

To the Graduate Council:

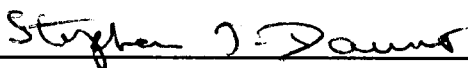
I am submitting herewith a thesis written by Walter F. Smith, Jr. entitled "Stellar Object Image Recovery Using Neural Network Atmospheric Modelling". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Physics.



Dr. William E. Blass, Major Professor

We have read this thesis
and recommend its acceptance:



Clemon Handl


Accepted for the Council:



Associate Vice Chancellor
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature

Walter F. Smith

Date

22 April 1992

**STELLAR OBJECT IMAGE
RECOVERY USING NEURAL NETWORK
ATMOSPHERIC MODELLING**

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Walter F. Smith, Jr.

May 1992

DEDICATION

This thesis is dedicated to my wife, Billie
who has given me invaluable support
in my professional and educational efforts.

ACKNOWLEDGMENTS

I want to extend a special thanks to a number of individuals who have directly or indirectly contributed to this work and without their invaluable assistance I would not have completed this project. First I want to thank Dr. William E. Blass, my major professor, who not only gave me the idea but also provided many of the tools to work with. I also want to thank Dr. Stephen J. Daunt and Dr. Tom Handler for agreeing to be members of my committee.

A special thanks goes to Dr. Gordon Chin who provided the foundations from which to work and who cheerfully answered all my questions no matter how inane. I also want to express my appreciation to Goddard Spaceflight Center for the use of the LEPVAX System on which I compiled and executed the IDL programs which were the foundation of my research. My fellow graduate students Steve Mahan and Forrest Hoffman also were of invaluable technical assistance since my computer skills are somewhat limited.

ABSTRACT

This paper describes the use of a back-propagation neural network to model a turbulent atmosphere. The modeling of the atmosphere allows the near real time resolution of a stellar object's image from its respective speckle images. These speckle images are produced when the light from a stellar object passes through a diffusing medium such as our atmosphere. If this modelling is eventually successful it will provide the astronomer the capability of near real time stellar object identification at low cost and with fairly simple technology.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
Speckle	4
Neural Networks	12
Desired Objective	18
2. ARTIFICIAL NEURAL SYSTEM	20
Back-propagation Algorithm	20
NETS Neural Network Program	26
VAX 9000	27
3. ARTIFICIAL SPECKLE IMAGE GENERATION	29
Atmosphere Simulation	30
Speckle Simulation	30
4. DATA PREPROCESSING	36
Training Set Input Preparation	36
Training Set Output Preparation	42
Control Set Preparation	43
5. TRAINING THE NET	44
Network Architectures	45
Neural Network Atmospheric Model	46
6. EXPERIMENTAL RECOVERY SIMULATIONS	50
Test One - Set-Up and Results	50
Test Two - Set-Up and Results	56
Test Three - Set-Up and Results	73

Test Four - Set-Up and Results	94
7. CONCLUSIONS	107
Future Work	108
REFERENCES	110
APPENDICES	114
VITA	127

LIST OF FIGURES

Figure		Page
1.1	Diagram of Different Speckle Producing Methods	5
1.2	Surface Intensity Plot of a Simulated Binary Star	7
1.3	Summation of FFT's of Speckle Images Created by Binary Star	9
1.4	Spatial Auto-Correlation of Image Intensity with DC Component Blocked	10
1.5	General Neural Network Configuration	14
1.6	The Exclusive OR Function	14
1.7	Generalized Processing Element Construction	16
3.1	Surface Plot of a Simulated Turbulent Atmosphere Array	31
3.2	Surface Plot of a Simulated Speckle Image	33
3.3	Contour Plot of a Simulated Speckle Image	34
3.4	Four Consecutive Speckle Patterns	35
4.1	FFT of a Single Speckle Image	38
4.2	FFT Peak Center Shifted	39
4.3	Modulus of Single Speckle Image FFT	40
4.4	Surface Plot of Phase Values for Center Shifted FFT	41
5.1	Single Hidden Layer Network with 30 Neurons	47
5.2	Double Hidden Layer Network with 15 Neurons in Each Hidden Layer	48
5.3	Triple Hidden Layer Network	49

6.1	Test One Single Star Simulations	51
6.2	Test One Training RMS versus Control RMS	53
6.3	Test One Training Object Image (6,6) and Recovery	54
6.4	Test One Random Placement of Single Stars	55
6.5	Binary Training Star Simulation	57
6.6	Binary Stars Training RMS versus Control RMS	58
6.7	Test Two Single Star Simulations	59
6.8	Test Two Single Hidden Layer Training RMS versus Control RMS	60
6.9	Test Two Training Object Image (6,6) and Recovery	62
6.10	Test Two Training Object Image (9,6) and Recovery	63
6.11	Test Two Training Object Image (6,9) and Recovery	64
6.12	Test Two Training Object Image (9,9) and Recovery	65
6.13	Test Two Double Hidden Layer Training RMS versus Control RMS	66
6.14	Test Two Triple Hidden Layer Training RMS versus Control RMS	67
6.15	Test Two Binary Test Object [(6,6), (9,6)] and Recovery	68
6.16	Test Two Binary Test Object [(6,6), (9,9)] and Recovery	69
6.17	Surface Plot of Decreased Turbulent Atmosphere Array	70
6.18	Test Two Training Object (6,6) and Recovery with Speckle From Less Turbulent Atmosphere	71

6.19	Test Two Training Object (9,6) and Recovery with Speckle from Less Turbulent Atmosphere	72
6.20	Test Two Random Placement of Single Star Simulations	74
6.21	Test Two Random Placement Double Hidden Layer Configuration Training RMS versus Control RMS	75
6.22	Test Two Random Placement Single Hidden Layer Configuration Training RMS versus Control RMS	76
6.23	Test Two Random Placement Training Object Image (5,7) and Recovery	77
6.24	Test Two Random Placement Training Object Image (9,5) and Recovery	78
6.25	Test Two Random Placement Training Object Image (7,8) and Recovery	79
6.26	Test Two Random Placement Training Object Image (6,10) and Recovery	80
6.27	Test Two Random Placement Trained Network with Test Object Image (9,8) and Recovery	81
6.28	Test Two Random Placement Trained Network with Test Object Image [Binary (6,7), (6,9)] and Recovery	82
6.29	Test Three Single and Binary Star Simulation	83
6.30	Test Three Double Hidden Layer Training RMS versus Control RMS	85
6.31	Test Three Training Object Image [Binary (5,5), (10,10)] and Recovery	86
6.32	Test Three Training Object Image (9,6) and Recovery	87
6.33	Test Three Test 1 Object Image [Binary (6,7), (11,12)] and Recovery	88
6.34	Test Three Test 2 Object Image (7,6) and Recovery	89

6.35	Test Three Training Object Image [Binary (5,5), (10,10)] and Recovery with Speckle from Less Turbulent Atmosphere	90
6.36	Test Three Training Object Image (9,6) and Recovery with Speckle from Less Turbulent Atmosphere	91
6.37	Test Three Test 1 Object Image [Binary (6,7), (11,12)] and Recovery with Speckle from Less Turbulent Atmosphere	92
6.38	Test Three Test 2 Object Image (7,6) and Recovery with Speckle from Less Turbulent Atmosphere	93
6.39	Test Four Single, Binary, and Triple Star Simulation	95
6.40	Test Four Double Hidden Layer Training RMS versus Control RMS	96
6.41	Test Four Training Object Image [Binary (6,8), (10,4)] and Recovery	97
6.42	Test Four Training Object Image [Triple (6,5), (7,8), (9,6)]	98
6.43	Test Four Training Object Image (5,5) and Recovery	99
6.44	Test Four Test 1 Object Image (10,9) and Recovery	100
6.45	Test Four Test 2 Object Image [Triple (8,8), (7,11), (5,9)] and Recovery	101
6.46	Test Four Training Object Image (5,5) and Recovery with Speckle from a More Turbulent Atmosphere	103
6.47	Test Four Training Object Image [Binary (6,8), (10,4)] and Recovery with a More Turbulent Atmosphere	104
6.48	Test Four Training Object Image [Triple (6,5), (7,8), (9,6)] and Recovery	105
6.49	Surface Plot of Increased Turbulent Atmosphere Array	106

CHAPTER 1

INTRODUCTION

The star light that eventually reaches us on earth must travel distances that are enormous. To an observer on earth, this light can be approximated as a coherent point source since by the time it arrives in our solar system, the wavefront of the light is approximately a plane wave. For the earth-bound observer, the real problems in imaging the starlight occur during the last 10-15 kilometers of its travel through the universe to earth. It is in this 10-15 kilometer band that the light encounters the earth's atmosphere. This encounter with the atmosphere is the source of most of the imaging problems found in astronomy.¹

Our atmosphere acts as a constantly changing, diffusing medium for approximately coherent sources (stars). One can see this effect occurring naturally if you have ever observed the bottom of a swimming pool on a sunny day. The water in the pool acts as a turbulent diffusing medium if the water surface is disturbed. You will observe bright varying patterns (speckle) which move across the bottom of the pool. This effect is also observed on a clear night when stars are clearly visible but appear to 'twinkle'.

This twinkling effect is particularly troublesome to astronomers using large telescopes. Theoretically, the resolution of a telescope should increase proportionally to an

increase in diameter of the lens of the telescope. Large telescopes should be able to resolve fine spatial details that would distinguish features on the surface of a planet or separate two close stars. Unfortunately, this doesn't happen due to the turbulence of the earth's atmosphere. The patch of illumination produced by a telescope is theoretically governed by the equation

$$\theta = .00244 \cdot \left(\frac{\lambda}{d} \right) \quad (1.1)$$

[Airy (1871)] where λ is the wave length and d is the diameter of the telescope. As the diameter increases the patch of illumination should decrease thereby increasing the focal resolution. In fact, little improvement is seen when the aperture exceeds 100 mm. The diffraction limited resolution of a 5m telescope should be about .02 arcseconds (seconds of arc is a standard astronomical measuring convention using subtended angle measurements). However, the images of stars using this size telescope have a resolution of approximately one arcsecond. Our largest telescopes (Palomar) should have a resolution of 1/50 of an arcsecond according to the Airy equation, but astronomers are unable to resolve detail finer than about one arcsecond. In fact, the degree of image resolution is highly variable, and increases or decreases according to the particular turbulence in the atmosphere that is present while viewing. The variability in resolution (degree of image degradation) is referred to as "seeing".

Seeing is defined as the broadening effect on stellar images caused by the atmospheric turbulence. As the turbulence increases so does the "seeing". Seeing is measured in arcseconds thus as the "seeing" increases, the resolution decreases. Seeing can increase as high as 20 arcseconds under certain conditions. This effectively stops observations by astronomers under these conditions.²

Since large telescopes are unable to resolve these images finer than one arcsecond, the question remained until late in this century; what does the image of a star look like when viewed through a very large telescope? The answer to this question was provided by Antoine Labeyrie in 1969. He suggested that the random speckle images which large telescopes produce were analogous to the speckle image produced by a laser when directed through a diffusing medium. He determined that the optical interference effects observed in the laser light passing through a diffusing medium were replicated in the star light passing through the earth's atmosphere. When laser light is passed through a lens, and projected on a screen, a classical Airy pattern is projected on the screen. If a diffusing medium such as ground glass is placed in the path of a laser, and lens, then an Airy pattern covered with speckle will result. The pattern however is stationary and does not fluctuate since the diffusing medium is constant. If the diffusing medium of the laser was in a constant state of flux much like our atmosphere, then speckle

images much like we find on the large telescopes would be projected on the screen of the laser.³ (See Figure 1.1)

SPECKLE

The first reference to speckle phenomena can be found in the writings of Lord Rayleigh and Verdet where they refer to Fraunhofer rings or coronas. In the latter half of this century, the speckle phenomena has been studied extensively since the invention of the first cw HeNe laser in 1960. Speckle was initially regarded as pure nuisance and most research efforts were geared towards its elimination.⁴

With more understanding of the cause of speckle, it became a method of studying such things as surface roughness, deformation and strain measurements, crack detection in natural materials, etc.^{5 6 7}

The first successful astronomical use of speckle was by Labeyrie, et al (1970). He showed that each short exposure ($\approx 1/100$ sec) speckle pattern intensity could be represented by

$$I(x,y) = O(x,y) \otimes P(x,y) \quad (1.2)$$

where $I(x,y)$ is the instantaneous image intensity, $O(x,y)$ is the objects intensity, and $P(x,y)$ is the point spread function of the atmosphere/ telescope. It was then shown that in the spatial domain, the ensemble average space auto-correlation is

$$\begin{aligned} C(x,y) &= \langle I(x,y) \star I(x,y) \rangle \\ &= \{ O(x,y) \star O(x,y) \} \otimes \{ \langle P(x,y) \star P(x,y) \rangle \} \end{aligned} \quad (1.3)$$

This is the imaging equation with $C(x,y)$ equal to the average auto-correlation of the intensity.⁸ By switching to frequency

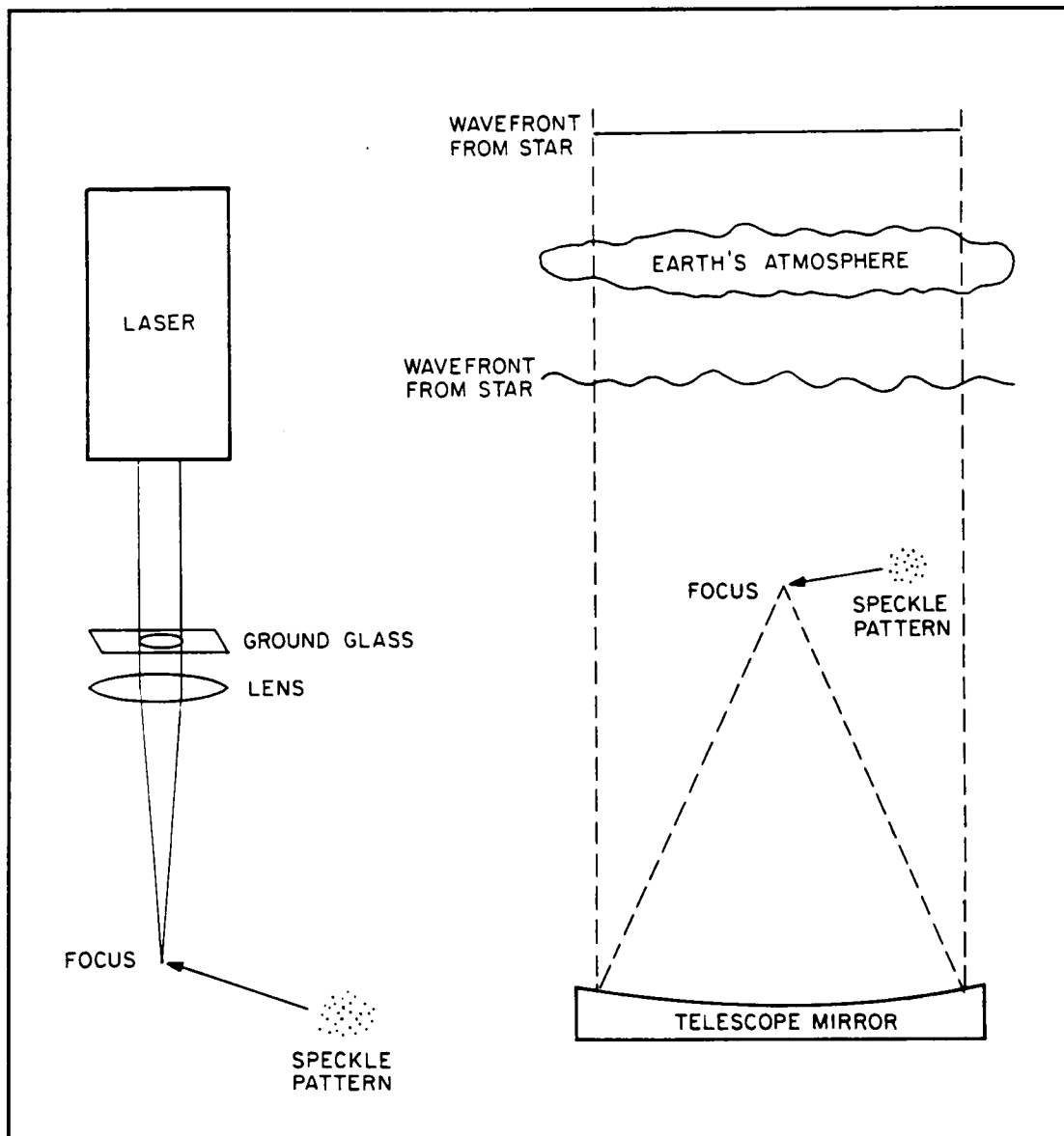


Figure 1.1 - Diagram of Different Speckle Producing Methods. Source: Robert Stachnik, "Infinite Vistas", Chapter 3, Ed. James Cornell & John Carr, Charles Scribner's Sons, New York, 1985, 46.

space the mathematics is easier and convolution can be replaced with multiplication.

By the Wiener-Khinchine theorem:

$$\mathcal{F}\{G(x,y) \star G(x,y)\} = |g(u,v)|^2$$

where the Fourier transform of $G(x,y)$ is

$$\begin{aligned}\mathcal{F}\{G(x,y)\} &= g(u,v) \\ \therefore \mathcal{F}\{C(x,y)\} &= \mathcal{F}\left[\{O(x,y) \star O(x,y)\} \otimes \{<P(x,y) \star P(x,y)>\}\right] \\ &= |o(u,v)|^2 \cdot <|T(u,v)|^2>\end{aligned}\quad (1.4)$$

where

$$\begin{aligned}\mathcal{F}\{O(x,y)\} &= o(u,v) \text{ and } \mathcal{F}\{<P(x,y)>\} = <T(u,v)> \\ \therefore W(u,v) &= <|i(u,v)|^2> \\ &= |o(u,v)|^2 \cdot <|T(u,v)|^2>\end{aligned}\quad (1.5)$$

$W(u,v)$ is equal to the power spectrum of the speckle image intensity and $i(u,v)$ is the Fourier transform of the image intensity.⁹

The essence of Labeyrie's method is to average the speckle image intensity FFT power spectrum.^{10 11}

$$\frac{1}{N} \sum_{i=1}^N |i(u,v)|^2 = <|i(u,v)|^2>\quad (1.6)$$

In Figure 1.2, a binary star's intensity is represented. From a speckle generation routine, 100 speckle images were generated. The Fast Fourier Transform (FFT) was taken of each image from which the modulus squared (power spectrum) was derived.

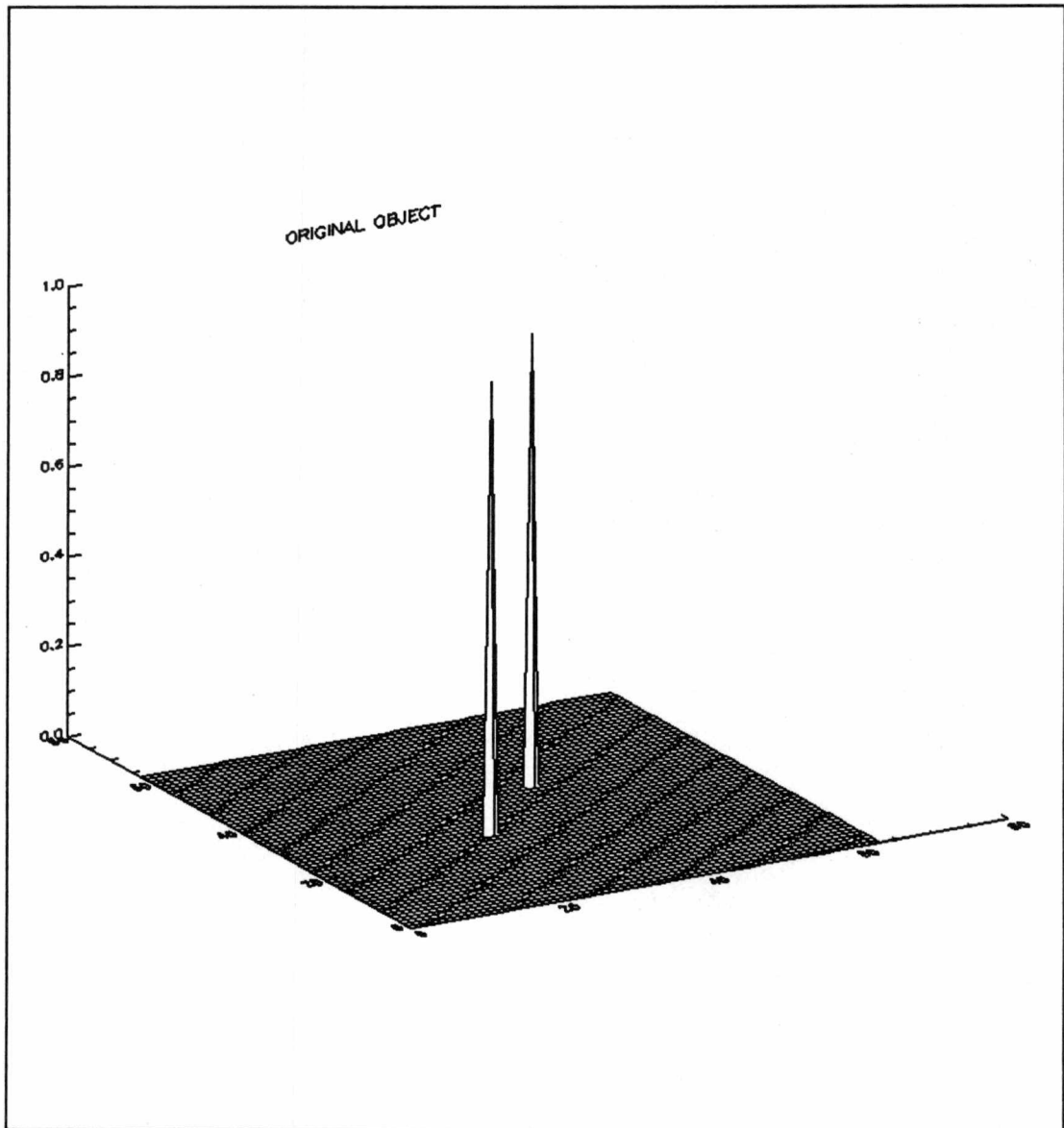


Figure 1.2 - Surface Intensity Plot of a Simulated Binary Star.

These were then added together to form the average squared modulus of the speckle FFT. (See Figure 1.3). The summed modulus was then inverse transformed (Frame 3, Fig. 1.3). and then shifted to center mass (Frame 4, Figure 1.3). This is now referred to as the spatial auto-correlation of the intensity (Figure 1.4).¹² This method allows for diameter and separation measurements of binary stars.¹³ This process is referred to as stellar speckle interferometry and has been the basis for all stellar speckle image recovery techniques.¹⁴ One fundamental drawback of the Labeyrie method is that the object's phase information is lost. The speckle interferometry recovery method only yields the power spectrum of the Fourier transform of image intensity or equivalently the auto-correlation of the image intensity.¹⁵ By observing an unresolved star, in the same portion of the sky the average transfer function of the atmosphere/telescope can be computed which yields the auto-correlation of the object. The actual intensity distribution of the object itself can be computed if the object is known to be centro-symmetric or has other special identifying features. In general though, this technique is limited by its inability to provide the actual object intensity distribution.¹⁶

Due to the short-comings of the stellar speckle interferometry technique, a process known as speckle imaging was developed by Knox and Thompson, et al (1974). This method produces the phase estimate of the object transform though it

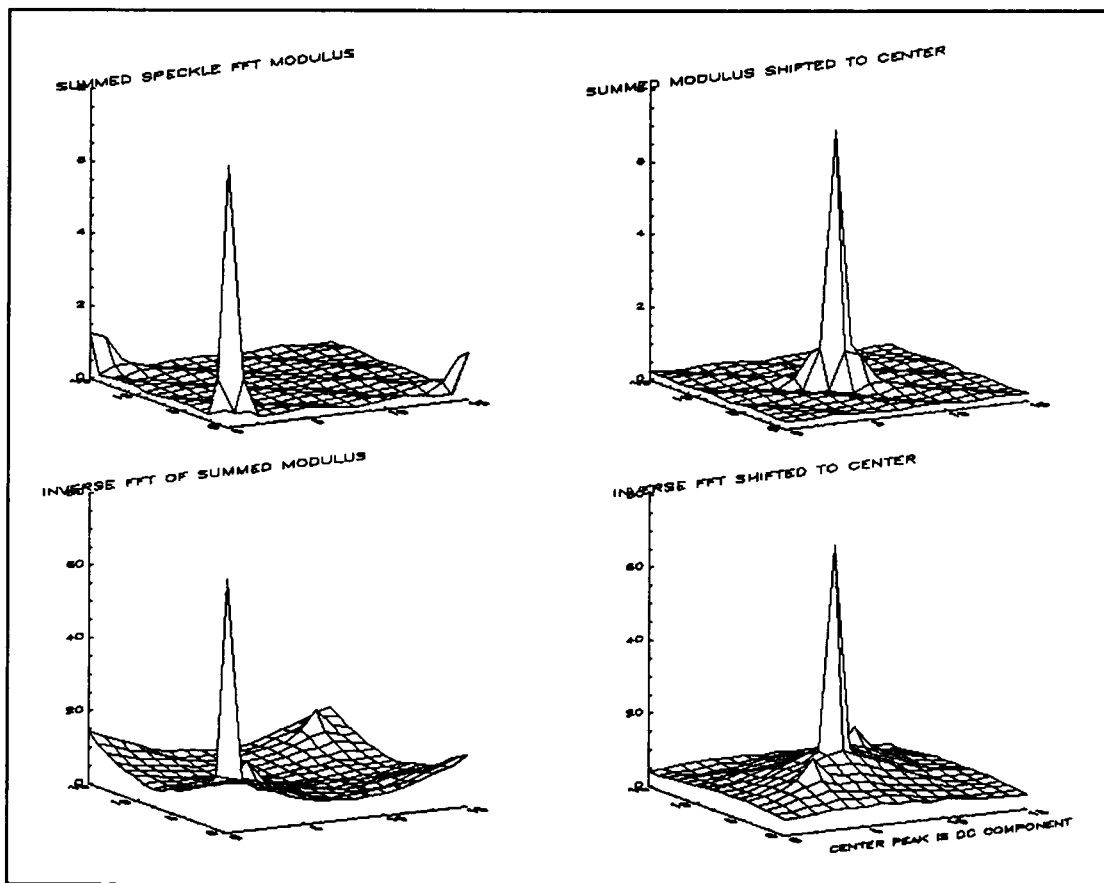


Figure 1.3 - Summation of FFT's of Speckle Images Created by Binary Star. Auto-correlation of image depicted in last frame.

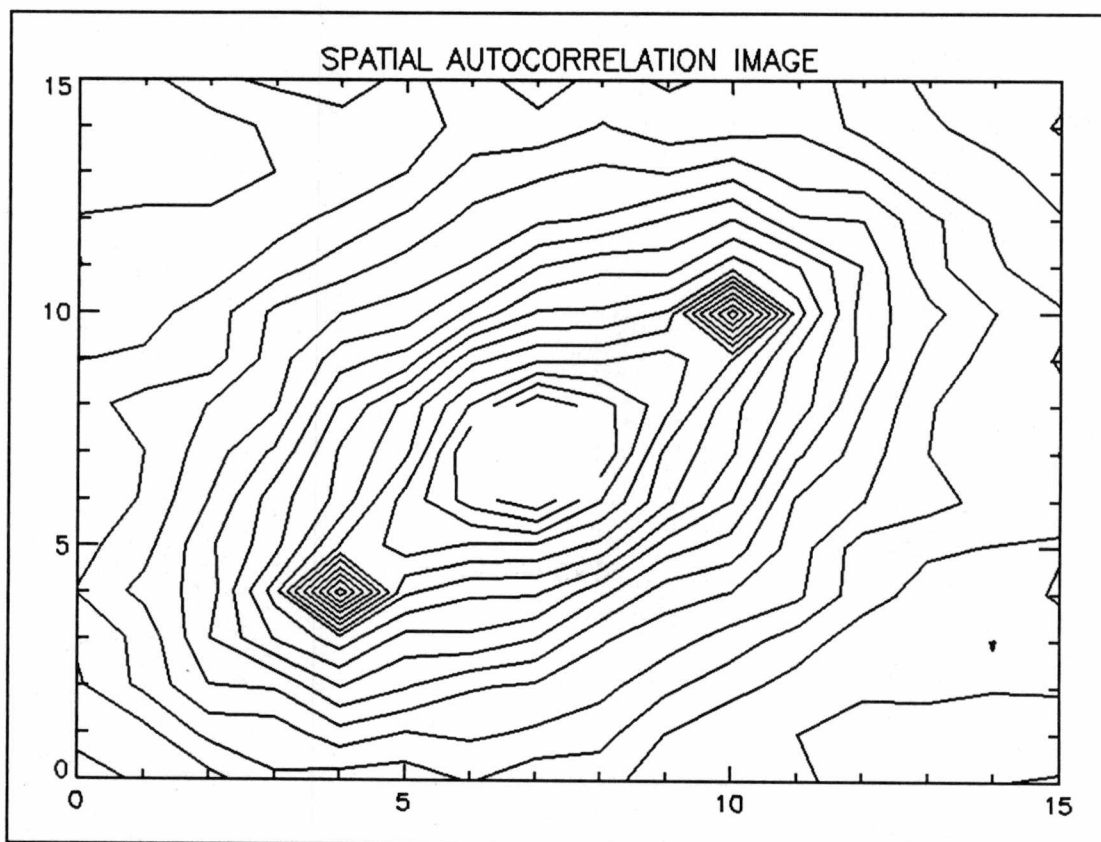


Figure 1.4 - Spatial Auto-Correlation of Image Intensity with DC Component Blocked.

uses a slightly different averaging technique. The phase information is derived from the auto-correlation of the Fourier transform of the Image Intensity $I(x,y)$

$$\langle i(u,v) \star i(u,v) \rangle = \{ o(u,v) \star o(u,v) \} \cdot \{ T(u,v) \star T(u,v) \}$$

(1.7)

This method permits the recovery of the phase of the object transform which has its modulus computed from the Labeyrie technique, Equation 1.6.¹⁷ Other investigators have also proposed various techniques (e.g. High Resolution Image Formation Through a Turbulent Atmosphere, Liu & Lohmann et al (1973) and Speckle Masking, Lohmann, Weiglt, and Wirnitzer (1983)).¹⁸ ¹⁹ The principal drawbacks to these methods are that they are not real time recovery and they are computationally intense (slow).

Recently declassified work done by Defense Advanced Research Project Agency (DARPA) and Strategic Defense Initiative (SDI) indicates a great deal of progress has been made in the use of adaptive optics systems for real time recovery of speckle distorted images. Adaptive optics works by measuring the turbulent atmosphere effects on the light of a guide star and then adjusting deformable mirrors to conjugate the effects.²⁰ These devices though suffer from extreme sophistication and expense, and still suffer some limitations on their capability.

Since none of the aforementioned techniques meets the desired goal of simplicity, near real time recovery and low

cost, this provided the impetus to examine the possibility of using a neural network to model/map the speckle producing atmosphere in hopes that with a proper model, speckle images would provide enough information for object recovery and afford the possibility of identifying previously unseen objects.

NEURAL NETWORKS

A neural network offers the user one very important advantage over other computational techniques. It will map from a given domain to a specified range without any information being provided with regard to what the mapping function looks like. In situations such as the modelling of a turbulent atmosphere, this is a highly desirable attribute. This, of course, does not guarantee that a proper mapping will be achieved or even that one is feasible, but it is a first step.

What is a neural network? Simply defined, it is a processing algorithm loosely based on a biological simulation of the nervous system/brain, that can "learn". The key words here are loosely and learn. Many of the networks in use today require a very loose definition of a nervous system to find any biological similarities.²¹

What is meant by learning? Usually, it is thought of as the network's ability through training to reproduce some output based on a specified input. Further, it is considered to be the ability of the network to generalize or respond

satisfactorily to some previously unseen input.²²

A very general network which might be found in current literature would probably look like Figure 1.5., where the letters refer to "layers" and the numbered circles correspond to processing elements (PE's) or neurons. The lines between PE's represent the interconnectivity of the network. It is the dot structures represented on the lines commonly referred to as weights which actually serve as the dynamic structures of "learning." Simply put, the network must be trained. The most popular networks today and the one used for this research accomplish this training in a supervised manner. This simply means a comparison must be available to the network to provide feedback on how well the network is learning which then dictates how the learning structures (weights) must be altered to accomplish the desired output for a specific input. The training set must consist of input signals or patterns that correspond to a specific output pattern. A common example is the XOR problem which serves as a standard benchmark test in neural network theory.

If A and B of Figure 1.6 are the four possible combinations for inputs to our network, then there are two possibilities which can occur for the output. The possible input patterns from the table are repeatedly applied to the inputs until the weights are altered in such a manner that the application of PATTERN 2 (INPUT A=1, INPUT B=0) for example results in an output signal of 1.

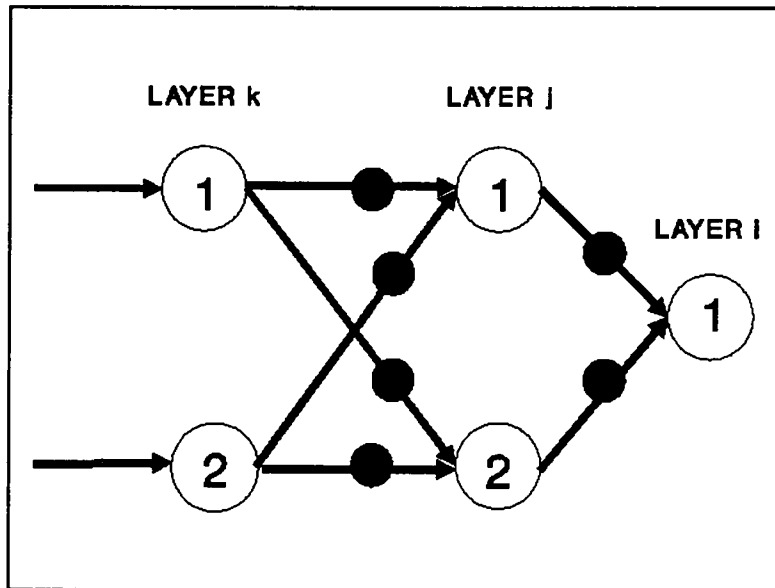


Figure 1.5 - General Neural Network Configuration.

XOR FUNCTION

	INPUT A	INPUT B	OUTPUT
PATTERN 1	0	0	0
PATTERN 2	1	0	1
PATTERN 3	0	1	1
PATTERN 4	1	1	0

Figure 1.6 - The Exclusive OR Function.

This learning is accomplished in a fairly straightforward manner. Most supervised networks use what is called the "Delta Rule" for learning. It requires that weights on the interconnects change in value in order for the net to learn. When an input is applied it propagates down the connection until it encounters the weight. At that point, it can be thought of as being multiplied by the weight. This happens on all the connections entering a PE. In linear algebra notation, this general equation is

$$I_j = \sum_k w_{jk} x_k \quad (1.8)$$

for most supervised learning networks regardless of architecture. Here, x_k symbolizes the specific input or signal and w_{jk} is the specific weight on the interconnection. For our net, Figure 1.5, the input into PE 1 of layer 1 is

$$I_1 = w_{11}x_1 + w_{12}x_2.$$

The weight matrix for layer 2 serves as the weight space for this layer.

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

The total input to PE1 and PE2 of layer 1 is therefore

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The value I_j is then used as the argument of a function called the activation function of the PE. This can be pictured as embedded in the PE. See Figure 1.7.

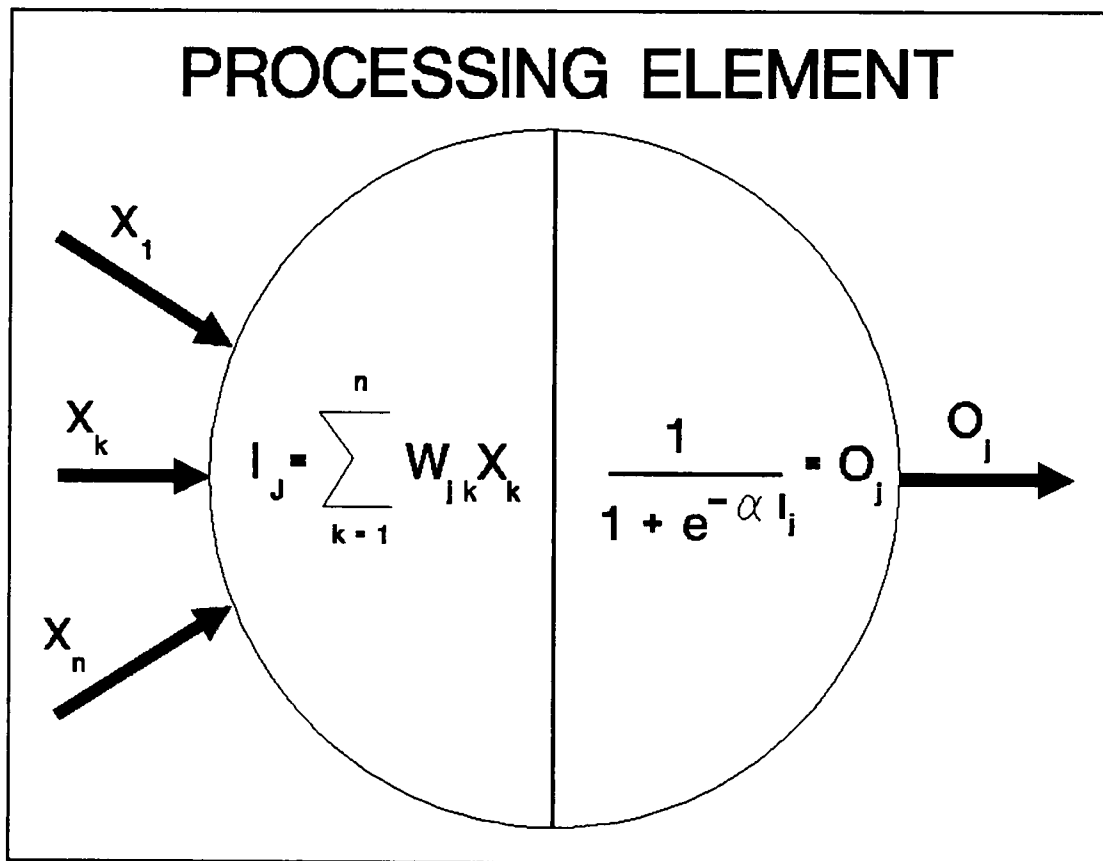


Figure 1.7 - Generalized Processing Element Construction.

The activation function $f(I_j) = 0$, then outputs a value which travels along the connection leaving the PE until it encounters another weight or if the PE is in the output layer, $f(I)$ is then a computed output for the network.

This process takes place at all PE's and can be thought of as happening almost simultaneously although software implementation currently cannot realize this aspect of real biological neural systems. It is the propagation of inputs forward through the net that explains why these types of networks as usually referred to as feed forward networks. I have not made mention of how the weights are changed to facilitate learning. Weight changing methods are specific to the type of network implemented. This paper will only focus on the multi-layer feed forward network called back-propagation. It will be discussed in detail in Chapter 2.

One of the strong capabilities of neural networks is their ability to recognize information not normally detected in other processing mediums. In this instance, speckle patterns are used as the training data/patterns. Speckle patterns are random spatial carriers of information about both the object and the atmosphere. The Fourier transform of a single speckle pattern provides intensity and position information about the stellar object. However, there is insufficient information in a single speckle pattern to be useful. The possibility exists though that a neural network can detect the relationship between a set of speckle patterns

and their stellar object. This problem is, however, more difficult than it may appear due to the inherent stochastic nature of speckle images caused by the randomness of a turbulent atmosphere.

DESIRED OBJECTIVE

The focus of this research is a combination of certain aspects of known speckle recovery methods with Neural Network Technology to produce an acceptable Speckle Image → Stellar Object Recovery model. The primary objective is recovery of a stellar object's image, intensity, and position from the speckle images it generates.

The speckle images used for recovery testing must be attributable to the stellar object, which the neural network is attempting to recover, but cannot be part of the training set of speckle images. It should be noted that the stochastic nature of speckle provides the network with a training set of unique images, each different to some degree than that of its successor captured moments later. This of course is due to the random turbulence of the atmosphere. (See Figure 3.3)

Upon completion of training, the network should recall a stellar object's image it has been trained to remember. It must utilize speckle images it has never seen before and of course these are also stochastic.

The follow-on objective of this model is to obtain a generalization capability such that a network trained on a particular training set,

$\{ \text{input}(\text{speckle patterns}) \mid \text{output}(\text{original stellar objects}) \}$,
will recover a previously unseen object's image from the
speckle images of the previously unseen object. This would be
in effect a model of a turbulent atmosphere, a mapping from
the speckle image space to the original stellar object image
space.

CHAPTER 2

ARTIFICIAL NEURAL SYSTEM

BACK-PROPAGATION ALGORITHM

At present, the most commonly used artificial neural learning model is the back-propagation algorithm. It has been found so far to be the most broadly applicable algorithm and is very easily implemented in software. Though it does have limitations, back-propagation successes can be largely credited with the resurgence of interest in artificial neural networks and their applications to real world problems. The method was first described by Werbos (1974), and subsequently by Parker (1982) and Rummelhart, Hinton, and Williams (1986).

The back-propagation training algorithm is simple to understand but tedious in detail and computation. It also suffers or is complemented (depending on your point of view) by numerous variations and modifications, depending on the particular application.

Back-propagation falls under the category of a supervised, off-line, learning algorithm. It requires that you have a desired output coupled to an input. This desired output is compared to the actual output of the net and the difference is defined as the error. It is the attempted minimization of this error that is the driving force of the back-propagation algorithm. The actual "learning" is accomplished by changing the weight values on the connections

between layers. By changing the weights, you reduce the error between the computed output and the desired output, eventually reaching a satisfactory error tolerance.

The algorithm is based on minimization of a cost function.

$$E(\vec{w}) = \frac{1}{2} [\sum_{\mu i} (D_i^\mu - O_i^\mu)^2] \quad (2.1)$$

Where μ refers to the pattern pair (input/output), i refers to the neurons of the output layer, and D and O are the desired and computed output respectively.²³

A set of inputs, x_k , is applied to layer 0 of the network. This network could be pictured as the general network depicted in Figure 1.5. Layer 0's only function is distribution of the applied input signals to the neurons of layer 1. No learning takes place at layer 0.

Inputs are then distributed from layer 0 along their respective connections and then multiplied by the weight on the connection prior to reaching the layer 1 PE's. This quantity $w_{jk}x_k$ is then summed to form the total input to a specific neuron of layer 1.

$$I_j = \sum_k w_{jk}x_k \quad (2.2)$$

Where j refers to the specific neuron of layer 1 and where w_{jk} refers to the weights on the connections to layer 1 from layer 0. This is a specific application of the general equation 1.8. The value I_j is now used as the argument of the activation

function of the PE. This activation function is usually defined as

$$O_i = \frac{1}{1 + e^{-aI_i}} \quad (2.3)$$

OR

$$O_i = \tanh(I_i) \quad (2.4)$$

Both of these equations are Sigmoidal in form with Equation 2.3 ranging in value from 0 to 1, and 2.4 ranging in value from -1 to 1.

In our example, we will refer to the activation functions of layer 1 as $g(I_j)$ and those of the output layer 2 as $g(I_i)$ where the subscript j and i refer to layers 1 and 2 respectively. This function is assumed to be of the form of Equation 2.3. The function $g(I_j)$ (activation function of the hidden layer) is defined as

$$g(I_j^\mu) = g(\sum_k w_{jk} x_k^\mu) = S_j^\mu \quad (2.5)$$

The weights on the connections between layer 1 and layer 2 are annotated by w_{ij} . Therefore, layer 2 will receive

$$I_i^\mu = \sum_j w_{ij} S_j^\mu = \sum_j w_{ij} g(w_{jk} x_k^\mu) \quad (2.6)$$

The output from layer 2 is then:

$$O_i^\mu = g(I_i^\mu) = g(\sum_j w_{ij} S_j^\mu) = g(\sum_j w_{ij} g(\sum_k w_{jk} x_k^\mu)) \quad (2.7)$$

Most networks use some type of thresholding in Equations 1.8 and 2.2.

$$I_j = \sum_k w_{jk} x_k - \theta_k \quad (2.8)$$

where θ_k is set to some constant value. Thresholds are optional and for that reason have not been included in this example. Equation 2.1 is now defined as

$$E(\vec{w}) = \frac{1}{2} [\sum_{\mu_i} (D_i^\mu - g(\sum_j w_{ij} g(\sum_k w_{jk} x_k^\mu)))^2] \quad (2.9)$$

The error function is now minimized by gradient descent. The gradient descent approach yields the weight changes necessary to minimize the error to an acceptable level.

The change in weights Δw_{jk} or Δw_{ij} is defined as proportional to the gradient of Equation 2.1 at that point in time and location in weight space w .

Through use of the chain rule for differentiation the change for weights w_{ij} is defined as

$$\begin{aligned} \Delta w_{ij} &= -\alpha \frac{\partial E}{\partial w_{ij}} = \alpha \sum_{\mu} (D_i^\mu - O_i^\mu) g'(I_i^\mu) S_j^\mu \\ &= \alpha \delta_i^\mu S_j^\mu \end{aligned} \quad (2.10)$$

where

$$\delta_i^\mu = (D_i^\mu - O_i^\mu) g'(I_i^\mu) \quad (2.11)$$

Alpha is referred to as a learning or step constant. This usually takes a value somewhere in the range between .001 to .5. This permits the steps or changes to remain small, therefore allowing the weight changes to slowly approach a desired error tolerance without overshooting the desired tolerance.

To determine the weights on the connections between layer 0 and layer 1 (hidden layer), simply requires finding the gradient of Equation 2.1 with respect to w_{jk} .

$$\begin{aligned}
 \Delta w_{jk} &= -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \sum_{\mu} \frac{\partial E}{\partial S_j^{\mu}} \frac{\partial S_j^{\mu}}{\partial w_{jk}} \\
 &= \eta \sum_{\mu} i (D_i^{\mu} - O_i^{\mu}) g'(I_i^{\mu} w_{ij}) g'(I_j^{\mu}) X_k^{\mu} \quad (2.12) \\
 &= \eta \sum_{\mu} i \delta_i^{\mu} w_{ij} g'(I_j^{\mu}) X_k^{\mu} \\
 &= \eta \delta_j^{\mu} X_k^{\mu}
 \end{aligned}$$

where

$$\delta_j^{\mu} = g'(I_j^{\mu}) \sum_{\mu i} w_{ij} \delta_i^{\mu} \quad (2.13)$$

This change in weights can be accomplished in one of two ways. They can be computed as sums over the total number of training sets and then changed as has been represented, or they can be changed (updated) after each training set is presented. Both methods allow accomplishment of the same goal, however the method used is dependent upon the problem or application.²⁴ The rules for back-propagation can now be delineated simply from the equations 2.10 and 2.12. The steps are:

1. Initially set the weights on all connections to random values between -1 to 1.
2. Propagate the input signal through the net resulting in the output for the neurons in layer 2.
3. Using the δ_i definition as defined in Equation 2.10, calculate Δw_{ij} . This is simply the

derivative of the activation function of layer 2, $g'(I_i)$, multiplied by the error $(D_i - O_i)$ for that output neuron, then multiplied by the incoming signal S_j on the particular connection for which the weight W_{ij} is being adjusted. The new weight is then

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}$$

where t refers to the particular cycle.

4. To adjust the weights between layer 0 and layer 1, δ_j must be computed according to Equation 2.13. δ_j is the derivative of the activation function of layer 1 neurons, multiplied by the $\sum_i W_{ij} \delta_i$. δ_j is multiplied by the incoming signal, x_k , on the connection of the particular weight that is being adjusted to yield the weight change for that layer. The only difference between the hidden layer adjustment and the output layer adjustments is the definitions of the δ for that layer. Both layer adjustments are equal to the activation function derivative of that particular layer multiplied by the δ for that layer multiplied by the incoming signal to the weight being changed. Therefore, the new weight for incoming layer 1 connections is

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}.$$

5. Propagate the next set of inputs through the net and repeat steps 3 and 5. Continue to repeat 3-5 until an acceptable error $D_i - O_i$ is reached for each output neuron per input signal/pattern presented.

NETS NEURAL NETWORK PROGRAM

NETS is a neural network software implementation of a back-propagation algorithm. It was developed by a NASA team at Johnson Space Flight Center to provide an additional tool for image processing research. It's primary use is found in pattern recognition problems of which the speckle problem is an excellent candidate.

NETS was initially chosen for one reason. It could be implemented on a mainframe. Most neural network software programs are currently written for PC's. NETS allows implementation on just about anything as long as a C compiler is available. The mainframe requirement was recognized early on since one training set of speckle images and their respective objects required 3.5 M-bytes of storage. Storage and throughput requirements for training and testing data were immense and impractical on a PC.

NETS is written in C source code which is provided to the user. This allows user modification if necessary for a particular application. NETS has certain built-in features alluded to in the earlier discussion of back-propagation. It allows the user to set the momentum and learning constant for

each layer. It also permits a dynamic learning rate that changes faster when the network error is at one of the two extremes. An extremely useful feature of NETS is the flexibility allowed with interconnections between layers. NETS allows the user certain flexibility in how layers of PE's are interconnected. This flexibility enhances the capability of the network over more conventional fully connected architectures. NETS proved to be an excellent choice of back-propagation algorithms. It's high degree of flexibility allowed for quick changes to the network architecture and parameters. It provided excellent feedback on network performance as the network learned and only under a very limited number of circumstances did the net fail to converge and this was usually due to operator error. If there is any semblance of a detectable pattern in the data, NETS appears to recognize it and "learn" it.²⁵

VAX 9000

The primary training platform for this research was a DEC VAX 9000. This machine was on loan for one year to the UTK computing center under a research grant. This system consisted of two vector processors, two scalar processors and 512 megabytes of memory. The VAX 9000 operated at roughly three times the speed of other available VAX processors at UTK. Without this system, the research would obviously have taken much longer and may have proved to be impractical. As an example, the VAX 9000 was able to process 3.5 M-bytes of

training data and complete one cycle of training in 26.7 seconds as compared to the VAX 4 processor which required 120.8 seconds. These times would vary somewhat based on the number of users. A system with the VAX 9000 computing power is required if serious research into neural network computer models is to be attempted.

CHAPTER 3

ARTIFICIAL SPECKLE IMAGE GENERATION

(When reading the following section, it will be necessary to reference Appendix 1. This will aid in your understanding of the discussion. All program routines referred to in Chapters 3 and 4 were written utilizing Interactive Data Language (IDL). These programs were then compiled and executed on the LEPVAX System at Goddard Spaceflight Center).

Knowledge of the actual mechanics of speckle creation has been understood since Labeyrie suggested the relation between stellar speckle images and those created by lasers.²⁶ Thompson and Knox (1974) also suggested algorithms for computer generated speckle. The specific algorithm used for the speckle creation in this paper was designed and written by Dr. Gordon Chin of Goddard Spaceflight Center. He initially modelled a telescopic aperture of 256 x 256 pixels, which then produced a speckle image of 64 x 64 pixels. This image size was too large for the available computer system to include the Vax 9000. Average training times per cycle were approximately 10 minutes. This was unacceptable. The program was modified to a telescopic aperture of 64 x 64 pixels with an actual object detection area of 16 x 16 pixels centered within the 64 x 64 aperture. This then produced a speckle image of 16 x 16 (256 pixel values).

ATMOSPHERE SIMULATION

The atmosphere is modelled on the concept that turbulent cells are present in the atmosphere. An increase or decrease in the number of turbulent cells increases or decreases the seeing respectively. These cells can be thought of as bubbles. In this algorithm, each turbulent cell is modelled as a bubble. The turbulent cells act as the dispersive points for the atmospheric model. The phase change which causes the turbulence is just the radius of the bubble. This phase is

$$\phi = \sqrt{(x-rx)^2 + (y-ry)^2} \quad (3.1)$$

where rx and ry are the randomly selected centers of the turbulent cells. Once the values for the turbulent atmosphere are computed, they are stored in a 64 x 64 array. A surface plot of this atmospheric model array is depicted in Figure 3.1.

The semi-hemisphere effect is a result of the phase difference from a perfect sphere. The small bumps and nodules are a result of the random radius of the turbulent cells which results in smaller nodes being superimposed upon larger ones.

SPECKLE SIMULATION

The values of the atmosphere array are then used as the arguments in a complex 2-D 64 x 64 array where each pixel value of the 64 x 64 complex atmosphere array has a real value of $R = \cos(\text{atmos})$ and imaginary value of $I = \sin(\text{atmos})$. The sine and cosine arguments in the equation implement the phase of a plane wave arriving at the aperture of the telescope at a

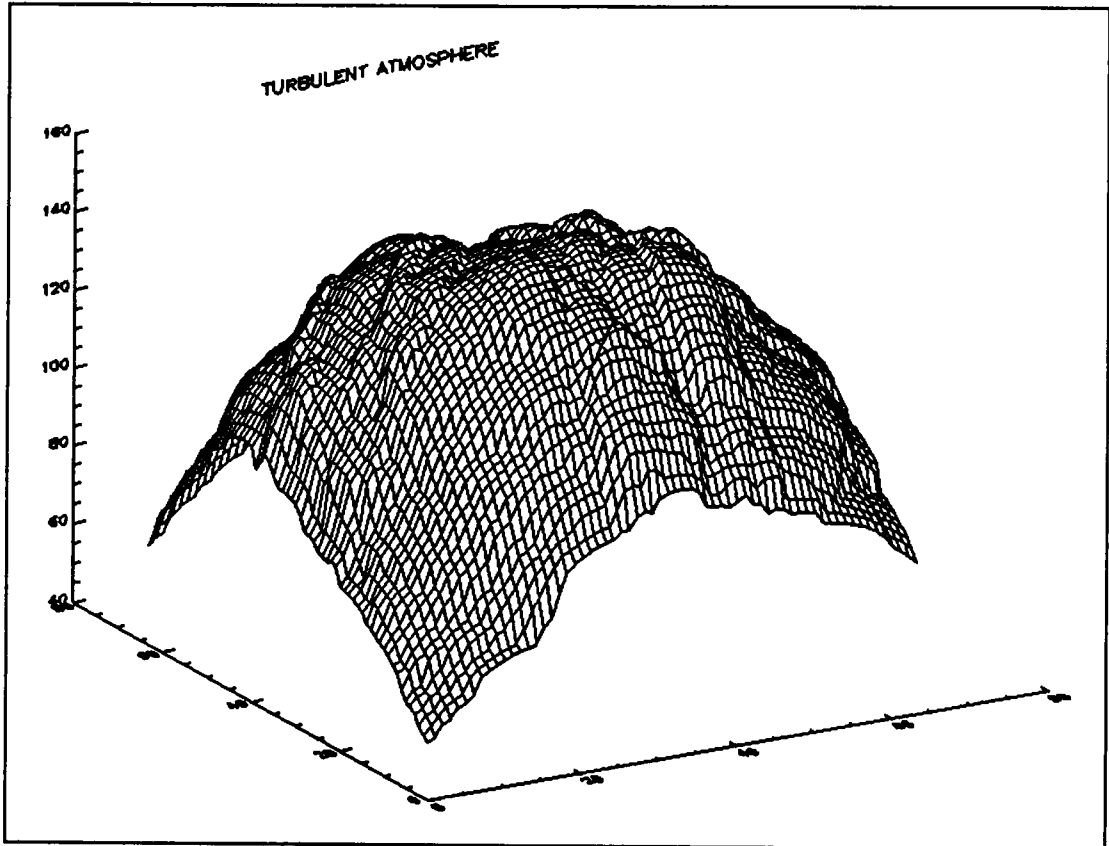


Figure 3.1 - Surface Plot of a Simulated Turbulent Atmosphere Array.

slight angle. The frequency of the sine/cosine function is equal to the angle of the plane wave. Increasing angles have higher and higher frequencies. This complex array is equivalent to the transfer function of the telescope/atmosphere. The equation in the speckle creation routine (64 x random (iseed)) randomizes the center of the turbulent cell and is also responsible for moving the cell across the aperture of the telescope.

Finally, the object image which produces a speckle image is Fourier transformed and multiplied by the transfer function (phase error) of the telescope/atmosphere which is in effect, the convolution of the object image with the atmosphere/telescope point spread function. This complex value created by the multiplication of the object FFT with the phase error (transfer function) is then inverse Fourier transformed to produce the actual speckle images. (See Figure 3.2, 3.3, and 3.4)²⁷

The entire process can be summed up as follows:

1. Atmosphere array created. (See Appendix 1).
2. Speckle procedure uses atmosphere array as argument.
3. FFT of the object's image multiplied with the phase error. This product is then inverse Fourier transformed to produce speckle image.
4. Speckle image written to a file. Number of speckle images created is user discretionary.

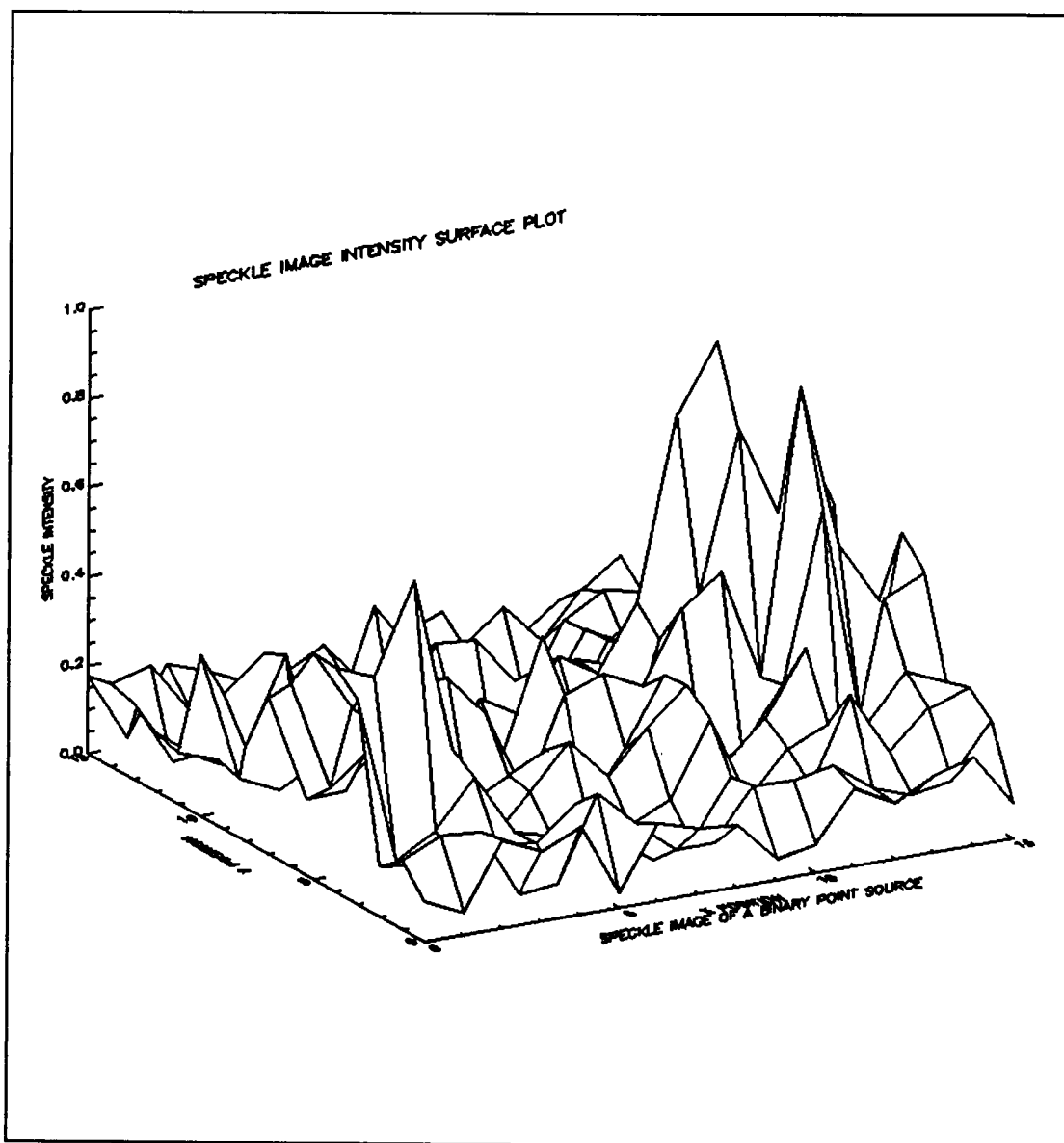


Figure 3.2 - Surface Plot of a Simulated Speckle Image.

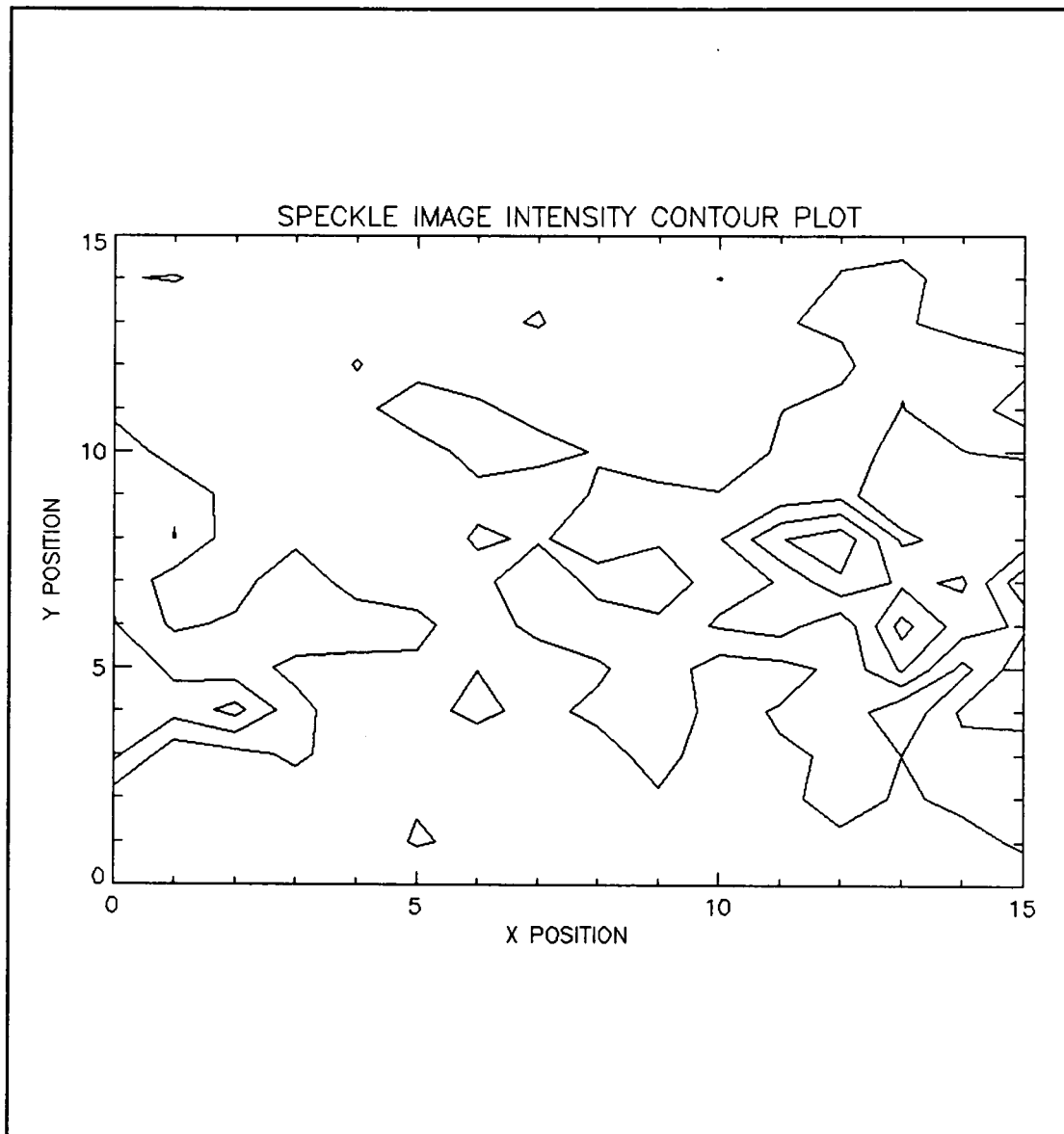


Figure 3.3 - Contour Plot of a Simulated Speckle Image (Figure 3.2).

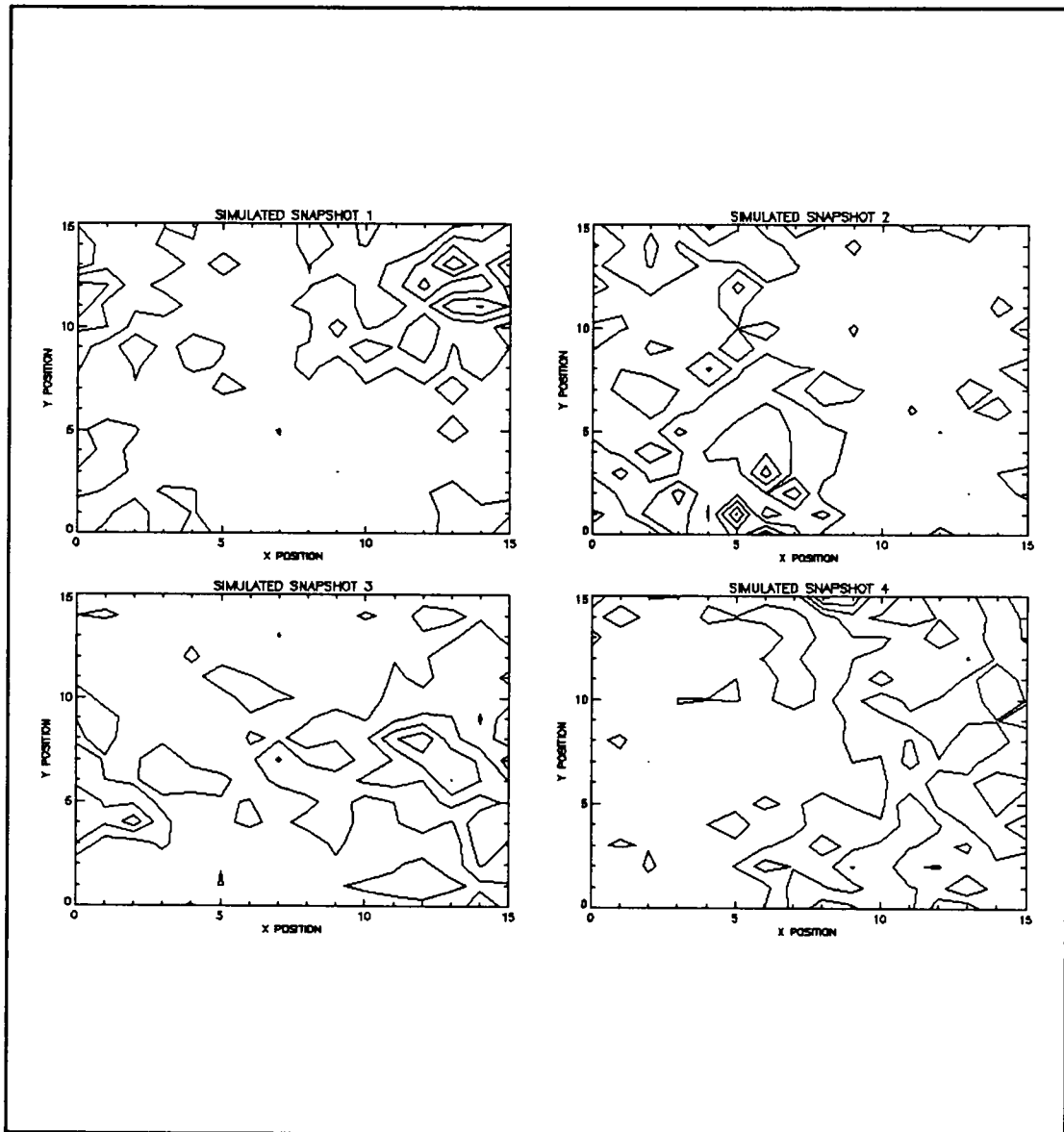


Figure 3.4 - Four Successive Speckle Patterns. These were created from the binary star in Figure 1.2. Approximate time between capture is 1/100 second.

CHAPTER 4

DATA PREPROCESSING

TRAINING SET INPUT PREPARATION

Probably more than any one factor, training data preparation is the most critical. Due to the nature of the output layer values which fall between 0 and 1, it is necessary that the desired output values which the net will use for comparison purposes with the computed output, be scaled to fall between 0 and 1. Though it is not necessary for input data to be scaled it usually is beneficial since scaling provides the numerical separation that proves beneficial in back-propagation algorithms. This numerical separation created between values coming into specific nodes on the input layer, allows for easier discernability between values of the PE's, and thus usually leads to convergence in less number of cycles. This mapping or scaling is defined by the equation:

$$Y_i = \text{NETMIN} + (\text{NETMAX} - \text{NETMIN}) (X_i - L_i) / (H_i - L_i) \quad (4.1)$$

where NETMIN and NETMAX describes the desired mapping interval min and max values respectively. The value X_i is the specific training data value to be scaled. H_i and L_i refer to the max and min value of the "working interval". The working interval is defined as the range of values of a specific input/output vector component. For example a five component input/output

vector will have five different working intervals, one for each component of the vector. If you have ten different training vectors then the max (H_i) and min (L_i) values will be selected from these ten different possibilities. This technique works quite well for additional input data outside the H_i and L_i range since it will still scale the data proportionally.²⁸

For de-scaling purposes it is merely a matter of solving Equation 4.1 for the X_i (original) values.

Prior to scaling, the speckle images were initially transformed to frequency space utilizing a Fast Fourier Transform (FFT). (See Figure 4.1)

Once transformed the modulus Equation (4.2) and phase Equation (4.3) will

$$modulus = \sqrt{R(u, v)^2 + I(u, v)^2} \quad (4.2)$$

$$phase = \tan^{-1} \left[\frac{I(u, v)}{R(u, v)} \right] \quad (4.3)$$

be calculated for the 16 x 16 FFT which provides modulus and phase values per pixel. See Figures 4.2, 4.3 and 4.4.

The phase and modulus values have all been center shifted. This has the effect of shifting the central peak of the speckle FFT to center mass while maintaining the phase relationship. The modulus and phase arrays (16 x 16) are now joined and used as a 16 x 32 input signal to the network. This concludes preprocessing of the input data.

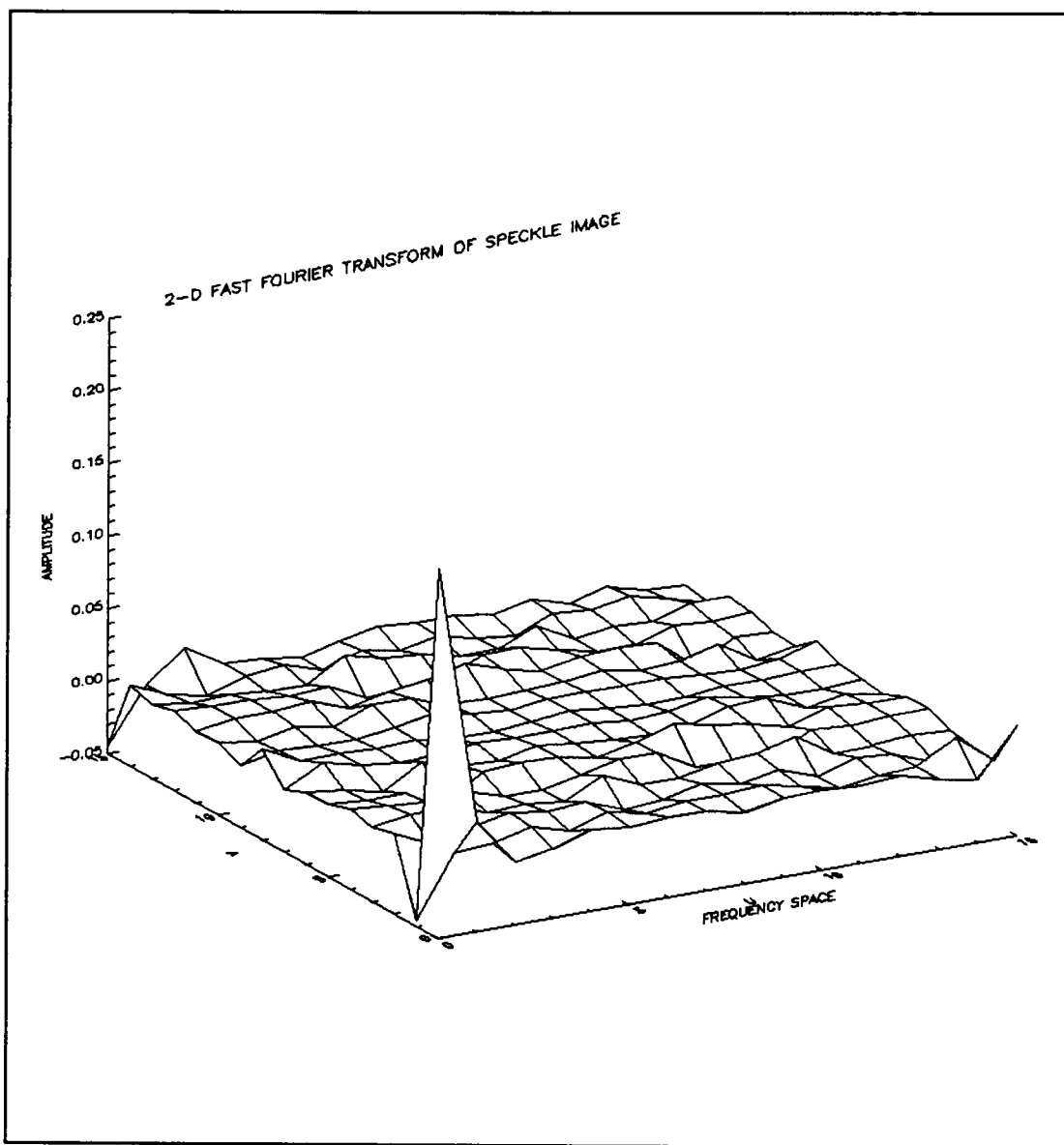


Figure 4.1 - FFT of a Single Speckle Image.

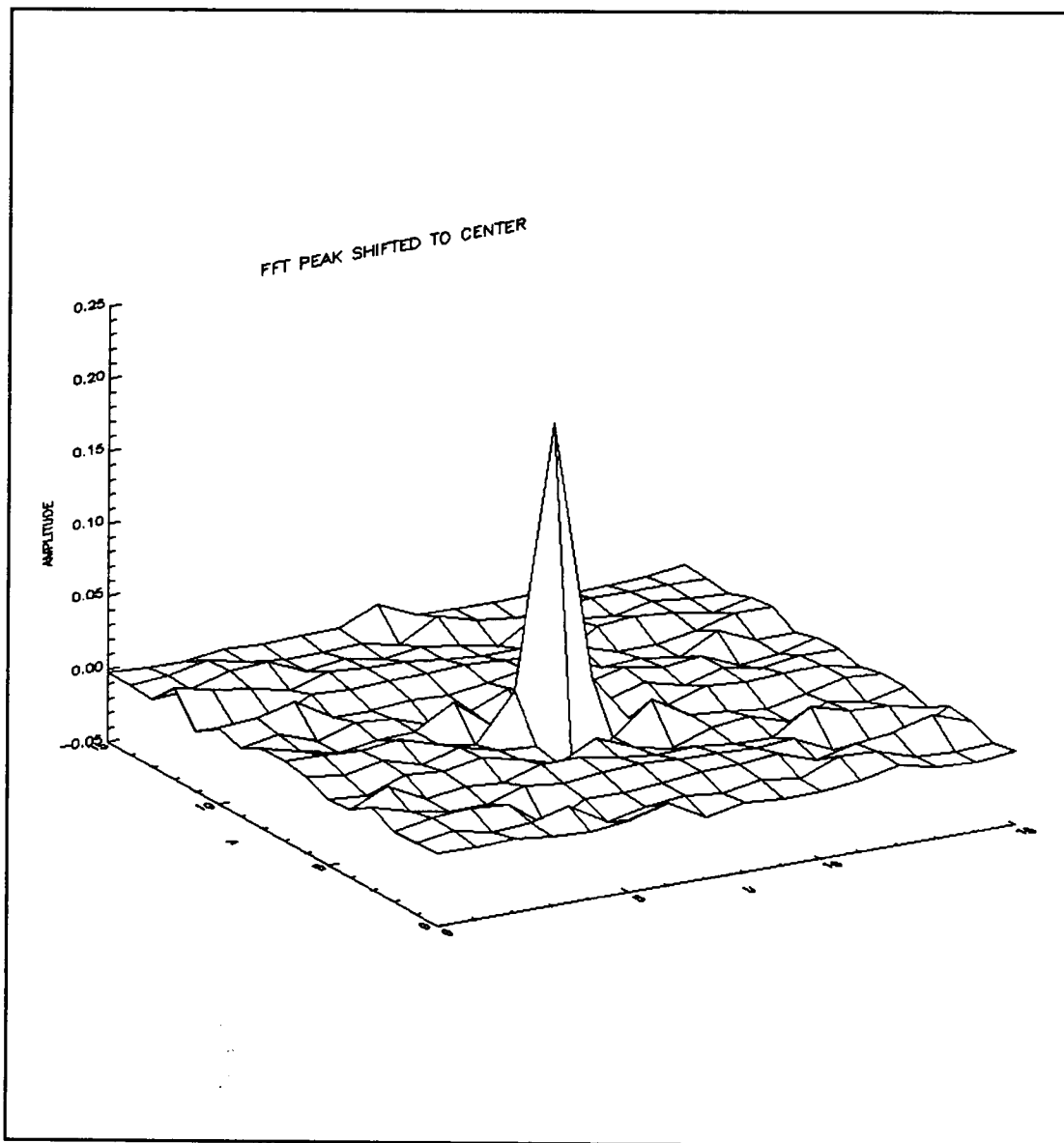


Figure 4.2 - FFT Peak Center Shifted.

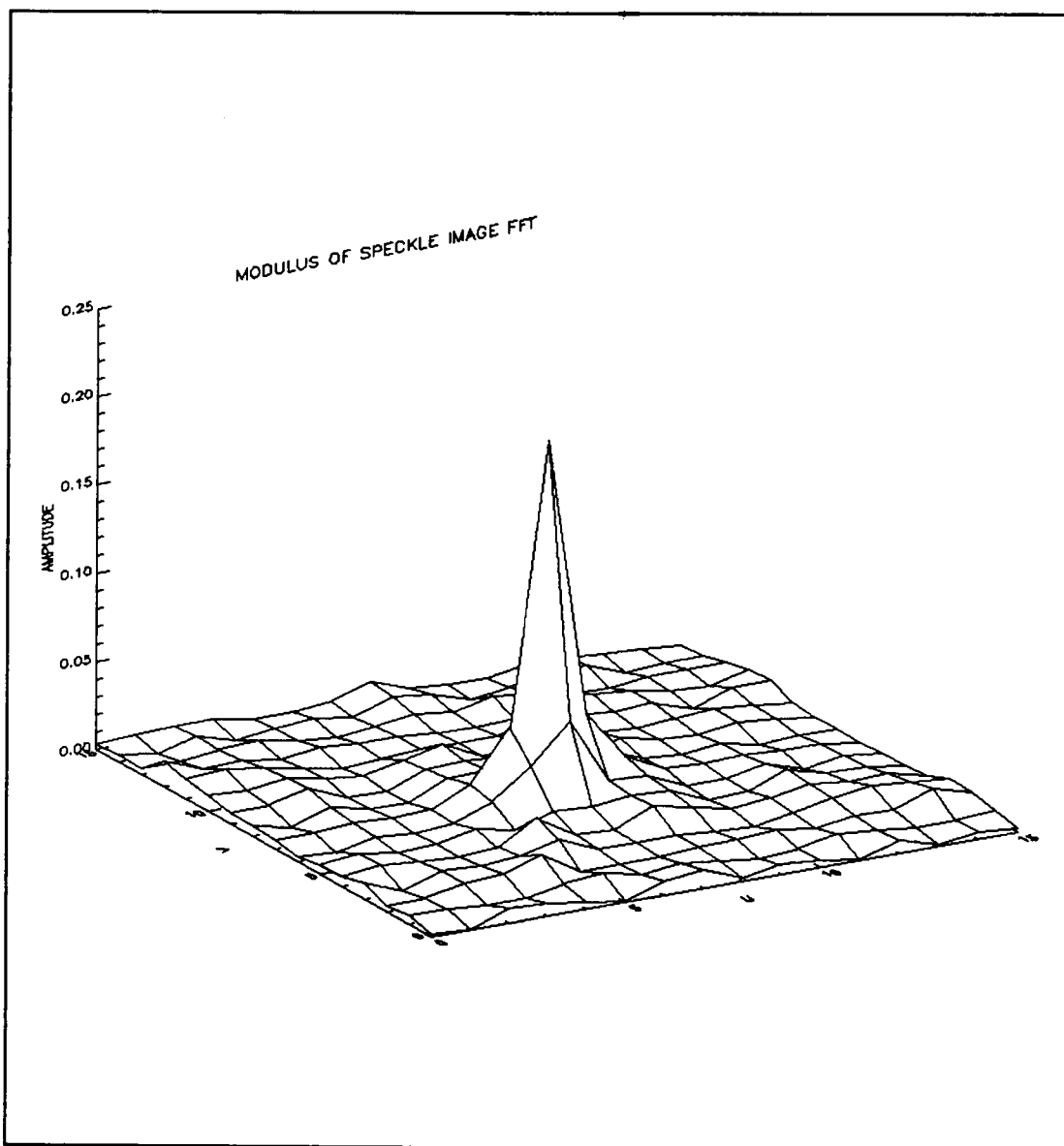


Figure 4.3 - Modulus of Single Speckle Image FFT.

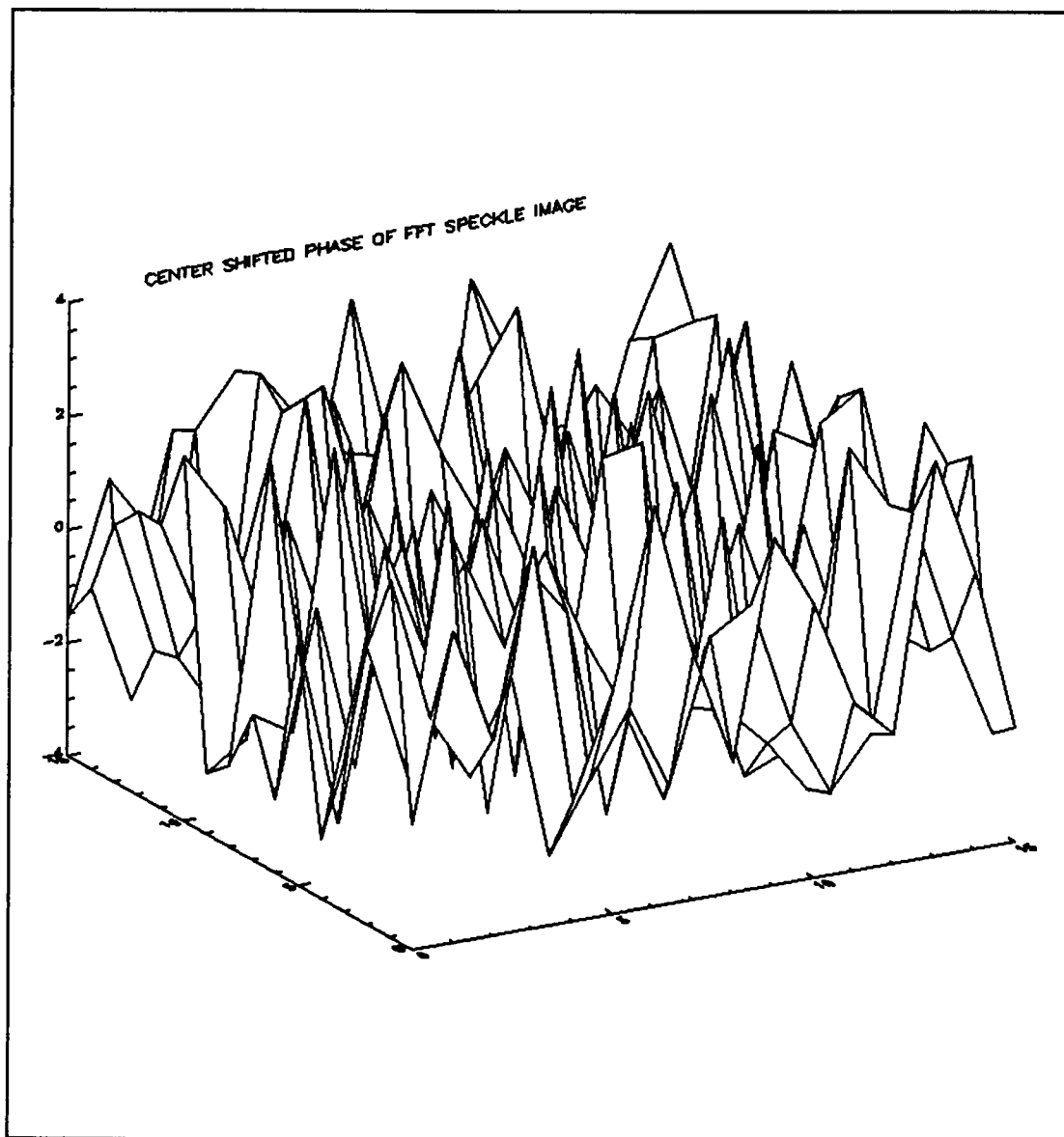


Figure 4.4 - Surface Plot of Phase Values for Center Shifted FFT.

It should be noted that other techniques were attempted with no generalization success. Initial efforts did not utilize the phase relationship and an effort was made to utilize only half of the 16×16 modulus since it is anti-symmetric. Though the net would converge under these conditions, it would not recover the original stellar objects.

TRAINING SET OUTPUT PREPARATION

The output training data is used as comparison (desired) values for the training algorithm. Because the output values of the output layer range between 0 and 1, this necessitates scaling the desired values to this interval. This is accomplished in the same scaling fashion as the input values using equation 4.1. For this problem, the raw data consisted of a 16×16 array with all values equal to zero except at the pixel value representing a point source or stellar object. This pixel would then represent the intensity and position of the point source. (See Figure 1.2). Training was attempted by utilizing the output data in this form, but extremely poor results were obtained. The poor results are primarily due to the fact that the net performs an averaging of error therefore, some distribution of desired values is necessary if convergence of the net is to be reached. After some trial and error, the following approach was implemented:

1. The Fourier transform of the output (object image) was computed.

2. The real and imaginary values for the complex Fourier transform were extracted giving two 16 x 16 arrays.
3. These values were then used as output (desired) values for which the net was trained. The output layer consisted of a 16 x 32 array. The first 16 x 16 values belonging to the real array and the second 16 x 16 values belonging to the imaginary.
4. These values were then scaled according to equation 4.1.

CONTROL SET PREPARATION

To provide some idea of how well the nets convergence will allow for generalization, a control set is created from the same objects which created the training speckle except at a later time. There is no difference in the generating programs for training and control.²⁹

During training, the net was stopped at even intervals and the control set would be propagated through on the weight values created by the training set to that point. If generalization takes place, the root mean square error of the control set will follow the training RMS towards convergence. The closer it follows the training RMS, the better the generalization.

CHAPTER 5

TRAINING THE NET

The primary drawback of Neural Networks is the training time. Training is usually done off-line and requires hours, days or in some cases weeks. The network can be constructed in such a way to optimize training time by keeping the network simple. Casimir Klimasauskas, et al (1991) suggest 5 steps to implement a neural network:

1. Collect all the data.
2. Separate the data into training and test sets.
3. Transform the DATA into network appropriate inputs/outputs.
4. Select a network. Train and Test the network.
5. Use the network in some application.

Step one was easily implemented since the data was all computer generated. Each training set had 53 speckle patterns generated per object. These were then divided into 50 training patterns and 3 test patterns. For back-propagation, the ideal training set is equally distributed between possible outcomes and this was implemented. For example, if the network was being taught four different objects (stars) located at four different positions in the field, then each star would have 50 speckle images created for that star at that location and intensity. It would also have 3 speckle images created for testing as well as a control data set. The

entire training set would then consist of 200 speckle images (50 per star). Generally, the more training examples, the better the performance of the network, but the size of your training set will be a factor in the time required to train the net. The more patterns you have, the longer the training time.³⁰

The ability to generalize is the primary attribute the network needs. The best way to achieve this is to limit its complexity. A neural network maps input to output via a sequence of weighted summations and activation functions. The changing parameters are the weights on the interconnects. To "Learn", the net must achieve the best weight values for the particular mapping. If the number of weights (connections) is reduced, then the number of mapping parameters is reduced. This will limit the possible relationships between the domain and range. This causes the network to learn the most consistent relationships between input and output. This eliminates the spurious relations that wouldn't translate from training to test without degrading performance. (Klimasauskas, 1991). This generalization is the essence of the power of neural networks.³¹

NETWORK ARCHITECTURES

A significant portion of network performance can be attributed to the architecture selected for the network. I chose three different variations of back propagation for

testing. The fully connected network, Figure 5.1, consisted of one hidden layer of 30 neurons, an input layer and an output layer of 16 x 32 neurons. This net converged to an output neuron maximum error of 1 percent, typically in 700 to 800 cycles.

The network depicted in Figure 5.2, consisted of two hidden layers with 15 processing elements each. This network achieved convergence of 1 percent error in approximately 500 cycles. It would reach a maximum error of 10 percent in as few as 30 cycles for some training sets. This network was the primary architecture used for the experiment.

The final architectures tested consisted of 3 hidden layers. Two hidden layers consisted of 15 neurons each with the other having 10 total PE. (See Figure 5.3)

This network showed no improvement over network 2 and required additional training time due to the additional interconnections (weights) which must be calculated.

NEURAL NETWORK ATMOSPHERIC MODEL

I stated in the beginning that the research goal was to create an atmospheric model. This model can most easily be thought of as the weight space of the trained neural network. Though the weight space is multidimensional and is not easily visualized, it can be thought of as the mapping parameters of the atmosphere. By altering the weights, you alter the atmospheric model. While training, the weights change until the atmospheric turbulence is effectively modelled.

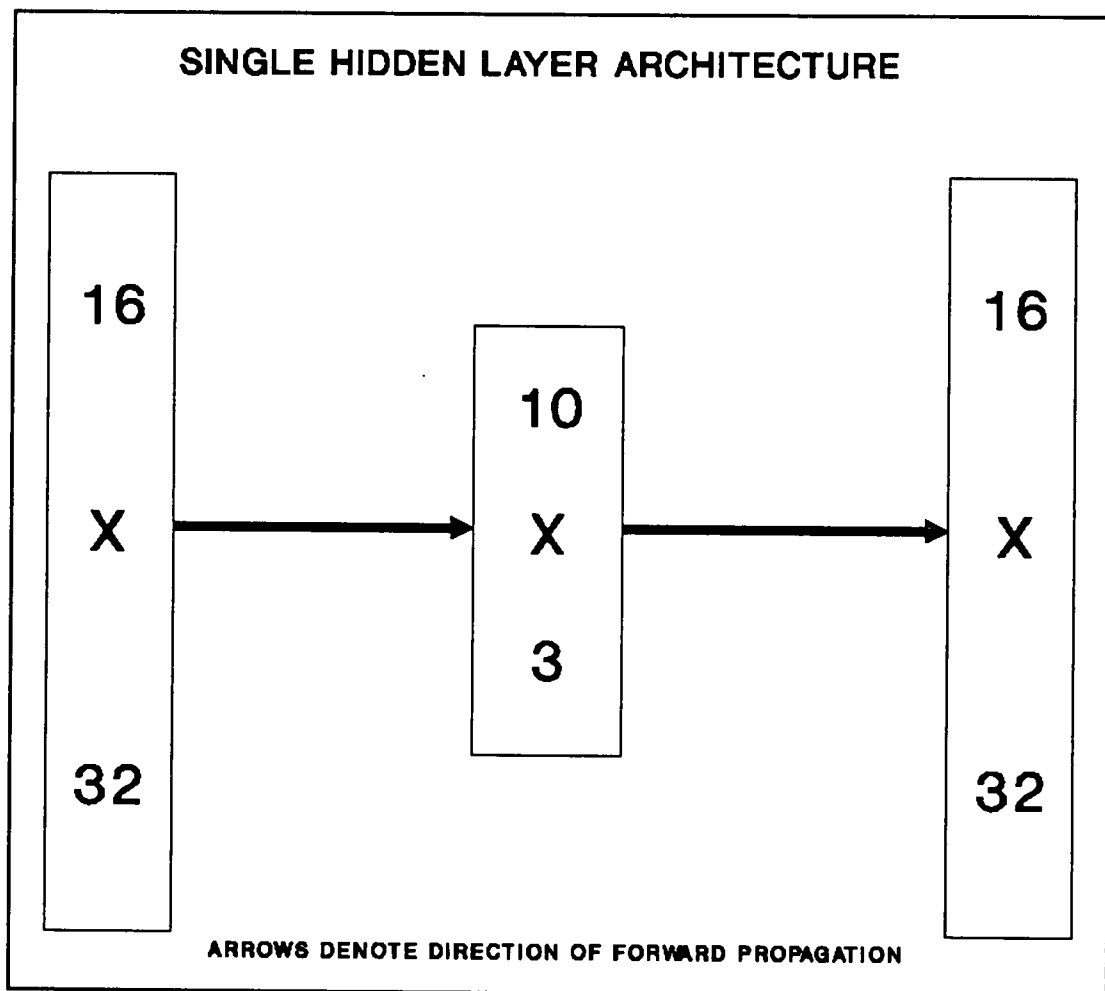


Figure 5.1 - Single Hidden Layer Network with 30 Neurons.

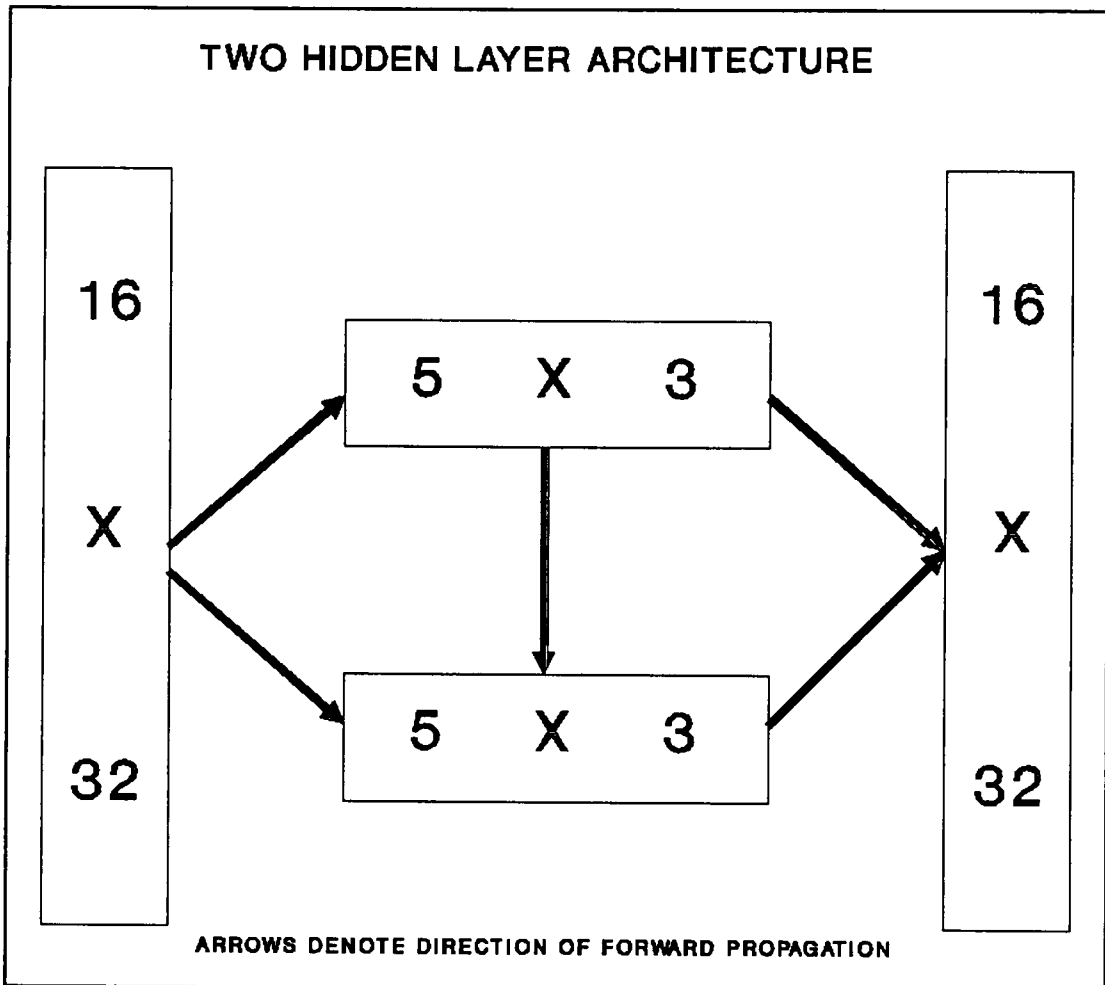


Figure 5.2 - Double Hidden Layer Network with 15 Neurons in Each Hidden Layer.

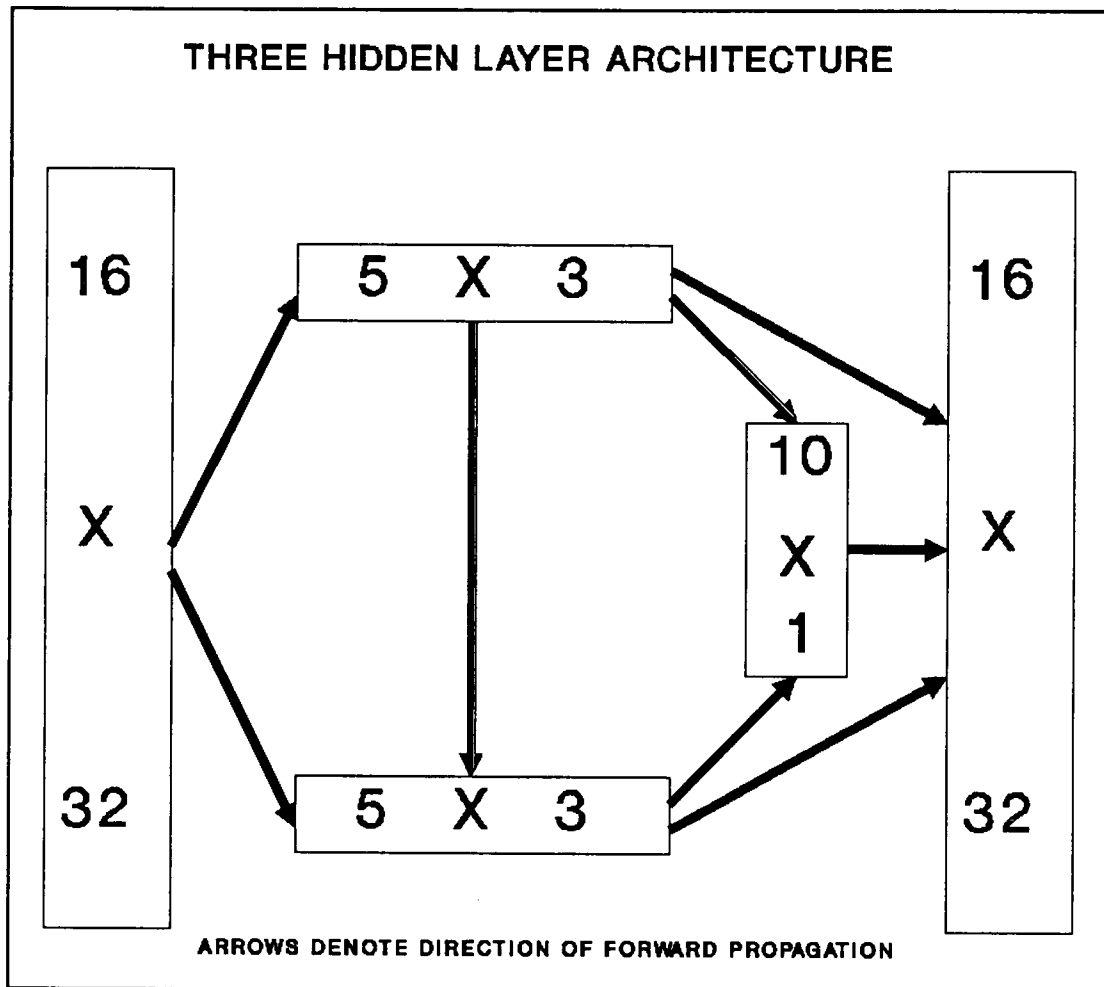


Figure 5.3 - Triple Hidden Layer Network. Two Hidden Layers have 15 Neurons each. One Hidden consist of 10 Neurons.

CHAPTER 6

EXPERIMENTAL RECOVERY SIMULATIONS

All star position charts depicted in Chapter 6 will be considered as center mass of a 64 x 64 aperture with a guard area of 24 pixels on each of the four sides. Each single, binary or triple star depicted will always be used to create 50 training speckle images and 3 test speckle images per star system. In addition, all speckle images used for recovery and referred to anywhere in Chapter 6 are **not** part of the training speckle image data set and have not been seen by the trained network prior to a recovery attempt by the trained network. Please note that in all recovery depictions, the first frame in each figure depicts the original object's image position and intensity. The second, third, and fourth frames depict the recovered object's image position, and intensity.

TEST ONE SET UP AND RESULTS

Initially, the network was presented with four single point sources (stars) of equal intensity to learn simultaneously. These were arrayed within the 64 x 64 aperture as depicted in Figure 6.1.

The training pattern for test one was presented to the net until a maximum error of .01 (convergence for all testing) was achieved for any output neuron. During training, a control data set was used to trace the degree of generalization that was taking place. The net converged in

STAR POSITION CHART

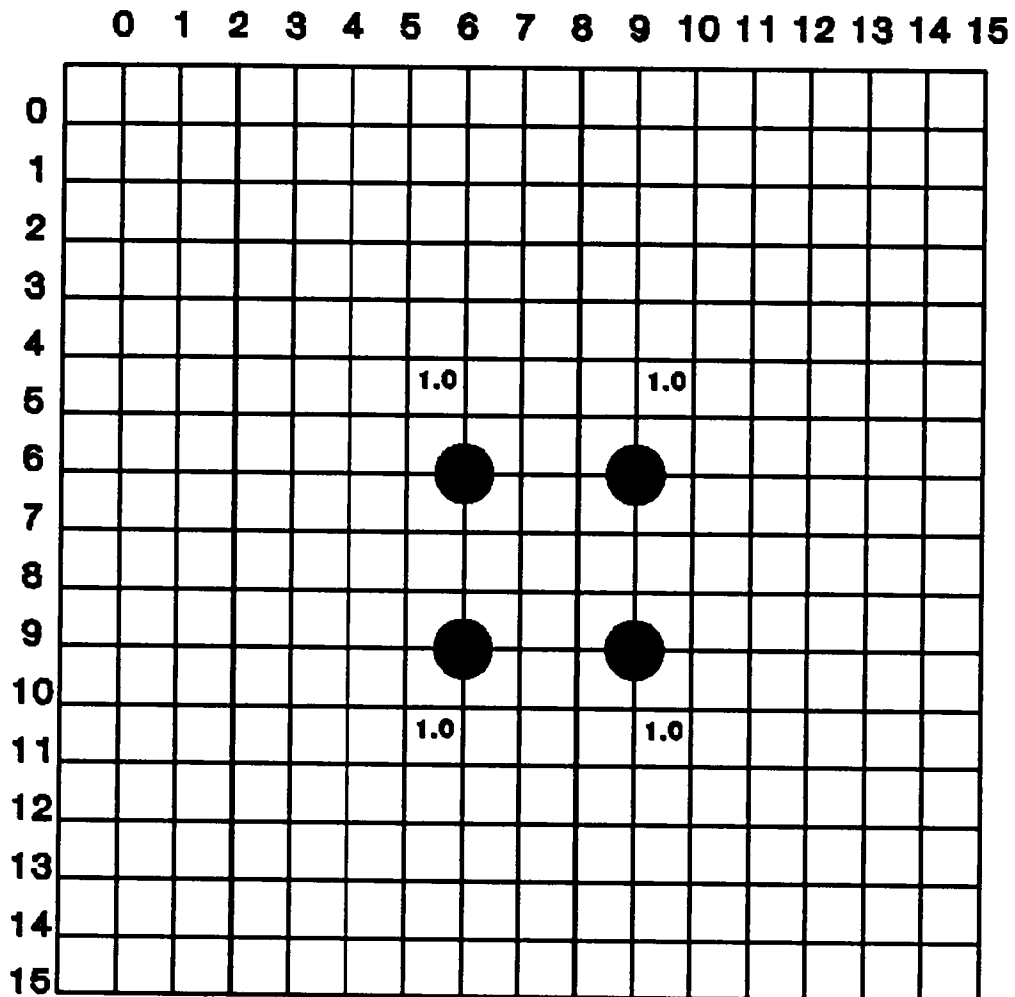


Figure 6.1 - Test One Single Star Simulations. Same Intensity but Different Positions.

approximately 780 cycles when utilizing a single hidden layer construction of 30 neurons as depicted in Figure 5.1

The control set never achieved an RMS error less than .37 during the entire training cycle. The net was reset to new initial random weights and trained again with similar results. See Figure 6.2.

Although the net converged as depicted in Figure 6.2, the lack of generalization suggest that the network memorized the training set. The net was reset with fewer neurons in the hidden layer but no improvement in generalization was seen. This was also repeated with two and three layer configurations, with similar results.

Once convergence was achieved the training test speckle patterns were propagated through the trained net. In Figure 6.3, the star located at $x=6$, $y=6$, is pictured in the first surface plot frame. The other three frames are the network output recovery from the test speckle for this particular star. As indicated in Figure 6.3, the network did not recover the original image.

Similar results were obtained with the other three original point sources depicted in Figure 6.1. Since the four original objects are arranged symmetrically in Figure 6.1, they were then randomly placed in the aperture field as depicted in Figure 6.4 to preclude the possibility that the symmetry of Figure 6.1 was responsible for the inability of the net to generalize. Note that the stars still all have the

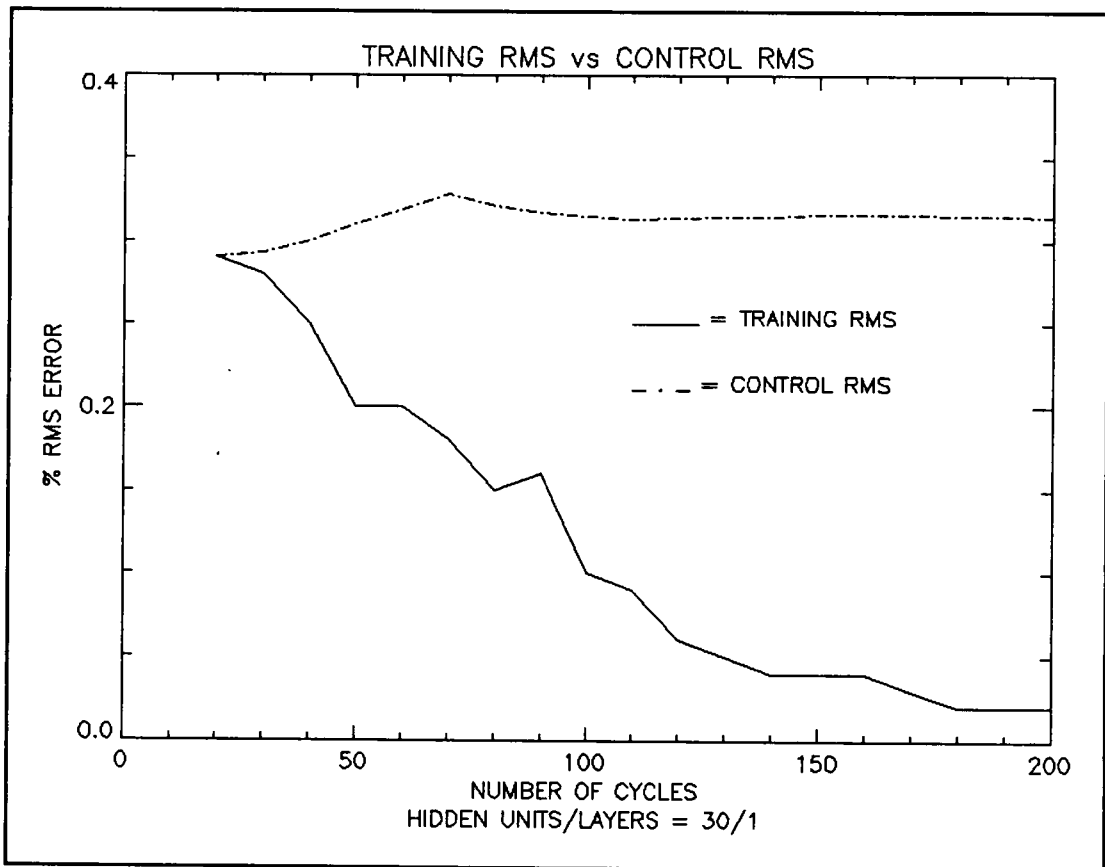


Figure 6.2 - Test One Training RMS versus Control RMS.

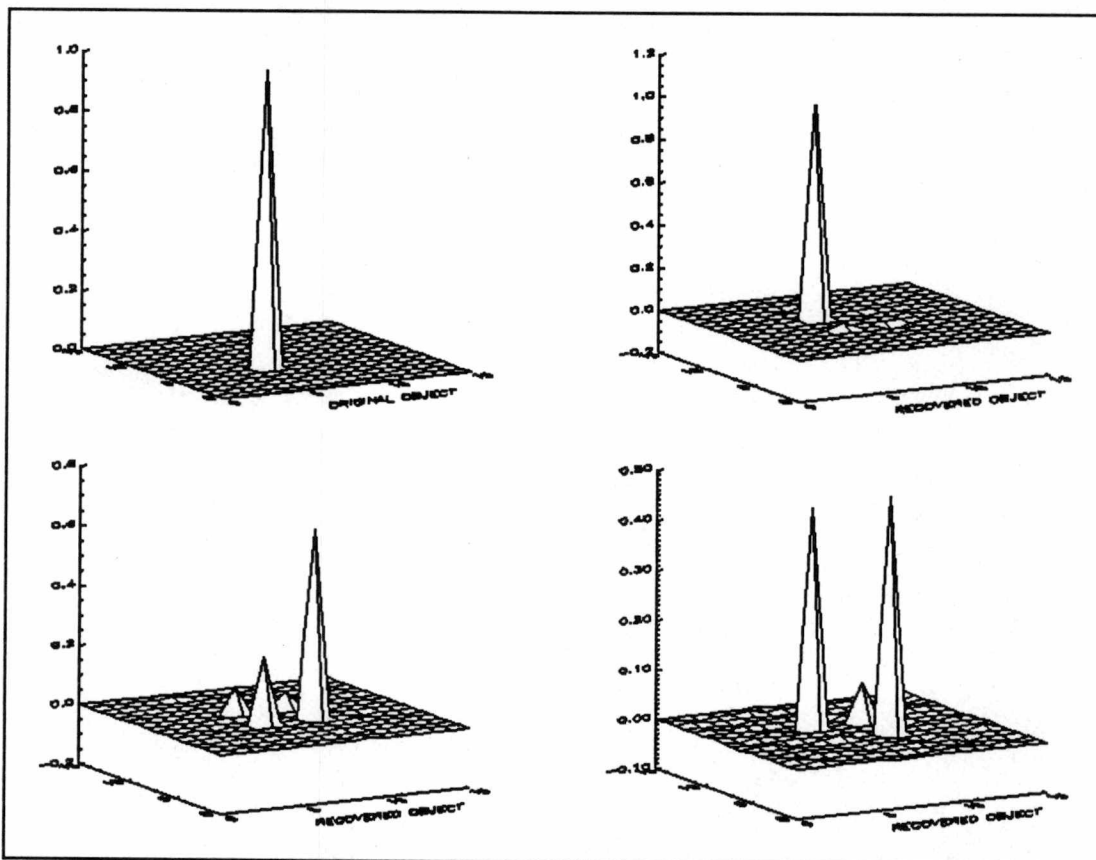


Figure 6.3 - Test One Training Object Image (6,6) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

STAR POSITION CHART

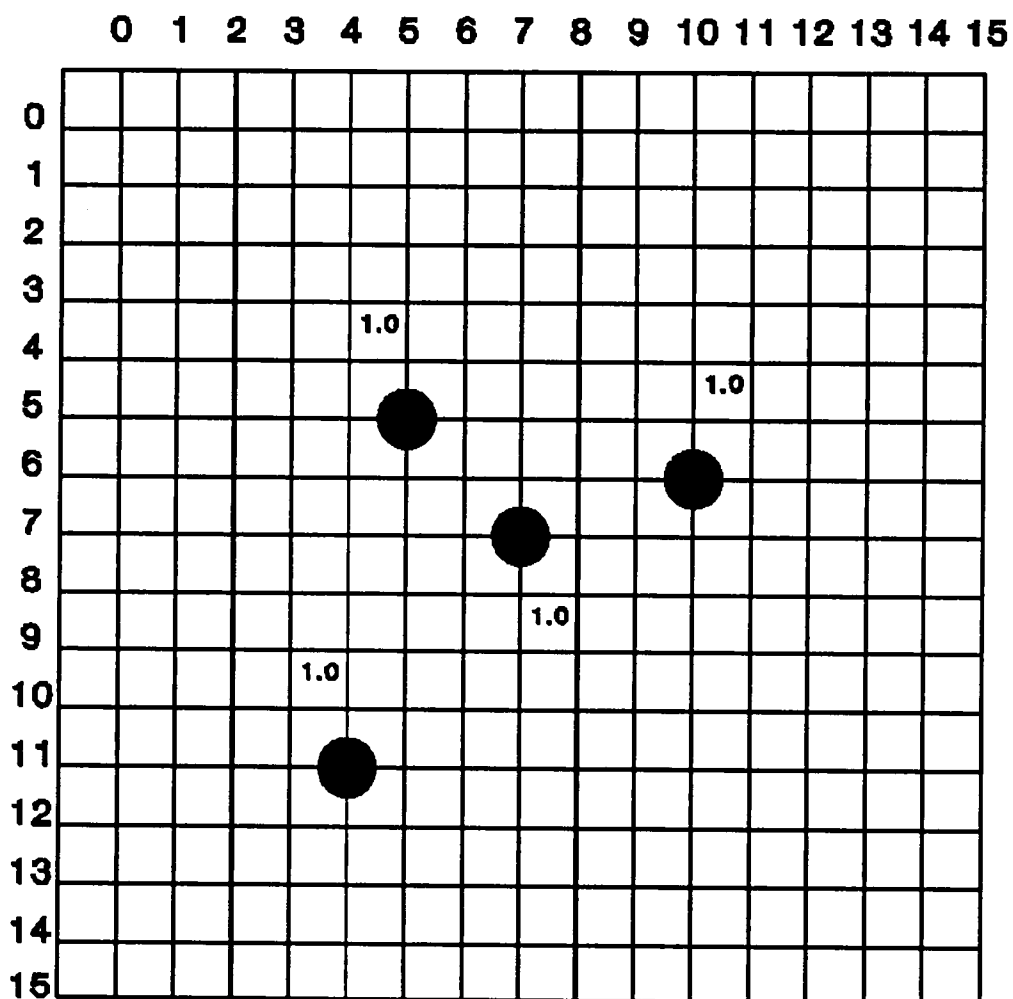


Figure 6.4 - Test One Random Placement of Single Stars. Single Stars of Same Intensity as Figure 6.1 but Randomly placed in the 64 x 64 Aperture.

same intensity.

The same process was followed again with the same results as depicted in Figure 6.2 and 6.3. Next a binary system as depicted in Figure 6.5 was trained. The only difference between these two binary sources was separation distances of the two binaries not their intensities. The net did converge as shown in Figure 6.6, but, once again, no generalization. Similar results to Figure 5.3 were obtained when recovery was attempted for either of the binary sources (i.e. it was unsuccessful).

This would suggest that translation of the single stars or alteration of separation distances in binary couples of equal intensity does not alter the frequency information in the FFT of the speckle images. This however does not agree with previous experimental results of Labeyrie. As noted in Chapter 1, the auto-correlation of a binary yields the relative separation distances of the binary and should be detectable.

TEST TWO SET UP AND RESULTS

Since test one failed to yield any useful result, the stellar objects intensities were altered slightly as shown in Figure 6.7.

The net was again trained with control data set readings taken as depicted in Figure 6.8.

The network converged to a maximum error of .01 in 590 cycles. The control RMS followed the training RMS to

STAR POSITION CHART

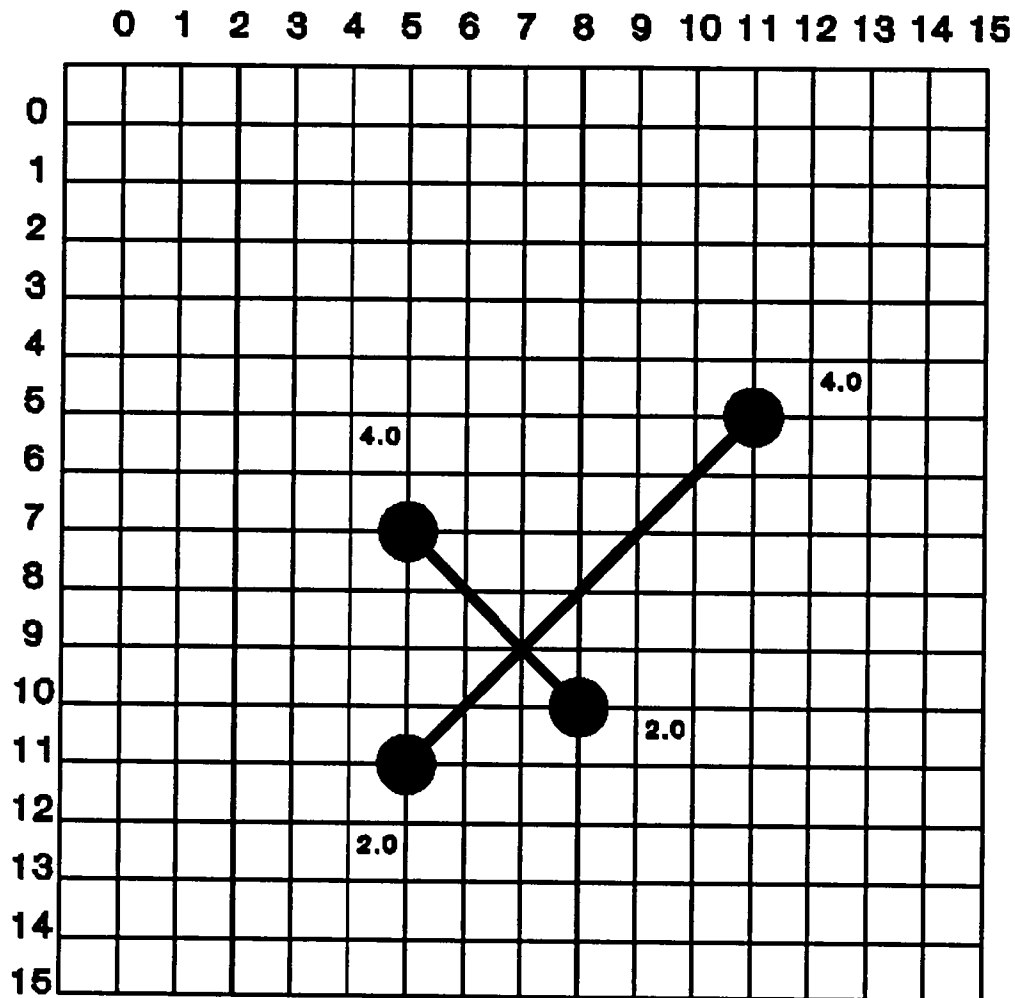


Figure 6.5 - Binary Training Star Simulation. Binary Stars of Same Intensity but Separation and Orientation Differ.

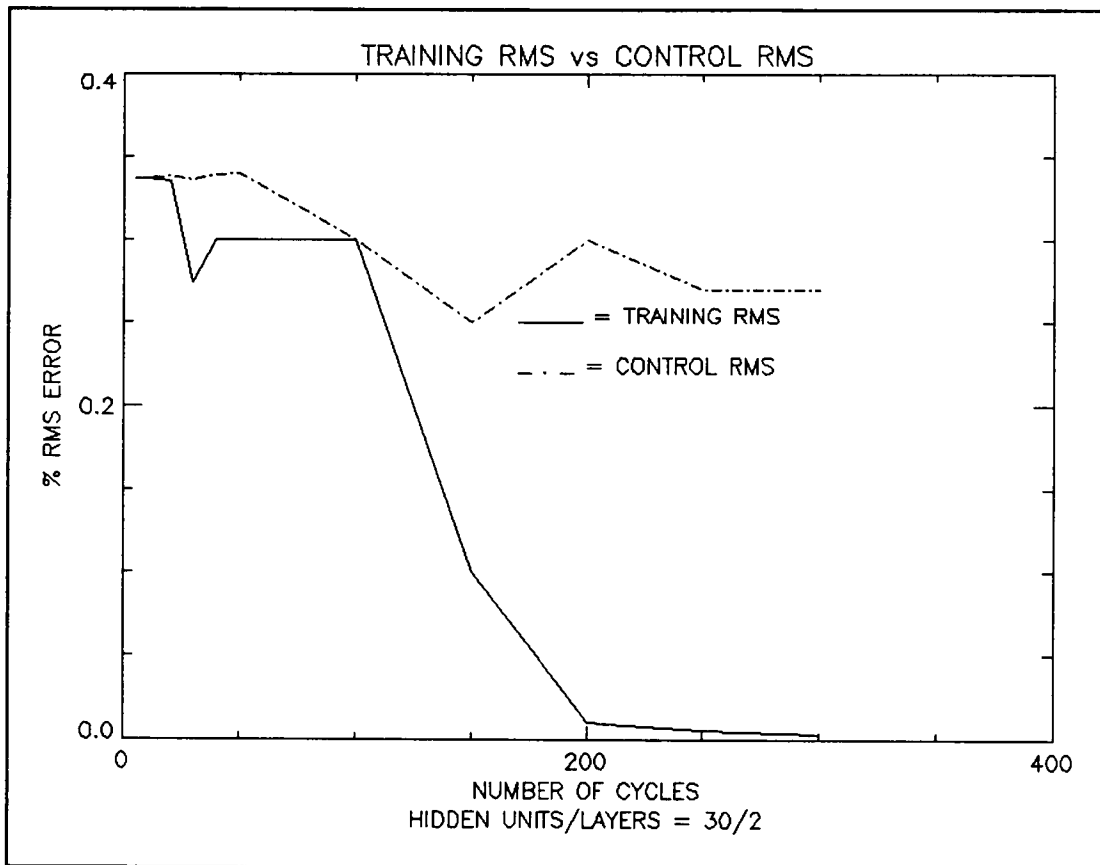


Figure 6.6 - Binary Stars Training RMS versus Control RMS.

STAR POSITION CHART

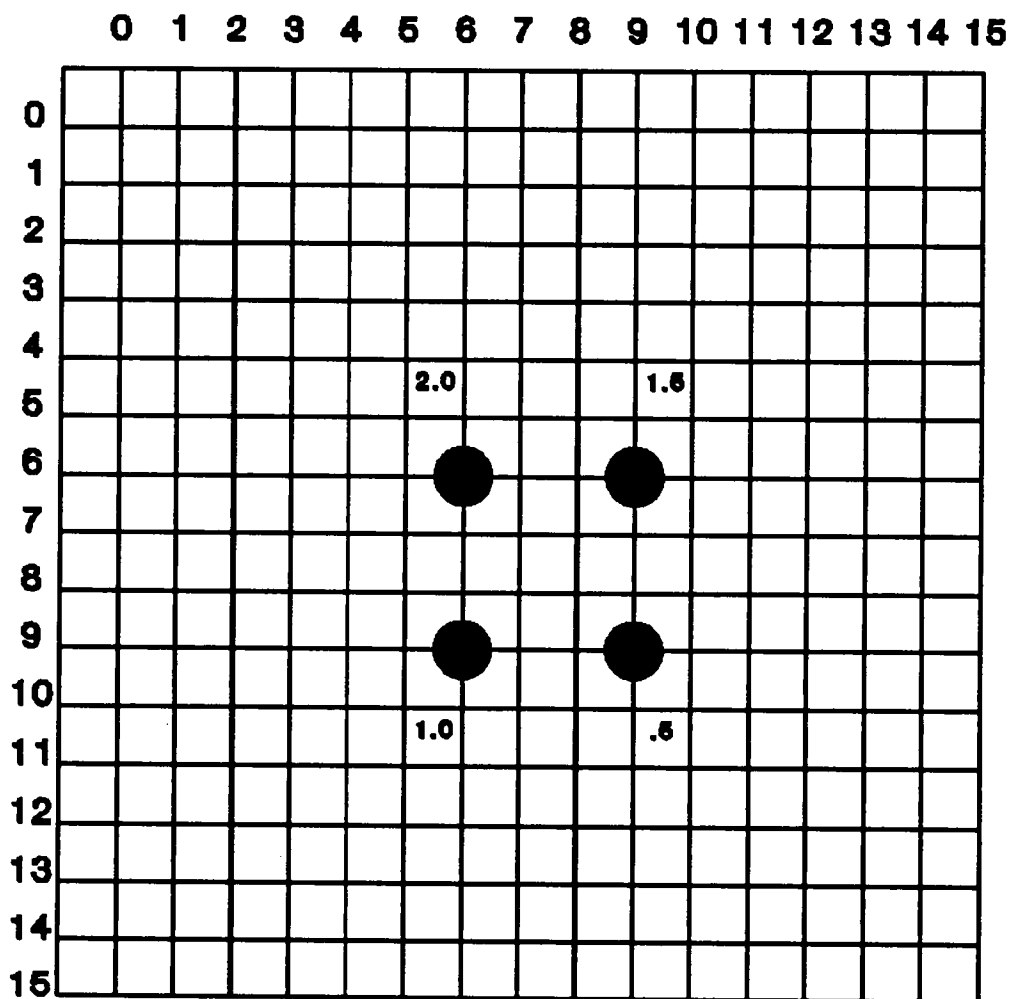


Figure 6.7 - Test Two Single Star Simulations.

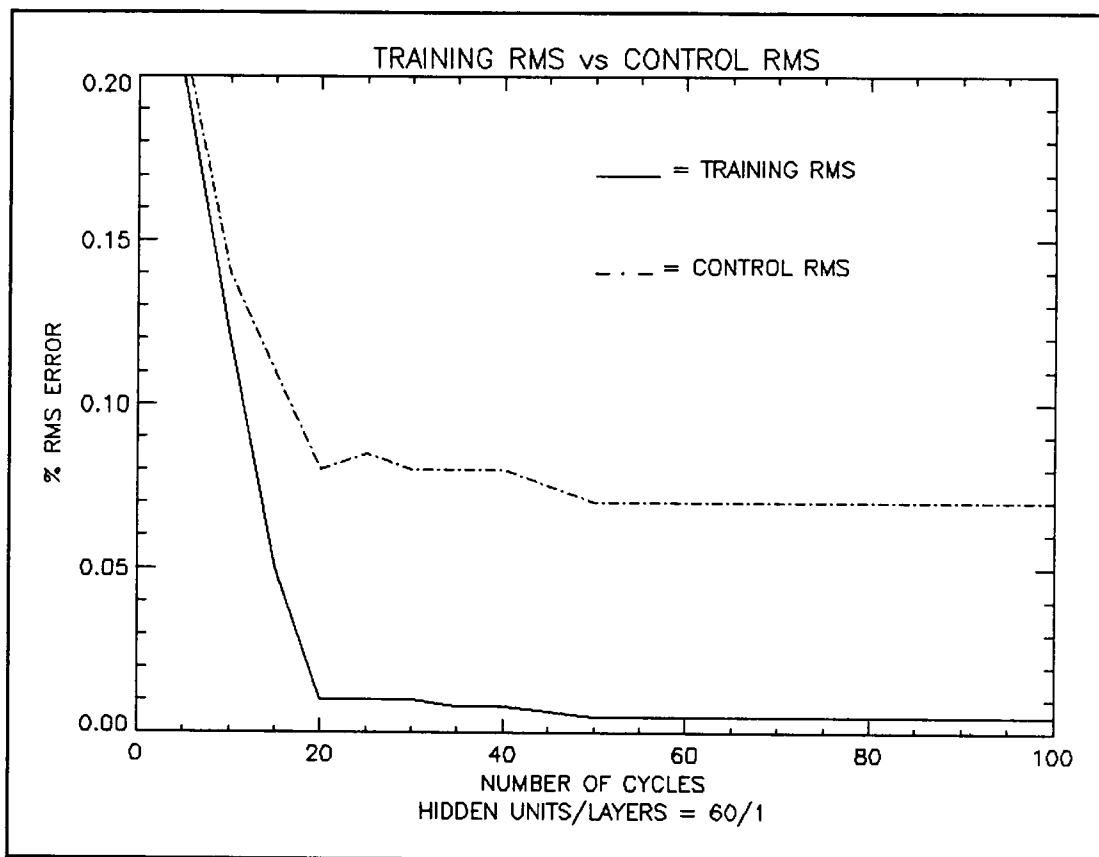


Figure 6.8 - Test Two Single Hidden Layer Training RMS versus Control RMS.

approximately .06 RMS at which time it leveled off. The recovery in this case was much better as seen in Figures 6.9, 6.10, 6.11, and 6.12.

The net was then retrained on the two hidden layer and three hidden layer configurations (Figures 5.2 and 5.3), with control data readings being taken to access generalization capability. See Figures 6.13 and 6.14.

In addition to the four point sources, two binary test images with three speckle patterns each, were propagated through the trained net. The network had not been trained on any binary objects. It was hoped that some generalization ability would be detected from the model of the four single varying intensity point sources. Figures 6.15 and 6.16 indicate that this was unsuccessful. The network was not able to recover the original objects for these two test cases.

Since some degree of recovery took place for speckle generated by the training objects, an atmospheric alteration to the speckle was then generated by the same training objects. This was accomplished by decreasing the seeing of the turbulent atmosphere model depicted in Figure 3.1. To decrease the seeing, the radius of the dispersing cells was increased from a relative cell radius of 70 to a radius of 90. The new atmosphere is depicted in picture 6.17. As can be seen in Figure 6.18 and 6.19, the alteration of the atmosphere to a less turbulent state had no real impact on the network recovery capability.

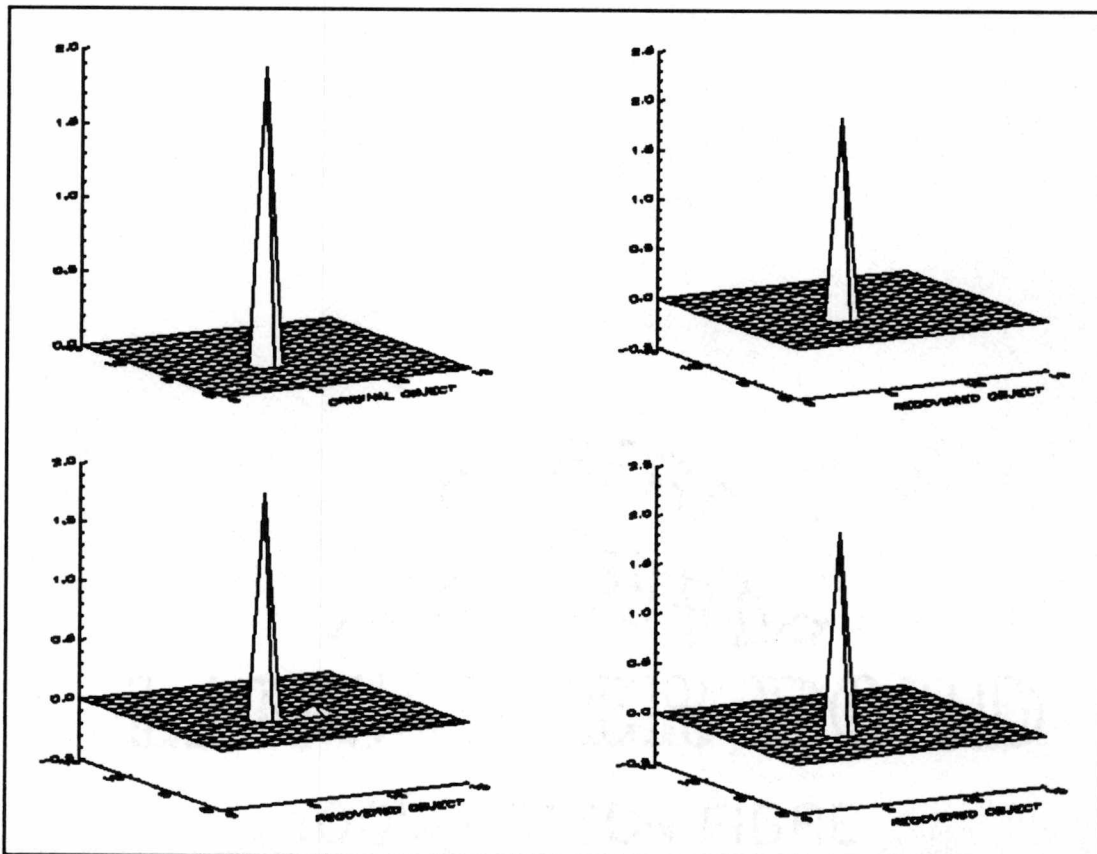


Figure 6.9 - Test Two Training Object Image (6,6) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

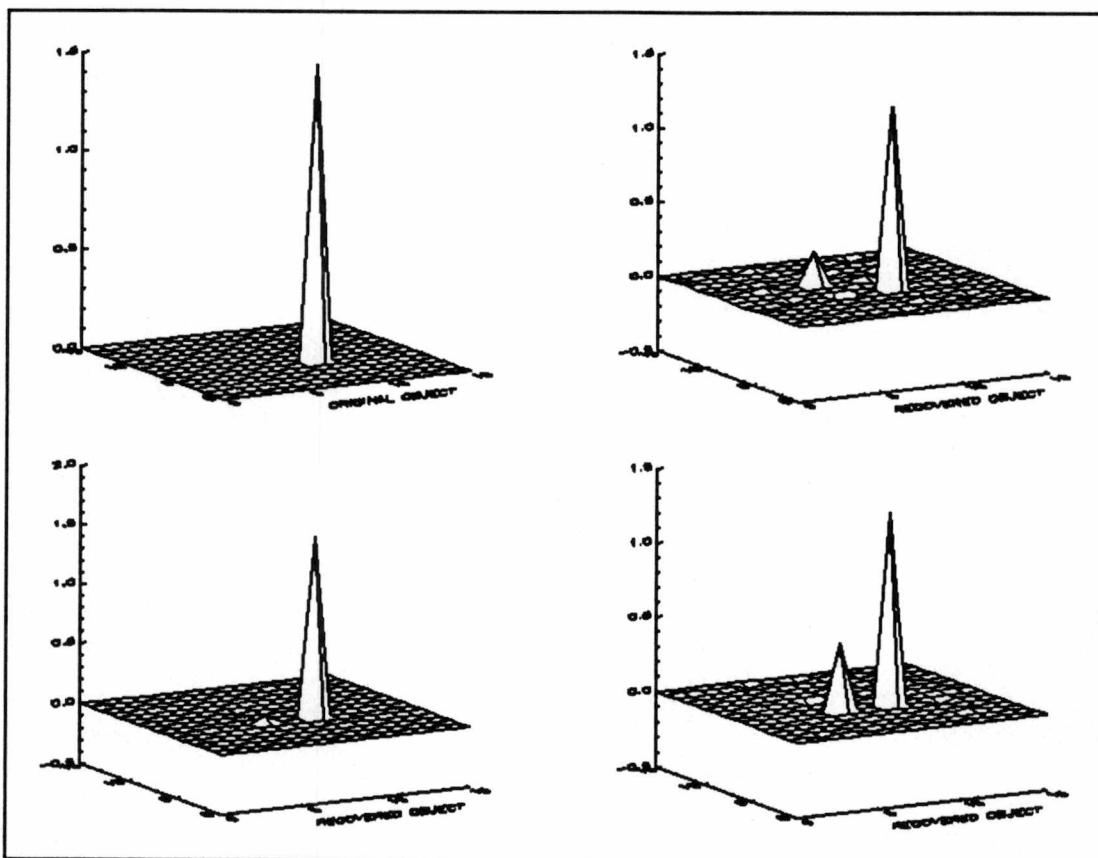


Figure 6.10 - Test Two Training Object Image (9,6) and Recovery. Original Plot 1 - Recovered Plot 2-4.

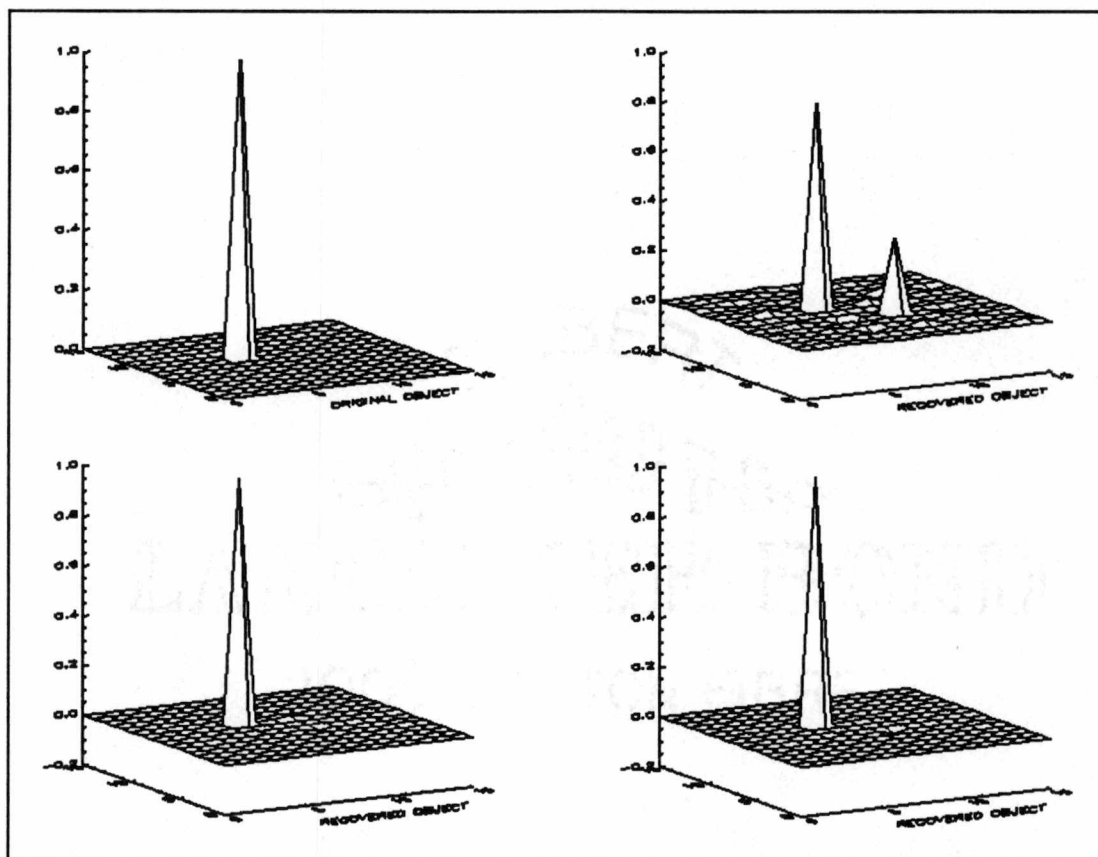


Figure 6.11 - Test Two Training Object Image (6,9) and Recovery. Original Plot 1 - Recovered Plot 2-4.

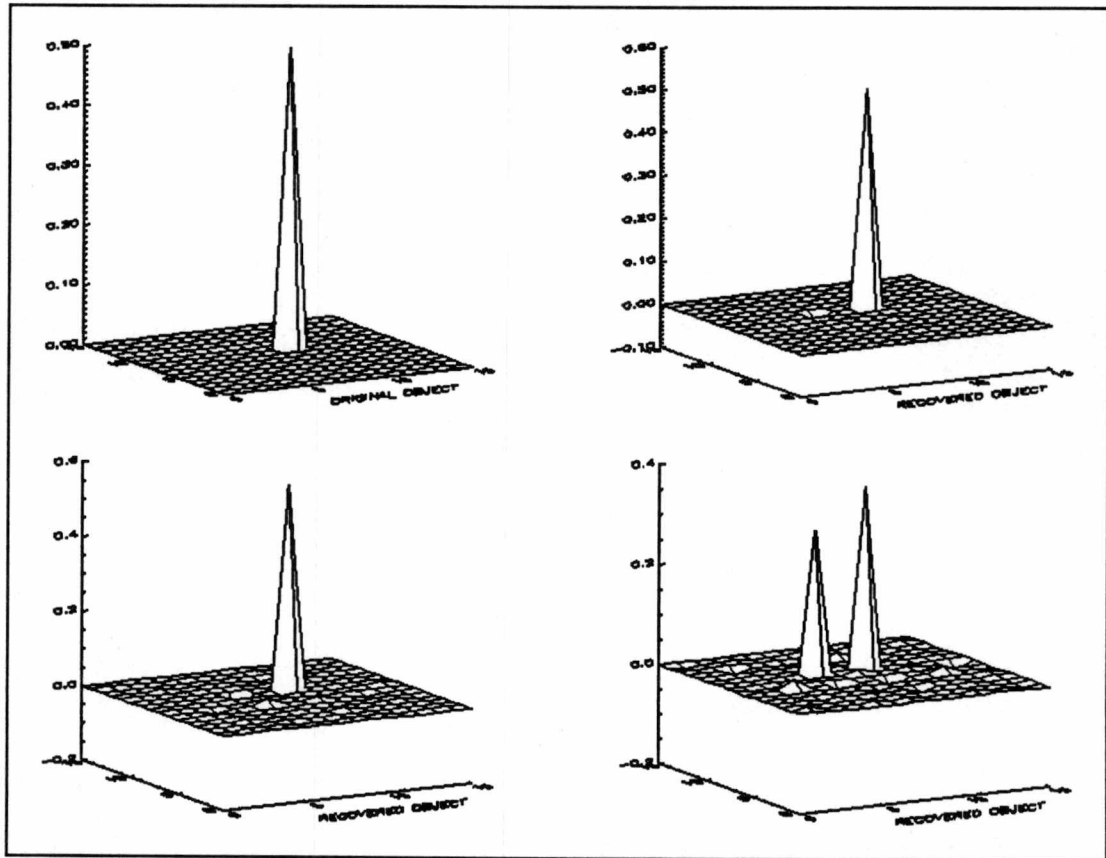


Figure 6.12 - Test Two Training Object Image (9,9) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

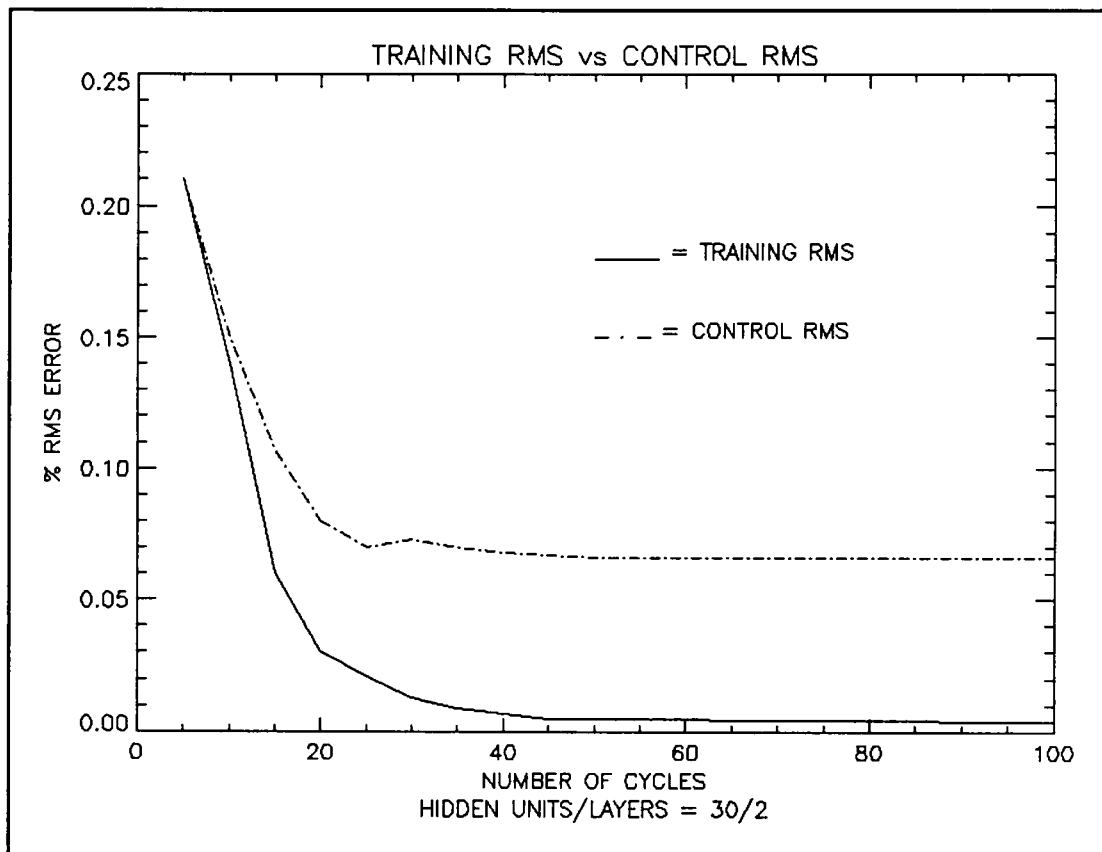


Figure 6.13 - Test Two Double Hidden Layer Training RMS versus Control RMS. Training Stars depicted in Figure 6.7.

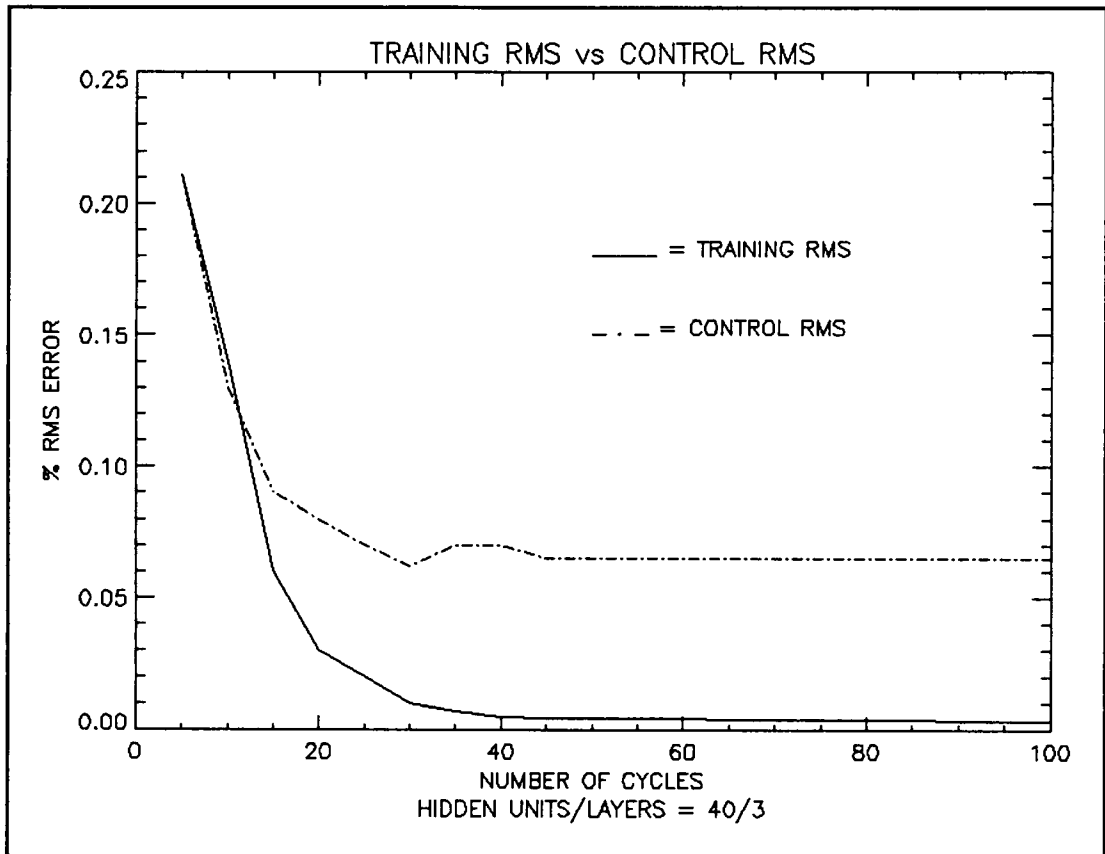


Figure 6.14 - Test Two Triple Hidden Layer Training RMS versus Control RMS.

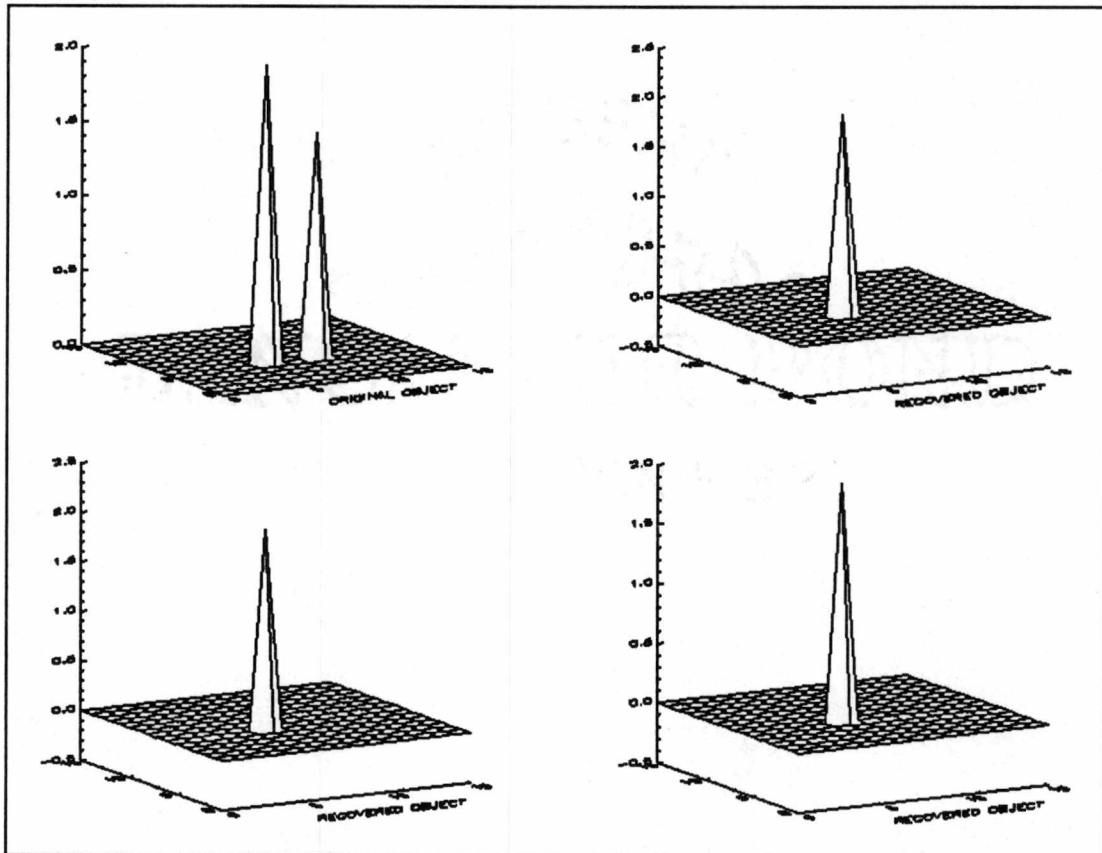


Figure 6.15 - Test Two Binary Test Object $[(6,6), (9,6)]$ and Recovery. Recovery was of the single star most closely matching the maximum intensity star of the binary couple.

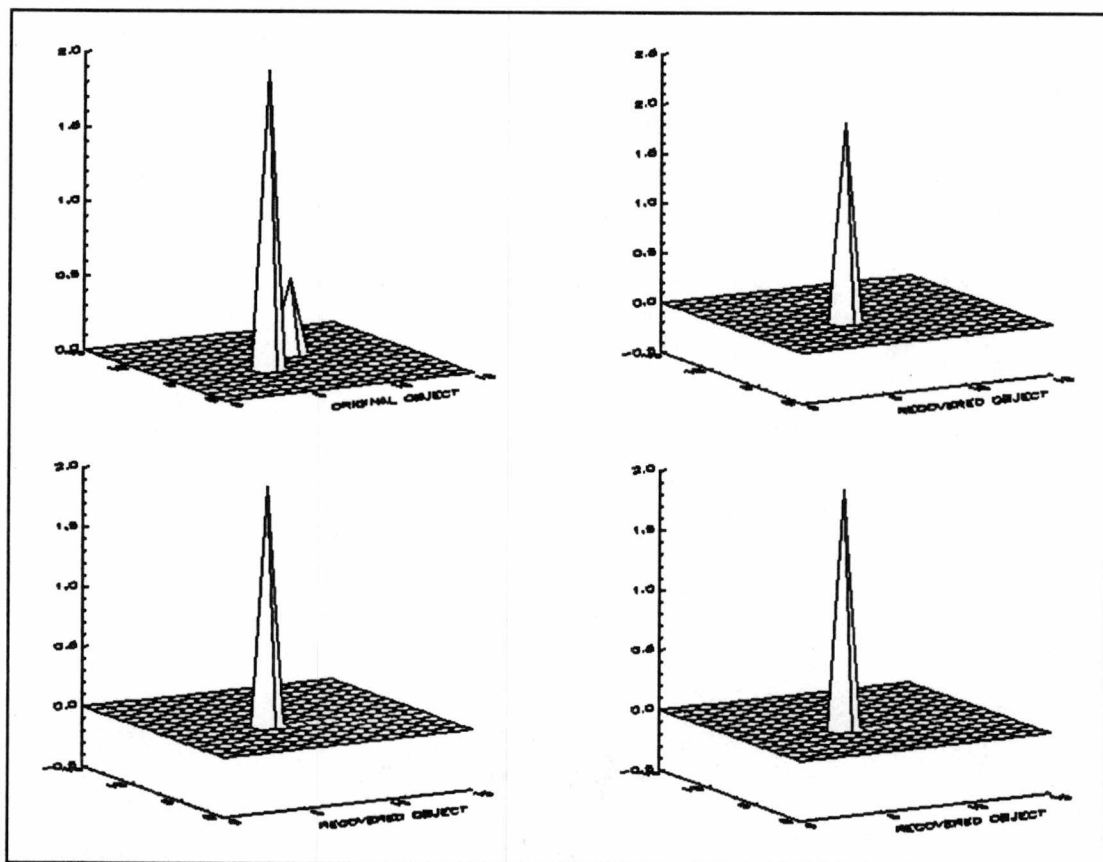


Figure 6.16 - Test Two Binary Test Object [(6,6), (9,9)] and Recovery. Recovery was of the trained single star most closely matching the maximum intensity star of the binary couple.

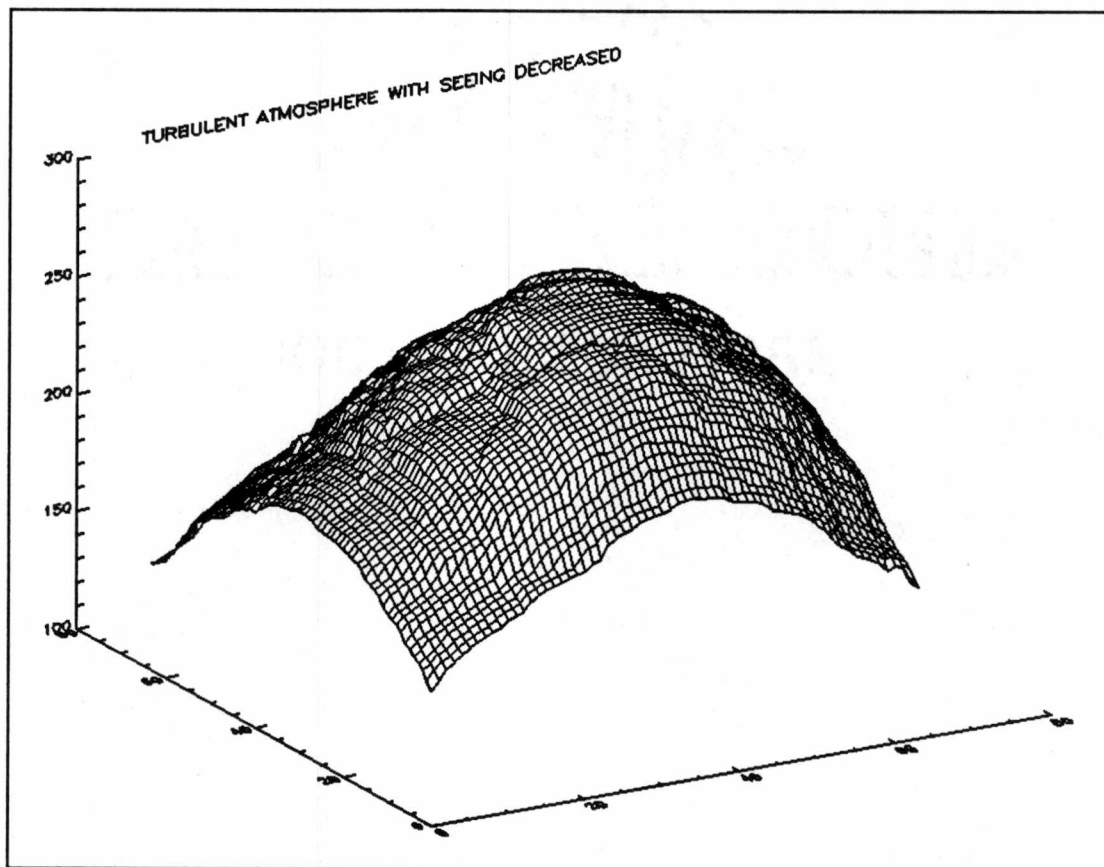


Figure 6.17 - Surface Plot of Decreased Turbulent Atmosphere Array.

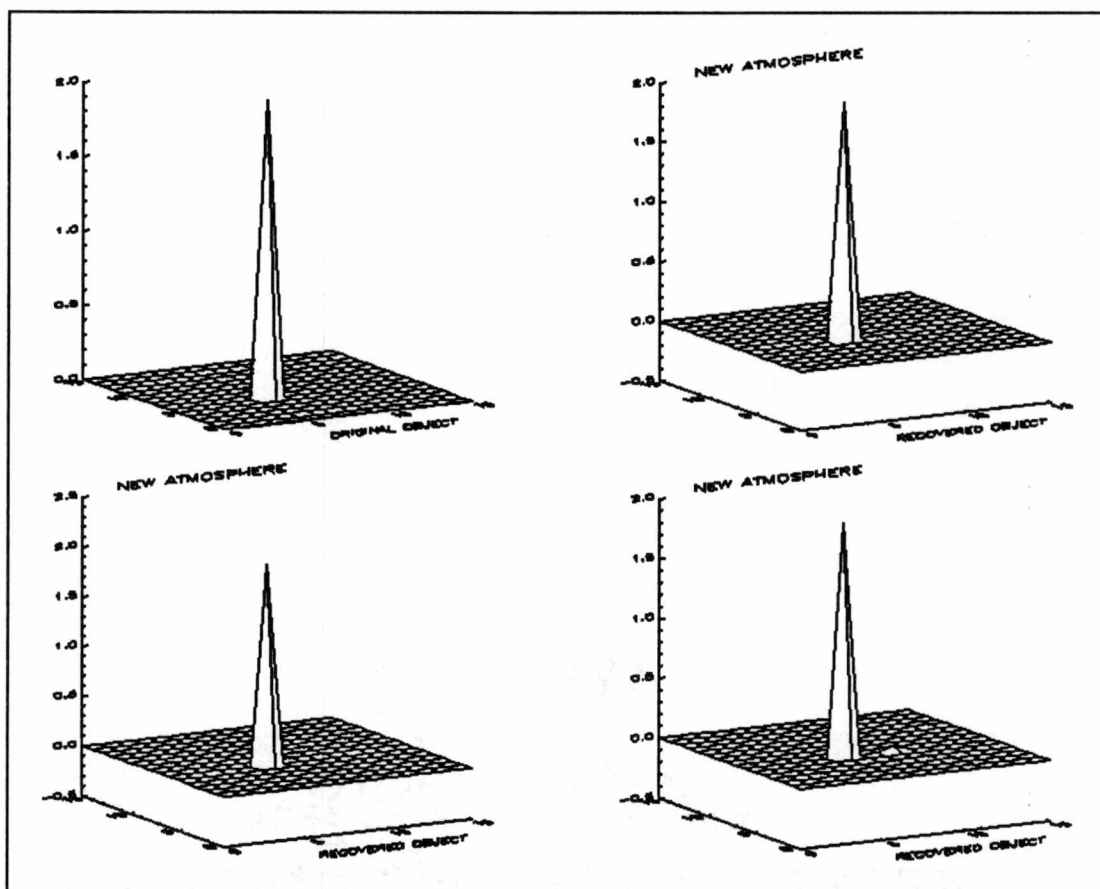


Figure 6.18 - Test Two Training Object (6,6) and Recovery with Speckle From Less Turbulent Atmosphere. Plot 1 Original Object Image Trained by Net - Plot 2-4 Recovered Object Image with New Atmosphere.

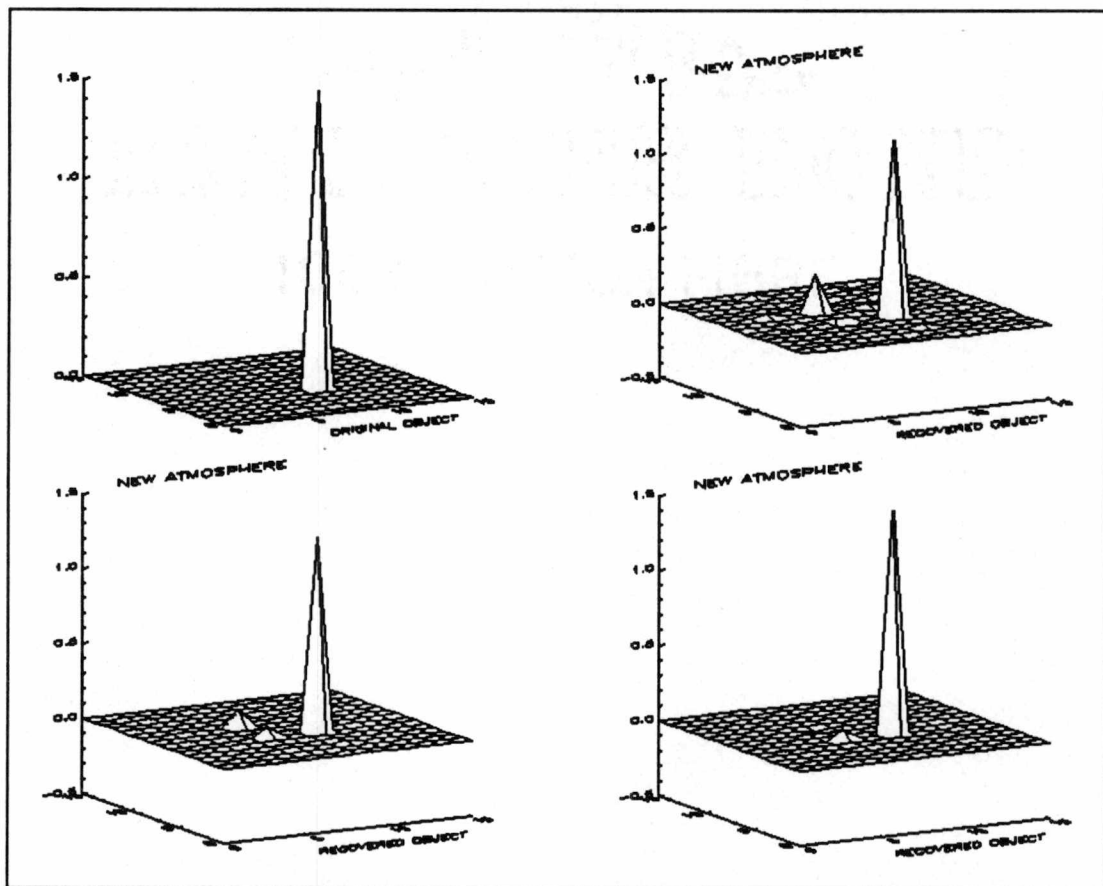


Figure 6.19 - Test Two Training Object (9,6) and Recovery with Speckle from Less Turbulent Atmosphere. Plot 1 Original Object Image trained by Net - Plot 2-4 Recovered Object Images with New Atmosphere.

Next, the point sources of varying intensity were moved to new locations in the aperture field to again preclude any possibility that the symmetry in Figure 6.7 would degrade the recovery capability of the network. This is depicted in Figure 6.20.

The two hidden layer configuration as depicted in Figure 5.2 was utilized for this test with some improvement in control RMS convergence. In addition, the single layer hidden configuration, Figure 4.1 was tried but again no improvement was noted over the two hidden layer configuration. See Figures 6.21 and 6.22.

The single layer also converged slower and showed slightly less generalization. In this recovery attempt, the two hidden layer architecture was used for the actual object recovery. These are shown in Figures 6.23, 6.24, 6.25, and 6.26.

The recovery of a single test object and a binary test object by their speckle, not previously trained, was attempted but again the network shows a single star which had the nearest matching intensity. These recovery attempts are shown in Figure 6.27 and 6.28.

TEST THREE SET UP AND RESULTS

In this simulation, two different star configurations were utilized. One single star was placed at position (9,6) and a binary at (5,5), and (10,10) with relative intensities of 5.0, 3.0, and 1.0 respectively. See Figure 6.29.

STAR POSITION CHART

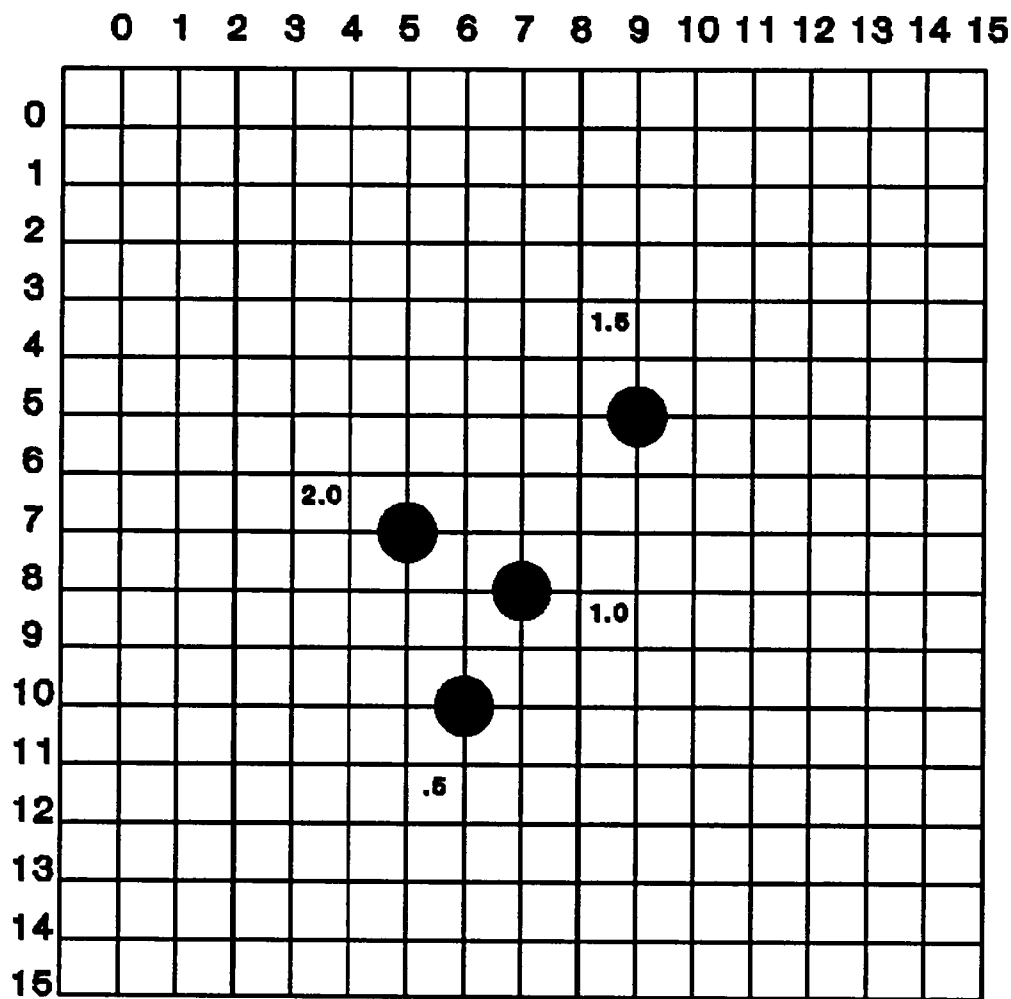


Figure 6.20 - Test Two Random Placement of Single Star Simulations. Single Star Simulations Same as Depicted in Figure 6.7 but Randomly Placed in the 64 x 64 Aperture.

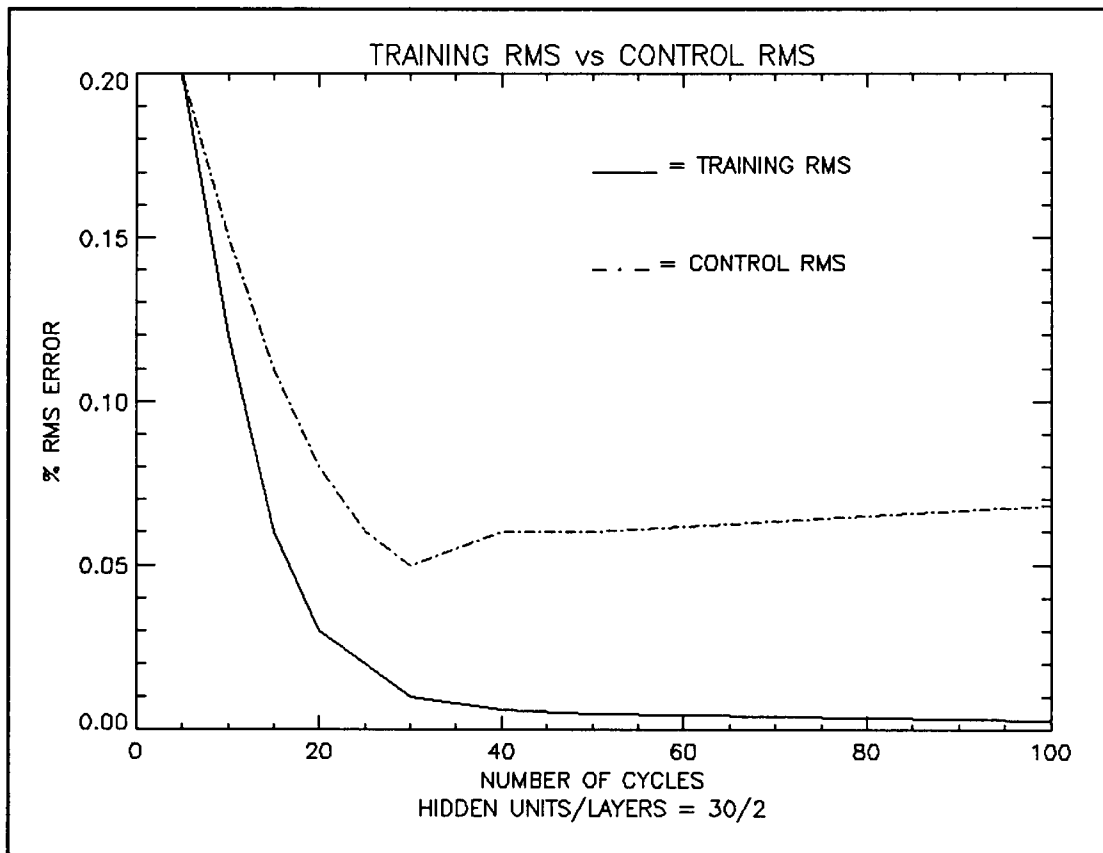


Figure 6.21 - Test Two Random Placement Double Hidden Layer Configuration Training RMS versus Control RMS.

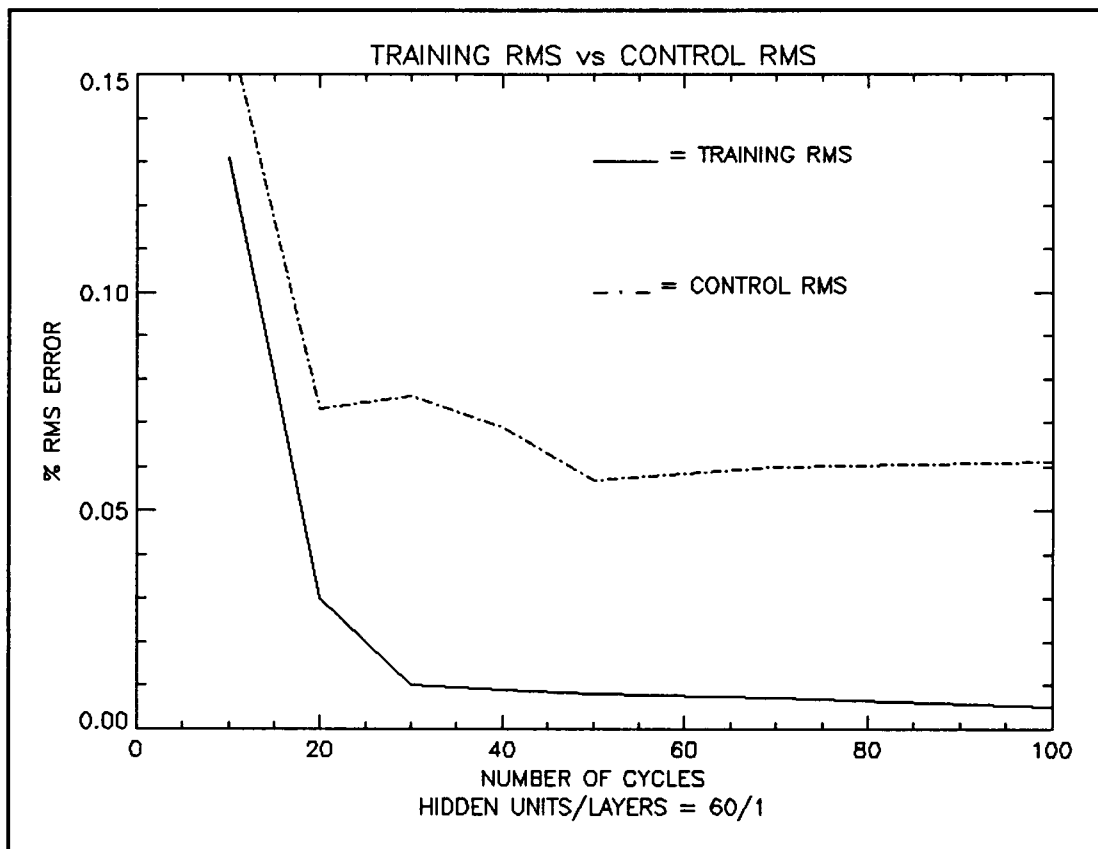


Figure 6.22 - Test Two Random Placement Single Hidden Layer Configuration Training RMS versus Control RMS. Original Object Image Plot 1 - Recovered Images Plot 2-4.

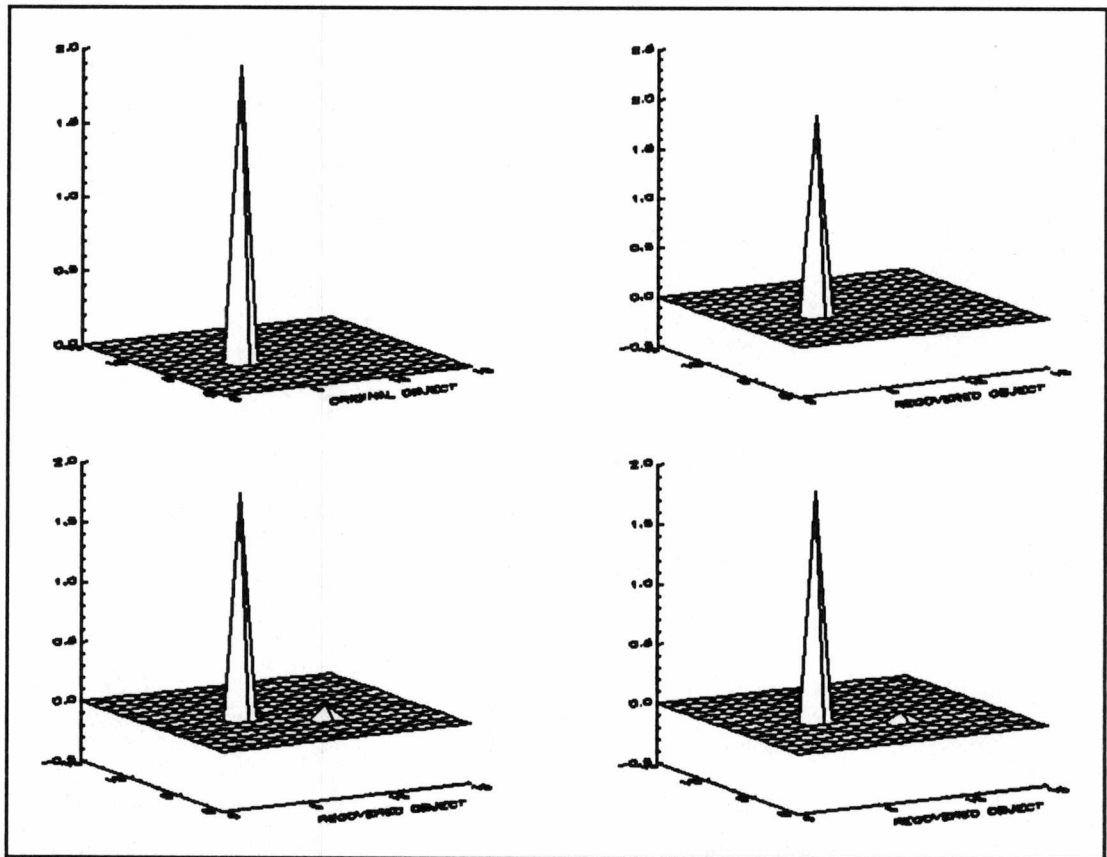


Figure 6.23 - Test Two Random Placement Training Object Image (5,7) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

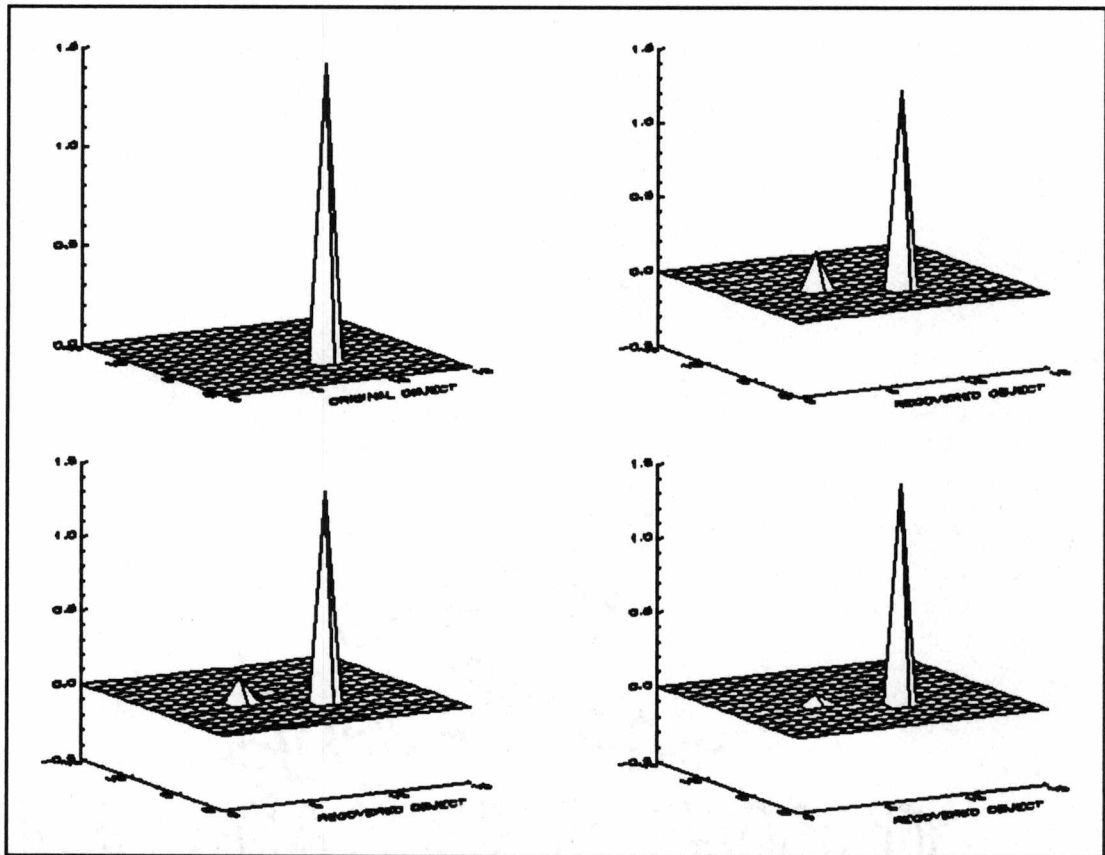


Figure 6.24 - Test Two Random Placement Training Object Image (9,5) and Recovery. Original Object Image Plot 1 - Recovered Images Plots 2-4.

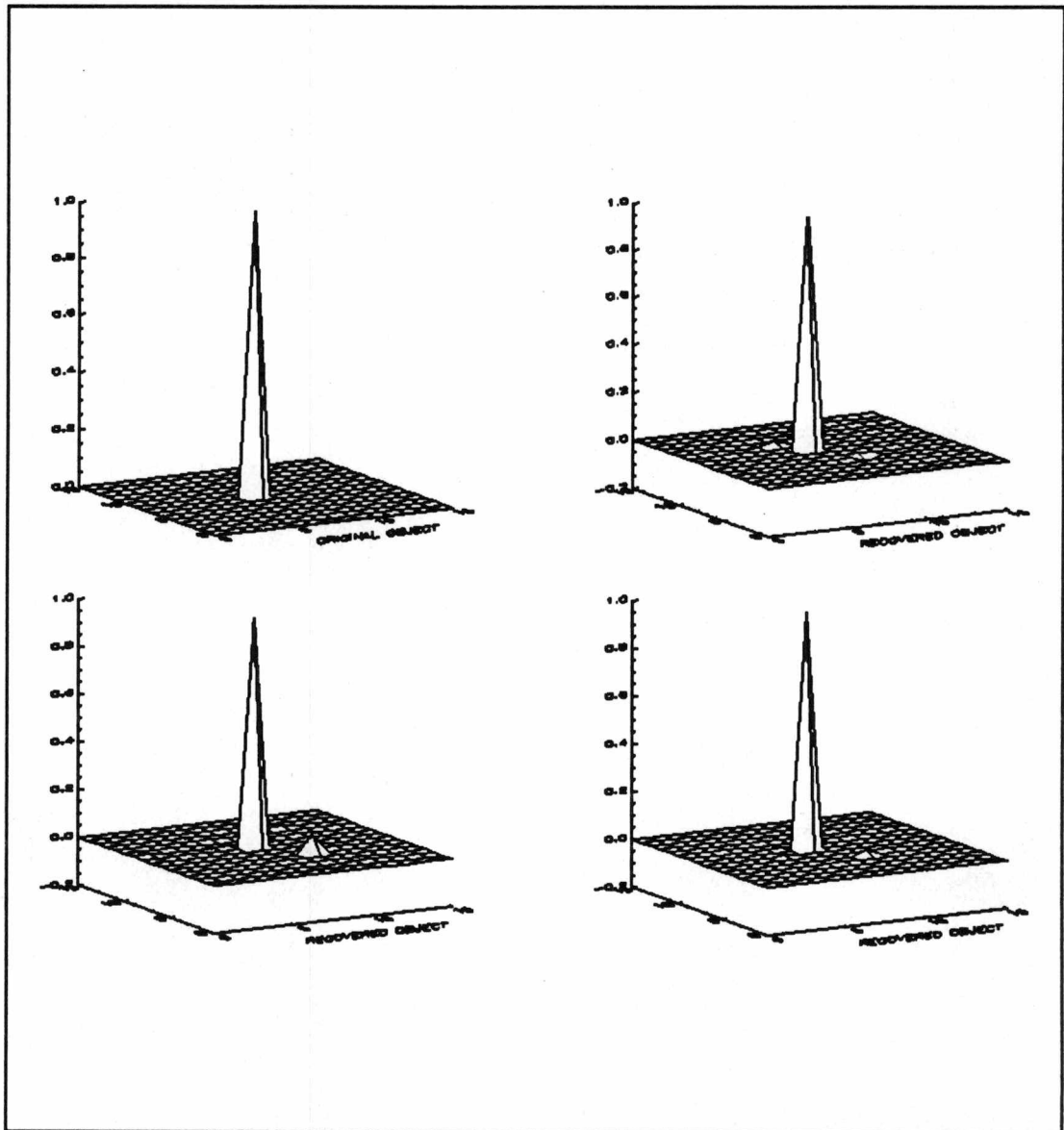


Figure 6.25 - Test Two Random Placement Training Object Image (7,8) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

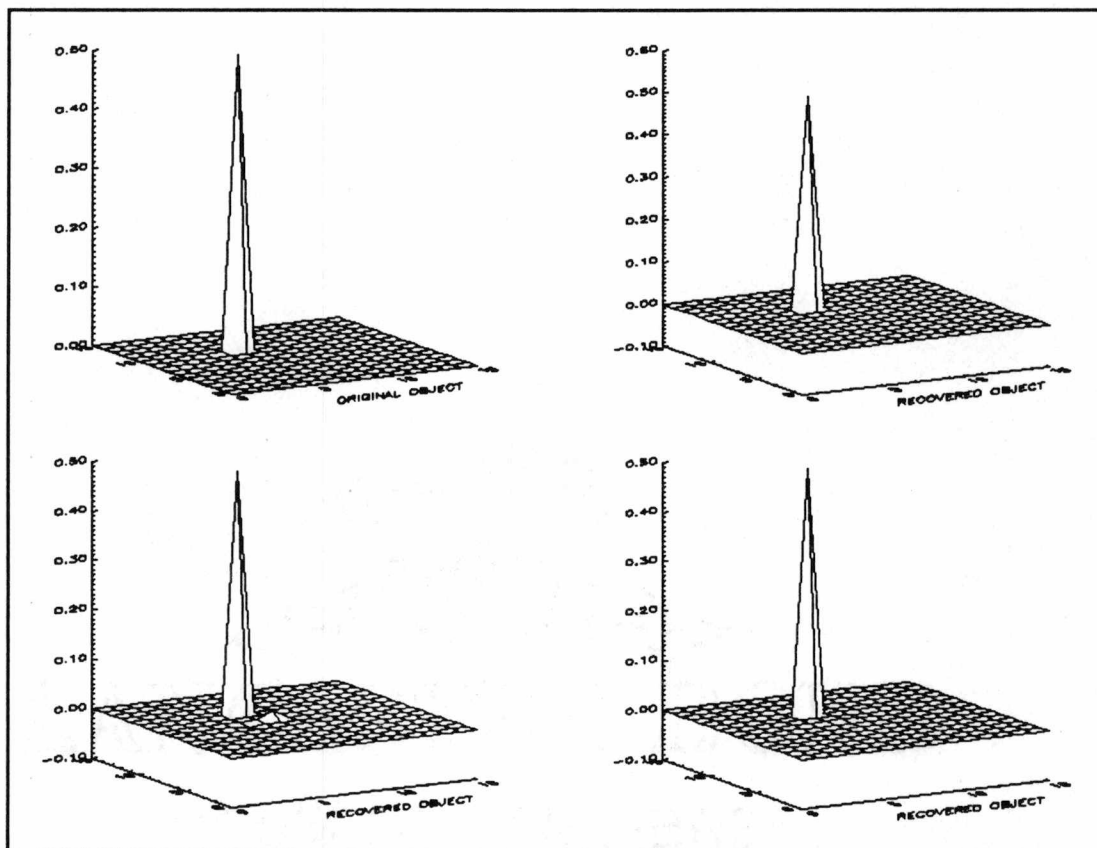


Figure 6.26 - Test Two Random Placement Training Object Image (6,10) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

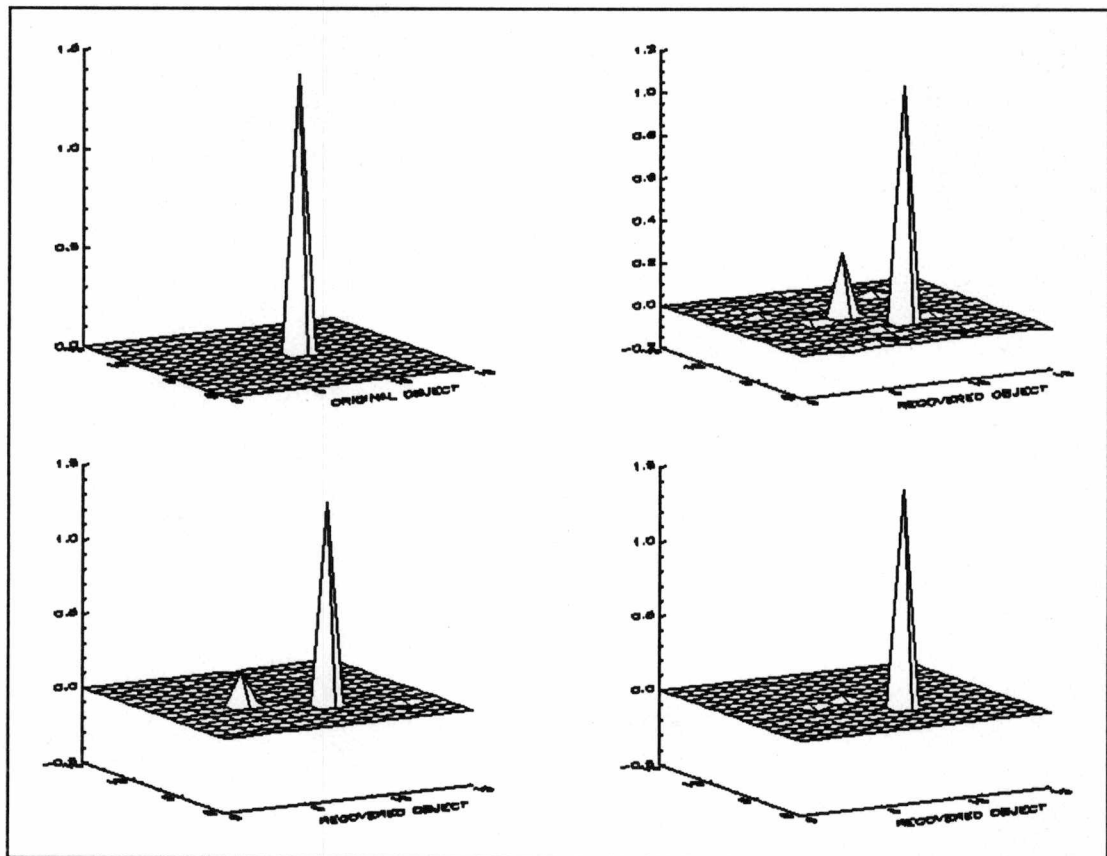


Figure 6.27 - Test Two Random Placement Trained Network with Test Object Image (9,8) and Recovery. Original Object Image Plot 1 - Recovered Images Plots 2-4.

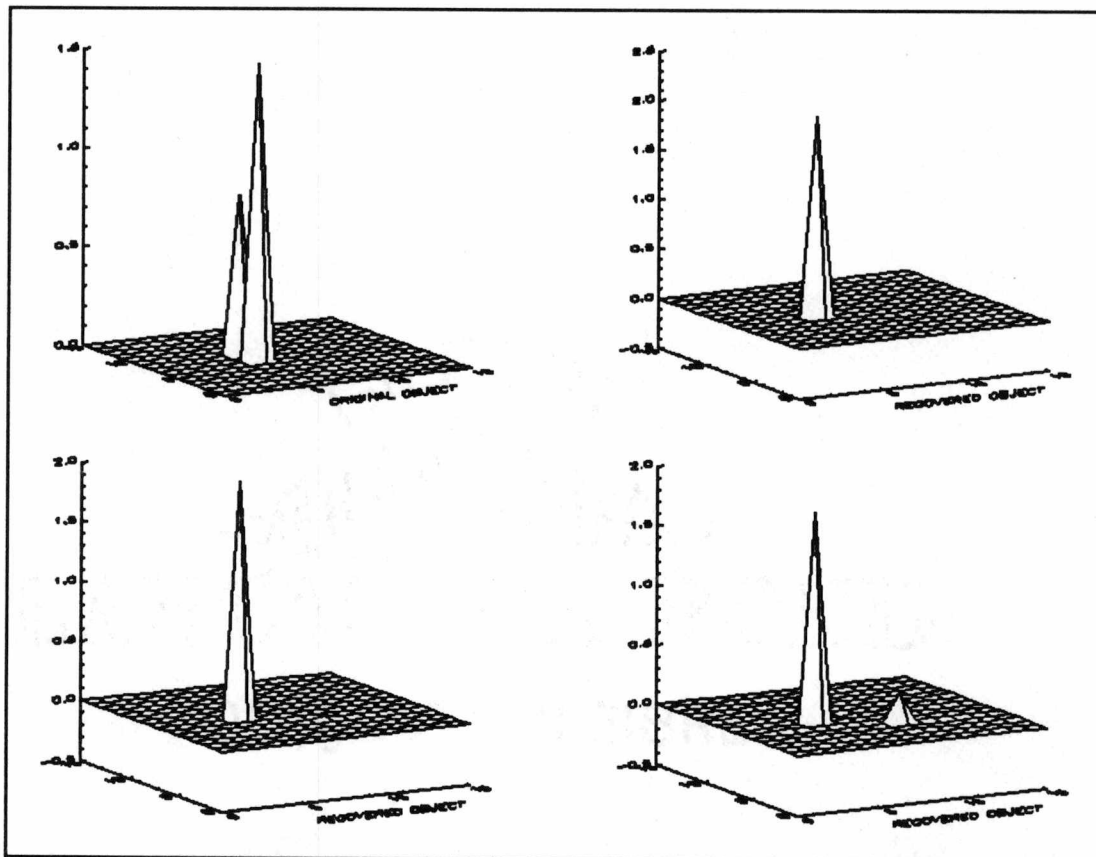


Figure 6.28 - Test Two Random Placement Trained Network with Test Object Image [Binary (6,7), (6,9)] and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

STAR POSITION CHART

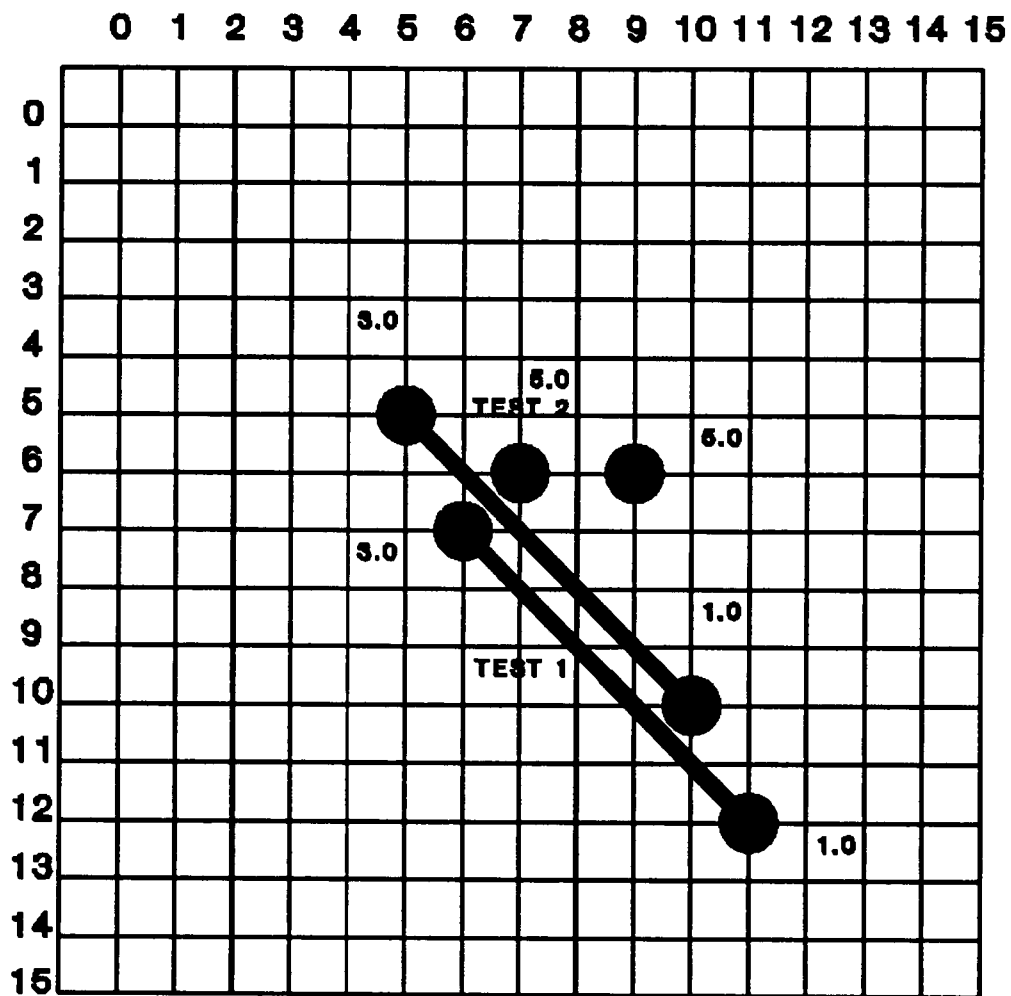


Figure 6.29 - Test Three Single and Binary Star Simulation.

This training was accomplished by utilizing a two hidden layer network. See figure 4.2. Much better results were achieved than with the single star system or binary star system. The control RMS closely followed the training RMS. This was rerun a number of times with similar results each time. See Figure 6.30.

The recovery of the training objects was much better than either of the two previous attempts. As noted previously, all recovery is with speckle not previously seen by the network. See Figures 6.31 and 6.32.

The test images this time were chosen to be translations of the training images. When the trained net was presented with the speckle image of Test Source 1 (binary), the original binary was recovered with some artifacts. (See Figure 6.33). Similarly with the single test star, Test Source 2. (See Figure 6.34).

Finally, the atmospheric turbulence or seeing was again decreased for the speckle creation routine and four test speckle sets were created and propagated through the net with stars at the same intensity and position distribution as Figure 6.29. (See Figures 6.35, 6.36, 6.37, and 6.38). The atmospheric model for this is depicted in Figure 6.17.

These last results suggest that the point source speckle is spatially invariant. Most likely, the intensity information in the FFT of the speckle is the discerning factor. This is supported by the work of Françon (1979) ³²

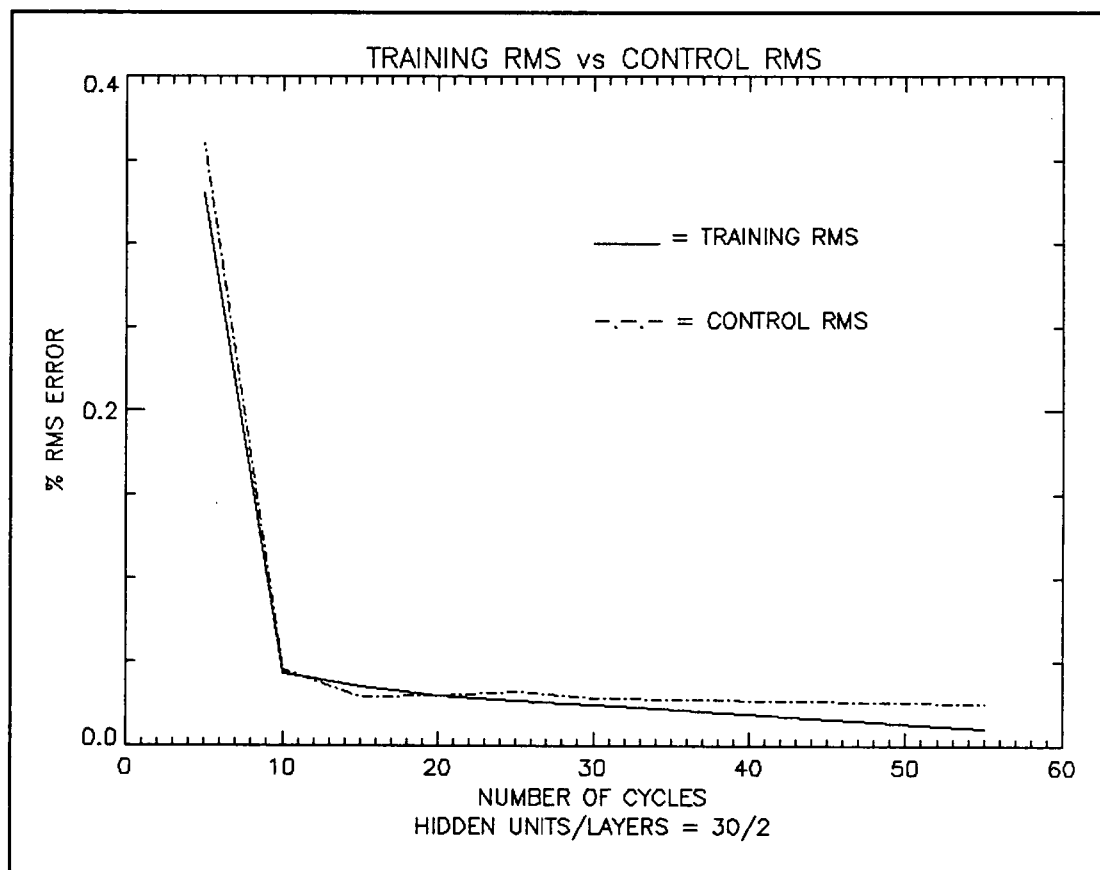


Figure 6.30 - Test Three Double Hidden Layer Training RMS versus Control RMS.

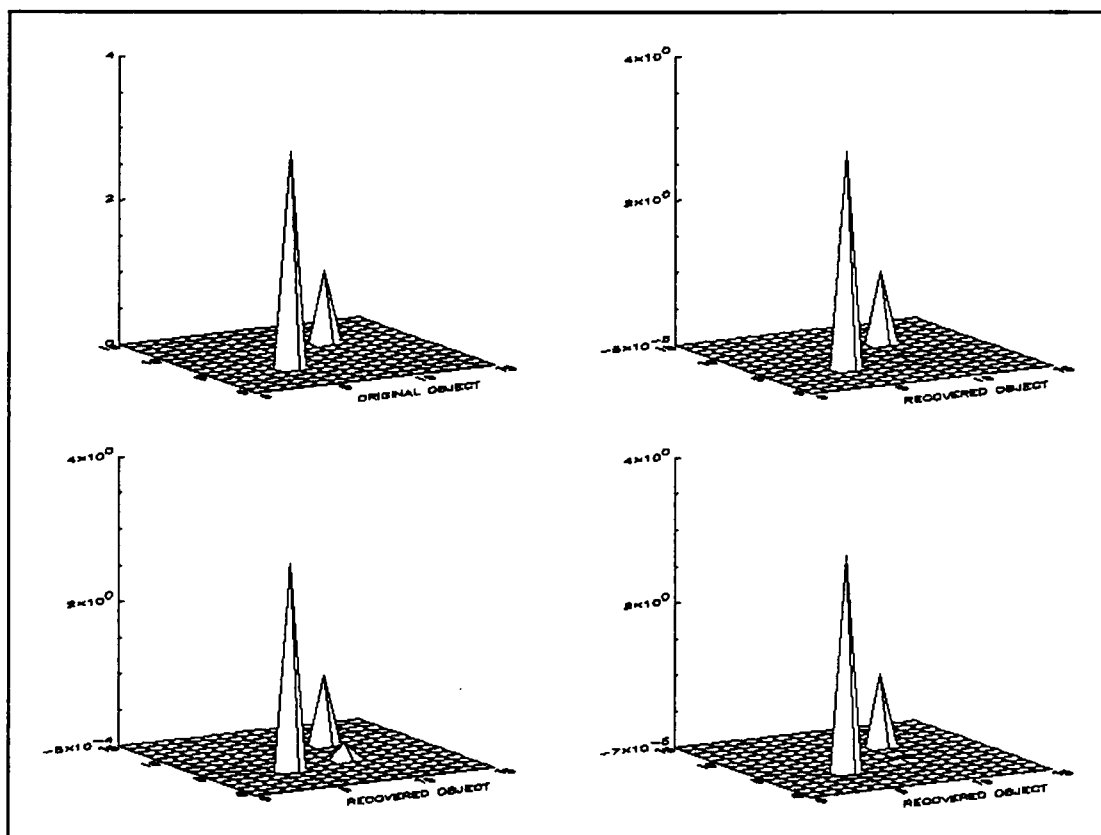


Figure 6.31 - Test Three Training Object Image [Binary (5,5), (10,10)] and Recovery. Original Object Image Plot 1 - Recovered Images Plots 2-4.

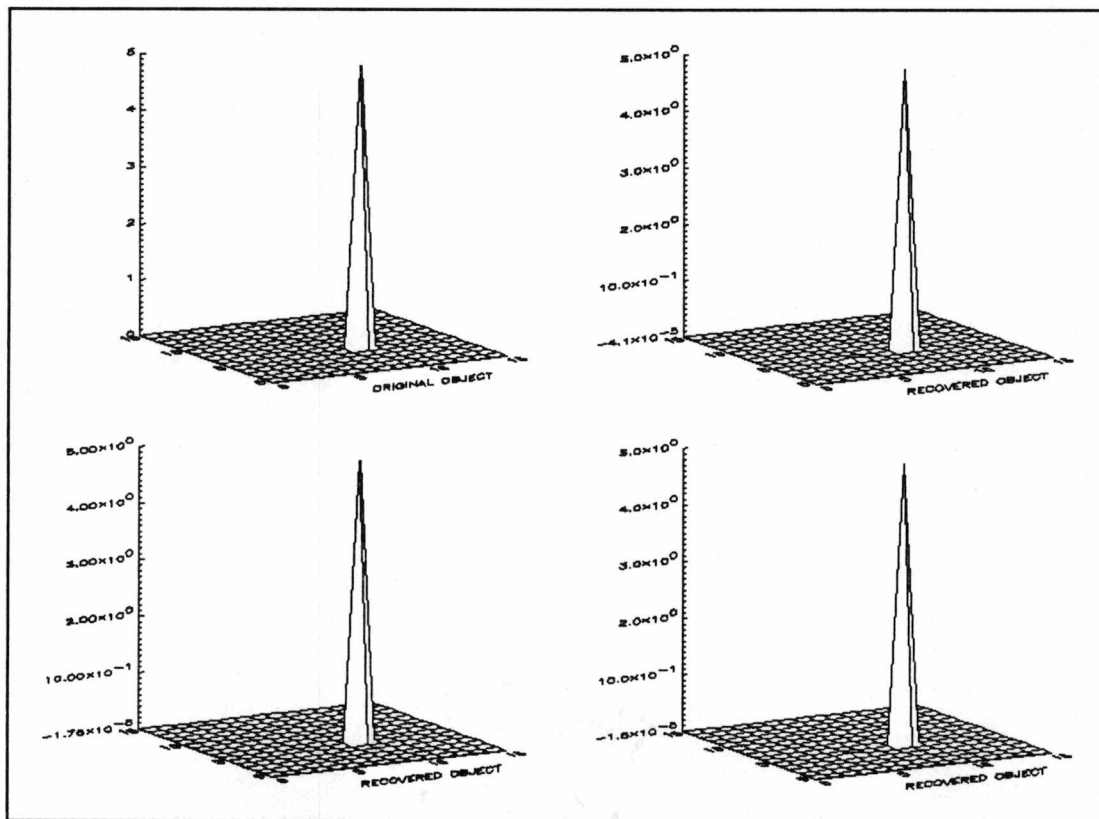


Figure 6.32 - Test Three Training Object Image (9,6) and Recovery. Original Object Image Plot 1 - Recovered Images - Plots 2-4.

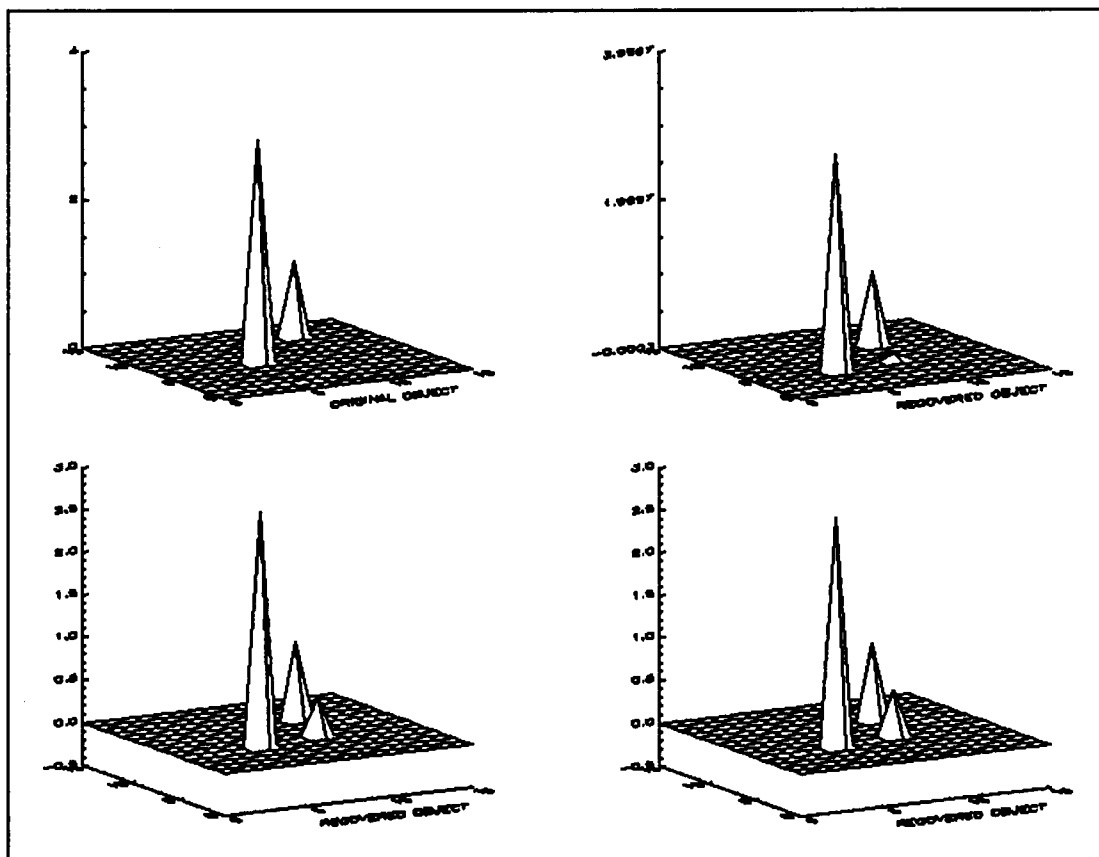


Figure 6.33 - Test Three Test 1 Object Image [Binary (6,7), (11,12)] and Recovery. Original Object Image Plot 1 - Recovered Images Plots 2-4.

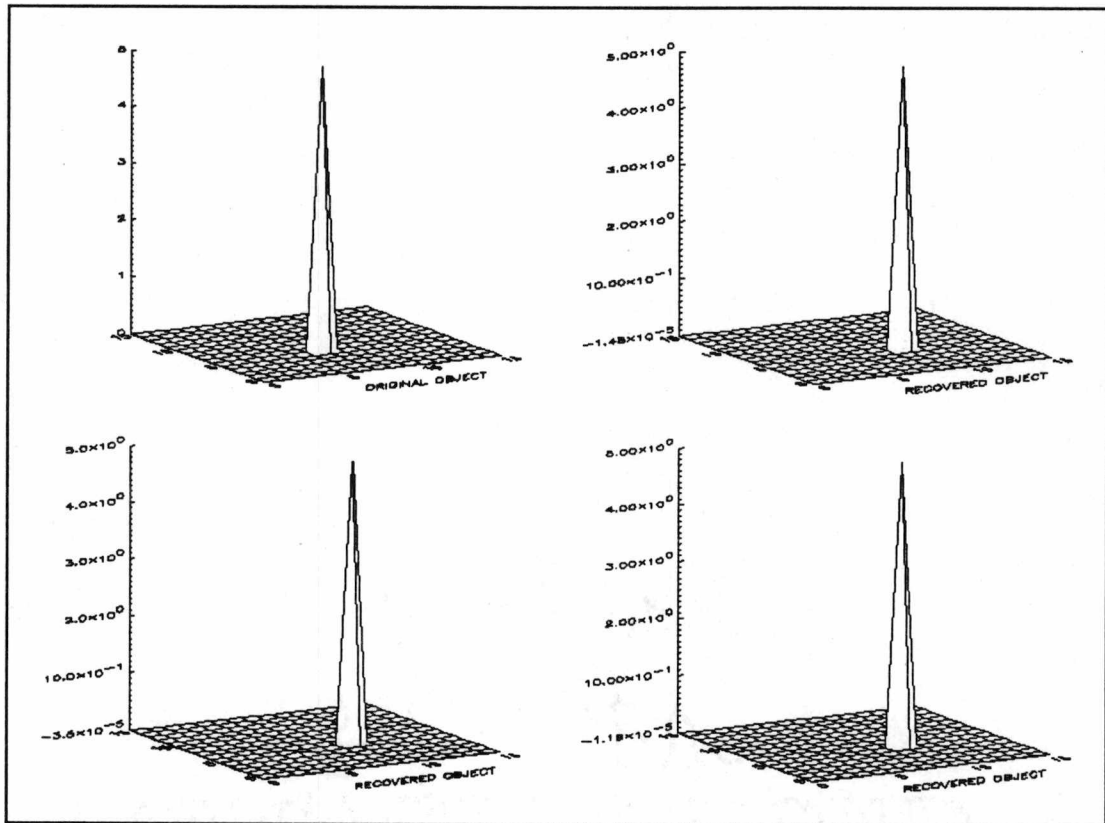


Figure 6.34 - Test Three Test 2 Object Image (7,6) and Recovery. Original Object Image Plot 1 - Recovered Images Plots 2-4.

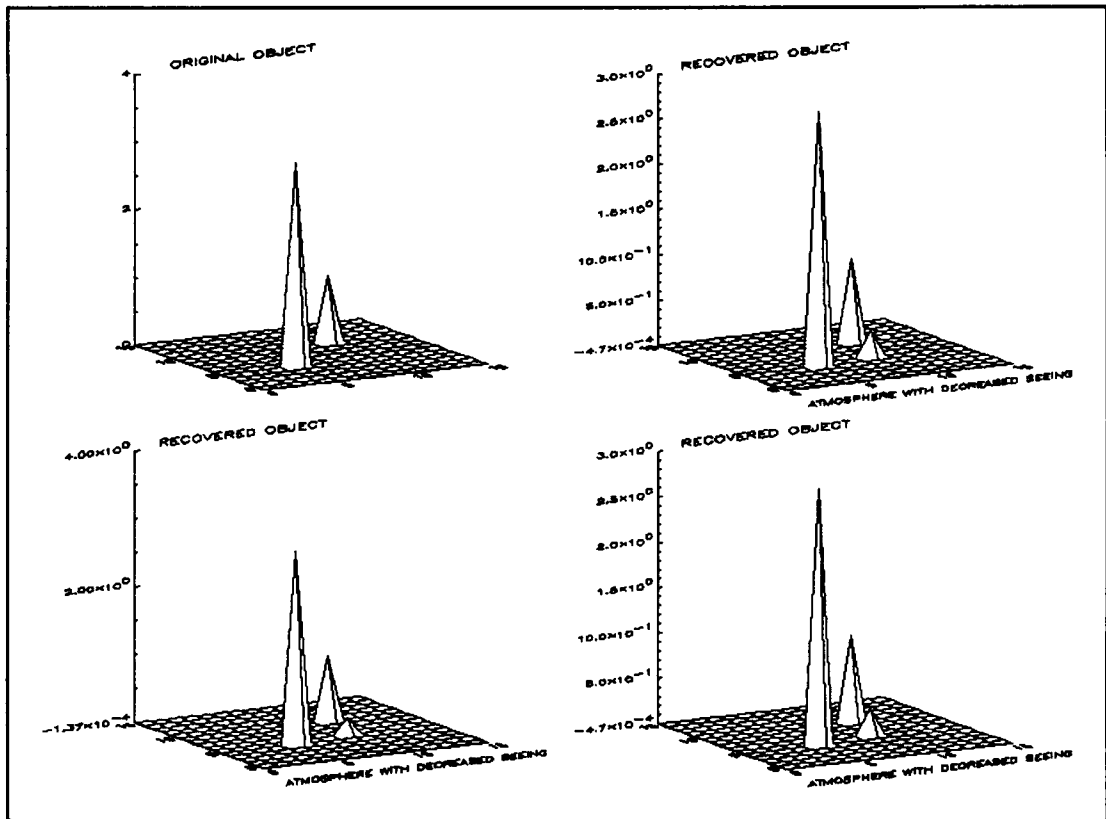


Figure 6.35 - Test Three Training Object Image [Binary (5,5), (10,10)] and Recovery with Speckle from Less Turbulent Atmosphere. Original Object Image Plot 1 - Recovered Images Plots 2-4.

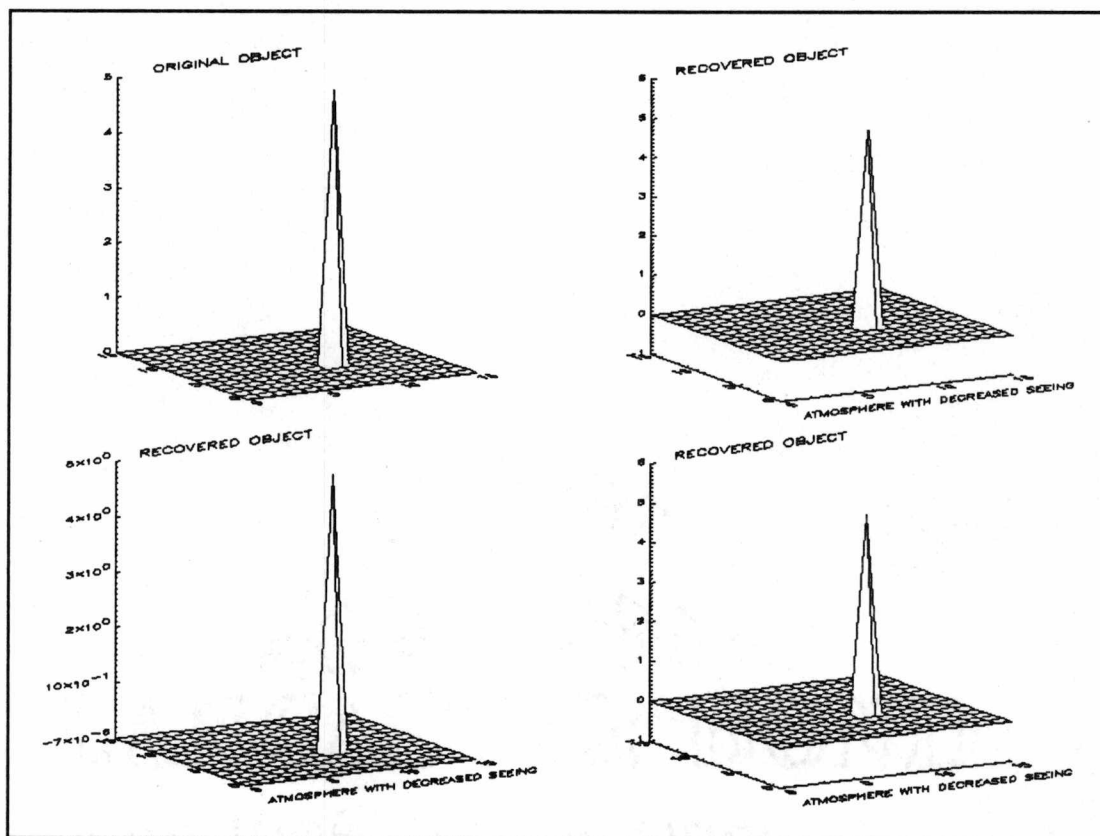


Figure 6.36 - Test Three Training Object Image (9,6) and Recovery with Speckle from Less Turbulent Atmosphere. Original Object Image Plot 1 - Recovered Images Plot 2-4.

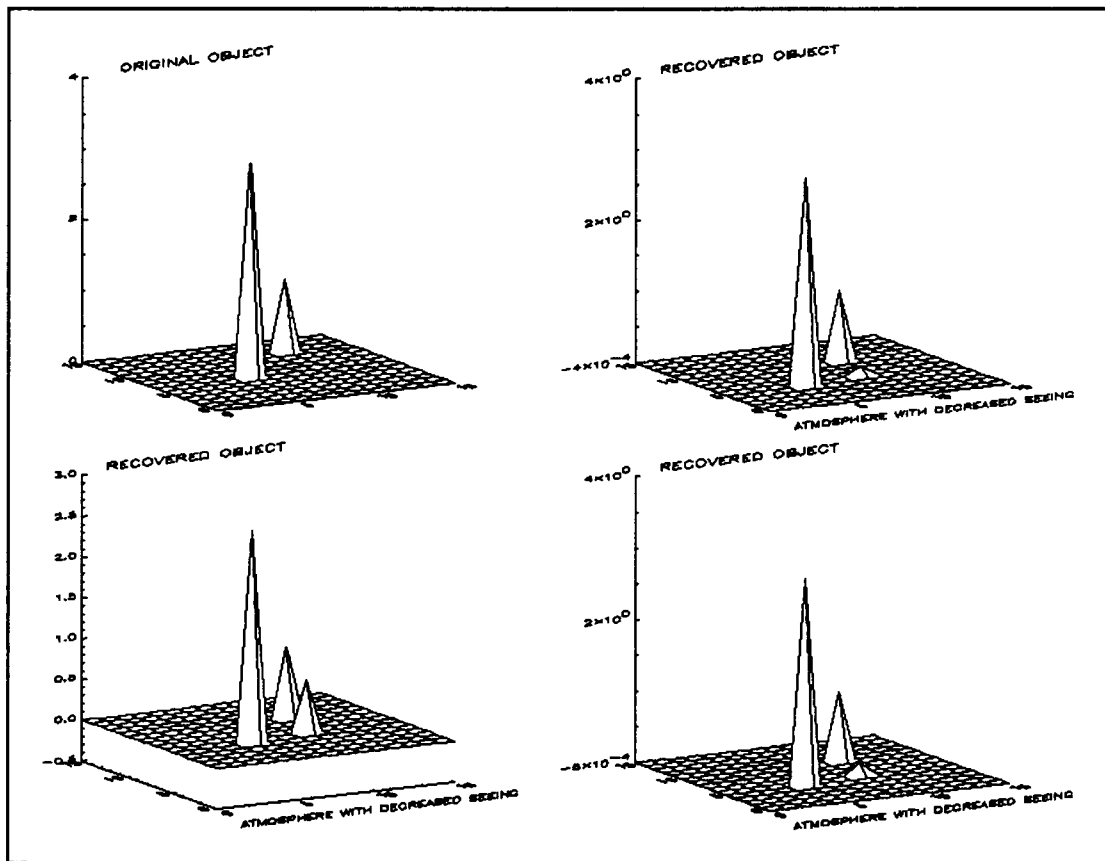


Figure 6.37 - Test Three Test 1 Object Image [Binary (6,7), (11,12)] and Recovery with Speckle from Less Turbulent Atmosphere. Original Object Image Plot 1 - REcovered Images Plot 2-4.

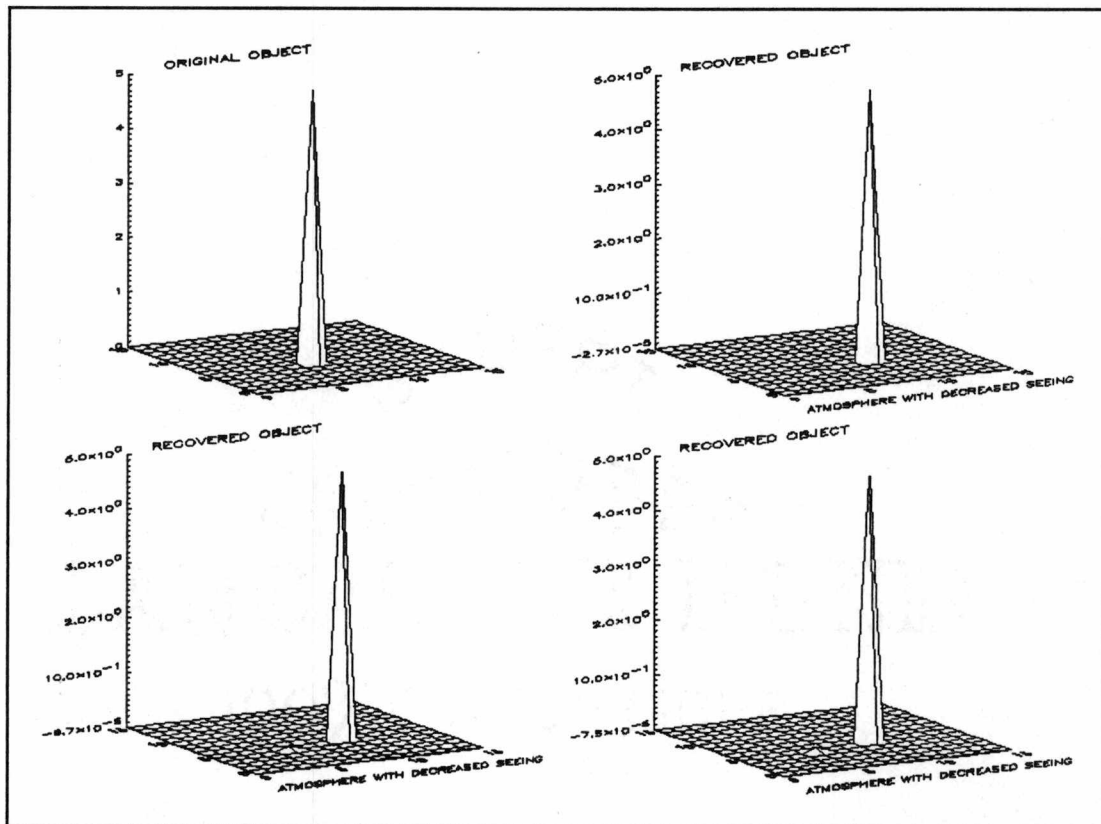


Figure 6.38 - Test Three Test 2 Object Image (7,6) and Recovery with Speckle from Less Turbulent Atmosphere. Original Object Image Plot 1 - Recovered Images Plot 2-4.

TEST FOUR SET UP AND RESULTS

In this last recovery effort, a single (5,5), binary [(6,8),(10,4)], and triple [(6,5),(7,8),(9,6)] star configurations were used to generate the 50 speckle patterns per object. (See Figure 6.39). Additionally, two test stars were utilized, a single (10,9) and a triple [(5,9),(8,8),(7,11)].

The control set RMS convergence indicated a fairly good convergence capability. (See Figure 6.40).

The recovery of objects from previously unseen speckle (speckle created by the training objects but not used as training data) was very good for the binary (See Figure 6.41) and triple star objects (See Figure 6.42) but not as adequate for the single star. (See Figure 6.43)

Once the net was trained, the two test images, Test 1 and Test 2, (See Figure 6.39) not previously trained by the network were used to produce three speckle images each. (See Figures 6.44 and 6.45).

These images were propagated through the net. The single test star speckle (Figure 6.44) recovered the single training star at position (5,5). The speckle produced by the triple star test object recovered the training triple star (Figure 6.42). This is depicted in Figure 6.45.

This suggest that the speckle is both rotationally and translationally invariant. This is in agreement with the experimental results of Knox and Thompson.

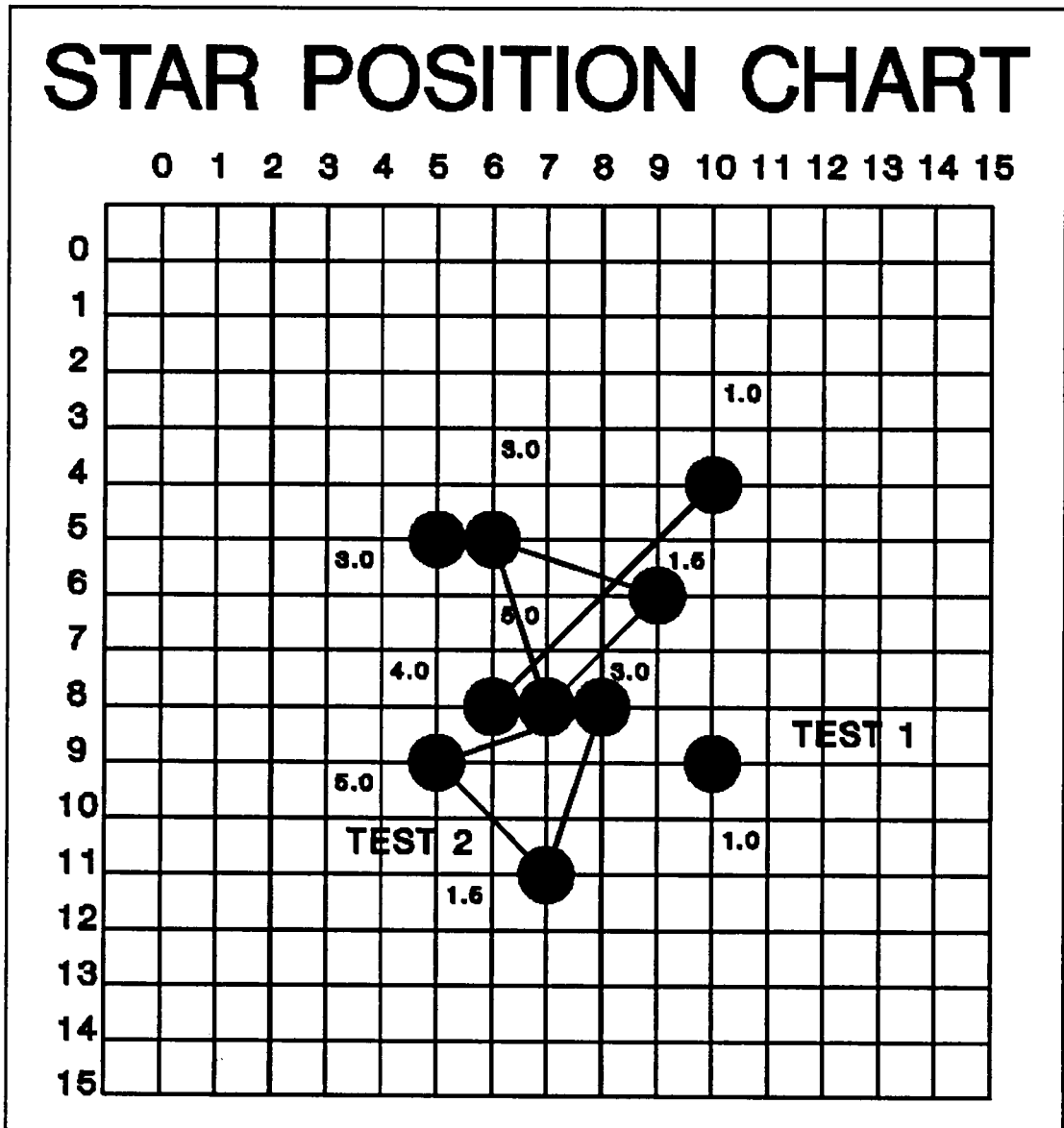


Figure 6.39 - Test Four Single, Binary and Triple Star Simulation.

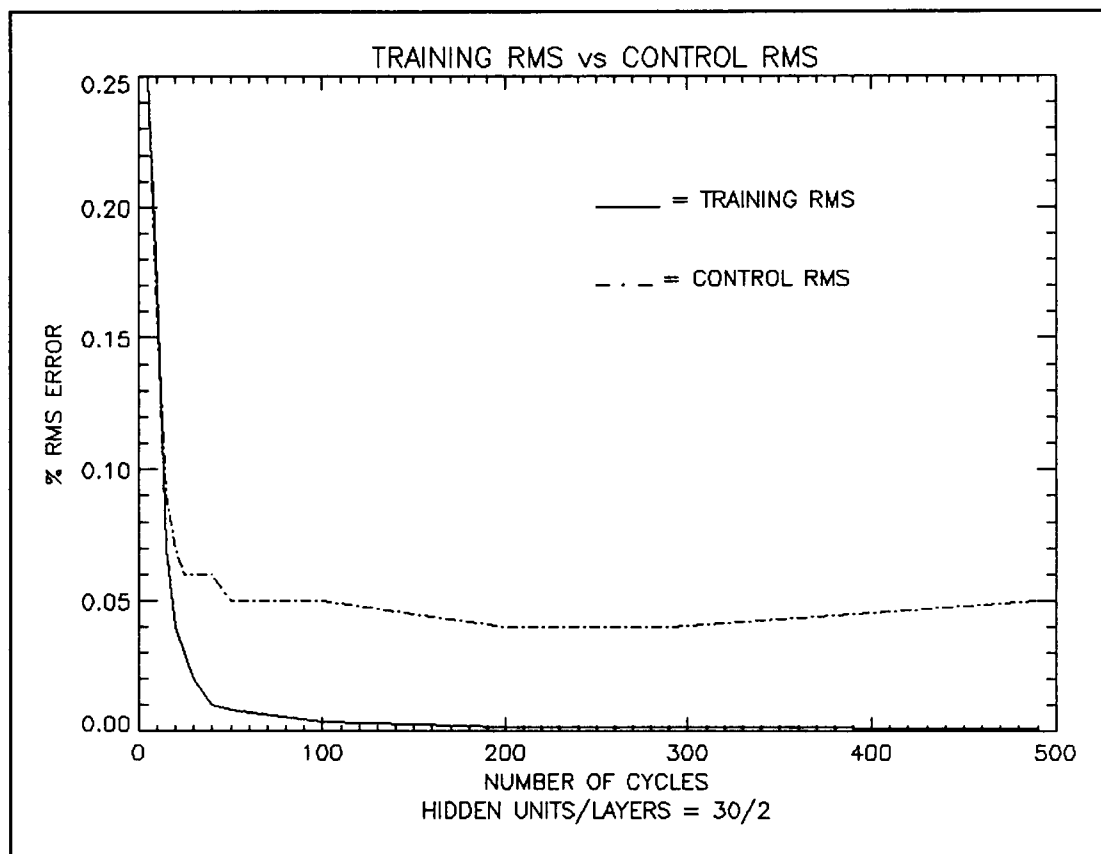


Figure 6.40 - Test Four Double Hidden Layer Training RMS versus Control RMS.

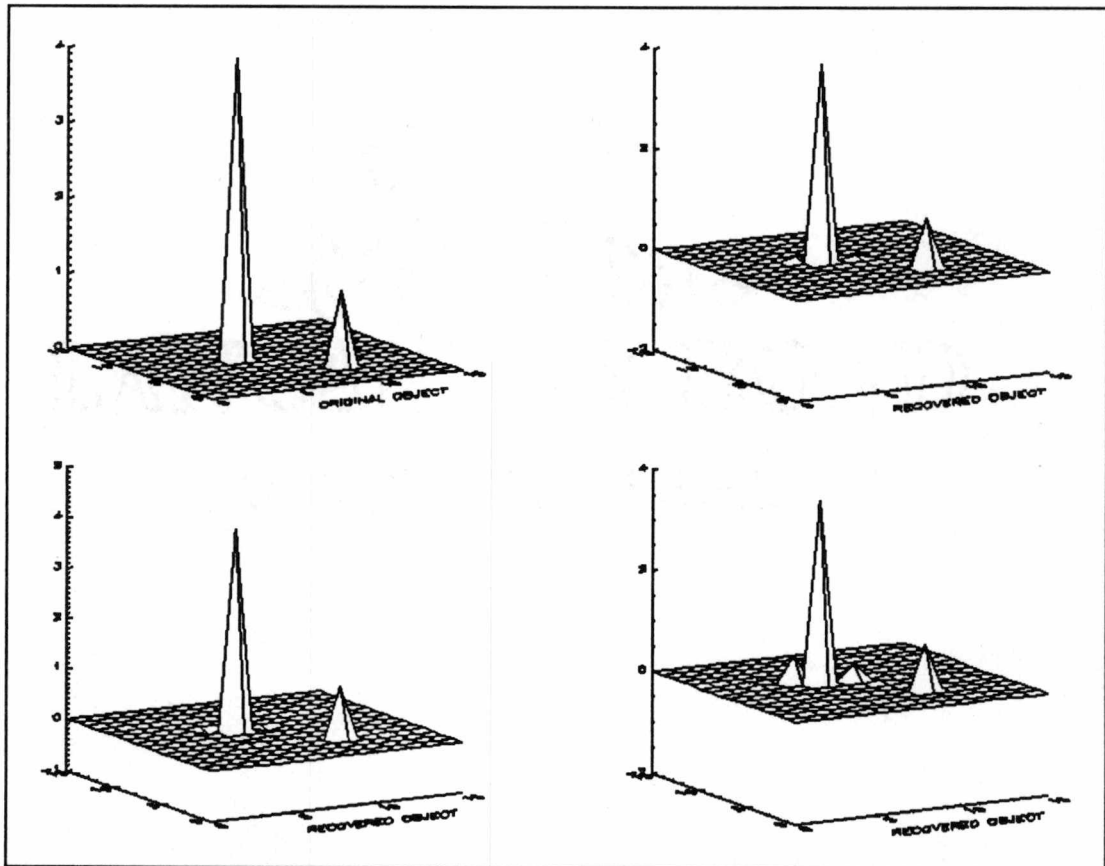


Figure 6.41 - Test Four Training Object Image [Binary (6,8), (10,4)] and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

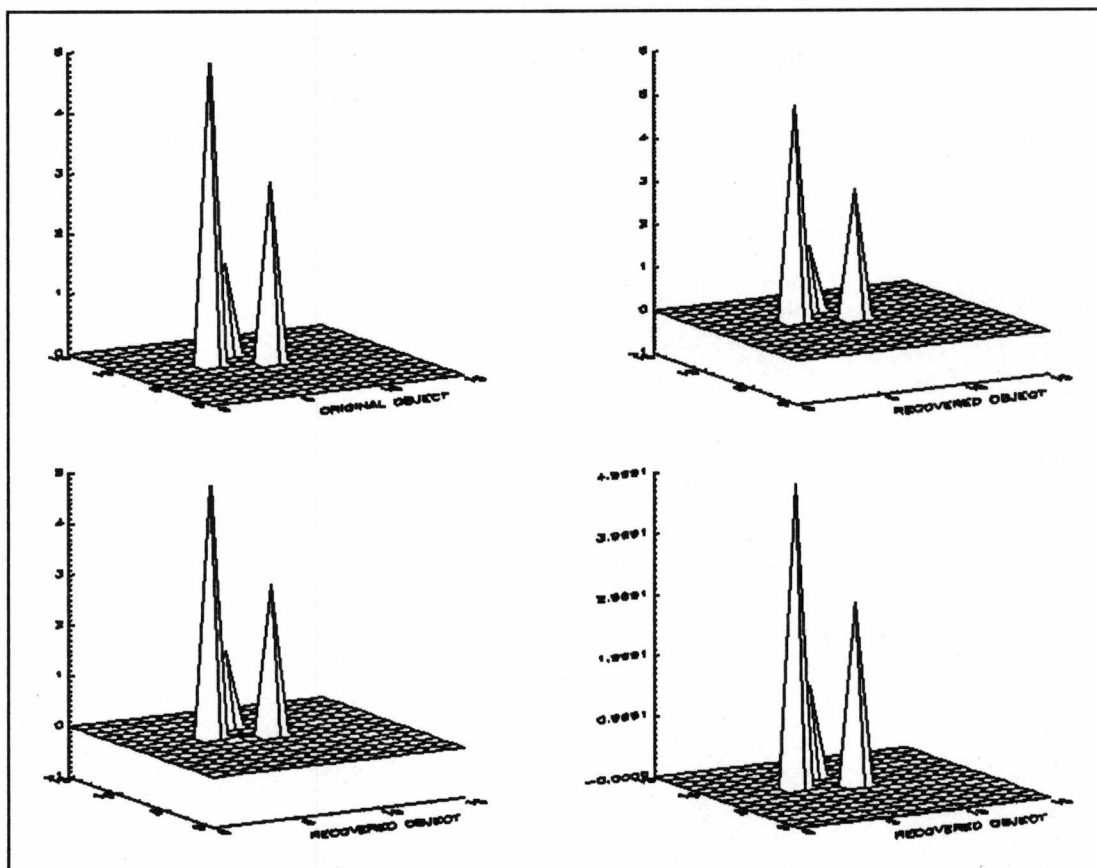


Figure 6.42 - Test Four Training Object Image [Triple (6,5), (7,8), (9.6)] and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

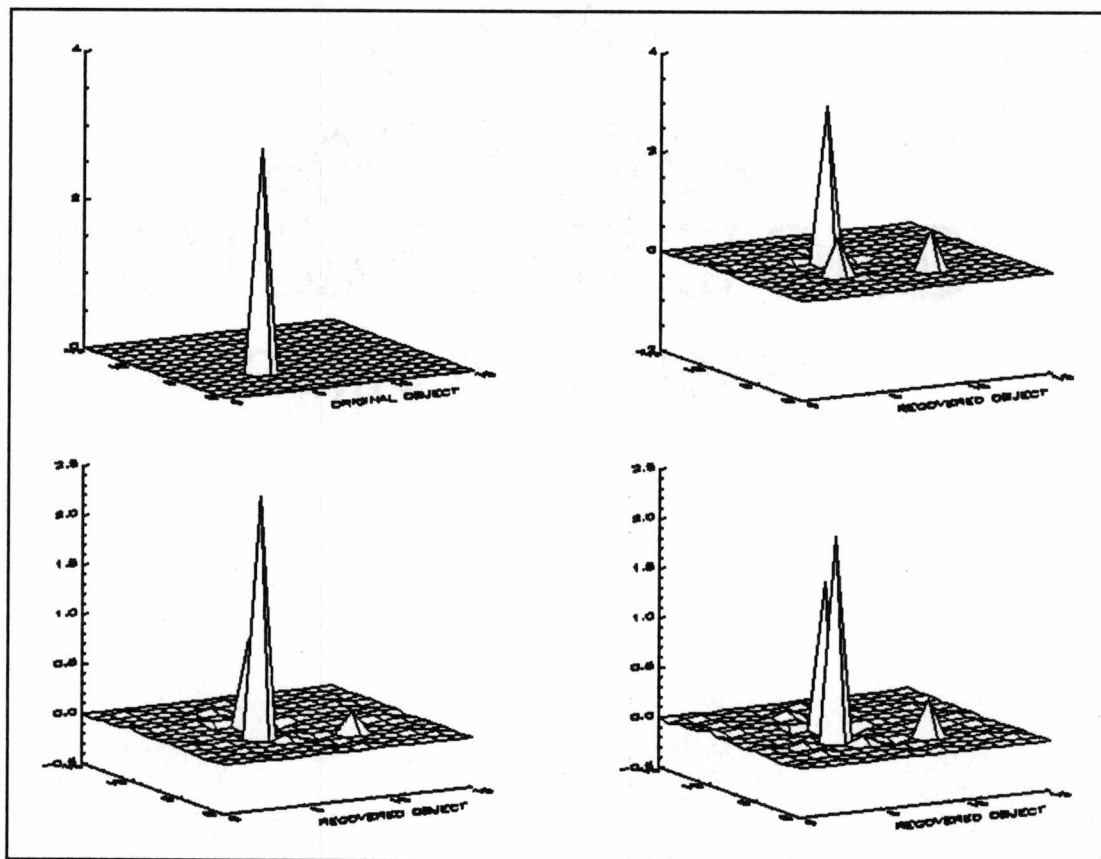


Figure 6.43 - Test Four Training Object Image (5,5) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

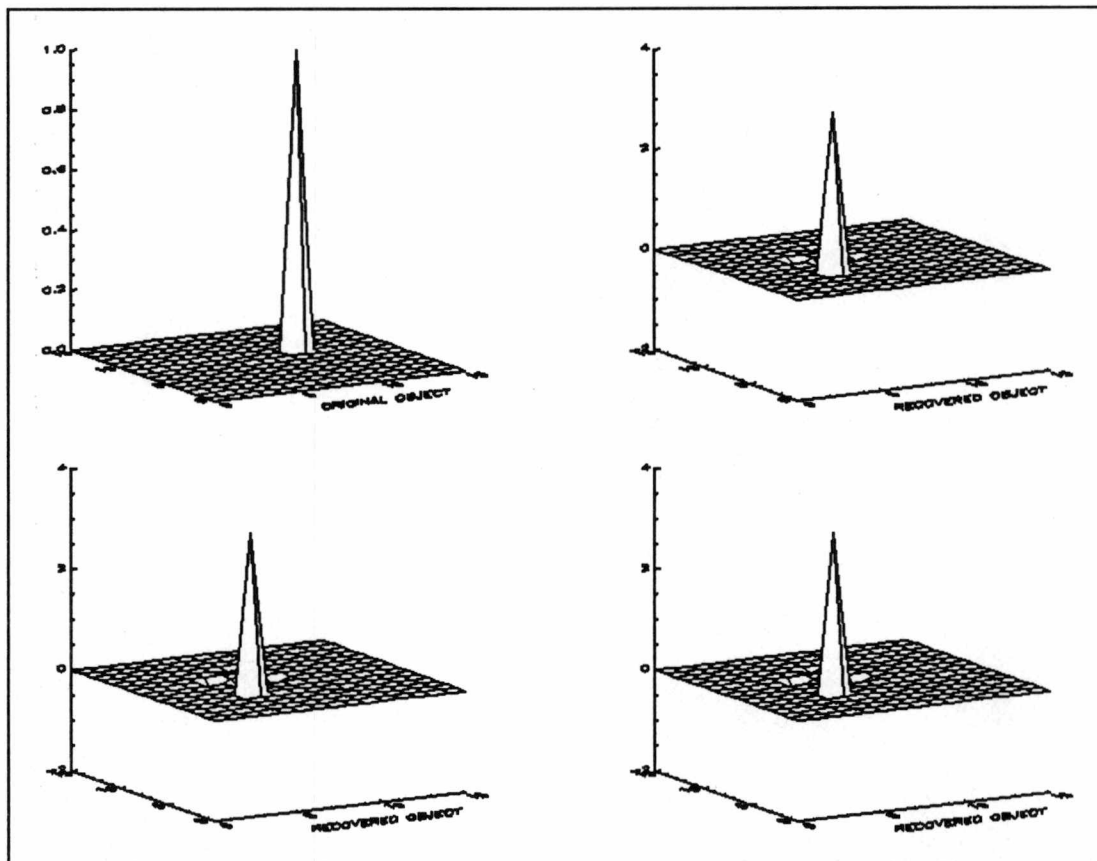


Figure 6.44 - Test Four Test 1 Object Image (10,9) and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

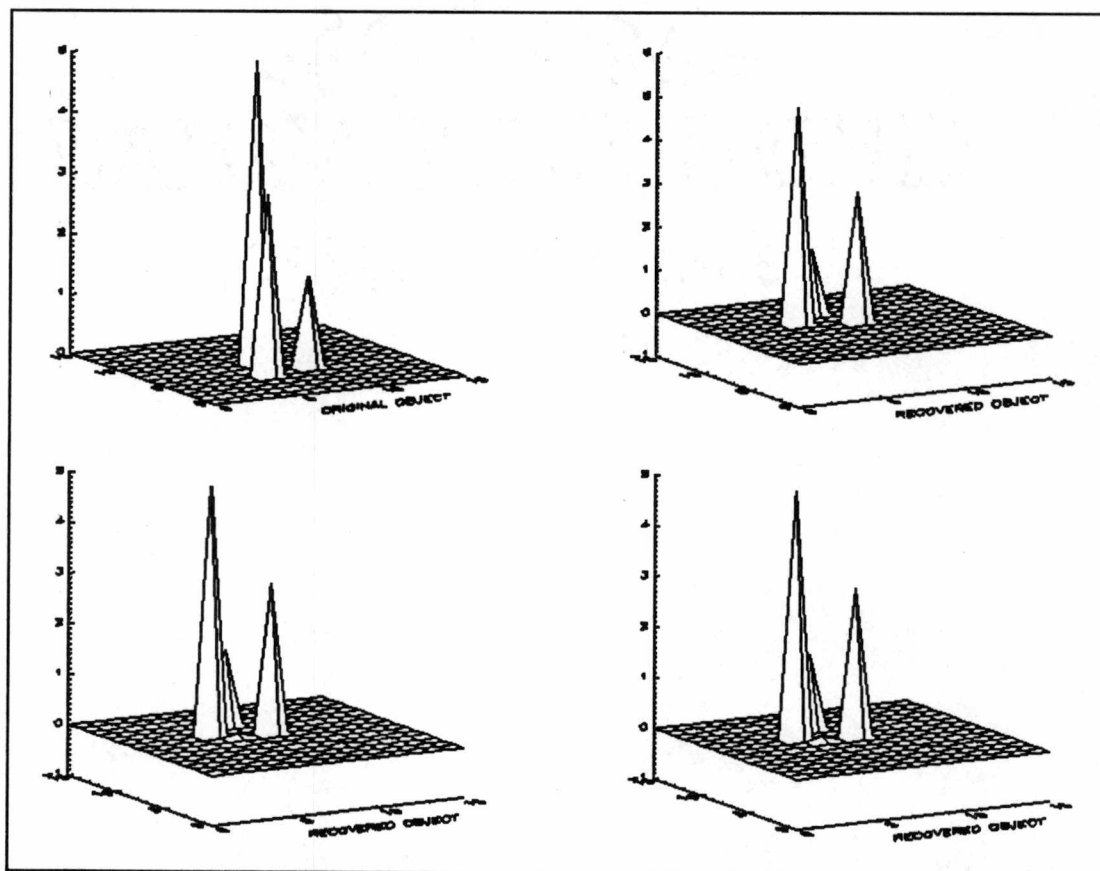


Figure 6.45 - Test Four Test 2 Object Image [Triple (8,8) (7.11), (5,9)] and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

Finally, the training images were again used to create speckle though this time through an atmosphere that had a marked increase in the seeing. The network was not able to recover any of these images adequately. (See Figures 6.46, 6.47, and 6.48). The altered atmosphere is notably more contorted (See Figure 6.49). This increased distortion seems to destroy most speckle to object relationships the trained network has established.

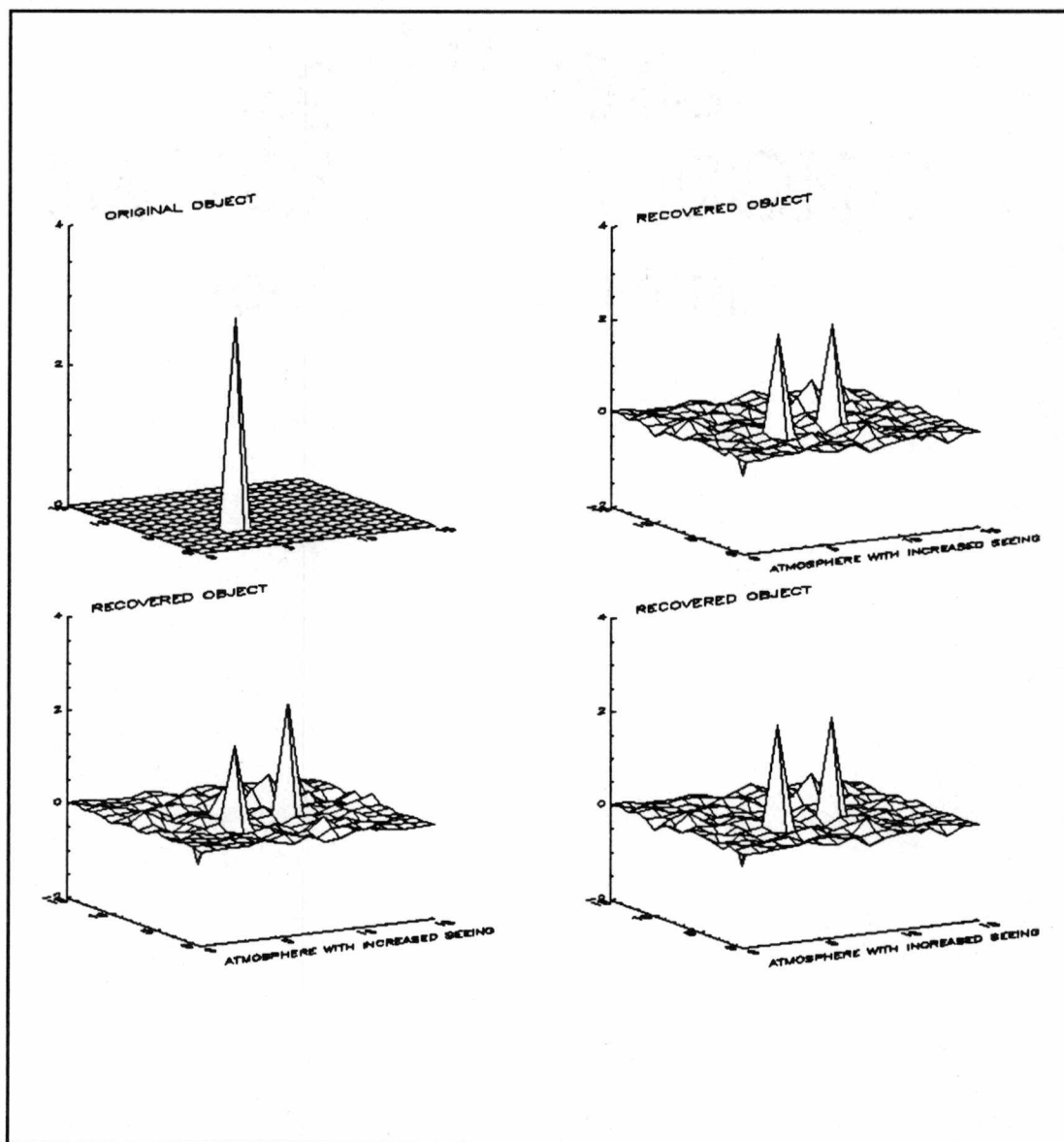


Figure 6.46 - Test Four Training Object Image (5,5) and Recovery with Speckle from a More Turbulent Atmosphere. Original Object Image Plot 1 - Recovered Images Plot 2-4.

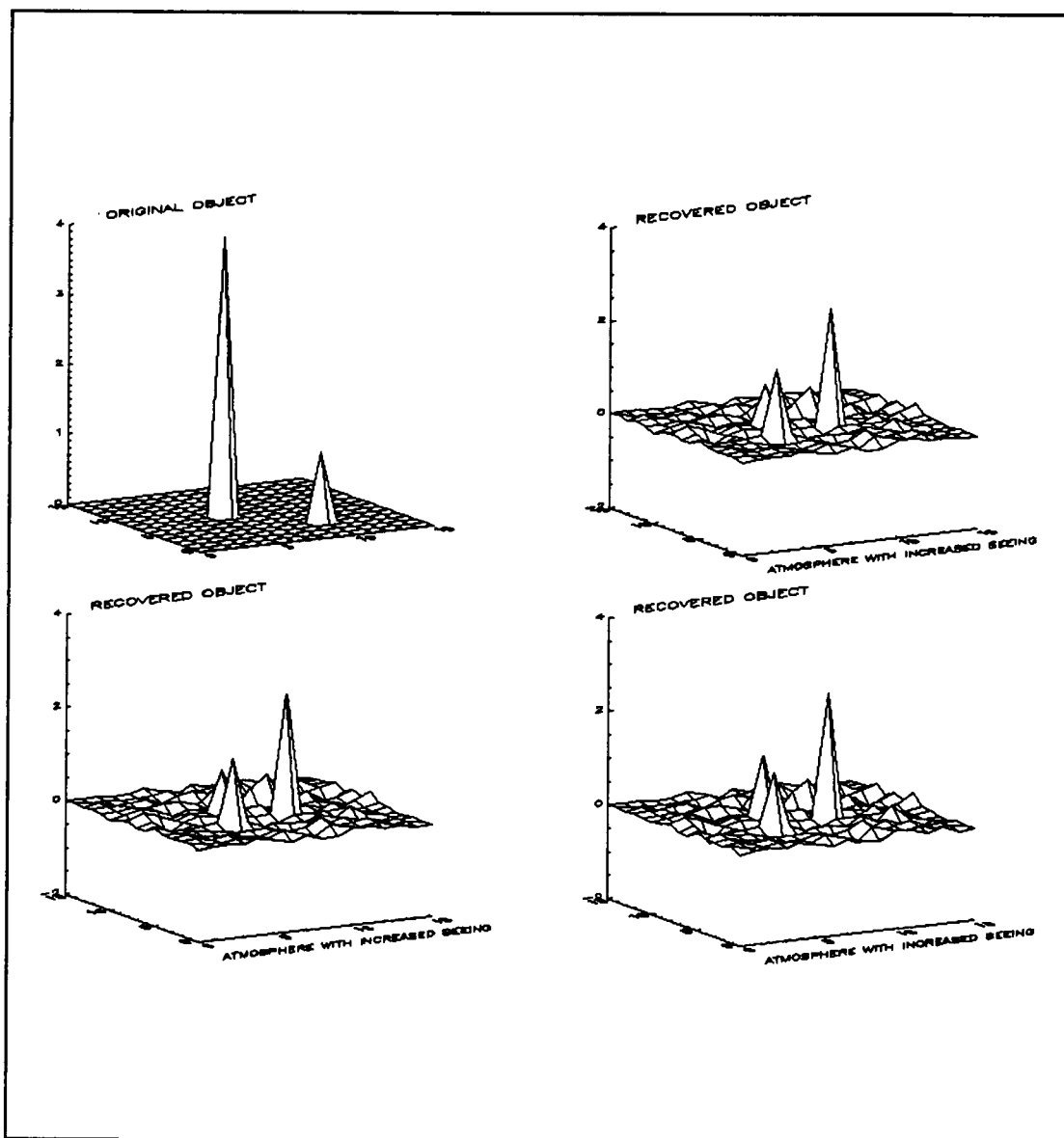


Figure 6.47 - Test Four Training Object Image [Binary (6,8), (10,4)] and Recovery with a More Turbulent Atmosphere. Original Obejct Image Plot 1 - Recovered Images Plot 2-4.

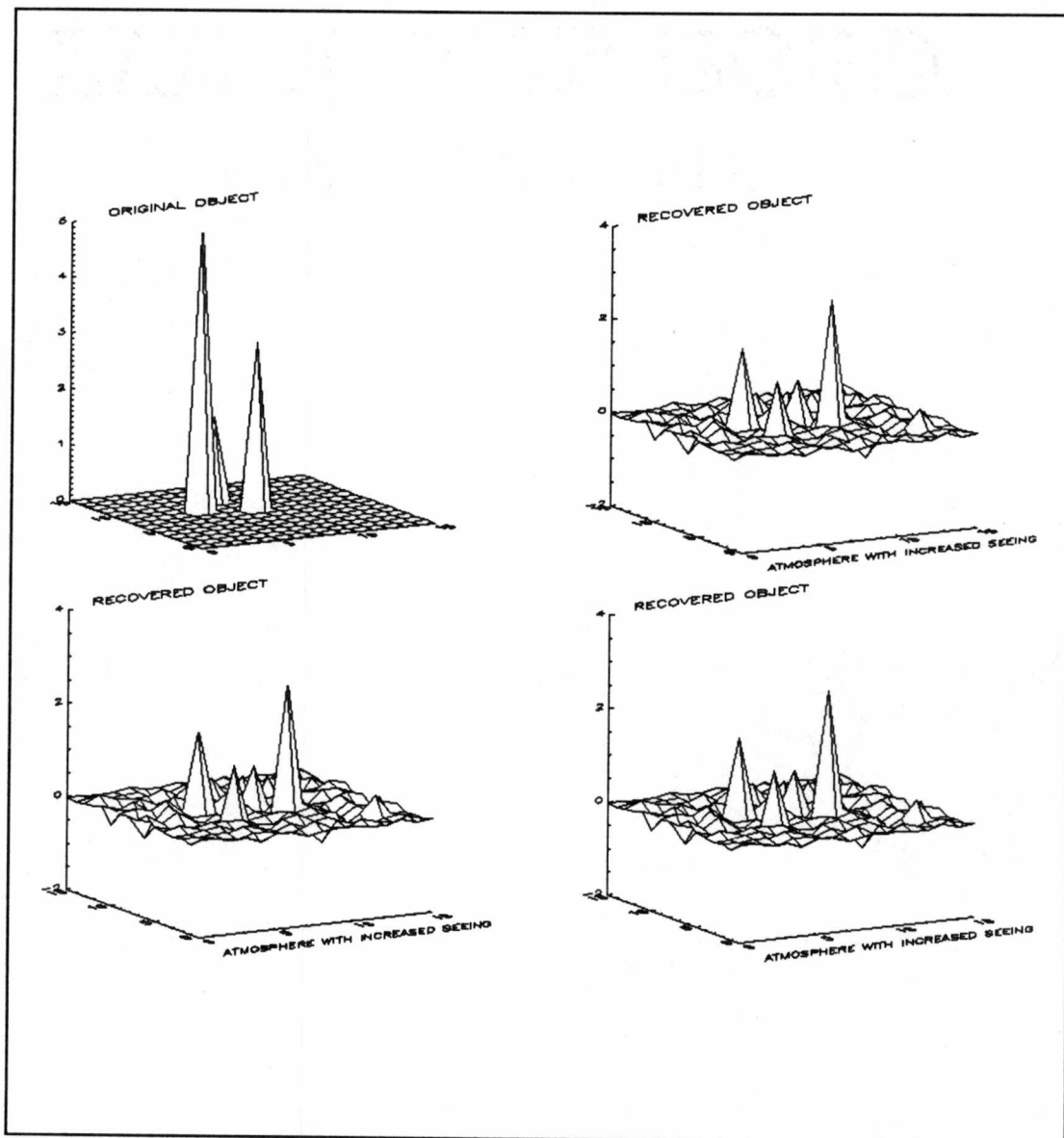


Figure 6.48 - Test Four Training Object Image [Triple (6,5), (7,8), (9,6)] and Recovery. Original Object Image Plot 1 - Recovered Images Plot 2-4.

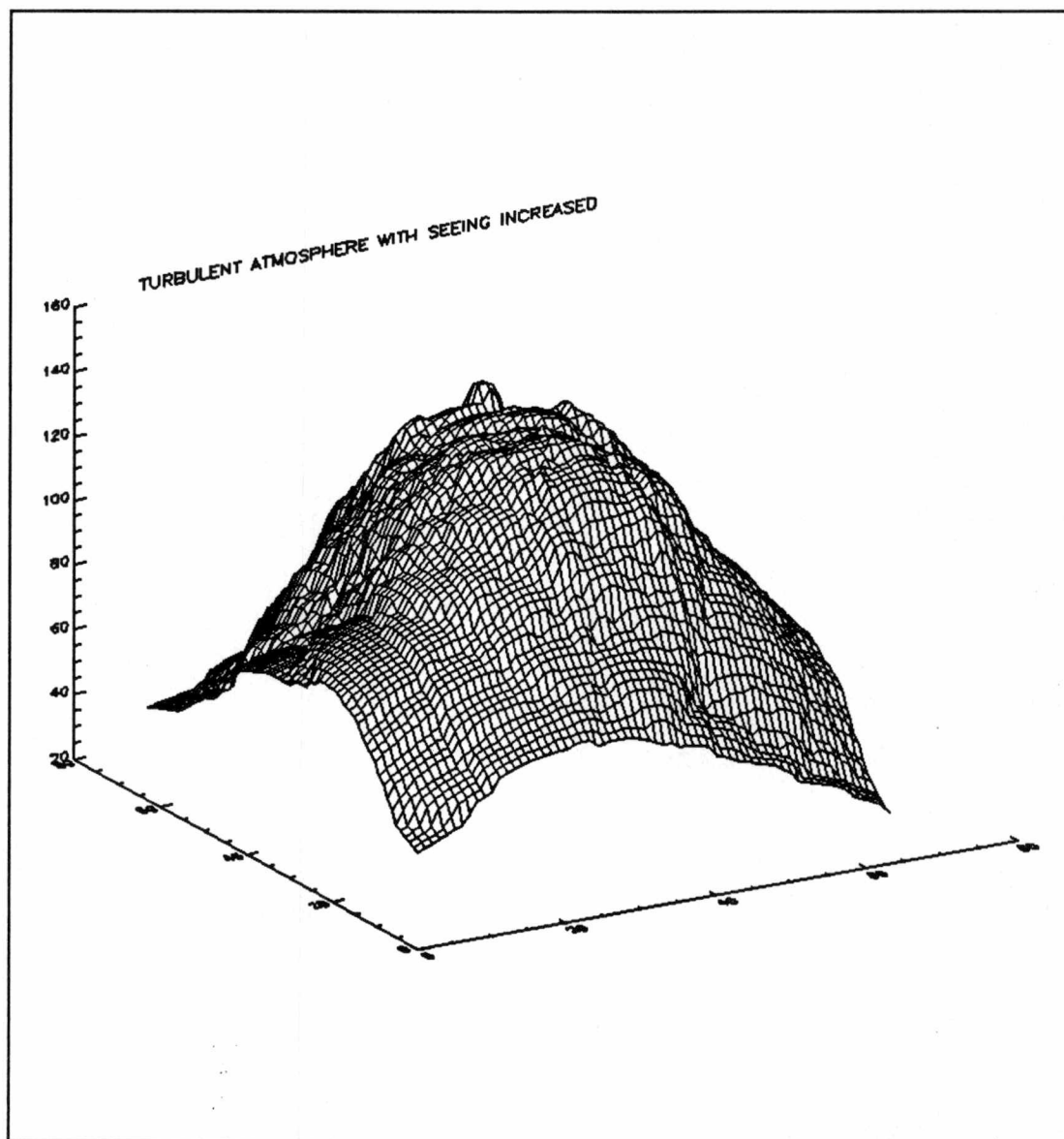


Figure 6.49 - Surface Plot of Increased Turbulent Atmosphere Array.

CONCLUSIONS

The results have been mixed. The network was able to recover stars which varied in intensity. This recovery still took place even though the atmosphere was altered to a different seeing condition (less seeing); however, it could not recover the objects when the atmosphere seeing was increased for test objects as depicted in Figures 5.46, 5.47, and 5.48 of test 4. This indicates training should be done with the atmosphere at maximum turbulence or seeing.

The work of Labeyrie (1970, 1972) indicates that the modulus of the FFT of the speckle image should contain separation information with regard to binaries. I was unable to achieve recovery of two binary objects differing in separation but not intensity in Test One. An effort was made to use the auto-correlation in place of the phase, however, the net would not converge. This is curious behavior since the auto-correlation of the speckle pattern contains information on the separation distances of binaries and triples.

Due to the position invariant characteristics of the speckle images, the orientation of the object appeared to be unimportant. The test images of Test 4 indicated that intensity variations alone would reproduce the trained pattern. The phase information portion of the training inputs produced better generalization and performance than when trained without them, but did not add the

positional/separation information hoped for when it was decided to include them as part of the training input data. It can be generally concluded that this present network configuration was capable of intensity detection differentiation but not separation/positional difference detection.

Future Work

The information on separation of binaries or triples is in the transform of the speckle image. Two approaches should be pursued:

1. Rethink training set data preparation.
Possible use of the point to point phase difference of each speckle image FFT suggested by Knox (1974), in combination with the auto-correlation might prove interesting.
2. Look at other neural network computational models as possible solutions.
3. Increase Training Set size to 200 Speckle per Object Image.

Finding the right form of the speckle image input to the net is ninety percent of the effort. The information is there and it is merely a matter of finding the correct extraction model.

Current neural network software is almost exclusively back-propagation. Extensive research into other network types (i.e. ART, Counter-propagation, BAM, HOPFIELDS, etc.) has

taken place but have seen only limited use. Possibly one of these learning algorithm's would provide better performance than back-propagation.

REFERENCES

REFERENCES CITED

1. Robert Stachnik, "Infinite Vistas", Chapter 3, J. Cornell, Ed., Charles Scribner's Sons, New York, 1985, 41.
2. Ref. 1, p. 42.
3. Ref. 1. p. 44.
4. J. W. Goodman, "Laser Speckle and Related Phenomena", Chapter 2, J.C. Dainty, Ed., Springer-Verlag, New York, 1975.
5. H.J. Tiziani, "Speckle Meteorology", Chapter 3, R.K. Erf, Ed., Academic Press, New York, 1978, 11.
6. Yau Y. Hung, "Speckle Meteorology", Chapter 4, R.K. Erf, Ed., Academic Press, New York, 1978, 51.
7. Robert K. Erf, "Speckle Meteorology", Chapter 9, R.K. Erf, Ed., Academic Press, New York, 1978, 257.
8. J.C. Dainty, "Laser Speckle and Related Phenomena", Chapter 7, J.C. Dainty, Ed., Springer-Verlag, New York, 1975, 259.
9. P. Hariharan, "Optical Interferometry", Appendix A1, Academic Press, Orlando, FL, 1985, 264.
10. Antoine Labeyrie, **Astronomy & Astrophysics**. 6, 85-87 (1970).
11. Y.L. Liu, **Optics Communications**. 8, No. 4, 372-377 (1973).
12. Keith T. Knox, **J. Optical Society of America**. 66, No. 11, 1236-1239 (1976).
13. Ref. 10, p. 87.
14. Ref. 8, p. 258.
15. Keith T. Knox, and Brian J. Thompson, **Astrophysical J.** 193:L45, (1974).
16. Ref. 8, p. 272.

17. Ref. 15, p. L46.
18. Ref. 11, p. 373-377.
19. Adolph Lohman, Gerd Weigelt & Bernhard Wirnitzer,
Applied Optics. 22, 4028-4037 (1983).
20. Graham Collins, **Physics Today.** 17, (Feb 1983).
21. Judith Dayhoff, "Neural Network Architectures", Chapter
7,8, VanNosstrand Reinhold, New York, NY (1990),
115, 136.
22. Patrick Simpson, "Artificial Neural Systems", Chapter 3,
Pergamon Pres, Elmsford, NY, 1990, 14.
23. John Hertz, Anders Krough and Richard Palmer,
"Introduction to the Theory of Neural
Computation", Chapter 6, Addison-Wesley, Redwood
City, CA (1991), 117.
24. Ref. 23, p. 117.
25. NETS Reference Manual accompanying NETS software.
26. Ref. 10, p. 85.
27. Gordon Chin, private communication.
28. NETS Reference manual accompanying NETS software.
29. William Blass, private communication.
30. Casimir Klimasauskas, **PC AI.** 27, (March 1991).
31. Casimir Klimasauskas, **PC AI.** 34, (July 1991).
32. M. Françon, "Laser Speckle and Applications in Optics",
Chapter 1, Translated by Henri Arsenault, Academic
Press, New York, (1979), 13.

REFERENCES NOT CITED IN TEXT

- Dainty, J.C., **Progress in Optics**, XIV, No. 1
- Goodman, J.W., **J. Opt Soc Am.**, 66, No. 11, 1145-1149 (Nov 1976).
- Jones, Robert, "Holographic and Speckle Interferometry", Chapter 1, Cambridge University Press, New York, (1983).
- Labeyrie, Antoine, **Progress in Optics**, XIV, No. 2, (1977).
- Lee, S.H., "Optical Image Processing", Chapter 7, S.H.Lee, Ed., Springer-Verlag, New York, (1981).
- Lee, Myung, **J. Opt Soc Am.**, 66, No. 11, (Nov 1976).
- Nisenson, P., D.C. Ehn, and R.V. Stachnik, **Proceedings of the SPIE/SPSE Symposium, Imaging through the Atmosphere**, Vol. 75, (Mar 1976).
- Welford, W.T., **J. Opt Soc Am.**, 66, No. 11, 1172-1174, (Nov 1976).

NEURAL NETWORK RELATED TOPICS

- Freeman, James A., "Neural Networks", Chapter 3, Addison-Wesley, New York, (1991).
- Hecht-Nielsen, Robert, "Neurocomputing", Chapters 3,5, Addison-Wesley, New York, (1990).
- Khanna, Tarun, "Foundations of Neural Networks", Chapter 4, Addison-Wesley, Reading, MA, (1990).
- Maren, Alianna, "Handbook of Neural Computing Applications", Chapter 15, 18, Academic Press, San Diego, CA, (1990).
- Pao, Yoh-Han, "Adaptive Pattern Recognition and Neural Networks", Chapter 10, Addison-Wesley, Reading, MA (1989).
- Reinhardt, J., "Neural Networks", Chapter 1, Springer-Verlag, New York, (1990).
- Wasserman, Philip D. "Neural Computing", Chapter 3, VanNostrand Reinhold, New York, (1989).

APPENDICES

APPENDIX 1

(Atmosphere, Speckle and Write Speckle Routines)

The following routines are all written in Interactive Data Language (IDL). The following are IDL specific functions which will

be encountered in the programs:

1. ABS(x)=absolute value function. If x is complex which the FFT is then

$$ABS(X) = \sqrt{\Re^2 + \text{Imaginary}^2}$$

2. FFT(f(x), -1)=Forward transform of the function f(x) or an array. A plus one is denotes the inverse FFT

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp[-j2\pi ux/N]$$

ATMOSPHERE SIMULATION ROUTINE

```
PRO make_atmos,nspec,radius
atmos=fltarr(64,64)
common rand,iseed
xcell=fltarr(100)
ycell=xcell
rcell=xcell
pcell=xcell
if(nspec gt 100) then nspec=100
for i=0,(nspec-1) do begin
xcell(i)=randomu(iseed)*63.0
ycell(i)=randomu(iseed)*63.0
rcell(i)=randomu(iseed)*radius
pcell(i)=randomu(iseed)*2.*3.14159265359
endfor
for ix=0,63 do begin
for iy=0,63 do begin
for i=0,(nspec-1) do begin
r1=(ix-xcell(i))^2 + (iy-ycell(i))^2
r2 = rcell(i)^2
if( r1 le r2) then begin
```

```

phase=(sqrt(r2-r1)/rcell(i) + .33)*pcell(i)
atmos(ix,iy)=atmos(ix,iy) + phase
endif
endfor
endfor
endfor
save,filename='atmos.dat',atmos
return
end

```

SPECKLE GENERATION ROUTINE

* note that atmos is the atmosphere 64x64 array passed to the routine
 * image refers to the definition of the original object image that is creating the speckle. It is also defined as a 64x64 array

```

function speckle,atmos,image
common rand,iseed
catmos=complexarr(64,64)
cimage=catmos
catmos=complex(cos(atmos),sin(atmos))
cimage=fft(image,1)
ir1=long(64*randomu(iseed))
ir2=long(64*randomu(iseed))
cimage=cimage*shift(catmos,ir1,ir2)
return,abs(fft(cimage(24:39,24:39),-1))
end

```

WRITE SPECKLE ROUTINE

* this routine combines the atmosphere array and speckle function to write the 16x16 speckle images to a file.

```

pro write_speckle,nimage,atmos,image
openw,1,'speckle_images.dat'
for i=0,nimage-1 do begin
    simage=speckle(atmos,image)
    print,"Processing image ",i+1
    printf,1,simage
endfor
close,1
return
end

```

APPENDIX 2

(Training Set Input/Output Routine)

This routine does the following:

1. Generates Speckle images
2. Takes the FFT of speckle and extracts the modulus and phase
3. Scales both the Input and Output
4. Writes the training set file for use with NETS software.

This particular example was used to write the training set for TEST 4.

```
PRO CREATE,nimage,dim1sp,dim2sp,dim1ob,dim2ob,nmaxsp,
nminsp,nmaxob,nminob
openw,10,'scalediot4.iop'
restore,'atmos.dat'
speck=fltarr(16,32)
;
openw,30,'fftspeckle_images.dat'
openw,40,'fftobject_images.dat'
;
;
image1=fltarr(64,64)
image1(29,29)=3.0
openw,1,'fftspeckle_images1.dat'
for i=0,nimage-1 do begin
  simage=specklet(atmos,image1)
  print,"Processing image1 ",i+1
    spimage=FFT(simage,-1)
    img=imaginary(spimage)
    real=float(spimage)
    img=shift(img,7,7)
    real=shift(real,7,7)
    phase=atan(img,real)
    modulus=(abs(spimage))
    modu=shift(modulus,7,7)
    printf,1,format='(f15.8)',modu
    printf,30,format='(f15.8)',modu
    printf,1,format='(f15.8)',phase
    printf,30,format='(f15.8)',phase
endfor
close,1
;object for image 1
OBJECT=FLTARR(16,16)
OBJECT(5,5)=3.0
```

```

OBJ1=FFT(object,-1)
real1=float(obj1)
imag1=imaginary(obj1)
printf,40,format='(f15.8)',real1
printf,40,format='(f15.8)',imag1
;
image1=fltarr(64,64)
image1(34,28)=1.0
image1(30,32)=4.0
openw,1,'fftspeckle_images2.dat'
for i=0,nimage-1 do begin
  simage=specklet(atmos,image1)
  print,"Processing image2 ",i+1
    spimage=FFT(simage,-1)
    img=imaginary(spimage)
    real=float(spimage)
    img=shift(img,7,7)
    real=shift(real,7,7)
    phase=atan(img,real)
    modulus=(abs(spimage))
    modu=shift(modulus,7,7)
    printf,1,format='(f15.8)',modu
    printf,30,format='(f15.8)',modu
    printf,1,format='(f15.8)',phase
    printf,30,format='(f15.8)',phase
endfor
close,1
;object for image 2
OBJECT=FLTARR(16,16)
OBJECT(10,4)=1.0
OBJECT(6,8)=4.0
OBJ1=FFT(object,-1)
real1=float(obj1)
imag1=imaginary(obj1)
printf,40,format='(f15.8)',real1
printf,40,format='(f15.8)',imag1
;
image1=fltarr(64,64)
image1(29,33)=5.0
image1(32,32)=3.0
image1(31,35)=1.5
openw,1,'fftspeckle_images3.dat'
for i=0,nimage-1 do begin
  simage=specklet(atmos,image1)
  print,"Processing image3 ",i+1
    spimage=FFT(simage,-1)
    img=imaginary(spimage)
    real=float(spimage)
    img=shift(img,7,7)
    real=shift(real,7,7)
    phase=atan(img,real)
    modulus=(abs(spimage))

```

```

        modu=shift(modulus,7,7)
        printf,1,format='(f15.8)',modu
        printf,30,format='(f15.8)',modu
        printf,1,format='(f15.8)',phase
        printf,30,format='(f15.8)',phase
    endfor
close,1
;object for image 1
OBJECT=FLTARR(16,16)
OBJECT(5,9)=5.0
OBJECT(8,8)=3.0
object(7,11)=1.5
OBJ1=FFT(object,-1)
reall=float(obj1)
imag1=imaginary(obj1)
printf,40,format='(f15.8)',reall
printf,40,format='(f15.8)',imag1
;
close,40
;
image1=fltarr(64,64)
image1(34,33)=1.0
openw,1,'fftspeckle_images11.dat'
for i=0,4 do begin
    simage=specklet(atmos,image1)
    print,"Processing image11 ",i+1
        spimage=FFT(simage,-1)
        img=imaginary(spimage)
        real=float(spimage)
        img=shift(img,7,7)
        real=shift(real,7,7)
        phase=atan(img,real)
        modulus=(abs(spimage))
        modu=shift(modulus,7,7)
        printf,1,format='(f15.8)',modu
        printf,30,format='(f15.8)',modu
        printf,1,format='(f15.8)',phase
        printf,30,format='(f15.8)',phase
    endfor
close,1
image1=fltarr(64,64)
image1(30,29)=3.0
image1(33,30)=1.5
image1(31,32)=5.0
openw,1,'fftspeckle_images12.dat'
for i=0,4 do begin
    simage=specklet(atmos,image1)
    print,"Processing image12 ",i+1
        spimage=FFT(simage,-1)
        img=imaginary(spimage)
        real=float(spimage)
        img=shift(img,7,7)

```

```

        real=shift(real,7,7)
        phase=atan(img,real)
        modulus=(abs(spimage))
        modu=shift(modulus,7,7)
        printf,1,format='(f15.8)',modu
        printf,30,format='(f15.8)',modu
        printf,1,format='(f15.8)',phase
        printf,30,format='(f15.8)',phase
    endfor
    close,1
    close,30
; 3 16x32 objects and nimage 16x32 speckle
;
;*****
;
; create the max and min table for both input and output
;
a=fltarr(16,32)
b=fltarr(16,32)
objects=fltarr(16,32)
openr,1,'fftspeckle_images.dat'
openr,2,'fftobject_images.dat'
readf,1,a
readf,2,b
maxvsp=a
minvsp=a
maxvob=b
minvob=b
;
for t=0,(3*nimage)+8 do begin
    readf,1,speck
    for j=0,dim2sp-1 do begin
        for i=0,dim1sp-1 do begin
            maxvsp(i,j)=speck(i,j) > maxvsp(i,j)
            minvsp(i,j)=speck(i,j) < minvsp(i,j)
        endfor
    endfor
endfor
close,1
;
for z=0,1 do begin
    readf,2,objects
    for j=0,dim2ob-1 do begin
        for i=0,dim1ob-1 do begin
            maxvob(i,j)=objects(i,j) > maxvob(i,j)
            minvob(i,j)=objects(i,j) < minvob(i,j)
        endfor
    endfor
endfor
close,2
save,filename='maxvalsp_scl.scl',maxvsp
save,filename='minvalsp_scl.scl',minvsp

```

```

save,filename='maxvalob_scl.scl',maxvob
save,filename='minvalob_scl.scl',minvob
;
;
;*****
;
openr,33,'fftobject_images.dat'
sclspk=fltarr(16,32)
sclobj=fltarr(16,32)
objtoscl=fltarr(16,32)
;*****
;scaling of object and image 1
;
openr,1,'fftspeckle_images1.dat'
openw,2,'scaled_input1.dat'
readf,33,objtoscl
for n=0,dim2ob-1 do begin
for m=0,dim1ob-1 do begin
    If maxvob(m,n) EQ minvob(m,n) then sclobj(m,n)=.5
    If maxvob(m,n) NE minvob(m,n) then sclobj(m,n)=nminob+(nmaxob- $
nminob)*(objtoscl(m,n)-minvob(m,n)) / (maxvob(m,n)-minvob(m,n))
endfor
endfor
;
for k=0,nimage-1 do begin
    readf,1,speck
    for n=0,dim2sp-1 do begin
    for m=0,dim1sp-1 do begin
        If maxvsp(m,n) EQ minvsp(m,n) then sclspk(m,n)=.5
        If maxvsp(m,n) NE minvsp(m,n) then sclspk(m,n)=nminsp+(nmaxsp- $
nminsp)*(speck(m,n)-minvsp(m,n)) / (maxvsp(m,n)-minvsp(m,n))
    endfor
    endfor
;
printf,2,sclspk
endfor
close,1
close,2
;
;*****
; output for object 1 scaled input/output
openw,3,'test1sc.iop'
openr,5,'scaled_input1.dat'
for x=0,nimage-4 do begin
    readf,5,sclspk
    printf,10,format='(" ", "(")'
    printf,10,format='(f15.8)',sclspk
    printf,10,format='(f15.8)',sclobj
    printf,10,format='(" ",")")'
endfor
for x=0,2 do begin
    readf,5,sclspk

```

```

        printf,3,format='(" ", "(")'
        printf,3,format='(f15.8)',sclspk
        printf,3,format='(" ", ")"'
    endfor
    close,3
    close,5
;
;
;
;
;*****
;scaling of object and image 2
;
openr,1,'fftspeckle_images2.dat'
openw,2,'scaled_input2.dat'
sclspk=fltarr(16,32)
sclobj=fltarr(16,32)
readf,33,objtoscl
for n=0,dim2ob-1 do begin
    for m=0,dim1ob-1 do begin
        If maxvob(m,n) EQ minvob(m,n) then sclobj(m,n)=.5
        If maxvob(m,n) NE minvob(m,n) then sclobj(m,n)=nminob+(nmaxob- $
            nminob)*(objtoscl(m,n)-minvob(m,n)) / (maxvob(m,n)-minvob(m,n))
    endfor
endfor
;
for k=0,nimage-1 do begin
    readf,1,speck
    for n=0,dim2sp-1 do begin
        for m=0,dim1sp-1 do begin
            If maxvsp(m,n) EQ minvsp(m,n) then sclspk(m,n)=.5
            If maxvsp(m,n) NE minvsp(m,n) then sclspk(m,n)=nminsp+(nmaxsp- $
                nminsp)*(speck(m,n)-minvsp(m,n)) / (maxvsp(m,n)-minvsp(m,n))
        endfor
    endfor
;
    printf,2,sclspk
endfor
close,1
close,2
;
;*****
; output for object 2 both scaled input/output and
unscaledin/scaledout
;
openw,3,'test2sc.iop'
openr,5,'scaled_input2.dat'
for x=0,nimage-4 do begin
    readf,5,sclspk
    printf,10,format='(" ", "(")'
    printf,10,format='(f15.8)',sclspk
    printf,10,format='(f15.8)',sclobj

```

```

        printf,10,format='(" ",")')'
    endfor
    print,'printing test I/O for speck 2'
    for x=0,2 do begin
        readf,5,sclspk
        printf,3,format='(" ",")'
        printf,3,format='(f15.8)',sclspk
        printf,3,format='(" ",")')'
    endfor
    close,3
    close,5
;
;*****
;
;
;*****
;scaling of object and image 3
;
objtoscl=fltarr(16,32)
openr,1,'fftspeckle_images3.dat'
openw,2,'scaled_input3.dat'
sclspk=fltarr(16,32)
sclobj=fltarr(16,32)
readf,33,objtoscl
for n=0,dim2ob-1 do begin
    for m=0,dim1ob-1 do begin
        If maxvob(m,n) EQ minvob(m,n) then sclobj(m,n)=.5
        If maxvob(m,n) NE minvob(m,n) then sclobj(m,n)=nminob+(nmaxob- $
            nminob)*(objtoscl(m,n)-minvob(m,n)) / (maxvob(m,n)-minvob(m,n))
    endfor
endfor
;
for k=0,nimage-1 do begin
    readf,1,speck
    for n=0,dim2sp-1 do begin
        for m=0,dim1sp-1 do begin
            If maxvsp(m,n) EQ minvsp(m,n) then sclspk(m,n)=.5
            If maxvsp(m,n) NE minvsp(m,n) then sclspk(m,n)=nminsp+(nmaxsp- $
                nminsp)*(speck(m,n)-minvsp(m,n)) / (maxvsp(m,n)-minvsp(m,n))
        endfor
    endfor
;
    printf,2,sclspk
endfor
close,1
close,2
;
;*****
; output for object 3 scaled input/output
;
openw,3,'test3sc.iop'
openr,5,'scaled_input3.dat'

```

```

for x=0,nimage-4 do begin
    readf,5,sclspk
    printf,10,format='(" ","(")'
    printf,10,format='(f15.8)',sclspk
    printf,10,format='(f15.8)',sclobj
    printf,10,format='(" ","")'
endfor
print,'printing I/O & test for sp & obj 3'
for x=0,2 do begin
    readf,5,sclspk
    printf,3,format='(" ","(")'
    printf,3,format='(f15.8)',sclspk
    printf,3,format='(" ","")'
endfor
close,3
close,5
;*****
;scaling of image 11
;
openr,1,'fftspeckle_images11.dat'
openw,2,'scaled_input11.dat'
for k=0,4 do begin
    readf,1,speck
    for n=0,dim2sp-1 do begin
        for m=0,dim1sp-1 do begin
            If maxvsp(m,n) EQ minvsp(m,n) then sclspk(m,n)=.5
            If maxvsp(m,n) NE minvsp(m,n) then sclspk(m,n)=nminsp+(nmaxsp- $
                nminsp)*(speck(m,n)-minvsp(m,n)) / (maxvsp(m,n)-minvsp(m,n))
        endfor
    endfor
;
    printf,2,sclspk
endfor
close,1
close,2
;
;*****
; output for IMAGE 11
;
OPENW,3,'test11sc.iop'
openr,5,'scaled_input11.dat'
for x=0,4 do begin
    readf,5,sclspk
    printf,3,format='(" ","(")'
    printf,3,format='(f15.8)',sclspk
    printf,3,format='(" ","")'
endfor
close,3
close,5
;
;*****
;scaling of image 12

```

```

;
openr,1,'fftspeckle_images12.dat'
openw,2,'scaled_inpuT12.dat'
for k=0,4 do begin
    readf,1,speck
    for n=0,dim2sp-1 do begin
        for m=0,dim1sp-1 do begin
            If maxvsp(m,n) EQ minvsp(m,n) then sclspk(m,n)=.5
            If maxvsp(m,n) NE minvsp(m,n) then sclspk(m,n)=nminsp+(nmaxsp- $
nminsp)*(speck(m,n)-minvsp(m,n)) / (maxvsp(m,n)-minvsp(m,n))
        endfor
    endfor
endfor
;
printf,2,sclspk
endfor
close,1
close,2
;
;*****
; output for IMAGE 12
;
OPENW,3,'test12sc.iop'
openr,5,'scaled_input12.dat'
for x=0,4 do begin
    readf,5,sclspk
    printf,3,format='(" ", "(")'
    printf,3,format='(f15.8)',sclspk
    printf,3,format='(" ", ")"'
endfor
close,3
close,5
;
return
end

```

APPENDIX 3

(Descaling routine for Neural Network Output Layer)

This routine descales the output of the net to allow recovery of image.

```
pro descale,dim1,dim2,nmax,nmin,nimage
sclobj=fltarr(dim1,dim2)
openw,2,'recovered_object.dat'
restore,'maxvalob_scl.scl'
restore,'minvalob_scl.scl'
desclspk=fltarr(dim1,dim2)
for k=0,nimage-1 do begin
  readf,1,sclobj
  for n=0,dim2-1 do begin
    for m=0,dim1-1 do begin

desclspk(m,n)=minvob(m,n)+(maxvob(m,n)-minvob(m,n))*(sclobj(m,n)-
nmin) $
      / (nmax-nmin)
    endfor
  endfor
  real=fltarr(16,16)
  img=fltarr(16,16)
  real=desclspk(*,0:15)
  img=desclspk(*,16:31)
  c=complex(real,img)
  d=fft(c,+1)
  d=float(d)
  printf,2,format='(f15.8)',d
endfor
close,1
close,2
end
```

VITA

Captain Walter F. Smith, Jr. is a 1983 graduate of the University of Tennessee at Chattanooga, where he received a B.S. degree in Physics, graduating Cum Laude.

He is currently an active duty Army officer assigned to the University of Tennessee at Knoxville, for graduate studies in physics. He was selected by the Army from among his peers to attend graduate school, whereupon his graduation in May, 1992, he will become a faculty member of the United States Military Academy's Department of Physics. Captain Smith has been a commissioned officer for 11 years. He is a graduate of the U.S. Army's Special Forces Qualification Course and has commanded a Special Forces Operational Detachment.