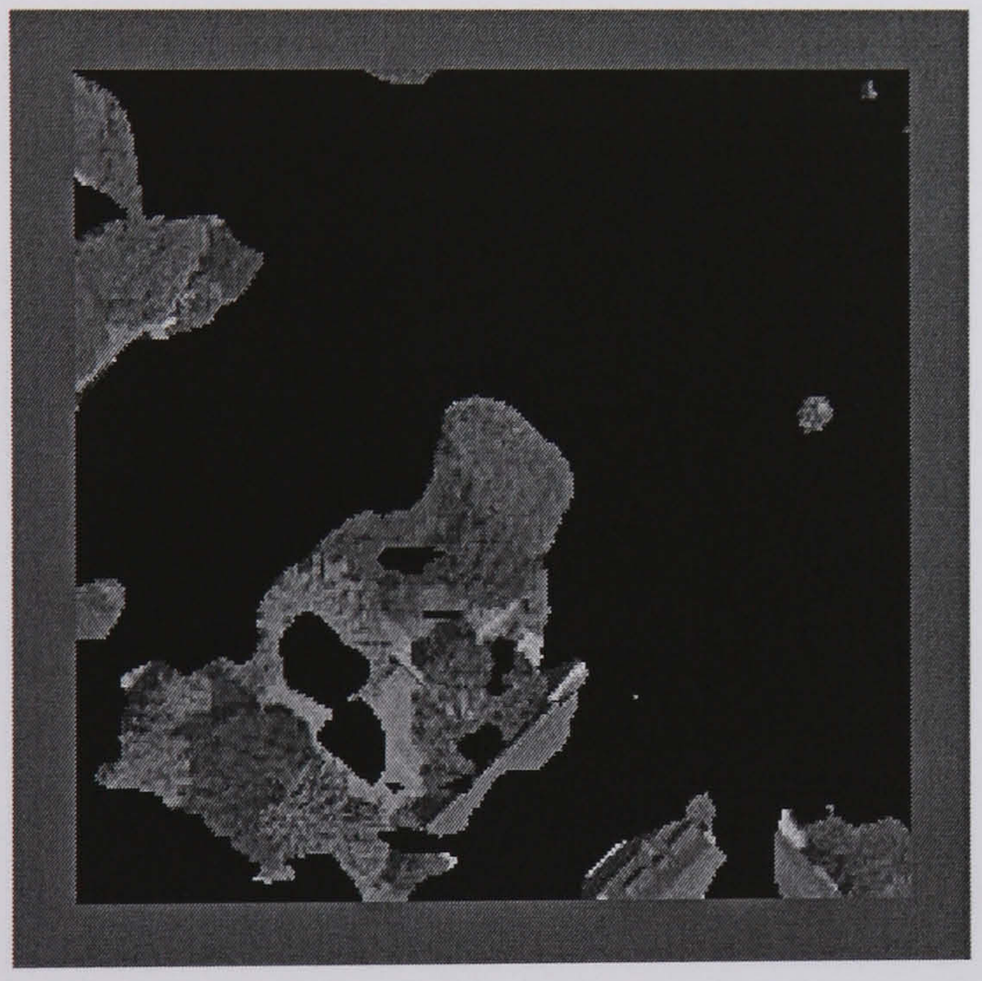
SGLDM /BPNN (68.58 %)

Figure E.52 Aerial Image 54 Results

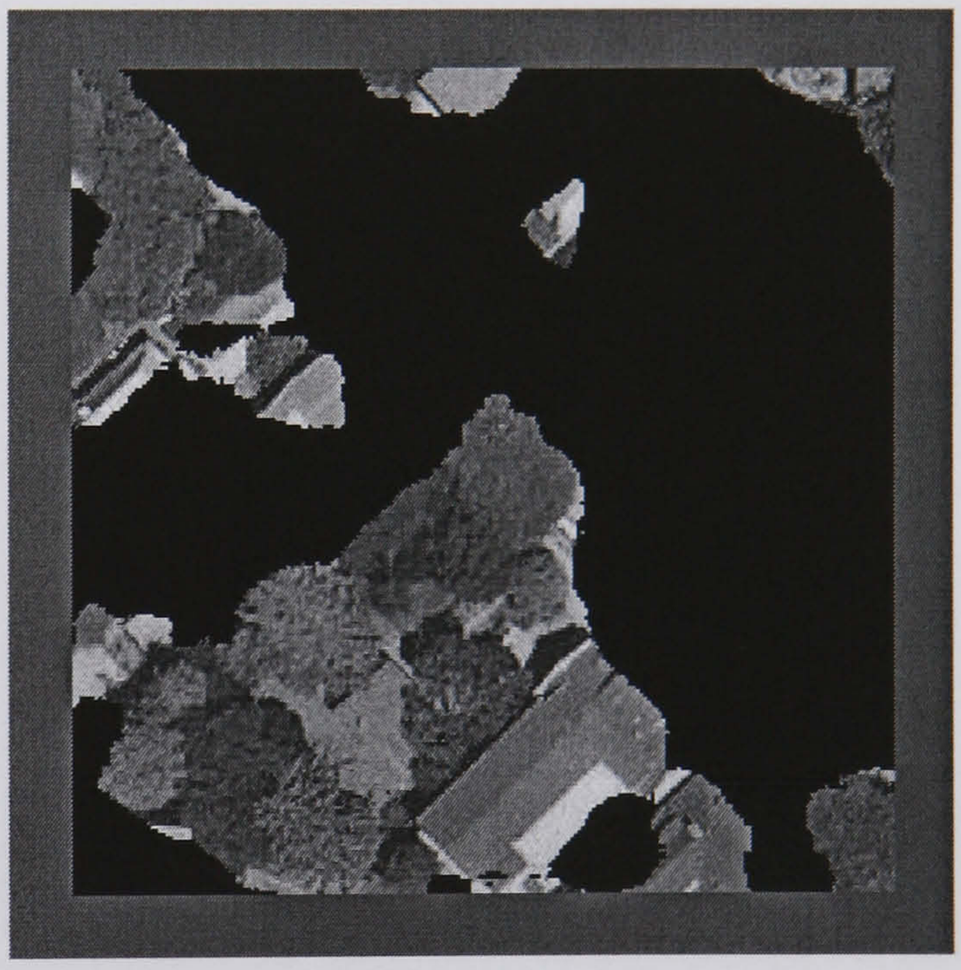
Aerial Image 55



Hand Segmented Image

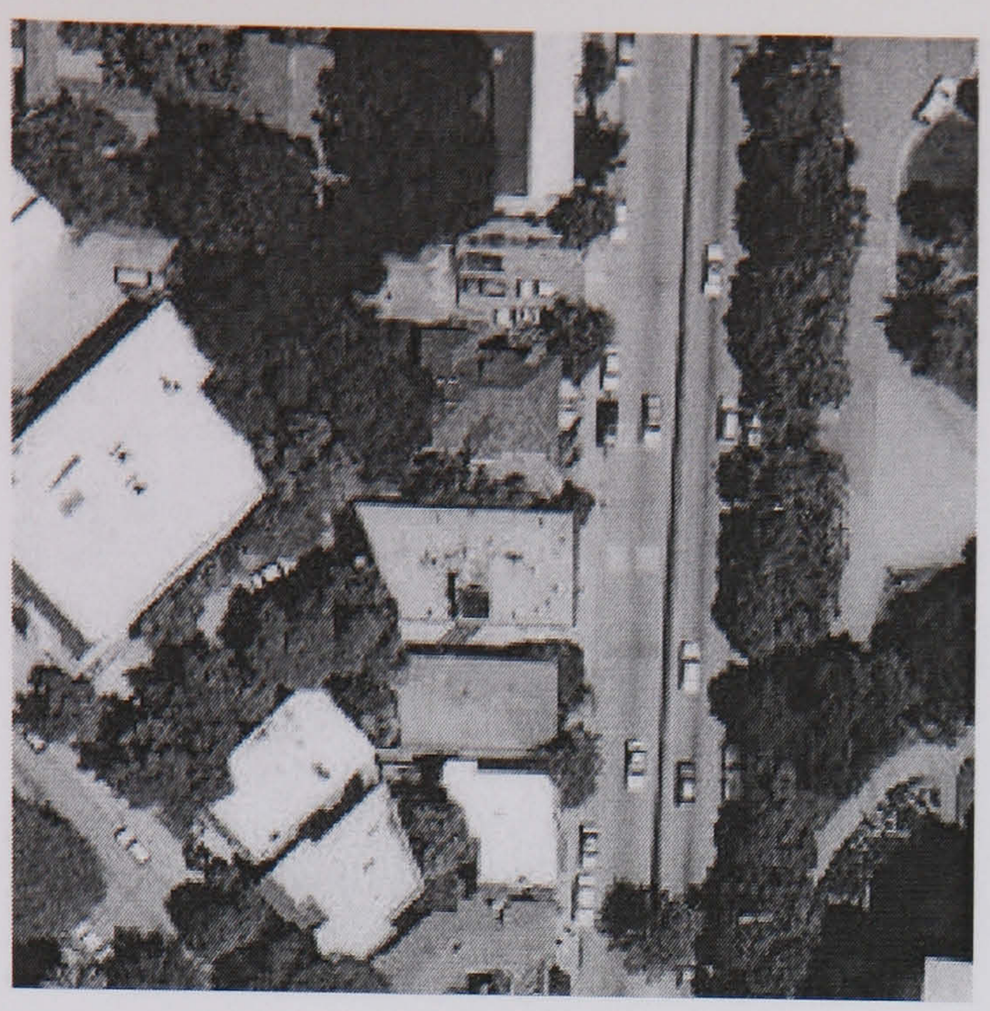


Hybrid Neural Network (90.19 %)

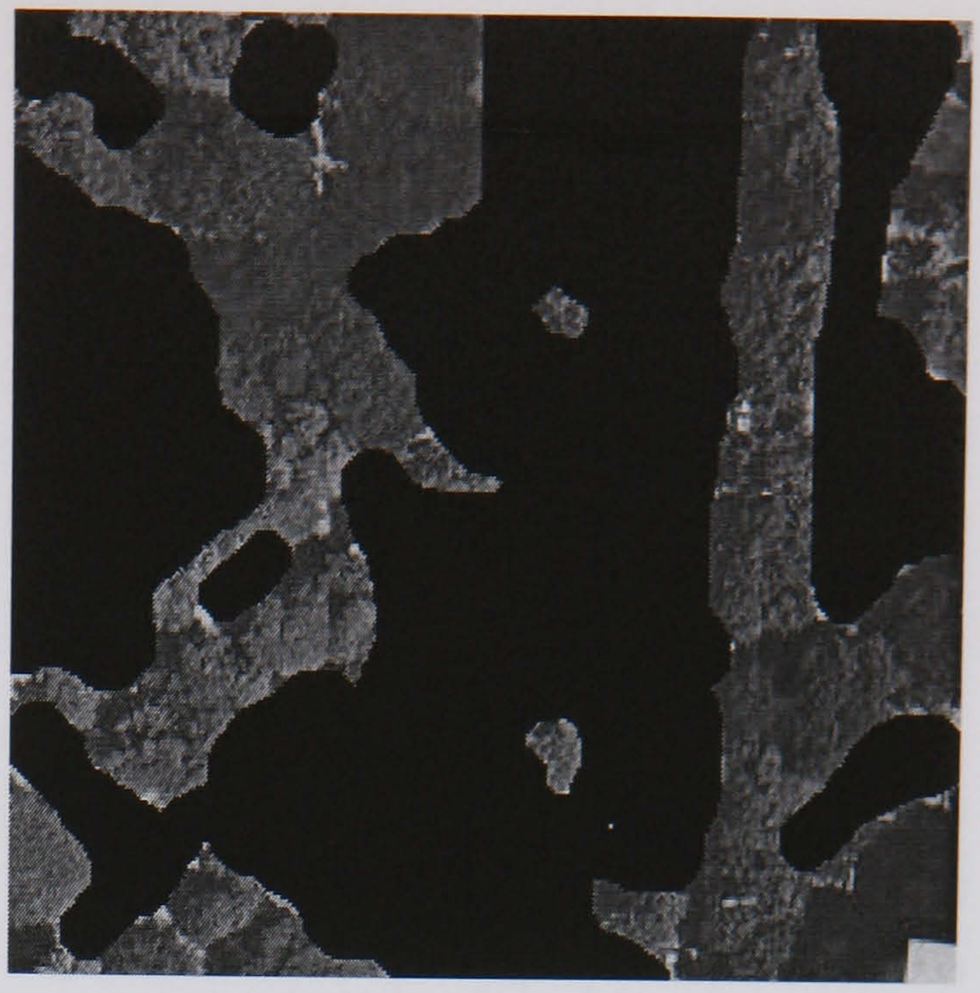


SGLDM /BPNN (84.66 %)

Figure E.53 Aerial Image 55 Results



Aerial Image 56



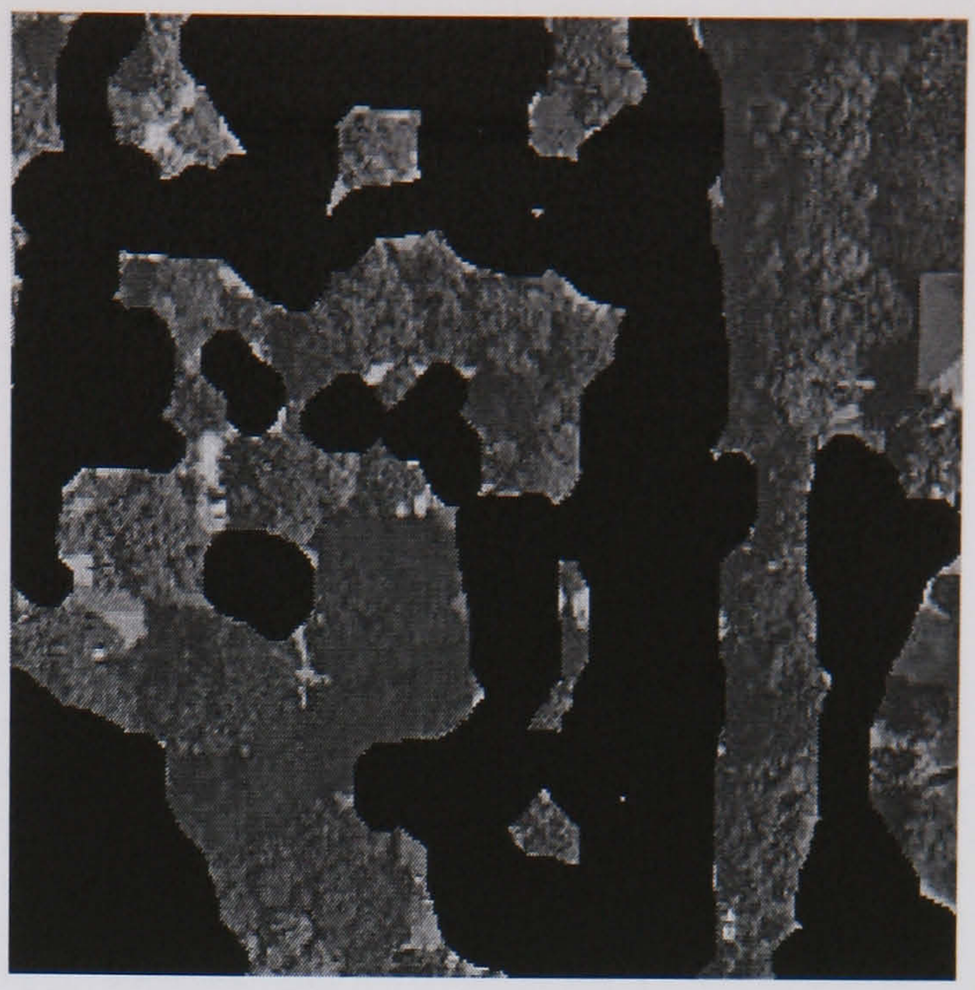
Hand Segmented Image

Hybrid Neural Network (88.66 %)

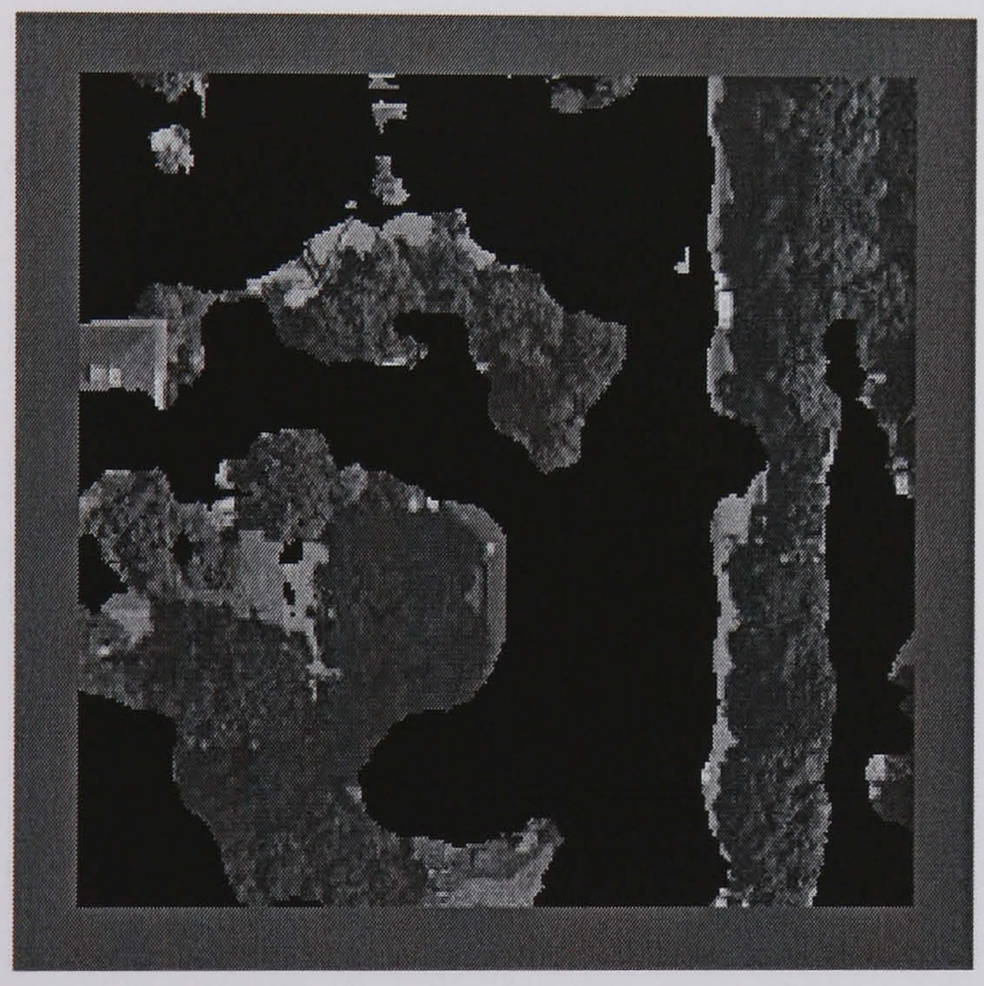
SGLDM /BPNN (72.29 %)

Figure E.54 Aerial Image 56 Results

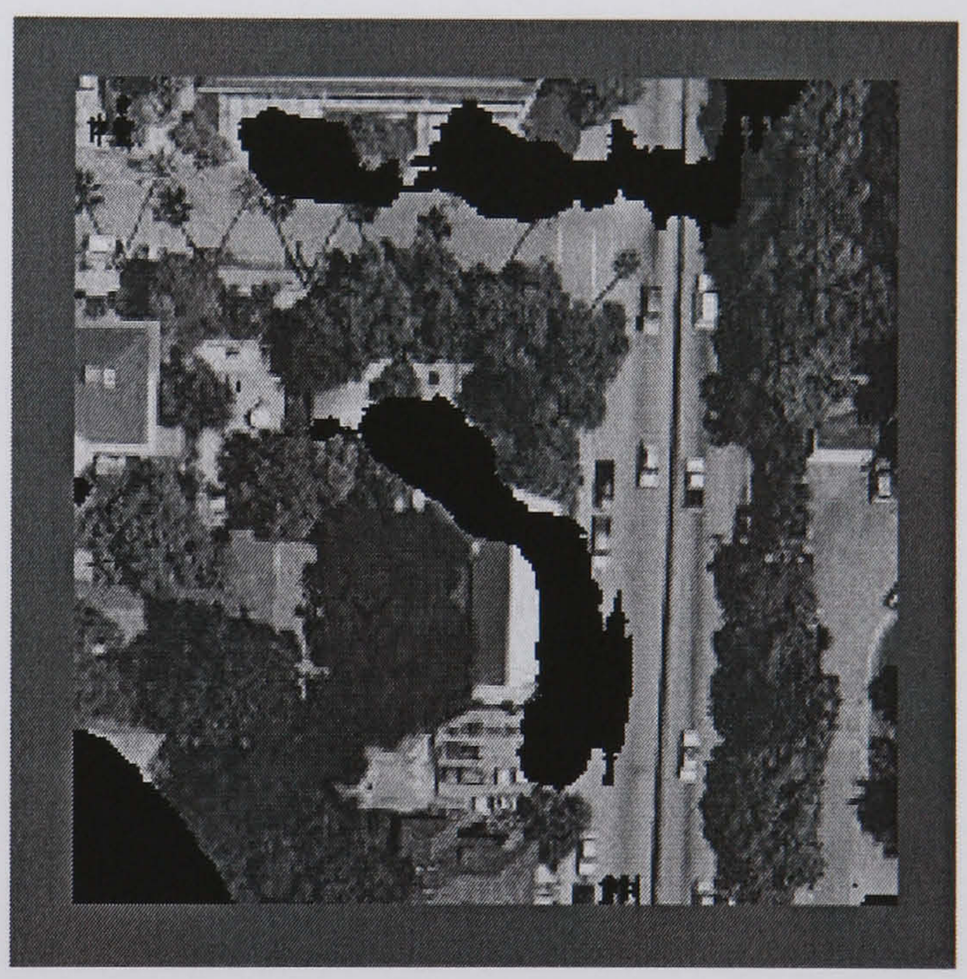
Aerial Image 57



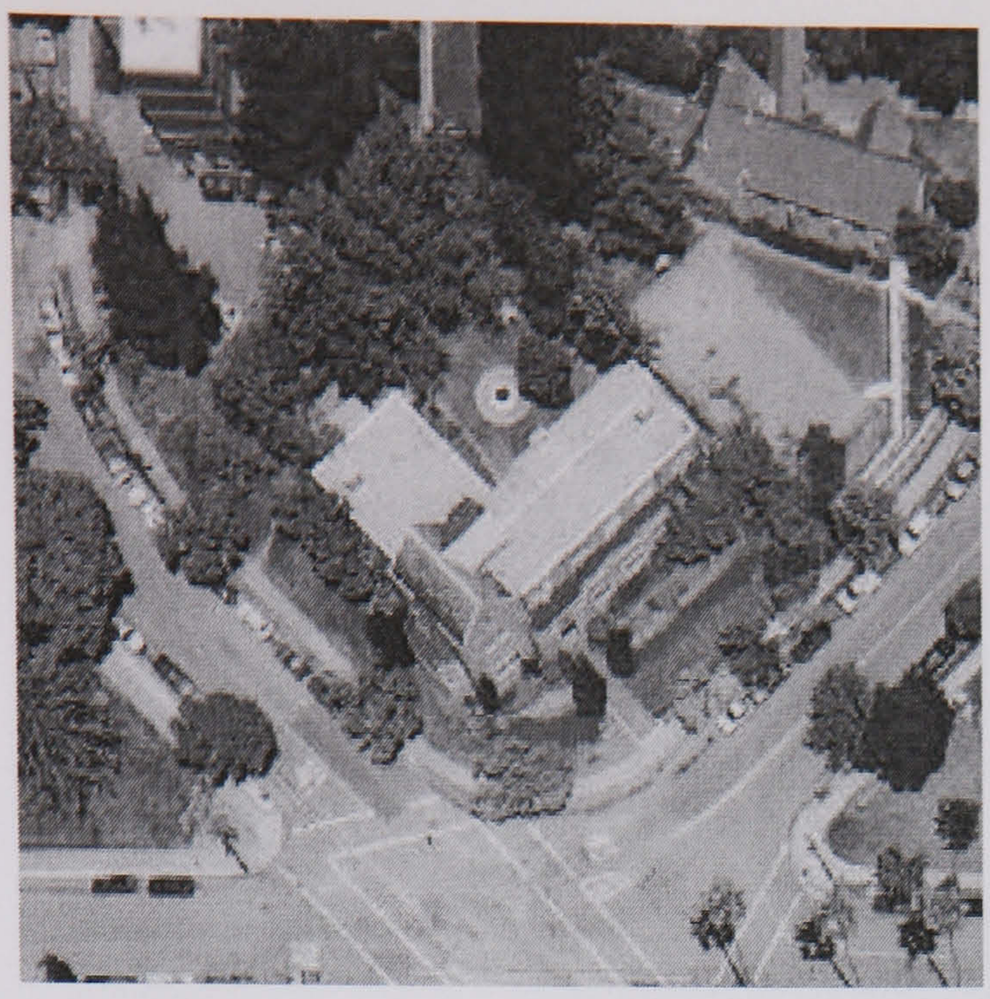
Hand Segmented Image



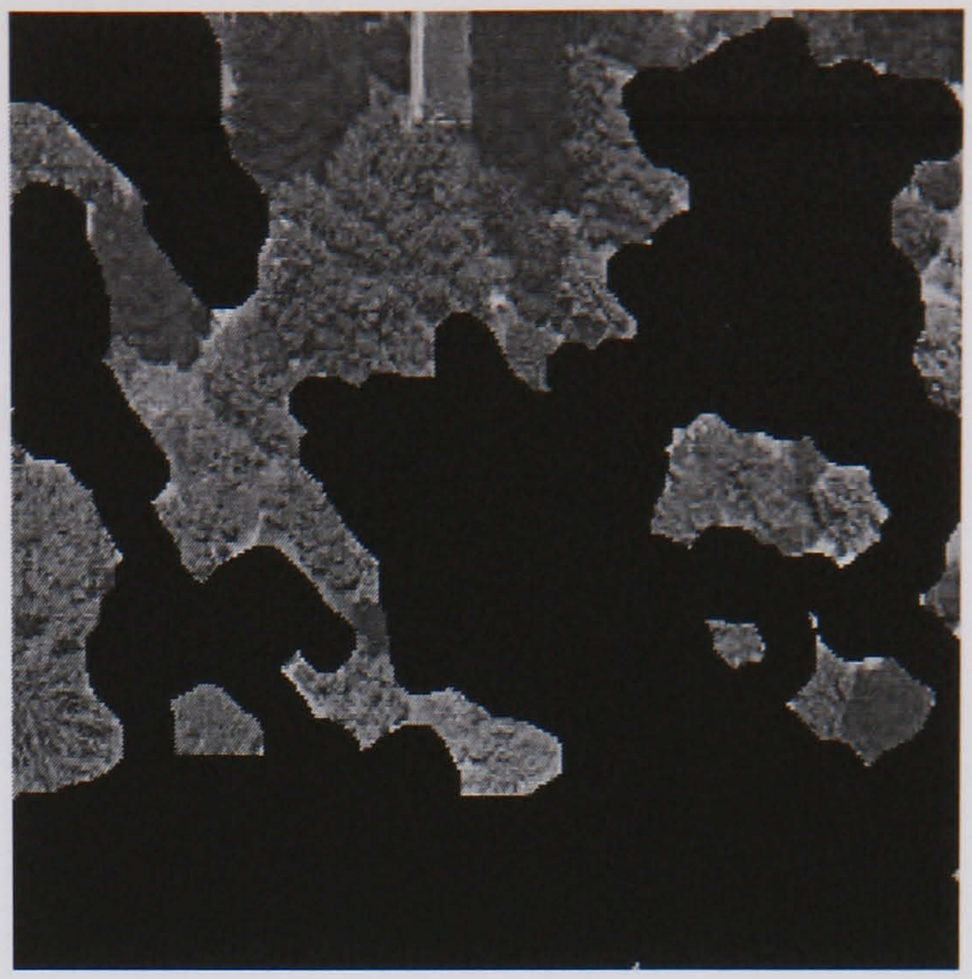
Hybrid Neural Network (85.94 %)



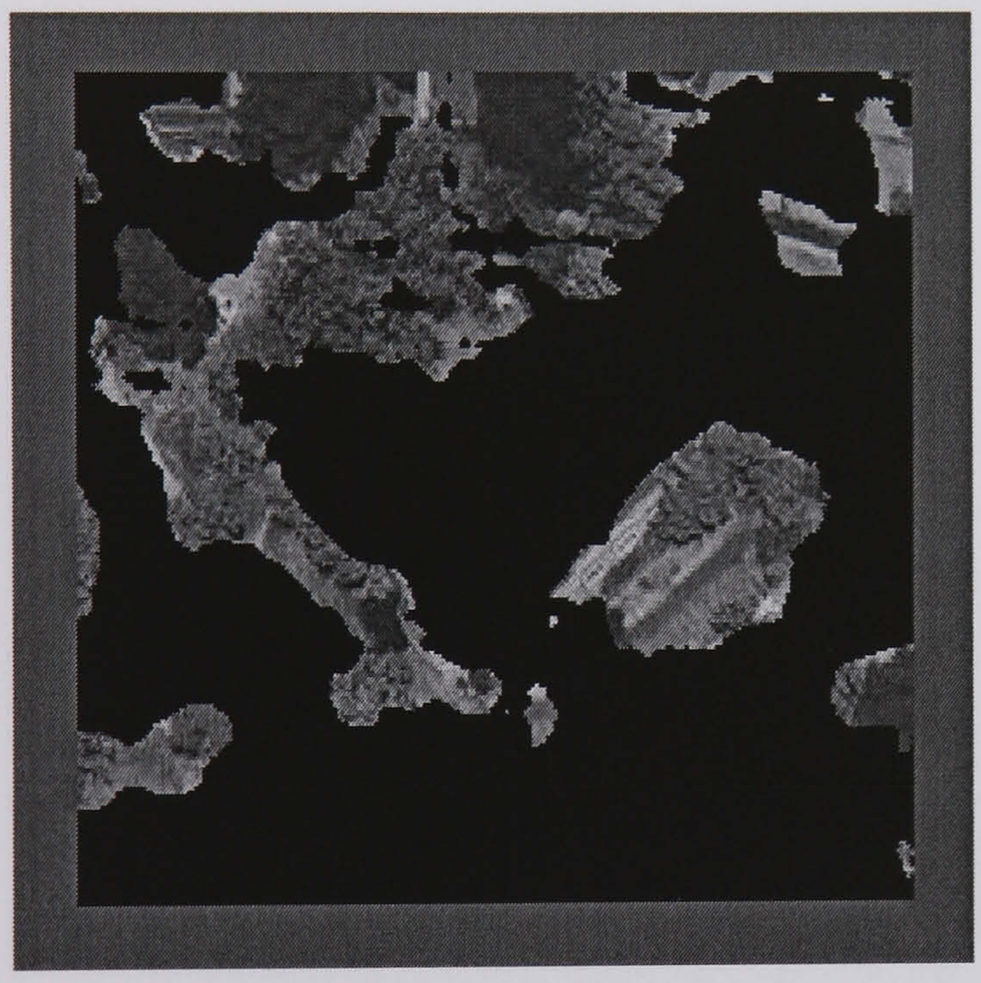
SGLDM /BPNN (68.12 %)



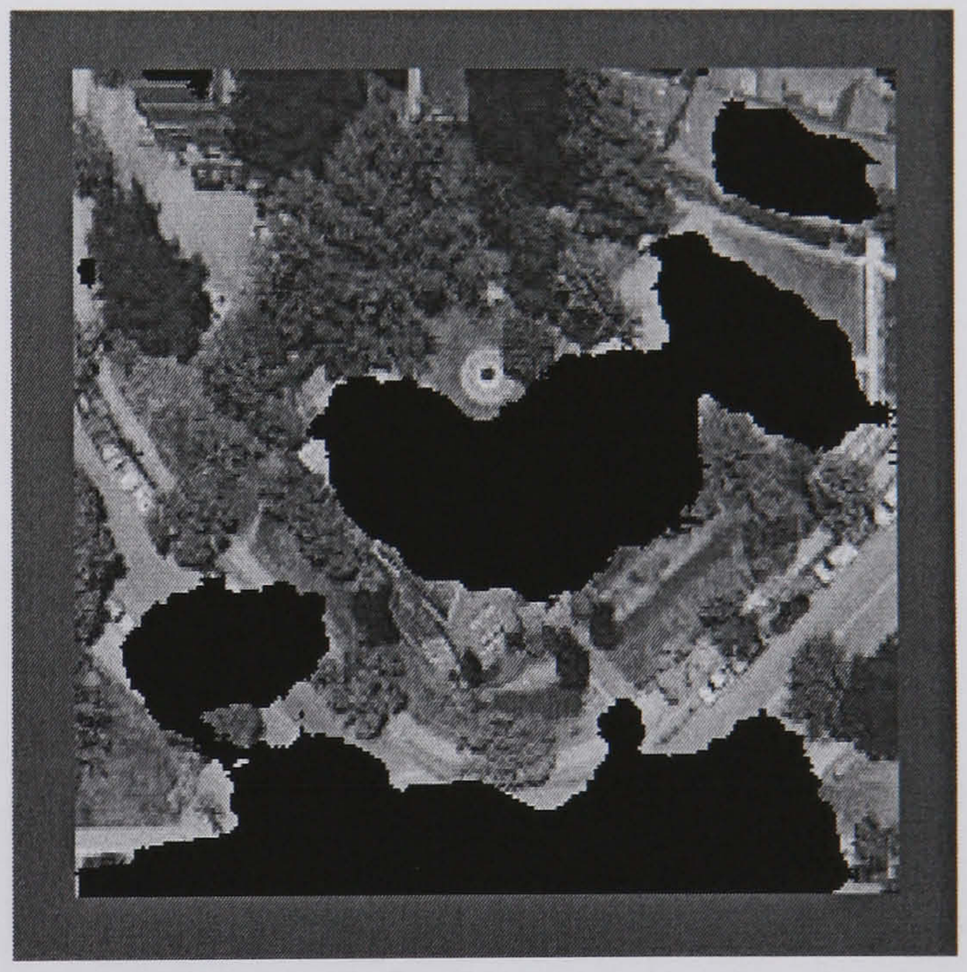
Aerial Image 58



Hand Segmented Image

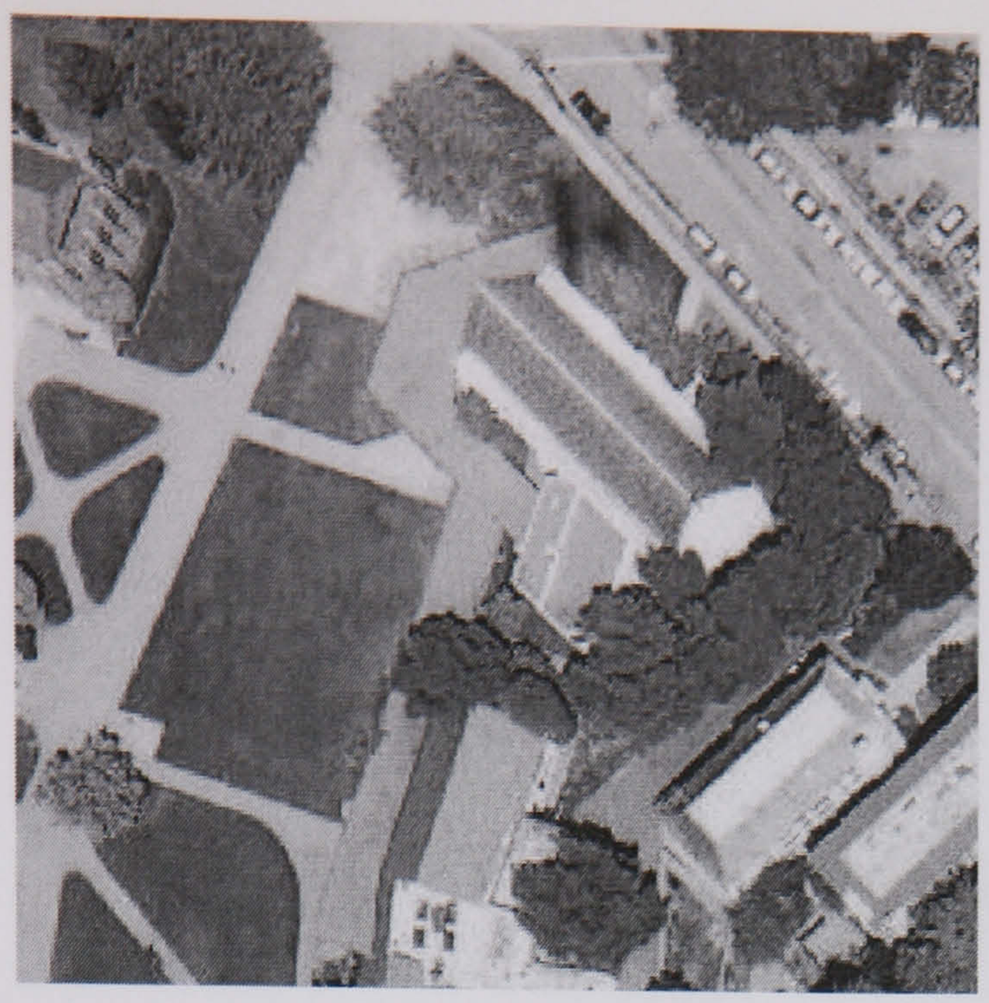


Hybrid Neural Network (85.17 %)

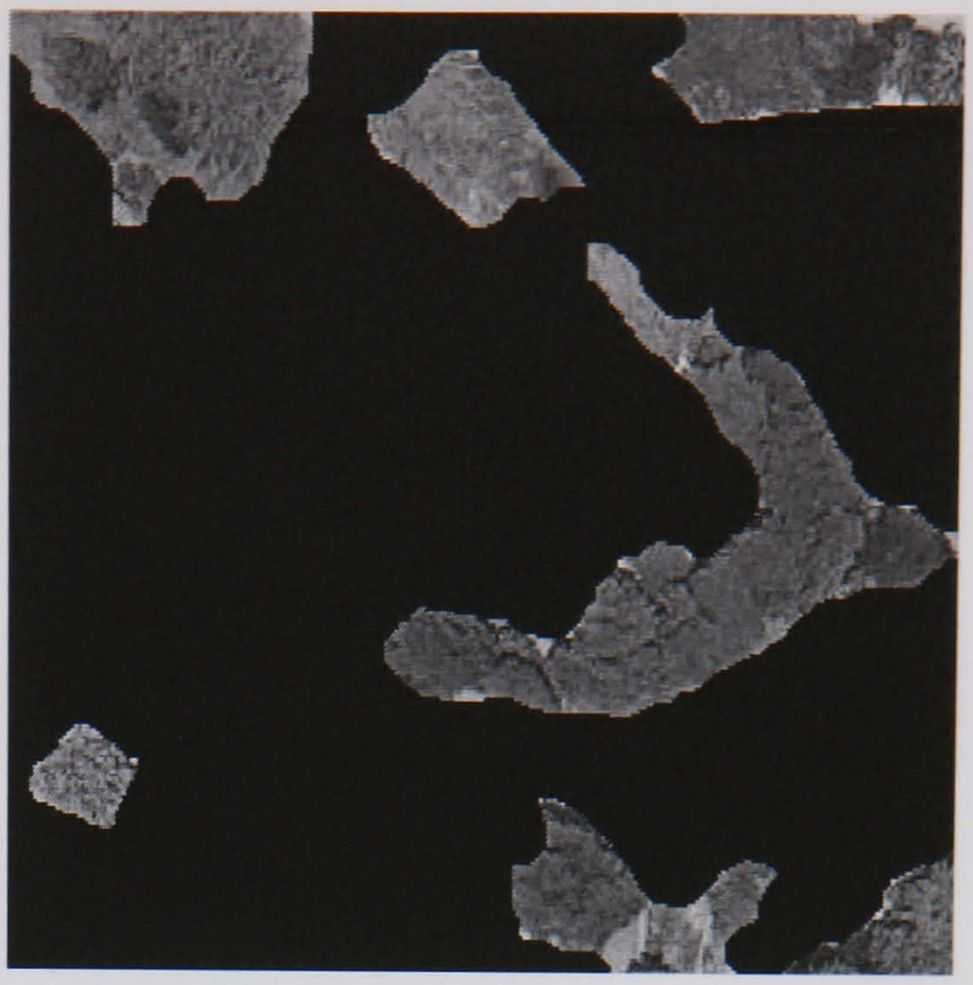


SGLDM /BPNN (72.85 %)

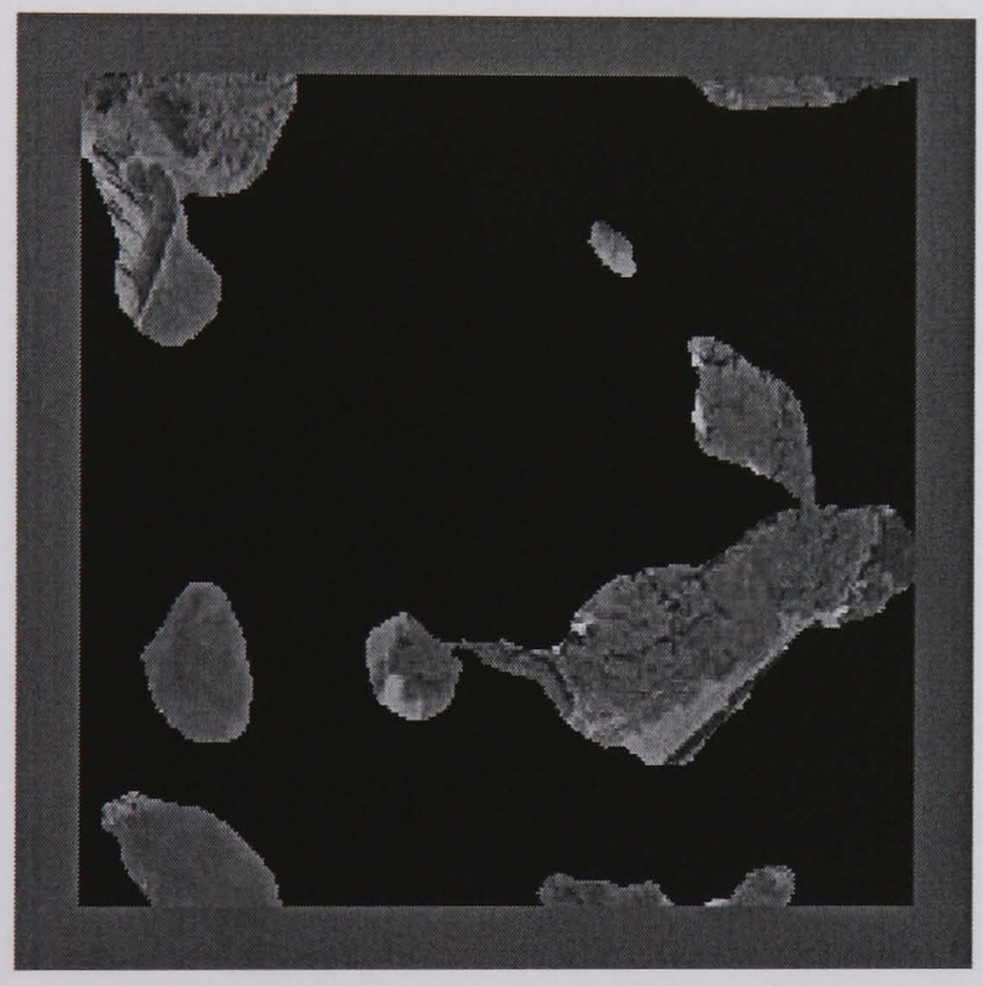
Figure E.55 Aerial Image 58 Results



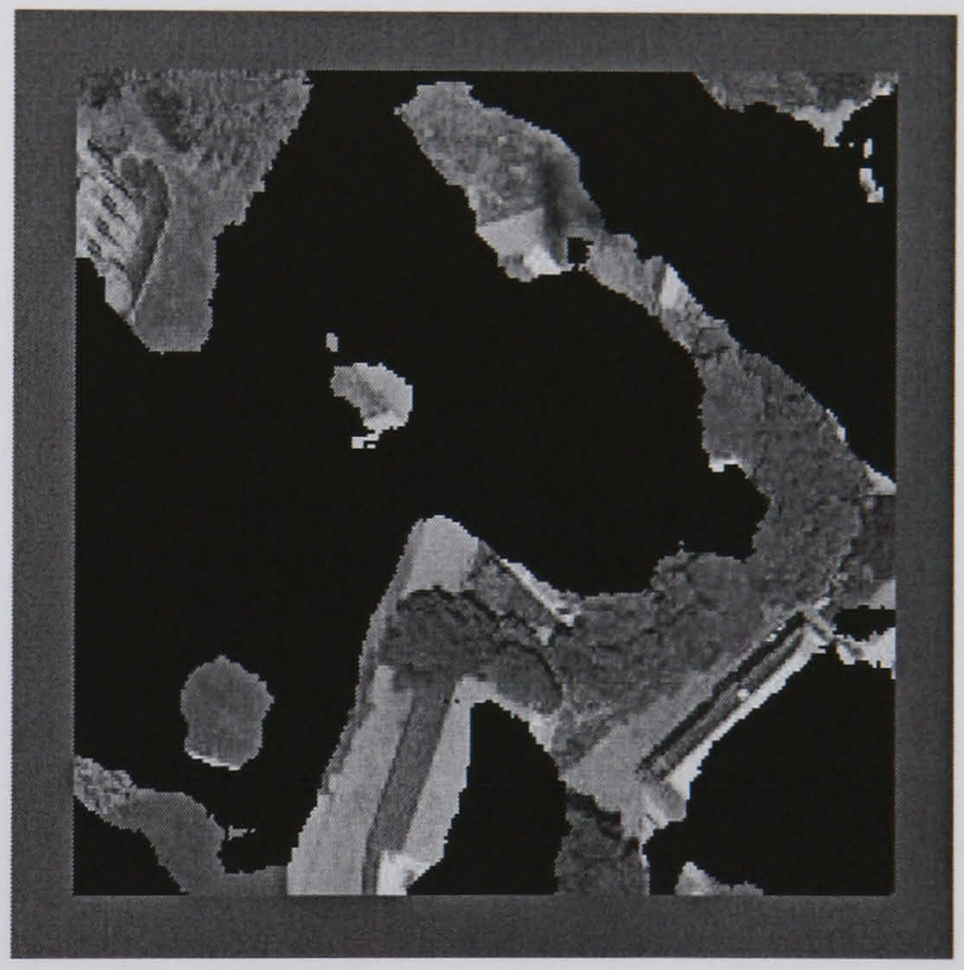
Aerial Image 59



Hand Segmented Image

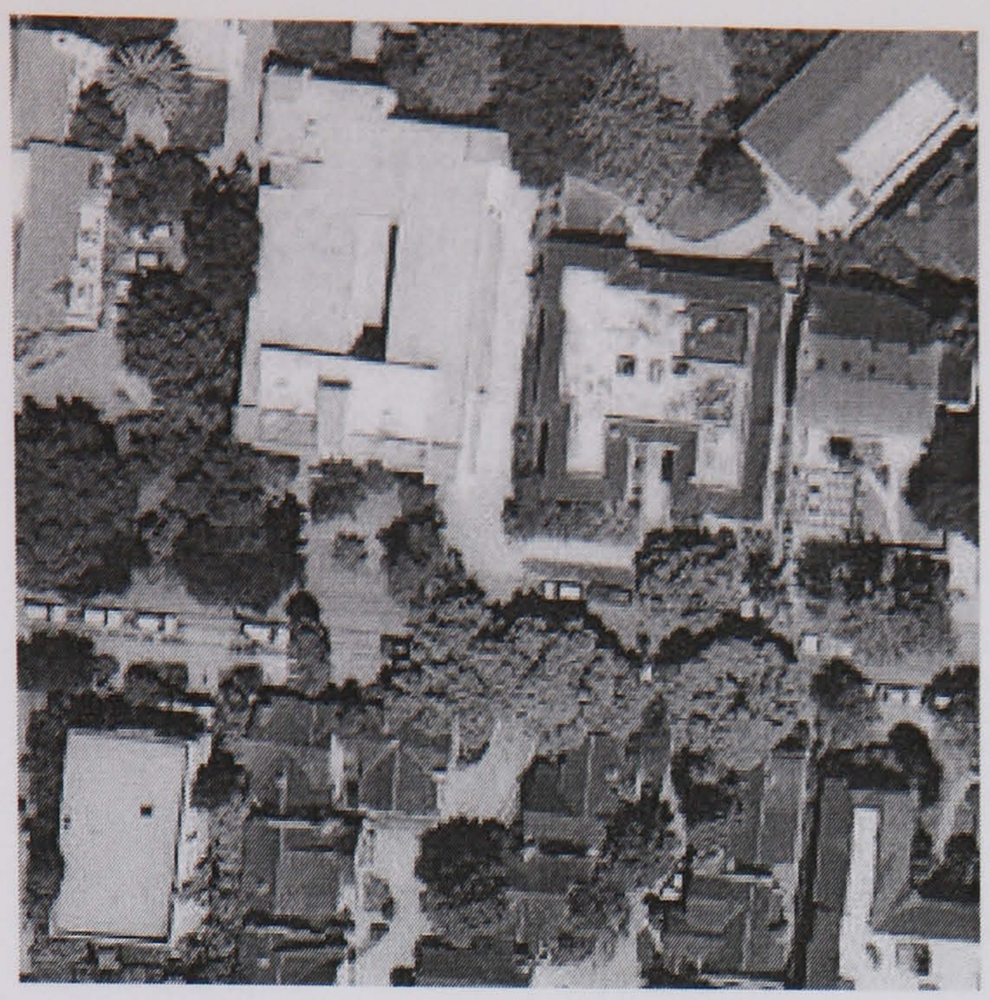


Hybrid Neural Network (86.93 %)

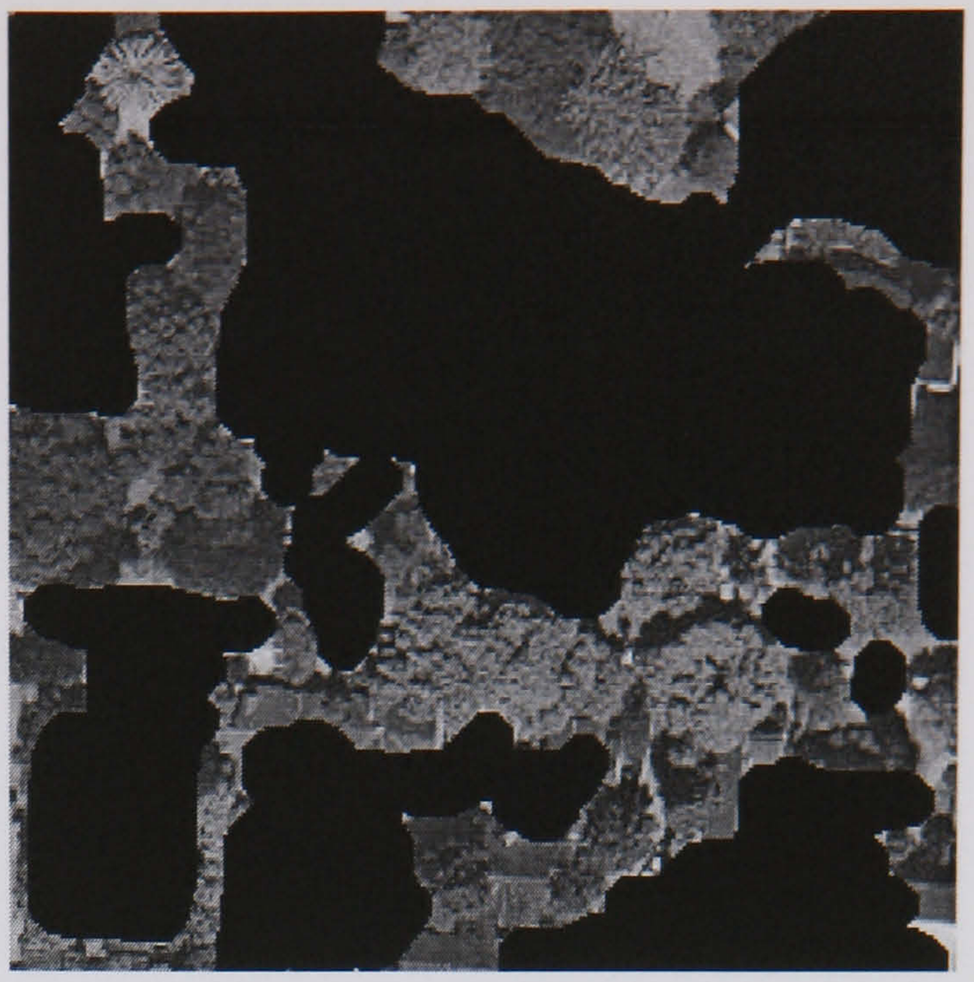


SGLDM /BPNN (82.06 %)

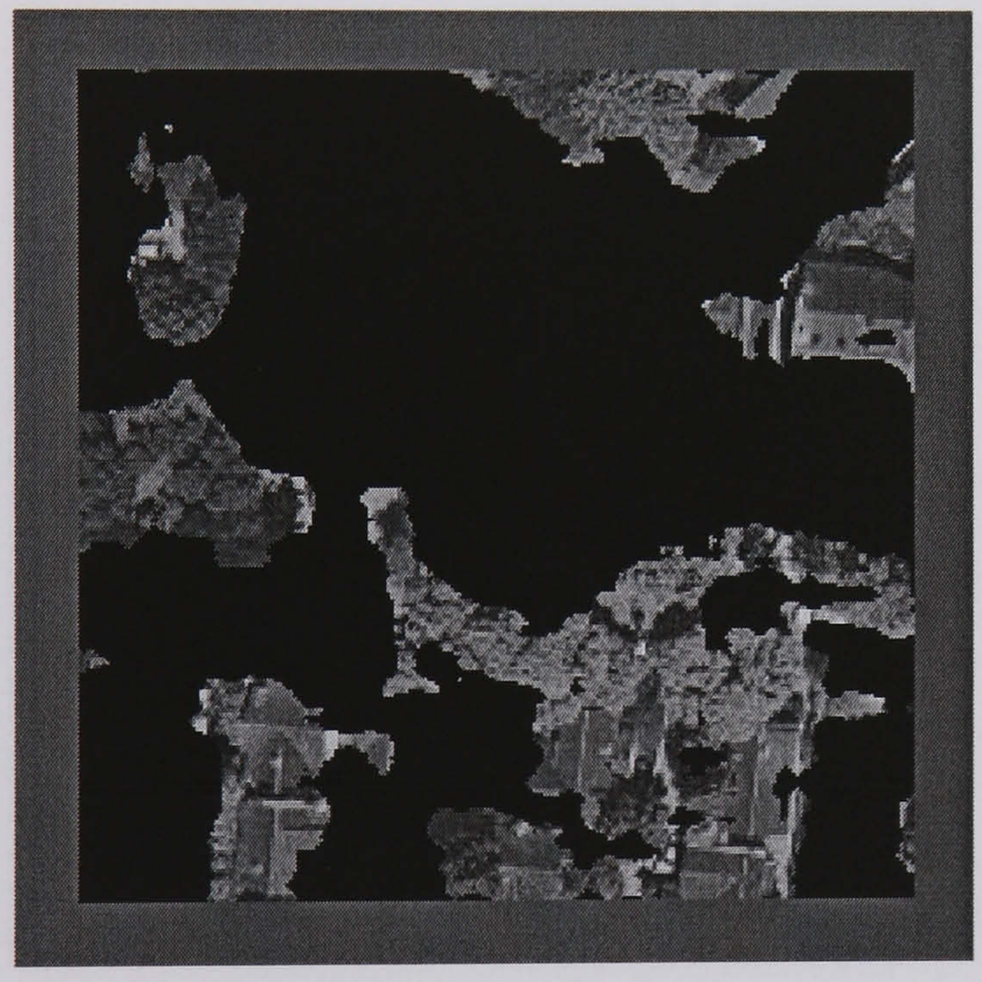
Figure E.56 Aerial Image 59 Results



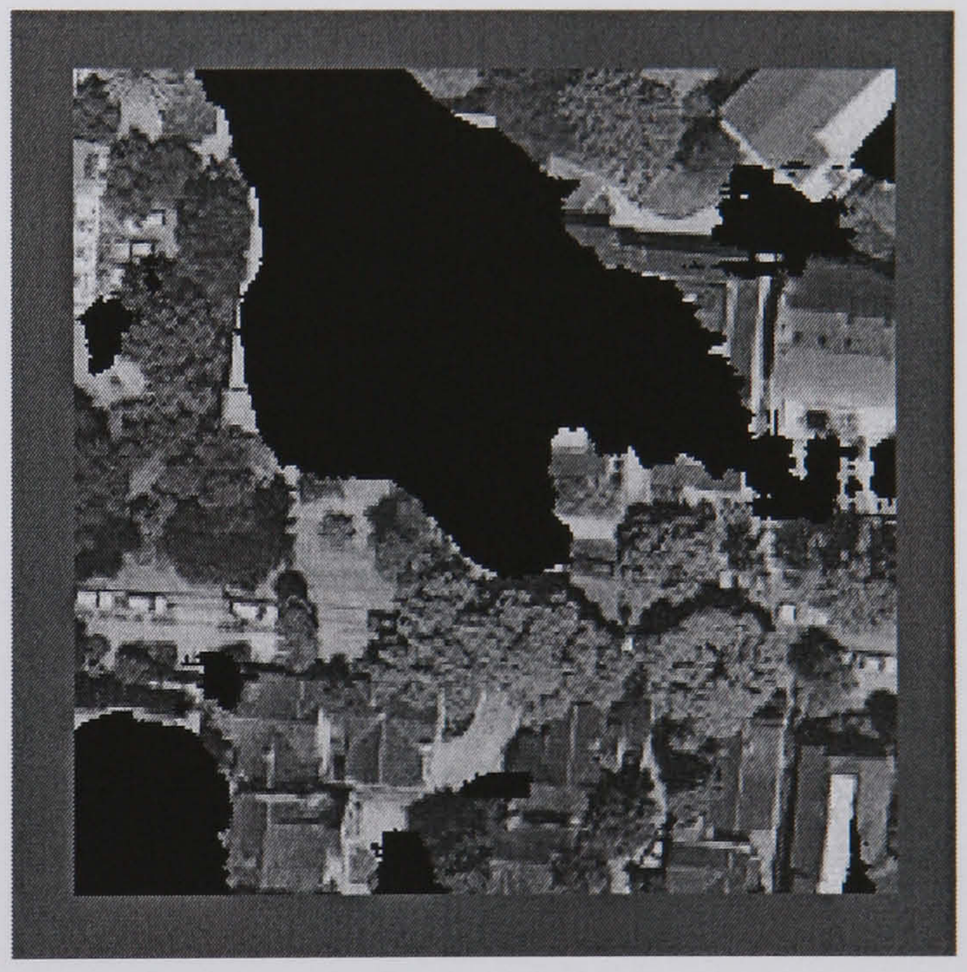
Aerial Image 60



Hand Segmented Image

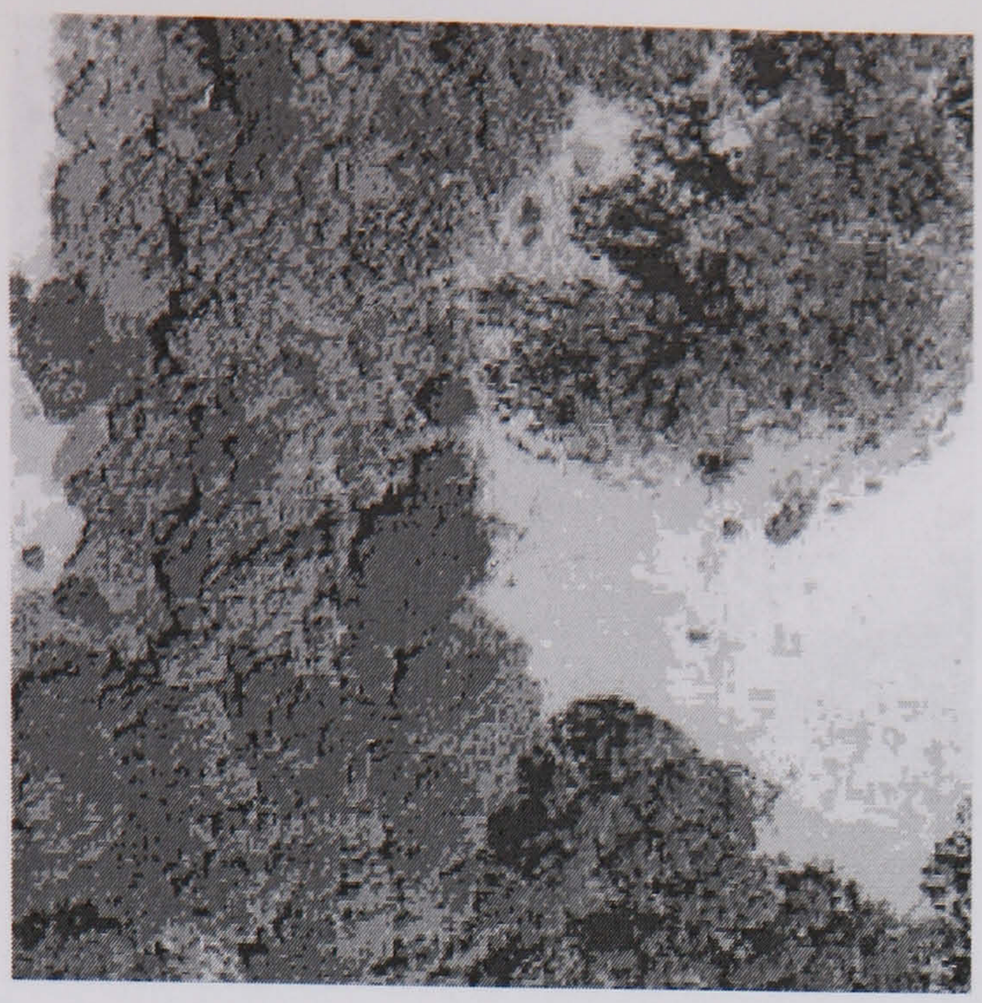


Hybrid Neural Network (83.19 %)

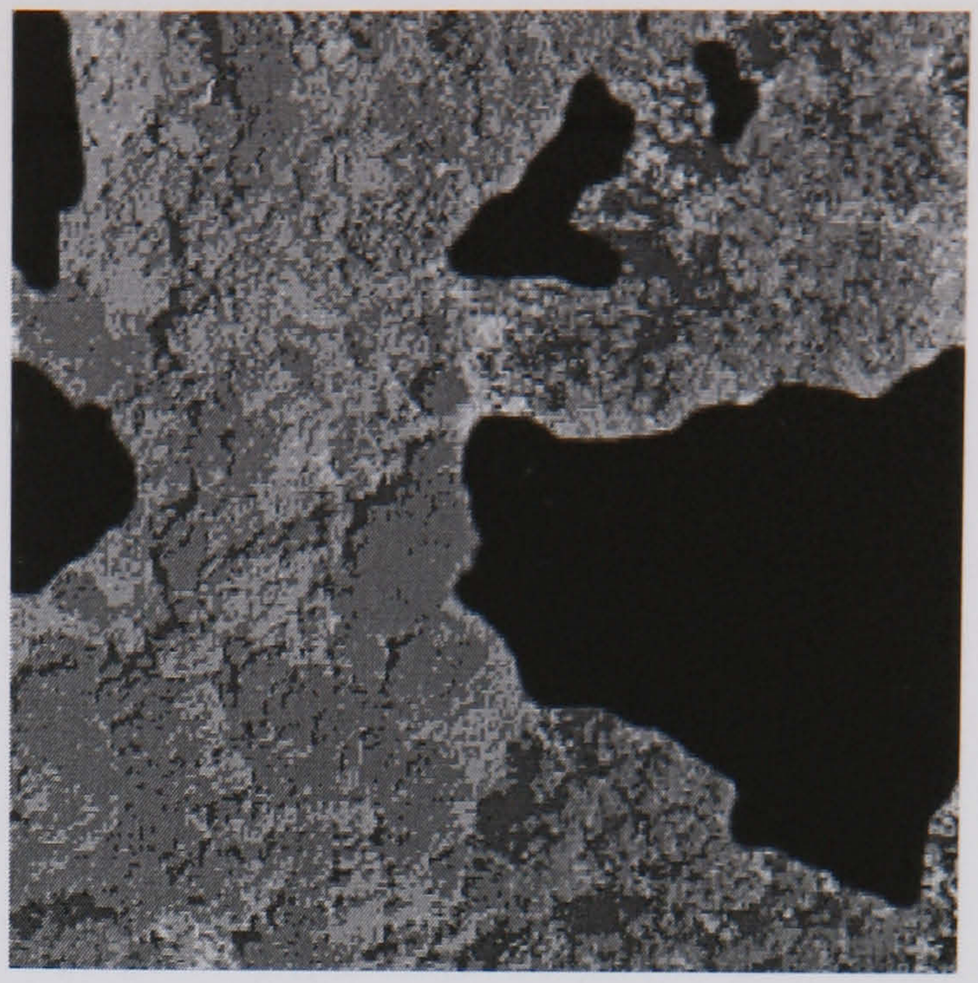


SGLDM /BPNN (74.04 %)

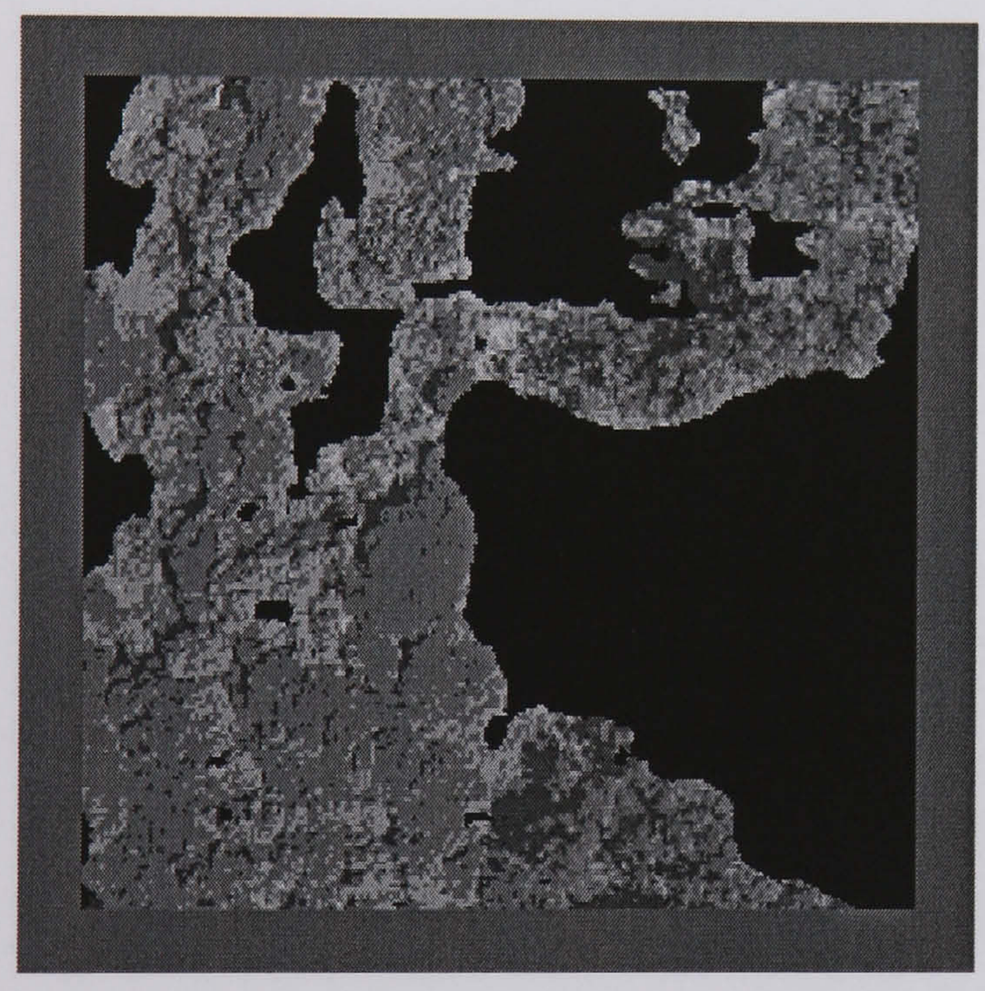
Figure E.57 Aerial Image 60 Results



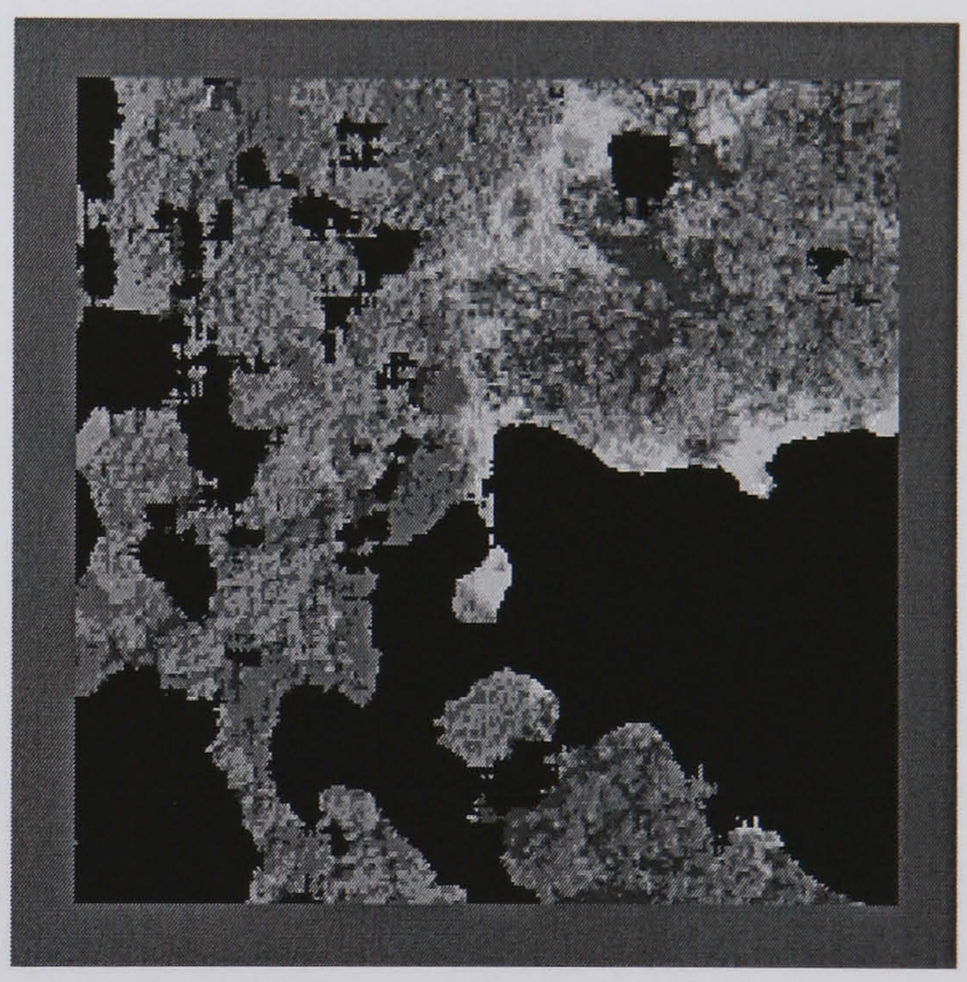
Aerial Image 61



Hand Segmented Image

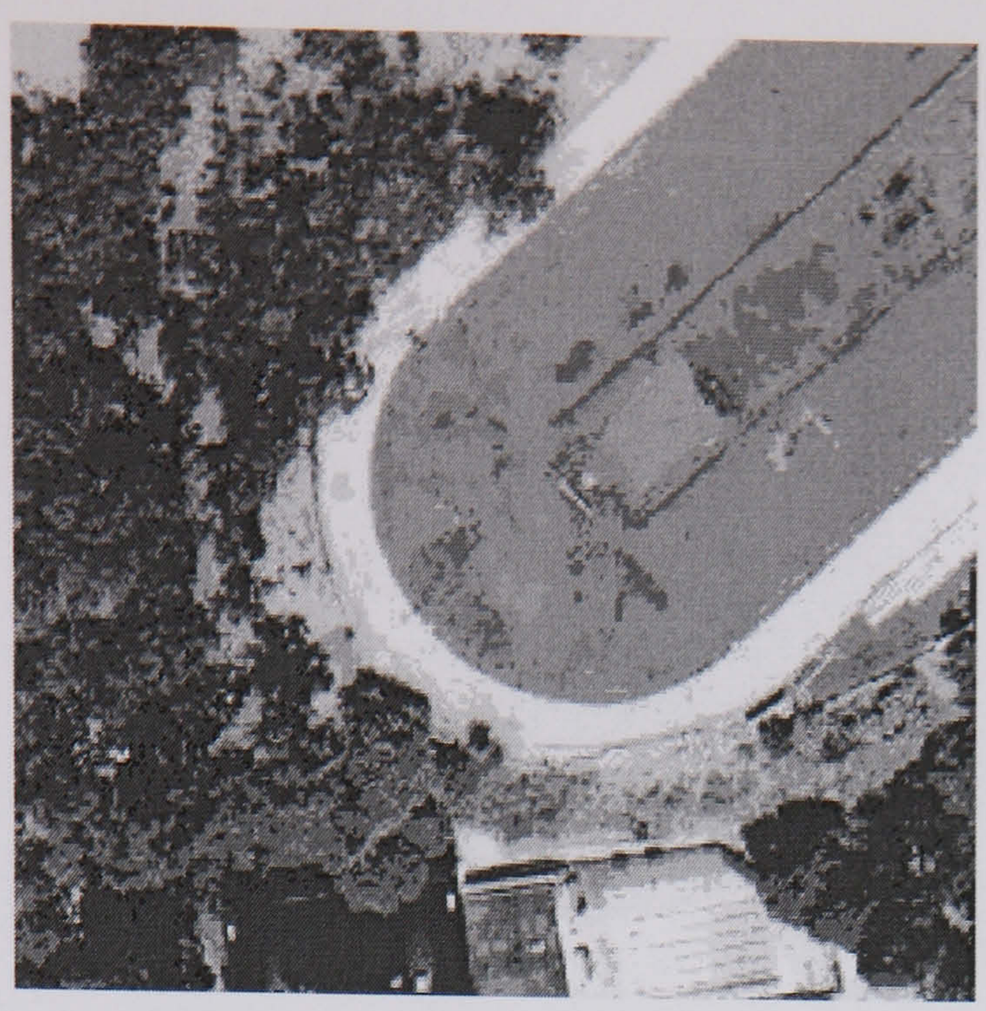


Hybrid Neural Network (87.18 %)

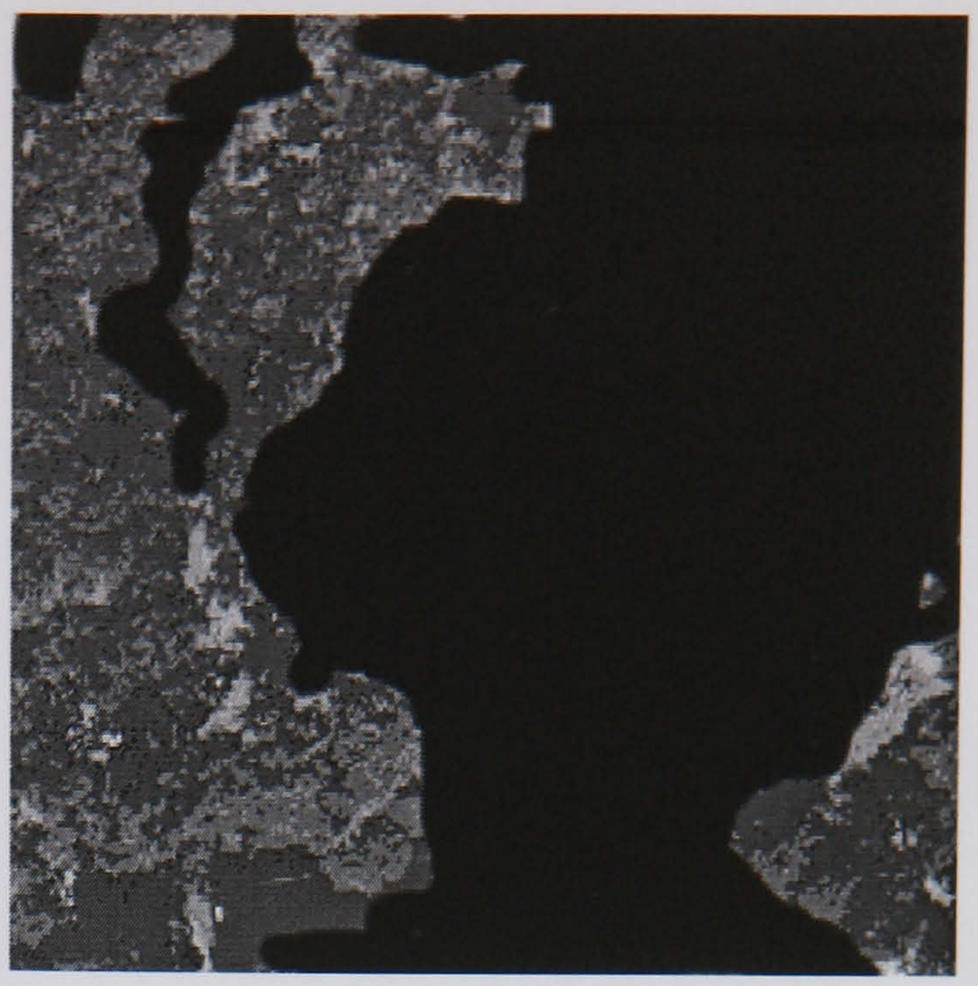


SGLDM /BPNN (75.82 %)

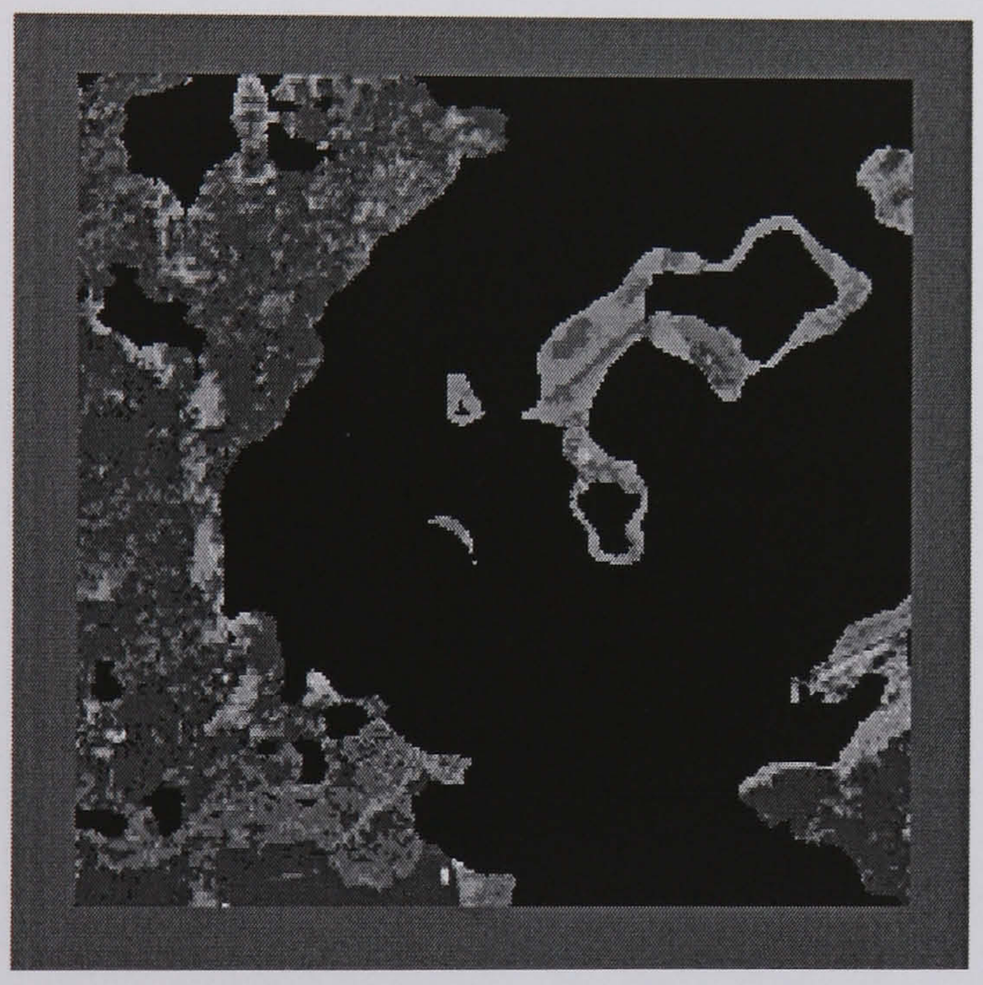
Figure E.58 Aerial Image 61 Results



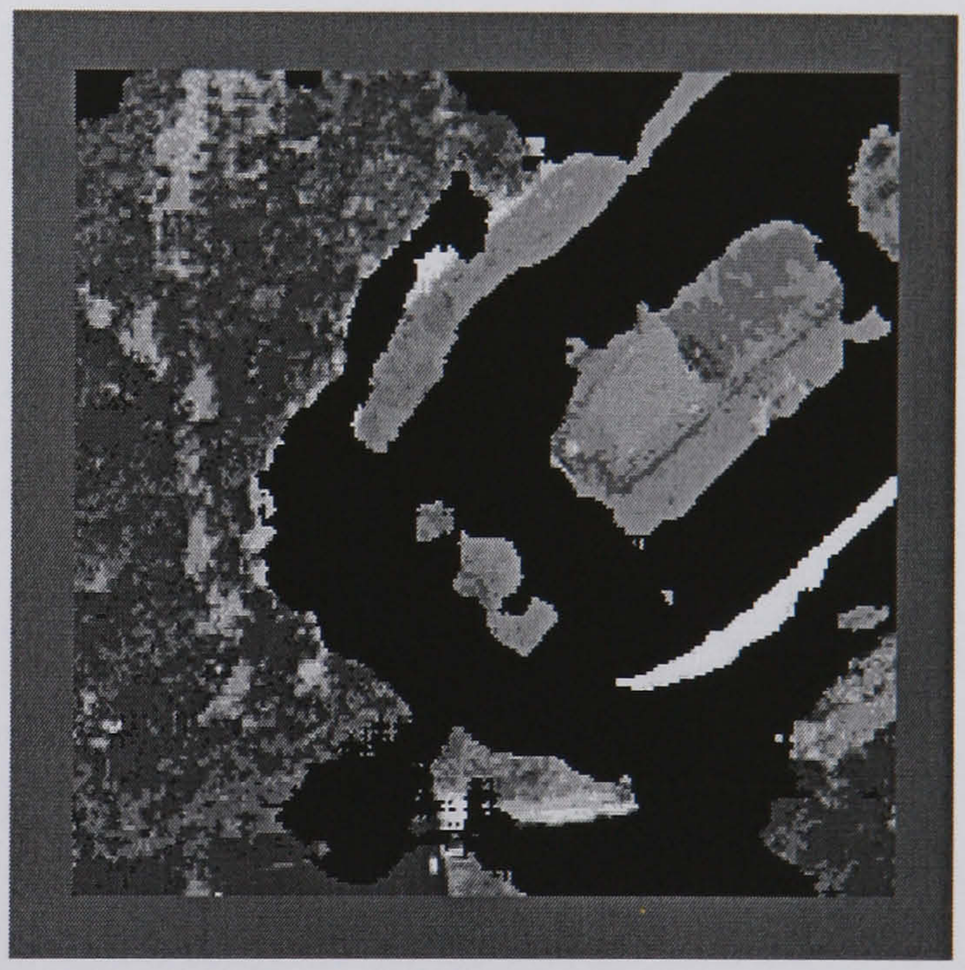
Aerial Image 62



Hand Segmented Image



Hybrid Neural Network (86.80 %)



SGLDM /BPNN (77.66 %)

Figure E.59 Aerial Image 62 Results